Channel Importance Matters in Few-Shot Image Classification

Xu Luo [1] **Jing Xu** [2] **Zenglin Xu** [2 3]

Abstract

Few-Shot Learning (FSL) requires vision models to quickly adapt to brand-new classification tasks with a shift in task distribution. Understanding the difficulties posed by this *task distribution shift* is central to FSL. In this paper, we show that a simple channel-wise feature transformation may be the key to unraveling this secret from a channel perspective. When facing novel few-shot tasks in the test-time datasets, this transformation can greatly improve the generalization ability of learned image representations, while being agnostic to the choice of datasets and training algorithms. Through an in-depth analysis of this transformation, we find that the difficulty of representation transfer in FSL stems from the severe *channel bias* problem of image representations: channels may have different importance in different tasks, while convolutional neural networks are likely to be insensitive, or respond incorrectly to such a shift. This points out a core problem of the generalization ability of modern vision systems which needs further attention in the future.

1. Introduction

Deep convolutional neural networks (Krizhevsky et al., 2012; He et al., 2016) have revolutionized computer vision in the last decade, making it possible to automatically learn representations from a large number of images. The learned representations can generalize well to brand-new images. As a result, image classification performance is close to humans on most benchmarks. However, in addition to recognizing previously-seen categories, humans can quickly change their focus of image patterns in changing environments and recognize new categories given only a few observations. This fast learning capability, known as

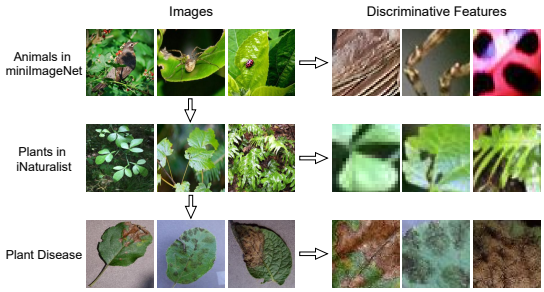

Figure 1. **Examples of task distribution shift.** Different classification tasks may focus on distinct discriminative information. Top: animals in *mini*ImageNet with different plants as background. Middle: plants as the main categories in iNaturalist. Bottom: Different types of plant diseases in the fine-grained Plant Disease dataset.

Few-Shot Learning (FSL), challenges current vision models on the ability to quickly adapt to novel classification tasks that are different from those in training. This *task distribution shift* means that categories, domains of images or granularity of categories in new tasks deviate from those in the training tasks.

Recent studies of few-shot image classification have highlighted the importance of the quality of learned image representations (Raghu et al., 2020; Doersch et al., 2020; Dhillon et al., 2020; Tian et al., 2020; Rizve et al., 2021), and also showed that representations learned by neural networks do not generalize well to novel few-shot classification tasks when there is task distribution shift (Chen et al., 2021; Doersch et al., 2020; Agarwal et al., 2021). Thus it is crucial to understand how task distribution shift affects the generalization ability of image representations in few-shot learning.

As shown in Figure 1, task distribution shift may lead to changes in discriminative image features that are critical to the classification task at hand. For example, in the task of recognizing animals, a convolutional neural network trained on *mini*ImageNet can successfully identify the discriminative information related to animals. Although the representations learned by the network may encode some plant information (from image background), plants do not appear as a main category in miniImageNet and it may be insufficient for the network to distinguish various plants in a novel few-shot task sampled from the iNaturalist dataset. Even when the network is well trained to recognize plants on iNaturalist, it is difficult to be adapted to the novel task

[1]University of Electronic Science and Technology of China [2]Harbin Institute of Technology Shenzhen [3]Pengcheng Laboratory. Correspondence to: Xu Luo <frank.luox@outlook.com>, Zenglin Xu <xuzenglin@hit.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

of identifying plant diseases due to the granularity shift, since the discriminative information now becomes the more fine-grained lesion part of leaves.

In this paper, we show that this difficulty encountered in few-shot learning leads to a *channel bias* problem in learned image representations (i.e., features). Specifically, in the layer after global pooling, different channels in the learned feature seek for different patterns (as verified in (Zhou et al., 2015; Bau et al., 2017)) during training, and the channels are weighted (in a biased way) based on their importance to the training task. However, when applied to novel few-shot classification tasks, the learned image features usually do not change much or have inappropriately changed without adapting to categories in novel tasks. This bias towards training tasks may result in imprecise attention to image features in novel tasks.

What leads to our discovery of the channel bias problem is a simple transformation function that we found in a mathematical analysis textbook. Applied to top of image representations channel-wisely only at test time on the fly, this transformation function can consistently and largely improve predictions for out-of-distribution few-shot classification tasks, being agnostic to the choice of datasets and training algorithms (e.g., 0.5-7.5% average improvement over 5-way 5-shot tasks on 19 different test-time datasets, as shown in Table 1). Through analysis, we reveal the existence of channel bias problem, and show that this transformation rectifies channel emphasis by adjusting the Mean Magnitude of Channels (MMC) of image representations over the target task. Concretely, it serves as a smoothing function that suppresses channels of large MMC and largely amplifies channels of small MMC.

To further understand the channel bias problem, we derive an oracle adjustment on the MMC of image representations in binary classification tasks. Such studies demonstrate that the channel bias problem exists in many different target tasks with various types of task distributions shift, and it becomes severe with the distribution shift expanding (as shown in Figure 6). In addition, through test-time shot analysis, we verify that the channel bias problem requires more attention in few-shot setting, while simple fine-tuning can help address this problem in many-shot setting.

2. A Channel-wise Feature Transformation

2.1. Problem Setup

In few-shot image classification, a training set $\mathcal{D}^{train}$ is used at first to train a neural network parametrized by θ, which will be evaluated on a series of few-shot classification tasks constructed from the test-time dataset $\mathcal{D}^{test}$. Importantly, there should be task distribution shift between $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$, which may include category shift, domain shift or granularity shift. Each evaluated N-way K-shot few-shot classification task τ is constructed by first sampling N classes from $\mathcal{D}^{test}$, and then sampling K and M images from each class to constitute a support set $\mathcal{S}_\tau$ and a query set $\mathcal{Q}_\tau$, respectively. The support set $\mathcal{S}_\tau = \{(x^\tau_{k,n}, y^\tau_{k,n})\}^{K,N}_{k,n=1}$ consisting of $K \times N$ images $x^\tau_{k,n}$ and corresponding labels $y^\tau_{k,n}$ from the N classes is used to construct a classifier $p_\theta(\cdot|x, \mathcal{S}_\tau)$, which is further evaluated on the query set $\mathcal{Q}_\tau = \{x^{*\tau}_{m,n}\}^{M,N}_{m,n=1}$. The evaluation metric is the average prediction accuracy on query set over all sampled few-shot classification tasks.

In order to evaluate on different types and degrees of task distribution shift, in the following experiments, we select a broad range of datasets for $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$. For $\mathcal{D}^{train}$, we choose (1) the train split of *mini*ImageNet (Vinyals et al., 2016) that contains 38400 images from 64 classes; (2) the train split of ImageNet 1K (Russakovsky et al., 2015) containing more than 1M images from 1000 classes; (3) train+val split of iNaturalist 2018 (Horn et al., 2018), a fine-grained dataset of plants and animals with a total of more than 450000 training images from 8142 classes. For $\mathcal{D}^{test}$, we choose the test split of *mini*ImageNet, and all evaluation datasets of Meta-dataset (Triantafillou et al., 2020), BSCD-FSL benchmark (Guo et al., 2020) and DomainNet (Peng et al., 2019), for a total of 19 datasets, to ensure adequate coverage of different categories, domains and task granularities.

2.2. Universal Performance Gains from a Test-time Simple Feature Transformation

Let $\boldsymbol{x} \in \mathbb{R}^D$ denote an image and $f_\theta(\cdot)$ a feature extractor learned from the training set $\mathcal{D}^{train}$. The l-th channel of the feature $\boldsymbol{z} = f_\theta(\boldsymbol{x}) \in \mathbb{R}^d$ is defined as the l-th dimension of $\boldsymbol{z}$, i.e., $\{z_i\}^d_{i=1}$ is the set of all d channels. The simple transformation function $\phi_k : [0, +\infty) \to [0, +\infty)$ that we consider is defined as

$$\phi_k(\lambda) = \begin{cases} \frac{1}{ln^k(\frac{1}{\lambda}+1)}, & \lambda > 0 \\ 0, & \lambda = 0 \end{cases} \tag{1}$$

where $k > 0$ is a hyperparameter. At test time, we simply use this function to transform each channel of image features, i.e.,

$$\phi_k(\boldsymbol{z}) = (\phi_k(z_1), ..., \phi_k(z_d)). \tag{2}$$

When applying this transformation, we transform all image features in the target classification task regardless of whether they are in the support set or query set; any subsequent operation keeps unchanged. Note that this function can only be applied to features taking non-negative values, common in most convolutional neural networks using ReLU as the activation function. We discuss one variant of the function dealing with features having negative values (e.g., networks

Table 1. **Performance gains of the simple feature transformation on various training and testing datasets with a broad range of choices of network architectures and algorithms**. The black values indicate the original accuracy, and the red values indicate the increase. Each running of evaluation contains 10000 5-way 5-shot tasks sampled using a fixed seed, and the average accuracy is reported. The three groups of test-time datasets come from MetaDataset, BSCD-FSL benchmark and DomainNet, respectively.

TrainData	*mini*-train							ImageNet			iNaturalist	
Algorithm	PN	PN	CE	MetaB	MetaOpt	CE	S2M2	PN	CE	MoCo-v2	CE	
Architecture	Conv-4	Res-12	Res-12	Res-12	Res-12	SE-Res50	WRN	Res-50	Res-50	Res-50	Res-50	Average
mini-test	66.6+1.2	73.5+2.2	75.9+1.6	74.7+2.6	74.8+0.5	76.2+0.2	82.5+1.2	82.2-1.6	89.1-0.5	93.7+2.2	69.9+2.2	78.1+1.1
CUB	52.0+2.8	57.0+3.0	59.6+2.3	60.1+2.6	60.3+1.7	59.9+2.2	68.5+2.8	65.3+2.5	78.2+0.4	70.0+6.8	94.7+0.0	66.0+2.5
Textures	50.9+2.3	57.1+4.2	63.1+2.4	61.2+3.7	60.2+1.8	63.5+0.6	69.3+2.9	61.9+2.4	71.6+0.8	82.8+0.9	63.2+2.3	64.1+2.0
Traffic Signs	52.6+2.1	64.8+2.2	65.6+1.4	67.3+1.5	67.1+4.9	62.2+2.9	69.6+3.1	64.0+2.2	67.2+3.5	68.4+8.8	60.5+4.0	64.4+3.3
Aircraft	32.1+0.9	31.3+1.6	34.7+1.9	34.7+2.3	35.6+2.4	38.2+2.0	40.5+4.7	38.4+1.7	46.6+2.5	34.5+8.8	42.1+2.5	34.0+2.9
Omniglot	61.0+10.0	77.6+7.8	86.9+3.7	81.6+7.9	78.0+9.9	89.9+2.3	85.9+7.4	76.4+2.9	88.6+5.3	74.5+15.8	83.8+9.0	80.4+7.5
VGG Flower	71.0+3.1	71.1+5.5	79.2+3.8	78.3+4.5	78.4+3.1	83.0+1.7	87.8+2.5	81.4+2.6	89.3+1.7	86.2+6.3	91.9+1.1	81.6+3.3
MSCOCO	52.0+1.2	58.2+1.1	59.0+0.7	58.0+1.6	58.4+0.1	57.1+0.5	63.5+0.1	61.3-0.5	64.3-0.4	71.4+1.4	50.4+1.9	59.4+0.7
Quick Draw	49.7+6.5	60.2+5.4	67.5+6.5	61.9+9.0	61.0+6.2	69.8+2.8	66.4+8.2	59.8+6.9	70.2+3.0	63.7+8.3	60.8+6.2	62.8+6.3
Fungi	48.5+1.5	49.0+3.7	52.2+3.3	51.5+4.0	54.6+1.9	55.2+0.5	61.6+3.8	58.5+1.3	65.1+1.1	60.2+9.2	70.0+1.8	56.9+2.9
Plant Disease	66.6+7.8	73.3+7.9	80.0+5.1	75.6+7.6	78.6+4.5	83.1+3.2	86.4+3.5	72.5+8.0	84.1+3.3	87.1+4.7	85.6+4.1	79.4+5.4
ISIC	38.5+1.6	36.8+2.9	40.4+1.0	38.8+1.7	39.5+2.3	37.7+3.9	40.5+5.5	39.5+4.0	37.8+3.6	43.2+2.8	39.0+4.3	39.2+3.1
EuroSAT	63.0+4.5	67.3+5.5	75.7+2.9	71.9+4.5	72.8+5.8	75.7+1.6	81.2+2.9	72.5+6.1	78.4+2.2	83.5+2.7	73.5+3.7	74.1+3.9
ChestX	22.9+0.2	23.0+0.5	24.1+0.3	23.5+0.5	24.5+0.4	23.6+0.2	24.2+0.9	23.2+0.3	24.2+0.8	25.4+0.9	23.9+0.1	23.9+0.5
Real	67.0+1.8	72.2+3.1	76.3+1.6	75.0+2.6	75.8+1.1	76.7+0.5	81.7+1.9	80.5+0.4	87.1-0.1	88.8+2.1	72.9+1.7	77.6+1.5
Sketch	42.6+2.9	45.3+5.0	51.1+2.6	50.2+3.4	50.6+2.0	50.9+2.4	56.8+4.1	53.1+1.5	63.2+2.5	63.9+5.8	51.9+1.4	52.7+3.1
Infograph	33.1+2.8	34.7+3.7	35.3+2.8	35.0+4.0	38.3+1.1	38.2+2.5	39.2+3.7	39.7+2.7	42.3+4.2	41.6+7.1	38.5+2.9	37.8+3.4
Painting	49.0+1.7	52.5+3.3	56.1+1.4	55.1+2.5	56.2+0.7	59.3+0.8	64.2+1.8	61.8-0.2	69.6+0.5	76.5+3.0	56.4+1.9	59.7+1.6
Clipart	47.5+3.6	49.7+4.8	55.5+3.1	54.9+4.3	56.4+2.6	60.4+2.3	63.0+4.3	60.9+1.8	72.7+1.5	67.4+7.0	58.4+2.2	58.8+3.4

with Leaky ReLU) in Appendix D. A plot of this function with various choices of k is shown in Figure 2.

Table 1 shows the performance gains brought by this transformation on 5-way 5-shot FSL tasks. We test the transformation on representations trained with different algorithms, including (1) the conventional training methods including cross-entropy (CE) and the S2M2 algorithm (Mangla et al., 2020), (2) meta-learning methods including ProtoNet (Snell et al., 2017) (PN), Meta-baseline (Chen et al., 2021) and MetaOpt (Lee et al., 2019), and (3) MoCo-v2 (He et al., 2020), a unsupervised contrastive learning method. We test these methods with various backbone networks: Conv-4 (Vinyals et al., 2016) and four variants of ResNet (He et al., 2016) including ResNet-12 (Oreshkin et al., 2018), WRN-28-10 (Zagoruyko & Komodakis, 2016), ResNet-50 and SE-ResNet50 (Hu et al., 2018). We replace Leaky ReLU with ReLU in ResNet-12 to obtain positive features (cause of performance degradation in Table 1). At test-time, we use the Nearest-Centroid Classifier (Snell et al., 2017) for CE, linear probing for S2M2 and MoCo-v2, and for meta-learning algorithms we use their own test-time classifier. Training and evaluation details can be found in Appendix B.

The result shows how this simple feature transformation substantially improves few-shot learning across various algorithms, datasets and architectural choices, with a fixed hyperparameter $k = 1.3$ (We show how performance varies with different choices of k in Appendix C). The only exception happens when the test-time task distribution is very similar to a subset of training distribution: training the supervised models on ImageNet and testing on *mini*ImageNet,

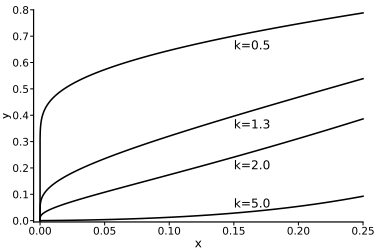

Figure 2. The simple transformation function ϕ_k with various choices of k.

MSCOCO, Real or Painting[1], or training on iNaturalist and testing on CUB. Is this transformation useful only if there exists task distribution shift between training and testing? To verify this, we train a CE model on each of ten datasets and test on 5-way 5-shot tasks sampled from each dataset. When testing on the training dataset, we evaluate on images not included in training. The results shown in Figure 3 clearly give evidence that the transformation is beneficial only to few-shot classification with task distribution shift—the performance is improved only when test-time task distribution deviates from training, and this distribution shift includes domain shift (e.g., from Sketch to QuickDraw), category shift (e.g., from Plant Disease to Fungi) and granularity shift (e.g., from iNaturalist to Plant Disease in Table 1).

[1]There are a lot of painting-style images in ImageNet. Contrastive learning (MoCo) can be seen as an infinitely fine-grained classification task, thus having a relatively large different task distribution shift from training to testing, even on the same dataset.

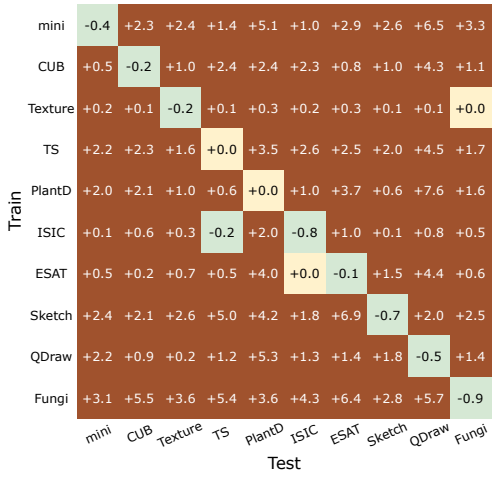

Figure 3. In-distribution (diagonal) and out-of-distribution (off-diagonal) performance gains of the simple channel-wise transformation on representations trained with CE. When the test-time dataset equals the training dataset (diagonal), the categories of images remain the same but test-time images are unseen during training (as in conventional classification).

3. The Channel Bias Problem

In this section, we analyze the simple transformation, which leads us to discover the channel bias problem of visual representations. Given the transformation function described in Eq.(1), it can be first noticed that

$$\phi_k'(\lambda) > 0, \lim_{\lambda \to 0^+} \phi_k'(\lambda) = +\infty,$$
$$\exists t > 0, \quad s.t. \quad \forall \lambda \in (0, t), \phi_k''(\lambda) < 0, \tag{3}$$

where t is a large value for most k, relative to the magnitudes of almost all channels (e.g., when $k = 1.3$, $t \approx 0.344$, while most channel values are less than 0.3). The positiveness of the derivative ensures that the relative relationship between channels will not change, while the negative second derivative narrows their gaps; the infinite derivative near zero pulls up small channels by a large margin, i.e., $\lim_{\lambda \to 0^+} \frac{\phi_k(\lambda)}{\lambda} = +\infty$. See Appendix E for the necessity of all these properties. A clear impact of these properties on features is to make channel distribution smooth: suppress channels with high magnitude, and largely amplify channels with low magnitude. This phenomenon is clearly shown in Figure 4, where we plot mean magnitudes of all 640 feature channels on *mini*ImageNet and PlantDisease, with red ones being the original distribution, blue ones being the transformed distribution. The transformed distribution becomes more uniform.

Intuitively, different channels have high responses to different features, and a larger Mean Magnitude of a Channel (MMC) implies that the model puts more emphasis on this channel, hoping that this channel is more important for the task at hand. Combining the analysis above with previous

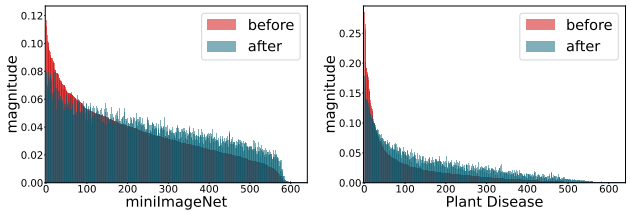

Figure 4. **Mean magnitudes of feature channels before and after applying the simple transformation.** The feature extractor is trained using PN on the training set of *mini*ImageNet. Left: test set of *mini*ImageNet. Right: The Plant Disease dataset. The change of relative magnitude is due to different variances of channels.

experiment results, we conjecture that the MMC of representations should change when testing on novel tasks with a shift in distribution. This meets our intuition that different tasks are likely to be characterized by distinct discriminative features, as shown in the examples of Fig 1.

3.1. Deriving the Oracle MMC of Any Binary Task

We now wonder how much the MMC estimated by neural networks in a task deviates from the best MMC or *channel importance* of that task. To achieve this goal, we first derive the optimal MMC for any classification task by multiplying a positive constant to each channel of features, given that we know the first-order and second-order statistics of features. For convenience, we consider the binary classification problem. Specifically, let $\mathcal{D}_1, \mathcal{D}_2$ denote probability distributions of two classes over feature space $\mathcal{Z} \subset [0, +\infty)^d$, and $\boldsymbol{z}_1 \sim \mathcal{D}_1$, $\boldsymbol{z}_2 \sim \mathcal{D}_2$ denote samples of each class. Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ denote their means and covariance matrices, respectively. We assume that the channels of features are uncorrelated with each other, i.e., there exist $\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2 \in [0, +\infty)^d$, *s.t.* $\boldsymbol{\Sigma}_1 = \mathrm{diag}(\boldsymbol{\sigma}_1)$, $\boldsymbol{\Sigma}_2 = \mathrm{diag}(\boldsymbol{\sigma}_2)$. The original MMC of the binary task is defined as $\boldsymbol{\omega}^o = (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$. We assume that the MMC after adjustment is $\boldsymbol{\omega} \in [0, +\infty)^d$. Let $\widetilde{\boldsymbol{z}}_1$, $\widetilde{\boldsymbol{z}}_2$ denote standadized version of $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ that have unit MMC, i.e., $\widetilde{z}_{1,l} = z_{1,l}/\omega_l^o, \widetilde{z}_{2,l} = z_{2,l}/\omega_l^o \Rightarrow (\widetilde{\mu}_{1,l} + \widetilde{\mu}_{2,l})/2 = 1, \forall l \in [d]$ ($[d]$ is equivalent to $\{1, 2, ..., d\}$). A simple approach to adjust MMC to $\boldsymbol{\omega}$ is to transform features to $\boldsymbol{\omega} \odot \widetilde{\boldsymbol{z}}_1$ and $\boldsymbol{\omega} \odot \widetilde{\boldsymbol{z}}_2$ respectively, where $\odot$ denotes the hadamard product. Here, we consider a metric-based classifier. Specifically, a standardized feature $\widetilde{\boldsymbol{z}}$ is classified as the first class if $||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}} - \widetilde{\boldsymbol{\mu}}_1)||_2 < ||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}} - \widetilde{\boldsymbol{\mu}}_2)||_2$ and otherwise the second class. This classifier is actually the Nearest-Centroid Classifier (NCC) (Snell et al., 2017) with accurate centroids. Assume that two classes of images are sampled equal times, then the expected misclassification rate of this classifier is

$$\mathcal{R} = \frac{1}{2}[\mathbb{P}_{\boldsymbol{z}_1 \sim \mathcal{D}_1}(||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_1 - \widetilde{\boldsymbol{\mu}}_1)||_2 > ||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_1 - \widetilde{\boldsymbol{\mu}}_2)||_2) + \mathbb{P}_{\boldsymbol{z}_2 \sim \mathcal{D}_2}(||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_2 - \widetilde{\boldsymbol{\mu}}_2)||_2 > ||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_2 - \widetilde{\boldsymbol{\mu}}_1)||_2)]. \tag{4}$$

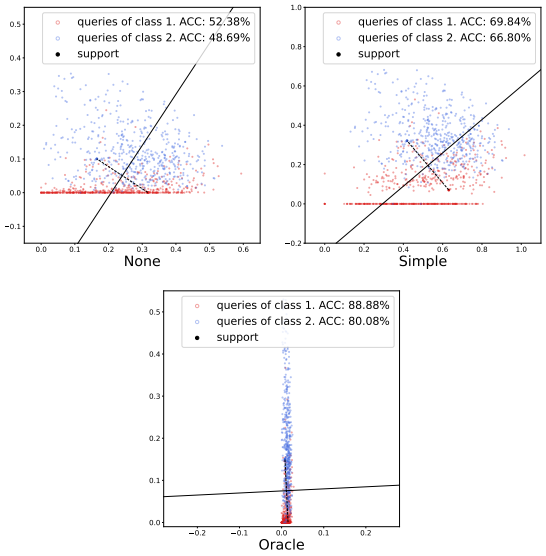

Figure 5. **Visualization of two channels of image features in two classes of Plant Disease.** The feature extractor is trained using PN on *mini*ImageNet. We visualize a one-shot task with only the two channels available for classification. The plot with "None" shows the original channels. The plots with "Simple" and "Oracle" show channels adjusted by the simple and oracle transformation. The per-class accuracy is calculated as the proportion of samples that are correctly classified by the classification boundary in each class.

The following theorem gives an upper bound of the misclassification rate and further gives the *oracle* MMC of any given task.

Proposition 3.1. *Assume that* $\mu_{1,l} \neq \mu_{2,l}$ *and* $\sigma_{1,l}+\sigma_{2,l} > 0$ *hold for any* $l \in [d]$*, then we have*

$$\mathcal{R} \leq \frac{8\sum_{l=1}^{d} \omega_l^4 (\widetilde{\sigma}_{1,l} + \widetilde{\sigma}_{2,l})^2}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2}. \tag{5}$$

To minimize this upper bound, the adjusted oracle MMC of each channel ω_l *should satisfy:*

$$\omega_l \propto \frac{|\mu_{1,l} - \mu_{2,l}|}{\sigma_{1,l} + \sigma_{2,l}}. \tag{6}$$

Proofs are given in Appendix A. We here use the word "oracle" because it is derived using the class statistics of the target dataset, which is not available in few-shot tasks. This derived MMC has an intuitive explanation: if the difference between the means of features from two classes is large but the variances of features from two classes are both small, the single channel can better distinguish the two classes and thus should be emphasized in the classification task. In fact, if we further assume $x_{1,l}$ and $x_{2,l}$ are Gaussian-distributed and consider only using the l-th channel for classification, then the misclassification error for the i-th class ($i = 1, 2$) is a strictly monotonically decreasing function of $|\mu_{1,l} - \mu_{2,l}|/\sigma_{i,l}$.

Table 2 shows the performance improvement over the simple feature transformation when adjusting the MMC to derived oracle one in each of the real few-shot binary classification tasks. For every sampled binary task in a dataset, we calculate the oracle adjustment based on Eq. (6); see Appendix F.1 for details. The oracle MMC improves performance on all datasets, and always by a large margin. Note that although the oracle MMC is derived using a metric-based classifier, it can also help a linear classifier to boost performance, which will be further discussed in Section 4. The large performance gains using the derived channel performance indicate that the MMC of features on new test-time few-shot task indeed has a large mismatch with ground-truth channel importance.

To obtain a better understanding, in Figure 5, we visualize image representations of two classes when transferred from *mini*ImageNet to Plant Disease. The two exhibited classes are apples with Apple Scab and Black Rot diseases, respectively. We visualize 2 out of 640 channels in the features, shown as the x-axis and y-axis in the figure. We select these channels by first selecting a channel that requires a large suppression of MMC (x-axis), and then a channel that requires a large increase (y-axis). As seen, the x-axis channel has a large intra-class variance (the variances are 0.13 and 0.11 in two classes on the x-axis channel, compared to 0.03 and 0.08 on the y-axis channel) and a small class mean difference (about 0.03, compared to 0.13 on the y-axis channel), so it is hard to distinguish two classes through this channel. By adjusting the mean magnitude of this channel, the simple transformation and oracle adjustment decrease the intra-class variance of the x-axis channel, and so decrease its influence on classification. Similarly, the y-axis channel can better distinguish two classes due to its relatively larger class mean difference and smaller intra-class variance, so the influence of the y-axis channel should be strengthened.

3.2. Analysis of Channel Importance

Next, we take the derived oracle MMC as an approximation of the ground-truth channel importance, and use it to observe how the simple transformation works, as well as how much the channel emphasis of neural networks deviates from the ground-truth channel importance of tasks in each test-time dataset. We define MMC of a dataset D as the average l_1-normalized MMCs over all possible binary tasks in that dataset. Specifically, suppose in one dataset D there are C classes, and let $\boldsymbol{\omega}_{ij}$ denote the MMC in the binary task discriminating the i-th and j-th class. $\overline{\boldsymbol{\omega}_{ij}} = \boldsymbol{\omega}_{ij}/||\boldsymbol{\omega}_{ij}||_1$ normalizes the MMC, such that the l-th component of the vector $\overline{\boldsymbol{\omega}_{ij}}$ represents the percentage of channel emphasis on the l-th channel. Then the

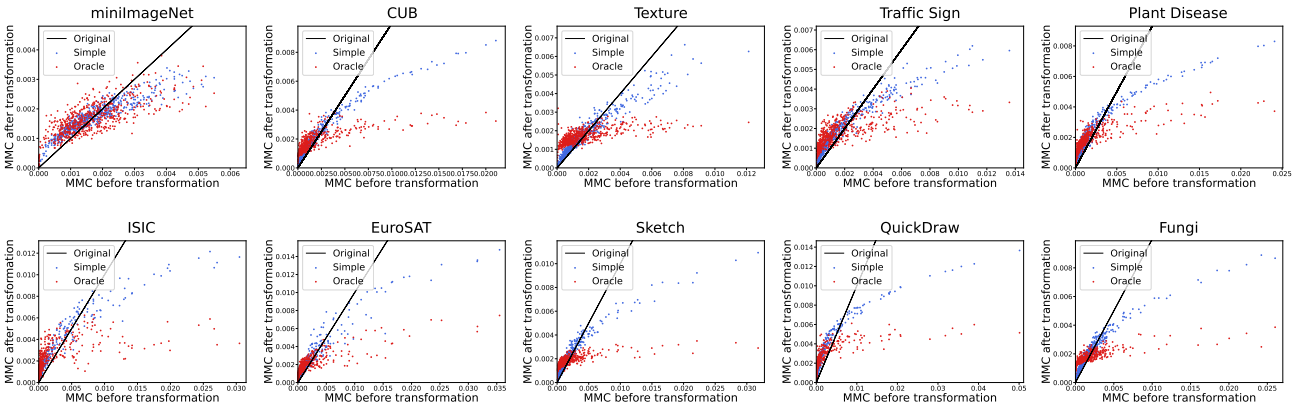

Figure 6. **Visualization of MMC of ten datasets ω_D before and after the use of simple and oracle transformation.** In each plot, a point represents a channel, and the x-axis and y-axis represent the MMC before and after transformation respectively, averaged over all possible binary tasks in the corresponding dataset. For comparison, we also plot the line $y = x$ representing the "None" scenario where none of the transformations are applied to features. The feature extractor is trained using PN on *mini*ImageNet.

Table 2. The performance gains of the oracle MMC on 5-shot binary classification tasks on various datasets. The derived MMC improves the few-shot performance of both metric and non-metric test-time methods: Nearest-Centroid Classifier (NCC) and Linear Classifier (LC).

Algorithm	Classifier	Transformation	mini	CUB	Texture	TS	PlantD	ISIC	ESAT	Sketch	QDraw	Fungi	Avg
PN	NCC	None	90.5	80.6	80.6	85.1	89.2	65.7	86.5	71.9	82.4	74.6	80.7
		Simple	91.3	82.4	83.1	85.8	93.0	68.6	89.2	75.2	85.1	77.2	83.1
		Oracle	**93.1**	**88.7**	**87.2**	**92.4**	**95.6**	**69.1**	**91.5**	**81.2**	**89.4**	**88.4**	**87.7**
S2M2	LC	None	94.0	87.1	85.7	88.7	95.0	68.7	93.5	78.7	85.5	82.8	86.0
		Simple	94.4	88.3	87.3	91.2	96.4	72.2	93.8	81.0	89.2	84.5	87.8
		Oracle	**96.3**	**94.0**	**90.7**	**96.1**	**98.3**	**72.6**	**95.2**	**87.0**	**93.0**	**93.3**	**91.7**

MMC of D is defined as $\boldsymbol{\omega}_D = \overline{\sum_{1\le i<j\le C} \overline{\boldsymbol{\omega}_{ij}}}$, which gives average percentages of channel emphasis over all binary tasks. We visualize the oracle MMC, compared with MMC adjusted by the simple transformation and the original MMC of each dataset in Figure 6. A point in each figure represents a channel of the image features, with x and y axis being its MMC of that dataset before and after transformation, respectively. To obtain a more precise understanding, we also want to quantitatively measure difference between different MMCs or image features. To achieve this, given a distance measure $d(\cdot,\cdot)$ (not necessarily a metric), we define three levels of distances: (1) dataset-level distance $d(\boldsymbol{\omega}_{D_a}, \boldsymbol{\omega}_{D_b})$ that measures the distance between MMCs of two datasets (or the same dataset with different transformations); (2) in-dataset task-level distance $\frac{C(C+1)}{2} \sum_{1\le i<j\le C} d(\overline{\boldsymbol{\omega}^a_{ij}}, \overline{\boldsymbol{\omega}^b_{ij}})$ that measures average distance between MMCs of all tasks from a dataset obtained by different feature transformations, and (3) image-level distance $\frac{1}{|D|}\sum_{i=1}^{|D|} d(\overline{\boldsymbol{z}^i_a}, \overline{\boldsymbol{z}^i_b})$, a more fine-grained one that measures average distance between all l_1-normalized image features $\overline{\boldsymbol{z}^i_a}, \overline{\boldsymbol{z}^i_b}$ of dataset D under different feature transformations. For dataset-level distance, we adopt the normalized mean square difference $d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d}\sum_{l=1}^{d}(x_l - y_l)^2/x_l^2$, since it treats each channel equally w.r.t. to the scale and is sensitive to high deviation. However, for task-level and image-level distance, we choose the mean square difference $d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d}\sum_{l=1}^{d}(x_l - y_l)^2$ instead to avoid high variations caused by a single task or image feature that has channels with very small magnitude; see Appendix F.2 for details. We calculate the distance (1) between the original MMC of the training set (mini-train) and each test set, to see how much neural networks change channel emphasis when faced with novel tasks, (2) between the original and oracle MMC to see how much the changed emphasis is biased on each dataset, and (3) between the simple and oracle MMC of each dataset to see how much the simple transformation alleviates the problem. The results are shown in Table 3.

Neural networks are overconfident in previously learned channel importance. Comparing the first and second rows in Table 3, we can see that the adjustment of MMC that the network made on new tasks is far from enough: the distance of original MMCs between train and test set (the first row) is much smaller than that between original and oracle MMCs on the test set. This suggests channels that are important to previously learned tasks are still considered by the neural network to be important for distinguishing entirely new tasks, but in fact, the discriminative channels are very likely to change on new tasks. This can be also observed from each plot in Figure 6, where the oracle MMC pushes up channels having small magni-

Table 3. Three levels of distance between different MMCs or l_1-normalized image features. The first row shows the dataset-level distance between the original MMC of the training set (mini-train) and each test set; the second row shows the dataset-level distance between the original and oracle MMCs on each dataset; rows 3-6 show the task-level and image-level distances (both amplified by 10^6 times) between MMCs obtained by simple and oracle transformation or between original MMCs (None) and the MMCs obtained by oracle transformation. The feature extractor is trained using PN on the training set of *mini*ImageNet (mini-train).

Level	Compared dataset	Trans.	Test dataset										
			mini-train	mini-test	CUB	Texture	TS	PlantD	ISIC	ESAT	Sketch	QDraw	Fungi
Dataset	Train *v.s.* Test	None	-	0.18	1.56	0.88	1.13	1.54	2.28	1.30	1.01	1.58	0.79
	Test	None *v.s.* Oracle	0.42	0.72	3.60	1.78	4.04	3.92	3.47	5.62	4.26	3.37	3.87
Task	Test	None *v.s.* Oracle	3.60	4.04	3.53	4.13	3.68	4.09	3.15	4.24	5.31	4.22	3.38
		Simple *v.s.* Oracle	3.54	3.80	2.93	3.62	3.22	3.65	2.59	3.71	4.35	3.18	2.78
Image	Test	None *v.s.* Oracle	10.52	10.65	11.53	25.20	9.88	9.75	13.04	16.33	27.36	13.46	11.39
		Simple *v.s.* Oracle	7.98	8.14	8.69	16.74	7.06	7.34	8.43	11.22	19.50	9.32	8.71

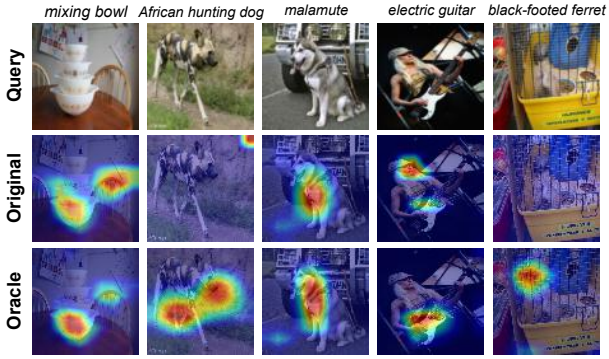

Figure 7. Examples of Grad-Cam (Selvaraju et al., 2017) class activation maps of query samples using PN before and after the oracle adjustment of MMC on binary 5-shot tasks sampled from the test set of *mini*ImageNet.

tudes and suppresses channels having large magnitudes. The magnitudes of a large number of small-valued channels are amplified $10\times$ times or more by the oracle MMC, while large-valued channels are suppressed $5\times$ times or more, and in most datasets originally large-valued channels eventually have similar channel importance to those of originally small-valued channels. The simple transformation, although not being perfect, also regularizes channels due to its smoothing property discussed in Section 3. We call this problem the *channel bias* problem.

The channel bias problem diminishes as task distribution shift lessens. The channel patterns in Figure 6 on all datasets look similar, except for *mini*ImageNet, whose overall pattern is close to the line $y = x$ representing the original MMCs. There does not exist *dominant* channels when testing on *mini*ImageNet (The maximum scale of channels is within 0.006), while on other datasets there are channels where the neural network assigns much higher but wrong MMCs which deviate far away from the $y = x$ line. In the second row of Table 3, we can also see that the distance between the original and oracle MMCs on *mini*ImageNet, especially on *mini*-train that the model trained on, is much smaller than that on other datasets[2]. Since *mini*-test has a similar task distribution with *mini*-train, we can infer that the channel bias is less serious on datasets that have similar task distribution. This explains why in Table 1 and Figure 3 the simple transformation gets a relatively low improvement when trained on *mini*-train and tested on *mini*-test, and even degrades performance when trained and tested on tasks sampled from the same task distribution.

The channel bias problem distracts the neural network from new objects. In Figure 7, we compare some class activation maps before and after the oracle adjustment of MMC. We observe that adjusting channel importance helps the model adjust the attention to the objects responsible for classification using a classifier constructed by only a few support images. This matches observation in previous work (Zhou et al., 2015; Bau et al., 2017) that different channels of image representations are responsible for detecting different objects. The task distribution shift makes models confused about which object to focus on, and a proper adjustment of channel emphasis highlights the objects of interest.

The simple transformation pushes MMCs towards the oracle ones. Observing Figure 6, it is evident that the simple transformation pushes MMCs towards the oracle ones (compared with the line $y = x$), albeit not perfectly. This observation is further confirmed by the None *v.s.* Oracle and Simple *v.s.* Oracle comparison of fine-grained task-level and image-level distance shown from the third row to the last row of Table 3. On each of the test-time dataset, the distance between MMCs obtained by simple and oracle transformation is significantly smaller than that bewteen original MMCs and the MMCs obtained by oracle transformation.

[2]Unnormalized mean square difference ignores critical changes of small-valued channels. This is why we do not observe similar phenomenon from the task and image-level difference; see Appendix F.2 for detailed explanations.

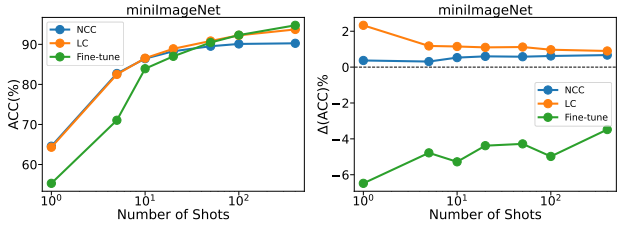

Figure 8. Shot analysis of *mini*ImageNet. Left: performance of different test-time methods. Right: performance gains of the simple transformation using different test-time methods.

4. Analysis of the Number of Shots

We have seen that the channel bias problem is one of the main reasons why image representations cannot generalize well to new few-shot classification tasks. However, two questions remain to be answered: (1) we are still unclear whether this problem is only tied with few-shot image classification. In all previous experiments, we tested on tasks where only 5 labeled images per class are given. What will happen if we have more training examples in the new task? (2) How much will different test-time methods be influenced by the channel bias problem? If we have the opportunity to fine-tune the learned representations, will the proposed simple transformation still work?

In order to give answers to these questions, we conduct shot analysis experiments on three representative test-time methods that are adopted or are the basis of most mainstream few-shot classification algorithms: (1) The metric-based method Nearest-Centroid Classifier (NCC) presented in ProtoNet, which first average image features of each class in the support set to form class centroids and then assign query features to the class of the nearest centroid; (2) Linear Classifier (LC), which trains a linear layer upon learned image features in the support set, and (3) Fine-tuning, which fine-tunes the feature extractor together with the linear layer using images in the support set. The feature extractor is trained using the state-of-the-art S2M2 algorithm on the training set of *mini*ImageNet, and we test it on the test set of *mini*ImageNet using the above three test-time methods with different numbers of labeled images in each class of the support set. The results are shown in Figure 8. We show the original accuracy of all methods, as well as the impact of simple transformation on the performance.

We first take a look at the right plot, which shows the impact of the simple transformation on all the methods. The performance gains on NCC and LC stay at a relatively high value for all tested shots, which is up to 400 labeled images per class. This indicates that the channel bias problem is not only linked to few-shot settings, but also exists in many-shot settings. However, when we have abundant support images, we have an alternative choice of fine-tuning the feature extractor directly. Fine-tuning methods have the potential to fully resolve the channel bias problem by directly modifying the image representation and rectifying the channel distribution. The right figure shows that the simple transformation does not improve fine-tuning methods, so indeed the channel bias problem has been largely alleviated. In the left figure, the fine-tuning method exhibits its advantages in many-shot setting, but falls short in few-shot settings. Therefore, we can infer that the channel bias problem exists only in the few-shot setting where freezing the feature extractor and building the classifier on learned features becomes a better choice.

We also have another notable observation. While the performance gain of simple transformation on NCC stays around a fixed value, the performance gain on LC decreases with the increase of shots. Thus the channel bias problem is alleviated to some extent in many-shot settings. This is because more labeled data tells the linear classifier sufficient information about intra-class variance of data, making it possible to adjust MMC by modifying the scale of each row of the linear transformation matrix. So Linear Classifier can stably increase its performance when more labeled data comes in, until no more linear separation can be achieved, and also the time fine-tuning should get into play to adjust the feature space directly.

5. Discussion and Related Work

Task distribution shift. Task distribution shift may happen when a model faces category shift, domain shift or granularity shift. Conventional benchmarks of FSL only consider category shift, i.e. the categories are disjoint for training and testing, such as *mini*ImageNet (Vinyals et al., 2016) and CIFAR-FS (Bertinetto et al., 2019). In *cross-domain* few-shot learning (Chen et al., 2019), domain shift exists between train and test-time tasks, and several later benchmarks such as BSCD-FSL (Guo et al., 2020) and Meta-dataset (Triantafillou et al., 2020) both target at such setting. Recently, the shift of granularities of categories has been considered as another type of task distribution shift, and is also called Coarse-to-Fine Few-Shot (C2FS) Learning (Luo et al., 2021a; Bukchin et al., 2021; Yang et al., 2021a), which trains a model on coarse-labeled images and tests on few-shot tasks that aim at distinguishing between fine-grained subclasses of training categories. Our work reveals that all three types of task distribution shift have a similar phenomenon of channel bias problem.

The influence of task distribution shift on FSL has been firstly studied in (Doersch et al., 2020). They find that the representation constructed by meta-learning algorithms cannot capture useful discriminative information outside of the training categories. They solve this problem by highlighting the crucial spatial information for classification, using a cross-attention module between support and query features

in new tasks. The algorithm COSOC (Luo et al., 2021b) also considers filtering task-irrelevant spatial information, but it achieves it more directly. They identify image background as harmful information in both training and testing and design a method to remove the background, in order to reduce the difficulty of category transfer. The perspective of our work is different, not focusing on discriminative spatial positions of features, but orthogonally taking inspections on discriminative channel information of features.

Test-time methods for FSL. The presented three methods in Section 4 represent three types of mainstream algorithms in FSL. (1) **Finetuning**—Optimization-based algorithms, originated mainly from MAML (Finn et al., 2017), optimizes both the learned feature extractor and classifier together at test-time. Most work that fall into this type use meta-learning to train the network (Rusu et al., 2019; Rajeswaran et al., 2019; Zintgraf et al., 2019; Park & Oliva, 2019). When training adopts conventional supervised approaches, the method turns to resemble transfer learning approaches, and is adopted in (Dhillon et al., 2020) and BiT (Kolesnikov et al., 2020). In the experiments of Section 4, we notice that in few-shot settings, although alleviating channel bias problem, fine-tuning method performs generally worse and may require very different hyperparameters for different test-time datasets to avoid overfitting, which is impossible to achieve in a realistic few-shot scenario, thus we believe finetuning would not be the best test-time choice. (2) **NCC**—metric-based algorithms (Vinyals et al., 2016; Snell et al., 2017; Zhang et al., 2020; Hou et al., 2019; Doersch et al., 2020) that aim at learning a well-shaped feature space equipped with a distance metric for comparing the similarity of images, on which the test-time prediction depends. Metric-based methods, as we have shown, benefit from inductive bias given by the metric and thus are widely adopted in state-of-the-art algorithms. (3) **LC**—most conventionally trained methods adopt LC as the test-time methods (Chen et al., 2019; Mangla et al., 2020; Tian et al., 2020; Liu et al., 2020; Rizve et al., 2021), and two meta-learning algorithms MetaOpt (Lee et al., 2019) and ANIL (Raghu et al., 2020) use LC in both training and testing. The importance of a good quality of image representation is mainly figured out from this line of work.

Other feature transformations in FSL. LFT (Tseng et al., 2020) introduces learnable channel-wise feature transformations into training for cross-domain few-shot learning. The transformations are put inside backbone, instead of on top of representations, and are only used at train time, learned in a learning-to-learn fashion using multiple domains of datasets. Z-score transformation upon image representations is introduced in (Fei et al., 2021) for solving the hubness problem of image representations in FSL. Feature-wise linear modulation (FiLM) (Perez et al., 2018) that turns scaling and shifting coefficients in batch normalization layer (seen as parameters of a linear feature transformation) into dataset- or task-dependent learnable parameters has been adopted in several FSL algorithms (Oreshkin et al., 2018; Requeima et al., 2019; Triantafillou et al., 2021; Li et al., 2022). The core idea of these methods is to only tune the FiLM modules at test time in order to reduce overfitting. Thus these methods in some sense belong to finetuning-based methods, and have the potential to perform better than vanilla finetuning in low-shot settings. Contrary to our work, all methods discussed above do not discover or target at the channel bias problem. The most relevant method to our paper may be ConFeSS (Das et al., 2022), a framework that masks task-irrelevant channels in image representations at test time for cross-domain few-shot learning. Our work shows that the success of ConFeSS may be attributed to alleviating the channel bias problem by abandoning overconfident channels when transferred to novel tasks.

6. Conclusion

In this paper, we reveal the channel bias problem in few-shot image classification. The problem can be alleviated by a simple channel-wise feature transformation presented in this work. This transformation, used at test-time without adding any computation overhead, can be applied to most pre-trained convolutional neural networks and few-shot learning algorithms. We show it serves as prior knowledge that regularizes the channel distribution of features. Further analysis, including a derivation of the oracle MMC adjustment, analyzes comprehensively the channel bias problem. We hope that the channel bias problem revealed in this work, along with analysis of different test-time methods, can provide the community with a better understanding of task distribution shift and representation transfer in few-shot classification, which may in turn help produce better algorithms.

Acknowledgments

Special thanks to Qi Yong for providing indispensable spiritual support for the work. We also would like to thank all reviewers for very constructive comments that help us improve the paper. This work was partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204), and a key program of fundamental research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826).

References

Agarwal, M., Yurochkin, M., and Sun, Y. On sensitivity of meta-learning to support data. In *Advances in Neural Information Processing Systems*, 2021.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba,

A. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6541–6549, 2017.

Bertinetto, L., Henriques, J. F., Torr, P. H. S., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

Bukchin, G., Schwartz, E., Saenko, K., Shahar, O., Feris, R., Giryes, R., and Karlinsky, L. Fine-grained angular contrastive learning with coarse labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8730–8740, 2021.

Cantelli, F. P. Sui confini della probabilita. In *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de settembre di 1928*, pp. 47–60, 1929.

Chen, W., Liu, Y., Kira, Z., Wang, Y. F., and Huang, J. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

Chen, Y., Liu, Z., Xu, H., Darrell, T., and Wang, X. Meta-baseline: exploring simple meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9062–9071, 2021.

Das, D., Yun, S., and Porikli, F. Confess: A framework for single source cross-domain few-shot learning. In *International Conference on Learning Representations*, 2022.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2020.

Doersch, C., Gupta, A., and Zisserman, A. Crosstransformers: spatially-aware few-shot transfer. In *Advances in Neural Information Processing Systems*, 2020.

Fei, N., Gao, Y., Lu, Z., and Xiang, T. Z-score normalization, hubness, and few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 142–151, 2021.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135, 2017.

Guo, Y., Codella, N., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. A broader study of cross-domain few-shot learning. In *European Conference on Computer Vision*, pp. 124–141, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. J. The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8769–8778, 2018.

Hou, R., Chang, H., Ma, B., Shan, S., and Chen, X. Cross attention network for few-shot classification. In *Advances in Neural Information Processing Systems*, pp. 4005–4016, 2019.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision*, pp. 491–507, 2020.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1106–1114, 2012.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Li, W.-H., Liu, X., and Bilen, H. Cross-domain few-shot learning with task-specific adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7161–7170, 2022.

Liu, B., Cao, Y., Lin, Y., Li, Q., Zhang, Z., Long, M., and Hu, H. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, pp. 438–455, 2020.

Luo, X., Chen, Y., Wen, L., Pan, L., and Xu, Z. Boosting few-shot classification with view-learnable contrastive learning. In *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2021a.

Luo, X., Wei, L., Wen, L., Yang, J., Xie, L., Xu, Z., and Tian, Q. Rectifying the shortcut learning of background for few-shot learning. In *Advances in Neural Information Processing Systems*, 2021b.

Mangla, P., Singh, M., Sinha, A., Kumari, N., Balasubramanian, V. N., and Krishnamurthy, B. Charting the right manifold: Manifold mixup for few-shot learning. In *IEEE Winter Conference on Applications of Computer Vision*, pp. 2207–2216, 2020.

Oreshkin, B. N., López, P. R., and Lacoste, A. TADAM: task dependent adaptive metric for improved few-shot learning. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, pp. 719–729, 2018.

Park, E. and Oliva, J. B. Meta-curvature. In *Advances in Neural Information Processing Systems*, pp. 3309–3319, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, pp. 2825–2830, 2011.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415, 2019.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Rizve, M. N., Khan, S. H., Khan, F. S., and Shah, M. Exploring complementary strengths of invariant and equivariant representations for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10836–10846, 2021.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. Imagenet large scale visual recognition challenge. In *IJCV*, volume 115, pp. 211–252, 2015.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: A good embedding is all you need? In *European Conference on Computer Vision*, pp. 266–282, 2020.

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P., and Larochelle, H. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

Triantafillou, E., Larochelle, H., Zemel, R. S., and Dumoulin, V. Learning a universal template for few-shot dataset generalization. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10424–10433, 2021.

Tseng, H., Lee, H., Huang, J., and Yang, M. Cross-domain few-shot classification via learned feature-wise transformation. In *International Conference on Learning Representations*, 2020.

Tukey, J. W. et al. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Yang, J., Yang, H., and Chen, L. Towards cross-granularity few-shot learning: Coarse-to-fine pseudo-labeling with visual-semantic meta-embedding. In *ACM Multimedia Conference*, pp. 3005–3014, 2021a.

Yang, S., Liu, L., and Xu, M. Free lunch for few-shot learning: Distribution calibration. In *International Conference on Learning Representations*, 2021b.

Ye, H. and Chao, W. How to train your maml to excel in few-shot classification. In *International Conference on Learning Representations*, 2022.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 2016.

Zhang, C., Cai, Y., Lin, G., and Shen, C. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12200–12210, 2020.

Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., and Torralba, A. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations*, 2015.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Fast context adaptation via meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 7693–7702, 2019.

A. Proof of Proposition 3.1

Lemma A.1. *(**Cantelli's inequality** (Cantelli, 1929)) Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then for any $k > 0$,*

$$\mathbb{P}(X - \mu \geq k\sigma) \leq \frac{1}{1+k^2}. \tag{7}$$

Lemma A.2. *Let $a_i > 0, b_i > 0, i = 1, ..., D$. Define $f : [0, +\infty)^D / \{\mathbf{0}\} \to \mathbb{R}$ by*

$$f(\boldsymbol{x}) = \frac{\sum_{i=1}^{D} b_i x_i^2}{(\sum_{i=1}^{D} a_i x_i)^2}, \tag{8}$$

then

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^{D} \frac{a_i^2}{b_i}}. \tag{9}$$

The mimimum value is reached when there exists a constant $c > 0$, such that $\forall i \in [D], x_i = \frac{a_i c}{b_i}$.

Proof. We show it by induction on dimension D. Denote the domain of f by U, i.e., $U = [0, +\infty)^D / \{\mathbf{0}\}$.

When $D = 1$, $f(x) \equiv \frac{b_1}{a_1^2}$ is a constant, so the result holds.

Assume that when $D \leq k$, the result holds. We now prove that when $D = k + 1$, the result holds. It is obvious that $\forall c > 0, f(c\boldsymbol{x}) = f(\boldsymbol{x})$, thus it suffices to find a minimum point in $\overline{\mathcal{S}} = \{\boldsymbol{x} | a \leq ||\boldsymbol{x}||_2 \leq b \text{ and } \boldsymbol{x} \in U\}$ for any chosen $0 < a < b$. Since $\overline{\mathcal{S}}$ is a closed set and f is continuous, the minimum point exists. The minimum point either lies on the hyperspheres: $\partial S = \{\boldsymbol{x} | ||\boldsymbol{x}||_2 = a \text{ and } \boldsymbol{x} \in U\} \cup \{\boldsymbol{x} | ||\boldsymbol{x}||_2 = b \text{ and } \boldsymbol{x} \in U\}$ or in between: $\mathcal{S} = \{\boldsymbol{x} | a < ||\boldsymbol{x}||_2 < b \text{ and } \boldsymbol{x} \in U\}$. If there exists a minimum point on one of the hyperspheres, say, the outer hypersphere $\{\boldsymbol{x} | ||\boldsymbol{x}||_2 = b \text{ and } \boldsymbol{x} \in U\}$, then there exists another minimum point $\frac{(a+b)\boldsymbol{x}}{2b} \in \mathcal{S}$ (or $\frac{(a+b)\boldsymbol{x}}{2a} \in \mathcal{S}$ for the inner hypersphere). Thus it suffices to find the minimum point in $\mathcal{S}$.

Let $\mathcal{S}_{/i} = \{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{S} \text{ and } x_i = 0\}$ and $\mathcal{S}_{>0} = \{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{S} \text{ and } x_i > 0 \text{ for all } i \in [k+1]\}$. We have $\mathcal{S} = (\cup_{i=1}^{k+1} \mathcal{S}_{/i}) \cup \mathcal{S}_{>0}$. If $\boldsymbol{x} \in \mathcal{S}_{/i}$, then

$$f(\boldsymbol{x}) = \frac{\sum_{j=1}^{i-1} b_j x_j^2 + \sum_{j=i+1}^{k+1} b_j x_j^2}{(\sum_{j=1}^{i-1} a_j x_j + \sum_{j=i+1}^{k+1} a_j x_j)^2}, \tag{10}$$

which can be seen as a function with input dimension k. Thus from the induction, we have

$$\min_{\boldsymbol{x} \in \mathcal{S}_{/i}} f(\boldsymbol{x}) = \frac{1}{\sum_{j=1}^{i-1} \frac{a_j^2}{b_j} + \sum_{j=i+1}^{k+1} \frac{a_j^2}{b_j}}. \tag{11}$$

Next, we handle the setting when $\boldsymbol{x} \in \mathcal{S}_{>0}$, i.e., find all possible extreme points of f inside $\mathcal{S}_{>0}$. Note that

$$\frac{\partial f}{\partial x_i} = \frac{2(b_i x_i \sum_{j=1}^{k+1} a_j x_j - a_i \sum_{j=1}^{k+1} b_j x_j^2)}{(\sum_{j=1}^{k+1} a_j x_j)^3}, \tag{12}$$

then an extreme point $\boldsymbol{x}$ must satisfy

$$b_i x_i \sum_{j=1}^{k+1} a_j x_j - a_i \sum_{j=1}^{k+1} b_j x_j^2 = 0, \forall i \in [k+1], \tag{13}$$

which is equivalent to

$$x_i = \frac{a_i \sum_{j=1}^{k+1} b_j x_j^2}{b_i \sum_{j=1}^{k+1} a_j x_j} = (\frac{\sum_{j=1}^{k+1} b_j x_j^2}{\sum_{j=1}^{k+1} a_j x_j}) \frac{a_i}{b_i}, \forall i \in [k+1]. \tag{14}$$

Thus $x_i = c\frac{a_i}{b_i}$, where $c = \frac{\sum_{j=1}^{k+1} b_j x_j^2}{\sum_{j=1}^{k+1} a_j x_j}$. Furthermore, it is easy to show that $x_i = c\frac{a_i}{b_i}$ satisfies Eq. (14) for any $c > 0$. Denote any of the points satisfying this property as $\boldsymbol{x}^*$, then

$$f(\boldsymbol{x}^*) = \frac{1}{\sum_{i=1}^{k+1} \frac{a_i^2}{b_i}}. \tag{15}$$

Comparing Eq. (11) and Eq. (15), it can be seen that

$$f(\boldsymbol{x}^*) < \min_{\boldsymbol{x} \in \mathcal{S}_{/i}} f(\boldsymbol{x}), \forall i \in [k+1]. \tag{16}$$

Finally, note that ∂S and $\{\mathcal{S}_{/i}\}_{i=1}^{k+1}$ constitute the boundary of $\mathcal{S}_{>0}$, thus Eq. (16) and earlier discussion about ∂S indicate that $\boldsymbol{x}^*$ is the minimum point of f, as desired. □

Proposition 3.1. *Assume that $\mu_{1,l} \neq \mu_{2,l}$ and $\sigma_{1,l} + \sigma_{2,l} > 0$ hold for any $l \in [d]$, then we have*

$$\mathcal{R} \leq \frac{8\sum_{l=1}^{d} \omega_l^4 (\widetilde{\sigma}_{1,l} + \widetilde{\sigma}_{2,l})^2}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2}. \tag{17}$$

To minimize this upper bound, the adjusted oracle MMC of each channel ω_l should satisfy:

$$\omega_l \propto \frac{|\mu_{1,l} - \mu_{2,l}|}{\sigma_{1,l} + \sigma_{2,l}}. \tag{18}$$

Proof.

$$\begin{aligned}
\mathcal{R} &= \frac{1}{2}[\mathbb{P}_{\boldsymbol{z}_1 \sim D_1}(||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_1 - \widetilde{\boldsymbol{\mu}}_1)||_2 > ||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_1 - \widetilde{\boldsymbol{\mu}}_2)||_2) + \mathbb{P}_{\boldsymbol{z}_2 \sim D_2}(||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_2 - \widetilde{\boldsymbol{\mu}}_2)||_2 > ||\boldsymbol{\omega} \odot (\widetilde{\boldsymbol{z}}_2 - \widetilde{\boldsymbol{\mu}}_1)||_2)] \\
&= \frac{1}{2}[\mathbb{P}_{\boldsymbol{z}_1 \sim D_1}(\sum_{l=1}^{d} \omega_l^2[(\widetilde{z}_{1,l} - \widetilde{\mu}_{1,l})^2 - (\widetilde{z}_{1,l} - \widetilde{\mu}_{2,l})^2] > 0) + \mathbb{P}_{\boldsymbol{z}_2 \sim D_2}(\sum_{l=1}^{d} \omega_l^2[(\widetilde{z}_{2,l} - \widetilde{\mu}_{2,l})^2 - (\widetilde{z}_{2,l} - \widetilde{\mu}_{1,l})^2] > 0)] \\
&= \frac{1}{2}[\mathbb{P}_{\boldsymbol{z}_1 \sim D_1}(\sum_{l=1}^{d} \omega_l^2(1 - \widetilde{z}_{1,l})(\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l}) > 0) + \mathbb{P}_{\boldsymbol{z}_2 \sim D_2}(\sum_{l=1}^{d} \omega_l^2(1 - \widetilde{z}_{2,l})(\widetilde{\mu}_{2,l} - \widetilde{\mu}_{1,l}) > 0)] \\
&\leq \frac{2\sum_{l=1}^{d} \omega_l^4 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2 (\widetilde{\sigma}_{1,l}^2 + \widetilde{\sigma}_{2,l}^2)}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2} && \text{[Applying Lemma A.1]} \\
&\leq \frac{2\sum_{l=1}^{d} \omega_l^4 (\widetilde{\mu}_{1,l} + \widetilde{\mu}_{2,l})^2 (\widetilde{\sigma}_{1,l}^2 + \widetilde{\sigma}_{2,l}^2)}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2} && [\widetilde{\mu}_{1,l}\widetilde{\mu}_{2,l} \geq 0] \\
&= \frac{8\sum_{l=1}^{d} \omega_l^4 (\widetilde{\sigma}_{1,l}^2 + \widetilde{\sigma}_{2,l}^2)}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2} && \text{[Standadization: } (\widetilde{\mu}_{1,l} + \widetilde{\mu}_{2,l})/2 = 1] \\
&\leq \frac{8\sum_{l=1}^{d} \omega_l^4 (\widetilde{\sigma}_{1,l} + \widetilde{\sigma}_{2,l})^2}{(\sum_{l=1}^{d} \omega_l^2 (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2)^2}. && [\widetilde{\sigma}_{1,l}\widetilde{\sigma}_{2,l} \geq 0]
\end{aligned} \tag{19}$$

Let $x_l = \omega_l^2, a_l = (\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2, b_l = (\widetilde{\sigma}_{1,l} + \widetilde{\sigma}_{2,l})^2$, then according to Lemma A.2, the minimum value of the upper bound (19) is reached when $\omega_l^2 \propto \frac{(\widetilde{\mu}_{1,l} - \widetilde{\mu}_{2,l})^2}{(\widetilde{\sigma}_{1,l} + \widetilde{\sigma}_{2,l})^2} = \frac{(\mu_{1,l} - \mu_{2,l})^2}{(\sigma_{1,l} + \sigma_{2,l})^2}$, i.e., $\omega_l \propto \frac{|\mu_{1,l} - \mu_{2,l}|}{\sigma_{1,l} + \sigma_{2,l}}$. □

B. Training and Evaluation Details

For S2M2 and MoCo-v2 in Table 1, we directly use the official publicly-available pre-trained checkpoints. All other algorithms in Table 1 are trained using a learning rate 0.1 with cosine decay schedule without restart. SGD with momentum 0.9 is adopted as the optimizer. For all meta-learning algorithms, a total of 60000 5-way 5-shot tasks are sampled for training,

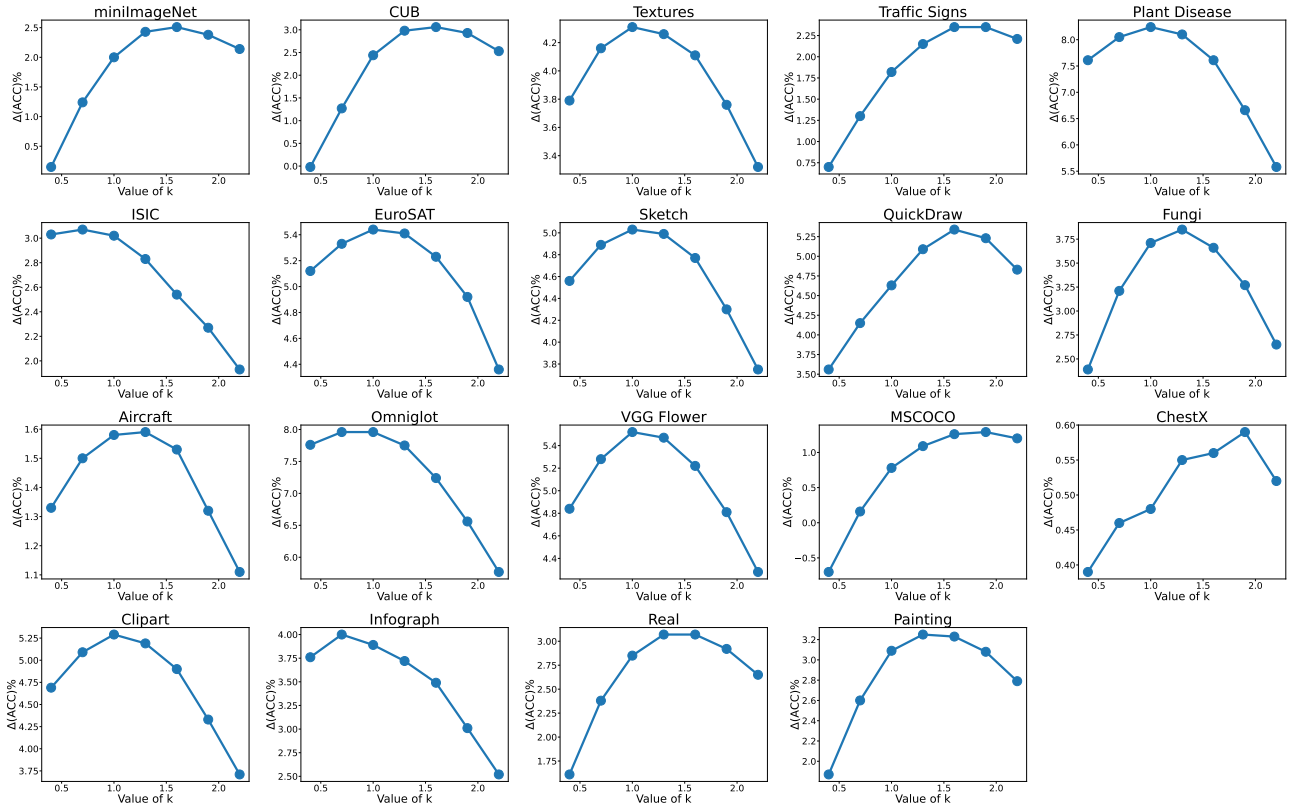

Figure 9. The effect of the hyperparameter k in the simple transformation on each test-time dataset. The feature extractor is trained on *mini*ImageNet using PN. All experiments are conducted on 10000 5-way 5-shot tasks sampled with a fixed seed.

each of which contains 15 query images per class. The batch size (number of sampled tasks of each iteration) is 4. All other hyperparameters of MetaOpt match the default settings in the original paper. All conventionally-trained algorithms are trained for 60 epochs, and the batch size is set to 128. For the training of the CE (Cross-Entropy) algorithm, we normalize the representation before the fully-connected layer. We find that if we do not normalize the representation during the training of CE, the simple transformation does not work. We leave it for future work to investigate this phenomenon.

For the test-time linear classification method we implement for MoCo and S2M2 in Table 1 and Figure 8, we adopt the Logistic Regression implementation of scikit-learn (Pedregosa et al., 2011).

C. The Effect of Hyperparameter k

In Figure 9, we show how the hyperparameter k in Eq. (1) influences the few-shot classification performance. On all datasets, As the k becomes larger, the accuracy first increases and then decreases. The optimal value of k varies for different datasets, ranging from 0.6 to 1.8. That being said, the simple transformation gives a relatively stable performance improvement on all datasets when $k \in [1, 2]$. Notably, datasets with larger task distribution shift often give a smaller optimal k. This phenomenon is reasonable because as seen from Figure 2, a smaller k leads to a larger smoothing effect, and the transformation can better rectify the channel distribution when the task distribution shift is also larger.

D. Attempts at Handling Negative Output Values of Neural Networks

As shown in Section 3, the MMC of image representations can represent the emphasis of neural network on different channels. However, things get complicated if the output of the neural network can take negative values. If the value of a channel represents the activation of a feature, it is difficult to say whether a large negative value means a large or small activation. At this time, we consider negative value as a signal of feature activation as well, which leads to the following

Table 4. Performance gains when applying the extended version of the simple transformation to ResNet-12 with Leaky ReLU trained on mini-train.

Algorithm	*mini*-test	CUB	Texture	TS	PlantD	ISIC	ESAT	Sketch	QDraw	Fungi	Avg
PN	76.0+0.5	59.3+0.6	62.0+0.7	66.3-0.2	78.2+2.8	38.1+1.3	75.1+0.8	52.7+0.3	66.5+2.9	55.4+0.0	63.0+1.0
CE	79.4+0.2	64.5+0.8	66.1-0.2	69.9+0.1	84.9+2.1	40.1+0.1	77.6+0.4	53.6+0.4	72.1+3.7	57.4+1.1	66.6+0.9

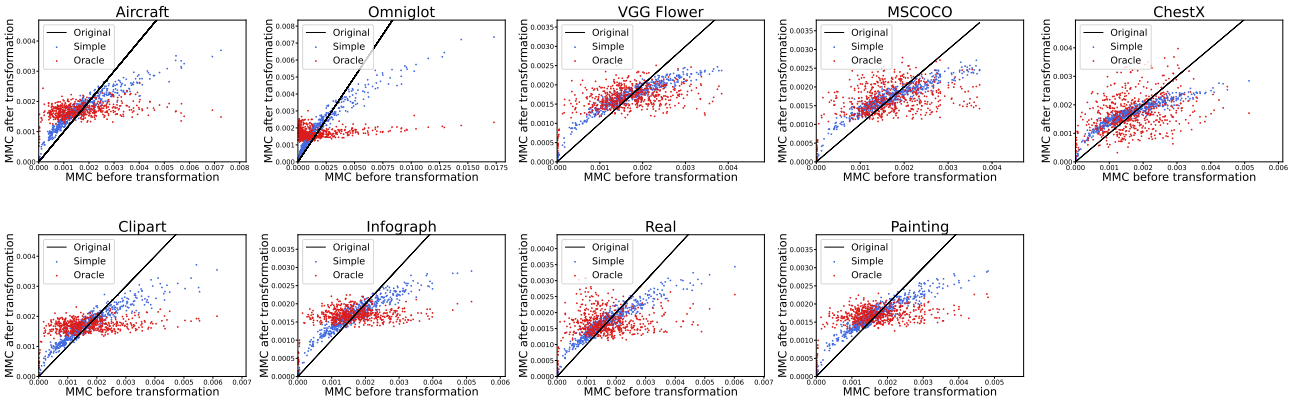

Figure 10. **The Visualization of MMC of the other nine datasets before and after the use of simple and oracle transformation.** All notations are the same as in Figure 6.

simple extension of the transformation:

$$\phi_k(\lambda) = \begin{cases} \frac{\text{sign}(\lambda)}{ln^k(\frac{1}{|\lambda|}+1)}, & |\lambda| > 0 \\ 0, & \lambda = 0 \end{cases} \tag{20}$$

now this extended simple transformation can be applied directly to the standard ResNet-12 using leaky ReLU, and the results are shown in Table 4. While leaky ReLU improves the basic performance compared to vanilla ReLU, the improvement of the simple transformation becomes significantly smaller. We conjecture that a large negative magnitude of a channel does not strictly mean that this channel is important. It is future work to investigate how to exactly measure channel importance in such circumstances.

E. Necessary Ingredients for a Good Transformation

One may ask that whether all of the three properties presented in Eq. (3) are necessary for a transformation to successfully improve few-shot learning performance, or whether there exist good transformations other than $\phi_k(\lambda)$ considered in the main article. To verify the necessity of all properties, we design several functions, each of which does not satisfy one of the properties. First, we consider the function $p(\lambda) = ln(a\lambda + 1)$, where $a > 0$. This function has positive derivative and negative second derivative, but does not have large enough derivative near zero ($p'(0) = a$). In the left plot of Figure 11, we see that the improvement brought by this function is smaller than $\phi_{1.3}(\lambda)$, and that the gap becomes smaller when a increases. This validates the necessity of having a large enough derivative near zero. We then consider the piece-wise function

$$q(\lambda) = \begin{cases} \phi_k(x), & 0 \leq \lambda < \lambda_0 \\ a_2\lambda^2 + a_1\lambda + a_0, & \lambda \geq \lambda_0 \end{cases} \tag{21}$$

where the values of a_2, a_1, a_0 ensure the smoothness of $q(\lambda)$ at $\lambda = \lambda_0$ up to first derivative, and also control the position of the extreme point $x_0 = -\frac{a_1}{2a_2}$. This function does not have positive derivative when $\lambda \geq x_0$. We set $x_0 = 0.05$, and change the value of λ_0. The results are shown in the second plot in Figure 11. As seen, introducing negative derivative into the transformation substantially degrades performance. Finally, the property of having negative second derivative can be naturally broken by increasing the value of k in $\phi_k(\lambda)$, and Figure 9 shows that doing this would degrade performance.

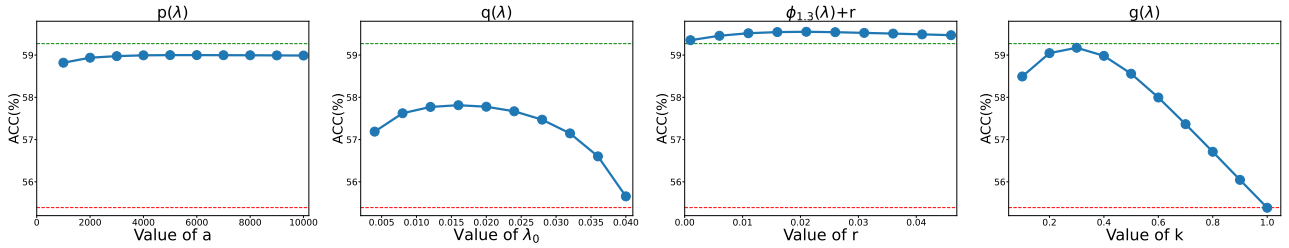

Figure 11. Exploration of necessary ingredients for a good channel-wise transformation. The accuracies show the average 5-way 5-shot performance over all 19 datasets using PN trained on *mini*ImageNet. The red dashed line shows the original performance; the green dashed line shows the performance when using the simple transformation $\phi_{1.3}(\lambda)$. The leftmost plot shows the performance when using the function $p(\lambda) = ln(a\lambda + 1)$. The second plot shows the performance when using the piece-wise function $q(\lambda)$. The third plot shows the performance when adding a constant r to the simple transformation $\phi_{1.3}(\lambda)$. The rightmost plot shows the performance when using the power function $g(\lambda) = x^k$.

Since inactivate channels may represent absence of a feature in a task instead of having low emphasis, thus they are likely to have no importance. Therefore another property $\phi_k(0) = 0$, not shown in Eq. (3), could also be important for a good transformation. To investigate this, we add a constant r to $\phi_k(\lambda)$ and see how the performance would change. The third plot in Figure 11 shows the opposite result: a small constant added to the transformation helps further improve the performance. As adding this constant has more influence on small-valued channels, we conjecture that this helps further alleviate the channel bias problem, and that some inactivate channels indeed should gain some focus.

Apparently, $\phi_k(\lambda)$ is not the only function that satisfies all the three properties. We consider the power function $g(\lambda) = \lambda^k$, where $k > 0$. Although being very simple, this function matches all desired properties. The rightmost plot in Figure 11 shows that this function can indeed improve the performance as well. Note that this function has been used in (Yang et al., 2021b), where it is called the Tukey's Ladder of Powers transformation (Tukey et al., 1977), and is used to transform the feature distribution to be more like a Gaussian distribution. Here we show that mitigating the channel bias problem may be another reason for why it works.

F. Details of MMC Calculation and Comparison

F.1. Oracle Transformation

To apply the oracle transformation, for every test-time dataset D, we first calculate feature mean $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\sigma}_c$ of each class c in D. Then for every sampled binary classification task $\tau = \{\mathcal{S}_\tau, \mathcal{Q}_\tau\}$ that aims at discriminating two classes c_1 and c_2, we calculate the oracle MMC $\boldsymbol{\omega}$ directly from Eq. (6). Next, we standardize each image feature $\boldsymbol{z}$ in $\mathcal{S}_\tau$ and $\mathcal{Q}_\tau$, and multiply it by $\boldsymbol{\omega}$ to obtain the transformed feature $\boldsymbol{z} \leftarrow \boldsymbol{\omega} \odot \widetilde{\boldsymbol{z}}$. The transformed features can be already used for classification, but we find that for some channels with very small means μ_l, the corresponding value of oracle MMC ω_l becomes too big, deviating from what we expect. To avoid generating such outliers, we additionally restrict that for every channel l, the ratio of transformed MMC to the original MMC should not surpass a threshold, i.e., $\omega_l/\omega_l^o \leq \alpha$ for some $\alpha \in \mathbb{R}^+$. If a channel l does not meet this requirement, we simply set $\omega_l = \omega_l^o$. In all of our experiments, we set $\alpha = 50$. The optimal MMC visualizied in Figure 6 and Figure 10 is also computed using this strategy. We leave it for future work to investigate the reason behind such phenomenon.

F.2. Choice of Distance Measure

The normalized mean square difference $d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d}\sum_{l=1}^{d}(x_l - y_l)^2/x_l^2$ has the advantage of having equal treatment for both channels with small and large values. However, it can be largely influenced by "outlier channel" with very small x_l. Since dataset-level MMC $\boldsymbol{\omega}_D$ is averaged over MMCs of all possible tasks in D, it is more stable and can use such distance measure. The task-level MMC and image features have much higher variances acorss tasks/images, thus for these two fine-grained settings we just use the mean square difference $d(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{d}\sum_{l=1}^{d}(x_l - y_l)^2$. This, however, introduces another problem that critical changes of channels with small values are always ignored by such unnormalized distance. Let's see a simple example. Let $\boldsymbol{\omega}_1 = (0.05, 0.08, 0.87)$ and $\boldsymbol{\omega}_2 = (0.4, 0.3, 0.3)$ be two 3-dimensional $l1$-normalized MMCs. Assume that after transformation, their $l1$-normalized values become $\boldsymbol{\omega}_1' = (0.15, 0.1, 0.75)$ and $\boldsymbol{\omega}_2' = (0.55, 0.22, 0.23)$. The value of the first dimension of $\boldsymbol{\omega}_1$ triples and surpasses that of the second dimension after transformation, thus the

Table 5. Found best hyperparameters of the test-time finetuning method on *mini*ImageNet. The feature extrator is ResNet-12, trained by S2M2 algorithm.

Shot	Batch size	Number of epochs	Learning rate
1	5	10	0.1
5	25	30	0.05
10	50	50	0.05
20	50	100	0.02
50	64	100	0.01
100	64	500	0.005
400	64	500	0.005

Table 6. The influence of using different seeds during training or testing. Average 5-way 5-shot performance of PN over 19 datasets with 95% confidence interval (over 5 trials) are shown. The training set is *mini*-train.

Seed	*mini*-test	CUB	Texture	TS	PlantD	ISIC	ESAT	Sketch	QDraw	Fungi
Test	+2.36±0.06	+2.98±0.03	+4.26±0.10	+2.37±0.06	+8.04±0.14	+2.80±0.10	+5.50±0.09	+5.02±0.09	+5.46±0.14	+3.90±0.11
Train	+2.08±0.33	+3.28±0.43	+3.78±0.77	+2.00±0.89	+8.21±0.80	+3.64±0.44	+4.03±1.37	+4.21±0.93	+7.01±2.62	+3.59±0.50

channel emphasis changes substantially, while the channel emphasis of ω_2 does not change much. We expect that the distance measuring the change of ω_1 should be much larger than that measuring the change of ω_2. The normalized mean square differences between the MMC before and after transformation are 1.36 and 0.09 for ω_1 and ω_2 respectively, which is in line with our intuition. However, the mean square differences are 0.008 and 0.011 for ω_1 and ω_2 respectively. Thus under such circumstances, the normalized mean square difference is a much better choice. Although being simple, ω_2 and ω_1 are a good analogy to the MMC pattern on *mini*ImageNet and some other datasets in Figure 6, respectively. In Figure 6, we can see that most MMC values on *mini*ImageNet are around mid-level, which resembles ω_2; most MMC values on other datasets are either very small or large, which resembles ω_1. This explains why in Table 3 the task-level and image-level differences on *mini*ImageNet are not smaller than those on other datasets. We leave it for future work to find a distance measure that could avoid unstable results, while being sensitive to small-valued channels.

G. More Details on Fine-tuning Based Method

For fine-tuning methods in Figure 8, we grid search the best hyperparameters in each shot setting *on the test set*. All best configurations are shown in Table 5. As seen, the hyperparameters of fine-tuning methods are very sensitive to the number of shots. In low-shot settings, care should be taken for controlling the total steps of finetuning and learning rate, in order to avoid overfitting. This phenomenon is also shown in (Ye & Chao, 2022), where the authors show that MAML (Finn et al., 2017), one of the most widely adopted finetuning-based methods, has a much higher optimal test-time fine-tuning steps than expected.

H. Error Bars

All results regarding performance in the main paper are shown without error bars. In Table 6, we show how different seeds affect the improvement brought by the simple transformation $\phi_k(\lambda)$. There are two seeds that could influence the result, one for training, and one for testing. When considering test seed, we fix the feature extractor and use different seeds to sample tasks; when considering train seed, we fix the test seed (same tasks) and evaluate different feature extractors trained with different seeds. As seen, while varying the test seed hardly affect the performance, varying the train seed produces some fluctuations. After considering the fluctuations, the improvement given by the transformation can still be statistically guaranteed.