# LP++: A Surprisingly Strong Linear Probe for Few-Shot CLIP

Supplementary Material

## 5. Proof of Prop. 1

Let us start with $H_{\mathbf{w}}$, the Hessian matrix of objective function (3) with respect to block $\mathbf{w} \in \mathbf{R}^{KD}$, with block $\boldsymbol{\alpha}$ being fixed. Let us introduce vector $\boldsymbol{F}_{ik} \in \mathbb{R}^{KD}$, which takes the following form:

$$\boldsymbol{F}_{ik} = [0, \ldots, 0, \boldsymbol{f}_i^t, 0, \ldots, 0]^t \tag{17}$$

where embedding vector $\boldsymbol{f}_i \in \mathbb{R}^D$ corresponds to the elements of $\boldsymbol{F}_{ik}$ going from the $((k-1)D+1)^{\text{th}}$ to the $(kD)^{\text{th}}$ position, with the rest of elements of $\boldsymbol{F}_{ik}$ all being equal to zero. Block-coordinatewise $KD \times KD$ Hessian matrix $H_{\mathbf{w}}$ reads:

$$\begin{aligned} H_{\mathbf{w}} &= \frac{1}{N}\sum_{i=1}^{N} \underbrace{\sum_{k=1}^{K} p_{ik}\boldsymbol{F}_{ik}\boldsymbol{F}_{ik}^t}_{\boldsymbol{D}_i \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^t)} \\ &- \frac{1}{N}\sum_{i=1}^{N} \underbrace{\sum_{k=1}^{K}\sum_{k'=1}^{K} p_{ik}p_{ik'}\boldsymbol{F}_{ik}\boldsymbol{F}_{ik'}^t}_{(\mathbf{p}_i\mathbf{p}_i^t)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^t)} \\ &= \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t) \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^t) \end{aligned} \tag{18}$$

where $\otimes$ denotes the Kronecker matrix product, $\boldsymbol{D}_i$ is the diagonal $K \times K$ matrix whose elements are given by $p_{ik}$:

$$\boldsymbol{D}_i = \text{Diag}((p_{ik})_{1 \leq k \leq K})$$

and $\mathbf{p}_i \in \mathbb{R}^K$ is the vector whose elements are $p_{ik}$:

$$\mathbf{p}_i = (p_{ik})_{1 \leq k \leq K}$$

Now, let us show the following intermediate result on diagonally dominant, symmetric matrices $(\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t)$, which we could establish using the the well-known Gershgorin circle theorem.

**Lemma 5.1.** *Let $\lambda$ denotes an eigen value of matrix $(\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t)$, then $\lambda \in [0, \frac{1}{2}]$. Therefore, $(\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t)$ is positive semidefinite and its maximum eigen value verifies:*

$$\lambda_{max}(\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t) \leq \frac{1}{2}$$

*Proof.* For $j \in 1, \ldots, K$, let $R_j$ denotes the sum of the absolute values of the non-diagonal entries in the $j^{\text{th}}$ row of a $K \times K$ matrix. For $\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t$, observe the following, due to probability simplex constraint $\sum_{k=1}^{K} p_{ik} = 1$:

$$R_j = p_{ij}\sum_{k \neq j} p_{ik} = p_{ij}(1 - p_{ij}) = p_{ij} - p_{ij}^2$$

Also, notice that $R_j$ is equal to the diagonal element of the $j^{\text{th}}$ row, which means matrix $\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t$ is diagonally dominant. The Gershgorin circle theorem states that, for any eigen value $\lambda$ of a matrix, there exists at least a row $j$, so that $\lambda$ is within the disc centered at the diagonal element of the row and whose radius is $R_j$. Using this fact for matrix $\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t$ means that, for any eigen $\lambda$ of the matrix, there exists at least one raw $j$ verifying:

$$|\lambda - (p_{ij} - p_{ij}^2)| \leq R_j$$

This yields:

$$0 \leq \lambda \leq 2(p_{ij} - p_{ij}^2) \leq \frac{1}{2}$$

The last inequality above is due to $0 \leq p_{ij} - p_{ij}^2 \leq \frac{1}{4}$ for $p_{ij} \in [0, 1]$. Since all the eigen values of matrix $\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t$ are in $[0, \frac{1}{2}]$, then the maximum eigen value is also within this interval. $\square$

From Lemma 5.1, it it straightforward to see that matrix $\boldsymbol{D}_i - \mathbf{p}_i\mathbf{p}_i^t - \frac{1}{2}\boldsymbol{I}_K$ is negative semidefinite and, therefore, its Kronecker multiplication by $\boldsymbol{f}_i\boldsymbol{f}_i^t$ is also negative semidefinite (as $\boldsymbol{f}_i\boldsymbol{f}_i^t$ is positive semidefinite). Hence, the following matrix is negative semi-definite, as it is the sum of negative semidefinite matrices:

$$H_{\mathbf{w}} - \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\boldsymbol{I}_K \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^t) \tag{19}$$

Therefore:

$$\boldsymbol{x}^t H_{\mathbf{w}}\boldsymbol{x} \leq \boldsymbol{x}^t \left(\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\boldsymbol{I}_K \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^t)\right)\boldsymbol{x} \quad \forall \boldsymbol{x} \tag{20}$$

This yields:

$$\max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^t H_{\mathbf{w}}\boldsymbol{x} \leq \max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^t \left(\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\boldsymbol{I}_K \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^t)\right)\boldsymbol{x} \tag{21}$$

Now, recall the following variational characterization of the maximum eigenvalue of a matrix, following the min-max theorem, also referred to as the variational principle:

$$\lambda_{max}(H) = \max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^t H\boldsymbol{x}$$

Using this variational characterization and Eq. (21), we obtain:

$$\lambda_{max}(H_{\mathbf{w}}) \leq \frac{1}{2N} \lambda_{max} \left( \sum_{i=1}^{N} \boldsymbol{I}_K \otimes (\boldsymbol{f}_i \boldsymbol{f}_i^t) \right) \tag{22}$$

Observe that $KD \times KD$ matrix $\sum_{i=1}^{N} \boldsymbol{I}_K \otimes (\boldsymbol{f}_i \boldsymbol{f}_i^t)$ is block-diagonal with each of its $D \times D$ diagonal blocks corresponding to matrix $\sum_{i=1}^{N} \boldsymbol{f}_i \boldsymbol{f}_i^t$. Therefore, we have:

$$\lambda_{max} \left( \sum_{i=1}^{N} \boldsymbol{I}_K \otimes (\boldsymbol{f}_i \boldsymbol{f}_i^t) \right) = \lambda_{max} \left( \sum_{i=1}^{N} \boldsymbol{f}_i \boldsymbol{f}_i^t \right)$$

This means that the expression in Eq. (10) in the paper is an upper bound on the maximum eigen value of $H_{\mathbf{w}}$, for $\tau_1 \geq 2$. Hence, it is a valid block Lipschitz constant for the set of variables in $\mathbf{w}$.

It is also possible to have a tighter, but *approximate* Lipschitz constant by omitting the off-diagonal elements in matrices $\boldsymbol{D}_i - \mathbf{p}_i \mathbf{p}_i^t$ in the Hessian-matrix expression in Eq. (18). The approximation is motivated by the fact that these matrices are diagonally dominant. This gives the following approximation of the Hessian:

$$\tilde{H}_{\mathbf{w}} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\boldsymbol{D}}_i \otimes (\boldsymbol{f}_i \boldsymbol{f}_i^t) \tag{23}$$

where $\tilde{\boldsymbol{D}}_i$ is the diagonal $K \times K$ matrix whose elements are given by $p_{ik} - p_{ik}^2, k = 1, \dots, K$. Since $p_{ik} - p_{ik}^2 \leq \frac{1}{4}$, we have: $\lambda_{max}(\tilde{\boldsymbol{D}}_i) \leq \frac{1}{4}$. Therefore, following the same arguments as before, we have:

$$\lambda_{max}(\tilde{H}_{\mathbf{w}}) \leq \frac{1}{4N} \lambda_{max} \left( \sum_{i=1}^{N} (\boldsymbol{f}_i \boldsymbol{f}_i^t) \right) \tag{24}$$

Hence, $\tau_1 \geq 1$ in Eq. (10) in the paper provides an approximate Lipschitz constant for the block of variables in $\mathbf{w}$.

Notice that we still need an eigenvalue decomposition of $\sum_{i=1}^{N} \boldsymbol{f}_i \boldsymbol{f}_i^t$. However, for feature embeddings of size 1024, i.e., a $1024 \times 1024$ matrix, this computation, which has to be performed only once, is manageable; it took 0.3 seconds on a single NVIDIA RTX A6000 GPU.

Now, let us look at the Hessian matrix of objective function (3) w.r.t block $\boldsymbol{\alpha} \in \mathbf{R}^K$, with the variables in $\mathbf{w}$ being fixed. Let vector $\boldsymbol{T}_{ik} \in \mathbb{R}^K$ takes the form:

$$\boldsymbol{T}_{ik} = [0, \dots, 0, \boldsymbol{f}_i^t \boldsymbol{t}_k, 0, \dots, 0]^t$$

where the kth element is equal to $\boldsymbol{f}_i^t \boldsymbol{t}_k$ and the rest of elements are equal to zero. This block-coordinatewise $K \times K$ Hessian reads:

$$\begin{aligned} H_{\boldsymbol{\alpha}} &= \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} p_{ik} \boldsymbol{T}_{ik} \boldsymbol{T}_{ik}^t \\ &- \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \sum_{k'=1}^{K} p_{ik} p_{ik'} \boldsymbol{T}_{ik} \boldsymbol{T}_{ik'}^t \\ &= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{D}_i - \mathbf{p}_i \mathbf{p}_i^t) \odot \boldsymbol{T}_i \end{aligned} \tag{25}$$

where $\odot$ denotes the Hadamard product and $\boldsymbol{T}_i$ is the $K \times K$ matrix whose element at $(k, k') \in [1, \dots, K]^2$ is given by: $\boldsymbol{f}_i^t \boldsymbol{t}_k \boldsymbol{f}_i^t \boldsymbol{t}_{k'}$. As before, omitting the off-diagonal elements in Eq. (25) yields an approximate Hessian, $\tilde{H}_{\boldsymbol{\alpha}}$, which corresponds to the diagonal $K \times K$ matrix whose diagonal elements are given by:

$$\frac{1}{N} \sum_{i=1}^{N} (p_{ik} - p_{ik}^2)(\boldsymbol{f}_i^t \boldsymbol{t}_k)^2, k = 1, \dots, K$$

Thus, using $p_{ik} - p_{ik}^2 \leq \frac{1}{4}$, we obtain for $\tau_2 \geq 1$:

$$\lambda_{max}(\tilde{H}_{\boldsymbol{\alpha}}) \leq \max_k \frac{\tau_2}{4N} \sum_{i=1}^{N} (\boldsymbol{f}_i^t \boldsymbol{t}_k)^2 \tag{26}$$

This gives the approximate Lipschitz constant in Eq. (11).

Finally, it is well-known that, for two blocks of variables with Lipschitz constants $\gamma_{\mathbf{w}}$ and $\gamma_{\boldsymbol{\alpha}}$, the following is a Global Lipschitz constant (See [1]):

$$\gamma = 2 \max(\gamma_{\mathbf{w}}, \gamma_{\boldsymbol{\alpha}}) \tag{27}$$

Therefore, for $\tau \geq 2$, the expression in Eq. (12) provides an approximate global Lipschitz constant.

## 6. Proof of Proposition 2

Consider the following expressions of $g_1$ and $g_2$, for some $\lambda > 0$:

$$\begin{aligned} g_1 &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \boldsymbol{f}_i^t (\boldsymbol{w}_k + \alpha_k \boldsymbol{t}_k) + \frac{\lambda}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|^2 \\ g_2 &= \frac{1}{N} \sum_{i=1}^{N} \ln \left( \sum_{j=1}^{K} \exp \boldsymbol{f}_i^t (\boldsymbol{w}_j + \alpha_j \boldsymbol{t}_j) \right) - \frac{\lambda}{2} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|^2 \end{aligned} \tag{28}$$

It is easy to verify that $L = g_1 + g_2$. Let us now write the gradients of $g_1$ and $g_2$ w.r.t $\boldsymbol{w}_k$, $k \in [1, \dots, K]$:

$$\begin{aligned} \frac{\partial g_1}{\partial \boldsymbol{w}_k} &= -\frac{1}{N} \sum_{i=1}^{N} y_{ik} \boldsymbol{f}_i + \lambda \boldsymbol{w}_k \\ \frac{\partial g_2}{\partial \boldsymbol{w}_k} &= \frac{1}{N} \sum_{i=1}^{N} p_{ik} \boldsymbol{f}_i - \lambda \boldsymbol{w}_k \end{aligned} \tag{29}$$

Independently of the value of $\lambda > 0$, $g_1$ is convex w.r.t each $\boldsymbol{w}_k$, $k \in [1, \dots, K]$, as it is the sum of linear and convex functions. By setting the gradient of $g_1$ w.r.t $\boldsymbol{w}_k$ to 0, we obtain the minimum given in Eq. (13). As for $g_2$, we can ensure it is convex w.r.t each $\mathbf{w}_k$ by setting $\lambda$ as a function of positive semi-definite (PSD) matrices $\boldsymbol{A}_k$, as given in Prop. 2. This ensures that the Hessian of $g_2$ is PSD, as $\boldsymbol{A}_k$ is the Hessian of the first term in $g_2$. Setting the gradient of $g_2$ w.r.t $\boldsymbol{w}_k$ to 0 yields the minimum in Eq. (14).

## 7. More details on the initialization

The technical observations made in the paper in Eq. (13) and Eq. (15) suggest expressions for initializing variables $\alpha_k$ and $\boldsymbol{w}_k$:

$$\begin{aligned}\alpha_k^0 &= \frac{1}{\beta N}\sum_{i=1}^{N} y_{ik}\boldsymbol{f}_i^t\boldsymbol{t}_k = \frac{1}{\beta N}\tilde{\alpha}_k \\ \boldsymbol{w}_k^0 &= \frac{1}{\lambda N}\sum_{i=1}^{N} y_{ik}\boldsymbol{f}_i = \frac{1}{\lambda N}\tilde{\boldsymbol{w}}_k\end{aligned} \tag{30}$$

There are two non-negative multiplicative factors, $\lambda$ and $\beta$, which appear in these expressions. Here, we provide more details on how to set $\lambda$ and $\beta$ systematically (without hyper-parameter search).

Consider the logit scores for each sample $j$, according to our image-text linear probe model:

$$l_{j,k} = \frac{1}{\lambda N}\boldsymbol{f}_j^t\tilde{\boldsymbol{w}}_k + \frac{1}{\beta N}\tilde{\alpha}_k\boldsymbol{f}_j^t\boldsymbol{t}_k \propto \boldsymbol{f}_j^t\tilde{\boldsymbol{w}}_k + \frac{\lambda}{\beta}\tilde{\alpha}_k\boldsymbol{f}_j^t\boldsymbol{t}_k \tag{31}$$

Notice that multiplying $\lambda$ and $\beta$ by the same non-negative multiplicative factor does not change the order of the class scores, i.e., we only need to set a value for the ratio $\frac{\lambda}{\beta}$, which balances in Eq. (31) the contribution of the text-embedding term vs. the contribution of the visual-prototype term. We set $\frac{\lambda}{\beta}$ systematically (without additional hyper-parameters) as a monotonically decreasing function of the number of support samples ($S = \frac{N}{K}$):

$$\frac{\lambda}{\beta} = \frac{250}{S} \tag{32}$$

The choice in Eq. (32) is motivated by two intuitive reasons. First, the text-embedding scores are at least one-order of magnitude smaller than the visual-embedding scores. Second, a monotonically decreasing function of $S$ makes sense: the smaller the number of visual support samples, the bigger the weight given to the text-knowledge. Finally, by setting $\frac{\lambda}{\beta}$ according to Eq. (32) and $\lambda = \frac{1}{N}$, we initialize the variables as follows:

$$\begin{aligned}\alpha_k^0 &= \frac{250}{S}\sum_{i=1}^{N} y_{ik}\boldsymbol{f}_i^t\boldsymbol{t}_k \\ \boldsymbol{w}_k^0 &= \sum_{i=1}^{N} y_{ik}\boldsymbol{f}_i\end{aligned} \tag{33}$$

## 8. More details on the training-free version of our method

Here we gave more details on the training-free version of our method LP++. To be specific, for a given test sample, we can compute the class scores (logits) as follows:

$$l_{\text{test},k} = \boldsymbol{f}_{\text{test}}^t(\boldsymbol{w}_k^0 + \alpha_k^0\boldsymbol{t}_k) \quad \forall k$$

where $\boldsymbol{f}_{\text{test}}$ is the vision embedding of the test sample and the initial, hyper-parameter free variables $\alpha_k^0$ and $\boldsymbol{w}_k^0$ are set according to Eq. (33). Then, the class of test sample $\boldsymbol{f}_{\text{test}}^t$ is predicted as:

$$\hat{k} = \arg\max_k l_{\text{test},k}$$

## 9. Detailed explanation of the flaw in the experimental evaluation of TIP-adapter [28]

By examining the official GitHub repository[6], we found that the original implementation of TIP-adapter [28] uses large test-data performance feedback for choosing the hyper-parameters and best model. Therefore, we re-evaluated TIP-adapter using the same small validation sets as those used for all the competing methods, for fairness. Indeed, the paper [28] does not provide details as to the hyper-parameter selection and use of validation data. According to the GitHub repository, the authors mentioned that key hyper-parameters ($\alpha$ and $\beta$ in [28]), which control the trade-off between the text and vision supervision, are both set to 1 as tuning baseline (see Issue #13 in the GitHub). They also mentioned that $\alpha$ has to be increased when the domain gap between the CLIP pre-trained model and the downstream task is large. Starting from some initial values of the hyper-parameters, with $\beta$ set to 1 and $\alpha \in [1, 10]$ but predetermined specifically for each dataset, the authors deployed an additional search function after the adaptation procedure. **This function finds the best combination of $\alpha$ and $\beta$ on either the test set for ImageNet or the entire validation set for the other 10 datasets, which invalidates the few-shot assumption**. Furthermore, the grid search intervals for $\alpha$ and $\beta$ vary significantly among the different datasets (and somtimes do not even overlap), e.g., $\alpha \in [1.17, 7]$ for ImageNet whereas $\alpha \in [10, 50]$ for Flowers102.

[6]https://github.com/gaopengcuhk/Tip-Adapter/issues/13

| Number of shots ($S$) | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| BMM (iter$_{\mathbf{w}}$=10, iter$_{\boldsymbol{\alpha}}$=1) | **63.43** | **66.20** | **69.16** | **72.04** | **74.42** |
| BMM (iter$_{\mathbf{w}}$=10, iter$_{\boldsymbol{\alpha}}$=10) | 62.85 | 66.01 | 69.12 | 71.98 | 74.33 |
| BCGD | 62.92 | 66.04 | 69.03 | 71.94 | 74.24 |
| GD (our Lipschitz cst) | 63.04 | 66.14 | 69.15 | 71.99 | 74.28 |

Table 5. Comparison of different block-cycling strategies using our initialization. BCGD corresponds to BMM with both iter$_{\mathbf{w}}$ and iter$_{\boldsymbol{\alpha}}$ equal to 1. GD (our Lipschitz cst) corresponds to the single-block version of LP++. Results are averaged over 11 datasets.

Through our experiments, we found that the initial value of $\alpha$ plays a substantial role in the final performance of the Tip-Adapter-F model. Hence, for a fair comparison, we re-evaluated Tip-Adapter-F* (Table 1) by (i) finding the initial value of $\alpha_{\text{init}} \in [1, 10]$ on the small validation set deployed for all methods, and (ii) setting $\beta_{\text{init}} = 1$. Then, we maintain the implementation of the original publication, and perform the final grid search after adaptation. Instead of employing a data-set specific range for these two hyper-parameters, we use $[\alpha_{\text{init}}, 50]$ for $\alpha$ and $[\beta_{\text{init}}, 50]$ for $\beta$ during grid search. Although these grid-search steps add significantly to the overall computational overhead of TIP-adapter, they are needed to achieve performances close to those reported in the paper [28] (refer to Fig. 1 for a comparison between Tip-Adapter-F and Tip-Adapter-F*).

As for the training-free version of Tip-adapter [28], which we evaluated in Table 3, we fixed these hyper-parameters, i.e., $\alpha = \beta = 1$, for a fair comparison with our training-free LP++. Indeed, one could argue that searching for these hyper-parameters on large validation sets would invalidate the claim that the method is ''training-free".

In addition to the mentioned flaws, the original implementation of [28] finds the best model using the performance on the test set. In our re-implementation, we addressed this issue by applying early stopping on the small validation set.

## 10. Ablation on the block-cycling strategies

Table 5 reports the performances for different block-cycling strategies in our Block Majorize-Minimize (BMM) procedure. For example, BMM (iter$_{\mathbf{w}}$=10, iter$_{\boldsymbol{\alpha}}$=1) corresponds to the LP++ version in Alg. 1, whose performance is reported in the paper. GD corresponds to the single-block MM version of LP++, and BCGD (Block Coordinate Gradient Descent) corresponds the block-wise version of LP++ with both iter$_{\mathbf{w}}$ and iter$_{\boldsymbol{\alpha}}$ equal to 1.

## 11. Method ranking

In Fig. 5, we show the ranking of different methods using Autorank[12] (on the 11 datasets); lower value indicates better performance.

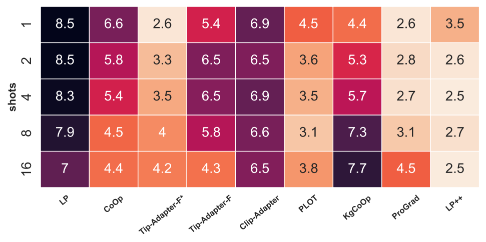

Figure 5. The ranking of different methods using Autorank [12]. The results are averaged over 11 datasets.

## 12. Implementation of the L-BFGS optimizer

When assessing the standard LP as in Table 6, we follow the instructions provided in [29], where the L-BFGS optimizer is implemented by NumPy, and used to optimize the standard CE loss jointly with L2 regularizers. As for the ablation study in Table 2, all the optimizers are implemented by Pytorch, including L-BFGS[7]. In this case, we did not use regularizers, for fair comparisons.

## 13. Details of prompt design

We follow the same strategy as in [28] for the text prompts. To be specific, we use the average of 7 templates per class for ImageNet, while for the other datasets, we use a single template. The latter might be different according to the type of the images those datasets include. For example, for Caltech101, we simply use "a photo of a [class$_k$]" for each class, while for Food101, we use "a photo of [class$_k$], a type of food" for each class.

## 14. Proof Lemma 2.1.

Assume a function $L(\mathbf{v})$ is $\gamma$-smooth, i.e., its gradient is Lipschitz (with a Lipschitz constant $\gamma > 0$): $\nabla^2 L(\mathbf{v}) \preceq \gamma \mathbf{I}$, with $\mathbf{I}$ the identity matrix. From this condition on the Hessian of $L$, it is easy to see that the following function is convex: $G(\mathbf{v}) = \frac{\gamma}{2}\mathbf{v}^t\mathbf{v} - L(\mathbf{v})$. Using the first-order condition for convexity of $G$ gives:

$$G(\mathbf{v}) \geq G(\mathbf{v}^j) + \nabla G(\mathbf{v}^j)^t(\mathbf{v} - \mathbf{v}^j) \tag{34}$$

Then using the expression of $G$ and its gradient in Eq. (34) yield the following quadratic upper bound on $L$, which corresponds to majorizing function $M(\mathbf{v}, \mathbf{v}^j)$ in the paper:

$$L(\mathbf{v}) \leq L(\mathbf{v}^j) + \nabla L(\mathbf{v}^j)^t(\mathbf{v} - \mathbf{v}^j) + \frac{\gamma}{2}\|\mathbf{v} - \mathbf{v}^j\|^2 \tag{35}$$

By setting the derivative of this convex upper bound to zero, we get the following solution, which corresponds to a specific gradient-descent step:

$$\mathbf{v}^{j+1} = \mathbf{v}^j - \frac{1}{\gamma}\nabla L(\mathbf{v}^j) = \arg\min_{\mathbf{v}} M(\mathbf{v}, \mathbf{v}^j) \tag{36}$$

[7]https://github.com/hjmshi/PyTorch-LBFGS

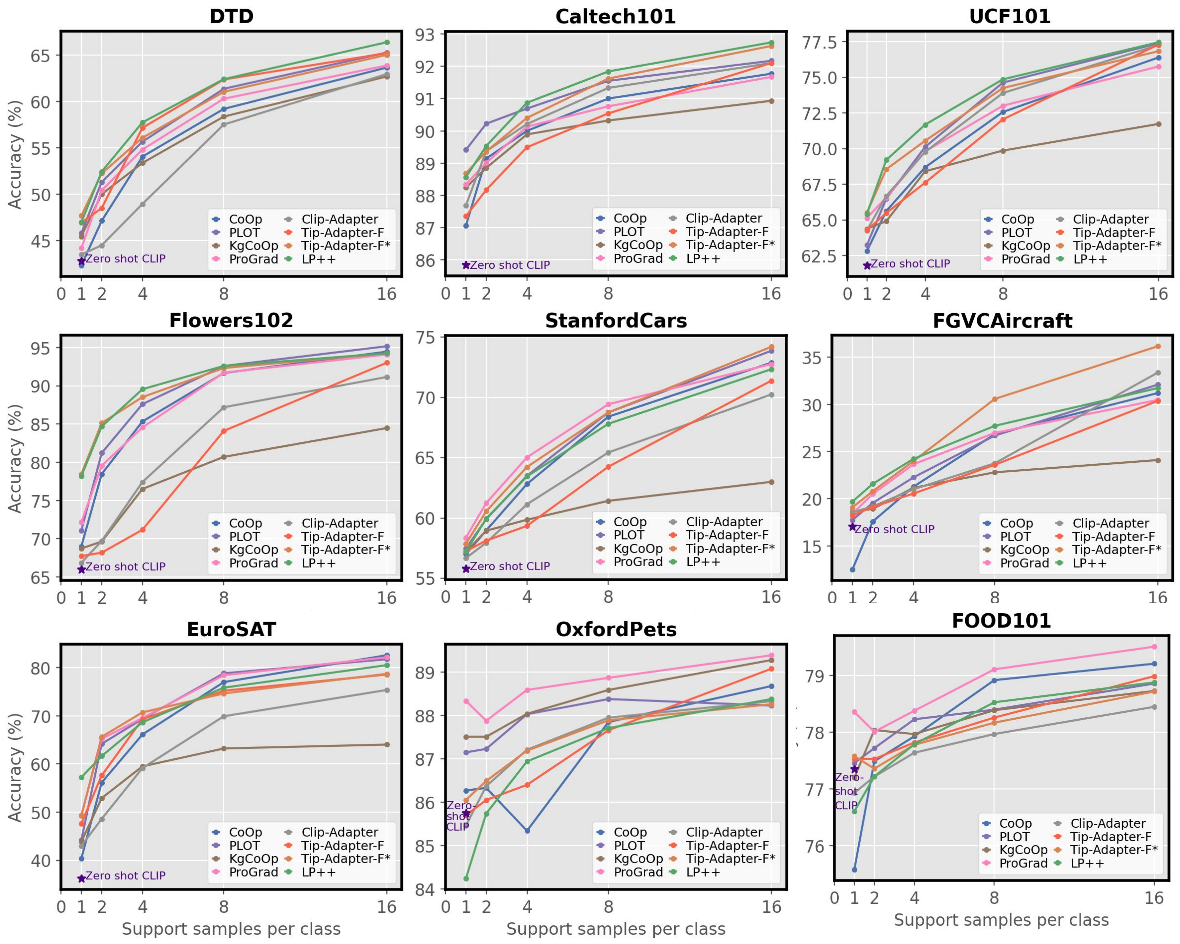

Figure 6. Comparison of LP++ with different adaptation methods on 9 benchmarks, which are averaged over 10 tasks.

Applying Eq. (35) to $\mathbf{v} = \mathbf{v}^{j+1}$, and plugging the gradient step in Eq. (36) into Eq. (35), one gets the following decrease of objective $L$, by at least $\frac{1}{2\gamma}\|\nabla L(\mathbf{v}^j)\|^2$:

$$L(\mathbf{v}^{j+1}) \leq L(\mathbf{v}^j) - \frac{1}{2\gamma}\|\nabla L(\mathbf{v}^j)\|^2 \tag{37}$$

## 15. Proof Theorem 2.2.

The standard result in Theorem 2.2 provides, for a convex, $\gamma$-smooth function, the convergence rate of gradient steps with step sizes $\frac{1}{\gamma}$. As $L$ is convex, we have:

$$L(\mathbf{v}^j) \leq L(\mathbf{v}^*) + \nabla L(\mathbf{v}^j)^t(\mathbf{v}^j - \mathbf{v}^*) \tag{38}$$

Plugging Eq. (38) in Eq. (37) yields:

$$\begin{aligned} & L(\mathbf{v}^{j+1}) - L(\mathbf{v}^*) && (39) \\ & \leq \nabla L(\mathbf{v}^j)^t(\mathbf{v}^j - \mathbf{v}^*) - \frac{1}{2\gamma}\|\nabla L(\mathbf{v}^j)\|^2 \\ & = \frac{\gamma}{2}(\|\mathbf{v}^j - \mathbf{v}^*\|^2 - \|\mathbf{v}^j - \frac{1}{\gamma}\nabla L(\mathbf{v}^j) - \mathbf{v}^*\|^2) \\ & = \frac{\gamma}{2}(\|\mathbf{v}^j - \mathbf{v}^*\|^2 - \|\mathbf{v}^{j+1} - \mathbf{v}^*\|^2) && (40) \end{aligned}$$

Summing both parts of the inequality in Eq. (39), we get:

$$\begin{aligned} & \sum_{j=0}^{J-1}\left(L(\mathbf{v}^{j+1}) - L(\mathbf{v}^*)\right) && (41) \\ & \leq \frac{\gamma}{2}\sum_{j=0}^{J-1}(\|\mathbf{v}^j - \mathbf{v}^*\|^2 - \|\mathbf{v}^{j+1} - \mathbf{v}^*\|^2) \\ & = \frac{\gamma}{2}(\|\mathbf{v}^0 - \mathbf{v}^*\|^2 - \|\mathbf{v}^J - \mathbf{v}^*\|^2) \\ & \leq \frac{\gamma}{2}(\|\mathbf{v}^0 - \mathbf{v}^*\|^2) && (42) \end{aligned}$$

Finally, using the fact that $L$ is decreasing at each iteration, along with inequality Eq. (41), we have:

$$\begin{aligned} L(\mathbf{v}^J) - L(\mathbf{v}^*) &\leq \frac{1}{J}\sum_{j=0}^{J-1}\big(L(\mathbf{v}^{j+1}) - L(\mathbf{v}^*)\big) \\ &\leq \frac{\gamma}{2J}(\|\mathbf{v}^0 - \mathbf{v}^*\|^2) \end{aligned} \tag{43}$$

## 16. Additional numerical results on 11 benchmarks

Tab. 6 shows the average accuracy and standard deviation over 10 tasks for LP++ and the baseline methods on 11 benchmarks. The per-dataset performance across all the methods and for different number of shots is depicted in Fig. 6, which complements Fig. 3 in the main paper. In these plots, one may observe that LP++ (*green curve*) yields the best overall performance in 6 out of 11 data sets. Furthermore, LP++ outperforms all adapter-based methods in nearly all the remaining datasets and scenarios (i.e., when varying the number of shots). Only in a few cases, e.g., FGVCAirCraft and OxfordPets, Tip-Adapter-F* yields a performance better than LP++. Indeed, the largest improvement gains are observed in FGVCAirCraft, where the accuracy obtained by all methods is considerably small. We believe that, in this particular scenario, the more intensive grid search carried out by Tip-Adapter-F* has a greater impact on its results. Considering prompt learning strategies, ProGrad is the top-performing approach, which obtains the best overall results in several datasets. Nevertheless, in addition to the limitations already highlighted for these methods (i.e., time complexity and non-suitability for black-box adaptation), the large standard deviations observed across runs (see also Fig. 1), make these approaches highly unreliable, specially in the low-labeled data regime. As discussed, this could be due to the fact that these approaches learn prompts, through the text encoder, which are "too specialized" for a given image support set.

| Dataset | **Number of shots** ($S$) | **1** | **2** | **4** | **8** | **16** |
|---|---|---|---|---|---|---|
| *ImageNet* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 60.35 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 61.19 ± 0.17 | 61.58 ± 0.40 | 62.22 ± 0.22 | 62.87 ± 0.21 | 63.70 ± 0.13 |
| | PLOT $_{\text{ICLR'23}}$[4] | 60.46 ± 0.34 | 60.73 ± 0.60 | 61.79 ± 0.39 | 62.48 ± 0.32 | 63.08 ± 0.26 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 60.90 ± 0.16 | 61.44 ± 0.15 | 62.00 ± 0.11 | 62.20 ± 0.15 | 62.43 ± 0.11 |
| | ProGrad $_{\text{ICCV'23}}$[30] | **61.58** ± 0.27 | **62.14** ± 0.13 | **62.59** ± 0.09 | 63.04 ± 0.11 | 63.54 ± 0.18 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 59.82 ± 0.11 | 59.94 ± 0.05 | 59.97 ± 0.04 | 59.98 ± 0.09 | 61.31 ± 0.39 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 60.59 ± 0.14 | 61.42 ± 0.05 | 62.12 ± 0.06 | 63.41 ± 0.07 | **65.06** ± 0.04 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | 60.98 ± 0.15 | 61.23 ± 0.11 | 61.72 ± 0.25 | 62.84 ± 0.10 | 64.03 ± 0.12 |
| | Standard LP $_{\text{ICML'21}}$[23] | 22.21 ± 0.31 | 31.96 ± 0.25 | 41.48 ± 0.25 | 49.49 ± 0.16 | 56.04 ± 0.13 |
| | LP++ | 61.18 ± 0.08 | 61.56 ± 0.14 | 62.55 ± 0.12 | **63.76** ± 0.07 | 64.73 ± 0.05 |
| *SUN397* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 58.85 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 61.79 ± 0.45 | 63.32 ± 0.47 | 65.79 ± 0.44 | 67.89 ± 0.38 | 70.15 ± 0.32 |
| | PLOT $_{\text{ICLR'23}}$[4] | 62.53 ± 0.30 | 63.87 ± 0.26 | 65.85 ± 0.48 | 67.83 ± 0.36 | 69.90 ± 0.31 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | **62.91** ± 0.59 | 64.38 ± 0.30 | 66.06 ± 0.37 | 66.66 ± 1.10 | 67.68 ± 0.78 |
| | ProGrad $_{\text{ICCV'23}}$[30] | 62.79 ± 0.50 | 64.12 ± 0.55 | 66.32 ± 0.59 | 68.33 ± 0.28 | 70.18 ± 0.27 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 60.78 ± 0.16 | 61.79 ± 0.23 | 63.84 ± 0.35 | 66.26 ± 0.14 | 67.66 ± 1.05 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 61.02 ± 0.36 | 62.15 ± 0.28 | 63.86 ± 0.19 | 67.25 ± 0.16 | 70.94 ± 0.13 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | 62.58 ± 0.22 | 63.79 ± 0.13 | 65.49 ± 0.35 | 67.43 ± 0.11 | 69.25 ± 0.16 |
| | Standard LP $_{\text{ICML'21}}$[23] | 32.56 ± 0.40 | 43.77 ± 0.41 | 54.49 ± 0.39 | 61.83 ± 0.30 | 67.03 ± 0.16 |
| | LP++ | 62.47 ± 0.27 | **64.65** ± 0.25 | **67.28** ± 0.27 | **69.34** ± 0.14 | **71.23** ± 0.07 |
| *DTD* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 42.69 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 42.31 ± 1.86 | 47.13 ± 1.93 | 54.06 ± 1.49 | 59.21 ± 0.91 | 63.67 ± 0.83 |
| | PLOT $_{\text{ICLR'23}}$[4] | 45.82 ± 1.72 | 51.32 ± 1.61 | 55.67 ± 1.14 | 61.38 ± 1.04 | 65.29 ± 1.05 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 45.46 ± 2.83 | 50.01 ± 2.71 | 53.37 ± 0.71 | 58.38 ± 1.34 | 62.71 ± 0.92 |
| | ProGrad $_{\text{ICCV'23}}$[30] | 44.19 ± 2.38 | 50.41 ± 1.74 | 54.82 ± 1.28 | 60.31 ± 0.99 | 63.89 ± 0.88 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 43.49 ± 0.68 | 44.49 ± 1.07 | 48.95 ± 0.85 | 57.52 ± 0.67 | 62.97 ± 0.60 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 46.92 ± 1.01 | 48.50 ± 1.08 | 57.16 ± 0.53 | 62.38 ± 0.47 | 65.23 ± 0.82 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | **47.68** ± 1.43 | 52.24 ± 0.74 | 56.09 ± 0.99 | 61.05 ± 0.71 | 65.04 ± 0.21 |
| | Standard LP $_{\text{ICML'21}}$[23] | 29.63 ± 1.53 | 41.19 ± 1.45 | 51.72 ± 0.57 | 58.78 ± 0.52 | 64.56 ± 0.69 |
| | LP++ | 46.97 ± 1.37 | **52.44** ± 0.99 | **57.75** ± 0.82 | **62.42** ± 0.53 | **66.40** ± 0.50 |
| *Caltech101* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 85.84 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 87.06 ± 1.24 | 89.14 ± 0.87 | 90.00 ± 0.63 | 91.00 ± 0.66 | 91.77 ± 0.29 |
| | PLOT $_{\text{ICLR'23}}$[4] | **89.41** ± 0.41 | **90.22** ± 0.25 | 90.69 ± 0.37 | 91.55 ± 0.38 | 92.17 ± 0.30 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 88.24 ± 0.49 | 88.85 ± 0.43 | 89.89 ± 0.31 | 90.32 ± 0.43 | 90.93 ± 0.26 |
| | ProGrad $_{\text{ICCV'23}}$[30] | 88.34 ± 1.64 | 89.01 ± 0.61 | 90.13 ± 0.45 | 90.76 ± 0.32 | 91.67 ± 0.39 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 87.69 ± 0.41 | 89.37 ± 0.29 | 90.21 ± 0.32 | 91.33 ± 0.15 | 92.10 ± 0.20 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 87.35 ± 0.64 | 88.17 ± 0.29 | 89.49 ± 0.25 | 90.54 ± 0.34 | 92.10 ± 0.25 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | 88.68 ± 0.44 | 89.36 ± 0.59 | 90.40 ± 0.26 | 91.62 ± 0.23 | 92.63 ± 0.21 |
| | Standard LP $_{\text{ICML'21}}$[23] | 68.88 ± 1.68 | 78.41 ± 0.54 | 84.91 ± 0.45 | 88.70 ± 0.40 | 91.14 ± 0.19 |
| | LP++ | 88.56 ± 0.43 | 89.53 ± 0.35 | **90.87** ± 0.19 | **91.84** ± 0.24 | **92.73** ± 0.17 |
| *UCF101* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 61.80 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 62.80 ± 1.26 | 65.62 ± 1.09 | 68.69 ± 0.76 | 72.57 ± 0.80 | 76.39 ± 0.54 |
| | PLOT $_{\text{ICLR'23}}$[4] | 63.22 ± 1.05 | 66.49 ± 0.92 | 70.12 ± 0.62 | 74.63 ± 0.79 | 77.39 ± 0.53 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 64.37 ± 1.66 | 64.91 ± 1.01 | 68.41 ± 0.38 | 69.86 ± 0.33 | 71.73 ± 0.78 |
| | ProGrad $_{\text{ICCV'23}}$[30] | 65.13 ± 0.87 | 66.57 ± 0.62 | 69.80 ± 0.62 | 73.01 ± 0.52 | 75.76 ± 0.47 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 64.25 ± 0.54 | 66.68 ± 0.31 | 69.77 ± 0.40 | 73.90 ± 0.50 | 77.26 ± 0.39 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 64.28 ± 0.54 | 65.48 ± 0.43 | 67.61 ± 0.28 | 72.05 ± 0.53 | 77.30 ± 0.21 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | **65.50** ± 0.59 | 68.55 ± 0.45 | 70.55 ± 0.58 | 74.25 ± 0.48 | 76.83 ± 0.24 |
| | Standard LP $_{\text{ICML'21}}$[23] | 40.80 ± 1.05 | 51.71 ± 0.79 | 61.64 ± 0.50 | 68.47 ± 0.44 | 73.38 ± 0.43 |
| | LP++ | 65.41 ± 0.37 | **69.20** ± 0.52 | **71.68** ± 0.41 | **74.86** ± 0.36 | **77.46** ± 0.39 |
| *Flowers102* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 65.98 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 69.00 ± 2.44 | 78.47 ± 1.88 | 85.34 ± 1.69 | 91.68 ± 0.82 | 94.47 ± 0.36 |
| | PLOT $_{\text{ICLR'23}}$[4] | 71.09 ± 1.44 | 81.22 ± 0.92 | 87.61 ± 0.79 | 92.60 ± 0.55 | **95.18** ± 0.40 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 68.73 ± 1.97 | 69.63 ± 1.25 | 76.51 ± 0.51 | 80.71 ± 0.63 | 84.48 ± 0.70 |
| | ProGrad $_{\text{ICCV'23}}$[30] | 72.16 ± 1.74 | 79.55 ± 0.88 | 84.56 ± 1.41 | 91.73 ± 0.35 | 94.10 ± 0.41 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 66.86 ± 0.73 | 69.71 ± 0.46 | 77.42 ± 0.60 | 87.20 ± 0.52 | 91.16 ± 0.23 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 67.73 ± 0.57 | 68.18 ± 0.84 | 71.17 ± 0.67 | 84.11 ± 0.49 | 93.02 ± 0.28 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | **78.46** ± 1.01 | **85.14** ± 0.81 | 88.53 ± 0.54 | 92.33 ± 0.32 | 94.26 ± 0.38 |
| | Standard LP $_{\text{ICML'21}}$[23] | 56.98 ± 1.56 | 73.40 ± 0.87 | 84.38 ± 0.53 | 91.81 ± 0.34 | 95.05 ± 0.29 |
| | LP++ | 78.21 ± 1.01 | 84.69 ± 0.70 | **89.56** ± 0.45 | **92.61** ± 0.32 | 94.26 ± 0.24 |
| *StanfordCars* | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 55.78 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | 57.00 ± 0.93 | 58.96 ± 0.78 | 62.81 ± 0.71 | 68.40 ± 0.61 | 72.87 ± 0.50 |
| | PLOT $_{\text{ICLR'23}}$[4] | 57.47 ± 0.58 | 59.89 ± 0.60 | 63.49 ± 0.80 | 68.75 ± 0.46 | 73.86 ± 0.39 |
| | KgCoOp $_{\text{CVPR'23}}$[27] | 57.19 ± 0.65 | 58.94 ± 0.33 | 59.85 ± 0.51 | 61.42 ± 0.40 | 62.99 ± 1.33 |
| | ProGrad $_{\text{ICCV'23}}$[30] | **58.63** ± 0.39 | **61.23** ± 0.65 | **65.02** ± 0.78 | **69.43** ± 0.40 | 72.76 ± 0.45 |
| | CLIP-Adapter $_{\text{IJCV'23}}$[9] | 56.67 ± 0.22 | 57.94 ± 0.27 | 61.13 ± 0.30 | 65.43 ± 0.10 | 70.24 ± 0.79 |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | 57.24 ± 0.23 | 58.12 ± 0.50 | 59.34 ± 0.20 | 64.25 ± 0.19 | 71.38 ± 0.23 |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | 57.85 ± 0.33 | 60.55 ± 0.34 | 64.22 ± 0.52 | 68.75 ± 0.31 | **74.19** ± 0.30 |
| | Standard LP $_{\text{ICML'21}}$[23] | 22.94 ± 0.61 | 35.48 ± 0.51 | 47.49 ± 0.67 | 59.34 ± 0.30 | 69.11 ± 0.18 |
| | LP++ | 57.20 ± 0.65 | 59.95 ± 0.36 | 63.44 ± 0.34 | 67.81 ± 0.24 | 72.33 ± 0.18 |

Table 6. **Comparison to state-of-the-art methods.** Average classification accuracy (%) and standard deviation over 10 tasks for 11 benchmarks, Best values are highlighted in bold.

| Dataset | **Number of shots** ($S$) | **1** | **2** | **4** | **8** | **16** |
|---|---|---|---|---|---|---|
| | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 17.07 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | $12.50 \pm 6.16$ | $17.59 \pm 3.70$ | $21.27 \pm 2.54$ | $26.85 \pm 0.63$ | $31.20 \pm 0.40$ |
| | PLOT $_{\text{ICLR'23}}$[4] | $17.75 \pm 1.36$ | $19.55 \pm 0.99$ | $22.26 \pm 0.89$ | $26.70 \pm 0.70$ | $32.09 \pm 0.68$ |
| | KgCoOp $_{\text{CVPR'23}}$[27] | $18.61 \pm 0.76$ | $18.93 \pm 1.01$ | $21.16 \pm 0.82$ | $22.80 \pm 0.44$ | $24.10 \pm 0.59$ |
| | ProGrad $_{\text{ICCV'23}}$[30] | $18.41 \pm 0.98$ | $20.51 \pm 1.11$ | $23.65 \pm 0.50$ | $26.98 \pm 0.50$ | $30.47 \pm 0.76$ |
| *FGVCAircraft* | CLIP-Adapter $_{\text{IJCV'23}}$[9] | $18.56 \pm 0.20$ | $19.18 \pm 0.28$ | $21.00 \pm 0.21$ | $23.76 \pm 0.33$ | $33.37 \pm 0.23$ |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | $18.23 \pm 0.19$ | $19.12 \pm 0.20$ | $20.55 \pm 0.20$ | $23.60 \pm 0.29$ | $30.37 \pm 0.25$ |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | $19.08 \pm 0.15$ | $20.79 \pm 0.59$ | $23.99 \pm 0.57$ | $\mathbf{30.58} \pm 0.29$ | $\mathbf{36.16} \pm 0.34$ |
| | Standard LP $_{\text{ICML'21}}$[23] | $12.66 \pm 0.59$ | $16.92 \pm 0.56$ | $21.11 \pm 0.83$ | $26.53 \pm 0.38$ | $32.42 \pm 0.54$ |
| | LP++ | $\mathbf{19.69} \pm 0.39$ | $\mathbf{21.58} \pm 0.46$ | $\mathbf{24.22} \pm 0.60$ | $27.73 \pm 0.48$ | $31.73 \pm 0.44$ |
| | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 36.22 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | $40.36 \pm 7.19$ | $56.15 \pm 5.82$ | $66.13 \pm 3.62$ | $77.02 \pm 1.78$ | $\mathbf{82.59} \pm 1.00$ |
| | PLOT $_{\text{ICLR'23}}$[4] | $44.22 \pm 9.14$ | $64.19 \pm 6.24$ | $69.37 \pm 3.26$ | $\mathbf{78.84} \pm 1.33$ | $81.76 \pm 1.43$ |
| | KgCoOp $_{\text{CVPR'23}}$[27] | $43.86 \pm 9.17$ | $52.92 \pm 5.92$ | $59.51 \pm 3.46$ | $63.23 \pm 3.03$ | $64.04 \pm 1.40$ |
| | ProGrad $_{\text{ICCV'23}}$[30] | $49.37 \pm 5.03$ | $65.22 \pm 4.01$ | $69.57 \pm 2.88$ | $78.44 \pm 1.69$ | $82.17 \pm 0.98$ |
| *EuroSAT* | CLIP-Adapter $_{\text{IJCV'23}}$[9] | $43.00 \pm 2.27$ | $48.60 \pm 2.76$ | $59.15 \pm 2.26$ | $69.92 \pm 1.49$ | $75.38 \pm 0.78$ |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | $47.63 \pm 2.64$ | $57.62 \pm 1.86$ | $69.30 \pm 2.41$ | $75.22 \pm 1.32$ | $78.59 \pm 1.48$ |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | $49.27 \pm 2.88$ | $\mathbf{65.66} \pm 1.39$ | $\mathbf{70.72} \pm 2.73$ | $74.66 \pm 3.15$ | $78.73 \pm 0.81$ |
| | Standard LP $_{\text{ICML'21}}$[23] | $48.29 \pm 2.95$ | $56.81 \pm 2.93$ | $64.99 \pm 3.47$ | $74.56 \pm 0.98$ | $80.29 \pm 0.90$ |
| | LP++ | $\mathbf{57.23} \pm 1.63$ | $61.65 \pm 1.66$ | $68.67 \pm 2.21$ | $75.86 \pm 0.99$ | $80.53 \pm 1.05$ |
| | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 85.75 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | $86.27 \pm 1.17$ | $86.33 \pm 1.13$ | $85.34 \pm 1.69$ | $87.85 \pm 1.21$ | $88.68 \pm 0.71$ |
| | PLOT $_{\text{ICLR'23}}$[4] | $87.15 \pm 0.72$ | $87.23 \pm 1.21$ | $88.03 \pm 0.49$ | $88.38 \pm 0.64$ | $88.23 \pm 0.54$ |
| | KgCoOp $_{\text{CVPR'23}}$[27] | $87.51 \pm 0.68$ | $87.51 \pm 0.75$ | $88.04 \pm 0.46$ | $88.59 \pm 0.34$ | $89.28 \pm 0.21$ |
| | ProGrad $_{\text{ICCV'23}}$[30] | $\mathbf{88.34} \pm 0.65$ | $\mathbf{87.88} \pm 0.69$ | $\mathbf{88.59} \pm 0.58$ | $\mathbf{88.87} \pm 0.42$ | $\mathbf{89.39} \pm 0.47$ |
| *OxfordPets* | CLIP-Adapter $_{\text{IJCV'23}}$[9] | $85.46 \pm 0.48$ | $86.37 \pm 0.25$ | $87.21 \pm 0.51$ | $87.95 \pm 0.26$ | $88.33 \pm 0.33$ |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | $85.70 \pm 0.16$ | $86.05 \pm 0.46$ | $86.40 \pm 0.29$ | $87.66 \pm 0.28$ | $89.08 \pm 0.27$ |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | $86.05 \pm 0.36$ | $86.49 \pm 0.61$ | $87.19 \pm 0.36$ | $87.89 \pm 0.34$ | $88.26 \pm 0.37$ |
| | Standard LP $_{\text{ICML'21}}$[23] | $30.62 \pm 1.61$ | $42.64 \pm 2.03$ | $55.60 \pm 0.98$ | $67.32 \pm 0.98$ | $76.23 \pm 0.38$ |
| | LP++ | $84.24 \pm 1.36$ | $85.74 \pm 0.56$ | $86.94 \pm 0.48$ | $87.71 \pm 0.65$ | $88.38 \pm 0.61$ |
| | Zero-shot CLIP $_{\text{ICML'21}}$[23] | | | 77.35 | | |
| | CoOp $_{\text{IJCV'22}}$[29] | $75.58 \pm 1.29$ | $77.49 \pm 0.41$ | $77.93 \pm 0.58$ | $78.92 \pm 0.19$ | $79.21 \pm 0.36$ |
| | PLOT $_{\text{ICLR'23}}$[4] | $77.46 \pm 0.55$ | $77.72 \pm 0.26$ | $78.23 \pm 0.25$ | $78.40 \pm 0.35$ | $78.86 \pm 0.19$ |
| | KgCoOp $_{\text{CVPR'23}}$[27] | $77.20 \pm 0.77$ | $\mathbf{78.04} \pm 0.18$ | $77.97 \pm 0.28$ | $78.39 \pm 0.40$ | $78.73 \pm 0.23$ |
| | ProGrad $_{\text{ICCV'23}}$[30] | $\mathbf{78.36} \pm 0.41$ | $78.01 \pm 0.70$ | $\mathbf{78.38} \pm 0.87$ | $\mathbf{79.11} \pm 0.18$ | $\mathbf{79.51} \pm 0.23$ |
| *Food101* | CLIP-Adapter $_{\text{IJCV'23}}$[9] | $76.93 \pm 0.19$ | $77.22 \pm 0.15$ | $77.64 \pm 0.17$ | $77.97 \pm 0.22$ | $78.45 \pm 0.14$ |
| | Tip-Adapter-F $_{\text{ECCV'22}}$[28] | $77.53 \pm 0.14$ | $77.53 \pm 0.22$ | $77.82 \pm 0.27$ | $78.26 \pm 0.22$ | $78.99 \pm 0.15$ |
| | Tip-Adapter-F* $_{\text{ECCV'22}}$[28] | $77.58 \pm 0.10$ | $77.36 \pm 0.39$ | $77.78 \pm 0.15$ | $78.17 \pm 0.11$ | $78.72 \pm 0.06$ |
| | Standard LP $_{\text{ICML'21}}$[23] | $31.59 \pm 1.20$ | $44.60 \pm 1.03$ | $56.13 \pm 0.63$ | $64.45 \pm 0.55$ | $70.97 \pm 0.19$ |
| | LP++ | $76.61 \pm 0.77$ | $77.22 \pm 0.55$ | $77.79 \pm 0.34$ | $78.53 \pm 0.14$ | $78.88 \pm 0.19$ |

Table 6. **Comparison to state-of-the-art methods** (Continued). Average classification accuracy (%) and standard deviation over 10 tasks for 11 benchmarks, Best values are highlighted in bold.