Black Box Few-Shot Adaptation for Vision-Language models

Yassine Ouali1 Adrian Bulat1 Brais Matinez1 Georgios Tzimiropoulos1,2
1Samsung AI Cambridge 2Queen Mary University of London
{y.ouali, brais.a}@samsung.com, adrian@adrianbulat.com, g.tzimiropoulos@qmul.ac.uk

Abstract

Vision-Language (V-L) models trained with contrastive learning to align the visual and language modalities have been shown to be strong few-shot learners. Soft prompt learning is the method of choice for few-shot downstream adaptation aiming to bridge the modality gap caused by the distribution shift induced by the new domain. While parameter-efficient, prompt learning still requires access to the model weights and can be computationally infeasible for large models with billions of parameters. To address these shortcomings, in this work, we describe a black-box method for V-L few-shot adaptation that (a) operates on pre-computed image and text features and hence works without access to the model weights, (b) it is orders of magnitude faster at training time, (c) it is amenable to both supervised and unsupervised training, and (d) it can be even used to align image and text features computed from uni-modal models. To achieve this, we propose Linear Feature Alignment (LFA), a simple linear approach for V-L re-alignment in the target domain. LFA is initialized from a closed-form solution to a least-squares problem and then it is iteratively updated by minimizing a re-ranking loss. Despite its simplicity, our approach can even surpass soft-prompt learning methods as shown by extensive experiments on 11 image and 2 video datasets.

Code available at: https://github.com/saic-fi/LFA

1. Introduction

Large-scale Vision-Language (V-L) models [60] trained with contrastive learning currently represent the de-facto approach for few-shot visual adaptation. Their unprecedented success lies in part in the strength of the joint V-L embedding space learned by aligning the image and text modalities. However, when a V-L model is applied to a new domain, the domain shift exacerbates the V-L modality gap [48], and some sort of adaptation is required to obtain high accuracy (see Fig. 1(a)). The question that we want to address in this paper is: “can we effectively adapt a V-L model to a new domain by having access to pre-computed features only?” We call this black-box adaptation.

Similar to their NLP counterparts [60, 44], soft prompt learning has emerged as the preferred technique for adapting a V&L to new tasks. Specifically, a number of works [86, 85, 11, 87, 15, 67, 51, 36] have proposed to replace the manually designed prompts of [60] (e.g., a photo of a {cls_name}), with a sequence of learnable vectors, coined soft prompts. These are passed as input to the text encoder jointly with the class name cls_name to create the new prototypes effectively reducing the modality gap. The prompts are learned in a supervised manner using a standard cross-entropy loss given a set of labeled images.

While soft-prompt learning approaches demonstrate promising results on various downstream tasks [86, 85, 11], they suffer from two limitations: (1) They require access to the model’s weights, and (2) the training cost can be prohibitive, especially on commodity hardware and low-power devices, as computing the gradients and updating the prompts for thousands of iterations [86] is required. As the model’s size continues to grow (e.g., billion-parameter models such as CoCa [80]), and the industry transitions to models as a service (e.g., via API), the existing methods can be rendered either inapplicable or impractical.

In this work, we seek to address these limitations by bridging the modality gap directly in the feature space without prompting or access to the model’s weights. We first empirically show that a simple linear transformation can approximate the alignment effect of prompt learning (e.g., see Fig. 1 and Sec. 3). Importantly, this shows that it is possible to derive a black-box method that manipulates the CLIP features directly for downstream adaptation. Then, motivated by this observation, we propose Linear Feature Alignment (LFA), a black-box method that learns a linear mapping \( \mathbf{W} \), obtained by solving a simple optimization problem, which effectively aligns the image features \( \mathbf{X} \) with their text class prototypes \( \mathbf{Y} \), \( i.e., \mathbf{X} \xrightarrow{\mathbf{W}} \mathbf{Y} \). Specifically, our contributions are:

- We propose the very first black-box method for the few-shot adaptation of V-L models.
- To this end, and motivated by the observation that prompting can be successfully approximated by a lin-
Figure 1: **Effect of Linear Feature Alignment (LFA):** We use 16-shot (per class) training data for two fine-grained image classification datasets: DTD and FGVC Aircraft. In (1), we show the training set modality gap between paired image embeddings and class prototypes following the same procedure as in [48], and the obtained test set accuracy. The embeddings are visualized in 2D using PCA. (a) With CLIP features, we observe a big modality gap, resulting in low test accuracy. (b) After learning a set of soft-prompts, we obtain a better alignment and improved results. However, the modality gap is still not sufficiently reduced. (c) A simple linear transformation $W$ that maps the original class prototypes (obtained using only the class names) to the ones obtained with soft-prompt learning induces a similar modality gap. (d) Motivated by (c) we propose LFA, which aligns the image embeddings with their class prototypes via linear mapping $W$, obtained by solving a simple optimization problem. LFA results in better alignment and improved accuracy. In (2), we show that with LFA, the test image features are closely aligned with their corresponding class prototypes, resulting in higher cosine similarity scores compared to the ones obtained with soft prompts.

Table 1: **Training Time:** train time for CoOp [86] and for the proposed LFA on ImageNet (16-shot) using ViT-B/16 as the visual encoder on a single V100 GPU.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time</th>
<th>Test Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoOp</td>
<td>3h 22min</td>
<td>71.92</td>
</tr>
<tr>
<td>LFA (Feature Extraction)</td>
<td>2min 37s</td>
<td></td>
</tr>
<tr>
<td>LFA (Procrustes Initialisation)</td>
<td>4s</td>
<td></td>
</tr>
<tr>
<td>LFA (Refinement)</td>
<td>28s</td>
<td></td>
</tr>
<tr>
<td>LFA (Total)</td>
<td>3min 9s</td>
<td>72.61</td>
</tr>
</tbody>
</table>

2. **Related Work**

**Vision-Language (V-L) Models:** Recently, we have witnessed an explosion of research in V-L foundation models, including CLIP [60], ALIGN [38], Florence [81], LiT [83], BASIC [59], ALBEF [46] and CoCa [80]. Such models are pre-trained on large amounts of image and text data to learn a joint multi-modal embedding space. After pre-training, they can be used on various downstream tasks in a few- or
zero-shot setting. For our work, we used CLIP [60] to extract the frozen image and text features.

Learnable Prompts for V-L Models: Despite pre-training, V-L models still suffer from a modality gap [48] which is further exacerbated during downstream adaptation. To address this issue, recently, soft prompt learning methods [86, 85, 11, 87, 15, 67, 51, 36] optimize a new set of learnable (soft) prompts to reduce the gap and align the visual and text modalities. CoOp [86] was the first method to apply prompt learning methods [47, 44, 30, 66] popularized in NLP to V-L models. Subsequent works have improved upon this by incorporating image conditioning for better generalization [85], test-time adaptation [67], gradient matching [67], or by using an additional text-to-text loss [11]. In contrast to all the aforementioned methods, we propose to bridge the domain gap for a given downstream task directly in the feature space, without requiring access to the model’s weights nor expensive training procedures.

Linear Alignment: The problem of linearly aligning two sets of embeddings or high-dimensional real vectors is a well-studied problem in machine learning, with various applications in computer vision and NLP. Classical applications range from sentence classification [26, 62], to shape and motion extraction [72], registration [19] and geometrical alignment [24, 43, 49]. In vision, linear mappings are widely used for zero-shot learning [25, 1, 2, 64] for aligning the image features and their class attributes. In NLP, and after the introduction of word embeddings [53, 9], this linear alignment problem was revisited and extensively studied, and improved upon for the task of word translation [3, 27, 35, 18, 79, 54, 13, 39, 84, 68, 7, 6, 5, 4]. In this paper, we take strong inspiration from this line of work and set to adapt them for the case of V-L models.


Herein, we empirically show that the V-L alignment effect achieved by prompt learning can be approximated by finding a simple linear transformation $W \in \mathbb{R}^{d \times d}$ that maps the class prototypes computed from the class names only (i.e., the class name text embeddings) to the ones obtained by soft prompt learning. To demonstrate this, let $Y \in \mathbb{R}^{N \times d}$ be the class name text embeddings represented in matrix form, and similarly, let $Y' \in \mathbb{R}^{N \times d}$ be the class prototypes obtained by soft prompt learning. Our objective is to learn a linear transformation $W \in \mathbb{R}^{d \times d}$ that tries to approximate prompt learning, i.e., $Y \xrightarrow{W} Y'$, by solving the following least square problem:

$$\min_{W \in \mathbb{R}^{d \times d}} \|YW - Y'\|_F^2,$$  \hspace{1cm} (1)

where $\| \cdot \|_F$ is the Frobenius norm.

If the classification results are consistent when using either $YW$ or $Y'$ as class prototypes, then we could use $YW$ to approximate the V-L realignment achieved by prompt optimization. As shown in Fig. 1 (1b) and (1c), $W$ can almost perfectly approximate the effects of prompt learning resulting in the same test set accuracy (i.e., same accuracy on DTD and 39.9 vs. 40.1 on FGVC Aircraft).

Note that, in practice, we want to avoid the training of the soft prompts. To this end, we can simply attempt to learn a linear transformation directly from the image features to the class name text embeddings. This is the main idea behind the proposed Linear Feature Alignment (LFA) which directly finds a linear transformation for image-text alignment.

4. Linear Feature Alignment

Our objective is to learn a linear mapping $W$ for aligning the image embeddings $X$ with their corresponding text class prototypes $Y$, i.e., $X \xrightarrow{W} Y$. Once $W$ is learned, in order to classify a new sample $x$, we obtain its $C$-way class probabilities from $\text{softmax}(xW \cdot Y^\top / \tau)$ with $\tau$ being the temperature parameter. To learn $W$, LFA firstly uses for initialization a closed-form solution to a least-squares optimization problem, then minimizes a re-ranking loss to refine the initial solution. LFA is described in detail in the following sections.

4.1. Problem Formulation

Let $X \in \mathbb{R}^{N \times d}$ be the image embeddings of $N$ examples produced by the CLIP image encoder, and let $Y \in \mathbb{R}^{C \times d}$ be the $C$ class prototypes corresponding to the encoded class names using CLIP text encoder (i.e., without any prompts). Moreover, let $P \in \mathcal{P}_{N \times C}$ be an assignment matrix that assigns each class prototype to its corresponding image embedding with $\mathcal{P}_{N \times C} = \{ P \in \{0,1\}^{N \times C}, P1_C = 1_N \}$ as the set of binary permutation matrices that map each one of the $N$ rows to one of the $C$ columns, i.e., the input images to their corresponding classes. In a supervised setting where we are provided with $N$ (image-class) pairs, $P$ is the stacked $N.C$-dimensional one-hot vectors.

Our objective is to find an optimal linear mapping that bridges the modality gap and aligns each image embedding with its text class prototype. To this end, the linear mapping can be learned by solving the following least squares:

$$\arg\min_{W \in \mathbb{R}^{d \times d}} \|XW - PY\|_F^2.$$

(2)

This is the standard Procrustes analysis formulation that aims to find a linear transformation between two sets of points $X$ and $PY$. 

15536
After refinement, the orthogonality constraint on $W$, Eq. (2) becomes an Orthogonal Procrustes (OP) analysis optimization problem:

$$W_\text{op} = \underset{W \in O_d}{\text{argmin}} \|XW - PY\|_F^2,$$

with $O_d = \{W \in \mathbb{R}^{d \times d}, W^\top W = I_d\}$ as the set of orthogonal matrices and $I_d$ as the $d$-dimensional identity matrix. As shown in [65], under this constraint, Eq. (3) admits a closed-form solution from the singular value decomposition (SVD) of $X^\top PY$:

$$W_\text{op} = \underset{W \in O_d}{\text{argmin}} \|XW - PY\|_F^2 = UV^\top,$$

with $\text{SVD}(X^\top PY) = U\Sigma V^\top$. Moreover, under the orthogonality constraint, the obtained mapping preserves the vector dot product and their $\ell_2$ distances, thus making it suitable for V-L models trained with a contrastive loss.

### 4.3. $\beta$-Procrustes

The orthogonal Procrustes solution is efficient and easy to compute, however, it suffers from extreme overfitting, especially if the initial modality gap is small. On ImageNet, for instance, and as shown in Fig. 2 and Tab. 2, the orthogonal Procrustes solution results in overly entangled class prototypes and a lower test accuracy than the original CLIP features ($i.e., 62.8 \rightarrow 52.5$). To solve this, we propose $\beta$-Procrustes, a regularized Procrustes solution that is pushed to be close to an identity mapping via the following update:

$$W_\beta \leftarrow W_\text{op} - \beta(W_\text{op} - I_d),$$

where $\beta \in [0, 1]$ is an interpolation hyperparameter between an identity mapping ($\beta = 1$) and the orthogonal solution of Eq. (4) ($\beta = 0$). This update is equivalent to a simple gradient descent step of the regularization term $R_\beta(W_\text{op}) = \frac{\beta}{2}\|W_\text{op} - I_d\|_F^2$, i.e. $\nabla W_\text{op} R_\beta(W_\text{op}) = \beta(W_\text{op} - I_d)$. As shown in Fig. 2 and Tab. 2, this simple update results in better alignment and improved test accuracy. For the choice of the hyperparameter $\beta$, it can be found via cross-validation on the training set or set to a fixed value with $\beta \in [0.6, 0.9]$ without a significant impact on the results. The hyperparameter $\beta$ can be determined through cross-validation on the training set or set to a fixed value $\beta \in [0.6, 0.9]$ without significantly impacting the results.

### 4.4. Mapping Refinement

While $\beta$-Procrustes improves the results, they are still not on par with those obtained with the soft prompts. To investigate why, in Fig. 3, we show the rank of the ground-truth class prototypes of different training examples. We see that even after the $\beta$-Procrustes alignment step, the embedding space still contains a high number of hubs. After a refinement step, we reduce the hubness and obtain better image-class prototype alignment than soft-prompts.

<table>
<thead>
<tr>
<th>Method</th>
<th>ImageNet</th>
<th>Aircraft</th>
<th>DTD</th>
<th>Food101</th>
<th>Caltech101</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP ViT-B/16</td>
<td>62.8</td>
<td>22.1</td>
<td>45.1</td>
<td>83.9</td>
<td>88.0</td>
</tr>
<tr>
<td>Procrustes</td>
<td>52.5</td>
<td>27.9</td>
<td>63.4</td>
<td>78.8</td>
<td>93.4</td>
</tr>
<tr>
<td>$\beta$-Procrustes</td>
<td>64.8</td>
<td>29.1</td>
<td>65.8</td>
<td>85.5</td>
<td>94.7</td>
</tr>
</tbody>
</table>

**Figure 2:** Effect of $\beta$-Procrustes: By pushing the orthogonal Procrustes solution of Eq. (4) towards an identity mapping via the update rule in Eq. (5), we avoid the overfitting exhibited using the original solution and obtain better alignment ($i.e.$ as suggested by the observed class prototypes-image embeddings cross-interference). Here, each class prototype and its image embeddings share the same color and the embeddings shown are from 50 randomly sampled ImageNet classes.

**Figure 3:** Hubness: We show the rank of the ground-truth class prototypes of different training examples. We see that even after the $\beta$-Procrustes alignment step, the embedding space still contains a high number of hubs. After a refinement step, we reduce the hubness and obtain better image-class prototype alignment than soft-prompts.
in many retrieval cases [8, 37], and is often caused by the 
hubness problem [61, 22, 18, 39]. Hubs are points (e.g.,
class prototypes) in the high dimensional vector space that 
are the nearest neighbors of many other points (e.g., image 
embeddings), and as a result, they greatly influence the clas-
sification probabilities (and thus the accuracy at test time).

To mitigate this effect, and inspired by popular met-
ric learning losses [29, 70, 75, 76, 12], we propose to re-
fine the mapping \( W \) by optimizing an Adaptive Reranking
(ARerank) loss designed specifically to reduce the hubness 
problem, and defined as follows:

\[
L_{\text{ARerank}}(x, W, Y) = \frac{1}{k} \sum_{y_j \in \mathcal{N}_k(x, W)} \ell_{ij} 
\]

where \( \ell_{ij} = \max\{d_{ii} - d_{ij} + m_{ij}, 0\} \),

where \( d_{ii} = \|x_i W - y_{c_i}\|_2 \) is the \( \ell_2 \) distance between the 
aligned image embedding \( x_i \) and its class prototype \( y_{c_i} \),
and similarly, \( d_{ij} = \|x_i W - y_j\|_2 \) the \( \ell_2 \) distance between the 
aligned image embedding and each of its \( k \) nearest class 
prototypes \( y_j \in \mathcal{N}_k(x, W) \), and \( m_{ij} \) the margin. Empiri-

cally, we found that \( k = 3 \) works well for most datasets.

To make the re-ranking dynamic and avoid having multi-
ple hyperparameters, we opt for an adaptive margin selection
approach similar to [45]. Specifically, the margin between 
image \( i \) and a given class prototype \( j \) is defined based on the cosine similarity between its class prototype \( y_{c_i} \) and the 
\( j \)-th class prototype, i.e., \( m_{ij} = (1.0 - y_{c_i}^T y_j)/s \), where \( s \) is 
a scalar set to 4 for all experiments. By doing so, we 
ensure that each image embedding is pushed away from 
nearby incorrect class prototypes with an adaptive margin,
thus mitigating the hubness problem and avoiding learning 
a degenerate mapping. As shown in Tab. 3, ARerank loss
outperforms standard embedding optimization losses, and 
also demonstrates better results than the CSLS criterion
proposed by [18] used to reduce hubness for word translation. 
Finally, as shown in Tab. 4, the coupling of \( \beta \)-Procrustes 
and ARerank-based refinement results in better accuracy.

4.5. Overall LFA algorithm

Herein, we define the overall algorithm obtained by com-
bining the steps defined in the previous sections. We con-
sider two cases: supervised learning, where labeled data is
provided, and unsupervised learning, where only unlabeled
images are available.

**Supervised Alignment:** In a supervised setting, we can di-
rectly construct the assignment matrix \( P \) between the image 
embeddings \( X \) and class prototypes \( Y \) using the ground-
truth data. The overall algorithm can be then defined as
(see also Alg. 1):

1. \( W_{op} \leftarrow \text{Orthogonal Procrustes } [X, PY] \) (Eq. (4)).
2. \( W_\beta \leftarrow \beta \text{-Procrustes} [W_{op}] \) (Eq. (5)).
3. \( W \leftarrow \text{Refine } [W_\beta, X, PY] \) (Eq. (6)).

**Unsupervised Alignment:** In an unsupervised setting, the 
correspondences between the image embeddings and their 
class prototypes are not known a priori. Thus the assign-
ment matrix \( P \) must be estimated jointly with learning the 
mapping \( W \). By keeping the orthogonality constraint 
over \( W \) and solving for both \( P \) and \( W \), the resulting opti-


\[
\begin{align*}
\mathbf{W}^*, \mathbf{P}^* = \arg\min_{\mathbf{W} \in O_d, \mathbf{P} \in \mathcal{P}_{N \times C}} & \| \mathbf{XW} - \mathbf{PY} \|^2_2 \\
\end{align*}
\]

As neither of the two sets \( \mathcal{O}_d \) and \( \mathcal{P}_{N \times C} \) are convex, this opti-


\[\text{Table 3: Refinement Loss: Top-1 acc. for 16-shot per class.}\]

<table>
<thead>
<tr>
<th>Refinement Loss</th>
<th>ImageNet</th>
<th>Aircraft</th>
<th>DTD</th>
<th>Food101</th>
<th>Caltech101</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP ViT-B/16</td>
<td>62.8</td>
<td>22.1</td>
<td>45.1</td>
<td>83.9</td>
<td>88.0</td>
</tr>
<tr>
<td>Contrastive</td>
<td>65.5</td>
<td>40.1</td>
<td>71.5</td>
<td>85.7</td>
<td>92.2</td>
</tr>
<tr>
<td>Triplet with margin</td>
<td>70.7</td>
<td>43.5</td>
<td>72.6</td>
<td>86.9</td>
<td>95.9</td>
</tr>
<tr>
<td>CSLS</td>
<td>71.1</td>
<td>40.9</td>
<td>72.2</td>
<td>86.8</td>
<td>95.9</td>
</tr>
<tr>
<td>ARerank</td>
<td>71.7</td>
<td>45.1</td>
<td>72.7</td>
<td>87.5</td>
<td>95.9</td>
</tr>
</tbody>
</table>

\[\text{Table 4: } \beta \text{-Procrustes & Mapping Refinement: Top-1 acc. for 16-shot per class.}\]

<table>
<thead>
<tr>
<th>Method</th>
<th>ImageNet</th>
<th>Aircraft</th>
<th>DTD</th>
<th>Food101</th>
<th>Caltech101</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP → Refine</td>
<td>71.6</td>
<td>43.4</td>
<td>72.6</td>
<td>87.2</td>
<td>95.9</td>
</tr>
<tr>
<td>CLIP → Proc → Refine</td>
<td>70.7</td>
<td>44.8</td>
<td>72.4</td>
<td>85.3</td>
<td>95.7</td>
</tr>
<tr>
<td>CLIP → β-Proc → Refine</td>
<td>71.7</td>
<td>45.1</td>
<td>72.7</td>
<td>87.5</td>
<td>95.9</td>
</tr>
</tbody>
</table>

\[\text{Algorithm 1 Linear Feature Alignment (LFA)}\]

```python
def LFA(img_feats, cls_prototypes, labels, beta, test_img_features):
    # M: number of test image features
    # N: number of training image features
    # C: number of classes
    # d: features dimensionality
    # W: number of test image features
    identity = torch.eye(d)
    W_op = u @ v.T
    W = adaptive_rerank_refine(W_beta)
    test_logits = (test_img_features @ W) @ cls_prototypes.T
    test_preds = test_logits.argmax(-1)
    return test_preds
```
Table 5: U-LFA: Top-1 acc. for 16-shot per class (without labels):

<table>
<thead>
<tr>
<th>Refinement Loss</th>
<th>ImageNet</th>
<th>Aircraft</th>
<th>DTD</th>
<th>Food101</th>
<th>Caltech101</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP ViT-B/16</td>
<td>62.8</td>
<td>22.1</td>
<td>45.1</td>
<td>83.9</td>
<td>88.0</td>
</tr>
<tr>
<td>U-LFA (n = 1)</td>
<td>66.9</td>
<td>24.5</td>
<td>48.0</td>
<td>86.7</td>
<td>94.4</td>
</tr>
<tr>
<td>U-LFA (n = 5)</td>
<td>68.6</td>
<td>24.2</td>
<td>47.5</td>
<td>86.1</td>
<td>95.1</td>
</tr>
</tbody>
</table>

*with template: a photo of a {cls name}.

Table 6: LFA analysis under distribution shift: Top-1 acc. for 16 shots per class on the Base (B) and New (N) sets and two IN variants. IN refers to ImageNet.

<table>
<thead>
<tr>
<th>Method</th>
<th>Language Shift</th>
<th>Image Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ImageNet</td>
<td>Aircraft</td>
</tr>
<tr>
<td>LFA</td>
<td>76.9</td>
<td>67.1</td>
</tr>
<tr>
<td>LFA (average)</td>
<td>76.8</td>
<td>68.6</td>
</tr>
<tr>
<td>LFA (two mappings)</td>
<td>76.9</td>
<td>69.4</td>
</tr>
</tbody>
</table>

Table 7: Augmentations: average Top-1 acc. for 16-shot per class on 11 datasets using 5 crops per image, with and without a prompt template.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>RN50</th>
<th>RN101</th>
<th>ViT-B/32</th>
<th>ViT-B/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFA</td>
<td>74.20</td>
<td>76.83</td>
<td>76.69</td>
<td>80.54</td>
</tr>
<tr>
<td>LFA + 5 crops</td>
<td>74.49</td>
<td>77.05</td>
<td>76.95</td>
<td>80.88</td>
</tr>
<tr>
<td>LFA + 5 crops + template</td>
<td>74.75</td>
<td>77.14</td>
<td>77.17</td>
<td>81.21</td>
</tr>
</tbody>
</table>

Finding the assignments $P$ using the efficient Sinkhorn algorithm [20] and refining the mapping following Eq. (6). Given this, Unsupervised LFA (U-LFA) takes the following form:

1. $P \leftarrow$ Sinkhorn $[X, Y]$.
2. $W_{op} \leftarrow$ Orthogonal Procrustes $[X, PY]$.
3. $W \leftarrow \beta$-Procrustes$[W_{op}]$.
4. Repeat for $n$ iterations:
   (a) $P \leftarrow$ Sinkhorn $[XW, Y]$.
   (b) $W \leftarrow$ Refine $[W, X, PY]$.

4.6. LFA for Base-to-New (Zero-Shot) Recognition

An important property of large-scale V-L models that recent few-shot adaptation methods seek to preserve is their zero-shot generalization ability, i.e., generalization from seen (base) classes to unseen (new) classes. Training LFA in this setting, i.e., on the base set, may result in a mapping that fails to generalise to the new set due to the distribution shift in between the two.

To address this, starting from a task-specific mapping $W$, during the iterative refinement step, we initialize a second $W_{tt}$ as an identity map $I_d$. At each optimization step $t$ of the refinement procedure, we then update $W_{tt}$ using $W$ with an exponential moving average as follows:

$$W_{tt}(t) \leftarrow \alpha(t)W_{tt}(t) + (1 - \alpha(t))W(t)$$  \hspace{1cm} (8)

with $\alpha(t) \in [0, 1]$ as the momentum parameter, which is initialized as 0.9, and is increased to 1.0 following a log schedule during the first half of the optimization. This way, we only incorporate the first refinement updates into $W_{tt}$, while the later ones, which tend to be more task-specific and may hinder generalization are largely ignored. At test time, $W$ can be used on the base classes, while $W_{tt}$ for the new classes. As shown in Tab. 6, this maintains the good accuracy on the base training domain, while demonstrating good generalization when a distribution shift occurs (i.e., on novel classes). Additionally, using a single mapping, obtained by taking the average of $W$ and $W_{tt}$ also achieves good results (see Tab. 6).

5. Experiments

Datasets & Evaluation settings: For image classification, we consider the following evaluation protocols and settings: (1) standard few-shot classification, as in [86], (2) generalisation from base-to-new classes, where the model is trained in a few-shot manner on the base classes and tested on a disjoint set of new classes, as in [85], and finally, (3) domain generalisation, where the model is trained on training set of ImageNet and is then tested on one of the four ImageNet variants with some form of distribution shift, as in [85, 86]. For standard few-shot evaluation and generalisation from base-to-new classes, we report results on the 11 datasets used in CoOp [86]: ImageNet [21], Caltech101 [23], OxfordPets [57], Stanford-Cars [41], Flowers102 [56], Food101 [10], FGVC-Aircraft [52], SUN397 [77], UCF101 [69], DTD [17] and EuroSAT [32]. For domain generalisation, we follow previous work [86, 85] and report the classification results on four ImageNet variants: ImageNetV2 [63], ImageNet-Sketch [74], ImageNet-A [34] and ImageNet-R [33].

For action recognition, we align our setting with [40] and consider both standard (i.e., using the full training set for adaptation) and few-shot classification settings and on two datasets, HMDB51 [42] and UCF101 [69]. To get the video features to be aligned their class prototypes, we take the max aggregate on the per-frame CLIP features.

Implementation Details: Unless stated otherwise, we base our experiments on a pre-trained CLIP model [60]. For each experiment, we pre-compute and save the image features alongside the class prototypes and follow the adaptation procedure as described in Section 4.5. The class prototypes are formed by inserting the class name in the standard

---

1Both image classification and action recognition experiments use UCF101, but take a single frame and a video segment respectively as input.
Table 8: Few-shot Classification: Top-1 acc. for 16-shot per class when using CLIP, CoOp [86] and LFA, with either RN50, RN101, ViT-B/32 or ViT-B/16 as the vision encoder.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pets</th>
<th>Flowers102</th>
<th>Aircraft</th>
<th>DTD</th>
<th>EuroSAT</th>
<th>Cars</th>
<th>Food101</th>
<th>SUN397</th>
<th>Caltech101</th>
<th>UCF101</th>
<th>ImageNet</th>
<th>Avg.</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP RN50</td>
<td>85.77</td>
<td>66.14</td>
<td>17.28</td>
<td>42.32</td>
<td>37.56</td>
<td>55.61</td>
<td>77.31</td>
<td>58.52</td>
<td>69.71</td>
<td>71.66</td>
<td>63.65</td>
<td>74.75</td>
<td>+1.40</td>
</tr>
<tr>
<td>CoOp</td>
<td>86.16</td>
<td>94.80</td>
<td>32.29</td>
<td>63.16</td>
<td>83.55</td>
<td>73.27</td>
<td>74.46</td>
<td>91.12</td>
<td>59.72</td>
<td>75.29</td>
<td>63.08</td>
<td>73.35</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td>86.75</td>
<td>94.56</td>
<td>35.86</td>
<td>66.35</td>
<td>84.13</td>
<td>73.58</td>
<td>76.32</td>
<td>71.32</td>
<td>92.68</td>
<td>77.00</td>
<td>63.65</td>
<td>74.75</td>
<td>+1.40</td>
</tr>
<tr>
<td>CLIP RN101</td>
<td>86.75</td>
<td>64.03</td>
<td>18.42</td>
<td>38.59</td>
<td>32.59</td>
<td>66.23</td>
<td>80.53</td>
<td>58.96</td>
<td>89.78</td>
<td>60.96</td>
<td>61.62</td>
<td>59.86</td>
<td></td>
</tr>
<tr>
<td>CoOp</td>
<td>88.57</td>
<td>95.19</td>
<td>34.76</td>
<td>65.47</td>
<td>83.54</td>
<td>79.74</td>
<td>79.08</td>
<td>71.19</td>
<td>93.42</td>
<td>77.95</td>
<td>66.60</td>
<td>75.96</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td>88.80</td>
<td>93.11</td>
<td>39.62</td>
<td>68.85</td>
<td>83.88</td>
<td>76.72</td>
<td>81.31</td>
<td>74.15</td>
<td>90.81</td>
<td>80.81</td>
<td>67.63</td>
<td>77.17</td>
<td>+1.18</td>
</tr>
<tr>
<td>CLIP ViT-B/32</td>
<td>87.49</td>
<td>66.95</td>
<td>19.23</td>
<td>43.97</td>
<td>45.19</td>
<td>60.55</td>
<td>80.50</td>
<td>61.91</td>
<td>90.87</td>
<td>62.01</td>
<td>62.05</td>
<td>61.88</td>
<td></td>
</tr>
<tr>
<td>CoOp</td>
<td>88.68</td>
<td>94.97</td>
<td>33.22</td>
<td>65.37</td>
<td>83.43</td>
<td>76.08</td>
<td>78.45</td>
<td>72.38</td>
<td>94.62</td>
<td>78.66</td>
<td>66.85</td>
<td>75.70</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td>88.62</td>
<td>93.84</td>
<td>38.01</td>
<td>68.87</td>
<td>83.88</td>
<td>76.72</td>
<td>81.31</td>
<td>74.15</td>
<td>90.81</td>
<td>80.81</td>
<td>67.63</td>
<td>77.17</td>
<td>+1.47</td>
</tr>
<tr>
<td>CLIP ViT-B/16</td>
<td>89.21</td>
<td>71.34</td>
<td>24.72</td>
<td>44.39</td>
<td>47.60</td>
<td>65.32</td>
<td>86.06</td>
<td>62.50</td>
<td>92.94</td>
<td>66.75</td>
<td>66.73</td>
<td>65.23</td>
<td></td>
</tr>
<tr>
<td>CoOp</td>
<td>92.53</td>
<td>96.47</td>
<td>42.91</td>
<td>68.50</td>
<td>80.87</td>
<td>83.09</td>
<td>87.21</td>
<td>75.29</td>
<td>95.77</td>
<td>82.24</td>
<td>71.92</td>
<td>79.71</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td>92.41</td>
<td>96.82</td>
<td>46.01</td>
<td>71.89</td>
<td>87.31</td>
<td>82.23</td>
<td>87.14</td>
<td>76.65</td>
<td>96.24</td>
<td>83.99</td>
<td>72.61</td>
<td>81.21</td>
<td>+1.50</td>
</tr>
</tbody>
</table>

Table 9: Out-of-Domain Generalization: the obtained average Top-1 acc. on various ImageNet variants after training on ImageNet (i.e., source) using 16-shot training data per class and using RN50 as the visual encoder. We show the results for the baseline CLIP [60], CoOp [86], CoCoOp [85], and the proposed LFA on each dataset, their average and the Out-of-Distribution (OOD) average (i.e., over the target datasets).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP-RN50</td>
<td></td>
<td>LFA</td>
<td>58.16</td>
<td>21.83</td>
<td>51.41</td>
<td>56.15</td>
<td>33.37</td>
<td>44.18</td>
<td>40.69</td>
<td></td>
</tr>
<tr>
<td>CoOp</td>
<td></td>
<td>CoOp</td>
<td>63.33</td>
<td>23.06</td>
<td>55.40</td>
<td>56.60</td>
<td>34.67</td>
<td>46.61</td>
<td>42.43</td>
<td></td>
</tr>
<tr>
<td>CoCoOp</td>
<td></td>
<td>CoCoOp</td>
<td>62.81</td>
<td>23.32</td>
<td>55.72</td>
<td>57.74</td>
<td>34.48</td>
<td>46.81</td>
<td>42.82</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td></td>
<td>LFA</td>
<td>63.88</td>
<td>24.31</td>
<td>55.79</td>
<td>58.13</td>
<td>34.37</td>
<td>47.29</td>
<td>43.15</td>
<td>+0.32</td>
</tr>
<tr>
<td>CLIP-ViT-B/16</td>
<td>66.73</td>
<td></td>
<td>47.87</td>
<td>60.86</td>
<td>73.98</td>
<td>46.09</td>
<td>59.11</td>
<td>57.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoOp</td>
<td></td>
<td>CoOp</td>
<td>71.51</td>
<td>49.71</td>
<td>64.20</td>
<td>75.21</td>
<td>47.99</td>
<td>61.72</td>
<td>59.28</td>
<td></td>
</tr>
<tr>
<td>CoCoOp</td>
<td></td>
<td>CoCoOp</td>
<td>71.02</td>
<td>50.63</td>
<td>64.07</td>
<td>76.18</td>
<td>48.75</td>
<td>62.13</td>
<td>59.91</td>
<td></td>
</tr>
<tr>
<td>LFA</td>
<td></td>
<td>LFA</td>
<td>72.65</td>
<td>51.50</td>
<td>64.72</td>
<td>76.09</td>
<td>48.01</td>
<td>62.59</td>
<td>60.08</td>
<td>+0.17</td>
</tr>
</tbody>
</table>

Table 10: Few-shot Action Recognition: Top-1 acc. for 8 and 16-shot per class. We compare against our implementation of Video Prompting [40], which is trained either with mean over the per-frame features (i.e., Temporal: ✓) or a single Transformer layer (i.e., Temporal: ✗).

<table>
<thead>
<tr>
<th>N-shot</th>
<th>Method</th>
<th>Soft-Prompt</th>
<th>Temporal</th>
<th>UCF-101</th>
<th>HMDB51</th>
<th>Avg.</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Video Prompting</td>
<td>✓</td>
<td>✗</td>
<td>64.7</td>
<td>40.1</td>
<td>52.40</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Video Prompting</td>
<td>✓</td>
<td>✗</td>
<td>76.22</td>
<td>53.90</td>
<td>65.06</td>
<td>+0.81</td>
</tr>
<tr>
<td>LFA</td>
<td>Video Prompting</td>
<td>✓</td>
<td>✗</td>
<td>89.47</td>
<td>65.08</td>
<td>77.24</td>
<td>+0.83</td>
</tr>
</tbody>
</table>

templates [86] (e.g., “a photo of a {cls name}” for image tasks and “a video frame of a person {action type}” for videos). As it is common practice to augment the images for prompting [85, 86], for each training image, we construct c = 5 random cropped views, noting that a large c is not crucial, as LFA still performs well without them (i.e., c = 1), as shown in Tab. 7.

Training Details: For standard image few-shot experiments, we set β based on cross-validation on the training set, while for the rest, we fix it to β = 0.9. For the refinement step, we set k = 3 for the ARerank loss, and finetune the mappings using AdamW [50] for 50-200 iterations using a learning rate of 5e-4, a weight decay of 5e-4, and a cosine scheduler. During refinement, we inject a small amount of Gaussian noise (i.e., std of 3.5e-2) and apply dropout (i.e., 2.5e-2) to the image embeddings to stabilize training. For few-shot experiments, we follow standard practices and report the average Top-1 accuracy over 3 runs. For additional details, refer the supplementary material.

Table 11: Action Recognition: the obtained Top-1 acc. on the test sets of UCF101 and HMDB51 using the full training sets. Video Prompting [40] is trained with two Transformer layers (i.e., Temporal: ✓) on top of the frozen per-frame CLIP features to model the temporal information in the input video segment. All results are obtained with ViT-B/16 as the visual encoder.

<table>
<thead>
<tr>
<th>Method</th>
<th>Soft-Prompt</th>
<th>Temporal</th>
<th>UCF-101</th>
<th>HMDB51</th>
<th>Avg.</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFA</td>
<td>✓</td>
<td>✓</td>
<td>66.4</td>
<td>93.6</td>
<td>80.0</td>
<td>+0.5</td>
</tr>
<tr>
<td>CoOp</td>
<td>✓</td>
<td>✗</td>
<td>59.2</td>
<td>91.8</td>
<td>80.5</td>
<td>+0.5</td>
</tr>
</tbody>
</table>
5.1. Comparisons with State-of-the-Art

Standard Few-shot Image Classification: As shown in Tab. 8, LFA outperforms CoOp [86] by 1% on average over the 11 datasets and with various visual backbones, with the largest gains observed on datasets with larger domain gaps, i.e., \( \approx 7\% \) on EuroSAT.

Base-to-New Generalisation: Using 2 mappings, and as shown in Tab. 12, LFA improves the prior best result of ProDA by 2.18% in terms of harmonic mean, with similar improvements for both base and new classes. Again, the largest gains are observed for datasets with larger domain gaps, such as UCF101 and EuroSAT.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Set</th>
<th>CLIP</th>
<th>CoOp</th>
<th>CoCoOp</th>
<th>ProDA</th>
<th>LFA</th>
<th>∆</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>Base</td>
<td>72.43</td>
<td>76.47</td>
<td>75.98</td>
<td>75.40</td>
<td>76.89</td>
<td>+1.78</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>68.14</td>
<td>67.88</td>
<td>70.43</td>
<td>70.23</td>
<td>69.36</td>
<td>-0.40</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>70.22</td>
<td>71.92</td>
<td>73.10</td>
<td>72.72</td>
<td>72.93</td>
<td>-0.17</td>
</tr>
<tr>
<td>Caltech101</td>
<td>Base</td>
<td>96.84</td>
<td>98.00</td>
<td>97.96</td>
<td>98.27</td>
<td>98.41</td>
<td>+0.50</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>94.00</td>
<td>89.91</td>
<td>93.81</td>
<td>93.23</td>
<td>93.93</td>
<td>+0.13</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>95.40</td>
<td>93.73</td>
<td>95.84</td>
<td>95.86</td>
<td>96.13+0.27</td>
<td></td>
</tr>
<tr>
<td>Pets</td>
<td>Base</td>
<td>91.17</td>
<td>93.67</td>
<td>95.20</td>
<td>95.43</td>
<td>95.13</td>
<td>-0.18</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>97.26</td>
<td>95.29</td>
<td>97.69</td>
<td>97.83</td>
<td>96.23</td>
<td>-1.56</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>94.12</td>
<td>94.47</td>
<td>96.43</td>
<td>96.62</td>
<td>95.68</td>
<td>-0.94</td>
</tr>
<tr>
<td>Cars</td>
<td>Base</td>
<td>63.37</td>
<td>78.12</td>
<td>70.49</td>
<td>74.70</td>
<td>76.32</td>
<td>+1.83</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>74.89</td>
<td>60.40</td>
<td>73.59</td>
<td>71.20</td>
<td>74.88</td>
<td>+3.02</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>68.85</td>
<td>68.13</td>
<td>72.01</td>
<td>72.91</td>
<td>75.59+2.68</td>
<td></td>
</tr>
<tr>
<td>Flowers102</td>
<td>Base</td>
<td>72.08</td>
<td>97.60</td>
<td>94.87</td>
<td>97.70</td>
<td>97.34</td>
<td>-0.36</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>77.80</td>
<td>59.67</td>
<td>71.75</td>
<td>68.68</td>
<td>75.44</td>
<td>+2.51</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>74.83</td>
<td>74.06</td>
<td>81.71</td>
<td>80.66</td>
<td>85.00+3.29</td>
<td></td>
</tr>
<tr>
<td>Food101</td>
<td>Base</td>
<td>90.10</td>
<td>88.33</td>
<td>90.70</td>
<td>90.30</td>
<td>90.52</td>
<td>-0.25</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>91.22</td>
<td>82.26</td>
<td>91.29</td>
<td>88.57</td>
<td>91.48</td>
<td>+1.52</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>90.66</td>
<td>85.19</td>
<td>90.99</td>
<td>89.43</td>
<td>91.00+0.0</td>
<td></td>
</tr>
<tr>
<td>Aircraft</td>
<td>Base</td>
<td>27.19</td>
<td>40.44</td>
<td>33.41</td>
<td>36.90</td>
<td>41.48</td>
<td>+2.80</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>36.29</td>
<td>22.33</td>
<td>23.71</td>
<td>34.13</td>
<td>32.29</td>
<td>-1.88</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>31.09</td>
<td>28.75</td>
<td>27.74</td>
<td>35.46</td>
<td>36.31+0.85</td>
<td></td>
</tr>
<tr>
<td>SUN397</td>
<td>Base</td>
<td>69.36</td>
<td>80.65</td>
<td>79.74</td>
<td>78.67</td>
<td>82.13</td>
<td>+2.45</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>75.35</td>
<td>65.89</td>
<td>76.86</td>
<td>76.93</td>
<td>77.20</td>
<td>+0.29</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>72.23</td>
<td>72.51</td>
<td>78.27</td>
<td>77.79</td>
<td>79.59+1.83</td>
<td></td>
</tr>
<tr>
<td>DTD</td>
<td>Base</td>
<td>53.24</td>
<td>79.44</td>
<td>77.01</td>
<td>80.67</td>
<td>81.29</td>
<td>+1.88</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>59.90</td>
<td>41.18</td>
<td>56.00</td>
<td>56.48</td>
<td>60.63</td>
<td>+4.53</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>56.37</td>
<td>54.24</td>
<td>64.85</td>
<td>66.44</td>
<td>69.46+3.02</td>
<td></td>
</tr>
<tr>
<td>EuroSAT</td>
<td>Base</td>
<td>56.48</td>
<td>92.19</td>
<td>87.49</td>
<td>83.90</td>
<td>93.40</td>
<td>+3.45</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>64.05</td>
<td>54.74</td>
<td>60.04</td>
<td>66.0</td>
<td>71.24</td>
<td>+5.31</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>60.03</td>
<td>68.9</td>
<td>71.21</td>
<td>73.88</td>
<td>80.83+6.95</td>
<td></td>
</tr>
<tr>
<td>UCF101</td>
<td>Base</td>
<td>70.53</td>
<td>84.69</td>
<td>82.35</td>
<td>85.23</td>
<td>86.97</td>
<td>+1.73</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>77.50</td>
<td>56.05</td>
<td>73.45</td>
<td>71.97</td>
<td>77.48</td>
<td>+4.39</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>73.85</td>
<td>67.46</td>
<td>77.64</td>
<td>78.04</td>
<td>81.95+3.90</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>Base</td>
<td>69.34</td>
<td>82.69</td>
<td>80.47</td>
<td>81.56</td>
<td>83.62</td>
<td>+2.81</td>
</tr>
<tr>
<td></td>
<td>New</td>
<td>74.22</td>
<td>63.22</td>
<td>71.69</td>
<td>72.30</td>
<td>74.56</td>
<td>+2.99</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>71.70</td>
<td>71.66</td>
<td>75.83</td>
<td>76.65</td>
<td>78.83+2.18</td>
<td></td>
</tr>
</tbody>
</table>

Domain Generalisation: As in base-to-new generalisation, we report the results obtained with two mappings as detailed in Section 4.6. As shown in Tab. 9, LFA outperforms CoOp [86] on the source domain, while also outperforming CoCoOp [85] on the target domains. These results further demonstrate the flexibility of LFA, which can be used to adapt to different domains and settings, and even under a test-time distribution shift, either on the language or the image side.

Standard Action Recognition: Tab. 11 shows the obtained action recognition results when using the full-training set for video-text alignment. LFA slightly outperforms the video soft-prompting method of [40] on average, and by a notable margin on HMDB51. However, and different from [40] that trains two transformer layers on top of the frozen per-frame CLIP features to model the temporal information in addition to the soft-prompt, LFA matches performances of [40] without any temporal modeling.

5.2. Ablation studies

In this section, we (1) ablate the impact of the proposed closed-form relaxed initialisation, (2) compare the new re-ranking loss with a series of baselines, and (3) analyse the effect of the number of image crops and of template-based text prompting for class prototyping. Moreover, to showcase the generalisability of our approach, (4) we explore our method’s behaviour on disjoint models, where the vision and text encoder are trained from separate sources, (5) the effectiveness of LFA with V-L models other than CLIP, (6) the results sensitivity to the choice of \( \beta \). Finally, (7) we report results for the unsupervised variant of our method. See supplementary material for some additional ablations.

Effect of closed-form initialisation and refinement: The proposed \( \beta \)-Procrustes regularisation significantly reduces overfitting (Tab. 2) and provides a notably better starting point for the refinement process (Tab. 4) across all datasets.

ARerank vs other losses: In Tab. 3 we compare the proposed ARerank loss with a series of baselines, out of which the CSLS loss is the closest conceptually to ours. As the results show, we outperform CSLS across all datasets by up to 4%. Moreover, we improve between 1.6% (for Imagenet) and 5% (for FGVC) on top of the standard contrastive loss.

Effect of the number of image crops and prototype templating: Our LFA is largely invariant to the exact number of crops per image used for training, showing strong results with a single crop (Tab. 7). Similar to [11, 85], we found template prompting for constructing the class prototypes to be beneficial (see Tab. 7 and 5).
Moreover, it doesn't require access to the weights, only to the produced features. To showcase this, we introduce a new evaluation setting in which the visual and text features are produced by disjoint models, that never interacted during training. Either one or both modalities can be sourced from behind-the-wall (i.e., blackbox) models. For this experiment, we consider 3 RN50 pre-trained visual backbones: BYOL, BarlowTwins and MoCo v3. As we do not require access to the model, we use the OpenAI embeddings API to get the text embeddings from the cpt-text encoder [55] and generate the class prototypes. After an initial random projection to project the image features into the 1536-d space and match the dimensionality of text features, we proceed with the supervised LFA procedure as with CLIP experiments. In a few-shot setting, alongside our method, we consider 2 baselines that also operate the frozen visual features: kNN and linear eval (see supplementary material for the hyper-parameters used). As the results from Tab. 13 show, our method (1) reaches performance comparable with that of aligned models (i.e., CLIP) and (2) outperforms both kNN and linear eval by a large margin.

Unsupervised LFA: As presented in Section 4.5, LFA can be adapted for label-free training. Tab. 5 shows that U-LFA improves by up to 7% on top of zero-shot CLIP.

6. Conclusions

In this work we proposed LFA, the first black box few-shot adaptation method for V-L models, that uses pre-computed image and text features without accessing the model's weights. Other advantages of LFA include fast training, applicability to both supervised and unsupervised setting, and even application for aligning image and text features computed from uni-modal models. Thanks to the use of precomputed features, we hope that LFA will enable few-shot adaptation of very large V-L foundation models that would otherwise be impossible to adapt or even deploy.
References


[38] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zaraana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In International Conference on Machine Learning, pages 4904–4916. PMLR, 2021.


[55] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry T worik, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. Text and

[56] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 6


