Focus Your Attention when Few-Shot Classification

Haoqing Wang  Shibo Jie  Zhi-Hong Deng
School of Intelligence Science and Technology, Peking University
{wanghaoqing, parsley, zhdeng}@pku.edu.cn

Abstract

Since many pre-trained vision transformers emerge and provide strong representation for various downstream tasks, we aim to adapt them to few-shot image classification tasks in this work. The input images typically contain multiple entities. The model may not focus on the class-related entities for the current few-shot task, even with fine-tuning on support samples, and the noise information from the class-independent entities harms performance. To this end, we first propose a method that uses the attention and gradient information to automatically locate the positions of key entities in the support images, denoted as position prompts. Then we employ the cross-entropy loss between their many-hot presentation and the attention logits to optimize the model to focus its attention on the key entities during fine-tuning. This ability then can generalize to the query samples. Our method is applicable to different vision transformers (e.g., columnar or pyramidal ones), and also to different pre-training ways (e.g., single-modal or vision-language pre-training). Extensive experiments show that our method can improve the performance of full or parameter-efficient fine-tuning methods on few-shot tasks. Code is available at https://github.com/Haoqing-Wang/FORT.

1 Introduction

Based on the accumulated experience, humans can learn from few labeled samples for solving novel tasks, and this is a key aspect of human intelligence. Few-shot learning aims to imitate this ability and has attracted widely attention from the machine learning community. Traditionally, a base dataset is collected for learning prior knowledge before solving novel few-shot tasks. The meta-learning methods learn the task-shared inductive bias, e.g., parameter initialization, metric function or optimizer. The transfer learning methods learn generalizing representation to novel tasks, and achieve comparable or better performance.

Some works find that the well-learned embedding model contributes the most to few-shot classification performance, and even more important than the sophisticated meta-learning algorithms. However, the commonly-used base datasets are far from large enough for learning generalizing embedding, e.g., miniImageNet, which severely limits the performance of few-shot classification. In fact, the few-shot learning ability of humans benefits from the supervised or unsupervised learning from massive signals since birth, which helps us accumulating enough general knowledge. On the other hand, self-supervised learning can learn well-generalizing representation from nearly-free massive unlabeled data and is beneficial for broad downstream tasks. Using these pre-trained models for few-shot tasks conforms to the learning paradigm of humans and has achieved remarkable success in Natural Language Processing (NLP). In this work, we aim to adapt the pre-trained models to few-shot image classification tasks, without using any base datasets.

Recently, vision transformers play an increasingly important role in computer vision, so we use them as the backbone for few-shot tasks. Note that it is a non-trivial problem to fine-tune the pre-trained large vision transformers on few support samples due to their data-hungry

*Corresponding author.
Figure 1: Focusing on key entities via position prompts. The original pre-trained model may attend to multiple entities in a single image, and the information from class-independent entities is actually noise in the current task. The extremely few support samples make the standard fine-tuning with only classification loss unable to remove the noise information. We propose position prompts (red patches) to prompt the model where are the key patches of the input and focusing on them. This ability gained during fine-tuning can generalize from support samples to query ones. Here the white patches in attention visualization have the top highest attention scores and cover about 95% attention.

Figure 1 illustrates our framework and motivation, where we visualize the patches with top highest attention scores and covering about 95% attention (details at Section 4.2) of [CLS] token in a ViT-B/16 [11]. The input images usually contain several different entities, e.g., a bird stands on the branch (Figure 1, ‘Input’). The pre-trained model may attend to multiple entities at the same time (Figure 1, ‘Orig Attn’) and the noise information from class-independent entities harms performance. The standard fine-tuning methods tune full or partial [24, 34] parameters of the model using the classification loss on the support samples, and can achieve better performance than the backbone-frozen methods. But even with fine-tuning, the extremely few support samples prevent the model from giving sufficient attention to the class-related key entities (Figure 1, ‘FT Attn’). To this end, besides the classification loss, we use the position information of the key entities in each support sample as guidance for fine-tuning, and the model is expected to focus its attention on the key entities after fine-tuning, not only for the support samples, but also for the query ones. Concretely, we first automatically locate the position of key patches as position prompts (Figure 1, ‘Prompts’) in each support sample based on the deep explanation method, where the key patches refer to the most class-related ones and correspond to key entities. Inspired by Rollout [1], we integrate the attention information from multiple layers linearly for explanation, and further introduce gradient information as an auxiliary to ensure class-specific explanation. Unlike existing prompts [35], ours are only used to guide the fine-tuning process and not available during testing, since we need label information to obtain them. Then, we use the position prompts as the prediction target of attention, and optimize the cross-entropy loss between their many-hot presentation and the attention logits during fine-tuning. This loss enables the model to identify the key patches and focus on them (Figure 1, ‘Our Attn’), and this ability obtained during fine-tuning can generalize to query samples. We also theoretically prove that our method can increase the information from the key patches in the output feature and reduce that from other patches (noise information).

Overall, our contributions can be summarized as follows:

- We propose a new form of prompts for vision data, position prompts, to effectively adapt the pre-trained vision transformers to few-shot image classification tasks.
- Our method, denoted as FORT (FOcus youR aTtention), is applicable to the attention modules in different vision transformers, e.g., Swin [36] or ViT, and not limited to specific pre-training ways, e.g., single or multi-modal ones.

property. Inspired by the prompt methods [72, 35] from NLP, some works [78, 77, 10] synthesize the classification weights by encoding the learnable text prompts that describe the objective classes, and then compare them with image features. Despite impressive performance on few-shot tasks, these works strongly rely on the semantic alignment between image encoder and text encoder, and thus are specific to vision-language pre-trained models [45]. In this work, we design a new form of prompts for vision data, named position prompts, to adapt the pre-trained vision transformers to few-shot image classification tasks. They can prompt explicit valuable information, and also be usable to different pre-trained vision transformers, without limited structures or pre-training ways.
• We conduct extensive experiments and verify that our method can effectively improve the performance of full or parameter-efficient \cite{24,34} fine-tuning methods on few-shot tasks.

2 Related work

**Few-shot classification.** The traditional paradigm of few-shot classification is the meta-learning framework \cite{22,67}, which learns the task-shared inductive bias \cite{13,32,63} on the training tasks constructed from a base dataset. When the base dataset is unlabeled or in the different domain to novel tasks, we also need to solve new challenges, unsupervised \cite{23,28,64} or cross-domain \cite{58,62,14} few-shot classification. Some works \cite{56,46} find the well-generalizing representation is actually the key factor of few-shot ability, while the base dataset is typically not large enough to obtain it. In fact, adapting self-supervised pre-trained models \cite{6,76,65} to few-shot tasks can address all these concerns simultaneously: the training data is unlabeled and thus can be sufficiently large, and the novel tasks can come from different target domains. The pre-trained large models have far more generalizing representation and thus can obtain significantly better performance than the traditional setting, as shown in Appendix \cite{B}. Besides, it is more friendly to privacy data, and we only need the pre-trained models instead of pre-training data. It also conforms to the learning paradigm of humans whose few-shot ability comes from learning on massive signals since birth, and has better realistic value. An exploratory work \cite{9} uses the ensemble of frozen pre-trained models and a simple classifier, but without further adaptation to novel tasks. CLIP \cite{45} introduces the vision-language pre-trained models, and some works \cite{78,77,7} use the text prompts to solve few-shot image classification based on them. These methods are limited to specific pre-trained models, and typically require the test classes to be seen in the training corpus \cite{40}. Humans actually can recognize unseen objects based on only reference images, without knowing their class names. Our method is usable for both single and multi-modal pre-trained models. Similar to our work, some existing models \cite{69,79} also propose to highlight the key local features of each input for few-shot classification. However, they all design parametric modules and meta-learn them using the base dataset. It is not suitable for our new setting, since the parametric modules can not been learned effectively using only few support samples. Our method instead introduces no new parametric modules and does not need meta-training.

**Prompt engineering.** Designing prompts \cite{35} for adapting the pre-trained models to few-shot tasks can originate from GPT-3 \cite{5}, which uses language prompts for in-context learning. Since the manual prompts require domain expertise and are likely to be sub-optimal even with significant effort, many works \cite{15,72,29} focus on automating prompt generation to fine-tune pre-trained language models. The language prompts can also be used for few-shot image classification \cite{78,77,7} based on the core transformation of each transformer layer, and is suitable for both single and multi-modal pre-trained models. But VT performs unsatisfactorily on few-shot tasks, even worse than simple plug-and-play solvers \cite{4,32} with frozen backbone \cite{Table[1]}. Its prompts do not prompt any explicit information and still follow the form of language prompts. In this work, we propose position prompts for vision data to explicitly prompt the key positions of input images to improve fine-tuning performance.

**Deep explanation method.** In order to build trust in intelligent systems, deep explanation methods \cite{16} calculate the local relevancy of the input with model predictions, and can be used to locate the key parts. Many classic works have been designed for CNNs, such as gradient based methods \cite{51,50,55,53} and relevance propagation based methods \cite{5,28,25,17}. For Transformers, these methods can not take advantage of the core attention information. We can use the raw attention scores to explain model predictions, but this ignores the attention components of other layers. Rollout \cite{1} thus rolls out the attention scores of multiple layers to capture the propagation of information from input, but it is not class-specific. Inspired by Rollout, we also integrate the attention information from multiple layers, and further introduce gradient information to ensure class-specific explanation.

3 Model

3.1 Problem formulation

In this work, we aim to adapt the pre-trained vision transformers to few-shot image classification tasks. These backbones can be columnar or pyramidal architectures, e.g., ViT \cite{11} or Swin \cite{36}, and
are pre-trained on massive data. In a few-shot classification task, we are typically given a support set \( T_s \) as reference and a query set \( T_q \) for test. The support set \( T_s = \{(x_i^s, y_i^s)\}_{i=1}^C \) contains \( C \) different classes with \( K \) labeled samples in each class, which is denoted as ‘\( C \)-way \( K \)-shot’. \( K \) is typically very small, e.g., 1 or 5, which makes it a challenging task, especially for the data-hungry vision transformers. The query set \( T_q = \{(x_j^q, y_j^q)\}_{j=1}^M \) contains the samples belong to the same classes with the support set, and we need to classify them correctly. Adapting a pre-trained model \( f_θ \) to a few-shot task \( T = \{T_s, T_q\} \) can be formulated as

\[
ψ = A(f_θ, T_s)
\]

where the task solver \( A \) is designed to obtain the solution model \( ψ \) of task \( T \). For example, the task solver \( A \) can be linear probing \([9]\) that combines the frozen model \( f_θ \) with a linear classifier, or the parameter-efficient fine-tuning methods \([34, 24, 27]\) that only tune partial existing or newly-added parameters for easy deployment, or using language prompts to generate classifier weights when \( f_θ \) is a vision-language pre-trained model \([28, 77, 7]\). The performance of solution \( ψ \) is evaluated on the query set \( T_q \) with accuracy and so on. In this work, we aim to design better task solver \( A \).

### 3.2 Multi-head self-attention

The Multi-head Self-Attention (MSA) modules are the core components of various vision transformers. The input image is split into multiple patches, which are then projected to patch tokens. The learnable [CLS] token is appended to these tokens for final classification if needed. Let the input of a MSA module be \( Z_{in} \in \mathbb{R}^{N \times d} \), containing \( N \) tokens with dimension \( d \). Here \( Z_{in} \) may contain all input tokens for global attention \([11]\), or partial ones for window attention \([36]\). MSA uses \( H \) attention heads to jointly attend to the information from different representation sub-spaces. Within \( h \)-th attention head, we first project \( Z_{in} \) to Query \( Q_h \), Key \( K_h \) and Value \( V_h \) as

\[
Q_h = Z_{in}W_q^h \in \mathbb{R}^{N \times d_k} \quad K_h = Z_{in}W_k^h \in \mathbb{R}^{N \times d_k} \quad V_h = Z_{in}W_v^h \in \mathbb{R}^{N \times d_v}
\]

where \( W_q^h, W_k^h \) and \( W_v^h \) are projection matrices. The self-attention matrix \( A_h \) is calculated as

\[
A_h = \text{softmax}(Q_hK_h^T / \sqrt{d_k}) \in \mathbb{R}^{N \times N}
\]

The matrix \( A_h \) reflects the attention of different tokens to each other, and the large score \( A_h^{ij} \) indicates the \( i \)-th token strongly attends to the \( j \)-th token. The output feature of the MSA module is

\[
\text{MSA}(Z_{in}) = \text{Concat}(\{A_h^iV_h\}_{h=1}^H)W_o + Z_{in}
\]

where \( \text{Concat}(\cdot) \) denotes the concatenation operation, \( W_o \) is a projection matrix. Obviously, the MSA module learns the increment of each token and introduces the information from other tokens, whose quantity is controlled by the attention matrices \( \{A_h^i\}_{h=1}^H \).

### 3.3 Analysis and motivation

We first examine various designs of the task solver \( A \): 1) simple machine learning classifiers, e.g., Nearest Neighbor classifier, Ridge Regression and Support Vector Machines; 2) plug-and-play inductive meta-solvers (e.g., ProtoNet \([52]\), R2D2 \([4]\) and MetaOptNet \([52]\) and linear probing; 3) full or parameter-efficient fine-tuning, e.g., VPT \([26]\), LoRA \([24]\) and SSF \([34]\). The results are provided in Table 1 where we use the ViT-B/16 pre-trained on ImageNet-1K \([49]\) with DINO \([6]\) as \( f_θ \) and conduct on CUB \([61]\), Cars \([30]\), Places \([78]\) and Plantae \([59]\) datasets. For each fine-tuning method, we use the best linear meta-solver, MetaOptNet, to obtain the initialization of the classifier weights. Otherwise, these fine-tuning methods perform even worse than MetaOptNet on few-shot tasks (Figure 2).

As shown in Table 1, the fine-tuning methods can achieve better performance than the backbone-frozen methods in most cases. But they sometimes cannot significantly improve performance beyond the classifier initialization (i.e., MetaOptNet) or even perform worse, especially on 1-shot tasks. In fact, the main challenges of few-shot tasks are: 1) the classes in the novel tasks could be unseen
3.4 Position prompts

We need to locate the positions of key entities in each support image as prompts for fine-tuning, i.e., the most class-related key patches for the current few-shot task. Manual labeling may provide accurate positions, but requires domain expertise and labor costs. We aim to design an automatic method, which should 1) be class-specific for locating class-related patches, and 2) make full use of the attention information, the core content of Transformers. Deep explanation methods [15] calculate the local relevancy of the input with model predictions, and can be used for highlighting the key patches for classification. For Transformers, Rollout [1] finds that using the attention of top layer for explanation ignores most of the attention components, thus miscalculating the importance of the patches, so we integrate the attention information from multiple layers like Rollout. Furthermore, only using the attention information cannot achieve class-specific explanation for some pre-trained models (Figure 3), so we further introduce the gradient information as an auxiliary.

For a columnar architecture [11] with $L$ layers, let the input feature map of $l$-th layer be $Z_{in}^l \in \mathbb{R}^{N \times d}$ and the attention matrices be $(A_{bh}^l)^H_{h=1}$. With the well-initialized classification head above, for a support sample-label pair $(x^s, y^s)$, we first compute the gradient of the prediction score for class $y^s$, $p_{y^s}$, with respect to feature map $Z_{in}^l$, as $\nabla_l = \partial p_{y^s}/\partial Z_{in}^l \in \mathbb{R}^{N \times d}, l = 1, \ldots, L$. The gradient computed for single sample contains many noises, which could be amplified layer by layer along the back-propagation path. To this end, we only use the gradient to the input feature of the top layer, i.e., $\nabla_L$, and reserve its first principle component for denoise. The obtained gradient term is

$$G = \nabla_L \cdot V_1 \in \mathbb{R}^{N \times 1}, \quad U, S, V = \text{svd}(\nabla_L)$$

where $\text{svd}(\cdot)$ denotes Singular Value Decomposition (SVD) operation with $\nabla_L = USV^T$, and $V_1$ is the first column vector of matrix $V$, corresponding to the largest singular value of $\nabla_L$. The gradient term $G$ can provide class-specific information. As shown in Equation 3, the residual connection plays a important role in propagating information from inputs to model predictions and can preserve the position information during forward propagation. To this end, we use an identity matrix $I$ to augment the attention matrix. Overall, the final attention graph of $l$-th layer is

$$\hat{A}^l = \text{norm} (I + A^l + \lambda \cdot G^T) \in \mathbb{R}^{N \times N}$$

where $\text{norm}(\cdot)$ means to re-normalize the input so that the row sum is equal to 1, $\lambda$ can control the proportion of attention and gradient information, and we average the attention matrix over all heads, i.e., $A^l = \frac{1}{L} \sum_{h=1}^{H} A_{bh}^l$. We assume the attentions are combined linearly and consider the path along the pairwise attention graph [1]. The patch importance scores are calculated as

$$s = \text{mean} (\hat{A}^1, \hat{A}^2, \ldots, \hat{A}^L) \in \mathbb{R}^N$$

where $(\cdot)$ denotes matrix multiplication, $\text{mean}(\cdot)$ denotes to average the row vectors of a matrix. We use the positions of the top $P$ patches with the highest importance scores as prompts.

For a pyramidal architecture [36, 70], since the downsampling operation destroys the patch position correspondence between layers, so we directly use the attention scores average on patches as the importance scores and introduce gradient information to achieve class-specific if necessary.
When we use a log-bilinear function as critic, i.e., $f$ where the critic with $\{I_n\}$ we consider the mutual information $I_{\theta(x)}$, $\theta(x)$ denotes tunable full or partial parameters of the model, $(x, y) \in \mathcal{T}$, we obtain the Query $Q = \{q_n\}_{n=1}^N = Z^L_m W_q$ and Key $K = \{k_n\}_{n=1}^N = Z^L_m W_k$ for calculating the attention logits $(\alpha, \beta)$, $\alpha$ is a loss coefficient, $\Omega$ is the index set of the key patches in $x^m$, $\tau$ is a temperature coefficient. We calculate the objective term $R$ only at the top layer to operate the features with more refine semantic. Specially, for the pyramidal architectures, $R$ is applied to the top layer of each stage. This loss enables the model to automatically identify the key patches and focus on them during fine-tuning, and this ability then can generalize to the query samples. The position prompts are fixed during fine-tuning since updating them only obtains trivial improvement but introduces obvious cost.

### Theoretical analysis.

The key patches contain the most class-related information of the input images, and the model should focus its attention on them to obtain discriminative representation in the current task, not only for the support samples, but also for the query ones. Since the position prompts (or key patches) of the query samples are unknown without label information, unlike the existing prompts [35], our position prompts can not be used in the input or middle phase of forward inference. To this end, we use the position prompts as the prediction target for attention, and optimize the attention enhancement.

<table>
<thead>
<tr>
<th>Model</th>
<th>CUB 1-shot</th>
<th>CUB 5-shot</th>
<th>Cars 1-shot</th>
<th>Cars 5-shot</th>
<th>Places 1-shot</th>
<th>Places 5-shot</th>
<th>Plantae 1-shot</th>
<th>Plantae 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>52.8</td>
<td>73.0</td>
<td>20.9</td>
<td>36.2</td>
<td>49.5</td>
<td>64.3</td>
<td>35.0</td>
<td>54.3</td>
</tr>
<tr>
<td>RR</td>
<td>56.6</td>
<td>84.0</td>
<td>24.0</td>
<td>56.2</td>
<td>49.9</td>
<td>68.7</td>
<td>36.8</td>
<td>62.0</td>
</tr>
<tr>
<td>SVM</td>
<td>52.8</td>
<td>78.7</td>
<td>20.9</td>
<td>37.3</td>
<td>49.5</td>
<td>70.8</td>
<td>35.0</td>
<td>57.7</td>
</tr>
<tr>
<td>ProtoNet [52]</td>
<td>52.8</td>
<td>79.8</td>
<td>20.9</td>
<td>39.3</td>
<td>49.5</td>
<td>71.2</td>
<td>35.0</td>
<td>57.7</td>
</tr>
<tr>
<td>R2D2 [4]</td>
<td>56.7</td>
<td>84.3</td>
<td>23.8</td>
<td>56.6</td>
<td>49.7</td>
<td>68.9</td>
<td>36.8</td>
<td>62.3</td>
</tr>
<tr>
<td>MetaOptNet [22]</td>
<td>57.0</td>
<td>85.1</td>
<td>24.1</td>
<td>57.9</td>
<td>50.0</td>
<td>71.1</td>
<td>36.9</td>
<td>63.7</td>
</tr>
<tr>
<td>Linear Probing</td>
<td>41.9</td>
<td>78.2</td>
<td>18.3</td>
<td>47.2</td>
<td>41.0</td>
<td>65.8</td>
<td>27.2</td>
<td>56.6</td>
</tr>
<tr>
<td>VFT [20]</td>
<td>52.9</td>
<td>81.1</td>
<td>23.3</td>
<td>54.5</td>
<td>48.0</td>
<td>69.6</td>
<td>33.9</td>
<td>60.2</td>
</tr>
<tr>
<td>FT</td>
<td>58.0</td>
<td>88.1</td>
<td>24.1</td>
<td>66.9</td>
<td>50.3</td>
<td>72.1</td>
<td>37.0</td>
<td>66.2</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>57.9</td>
<td>88.2</td>
<td>23.3</td>
<td>64.3</td>
<td>49.9</td>
<td>71.3</td>
<td>37.1</td>
<td>65.7</td>
</tr>
<tr>
<td>SSF [44]</td>
<td>57.8</td>
<td>88.4</td>
<td>23.8</td>
<td>62.3</td>
<td>50.2</td>
<td>73.4</td>
<td>37.2</td>
<td>66.0</td>
</tr>
<tr>
<td>FT + FORT</td>
<td>59.5 (1.5)</td>
<td>89.2 (1.1)</td>
<td>25.5 (1.4)</td>
<td>68.0 (1.1)</td>
<td>51.1 (0.8)</td>
<td>72.9 (0.8)</td>
<td>38.7 (1.7)</td>
<td>67.2 (1.0)</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>62.5 (4.6)</td>
<td>89.5 (1.3)</td>
<td>26.8 (3.5)</td>
<td>65.7 (1.4)</td>
<td>50.8 (0.9)</td>
<td>72.4 (1.1)</td>
<td>38.5 (1.4)</td>
<td>66.9 (1.2)</td>
</tr>
<tr>
<td>SSF + FORT</td>
<td>62.3 (4.5)</td>
<td>89.6 (1.2)</td>
<td>26.5 (2.7)</td>
<td>64.2 (1.9)</td>
<td>51.3 (1.1)</td>
<td>74.4 (1.0)</td>
<td>39.0 (1.8)</td>
<td>67.5 (1.5)</td>
</tr>
</tbody>
</table>

3.5 Attention enhancement

The key patches contain the most class-related information of the input images, and the model should focus its attention on them to obtain discriminative representation in the current task, not only for the support samples, but also for the query ones. Since the position prompts (or key patches) of the query samples are unknown without label information, unlike the existing prompts [35], our position prompts can not be used in the input or middle phase of forward inference. To this end, we use the position prompts as the prediction target for attention, and optimize the cross-entropy loss between their many-hot presentation and attention logits during fine-tuning. As shown in Equation [2] and [3] given the input tokens (e.g., [CLS] token and patch tokens) of the top layer $Z^L_m = \{z_n\}_{n=1}^N \in \mathbb{R}^{N \times d}$, we obtain the Query $Q = \{q_n\}_{n=1}^N = Z^L_m W_q$ and Key $K = \{k_n\}_{n=1}^N = Z^L_m W_k$ for calculating the attention logits $(\alpha, \beta)$. The fine-tuning objective is

$$
\min_{\theta} \sum_{(x^a, y^a) \in \mathcal{T}} \left[ ce(f_{\theta}(x^a), y^a) - \alpha \cdot R \right], \quad R = \frac{1}{N} \sum_{n=1}^N \sum_{i \in \Omega} \ln \frac{\exp \left( q_n k_n^T / \tau \right)}{\sum_{m=1}^N \exp \left( q_m k_m^T / \tau \right)}
$$

where $\theta$ denotes tunable full or partial parameters of the model, $ce(\cdot, \cdot)$ denotes cross-entropy classification loss, $\alpha$ is a loss coefficient, $\Omega$ is the index set of the key patches in $x^a$, $\tau$ is a temperature coefficient. We calculate the objective term $R$ only at the top layer to operate the features with more refine semantic. Specially, for the pyramidal architectures, $R$ is applied to the top layer of each stage. This loss enables the model to automatically identify the key patches and focus on them during fine-tuning, and this ability then can generalize to the query samples. The position prompts are fixed during fine-tuning since updating them only obtains trivial improvement but introduces obvious cost.

**Theoretical analysis.** The key patches contain the most class-related information of the input and we want to increase their information in the output feature. We first prove that the objective term $R$ can increase the information from the key patches in the input tokens $Z^L_m = \{z_n\}_{n=1}^N$. Concretely, we consider the mutual information $I(z, z_\Omega)$, where the random variable $z$ presents the input token with $\{z_n\}_{n=1}^N$ as its samples, the random variable $z_\Omega$ presents the key patch with $\{z_i\}_{i \in \Omega}$ as its samples. The InfoNCE lower bound estimate [42, 43] of $I(z, z_\Omega)$ thus is

$$
\hat{I}_{\text{InfoNCE}}(z, z_\Omega) = \frac{1}{N} \sum_{n=1}^N \sum_{i \in \Omega} \ln \frac{f(z_n, z_i)}{\sum_{m=1}^N f(z_n, z_m)} \leq I(z, z_\Omega)
$$

where the critic $f(\cdot, \cdot)$ is used to model the density ratio and can be any positive real score function. When we use a log-bilinear function as critic, i.e.,

$$
f(z_n, z_m) = \exp (z_n W_q W_k^T z_m^T / \tau) = \exp (q_n k_m^T / \tau), \quad \forall n, m \in \{1, \ldots, N\}
$$

2For index set $\Omega$, it is a vector of length $|\Omega|$, with the values of the positions in $\Omega$ be 1 and other ones be 0.
Table 2: Few-shot classification accuracy (%) on 20-way 1-shot / 5-shot tasks. We use the Swin-T/7 pre-trained on ImageNet-1k with iBOT for backbone initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>CUB</th>
<th>Cars</th>
<th>Places</th>
<th>Plantae</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>MetaOptNet</td>
<td>49.1</td>
<td>81.1</td>
<td>21.0</td>
<td>51.1</td>
</tr>
<tr>
<td>FT</td>
<td>52.7</td>
<td>87.1</td>
<td>20.9</td>
<td>59.7</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>53.6</td>
<td>87.4</td>
<td>20.4</td>
<td>57.9</td>
</tr>
<tr>
<td>SSF [34]</td>
<td>54.9</td>
<td>87.6</td>
<td>21.4</td>
<td>57.2</td>
</tr>
<tr>
<td>FT + FORT</td>
<td>59.1 (6.4)</td>
<td>88.3 (1.2)</td>
<td>22.3 (1.4)</td>
<td>61.6 (1.9)</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>59.9 (6.3)</td>
<td>88.5 (1.1)</td>
<td>21.9 (1.5)</td>
<td>59.4 (1.5)</td>
</tr>
<tr>
<td>SSF + FORT</td>
<td>60.1 (5.2)</td>
<td>88.9 (1.3)</td>
<td>23.4 (2.0)</td>
<td>58.8 (1.6)</td>
</tr>
</tbody>
</table>

Table 3: Few-shot classification accuracy (%) on 20-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained on WIT [45] with CLIP for backbone initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>CUB</th>
<th>Cars</th>
<th>Places</th>
<th>Plantae</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>MetaOptNet</td>
<td>68.0</td>
<td>88.7</td>
<td>67.6</td>
<td>90.7</td>
</tr>
<tr>
<td>zero-shot</td>
<td>84.1</td>
<td>84.1</td>
<td>88.0</td>
<td>88.0</td>
</tr>
<tr>
<td>CoOp [78]</td>
<td>84.4</td>
<td>90.4</td>
<td>91.3</td>
<td>94.6</td>
</tr>
<tr>
<td>Tip-Adapter-F [73]</td>
<td>86.9</td>
<td>92.0</td>
<td>92.2</td>
<td>95.3</td>
</tr>
<tr>
<td>PLOT++</td>
<td>87.4</td>
<td>92.0</td>
<td>92.2</td>
<td>95.5</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>86.3</td>
<td>92.6</td>
<td>92.3</td>
<td>95.8</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>87.8 (1.5)</td>
<td>93.8 (1.2)</td>
<td>93.6 (1.3)</td>
<td>97.0 (1.2)</td>
</tr>
</tbody>
</table>

4 Experiments

4.1 Few-shot classification

Settings. The experiments are conducted on the few-shot benchmark from [58] (i.e., CUB [61], Cars [30], Places [75] and Plantae [59] datasets), and two other fine-gained datasets: Aircraft [39] and Pets [43]. We use the pre-trained ViT-B/16 released by DINO [6] and CLIP [45] and Swin-T/7 released by iBOT [76] for examining different architectures and pre-training ways. Here we only consider the contrastive pre-trained models [6] since their representations have better linear separability than the generative pre-trained models [18, 68] and are more suitable for few-shot classification. We evaluate both the backbone-frozen methods, e.g., machine learning classifiers (Nearest Neighbor classifier, Ridge Regression and Support Vector Machine), meta-solvers (ProtoNet [52], R2D2 [4], and others).
Figure 3: The attention map of [CLS] token in the pre-trained ViT-B/16 from DINO and CLIP, and the position prompts (red patches) obtained using the attention w/ or w/o gradient information.

Table 4: Few-shot classification accuracy (%) on 20-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained on ImageNet-1K with DINO for backbone initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>Aircraft 1-shot</th>
<th>Aircraft 5-shot</th>
<th>Pets 1-shot</th>
<th>Pets 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaOptNet</td>
<td>25.5</td>
<td>55.5</td>
<td>71.9</td>
<td>89.6</td>
</tr>
<tr>
<td>FT</td>
<td>27.1</td>
<td>64.6</td>
<td>71.6</td>
<td>89.0</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>26.0</td>
<td>62.7</td>
<td>72.7</td>
<td>89.8</td>
</tr>
<tr>
<td>SSF [34]</td>
<td>26.5</td>
<td>61.6</td>
<td>72.7</td>
<td>90.0</td>
</tr>
<tr>
<td>FT + FORT</td>
<td>28.7</td>
<td>66.0</td>
<td>72.6</td>
<td>89.9</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>31.3</td>
<td>63.9</td>
<td>74.8</td>
<td>90.9</td>
</tr>
<tr>
<td>SSF + FORT</td>
<td>30.9</td>
<td>63.1</td>
<td>74.4</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Table 5: Attention-weighted summation of importance scores ($\times 10^{-2}$) on 20-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained on ImageNet-1K with DINO for initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-shot support</th>
<th>1-shot query</th>
<th>5-shot support</th>
<th>5-shot query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig Attn</td>
<td>0.72</td>
<td>0.72</td>
<td>0.72</td>
<td>0.72</td>
</tr>
<tr>
<td>FT</td>
<td>0.74</td>
<td>0.73</td>
<td>0.74</td>
<td>0.75</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>0.71</td>
<td>0.71</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>SSF [34]</td>
<td>0.74</td>
<td>0.73</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>FT + FORT</td>
<td>0.80</td>
<td>0.77</td>
<td>0.90</td>
<td>0.91</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>1.00</td>
<td>0.97</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>SSF + FORT</td>
<td>1.06</td>
<td>1.03</td>
<td>1.04</td>
<td>1.04</td>
</tr>
</tbody>
</table>

Results. The results are provided in Table 1, 2, 3, and 4, where the 95% confidence interval is ±0.3 (1-shot) or ±0.2 (5-shot) and the number in parentheses is the increment of accuracy beyond the base fine-tuning methods. As shown in Table 1, 2, and 3 with classifier initialization, the fine-tuning methods typically can achieve better performance than the model-frozen methods, since the backbone is further adapted to task distribution. However, fine-tuning sometimes struggles to achieve non-trivial improvement beyond the classifier initialization (i.e., MetaOptNet) or even performs worse, especially on the 1-shot tasks. Our FORT can help the fine-tuning methods to obtain significant improvement on both ViT and Swin. Further, FORT typically achieves more improvement on 1-shot tasks than that on 5-shot tasks. This is intuitive, since with more support samples the model can attend to the frequently-occurring entities more in each class, which are typically the key entities. FORT is more suitable for solving the hardest 1-shot tasks. As shown in Table 3 on the vision-language pre-trained model, fine-tuning with simple classifier head (i.e., LoRA) achieves comparable or better performance than the specifically-designed methods: CoOp, Tip-Adapter-F and PLOT++. In fact, Tip-Adapter-F uses the ensemble of a linear classifier head and the similarity-based label propagation module. PLOT++ introduces multiple learnable language prompts and visual prompts for obtaining multiple classifier weights and image embeddings, and then calculates their optimal-transport distance for classification, which has huge computational burden. Our FORT can further improve the performance beyond the fine-tuning methods. Finally, FORT is compatible with the parameter-efficient fine-tuning even better than with full fine-tuning, which is convenient for industrial deployment.

4.2 Ablation studies
Our method. The original model can not focus its attention on the key patches, nor can the
SSF to avoid over-fitting. The full attention maps are provided in appendix and we can observe the
SSF to more parts besides them. This is intuitive, since the supervision signals for the image encoder in
CLIP are the sentence descriptions, which typically involve both the foreground objects and scene,
e.g., `a bird flies over the sea`. Importantly, the key entities typically can not get the highest attention
scores in CLIP pre-trained model, which makes it difficult to accurately locate them only based on the
attention information. Figure 3 ‘Attention+Grad Prompts’ shows the position prompts obtained using both
the attention and gradient information, where $\lambda = 1$ for DINO and $\lambda = 50$ for CLIP. We use MetaOptNet to calculate
the classifier weight for DINO, and use language prompts for CLIP. The gradient information helps
to accurately locate the key entities in CLIP, and also improve that in DINO.

**Attention enhancement.** We conduct qualitative and quantitative experiments to verify that our
method can enhance the attention of the model to the key patches. Qualitatively, Figure 4 visualizes
the patches with top highest attention scores of [CLS] token in ViT-B/16. Concretely, when the
attention scores of all patches are $a_1, \ldots, a_N$ in descending order, we choose the patches \{1, \ldots, n\},
which satisfy $S_{n-1} < 0.95 \leq S_n$ with $S_n = \sum_{i=1}^n a_i$ and $S_N = 1$. We use the pre-trained model
from DINO for backbone initialization and fine-tune on random 1-shot tasks. We provide the results
of original attention (`Orig Attn`) and the attention after fine-tuning w/ (`Our Attn`) or w/o (`FT
Attn`) our method. The original model can not focus its attention on the key patches, nor can the
full or parameter-efficient fine-tuning achieve this property. Our method can prompt the model
the positions of key patches and help the model to focus most attention (95%) on them, and this
ability can generalize from the support set to query samples. This phenomenon is not obvious on
full fine-tuning, and the reason may be it uses smaller learning rate and less epochs than LoRA and
SSF to avoid over-fitting. The full attention maps are provided in appendix and we can observe the
same phenomenon. Further, we examine the attention-weighted summation of importance scores,
i.e., $\sum_{i=1}^N s_i \cdot a_i^0$, where $s$ is defined in Equation 7 and $a_i^0$ is its position-aligned attention scores.
The larger the value, the more focused on the key patches the attention. Table 5 provides the results
averaged on CUB, Cars, Places and Plantae datasets with 2000 tasks each setting. The fine-tuning
methods obtain the similar scores with original attention, while our FORT achieves non-trivial
increase. Consistently, the increment on full fine-tuning is smaller than that on LoRA and SSF. For
standard fine-tuning, the increment on 5-shot tasks is larger than that on 1-shot tasks. In fact, with
more support samples, the model can attend to the frequently-occurring key entities in each class.

**Importance calculation.** In Section 3.4 we combine both gradient and attention information for
calculating the importance score of each patch. Here we ablate the different information item and
compare their performance. We follow the experimental settings in Section 4.1 and use the per-trained
models from DINO or CLIP for initialization. The results are provided in Table 6 where `FORT
(attn)` denotes using only attention information (i.e., setting $\lambda = 0$ in Equation 6), `FORT (grad)`
denotes using only gradient term $G$. `FORT (attn+grad)` denotes using both attention and gradient
information, that is our final method. Here we pick the best hyper-parameters for each method, even
though they may differ from each other. For DINO pre-trained models, the attention information

![Figure 4: Visualization of the patches (white parts) with top highest attention scores and covering about 95% attention of [CLS] token in ViT-B/16. We use DINO pre-trained model for initialization.](image)
Table 6: Few-shot classification accuracy (%) on 20-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained by DINO or CLIP for backbone initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>DINO</th>
<th></th>
<th>ClIP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CUB</td>
<td>Cars</td>
<td>CUB</td>
<td>Cars</td>
</tr>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>LoRA</td>
<td>57.9</td>
<td>88.2</td>
<td>23.3</td>
<td>64.3</td>
</tr>
<tr>
<td>+ FORT (attn)</td>
<td>62.1</td>
<td>89.0</td>
<td>26.3</td>
<td>65.0</td>
</tr>
<tr>
<td>+ FORT (grad)</td>
<td>60.0</td>
<td>88.8</td>
<td>25.2</td>
<td>64.7</td>
</tr>
<tr>
<td>+ FORT (attn+grad)</td>
<td>62.5</td>
<td>89.5</td>
<td>26.8</td>
<td>65.7</td>
</tr>
</tbody>
</table>

Figure 5: Ablation studies on hyper-parameter $\lambda$, $\tau$, and $\alpha$. We examine the absolute increment of accuracy beyond the base fine-tuning methods averaged on CUB, Cars, Places and Plantae datasets.

Hyper-parameters. We conduct ablation studies on the hyper-parameters from Equation 6 and 8: $\lambda$, $\tau$, and $\alpha$, and examine the absolute increment of accuracy beyond the base fine-tuning methods averaged on CUB, Cars, Places and Plantae datasets. For each hyper-parameter, we set the other ones to its corresponding best values. For $\lambda$, we use LoRA as the base fine-tuning method and evaluate different pre-trained models, i.e., ViT-B/16 from DINO and CLIP. For $\tau$ and $\alpha$, we use the ViT-B/16 from DINO and evaluate different base fine-tuning methods, i.e., full fine-tuning, LoRA, and SSF. Figure 5 (a) shows that using DINO pre-trained model is suitable for small $\lambda$ since its attention is enough for locating key patches, while using CLIP pre-trained model needs large $\lambda$ to utilize gradient information. Figure 5 (b) shows that our method is generally robust to the temperature coefficient $\tau$, and the performance on 1-shot tasks is relatively more sensitive than that on 5-shot tasks. Figure 5 (c) shows that the loss coefficient $\alpha$ is an important hyper-parameter and needs to be carefully chosen, and the performance on 5-shot tasks is more sensitive than that on 1-shot tasks. In fact, if the coefficient $\alpha$ is too large, the attention enhancement loss could relatively inhibit the optimization of the classification loss (e.g., skew gradient direction), which is harmful to fitting the input-label distribution. The more training samples, the more important it is to fit input-label distribution, so the 5-shot tasks prefer smaller coefficient $\alpha$ than 1-shot tasks.

5 Conclusion and limitations

In this work, we aim to adapt the pre-trained vision transformers to few-shot image classification tasks. It is nontrivial, since even with well classifier initialization, the standard fine-tuning methods sometimes can not obtain significant improvement beyond the initialization, especially on 1-shot tasks. To this end, we propose a new prompt for vision data, position prompts, to explicitly prompt the model where are the key entities and focus on them during fine-tuning. Our method can effectively improve the performance of full or parameter-efficient fine-tuning on few-shot tasks. The limitations of our methods are: 1) our method is suitable for multi-head attention module, and unusable for other vision backbones without attention, e.g., ResNet [20] or MLP-Mixer [57], 2) for some professional fields, e.g., medical imaging, we may still need experts to manually label the position prompts.
References


[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16000–16009, 2022.


A Derivation of Monte-Carlo estimate

In Section 3.5, we claim that if \( p(z_n|z_m) \propto \exp (z_n W_q W_k^T z_m^T / \tau) \), \( n, m \in \{1, \ldots, N\} \), \( R^U \) and \( R^A \) have constant bias with the Monte-Carlo estimate of \( H(z) \) and \( -H(z|z_\Omega) \) respectively, where

\[
R^U = -\frac{1}{N} \sum_{n=1}^{N} \ln (\sum_{m=1}^{N} \exp (q_n k_m^T / \tau)), \quad R^A = \frac{1}{N \cdot |\Omega|} \sum_{n=1}^{N} \sum_{t \in \Omega} q_n k_t^T / \tau
\]  
(13)

We define

\[
p(z|\hat{z}) = \frac{\exp (z W_q W_k^T \hat{z}^T / \tau)}{Z(\hat{z})}, \quad p(z|z_\Omega) = \frac{\exp (z W_q W_k^T z_\Omega^T / \tau)}{Z(z_\Omega)}
\]  
(14)

where the random variable \( \hat{z} \) follows the same distribution as \( z \) and also has \( \{z_m\}_{m=1}^{N} \) as its some samples, and \( Z(\cdot) \) is the partition function for normalization. Here we assume that the energy state remains the same for different \( \hat{z} \) or \( z_\Omega \), i.e., \( Z(\cdot) \) keeps a constant \( Z \). The detailed derivations are

\[
H(z) = -E_{p(z)} \ln p(z) = -E_{p(z)} \ln E_{p(\hat{z})} p(z|\hat{z})
\]

\[
\approx -\frac{1}{N} \sum_{n=1}^{N} \ln E_{p(\hat{z})} p(z_n|\hat{z})
\]

\[
\approx -\frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{N} \sum_{m=1}^{N} p(z_n|z_m) \right)
\]

\[
\approx -\frac{1}{N} \sum_{n=1}^{N} \ln \left( \frac{1}{N} \sum_{m=1}^{N} \exp (q_n k_m^T / \tau) + \ln N Z \right)
\]

\[
\approx R^U + \ln N Z
\]  
(15)

and

\[
-H(z|z_\Omega) = E_{p(z|z_\Omega)} \ln p(z|z_\Omega)
\]

\[
\approx \frac{1}{N \cdot |\Omega|} \sum_{n=1}^{N} \sum_{t \in \Omega} \ln p(z_n|z_t)
\]

\[
\approx \frac{1}{N \cdot |\Omega|} \sum_{n=1}^{N} \sum_{t \in \Omega} q_n k_t^T / \tau - \ln Z
\]

\[
\approx R^A - \ln Z
\]  
(16)

From these results, the above claim is proved. Besides, we have

\[
I(z, z_\Omega) = H(z) - H(z|z_\Omega) \approx R^U + R^A + \ln N = \mathcal{R} + \ln N, \text{i.e., we obtain the Equation 11 again.}
\]

B More experiments

Fine-tuning details. The resolution of input images is 224 \( \times \) 224, and we do not use data augmentation following the standard setting. In fact, we find the base data augmentation (e.g., random resized cropping, flip or random erasing) can not obtain improvement on the chosen datasets of our

Table 7: Candidate values of hyper-parameters for fine-tuning.

<table>
<thead>
<tr>
<th>hyper-parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate</td>
<td>1e-5, 3e-5, 5e-5, 8e-5, 1e-4, 3e-4, 5e-4, 8e-4, 1e-3, 3e-3, 5e-3, 8e-3, 1e-2</td>
</tr>
<tr>
<td>#epoch</td>
<td>5, 10, 15, 20</td>
</tr>
<tr>
<td>temperature ( \tau ) coefficient ( \alpha )</td>
<td>1, 5, 10, 15, 20, 5e-4, 8e-4, 1e-3, 3e-3, 5e-3, 8e-3, 1e-2, 3e-2, 5e-2, 8e-2, 1e-1, 3e-1, 5e-1, 8e-1, 1</td>
</tr>
</tbody>
</table>
experiments. We do not consider the pre-trained backbones from masked image modeling models for initialization, e.g., BEiT [76] or MAE [18], since these generative pre-training models cannot obtain highly linearly separable features and perform poorly on few-shot tasks. We set the rank to 4 for LoRA [24] and use 10 learnable prompts at each layer for VPT [26]. We mainly use AdamW [37] optimizer for fine-tuning, except for linear probing and VPT where we use SGD optimizer with weight decay of 1e-3. We set the batch size to 20, same as the number of classes, for both 1-shot and 5-shot tasks. We set $\lambda = 1$ for DINO pre-trained model and $\lambda = 50$ for CLIP pre-trained model for simplicity. The other hyper-parameters, including learning rate, number of fine-tuning epochs, temperature $\tau$ and coefficient $\alpha$, are changeable for different fine-tuning methods, with or without classifier initialization and different datasets. To this end, we only provide their candidate values in Table 7 and choose them according to the average accuracy on 50 randomly sampled few-shot tasks. We only use 50 tasks for quick hyper-parameter selection. Considering the confidence intervals on random 50 tasks are too large to be an accurate indication of performance, e.g., $\pm 1.7$, we fix the random seed so as to compare the performance of different hyper-parameters on the same 50 few-shot tasks. We first determine the learning rate and the number of epochs for the base fine-tuning methods, and then determine the temperature $\tau$ and the coefficient $\alpha$ for our method.

**Attention maps with sufficient training samples.** We follow the experimental settings of Figure 4 except fine-tuning on the 20-way 50-shot task from CUB and 20-way 200-shot task from Pets without attention enhancement. The attention maps after fine-tuning are shown in Figure 6 and Figure 7 as “FT Attn (sufficient)”. As we can see, fine-tuning with sufficient samples indeed focuses on the key entities more than with few samples (“FT Attn”) or original attention (“Orig Attn”), but is less focused than explicit attention enhancement (“Our Attn”). Note that this does not mean that few-shot fine-tuning with attention enhancement can obtain higher accuracy than sufficient-shot fine-tuning. In fact, sufficient training samples allow the models to fit the input-label distribution better, which is more important for classification than attention enhancement. But when we can not obtain sufficient samples, attention enhancement can reduce class-independent noise information in the final embedding as shown experimentally and theoretically, and thus is useful for classification.

**Performance on 5-way tasks.** In order to facilitate comparison with traditional few-shot learning setting and show the advantages of our new setting, we provide the results on 5-way 1-shot / 5-shot tasks in Table 8. The performance are far more better than traditional few-shot learning setting. On miniImageNet, we can achieve 94.2% accuracy for 1-shot tasks and 98.5% accuracy for 5-shot tasks, while the SOTA traditional model, FeLMi [48], only achieve 67.47% and 86.08% accuracy for 1-shot
Table 8: Few-shot classification accuracy (%) on 5-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained on ImageNet-1k with DINO for backbone initialization.

<table>
<thead>
<tr>
<th>Model</th>
<th>miniImageNet 1-shot</th>
<th>miniImageNet 5-shot</th>
<th>CUB 1-shot</th>
<th>CUB 5-shot</th>
<th>Cars 1-shot</th>
<th>Cars 5-shot</th>
<th>Pets 1-shot</th>
<th>Pets 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>93.1</td>
<td>97.8</td>
<td>80.1</td>
<td>94.7</td>
<td>43.3</td>
<td>74.9</td>
<td>88.7</td>
<td>95.9</td>
</tr>
<tr>
<td>LoRA + FORT</td>
<td>94.2</td>
<td>98.5</td>
<td>84.4</td>
<td>96.2</td>
<td>45.2</td>
<td>76.1</td>
<td>89.9</td>
<td>96.8</td>
</tr>
<tr>
<td>SSF</td>
<td>92.9</td>
<td>97.5</td>
<td>78.1</td>
<td>94.1</td>
<td>43.5</td>
<td>74.7</td>
<td>89.4</td>
<td>96.5</td>
</tr>
<tr>
<td>SSF + FORT</td>
<td>94.2</td>
<td>98.1</td>
<td>81.8</td>
<td>96.4</td>
<td>46.2</td>
<td>76.6</td>
<td>91.2</td>
<td>97.1</td>
</tr>
</tbody>
</table>

and 5-shot tasks respectively. Besides, although TDM [33] can achieve comparable performance on CUB with our method, it far lags behind our method on Pets, i.e., 55.8% vs 91.2% on 1-shot tasks and 72.5% vs 97.1% on 5-shot tasks. This means that TDM cannot obtain stable good performance while our method can achieve satisfactory performance on different datasets and are more general. In fact, the traditional setting which uses a base dataset is not inherently a true few-shot setting. If we want to obtain a few-shot model for CUB, we have to collect many labeled bird images for meta-training, which still needs massive labeling labor and seriously hinders the practical application. Importantly, for different domains (e.g., CUB, Cars), we have to label separate base datasets for them. Our new setting uses open-source large models pre-trained on massive unlabeled data and only uses the few support samples from different target domains for fine-tuning, which is a true few-shot setting and can handle different domains simultaneously. Using pre-trained large models is more effective and practically valuable than traditional setting. Finally, our method can further improve the fine-tuning performance of base fine-tuning methods, e.g., LoRA and SSF, even on 5-way tasks.

**Attention map of different fine-tuning methods.** In Figure 4, we choose the patches with top highest attention scores and covering about 95% attention for visualization. Here we provide the full attention map of original attention (‘Orig Attn’) and that after fine-tuning w/ (‘Our Attn’) or w/o (‘FT Attn’) our method in Figure 8. The settings are the same, i.e., we use the ViT-B/16 pre-trained on ImageNet-1K [49] with DINO for backbone initialization and fine-tune on random 1-shot tasks. Neither the original nor fine-tuned models can focus their attention on the key entities. Our method can obviously make the model to focus on the key entities and reduces the attention to the class-independent parts. This ability can generalize from the support set to query samples.

**Fine-tuning without classifier initialization.** In Figure 2, we compare MetaOptNet [32] with full or parameter-efficient fine-tuning methods, i.e., VPT [26], LoRA [24] and SSF [34], via average accuracy on CUB [61], Cars [30], Places [75] and Plantae [59] datasets. Here we provide detailed results in Table 9, where we use the ViT-B/16 pre-trained on ImageNet-1K [49] with DINO for backbone initialization and the 95% confidence interval is ±0.3 (1-shot) or ±0.2 (5-shot). As we can see, all fine-tuning methods perform worse than the backbone-frozen MetaOptNet on few-shot tasks. The extremely few labeled samples can not simultaneously learn a good classifier and adapt the backbone to the data distribution. In fact, the randomly initialized classifier back-propagates noisy gradients, which destroys the knowledge learned during pre-training and also fails to adapt new data distributions. Further, this phenomenon also shows it is very difficult to learn new parametric modules from scratch on few-shot tasks, e.g., classifier head, and we need to construct the well-
Table 9: Few-shot classification accuracy (%) on 20-way 1-shot / 5-shot tasks. We use the ViT-B/16 pre-trained on ImageNet-1K with DINO for backbone initialization. ‘FT’ denotes full fine-tuning. Here all fine-tuning methods use the randomly-initialized classifier.

<table>
<thead>
<tr>
<th>Model</th>
<th>CUB</th>
<th></th>
<th>Cars</th>
<th></th>
<th>Places</th>
<th></th>
<th>Plantae</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>MetaOptNet</td>
<td>57.0</td>
<td>85.1</td>
<td>24.1</td>
<td>57.9</td>
<td>50.0</td>
<td>71.1</td>
<td>36.9</td>
<td>63.7</td>
</tr>
<tr>
<td>VPT [26]</td>
<td>38.3</td>
<td>73.3</td>
<td>17.5</td>
<td>43.1</td>
<td>35.9</td>
<td>64.6</td>
<td>25.1</td>
<td>54.1</td>
</tr>
<tr>
<td>FT</td>
<td>25.0</td>
<td>66.7</td>
<td>15.3</td>
<td>42.2</td>
<td>23.6</td>
<td>55.0</td>
<td>22.0</td>
<td>51.4</td>
</tr>
<tr>
<td>LoRA [24]</td>
<td>55.3</td>
<td>83.6</td>
<td>22.5</td>
<td>54.7</td>
<td>48.4</td>
<td>68.2</td>
<td>34.2</td>
<td>62.1</td>
</tr>
<tr>
<td>SSF [34]</td>
<td>54.8</td>
<td>83.4</td>
<td>22.6</td>
<td>53.9</td>
<td>47.9</td>
<td>69.4</td>
<td>33.7</td>
<td>61.2</td>
</tr>
</tbody>
</table>

initialization for them. Considering MetaOptNet is a linear solver with excellent performance, we use it to calculate the classifier weights as initialization, which significantly improves the performance of various fine-tuning methods. Since the prompts in VPT are learned from scratch and we cannot obtain their initialization, VPT still underperforms MetaOptNet on few-shot tasks even with classifier initialization. We recommend using a linear solver (e.g., ProtoNet [52], R2D2 [4] or MetaOptNet [32]) to obtain a good initialization of the classifier as a necessary preemptive operation when adapting pre-trained models to few-shot classification tasks in the future.

Patch attention. In the main text, we only examine the attention of the [CLS] token at the last layer. Since our method calculates the attention enhancement loss on both the [CLS] and patch tokens, here we visualize the attention map of those tokens at the penultimate (11-th) and last (12-th) layer of ViT-B/16. We use the ViT-B/16 pre-trained on ImageNet-1K with DINO for backbone initialization, and fine-tune the model using LoRA or SSF and evaluate the effect of our method on them. We choose the [CLS] token and {14, 28, 42, 56, 70, 84, 98, 112, 126, 140, 154, 168, 182, 196}-th patches for visualization and the results are shown in Figure 9. After fine-tuning with LoRA or SSF, both the [CLS] and patch tokens still attend to the other parts besides the key entities, introducing class-independent noise information, and some patch tokens even attend to those parts more than the key entities. Our method can enhance both the [CLS] and patch attention to make them focusing on the key entities. Interestingly, although we do not calculate the attention enhancement loss at the penultimate layer, the attention of its [CLS] and patch tokens are still significantly enhanced and focus on the key entities. As the theoretical analysis in Section 3.5, our method can increase the mutual information between the input tokens of the last layer (i.e., the output tokens of the penultimate layer) and the key patches. In order to achieve this objective, the [CLS] and patch tokens of the penultimate layer should also pay more attention to the key patches to introduce more information from them.
Figure 9: Visualization of the attention map of the [CLS] token and different patches at the penultimate (11-th) and last (12-th) layer of ViT-B/16. Here we use the ViT-B/16 pre-trained on ImageNet-1K with DINO for backbone initialization and evaluate the effect of our method on LoRA and SSF.