# Rethinking Few-Shot Adaptation of Vision-Language Models in Two Stages

Matteo Farina[1,*] Massimiliano Mancini[1] Giovanni Iacca[1] Elisa Ricci[1,2]
[1]University of Trento [2]Fondazione Bruno Kessler

## Abstract

*An old-school recipe for training a classifier is to (i) learn a good feature extractor and (ii) optimize a linear layer atop. When only a handful of samples are available per category, as in Few-Shot Adaptation (FSA), data are insufficient to fit a large number of parameters, rendering the above impractical. This is especially true with large pre-trained Vision-Language Models (VLMs), which motivated successful research at the intersection of Parameter-Efficient Fine-tuning (PEFT) and FSA. In this work, we start by analyzing the learning dynamics of PEFT techniques when trained on few-shot data from only a subset of categories, referred to as the "base" classes. We show that such dynamics naturally splits into two distinct phases: (i) task-level feature extraction and (ii) specialization to the available concepts. To accommodate this dynamic, we then depart from prompt- or adapter-based methods and tackle FSA differently. Specifically, given a fixed computational budget, we split it to (i) learn a task-specific feature extractor via PEFT and (ii) train a linear classifier on top. We call this scheme Two-Stage Few-Shot Adaptation (2SFS). Differently from established methods, our scheme enables a novel form of selective inference at a category level,* i.e.*, at test time, only novel categories are embedded by the adapted text encoder, while embeddings of base categories are available within the classifier. Results with fixed hyperparameters across two settings, three backbones, and eleven datasets, show that 2SFS matches or surpasses the state-of-the-art, while established methods degrade significantly across settings. .*

## 1. Introduction

Effective visual classification requires two key components: a good feature extractor and a strong classifier operating on features. This is established knowledge in computer vision, and multiple influential works [35, 44] have demonstrated that a simple linear classifier, optimized atop a frozen feature extractor pre-trained at a larger scale (*e.g.*, on ImageNet

* Corresponding author: m.farina@unitn.it. Code at https://github.com/FarinaMatteo/rethinking_fewshot_vlms

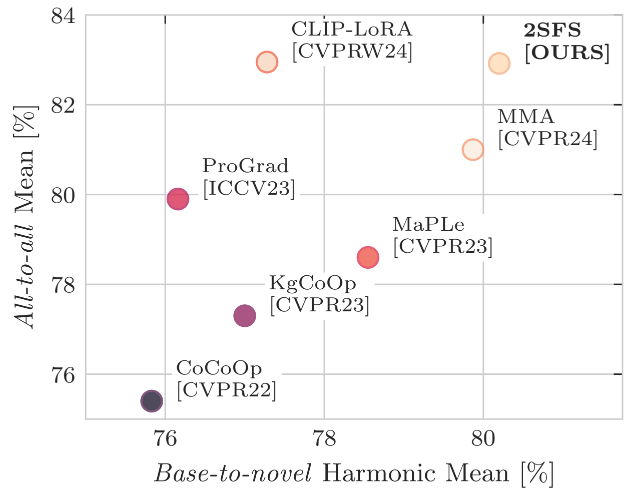



Figure 1. We present 2SFS, a technically simple revision of classifier tuning. 2SFS exhibits favorable performance both in *all-to-all* FSA, where train/test categories coincide, as well as in the more challenging *base-to-novel* setup, where only a subset of *base* annotated categories are available, and the test suite further spans a set of unseen (*novel*) classes. Conversely, setting-specific SOTA approaches [42, 47] degrade between settings.

[34]), achieves state-of-the-art or competitive results on a variety of downstream tasks [49]. Throughout adaptation, the feature extractor can also be fine-tuned together with the classifier, commonly with a lower learning rate to avoid disrupting the knowledge instilled from pretraining. Notably, the success of this latter paradigm is linked to the curse of dimensionality and the risk of overfitting when there are too few annotated samples relative to the parameters.

In this work, we deal with the Few-Shot Adaptation (FSA) of Vision-Language Models (VLMs), where the data-to-parameter ratio is at a critical level: there are typically hundreds of millions of parameters, but only a handful of data points are available on a per-category basis (*i.e.*, the "shots"). Such a dilemma is emphasized when shots are available for a *restricted* subset of categories, often referred to as the "base" categories, and the downstream task is assumed to span a broader range, comprising both base and "novel" semantic concepts. This non-trivial challenge fueled modern research on Parameter-Efficient Fine-Tuning (PEFT) [6, 12, 23], with successful examples in the FSA of VLMs that are mostly categorized into two (non-conflicting) paradigms: (i) *prompt tuning*, which optimizes a set of context vectors in either the text [52, 53] or the vision encoder [43] and (ii) *adapter-based* methods, where parametric functions are wrapped as external entities around (or on top of) the frozen VLM [10, 42, 47].

*All* of the methods above consistently outperform full fine-tuning and linear probing. However, this is mostly an *a-posteriori* observation, and despite the abundance of literature in the field, very little is known about the *learning dynamics* in FSA, *i.e*., about what happens throughout training with only a handful of examples. We start this work by filling this gap, analyzing the behavior of CLIP [33] when adapting it to downstream tasks with different PEFT strategies. Surprisingly, we find that the learning dynamics exhibit consistent patterns across datasets when evaluated on held-out data: (i) an initial stage of *joint* performance increase on both seen and unseen concepts; (ii) a *breakpoint*, after which PEFT strategies still largely improve on base categories at the expense of degradation on unseen semantic concepts. In other words, *the initial stages of PEFT make for good task-level feature extraction, while the second stage specializes in the available data*.

Exploiting this finding, we introduce **Two S**tage **F**ew-**S**hot Adaptation (2SFS), a technically simple revision of the old-school classifier tuning paradigm. Specifically, we split a fixed compute budget into two stages: first, we fine-tune only LayerNorm [21] instances of both modality-encoders to obtain a generalizable task-level feature extractor; second, we optimize a classifier on top, *i.e*., the text embeddings of base categories, to improve the discrimination ability of the model. In contrast to existing methods, 2SFS enables a novel form of selective inference at a category level, *i.e*., at test-time, only novel categories are embedded by the adapted text encoder, while base categories are available in $\mathcal{O}(1)$, being rows of the classifier matrix.

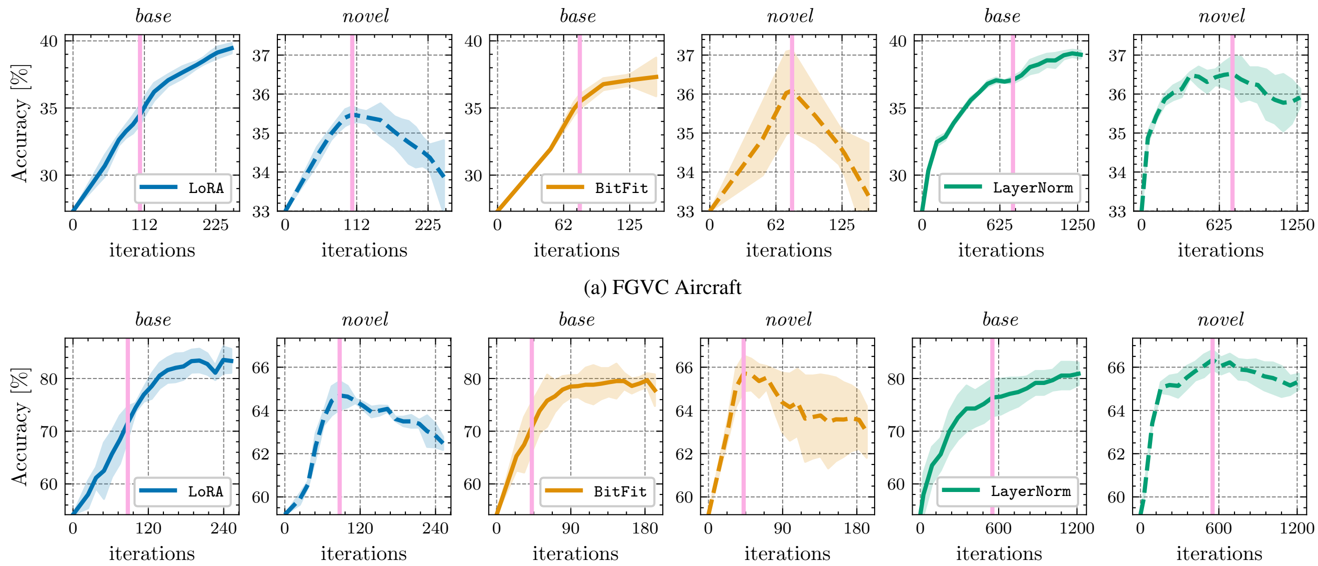
We thoroughly validate 2SFS *with fixed hyperparameters* across a suite of 11 publicly available datasets, 2 different settings (*i.e*., base-to-novel and all-to-all), and 3 backbones, showing that our simple approach matches or surpasses setting-specific state-of-the-art methods, while they conversely degrade across settings (see Fig. 1).

## 2. Related work

**Parameter-efficient fine-tuning** (PEFT) [24] adapts a large pretrained model by optimizing a small subset of parameters, while keeping the rest of the model frozen. One can identify three main categories among PEFT techniques: prompt tuning, selective approaches and adapter-based methods. Prompt tuning [15, 22] adds trainable tokens either to the input or within intermediate layers. Initially designed for language prompts [23], recent works have incorporated visual tokens into the recipe [15]. Selective methods optimize only a carefully chosen subset of *existing* model parameters. With vision architectures, popular choices are batch- or layer-normalization modules [14, 21]. Within the same categorization, BitFit (BIas-Term FIne-Tuning) [46], originally evaluated on NLP tasks, focuses on bias terms. In contrast, adapter-based methods introduce *external functions* to adapt the model. Examples are scaling and shifting hidden features [25], adding non linear parametric functions on top of the frozen model outputs [10], or side adapters, slightly refining features across layers [30]. Recently, *reparametrization-based* adapters have gained significant attention [6, 12, 19, 26], since their design allows to *merge* the newly introduced parameters with the frozen model, incurring no additional inference cost. LoRA [12] is arguably the most popular example, optimizing a low-rank decomposition of the network's parameters.

**Few-shot adaptation of VLMs.** The impressive performance of modern VLMs has motivated researchers to develop adaptation techniques to expand their capabilities to specific tasks [41]. While some of these techniques are purely test-time methods requiring no supervision [8, 36, 48], a common setup is to assume a handful of labeled examples are given. In such a context, PEFT strategies have been designed to account for the multimodal nature of VLMs. Following the previous categorization, the earliest approaches belong to the prompt tuning family [51–53]. CoOp (Context Optimization) [52] learns a set of soft context vectors to circumvent effortful manual prompting. CoCoOp (Conditional Context Optimization) [51] improves CoOp, by generating an image-conditioned context. ProGrad (Prompt-aligned Gradient) [53] strives to preserve pretraining knowledge by only updating prompts when the gradient aligns with a specific direction. Similarly, KgCoOp (Knowledge-guided CoOp) [43] mitigates forgetting by guiding prompt learning with a hand-crafted prompt. PLOT (Prompt Learning with Optimal Transport) [2] and MaPLe (Multi-modal Prompt Learning) [16] make a step further, connecting the visual and textual branches. In PLOT [2], class specific prompts are transported to visual features via optimal transport; MaPLe [16] conditions deep visual prompts with their textual counterparts, with successful results. A second category of FSA techniques relies on adapters [10, 38, 45, 50]. For example, CLIP-Adapter [10] mixes pretrained features with their non-linearly-transformed versions. TaskRes [45] tunes a set of residual parameters instead of introducing a non-linear bottleneck. Other methods [38, 50] avoid fine-tuning and use the few-shot data as a cache to refine predictions.

In this work, we introduce a new perspective: the benefit of a *two-stage* design that adapts to the downstream task while improving generalization to unavailable categories.




(a) FGVC Aircraft

(b) DTD

Figure 2. The natural emergence of a breakpoint during few-shot adaptation with different PEFT strategies. Before the breakpoint (to the left of the line), PEFT learns good task-level features, showed by *joint* performance increase on both *base* and *novel* categories. After the breakpoint (to the right of the line), PEFT specializes in the available data, incurring unrecoverable performance degradation on novel categories accompanied by consistent improvement in base concepts. Results refer to CLIP [33] with the ViT-B/16 visual backbone [7].

## 3. Preliminaries

In this section, we first formalize the FSA setup (Sec. 3.1). We then conduct a preliminary investigation on the impact of PEFT on the learning dynamics of FSA, with a particular emphasis on the relationship between generalization performance on base and novel categories (Sec. 3.2).

### 3.1. Problem formulation

FSA aims at adapting a model $f$ to a specific downstream task given a few annotated examples per category. Formally, the available data in FSA are a collection of (image, category) pairs $D = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{B}$ being the available images and the set of *base* categories, respectively. A small number $k \in \mathbb{N}$ of available samples, referred to as *shots*, is constant across categories, entailing that the available dataset has cardinality $n = k \times |\mathcal{B}|$. In general, the downstream task spans a set $\mathcal{C}$ of semantic categories, which is a *superset* of the available annotated categories, *i.e.*, $\mathcal{B} \subseteq \mathcal{C}$, and no reliable assumptions can be made on the annotated data. Hence, $\mathcal{B}$ can coincide with $\mathcal{C}$, but also a set of *novel* categories $\mathcal{N} \subseteq \mathcal{C}$ may exist such that $\mathcal{B} \cap \mathcal{N} = \emptyset$ and $\mathcal{B} \cup \mathcal{N} = \mathcal{C}$. One talks about *all-to-all* FSA when $\mathcal{N} = \emptyset$ (or, equivalently, $\mathcal{B} = \mathcal{C}$). Conversely, we talk about *base-to-novel* generalization.

The function $f$ is typically a contrastive VLM [33], thus it contains a visual encoder $f^v : \mathcal{I} \to \mathbb{R}^d$ and a text encoder $f^t : \mathcal{T} \to \mathbb{R}^d$ mapping images and texts in a shared $d$-dimensional Euclidean space. The domains $\mathcal{I}$ and $\mathcal{T}$ can be thought of as the image and text spaces. When classes are specified in natural language, *i.e.*, each $c \in \mathcal{C}$ is mapped to a string in $\mathcal{T}$, zero-shot classification with $f$ takes the form:

$$f(x, \mathcal{C}) = \arg\max_{c \in \mathcal{C}} \left\{ < f^v(x), f^t(c) >, \forall c \in \mathcal{C} \right\} \tag{1}$$

where $< \cdot, \cdot >$ denotes cosine similarity. Typically, the VLM $f$ is parametrized by $\theta \in \mathbb{R}^N$, which largely exceed the number of available examples (*i.e.*, $N >> n$), rendering the optimization of $\theta$ subject to a high risk of overfitting. Thus, a common strategy is to optimize a smaller set of task-specific parameters $\omega$, according to a PEFT technique. Since $\omega$ may influence both vision and text encoders, without loss of generality, we write it as $\omega = \omega_v \cup \omega_t$. In most cases, $\omega$ are optimized through the standard softmax cross-entropy objective computed across *base* categories (we omit the temperature for ease of notation):

$$\mathcal{L}(x_i, y_i) = -\log \frac{\exp(< f^v_{\omega_v}(x_i), f^t_{\omega_t}(y_i) >)}{\sum_{b \in \mathcal{B}} \exp(< f^v_{\omega_v}(x_i), f^t_{\omega_t}(b) >)}, \tag{2}$$

where $f^v_{\omega_v}$ and $f^t_{\omega_t}$ are the visual and textual encoders modified by $\omega_v$ and $\omega_t$, respectively. The objective in Eq. (2) is minimized for a predefined number of iterations $m$.When optimization ends, the updated parameters $\omega^* = \omega^*_v \cup \omega^*_t$ are used to parametrize $f$, and Eq. (1) reads:

$$f(x, \mathcal{C}) = \arg\max_{c \in \mathcal{C}} \left\{ < f^v_{\omega^*_v}(x), f^t_{\omega^*_t}(c) >, \forall c \in \mathcal{C} \right\}. \tag{3}$$

### 3.2. What happens during Few-Shot Adaptation?

A desideratum of FSA methods is that, by minimizing Eq. (2), the optimized set of parameters $\omega^*$ improves on base classes and generalizes to novel ones. Mechanisms such as weight sharing across encoders [42] or cross-modal prompting [16, 51] have been explicitly developed to promote such behavior, suggesting that *ad-hoc* designs are required to instill general task knowledge into $f$. In this section, we challenge this assumption by studying the learning dynamics of PEFT in FSA, *when no explicit mechanism promoting generalization is involved in the learning recipe*.

**Preliminary PEFT Setup.** To conduct our study, we focus on three PEFT methods: LayerNorm tuning [18], LoRA [12], and BitFit [46]. In **LayerNorm** tuning, typically used for visual adaptation [5], $\omega$ are the scale and bias parameters of the LayerNorm instances of the model. **LoRA** (low-rank adapters), recently shown effective for VLMs [47], adds trainable low-rank matrices into each linear projection of the model, following a low-rank decomposition of the form $\mathbf{W}\mathbf{x}+\gamma\mathbf{B}\mathbf{A}\mathbf{x}$. In this case, $\omega$ is the set of all low-rank matrices $\mathbf{B}$, $\mathbf{A}$. We adopt the design of CLIP-LoRA [47], which has extensively studied rank and placement of LoRA modules within CLIP, thereby plugging low-rank modules in all query, key, and value matrices of both encoders, with a rank $r$=2. **BitFit**, originally developed for language encoders [46], adapts the bias terms of all layers. In this latter case, $\omega$ contains the shift vectors of all affine transformations in the network. We apply such PEFT techniques on CLIP ViT-B/16, training with $k$=16 shots from both Describable Textures [3] and FGVC Aircraft [29]. We follow the established base/novel classes split introduced in CoCoOp [51] and set a budget of $m$=8000 iterations (*i.e.*, gradient steps) for all PEFT techniques, which allows us to study the exact learning dynamics of the recent CLIP-LoRA [47]. See Appendix E for the same analysis on other datasets.

**Experimental Outcome.** Fig. 2 displays the learning dynamics when the adapted model is evaluated on held-out samples. From the outcome, it emerges that the dynamics of PEFT methods exhibit a breakpoint, *naturally* separating adaptation into two distinct phases:

1. **First Stage**: consistent and *joint* improvement on *both* base and novel categories. In other words, this means that PEFT is learning *good task-level features* from both encoders. If this were not the case, then only the performance on available categories would increase;
2. **Second Stage**: good task-level features are disrupted in favor of exploitation of the available categories. Note that, while this phenomenon looks akin to overfitting, it is not, since models are evaluated on held-out data for both $\mathcal{B}$ and $\mathcal{N}$. In contrast, it has a different flavor: after the first stage, PEFT *specializes* in the available categories, overriding knowledge that would have been helpful for the downstream task as a whole.

We find that the pattern is consistent across datasets and PEFT techniques. Between the two stages, the **First Stage** is the most counter-intuitive: PEFT learns good vision-language features, although *no* ad-hoc designs were injected to promote such behavior. We also observe traits characterizing PEFT methods. For example, LoRA tends to better discriminate among base categories, while LayerNorm appears less capable in this respect. However, it is much more robust in the second stage and reaches the breakpoint far later than LoRA and BitFit. We now leverage these observations to design a simple yet effective FSA strategy.

## 4. Two-Stage Few-Shot Adaptation

This section introduces **Two-S**tage **F**ew-**S**hot Adaptation (2SFS), and describes how we leverage the natural emergence of two stages during FSA. 2SFS revises the old-school paradigm of classifier tuning [39], which has been set aside in favor of prompt- or adapter-based methods. Given a fixed computational budget expressed as a maximum number of iterations $m$, 2SFS acts in two short stages:

1. **First Stage**: initially, 2SFS allocates $\alpha \times m$ iterations to update the feature extractor via PEFT to improve base *and* novel class performance, with $\alpha \in [0, 1]$. Motivated by the results of Fig. 2, highlighting the enhanced robustness of Layer Normalization, we describe the rest of this section assuming that $\omega$ are LayerNorm scale and shifts. We additionally report 2SFS with LoRA in Appendix A.1.
2. **Second Stage**: for the remaining $(1 - \alpha) \times m$ steps, 2SFS *switches* parameters to avoid disrupting helpful task-level features and learns a *base* classifier.

At inference time 2SFS enables a novel form of adaptive inference on a *per-category basis*, *i.e.*, we skip the computation of the text encoder when embedding categories in $\mathcal{B}$, as they are readily available as rows of the classifier. We now dive into the details of these components.

### 4.1. First stage: learning task-level features

In the first stage, we want to learn a good feature extractor for base categories that also generalizes to novel concepts. To achieve this, we exploit the first learning phase in PEFT, where the performances on both sets increase. In this phase, we tune all LayerNorm instances of both visual and textual encoders. Formally, given a $d$-dimensional vector $\mathbf{a} \in \mathbb{R}^d$ of activations, LayerNorm [21] operates by scaling and shifting the standardized features in $\mathbf{a}$, *i.e*.

$$\texttt{LayerNorm}(\mathbf{a}) = \gamma \odot \left(\frac{\mathbf{a} - \mu(\mathbf{a})}{\sigma(\mathbf{a})}\right) + \beta, \qquad (4)$$

where $\mu(\mathbf{a}) \in \mathbb{R}$ is the mean of the vector, $\sigma(\mathbf{a}) \in \mathbb{R}$ is its standard deviation, while $\gamma$ and $\beta$ are known as the "scale" and "shift" parameters. Both $\gamma$ and $\beta$ are learnable $d$-dimensional vectors and are subject to fine-tuning.

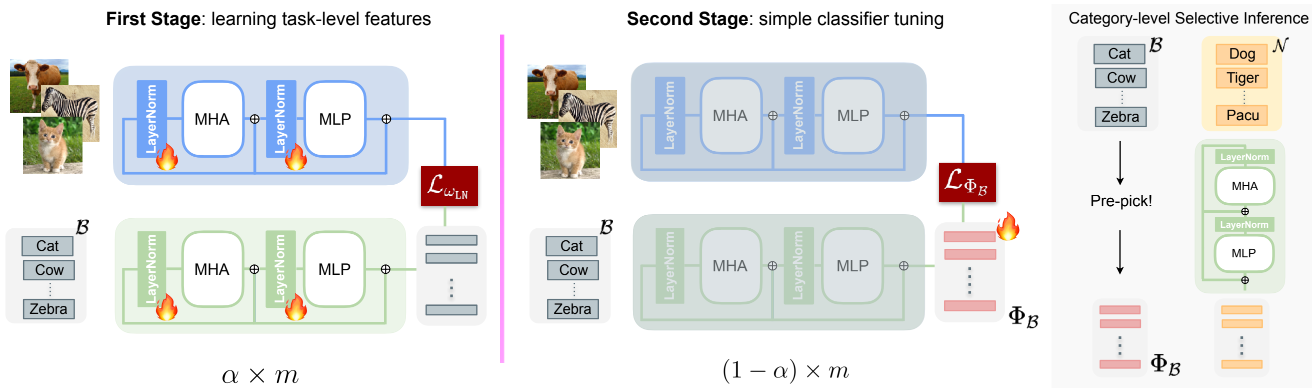



Figure 3. **2SFS**. Given a computational budget of $m$ iterations, 2SFS operates in two separate stages. In the **First Stage**, 2SFS learns task-level features by tuning LayerNorm instances. In the **Second Stage**, a simple classifier initialized with the text embeddings of base categories learns to separate task-level features. At inference time, 2SFS allows to selectively embed categories. Specifically, only novel categories are embedded by the adapted text encoder, while embeddings of base categories are available as rows within $\Phi_\mathcal{B}$.

Borrowing from the notation of Sec. 3, we denote as task-specific parameters the scale and shift vectors of each LayerNorm instance of the network. Assuming a total of $L$ instances, we have $\omega_{\text{LN}} = \{(\gamma_1, \beta_1), \cdots, (\gamma_L, \beta_L)\}$, which are the union of modality-specific parameters $\omega^v_{\text{LN}}$ and $\omega^t_{\text{LN}}$. The objective function in Eq. (2) now reads:

$$\mathcal{L}_{\omega_{\text{LN}}}(x_i, y_i) = -\log \frac{\exp(< f^v_{\omega^v_{\text{LN}}}(x_i), f^t_{\omega^t_{\text{LN}}}(y_i) >)}{\sum_{b \in \mathcal{B}} \exp(< f^v_{\omega^v_{\text{LN}}}(x_i), f^t_{\omega^t_{\text{LN}}}(b) >)}, \tag{5}$$

which we optimize for a $\alpha \times m$ iterations, where $\alpha \in [0, 1]$. Thus, this stage only takes a *fraction* of the available compute budget and ensures that no further optimization is carried out w.r.t. $\omega_{\text{LN}}$ after the natural breakpoint is reached (*i.e.*, ideally, $\alpha \times m$ should match the pink line of Fig. 2).

## 4.2. Second stage: simple classifier tuning

After the first stage, we obtain optimized parameters $\omega^*_{\text{LN}} = \omega^{*v}_{\text{LN}} \cup \omega^{*t}_{\text{LN}}$ that incorporate general task knowledge. From Sec. 3.2, we know that further tuning $\omega^*_{\text{LN}}$ disrupts helpful features for novel categories, hence we circumvent this issue by carrying out optimization w.r.t. a different set of parameters. We employ arguably the simplest form to do so: we *freeze* $\omega^*_{\text{LN}}$ and train a linear classifier on top. Let $\phi_b = f^t_{\omega^{*t}_{\text{LN}}}(b)$ denote the embedding of the category $b$, obtained with the first-stage text encoder. We initialize the classifier weights $\Phi_\mathcal{B} = \{\phi_b\}_{b \in \mathcal{B}}$ by stacking all embeddings of base categories, and optimize $\Phi_\mathcal{B}$ for the remaining $(1-\alpha) \times m$ steps. At this stage, Eq. (2) becomes:

$$\mathcal{L}_{\Phi_\mathcal{B}}(x_i, y_i) = -\log \frac{\exp(< f^v_{\omega^{*v}_{\text{LN}}}(x_i), \phi_{y_i} >)}{\sum_{b \in \mathcal{B}} \exp(< f^v_{\omega^{*v}_{\text{LN}}}(x_i), \phi_b >)}, \tag{6}$$

where $\Phi_\mathcal{B}$ learns to separate features obtained with the frozen first-stage visual parameters $\omega^{*v}_{\text{LN}}$. Along with its simplicity, this design fully exploits helpful task-level representations: $\Phi_\mathcal{B}$ is initialized with first-stage text embeddings and learns to separate first-stage visual embeddings. The optimized classifier weights are denoted as $\Phi^*_\mathcal{B} = \{\phi^*_b\}_{b \in \mathcal{B}}$.

## 4.3. Category-level Selective Inference

The proposed two-stage design provides unique advantages, allowing for selective inference within the text encoder. Specifically, given an image $x$ and a set of downstream categories $\mathcal{C}$ to discriminate, our prediction function follows:

$$f_{\omega^*_{\text{LN}}, \Phi^*_\mathcal{B}}(x) = \arg\max_{c \in \mathcal{C}} < f^v_{\omega^{*v}_{\text{LN}}}(x), \phi^*_c > \tag{7}$$

with each $\phi^*_c$ being:

$$\phi^*_c = \begin{cases} \phi^*_b & \text{if } \exists b \in \mathcal{B} \mid b = c \\ f^t_{\omega^{*t}_{\text{LN}}}(c) & \text{otherwise.} \end{cases} \tag{8}$$

In essence, at inference time, the two-stage design allows to *skip* computations within the text encoder, instead of embedding the categories that were available in the annotated data (which are always known in FSA), and to pick the corresponding row of the classifier in $\mathcal{O}(1)$ directly. In contrast, images and unseen categories are always processed with the parameters obtained after the first stage.

In summary, 2SFS provides a *unified recipe* for FSA: splitting into stages and optimizing a different parameter subset allows to ① preserve helpful knowledge for unseen categories and ② exploit available annotations. The overall pipeline of 2SFS is schematized by Fig. 3.

# 5. Experiments

This section highlights the experimental results of 2SFS, evaluating: (i) *all-to-all* FSA, in which train/test categories

Table 1. Experiments in *base-to-novel* generalization with the ViT-B/16 visual backbone. All methods use $k$=16 shots per base class. "CLIP" refers to zero-shot performance with dataset-specific templates, *e.g.*, "*a photo of a* {}*, a type of flower*" for Oxford Flowers.

**Average across datasets.**

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 69.34 | 74.22 | 71.70 |
| CoOp [52] | 82.69 | 63.22 | 71.66 |
| CoCoOp [51] | 80.47 | 71.69 | 75.83 |
| MaPLe [16] | 82.28 | 75.14 | 78.55 |
| ProGrad [53] | 82.48 | 70.75 | 76.16 |
| KgCoOp [43] | 80.73 | 73.60 | 77.00 |
| CLIP-LoRA [47] | 85.32 | 70.63 | 77.28 |
| MMA [42] | 83.20 | 76.80 | 79.87 |
| 2SFS | 85.55 | 75.48 | **80.20** |

ImageNet

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 72.43 | 68.14 | 70.22 |
| CoOp [52] | 76.47 | 67.88 | 71.92 |
| CoCoOp [51] | 75.98 | 70.43 | 73.10 |
| MaPLe [16] | 76.66 | 70.54 | 73.47 |
| ProGrad [53] | 77.02 | 66.66 | 71.46 |
| KgCoOp [43] | 75.83 | 69.96 | 72.78 |
| CLIP-LoRA [47] | 77.58 | 68.76 | 72.91 |
| MMA [42] | 77.31 | 71.00 | 74.02 |
| 2SFS | 77.71 | 70.99 | **74.20** |

Caltech101

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 96.84 | 94.00 | 95.40 |
| CoOp [52] | 98.00 | 89.81 | 93.73 |
| CoCoOp [51] | 97.96 | 93.81 | 95.84 |
| MaPLe [16] | 97.74 | 94.36 | 96.02 |
| ProGrad [53] | 98.02 | 93.89 | 95.91 |
| KgCoOp [43] | 97.72 | 94.39 | 96.03 |
| CLIP-LoRA [47] | 98.19 | 93.05 | 95.55 |
| MMA [42] | 98.40 | 94.00 | 96.15 |
| 2SFS | 98.71 | 94.43 | **96.52** |

Oxford Flowers

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 72.08 | 77.80 | 74.83 |
| CoOp [52] | 97.60 | 59.67 | 74.06 |
| CoCoOp [51] | 94.87 | 71.75 | 81.71 |
| MaPLe [16] | 95.92 | 72.46 | 82.56 |
| ProGrad [53] | 95.54 | 71.87 | 82.03 |
| KgCoOp [43] | 95.00 | 74.73 | 83.65 |
| CLIP-LoRA [47] | 97.91 | 68.61 | 80.68 |
| MMA [42] | 97.77 | 75.93 | 85.48 |
| 2SFS | 98.29 | 76.17 | **85.83** |

Oxford Pets

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 91.17 | 97.26 | 94.12 |
| CoOp [52] | 93.67 | 95.29 | 94.47 |
| CoCoOp [51] | 95.20 | 97.69 | 96.43 |
| MaPLe [16] | 95.43 | 97.76 | 96.58 |
| ProGrad [53] | 95.07 | 97.63 | 96.33 |
| KgCoOp [43] | 94.65 | 97.76 | 96.18 |
| CLIP-LoRA [47] | 94.36 | 95.71 | 95.03 |
| MMA [42] | 95.40 | 98.07 | **96.72** |
| 2SFS | 95.32 | 97.82 | 96.55 |

Stanford Cars

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 63.37 | 74.89 | 68.65 |
| CoOp [52] | 78.12 | 60.40 | 68.13 |
| CoCoOp [51] | 70.49 | 73.59 | 72.01 |
| MaPLe [16] | 72.94 | 74.00 | 73.47 |
| ProGrad [53] | 77.68 | 68.63 | 72.88 |
| KgCoOp [43] | 71.76 | 75.04 | 73.36 |
| CLIP-LoRA [47] | 83.93 | 65.54 | 73.60 |
| MMA [42] | 78.50 | 73.10 | 75.70 |
| 2SFS | 82.50 | 74.80 | **78.46** |

Food 101

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 90.10 | 91.22 | 90.66 |
| CoOp [52] | 88.33 | 82.26 | 85.19 |
| CoCoOp [51] | 90.70 | 91.29 | 90.99 |
| MaPLe [16] | 90.71 | 92.05 | **91.38** |
| ProGrad [53] | 90.37 | 89.59 | 89.98 |
| KgCoOp [43] | 90.50 | 91.70 | 91.09 |
| CLIP-LoRA [47] | 86.84 | 86.67 | 86.76 |
| MMA [42] | 90.13 | 91.30 | 90.71 |
| 2SFS | 89.11 | 91.34 | 90.21 |

FGVC Aircraft

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 27.19 | 36.29 | 31.09 |
| CoOp [52] | 40.44 | 22.30 | 28.75 |
| CoCoOp [51] | 33.41 | 23.71 | 27.74 |
| MaPLe [16] | 37.44 | 35.61 | 36.50 |
| ProGrad [53] | 40.54 | 27.57 | 32.82 |
| KgCoOp [43] | 36.21 | 33.55 | 34.83 |
| CLIP-LoRA [47] | 50.10 | 26.03 | 34.26 |
| MMA [42] | 40.57 | 36.33 | 38.33 |
| 2SFS | 47.48 | 35.51 | **40.63** |

SUN 397

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 69.36 | 75.35 | 72.23 |
| CoOp [52] | 80.60 | 65.89 | 72.51 |
| CoCoOp [51] | 79.74 | 76.86 | 78.27 |
| MaPLe [16] | 80.82 | 78.70 | 79.75 |
| ProGrad [53] | 81.26 | 74.17 | 77.55 |
| KgCoOp [43] | 80.29 | 76.53 | 78.36 |
| CLIP-LoRA [47] | 81.11 | 74.53 | 77.68 |
| MMA [42] | 82.27 | 78.57 | 80.38 |
| 2SFS | 82.59 | 78.91 | **80.70** |

DTD

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 53.24 | 59.90 | 56.37 |
| CoOp [52] | 79.44 | 41.18 | 54.24 |
| CoCoOp [51] | 77.01 | 56.00 | 64.85 |
| MaPLe [16] | 80.36 | 59.18 | 68.16 |
| ProGrad [53] | 77.35 | 52.35 | 62.45 |
| KgCoOp [43] | 77.55 | 54.99 | 64.35 |
| CLIP-LoRA [47] | 83.95 | 62.84 | 71.39 |
| MMA [42] | 83.20 | 65.63 | 73.38 |
| 2SFS | 84.60 | 65.01 | **73.52** |

EuroSAT

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 56.48 | 64.05 | 60.03 |
| CoOp [52] | 92.19 | 54.74 | 68.69 |
| CoCoOp [51] | 87.49 | 60.04 | 71.21 |
| MaPLe [16] | 94.07 | 73.23 | 82.35 |
| ProGrad [53] | 90.11 | 60.89 | 72.67 |
| KgCoOp [43] | 85.64 | 64.34 | 73.48 |
| CLIP-LoRA [47] | 97.04 | 62.50 | 76.03 |
| MMA [42] | 85.46 | 82.34 | **83.87** |
| 2SFS | 96.91 | 67.09 | 79.29 |

UCF101

| Method | Base | Novel | HM |
|---|---|---|---|
| CLIP [33] | 70.53 | 77.50 | 73.85 |
| CoOp [52] | 84.69 | 56.05 | 67.46 |
| CoCoOp [51] | 82.33 | 73.45 | 77.64 |
| MaPLe [16] | 83.00 | 78.66 | 80.77 |
| ProGrad [53] | 84.33 | 74.94 | 79.35 |
| KgCoOp [43] | 82.89 | 76.67 | 79.65 |
| CLIP-LoRA [47] | 87.52 | 72.74 | 79.45 |
| MMA [42] | 86.23 | 80.03 | 82.20 |
| 2SFS | 87.85 | 78.19 | **82.74** |

coincide (*i.e.*, $\mathcal{B} = \mathcal{C}$) and (ii) *base-to-novel* generalization, where each task further spans a set $\mathcal{N}$ of unseen categories.

**Datasets.** For both settings, we consider a suite of 11 benchmarks. Specifically, these include ImageNet [34], Caltech101 (CAL) [9], SUN397 [40], Describable Textures (DTD) [3], FGVC Aircraft (AIR) [29], Oxford Pets (PETS) [32], Oxford Flowers 102 (FLWR) [31], Stanford Cars (CARS) [20], Food-101 (FOOD) [1], EuroSAT (ESAT) [11] and UCF-101 (UCF) [37]. We use the splits of [51].

**Baselines.** In *base-to-novel* generalization, we consider a total of 8 comparative methods: CoOp [52] and CoCoOp [51] as established baselines of the field; ProGrad [53], KgCoOp [43] and MaPLe [16] as modern advancements; Multi-Modal Adapter (MMA) [42] and CLIP-LoRA [47] as the strongest and most recent strategies. Zero-shot performance using hand-crafted templates is also reported as a reference. In the *all-to-all* setup, we expand the suite to a total of 12 methods, with 5 additional strategies that are *in-*

Table 2. *All-to-all* experiments, where train/test categories coincide, with the ViT-B/16 (top), ViT-B/32 (middle), and ViT-L/14 (bottom) backbones. All methods use $k = 16$ shots per class. In each group, the best performer is marked by **bold text**; the second best is <u>underlined</u>.

| Backbone | Method | ImageNet | SUN | AIR | ESAT | CARS | FOOD | PETS | FLWR | CAL | DTD | UCF | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ViT-B/16 | *Zero-Shot* | 66.7 | 62.6 | 24.7 | 47.5 | 65.3 | 86.1 | 89.1 | 71.4 | 92.9 | 43.6 | 66.7 | 65.1 |
| | CoOp [52] (ctx=16) | 71.9 | 74.9 | 43.2 | 85.0 | 82.9 | 84.2 | 92.0 | 96.8 | 95.8 | 69.7 | 83.1 | 80.0 |
| | CoCoOp [51] | 71.1 | 72.6 | 33.3 | 73.6 | 72.3 | **87.4** | 93.4 | 89.1 | 95.1 | 63.7 | 77.2 | 75.4 |
| | TIP-Adapter-F [50] | 73.4 | 76.0 | 44.6 | 85.9 | 82.3 | 86.8 | 92.6 | 96.2 | 95.7 | 70.8 | 83.9 | 80.7 |
| | CLIP-Adapter [10] | 69.8 | 74.2 | 34.2 | 71.4 | 74.0 | 87.1 | 92.3 | 92.9 | 94.9 | 59.4 | 80.2 | 75.5 |
| | PLOT++ [2] | 72.6 | 76.0 | 46.7 | 92.0 | 84.6 | 87.1 | 93.6 | 97.6 | 96.0 | 71.4 | 85.3 | 82.1 |
| | KgCoOp [43] | 70.4 | 73.3 | 36.5 | 76.2 | 74.8 | 87.2 | 93.2 | 93.4 | 95.2 | 68.7 | 81.7 | 77.3 |
| | TaskRes [45] | 73.0 | 76.1 | 44.9 | 82.7 | 83.5 | 86.9 | 92.4 | 97.5 | 95.8 | 71.5 | 84.0 | 80.8 |
| | MaPLe [16] | 71.9 | 74.5 | 36.8 | 87.5 | 74.3 | **87.4** | 93.2 | 94.2 | 95.4 | 68.4 | 81.4 | 78.6 |
| | ProGrad [53] | 72.1 | 75.1 | 43.0 | 83.6 | 82.9 | 85.8 | 92.8 | 96.6 | 95.9 | 68.8 | 82.7 | 79.9 |
| | LP++ [13] | 73.0 | 76.0 | 42.1 | 85.5 | 80.8 | 87.2 | 92.6 | 96.3 | 95.8 | 71.9 | 83.9 | 80.5 |
| | CLIP-LoRA [47] | <u>73.6</u> | 76.1 | **54.7** | <u>92.1</u> | **86.3** | 84.2 | 92.4 | **98.0** | **96.4** | 72.0 | **86.7** | **83.0** |
| | MMA [42] | 73.2 | <u>76.6</u> | 44.7 | 85.0 | 80.2 | 87.0 | **93.9** | 96.8 | 95.8 | <u>72.7</u> | 85.0 | 81.0 |
| | 2SFS | **73.7** | **77.0** | <u>50.0</u> | **92.4** | <u>85.4</u> | 86.1 | <u>93.7</u> | <u>97.7</u> | **96.4** | **73.2** | <u>86.6</u> | <u>82.9</u> |
| ViT-B/32 | *Zero-Shot* | 61.9 | 62.0 | 19.3 | 45.1 | 60.4 | 80.5 | 87.5 | 67.0 | 91.1 | 42.6 | 62.2 | 61.8 |
| | CoOp [52] (ctx=16) | 66.8 | 72.2 | 32.9 | 83.3 | 76.0 | 78.6 | 88.7 | 95.4 | 94.9 | 65.3 | 78.6 | 75.7 |
| | CoCoOp [51] | 66.0 | 69.8 | 22.6 | 70.4 | 64.6 | **81.9** | <u>91.0</u> | 82.5 | 94.3 | 59.7 | 75.3 | 70.7 |
| | TIP-Adapter-F [50] | **68.4** | <u>74.1</u> | 34.8 | 83.4 | 77.0 | 81.7 | 90.4 | 94.3 | 95.1 | 68.0 | 80.5 | 77.1 |
| | CLIP-Adapter [10] | 64.9 | 71.8 | 26.7 | 64.7 | 68.9 | **81.9** | 90.1 | 88.7 | 94.8 | 58.1 | 76.5 | 71.6 |
| | PLOT++ [2] | 67.4 | 73.4 | 36.3 | 91.1 | 77.4 | 79.7 | 89.1 | **96.3** | 94.9 | 67.0 | 81.5 | 77.6 |
| | KgCoOp [43] | 65.4 | 71.0 | 23.7 | 70.1 | 67.3 | 81.7 | 90.8 | 86.1 | 94.4 | 65.1 | 77.5 | 72.1 |
| | TaskRes [45] | 68.2 | 73.6 | 37.0 | 77.7 | 78.0 | 81.4 | 89.4 | 95.5 | 95.7 | 68.3 | 80.6 | 76.9 |
| | MaPLe [16] | 66.7 | 72.0 | 28.0 | 83.3 | 66.9 | 82.1 | 91.7 | 89.0 | 95.1 | 63.4 | 77.3 | 74.1 |
| | ProGrad [53] | 66.9 | 73.2 | 33.3 | 81.0 | 76.1 | 80.1 | 89.3 | 95.1 | 95.0 | 65.8 | 79.6 | 75.9 |
| | LP++ [13] | 68.1 | 74.0 | 34.3 | 82.8 | 75.2 | 81.8 | 90.5 | 93.9 | 95.0 | 67.8 | 80.1 | 76.7 |
| | CLIP-LoRA [47] | **68.4** | 74.0 | **44.9** | <u>91.8</u> | <u>79.7</u> | 78.2 | 88.8 | 96.2 | 95.2 | 68.2 | **82.8** | <u>78.9</u> |
| | MMA [42] | 68.0 | 74.0 | 34.0 | 80.1 | 73.5 | 81.4 | **91.5** | 94.3 | <u>95.6</u> | <u>68.9</u> | 81.7 | 76.7 |
| | 2SFS | **68.4** | **74.8** | <u>40.2</u> | **92.1** | **80.2** | 80.8 | 90.3 | **96.3** | **95.8** | **70.4** | <u>82.3</u> | **79.2** |
| ViT-L/14 | *Zero-Shot* | 72.9 | 67.6 | 32.6 | 58.0 | 76.8 | 91.0 | 93.6 | 79.4 | 94.9 | 53.6 | 74.2 | 72.2 |
| | CoOp [52] (ctx=16) | 78.2 | 77.5 | 55.2 | 88.3 | 89.0 | 89.8 | 94.6 | **99.1** | 97.2 | 74.4 | 87.3 | 84.6 |
| | CoCoOp [51] | 77.8 | 76.7 | 45.2 | 79.8 | 82.7 | 91.9 | <u>95.4</u> | 95.3 | 97.4 | 71.4 | 85.2 | 81.7 |
| | TIP-Adapter-F [50] | 79.3 | 79.6 | 55.8 | 86.1 | 88.1 | 91.6 | 94.6 | 98.3 | <u>97.5</u> | 74.0 | 87.4 | 84.8 |
| | CLIP-Adapter [10] | 76.4 | 78.0 | 46.4 | 75.8 | 83.8 | 91.6 | 94.3 | 97.3 | 97.3 | 71.3 | 86.1 | 81.7 |
| | PLOT++ [2] | 78.6 | 79.1 | 44.1 | 92.2 | 87.2 | 90.2 | 93.6 | 98.8 | <u>97.5</u> | 75.0 | 87.1 | 83.9 |
| | KgCoOp [43] | 76.8 | 76.7 | 47.5 | 83.6 | 83.2 | 91.7 | 95.3 | 96.4 | 97.4 | 73.6 | 86.4 | 82.6 |
| | TaskRes [45] | 78.1 | 76.9 | 55.0 | 84.3 | 87.6 | 91.5 | 94.7 | 97.8 | 97.3 | 74.4 | 86.6 | 84.0 |
| | MaPLe [16] | 78.4 | 78.8 | 46.3 | 85.4 | 83.6 | **92.0** | 95.4 | 97.4 | 97.2 | 72.7 | 86.5 | 83.1 |
| | ProGrad [53] | 78.4 | 78.3 | 55.6 | 89.3 | 88.8 | 90.8 | 94.9 | 98.7 | <u>97.5</u> | 73.7 | 87.7 | 84.9 |
| | LP++ [13] | 79.3 | 79.7 | 54.6 | 89.3 | 87.7 | 91.7 | 94.9 | 98.5 | 97.4 | 76.1 | 88.1 | 85.2 |
| | CLIP-LoRA [47] | <u>79.6</u> | 79.4 | **66.2** | **93.1** | **90.9** | 89.9 | 94.3 | 99.0 | 97.3 | <u>76.5</u> | **89.9** | <u>86.9</u> |
| | MMA [42] | **79.9** | <u>80.2</u> | 56.4 | 76.3 | 88.0 | **92.0** | **95.5** | 98.4 | **97.6** | 75.8 | 88.0 | 84.4 |
| | 2SFS | 79.4 | **80.3** | <u>64.1</u> | <u>92.9</u> | <u>90.3</u> | 91.1 | **95.5** | **99.1** | <u>97.5</u> | **78.0** | <u>89.5</u> | **87.1** |

*applicable* to unseen categories: TIP-Adapter-F [50], CLIP-Adapter [10], PLOT++ [2], TaskRes [45] and LP++ [13].

**Implementation Details.** Following CLIP-LoRA [47], we express the number of iterations as $m = M \times k$, where $k$ is the number of shots and $M \in \mathbb{N}$ is a hyperparameter. We also inherit the optimizer setup, using AdamW [27] with a learning rate of $2 \times 10^{-4}$, a weight decay factor of 0.01 and a mini-batch size of 32. We do *not* use dataset-specific templates, but we format all categories via the generic template "`a photo of a {}`". For all settings and competitors, we report numbers from the original works or from [47]. When unavailable, we reproduce the results using the official code. Results are averaged over 3 different runs.

**Shots configurations.** Due to space constraints, here we only report results with $k$=16 available shots. Complementary experiments with $k \in \{4, 8\}$ are in Appendix C.

**Fixed hyperparameters.** For a realistic evaluation, we fix $M$ and $\alpha$ by tuning on ImageNet [34] with the ViT-B/16 backbone and $k = 16$ shots only. Search intervals are defined as $M \in \{100, 300, 500\}$ and $\alpha \in [0.2, 0.8]$ (using a coarse step size of 0.1). We obtain values of $M = 300$ and $\alpha = 0.6$, which are always fixed unless stated otherwise.

### 5.1. Base-to-novel generalization

For this setting, we follow recent works [16, 42] and experiment with the ViT-B/16 backbone and $k = 16$ shots. The evaluation metrics are threefold: (i) top-1 accuracy on *base* categories, (ii) top-1 accuracy on *novel* categories, and (iii) the harmonic mean between them.

**Results** for all datasets are reported in Tab. 1, where (i), (ii), and (iii) are denoted as **Base**, **Novel** and **HM**, respectively. 2SFS provides a greater harmonic mean than all competitors in 8 out of 11 benchmarks, and outperforms all competitors on average across datasets. Notable success cases are Stanford Cars [20] and FGVC Aircraft [29], with margins in the harmonic mean of $+2.76\%$ and $+2.30\%$, while the worst case is EuroSAT. We attribute this to the tiny size of the dataset, as there are only 5 base categories, which leads 2SFS to exit the first stage when $\omega^*_{\texttt{LN}}$ are disrupted already. We verify this hypothesis by simply decreasing the batch size, hence collecting gradients for fewer examples in the first stage, observing notable performance improvements (*i.e.*, HM of $84.32\%$ with a batch size of 1).

### 5.2. All-to-all Few-Shot Adaptation

We further validate 2SFS in the all-to-all setup, where train/test categories coincide, *i.e.*, $\mathcal{B} = \mathcal{C}$. Intuitively, strong performance in this setting should support the hypothesis that the first stage learns good task-level features since a linear classifier is trained to separate them. Here, note that we have $\phi^*_c$ for all categories, hence selective inference *always* skips the computations within the text encoder.

**Setup details.** We follow recent work [47], and evaluate 2SFS across a variety of 3 different backbones: ViT-B/16, ViT-B/32 and ViT-L/14. We keep hyperparameters fixed.

**Results** are given in Tab. 2, confirming the hypothesis that a simple linear classifier, trained on top of frozen task-level features obtained with $\omega^*_{\texttt{LN}}$, is sufficient to match or surpass state-of-the-art methods. The best competitor in this setting is definitely CLIP-LoRA [47], which, nevertheless, is outperformed by 2SFS on average across backbones. We emphasize that **2SFS exhibits stable performance across settings**, a missing trait in both CLIP-LoRA and MMA [42]. 2SFS outperforms the former by $+2.9\%$ in base-to-novel settings, and surpasses the latter by $+1.9\%$, $+2.5\%$ and $+2.7\%$ in the all-to-all setup for the three backbones.

### 5.3. Analysis

We conclude the experimental section by (i) reporting the sweep conducted on $\alpha$ and (ii) highlighting the benefits of *switching* parameters in the second stage.

**Selection of $\alpha$.** Fig. 4 reports the outcome of the hyperparameter sweep on the fraction of iterations $\alpha$. We conduct this search for the *base-to-novel* setting using the validation set of ImageNet [34] and report the Harmonic Mean corresponding to each $\alpha$ value in $[0.2, 0.8]$, with a step size of

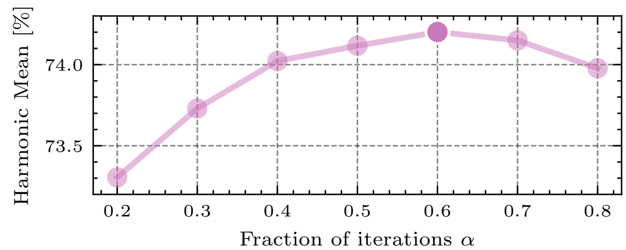



Figure 4. Hyperparameter sweep on $\alpha \in [0.2, 0.8]$ with a step size of 0.1, conducted on ImageNet [34]. The optimal value $\alpha = 0.6$ is transferred to every other experiment of this work.

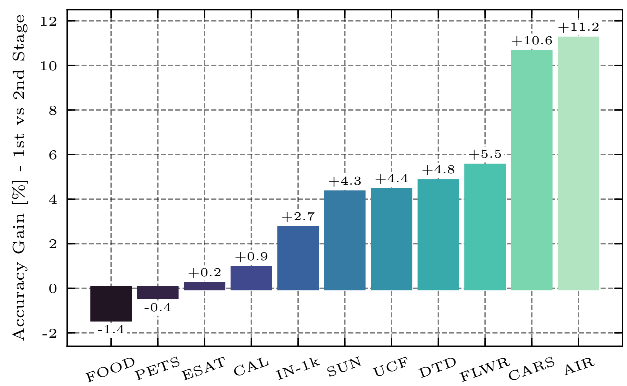



Figure 5. Visualization of the absolute accuracy improvement across the 11 benchmarks in the *all-to-all* setting, relative to exiting after the first stage, *i.e.*, relative to using $\omega^*_{\texttt{LN}}$ only.

0.1. We use the CLIP ViT-B/16 model and $k$=16 shots. In line with the evidence of Sec. 3.2, an optimal value $\alpha$=0.6 emerges, preceded and followed by decreased performance.

**The benefit of having two stages.** Fig. 5 compares 2SFS relative to only optimizing LayerNorm instances, showing that large margins are obtained on average (*e.g.*, more than $+10\%$ improvement on Stanford Cars and FGVC Aircraft, and $+4\%$ margin for SUN397, UCF-101, DTD, and Oxford Flowers further). We spot failure cases in Food-101 and Oxford Pets, which we examine in detail in Appendix E.

## 6. Conclusions

In this work, we investigated the dynamics of PEFT techniques in the FSA of VLMs. We empirically showed that PEFT learns good task-level features, even without *ad-hoc* mechanisms promoting cross-modal generalization, and that PEFT techniques exhibit different degrees of robustness to unseen categories. Through experiments on 11 benchmarks with fixed hyperparameters, we showed that training a linear classifier on top of frozen PEFT features, a scheme we call 2SFS, matches or outperforms the established state-of-the-art in different settings, thereby serving as a simple unified recipe. We hope our work will be useful for future research on VLM transfer.

**Acknowledgements.** The authors acknowledge the CINECA award under the ISCRA initiative for the availability of high-performance computing resources and support. M.F. is supported by the PRIN B-FAIR (Prot. 2022EX F3HX) project and the PAT project "AI@TN". This work was supported by the projects EU Horizon ELIAS (No. 101120237), AI4TRUST (No.101070190), FAIR - Future AI Research (PE00000013), funded by NextGeneration EU, and carried out in the Vision and Learning joint laboratory of Fondazione Bruno Kessler and the University of Trento, Italy.

## References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, 2014. 6, 11, 16, 17, 20

[2] Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. PLOT: Prompt Learning with Optimal Transport for Vision-Language Models. In *ICLR*, 2023. 2, 7, 11, 13, 16, 17, 18

[3] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 4, 6, 16

[4] Victor G Turrisi da Costa, Nicola Dall'Asen, Yiming Wang, Nicu Sebe, and Elisa Ricci. Diversified in-domain synthesis with efficient fine-tuning for few-shot classification. *arXiv preprint arXiv:2312.03046*, 2023. 16

[5] Thomas De Min, Massimiliano Mancini, Karteek Alahari, Xavier Alameda-Pineda, and Elisa Ricci. On the effectiveness of layernorm tuning for continual learning in vision transformers. In *ICCVW*, 2023. 4

[6] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS*, 2023. 2

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 3

[8] Matteo Farina, Gianni Franchi, Giovanni Iacca, Massimiliano Mancini, and Elisa Ricci. Frustratingly easy test-time adaptation of vision-language models. In *NeurIPS*, 2024. 2

[9] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 6

[10] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *IJCV*, 2024. 2, 7, 13, 16, 17, 18

[11] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 6, 16, 19

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022. 2, 4, 11, 19

[13] Yunshi Huang, Fereshteh Shakeri, Jose Dolz, Malik Boudiaf, Houda Bahig, and Ismail Ben Ayed. Lp++: A surprisingly strong linear probe for few-shot clip. In *CVPR*, 2024. 7, 13, 17, 18

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[15] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 2, 11

[16] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. MaPLe: Multi-modal Prompt Learning. In *CVPR*, 2023. 2, 4, 6, 7, 8, 11, 12, 13, 14, 16, 17, 18

[17] Muhammad Uzair Khattak, Syed Talal Wasim, Muzammal Naseer, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Self-regulating prompts: Foundational model adaptation without forgetting. In *ICCV*, 2023. 16

[18] Konwoo Kim, Michael Laskin, Igor Mordatch, and Deepak Pathak. How to adapt your large-scale vision-and-language model, 2022. 4

[19] Sanghyeon Kim, Hyunmo Yang, Yunghyun Kim, Youngjoon Hong, and Eunbyung Park. Hydra: Multi-head low-rank adaptation for parameter efficient fine-tuning. *Neural Networks*, 2024. 2

[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, 2013. 6, 8

[21] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization, 2016. arXiv:1607.06450. 2, 4

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. EMNLP. 2

[23] Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL*, 2021. 2, 11

[24] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2023. arXiv:2303.15647. 2

[25] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, 2022. 2

[26] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*, 2022. 2

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 7

[28] Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*, 2017. 17

[29] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft, 2013. arXiv:1306.5151. 4, 6, 8, 16

[30] Otniel-Bogdan Mercea, Alexey Gritsenko, Cordelia Schmid, and Anurag Arnab. Time- memory- and parameter-efficient visual adaptation. In *CVPR*, 2024. 2

[31] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *IEEE Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 6

[32] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 6, 11, 16, 17, 20

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 3, 6, 12, 15, 17, 18, 20, 21, 22, 23

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, 2015. 1, 6, 7, 8, 16, 19

[35] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPRW*, 2014. 1

[36] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *NeurIPS*, 2022. 2, 11

[37] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6, 16, 19

[38] Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. uS-X: Training-Free Name-Only Transfer of Vision-Language Models. In *ICCV*, 2023. 2

[39] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *CVPR*, 2022. 4

[40] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 6, 16, 19

[41] Jialu Xing, Jianping Liu, Jian Wang, Lulu Sun, Xi Chen, Xunxun Gu, and Yingfei Wang. A survey of efficient fine-tuning methods for Vision-Language Models—Prompt and Adapter. *Computers & Graphics*, 2024. 2

[42] Lingxiao Yang, Ru-Yuan Zhang, Yanchen Wang, and Xiaohua Xie. MMA: Multi-Modal Adapter for Vision-Language Models. In *CVPR*, 2024. 1, 2, 4, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23

[43] Hantao Yao, Rui Zhang, and Changsheng Xu. Visual-language prompt tuning with knowledge-guided context optimization. In *CVPR*, 2023. 2, 6, 7, 11, 12, 13, 14, 16, 17, 18

[44] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NeurIPS*, 2014. 1

[45] Tao Yu, Zhihe Lu, Xin Jin, Zhibo Chen, and Xinchao Wang. Task residual for tuning vision-language models. In *CVPR*, 2023. 2, 7, 13, 16, 17, 18

[46] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *ACL*, 2022. 2, 4, 19

[47] Maxime Zanella and Ismail Ben Ayed. Low-Rank Few-Shot Adaptation of Vision-Language Models. In *CVPRW*, 2024. 1, 2, 4, 6, 7, 8, 11, 12, 13, 14, 16, 17, 18

[48] Maxime Zanella and Ismail Ben Ayed. On the test-time zero-shot generalization of vision-language models: Do we really need prompt learning? In *CVPR*, 2024. 2

[49] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark, 2019. arXiv:1910.04867. 1

[50] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-Adapter: Training-free Adaption of CLIP for Few-shot Classification. In *ECCV*, 2022. 2, 7, 13, 16, 17, 18

[51] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022. 2, 4, 6, 7, 11, 12, 13, 17, 18

[52] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 2022. 2, 6, 7, 11, 12, 13, 14, 17, 18

[53] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *ICCV*, 2023. 2, 6, 7, 12, 13, 16, 17, 18