TRANSLP++: TRANSDUCTIVE MULTIMODAL LINEAR PROBING FOR FEW-SHOT CLIP

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language Models (VLMs) such as CLIP provide strong transferable representations for few-shot learning. Existing methods such as Linear Probe++ (LP++) efficiently adapt pretrained VLMs through multimodal linear probing, but remain inductive and do not exploit unlabeled query samples. We propose **TransLP++**, a transductive extension of multimodal linear probing that incorporates query-set information into LP++ while keeping the pretrained VLM frozen. Specifically, TransLP++ jointly optimizes the multimodal classifier and auxiliary query assignments by combining labeled support supervision, query-set mutual information, and CLIP-guided semantic regularization. We develop an alternating optimization procedure with block-wise Lipschitz analysis and data-driven initialization for efficient adaptation. The proposed method requires no modification to the pretrained VLM and retains the lightweight, black-box nature of classifier-based adaptation. Experiments on 11 few-shot classification benchmarks and multiple CLIP backbones demonstrate consistent improvements over strong inductive and transductive baselines, including LP++, TransCLIP, and TIM++. Our code is available at `https://anonymous.4open.science/r/ICLR-8924-TransLP`.

1 INTRODUCTION

Vision-Language Models (VLMs), such as CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021), have emerged as powerful foundation models by learning transferable multimodal representations from large-scale image-text data. By aligning visual and linguistic representations in a shared embedding space, CLIP enables effective zero-shot recognition and provides strong generalizable representations for downstream vision tasks. These capabilities have motivated extensive research on adapting pretrained VLMs to downstream tasks with limited supervision.

Few-shot learning is particularly important in this context, where only a small number of labeled examples are available for each target class. Although pretrained VLMs provide strong transferable representations, directly adapting them to a new task remains challenging because a limited number of labeled examples provides insufficient supervision for reliable adaptation. Existing few-shot VLM adaptation methods can be broadly categorized into prompt-based, adapter-based, and

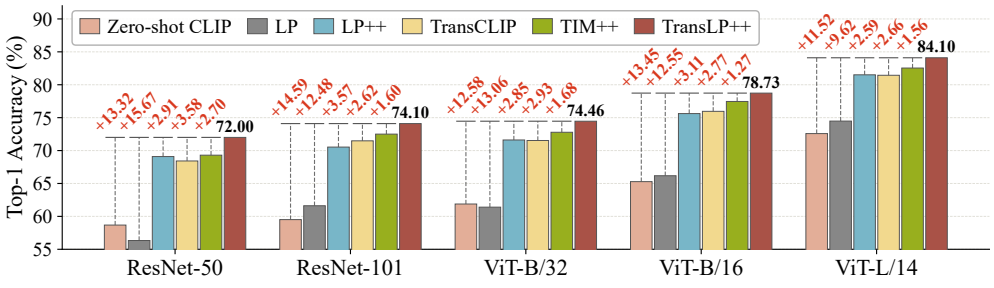


Figure 1: **TransLP++ consistently outperforms state-of-the-art methods across CLIP backbones.** Top-1 accuracy in the 4-shot setting is reported, averaged over 11 datasets and three random seeds.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

classifier-based adaptation. Prompt-based methods, such as CoOp (Zhou et al., 2022b), CoCoOp (Zhou et al., 2022a), ProGrad (Zhu et al., 2023), and KgCoOP (Yao et al., 2023), optimize learnable textual prompts while keeping the pretrained VLM largely frozen. Adapter-based approaches, including CLIP-Adapter (Gao et al., 2024) and Tip-Adapter (Zhang et al., 2022), introduce lightweight trainable modules or cache-based mechanisms to exploit the few-shot support set. In contrast, classifier-based approaches, including LP++ (Huang et al., 2024), TransCLIP (Zanella et al., 2024a), and TIM++ (Li et al., 2026), directly adapt lightweight classifiers or class prototypes on top of frozen VLM representations without introducing additional adaptation modules.

Among these approaches, classifier-based adaptation is particularly attractive for its simplicity, efficiency, and low computational overhead. A representative approach is Linear Probe (LP), which freezes the pretrained visual encoder and learns a linear classifier from the labeled support set. While effective, standard LP does not explicitly exploit the semantic knowledge encoded by the textual encoder. LP++ (Huang et al., 2024) addresses this limitation by incorporating both visual prototypes and text embeddings into the classifier, providing a strong and lightweight multimodal linear probing framework for few-shot CLIP adaptation. Importantly, LP++ keeps the pretrained VLM frozen and performs adaptation only at the classifier level, making it computationally efficient and suitable for black-box VLM settings.

Despite its effectiveness, LP++ follows an *inductive* learning paradigm. The classifier is optimized using the labeled support set, while query samples are subsequently classified independently without contributing to the adaptation process. Consequently, potentially valuable information in the unlabeled query set remains underexploited, which is particularly relevant in few-shot settings where the support set contains only a limited number of labeled examples. This limitation naturally motivates *transductive* learning, which incorporates unlabeled query samples into the adaptation process. Rather than treating query samples as independent test instances, transductive methods exploit their collective information to refine predictions and decision boundaries. Classical approaches such as LaplacianShot (Ziko et al., 2020) and Transductive Information Maximization (TIM) (Boudiaf et al., 2020) demonstrate that unlabeled query samples can provide useful information beyond the labeled support set. Recent methods have further explored transductive adaptation for VLMs, including TransCLIP (Zanella et al., 2024b), TIM++ (Li et al., 2026), and LIMO (Baklouti et al., 2026). However, LP++ does not provide a mechanism to incorporate query-set information into its multimodal linear probing adaptation. This motivates the following question: *Can LP++ be naturally extended from inductive to transductive learning while preserving its lightweight, computationally efficient, and black-box properties?*

To address this question, we propose **TransLP++**, a transductive extension of LP++ (Huang et al., 2024). TransLP++ incorporates unlabeled query-set information into multimodal linear probing adaptation while keeping the pretrained VLM frozen. We further develop an alternating optimization procedure and evaluate the method across diverse few-shot benchmarks and CLIP backbones. Our main contributions are summarized as follows:

- We propose **TransLP++**, a transductive multimodal linear probing framework that extends the inductive LP++ paradigm to explicitly exploit unlabeled query-set information. Its unified formulation jointly leverages support supervision, query-set information, and CLIP semantic knowledge while keeping the pretrained VLM frozen.
- We develop an alternating optimization procedure with block-wise Lipschitz analysis, which provides theoretically motivated step sizes for classifier updates. In practice, we further employ momentum-based optimization to improve optimization efficiency and stability.
- Extensive experiments on 11 diverse few-shot classification benchmarks and multiple CLIP backbones demonstrate consistent improvements over strong inductive and transductive baselines, including LP++, TransCLIP, and TIM++, as shown in Figure 1.

2 Related Work

Transductive Few-Shot Learning. Inductive few-shot learning relies solely on the labeled support set to adapt the model and then predicts unseen query samples independently. In contrast, transductive few-shot learning incorporates the unlabeled query set into the adaptation process, allowing

the model to exploit information from both labeled and unlabeled samples. The key idea is that the query samples provide additional information about the target distribution, which can be used to refine class assignments and decision boundaries beyond what can be inferred from the limited support set alone. A broad range of transductive few-shot methods has been developed in the vision-only setting Dhillon et al. (2019); Ziko et al. (2020); Liu et al. (2020); Hu et al. (2021). These methods explore different forms of query-set information, including entropy regularization in Transductive Fine-Tuning (TF) Dhillon et al. (2019), graph-based consistency in LaplacianShot Ziko et al. (2020), prototype refinement in BD-CSPN Liu et al. (2020), and distributional or class-proportion priors in PT-MAP Hu et al. (2021). Transductive Information Maximization (TIM) (Boudiaf et al., 2020) instead maximizes the mutual information between query samples and their predicted labels while leveraging labeled support supervision. These studies demonstrate the value of jointly modeling unlabeled query samples in low-data regimes.

Transductive Adaptation of Vision-Language Models. Recent studies have begun to investigate transductive adaptation in the context of pretrained VLMs. TransCLIP (Zanella et al., 2024b) performs transductive CLIP adaptation by jointly modeling unlabeled query features with a Gaussian mixture model and aligning their predictions with CLIP-derived textual distributions via Kullback–Leibler (KL) divergence. TIM++ (Li et al., 2026) further enhances the classical TIM framework for few-shot CLIP by incorporating textual knowledge from the pretrained VLM through CLIP-guided regularization and prototype initialization. More recently, LIMO (Baklouti et al., 2026) introduces a language-aware information maximization framework for transductive few-shot CLIP and explores parameter-efficient fine-tuning strategies. These methods demonstrate the benefit of exploiting unlabeled query-set information for adapting pretrained VLMs.

Despite these advances, LP++ remains inductive and does not exploit the collective information in the unlabeled query set during adaptation. The proposed TransLP++ extends LP++ to the transductive setting by incorporating information from the unlabeled query set into multimodal linear probing adaptation while keeping the pretrained VLM frozen, thereby retaining the lightweight and black-box nature of classifier-level adaptation.

3 METHOD

3.1 PRELIMINARIES

Given a pretrained vision-language model (VLM) such as CLIP, let $\boldsymbol{x}_i$ denote an input image and $\boldsymbol{c}_k$ denote the textual prompt corresponding to the k-th class, where $k = 1, \dots, K$. The l_2-normalized image and text embeddings are obtained by $\boldsymbol{f}_i = \frac{\theta_v(\boldsymbol{x}_i)}{\|\theta_v(\boldsymbol{x}_i)\|_2} \in \mathbb{R}^d$, $\boldsymbol{t}_k = \frac{\theta_t(\boldsymbol{c}_k)}{\|\theta_t(\boldsymbol{c}_k)\|_2} \in \mathbb{R}^d$, where $\theta_v(\cdot)$ and $\theta_t(\cdot)$ denote the visual and text encoders of the pretrained VLM, respectively.

Linear Probe (LP) (Radford et al., 2021) trains a linear classifier on top of the frozen image features. Let $\mathbf{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_K] \in \mathbb{R}^{d \times K}$ denote the learnable classifier weights, where $\boldsymbol{w}_k \in \mathbb{R}^d$ represents the weight vector (or visual prototype) for the k-th class. The classification logits are computed as $l_{ik} = \boldsymbol{f}_i^\top \boldsymbol{w}_k$. LP++ (Huang et al., 2024) extends LP by incorporating the semantic information encoded in the text embeddings. Specifically, the logits are computed as $l_{ik} = \boldsymbol{f}_i^\top(\boldsymbol{w}_k + \alpha_k \boldsymbol{t}_k)$, where α_k is a learnable class-specific coefficient balancing the contributions of the visual prototype and the text embedding. LP can be viewed as a special case of LP++ with $\alpha_k = 0$ for all classes. We further denote $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_K) \in \mathbb{R}^K$. Then the predicted probability for the k-th class is defined as:

$$p_{ik} = \frac{\exp(\tau l_{ik})}{\sum_{j=1}^{K} \exp(\tau l_{ij})} = \frac{\exp\big(\tau \boldsymbol{f}_i^\top(\boldsymbol{w}_k + \alpha_k \boldsymbol{t}_k)\big)}{\sum_{j=1}^{K} \exp\big(\tau \boldsymbol{f}_i^\top(\boldsymbol{w}_j + \alpha_j \boldsymbol{t}_j)\big)}, \tag{1}$$

where $\boldsymbol{p}_i = (p_{i1}, \dots, p_{iK}) \in \Delta_K$ denotes the predicted class distribution, and τ is the temperature parameter. We further denote the prediction matrix by $\boldsymbol{p} = [p_{ik}] \in \mathbb{R}^{N \times K}$ for a set of N samples. Here, Δ_K denotes the K-dimensional probability simplex. Both LP and LP++ are optimized using the standard cross-entropy loss: $L_{\mathrm{CE}}(\mathbf{W}, \boldsymbol{\alpha}) = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{k=1}^{K} y_{ik} \log p_{ik}$, where $y_{ik} \in \{0, 1\}$ denotes the ground-truth class indicator, and $\mathcal{S}$ denotes the *support* set with size $|\mathcal{S}|$. In parallel, we denote the *query* set as $\mathcal{Q}$, with size $|\mathcal{Q}|$.

3.2 FORMULATION OF TRANSLP++

Both LP and LP++ follow an inductive learning paradigm, where the classifier is adapted on the labeled support set and the query samples are subsequently classified independently. To extend this classifier-level adaptation to the transductive setting, we propose **TransLP++**, in which the unlabeled query set participates directly in classifier adaptation through a transductive objective. Specifically, TransLP++ combines three complementary signals: labeled support supervision, entropy-based mutual information over the unlabeled query set, and the semantic prior encoded by the pretrained VLM. Accordingly, we formulate the objective of TransLP++ as

$$\min_{\mathbf{W},\,\boldsymbol{\alpha}} L(\mathbf{W},\boldsymbol{\alpha}) = \min_{\mathbf{W},\,\boldsymbol{\alpha}} \underbrace{-\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log(p_{ik})}_{\text{Support supervision}} + \underbrace{\sum_{k=1}^{K}\widehat{p}_k\log\widehat{p}_k - \frac{\beta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}p_{ik}\log(p_{ik})}_{\text{Query-set mutual information}} + \underbrace{\frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}p_{ik}\log\frac{p_{ik}}{\hat{y}_{ik}}}_{\text{CLIP-guided regularization}}, \tag{2}$$

where p_{ik} denotes the predicted probability of sample i belonging to class k, as defined in Eq. (1). The coefficients $\mu > 0$ and $\gamma > 0$ control the strengths of the support supervision and the CLIP-guided regularization, while $\beta > 0$ balances the two entropy components of the query-set mutual information. We detail the three terms below.

Support supervision. In the few-shot setting, the limited support set may not fully represent the target distribution, and individual samples can be ambiguous or atypical. Direct optimization with hard one-hot labels may therefore lead to excessive confidence on individual support samples. To mitigate this issue, we employ label smoothing by replacing the one-hot target y_{ik} with $\tilde{y}_{ik} = (1-\epsilon)y_{ik} + \frac{\epsilon}{K}$, where $\epsilon \in [0,1)$ denotes the smoothing factor. The first term of Eq. (2) is thus a label-smoothed cross-entropy on the support set. Compared with hard-label supervision, it reduces excessive confidence on individual support examples and provides a more robust supervised signal for transductive adaptation.

Query-set mutual information. To exploit collective information from the unlabeled query set, we use an entropy-based objective motivated by mutual information. Specifically, the query-set mutual information term in Eq. (2) is

$$\underbrace{\sum_{k=1}^{K}\widehat{p}_k\log\widehat{p}_k}_{\text{Negative marginal entropy}} \quad \underbrace{-\frac{\beta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}p_{ik}\log p_{ik}}_{\text{Conditional entropy}}, \tag{3}$$

where $\widehat{p}_k = \frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} p_{ik}$ represents the empirical marginal probability of class k, and β controls the weight of the conditional entropy term. When $\beta = 1$, this objective is exactly the negative mutual information. The negative marginal entropy term discourages degenerate class assignments over the query set, while the conditional entropy term encourages confident predictions for individual query samples.

CLIP-guided semantic regularization. Although the query-set structure provides useful distributional information, transductive optimization alone may lead to predictions that deviate from the semantic knowledge encoded by the pretrained CLIP. We therefore regularize the adapted predictions toward CLIP's zero-shot predictions. For each query sample, the zero-shot probability of class k is defined as $\hat{y}_{ik} = \mathrm{softmax}(\eta \boldsymbol{f}_i^\top \boldsymbol{t}_k)$, where η is the temperature for zero-shot prediction, and $\boldsymbol{f}_i$ and $\boldsymbol{t}_k$ are the query image feature and the class text feature, respectively. Let $\hat{\boldsymbol{y}}_i = (\hat{y}_{i1}, \ldots, \hat{y}_{iK})$ denote the corresponding zero-shot prediction distribution for sample i. The third term of Eq. (2) is then the KL divergence $\mathcal{D}_{\mathrm{KL}}(\boldsymbol{p}_i \,\|\, \hat{\boldsymbol{y}}_i)$ averaged over the query set. Minimizing it encourages the adapted posterior to remain close to CLIP's zero-shot prediction while allowing the classifier to exploit the target query information.

3.3 OPTIMIZATION

In this subsection, we address the optimization problem in Eq. (2). Directly solving this objective is non-trivial because the nonlinear entropy and KL divergence terms involve the prediction probabilities p_{ik} in both linear and logarithmic forms. To facilitate optimization, we introduce auxiliary variables representing the latent class assignments of the query samples. This yields a relaxed constrained objective in which the prediction probabilities p_{ik} appear only inside logarithmic terms. The resulting formulation is given below, with its derivation provided in Appendix A.1.

Proposition 1. *The objective in Eq. (2) admits the following relaxed constrained formulation:*

$$\begin{aligned}\min_{\mathbf{W},\,\boldsymbol{\alpha},\,\boldsymbol{q}} \quad & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log(p_{ik}) + \sum_{k=1}^{K}\widehat{q}_k\log\widehat{q}_k \; - \frac{\beta+\delta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log(p_{ik}) \\ & - \frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log\hat{y}_{ik} + \frac{\delta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log q_{ik} \qquad (4) \\ \text{s.t.} \quad & \sum_{k=1}^{K}q_{ik}=1, \quad i\in\mathcal{Q}, \quad q_{ik}\geq 0, \quad i\in\mathcal{Q}, \quad k\in\{1,\cdots,K\},\end{aligned}$$

where $\boldsymbol{q}=[q_{ik}]\in\mathbb{R}^{|\mathcal{Q}|\times K}$ denotes the auxiliary class assignment probabilities for query samples, and $\widehat{q}_k=\frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}$ denotes the corresponding marginal assignment probability. The parameter $\delta>0$ controls the KL-based penalty that encourages consistency between the auxiliary assignments $\boldsymbol{q}$ and the model predictions $\boldsymbol{p}$.

The relaxed formulation in Proposition 1 introduces two sets of variables, namely the classifier parameters $(\mathbf{W},\boldsymbol{\alpha})$ and the auxiliary query assignments $\boldsymbol{q}$. We therefore employ an alternating optimization strategy that decomposes the problem into two blocks. Specifically, we alternately optimize $(\mathbf{W},\boldsymbol{\alpha})$ with $\boldsymbol{q}$ fixed and update $\boldsymbol{q}$ with the classifier parameters fixed. This decomposition allows the classifier-related and query-assignment components to be handled separately while preserving their interaction through the KL-based consistency penalty.

Updating $\boldsymbol{q}$ with fixed $\mathbf{W}$ and $\boldsymbol{\alpha}$. With the classifier parameters $\mathbf{W}$ and $\boldsymbol{\alpha}$ fixed, the relaxed objective in Eq. (4) is convex with respect to the auxiliary assignments $\boldsymbol{q}$ under the probability-simplex constraints. We therefore derive the optimality condition for $\boldsymbol{q}$ using the Karush-Kuhn-Tucker (KKT) conditions. The resulting update is given in the following proposition, with its complete derivation provided in Appendix A.2.

Proposition 2. *Applying the Karush-Kuhn-Tucker (KKT) conditions yields the following update for the query assignments:*

$$q_{ik}=\left(\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}}\right)\Bigg/\left(\sum_{j=1}^{K}\frac{p_{ij}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ij}^{\frac{\gamma}{\delta}}}{\widehat{q}_j^{\frac{1}{\delta}}}\right). \qquad (5)$$

Since $\widehat{q}_k$ depends on the assignments $\{q_{ik}\}_{i\in\mathcal{Q}}$, Eq. (5) cannot be directly solved. We therefore use the following approximate update:

$$q_{ik}\propto\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\left(\sum_{i\in\mathcal{Q}}p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ik}^{\frac{\gamma}{\delta}}\right)^{\frac{1}{\delta+1}}}. \qquad (6)$$

Update $\mathbf{W}$ and $\boldsymbol{\alpha}$ with fixed $\boldsymbol{q}$. With $\boldsymbol{q}$ fixed, we optimize the classifier parameters $\mathbf{W}$ and $\boldsymbol{\alpha}$. Substituting the expression for p_{ik} in Eq. (1) into Eq. (4), we isolate the terms that depend on $(\mathbf{W},\boldsymbol{\alpha})$ and obtain the sub-objective $L(\mathbf{W},\boldsymbol{\alpha};\boldsymbol{q})$:

$$\begin{aligned}L(\mathbf{W},\boldsymbol{\alpha};\boldsymbol{q}) = & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}[\tau\,\boldsymbol{f}_i^{\top}(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)] + \frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^{\top}(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big) \\ & - \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}[\tau\,\boldsymbol{f}_i^{\top}(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)] + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^{\top}(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big)\end{aligned} \qquad (7)$$

For the $\mathbf{W}$ and $\boldsymbol{\alpha}$ blocks, $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ is smooth and block-coordinatewise convex but does not admit a closed-form solution due to the softmax coupling. Following LP++ (Huang et al., 2024), we therefore employ a block-coordinate Majorize-Minimize (MM) scheme, where each block update is equivalent to a gradient descent step on a quadratic majorizer, with a task-specific step size determined by its block-specific Lipschitz constant (Huang et al., 2024; Bubeck et al., 2015). At iteration j, the corresponding majorizing functions for $\mathbf{W}$ and $\boldsymbol{\alpha}$ are given by:

$$\begin{aligned} M(\mathbf{W}, \mathbf{W}^{(j)}) &= L(\mathbf{W}^{(j)}; \boldsymbol{\alpha}, \boldsymbol{q}) + \nabla L_{\mathbf{W}}(\mathbf{W}^{(j)})^\top (\mathbf{W} - \mathbf{W}^{(j)}) + \frac{\gamma_{\mathbf{W}}}{2} \|\mathbf{W} - \mathbf{W}^{(j)}\|^2, \\ M(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(j)}) &= L(\boldsymbol{\alpha}^{(j)}; \mathbf{W}, \boldsymbol{q}) + \nabla L_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^{(j)})^\top (\boldsymbol{\alpha} - \boldsymbol{\alpha}^{(j)}) + \frac{\gamma_{\boldsymbol{\alpha}}}{2} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^{(j)}\|^2, \end{aligned} \tag{8}$$

where $\gamma_{\mathbf{W}}$ and $\gamma_{\boldsymbol{\alpha}}$ are the block-specific Lipschitz constants. Then the following updates minimize these majorizing functions:

$$\begin{aligned} \mathbf{W}^{(j+1)} &= \mathbf{W}^{(j)} - \frac{1}{\gamma_{\mathbf{W}}} \nabla L_{\mathbf{W}}(\mathbf{W}^{(j)}) = \arg\min_{\mathbf{W}} M(\mathbf{W}, \mathbf{W}^{(j)}), \\ \boldsymbol{\alpha}^{(j+1)} &= \boldsymbol{\alpha}^{(j)} - \frac{1}{\gamma_{\boldsymbol{\alpha}}} \nabla L_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}^{(j)}) = \arg\min_{\boldsymbol{\alpha}} M(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(j)}). \end{aligned} \tag{9}$$

Following LP++, we adopt an essentially cyclic block-coordinate strategy, where multiple successive updates are performed on one block while keeping the other fixed, alternating cyclically between blocks. Theorem 2.2 in LP++ ensures the sublinear convergence of these updates. To implement the above updates, we must determine the Lipschitz constants $\gamma_{\mathbf{W}}$ and $\gamma_{\boldsymbol{\alpha}}$ for the sub-objective $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$. A valid choice for the Lipschitz constant is the largest eigenvalue of the Hessian, but performing a naive spectral decomposition on the full Hessian can be computationally intensive. To address this, we derive approximate block-wise Lipschitz constants in the following proposition, with detailed proofs provided in Appendix A.3. This reduces the eigenvalue computation from a $KD \times KD$ matrix to a much smaller $D \times D$ matrix.

Proposition 3. *Considering the variable blocks $\mathbf{W}$ and $\boldsymbol{\alpha}$, the gradient of $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ in Eq. (7) is block-coordinatewise Lipschitz continuous. For $\pi_1 \geq 2$, a valid block Lipschitz constant with respect to $\mathbf{W}$ is*

$$\gamma_{\mathbf{W}} = \frac{\pi_1 \tau^2}{4} \lambda_{\max}\Big(\frac{\mu}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \boldsymbol{f}_i \boldsymbol{f}_i^\top + \frac{\delta + \beta - \gamma}{|\mathcal{Q}|} \sum_{i \in \mathcal{Q}} \boldsymbol{f}_i \boldsymbol{f}_i^\top\Big). \tag{10}$$

For practical use, we also consider $\pi_1 \geq 1$ under a diagonal Hessian approximation, which yields the same expression as an approximate block Lipschitz constant. Similarly, for $\pi_2 \geq 1$, an approximate block Lipschitz constant for $\boldsymbol{\alpha}$ is

$$\gamma_{\boldsymbol{\alpha}} = \frac{\pi_2 \tau^2}{4} \cdot \max_k \Big(\frac{\mu}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{\delta + \beta - \gamma}{|\mathcal{Q}|} \sum_{i \in \mathcal{Q}} (\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2\Big). \tag{11}$$

Here, $\lambda_{\max}(\mathbf{A})$ denotes the largest eigenvalue of matrix $\mathbf{A}$, and we require $\delta + \beta - \gamma > 0$ for stability. The corresponding gradient descent step sizes are $1/\gamma_{\mathbf{W}}$ and $1/\gamma_{\boldsymbol{\alpha}}$, respectively.

Initialization of the variables. Initialization plays an important role in the alternating optimization procedure. We initialize the auxiliary assignments $\boldsymbol{q}$ with the zero-shot pseudo-labels $q_{ik}^{(0)} = \mathrm{softmax}(\eta \boldsymbol{f}_i^\top \boldsymbol{t}_k)$. The classifier parameters are then initialized using a closed-form approximation to the minimizer of the sub-objective $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ under these assignments. Specifically, the resulting initialization combines support- and query-based class centroids as

$$\begin{aligned} \boldsymbol{w}_k^{(0)} &\propto \mu \frac{\sum_{i \in \mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i}{\sum_{i \in \mathcal{S}} \tilde{y}_{ik}} + (\delta + \beta - \gamma) \frac{\sum_{i \in \mathcal{Q}} q_{ik}^{(0)} \boldsymbol{f}_i}{\sum_{i \in \mathcal{Q}} q_{ik}^{(0)}}, \\ \alpha_k^{(0)} &\propto \mu \frac{\sum_{i \in \mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i \in \mathcal{S}} \tilde{y}_{ik}} + (\delta + \beta - \gamma) \frac{\sum_{i \in \mathcal{Q}} q_{ik}^{(0)} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i \in \mathcal{Q}} q_{ik}^{(0)}}. \end{aligned} \tag{12}$$

The resulting initializers are normalized to remove their global scaling factors. Detailed derivations are provided in Appendix B.

Practical optimization with Momentum. Although the Lipschitz-guided updates in Eq. (9) provide a theoretically justified optimization scheme, their step sizes can be conservative due to worst-case curvature bounds. To improve practical efficiency, we use a momentum-based variant to accelerate the optimization of $\mathbf{W}$ and $\boldsymbol{\alpha}$. After each $\mathbf{W}$ update, we normalize each $\boldsymbol{w}_k$ to unit ℓ_2 norm to stabilize the scale of the learned visual prototypes. As shown in the experiments, the Lipschitz-guided Gradient Descent (GD) and momentum-based variants of TransLP++ achieve comparable performance, while the latter requires fewer optimization iterations and substantially reduces the adaptation time. The complete procedures for both variants are provided in Appendix 1.

4 Experiments

4.1 Experimental Settings

Datasets. Following the evaluation protocols of TransCLIP (Zanella et al., 2024b), LP++ (Huang et al., 2024), and TIM++ (Li et al., 2026), we evaluate our proposed TransLP++ on 11 widely used public datasets: ImageNet (Deng et al., 2009), SUN397 (Xiao et al., 2010), FGVC-Aircraft (Maji et al., 2013), EuroSAT (Helber et al., 2019), Stanford Cars (Krause et al., 2013), Food101 (Bossard et al., 2014), Oxford Pets (Parkhi et al., 2012), Oxford Flowers (Nilsback & Zisserman, 2008), Caltech101 (Fei-Fei et al., 2004), DTD (Cimpoi et al., 2014), and UCF101 (Soomro et al., 2012).

Comparison Methods. We include LP (Radford et al., 2021) and LP++ (Huang et al., 2024) as direct baselines, along with zero-shot CLIP as a reference for few-shot adaptation. We further compare TransLP++ with representative state-of-the-art few-shot CLIP methods, including TransCLIP (Zanella et al., 2024b) and TIM++ (Li et al., 2026). All methods operate in a black-box setting without access to the internal parameters or architecture of the pretrained VLM. We follow the official implementations and recommended hyperparameters of all baselines, and evaluate all methods under the same datasets, few-shot settings, and evaluation protocols.

Implementation Details. Experiments are conducted on an Intel Core i9-14900K CPU with an NVIDIA GeForce RTX 4090 D GPU and 128 GB RAM, using PyTorch 2.4.1 with CUDA 12.5 on Ubuntu 20.04. We use three random seeds (1, 2, and 3) and report the average Top-1 accuracy. Unless otherwise specified, we use the pretrained CLIP ViT-B/16 backbone and the momentum optimizer for faster runtime. We set $T = 1000$ outer iterations, with $\text{iter}_{\mathbf{W}} = 20$ and $\text{iter}_{\boldsymbol{\alpha}} = 1$ inner updates per iteration. The block Lipschitz parameters are set to $\pi_1 = 2$ and $\pi_2 = 4$. We use $\mu = 1.0$, $\beta = 0.5$, $\gamma = 0.25$, $\delta = 1.0$, $\epsilon = 0.1$, $\eta = 60$, and $\tau = 30$ across all datasets, backbones, and few-shot settings. Here, $\delta + \beta - \gamma = 1.25 > 0$, satisfying the stability condition in our theoretical analysis. These hyperparameters are selected based on the best average validation performance across the 11 datasets under the 4-shot setting with seed 1 on CLIP ViT-B/16, and the selected configuration is fixed for all experiments. More details are provided in Appendix C.1.

4.2 Main Results

Comparison with State-of-the-Art Methods. We compare TransLP++ with representative few-shot CLIP methods. As shown in Table 1, Figure 2, and Figure 3 in Appendix, TransLP++ consistently achieves the highest average accuracy across all shot settings, reaching 75.88%, 77.25%, 78.73%, 80.36%, and 82.01% for 1-, 2-, 4-, 8-, and 16-shot settings, respectively. Compared with the best-performing baseline TIM++, TransLP++ improves the average accuracy across the 11 datasets by 1.56, 1.55, 1.27, 1.34, and 1.16 percentage points, respectively. Compared with the inductive LP++, TransLP++ achieves gains of 5.61, 4.42, 3.11, 2.42, and 1.74 points, with the largest improvement observed in the 1-shot setting and a gradual decrease as more labeled samples become available. This trend highlights the particular benefit of exploiting unlabeled query-set information with limited labeled supervision.

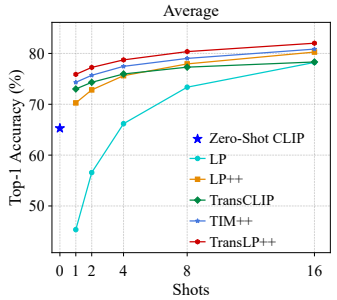


Figure 2: **TransLP++ consistently outperforms baselines across different shot settings.** Average accuracy (%) over 11 datasets.

Generalization Across VLM Backbones. We further evaluate TransLP++ across different CLIP backbones. As shown in Table 2 and Figure 1, TransLP++ consistently achieves the best average

Table 1: **Comparison with state-of-the-art methods across 11 datasets on ViT-B/16.** Results are averaged over three random seeds for each shot setting. **Best** and second-best results are reported, respectively.

Shot	Method	ImageNet	SUN397	Aircraft	EuroSAT	Cars	Food101	Pets	Flower	Caltech101	DTD	UCF101	Average
0	**CLIP-ViT-B/16**	66.70	62.56	24.72	48.31	65.68	85.88	89.07	70.77	93.31	43.50	67.54	65.28
1	LP ICML'21	30.93	40.00	20.13	54.07	34.21	43.79	40.50	71.25	79.76	35.64	48.57	45.35
	LP++ CVPR'24	67.93	67.55	29.04	64.95	67.03	83.20	89.22	86.29	92.86	52.54	72.32	70.27
	TransCLIP NeurIPS'24	70.08	70.64	29.84	72.46	70.89	87.88	91.74	84.80	93.14	53.25	78.39	73.01
	TIM++ AAAI'26	69.94	71.64	30.18	80.09	73.23	86.94	93.27	84.37	90.43	57.76	79.65	74.32
	TransLP++	68.70	72.76	29.55	85.87	73.42	87.23	93.85	91.57	89.71	59.85	82.14	**75.88**
2	LP ICML'21	43.18	52.67	23.92	59.62	48.65	61.63	54.44	84.50	85.71	46.00	61.91	56.57
	LP++ CVPR'24	68.92	70.13	32.30	67.97	70.35	86.12	89.89	90.45	94.12	56.03	74.83	72.83
	TransCLIP NeurIPS'24	70.38	70.86	29.97	77.76	71.72	87.01	92.32	90.54	93.47	54.98	78.46	74.32
	TIM++ AAAI'26	70.56	72.51	32.34	81.92	75.73	86.97	92.98	87.35	90.95	60.54	80.80	75.70
	TransLP++	69.23	73.90	31.84	86.70	75.55	87.21	93.52	94.22	91.28	63.16	83.11	**77.25**
4	LP ICML'21	53.91	63.00	29.50	69.49	61.89	72.83	67.80	91.35	91.76	56.11	70.30	66.18
	LP++ CVPR'24	70.36	73.39	32.80	73.98	74.54	85.99	91.10	92.96	94.89	62.23	79.62	75.62
	TransCLIP NeurIPS'24	70.40	71.72	34.65	79.35	73.97	86.41	91.26	93.57	94.04	61.09	79.10	75.96
	TIM++ AAAI'26	71.25	74.01	36.77	83.63	78.22	87.02	92.92	89.47	91.58	64.60	82.54	77.46
	TransLP++	70.39	75.06	36.05	86.98	78.15	87.27	93.48	95.53	91.72	66.53	84.88	**78.73**
8	LP ICML'21	61.80	69.40	36.33	78.65	70.96	79.26	80.04	95.72	93.91	63.75	77.13	73.36
	LP++ CVPR'24	71.71	75.17	38.96	76.35	76.14	86.68	92.01	95.29	95.43	67.42	82.17	77.94
	TransCLIP NeurIPS'24	70.27	73.20	36.43	79.74	76.92	86.70	91.91	93.84	94.17	65.68	81.45	77.30
	TIM++ AAAI'26	71.86	74.95	41.10	85.88	80.35	86.92	93.35	91.07	91.17	67.83	84.69	79.02
	TransLP++	71.78	76.19	40.24	88.38	81.15	87.29	93.96	95.95	92.66	69.80	86.56	**80.36**
16	LP ICML'21	67.26	73.18	42.73	86.07	79.77	83.60	85.71	96.93	95.09	69.50	80.70	78.23
	LP++ CVPR'24	72.64	76.13	41.47	84.65	80.62	87.24	92.90	96.44	95.88	71.34	83.67	80.27
	TransCLIP NeurIPS'24	71.77	74.67	38.58	81.83	79.73	86.86	92.46	94.38	93.93	65.15	82.10	78.32
	TIM++ AAAI'26	72.59	76.22	44.84	89.15	83.47	87.06	93.66	93.35	91.81	71.39	85.75	80.85
	TransLP++	73.42	77.51	45.18	90.79	84.07	87.43	94.12	96.98	92.87	72.62	87.07	**82.01**

accuracy across all backbones and shot settings. Compared with TIM++, the strongest competing method in most settings, TransLP++ yields improvements of 0.88 – 2.74 percentage points across the 25 backbone–shot combinations. Compared with LP++, its inductive counterpart, TransLP++ improves the average accuracy by 0.45 – 5.79 points across the same settings. The consistent gains are observed on both convolutional and vision-transformer architectures, as well as across different model capacities, demonstrating the robustness of TransLP++ to the underlying VLM backbone.

4.3 Ablation Study

We conduct an ablation study on the two query-related information sources in TransLP++. As shown in Table 3, adding query-set mutual information (MI) to the support-only variant improves accuracy by 14.72, 5.07, and 0.98 points under the 1-, 4-, and 16-shot settings, respectively, whereas adding only the CLIP-guided KL regularization term yields smaller gains of 10.72, 0.72, and -0.32 points. Combining both components achieves the best performance across all shot settings, reaching 75.88%, 78.73%, and 82.01%, with further gains of 2.18, 1.27, and 0.69 points over query MI alone. These results demonstrate the complementary roles of the two components, with query MI exploiting collective information from the unlabeled query set and the KL term incorporating the semantic prior encoded by pretrained CLIP.

Table 3: **Ablation of information sources in TransLP++.** Average accuracy over 11 datasets and three random seeds.

Variant	Query MI	CLIP KL	1-shot	4-shot	16-shot
Support	×	×	58.98	72.39	80.34
Support + KL	×	✓	69.70	73.11	80.02
Support + MI	✓	×	73.70	77.46	81.32
TransLP++	✓	✓	**75.88**	**78.73**	**82.01**

4.4 Computational Efficiency

We compare the computational efficiency of TransLP++ with representative few-shot CLIP methods. As shown in Table 4, all methods operate in a black-box setting without accessing the pretrained VLM. On `ImageNet` with 16 shots ($d = 512$, $K = 1000$, $|\mathcal{Q}| = 50{,}000$), the Lipschitz-guided GD implementation requires approximately 480 s, substantially longer than LP++ (113 s),

Table 2: **Comparison with state-of-the-art methods across different VLM backbones.** Average accuracy (%) over 11 datasets and three random seeds, with ± denoting the average per-dataset standard deviation. **Best** and second-best results are reported, respectively.

Method	1-shot	2-shot	4-shot	8-shot	16-shot
CLIP-ResNet-50			58.68		
$\text{LP}_{\text{ICML'21}}$	35.60 ± 1.23	45.38 ± 1.31	56.33 ± 0.84	64.21 ± 0.84	70.21 ± 0.64
$\text{LP++}_{\text{CVPR'24}}$	62.50 ± 1.29	65.23 ± 0.86	69.09 ± 0.47	71.72 ± 0.36	75.55 ± 0.30
$\text{TransCLIP}_{\text{NeurIPS'24}}$	65.80 ± 1.21	67.45 ± 1.13	68.42 ± 1.45	70.87 ± 0.81	72.73 ± 0.95
$\text{TIM++}_{\text{AAAI'26}}$	65.18 ± 0.88	67.44 ± 0.73	69.30 ± 0.70	71.53 ± 0.70	73.92 ± 0.46
TransLP++	**67.69** ± 0.47	**70.18** ± 0.81	**72.00** ± 0.77	**73.83** ± 0.43	**76.00** ± 0.47
CLIP-ResNet-101			59.51		
$\text{LP}_{\text{ICML'21}}$	40.31 ± 1.48	49.60 ± 1.94	61.62 ± 0.67	68.84 ± 0.49	74.04 ± 0.64
$\text{LP++}_{\text{CVPR'24}}$	64.72 ± 1.14	66.92 ± 0.75	70.53 ± 0.44	73.16 ± 0.33	75.11 ± 0.29
$\text{TransCLIP}_{\text{NeurIPS'24}}$	68.29 ± 1.12	69.42 ± 1.10	71.48 ± 0.77	72.80 ± 0.80	73.71 ± 0.94
$\text{TIM++}_{\text{AAAI'26}}$	68.42 ± 0.62	70.16 ± 1.05	72.50 ± 0.59	74.55 ± 0.57	76.48 ± 0.53
TransLP++	**70.51** ± 0.74	**72.46** ± 0.85	**74.10** ± 0.48	**76.21** ± 0.43	**78.09** ± 0.45
CLIP-ViT-B/32			61.88		
$\text{LP}_{\text{ICML'21}}$	40.42 ± 1.44	51.39 ± 1.42	61.40 ± 0.83	68.61 ± 0.65	73.79 ± 0.44
$\text{LP++}_{\text{CVPR'24}}$	66.22 ± 1.08	68.78 ± 0.62	71.61 ± 0.30	74.22 ±0.52	76.59 ± 0.35
$\text{TransCLIP}_{\text{NeurIPS'24}}$	68.84 ± 2.15	70.55 ± 1.03	71.53 ± 0.94	73.15 ±0.55	74.18 ± 0.43
$\text{TIM++}_{\text{AAAI'26}}$	69.08 ± 0.86	70.57 ± 1.02	72.78 ± 0.67	74.58 ± 0.55	76.23 ± 0.36
TransLP++	**70.98** ± 1.00	**72.77** ± 0.62	**74.46** ± 0.48	**76.23** ± 0.32	**78.13** ± 0.39
CLIP-ViT-B/16			65.28		
$\text{LP}_{\text{ICML'21}}$	45.35 ± 1.44	56.57 ± 1.06	66.18 ± 1.40	73.36 ± 0.48	78.23 ± 0.46
$\text{LP++}_{\text{CVPR'24}}$	70.27 ± 0.48	72.83 ± 0.89	75.62 ± 0.84	77.94 ± 0.63	80.27 ± 0.38
$\text{TransCLIP}_{\text{NeurIPS'24}}$	73.01 ± 1.39	74.32 ± 1.13	75.96 ± 0.68	77.30 ± 0.65	78.32 ± 0.66
$\text{TIM++}_{\text{AAAI'26}}$	74.32 ± 0.61	75.70 ± 0.64	77.46 ± 0.72	79.02 ± 0.48	80.85 ± 0.34
TransLP++	**75.88** ± 0.59	**77.25** ± 0.53	**78.73** ± 0.55	**80.36** ± 0.44	**82.01** ± 0.35
CLIP-ViT-L/14			72.58		
$\text{LP}_{\text{ICML'21}}$	54.14 ± 1.60	64.92 ± 1.35	74.48 ± 0.66	79.85 ± 0.48	83.61 ± 0.37
$\text{LP++}_{\text{CVPR'24}}$	77.03 ± 0.65	78.63 ± 0.51	81.51 ± 0.51	83.05 ± 0.54	85.13 ± 0.29
$\text{TransCLIP}_{\text{NeurIPS'24}}$	79.13 ± 1.81	80.17 ± 1.13	81.44 ± 0.71	82.70 ± 0.63	83.70 ± 0.43
$\text{TIM++}_{\text{AAAI'26}}$	79.47 ± 1.07	81.20 ± 0.43	82.54 ± 0.63	84.07 ± 0.40	85.53 ± 0.38
TransLP++	**81.76** ± 0.82	**82.89** ± 0.42	**84.10** ± 0.46	**85.10** ± 0.44	**86.41** ± 0.29

TransCLIP (57 s), and TIM++ (2.28 s), mainly due to its conservative learning rates and more optimization iterations. Replacing Lipschitz-guided GD with momentum reduces the runtime to 118 s while maintaining comparable performance, as shown in Table 5 and Appendix C.2. This acceleration is mainly attributed to the reduced number of optimization iterations. Notably, TransLP++ with momentum achieves a runtime comparable to LP++ (118 s vs. 113 s) while achieving substantially higher classification accuracy by exploiting unlabeled query-set information.

Table 4: **Computational efficiency comparison.** Results on `ImageNet` under the 16-shot setting. Transd.: transductive; TransLP++*: Lipschitz-guided GD; TransLP++: momentum.

Method	Transd.	Black Box	Params. (w/o Aux.)	Variables (w/ Aux.)	Adaptation Time
LP	×	✓	Kd	Kd	∼ 43min
LP++	×	✓	$K(d+1)$	$K(d+1)$	∼ 113s
TransCLIP	✓	✓	$d(K+1)$	$d(K+1)+\lvert\mathcal{Q}\rvert K$	∼ 57s
TIM++	✓	✓	Kd	$Kd+\lvert\mathcal{Q}\rvert K$	∼ 2.28 s
TransLP++*	✓	✓	$K(d+1)$	$K(d+1)+\lvert\mathcal{Q}\rvert K$	∼ 480s
TransLP++	✓	✓	$K(d+1)$	$K(d+1)+\lvert\mathcal{Q}\rvert K$	∼118s

Table 5: **Comparison of optimization strategies for TransLP++.** Average accuracy ± standard deviation over 11 datasets and three random seeds. TransLP++*: Lipschitz-guided GD; TransLP++: momentum.

Optimizer	T	$\text{iter}_{\mathbf{W}}$	1-shot	4-shot	16-shot
TransLP++*	2000	40	75.91 ± 0.74	78.83 ± 0.59	82.03 ± 0.33
TransLP++	1000	20	75.88 ± 0.59	78.73 ± 0.55	82.01 ± 0.35

5 Conclusions

In this work, we presented **TransLP++**, a transductive extension of multimodal linear probing for few-shot CLIP. It addresses the limitation of inductive LP++ by incorporating unlabeled query-set information. By jointly leveraging support supervision, query-set mutual information, and CLIP-guided semantic regularization, the proposed method effectively exploits information from both unlabeled samples and the pretrained CLIP. We further developed an alternating optimization framework with block-wise Lipschitz analysis and data-driven initialization, together with a momentum-based variant for efficient practical adaptation. Extensive experiments on 11 few-shot benchmarks and multiple CLIP backbones demonstrate consistent improvements over representative baselines, highlighting the effectiveness of transductive adaptation for few-shot VLMs.

AI USE STATEMENT

In this work, we used generative AI tools mainly for improving paragraph structure, polishing the writing and checking for coding errors. We also use generative AI tools to optimize plotting codes for clear and more effective visualization of the experimental results. All AI-assisted text editing and code modifications were manually inspected and verified to ensure accuracy.

We have **not** used generative AI tools for formulating conceptual frameworks, theoretical models, mathematical claims, or mathematical proofs. All theoretical analyses, mathematical derivations, and proofs for our TransLP++ were developed by the authors without AI assistance. The original implementation of our proposed TransLP++ was written by the authors, with AI used only to check the potential coding errors. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide complete theoretical proofs and detailed derivations in the Appendix. Our anonymous source code is available at `https://anonymous.4open.science/r/ICLR-8924-TransLP`. The `README.md` file provides step-by-step instructions for obtaining the 11 publicly available benchmark datasets and reproducing the main experimental results in this paper. All experiments use fixed random seeds to ensure reproducible results.

REFERENCES

Ghassen Baklouti, Maxime Zanella, and Ismail Ben Ayed. Language-aware information maximization for transductive few-shot clip. *Transactions on Machine Learning Research*, 2026-April, 2026. ISSN 2835-8856.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Malik Boudiaf, Imtiaz Ziko, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive information maximization for few-shot learning. *Advances in Neural Information Processing Systems*, 33:2445–2457, 2020.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. IEEE, 2009.

Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification, 2019.

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178. IEEE, 2004.

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. *International Journal of Computer Vision*, 132(2):581–595, 2024.

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*, 12(7):2217–2226, 2019.

Yuqing Hu, Vincent Gripon, and Stéphane Pateux. Leveraging the feature distribution in transfer-based few-shot learning. In *International conference on artificial neural networks*, pp. 487–499. Springer, 2021.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23773–23782, 2024.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana 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*, pp. 4904–4916. PMLR, 2021.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Yingping Li, Yutong Zou, Yunshi Huang, Changzhe Jiao, Xinlin Wang, Shen Peng, Zhang Guo, and Shuiping Gou. Tim++: Transductive information maximization for few-shot clip. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6671–6680, 2026.

Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *European conference on computer vision*, pp. 741–756. Springer, 2020.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft, 2013.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

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 *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

Hantao Yao, Rui Zhang, and Changsheng Xu. Visual-language prompt tuning with knowledge-guided context optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6757–6767, 2023.

Maxime Zanella, Benoît Gérin, and Ismail Ayed. Boosting vision-language models with transduction. *Advances in Neural Information Processing Systems*, 37:62223–62256, 2024a.

Maxime Zanella, Benoît Gérin, and Ismail B Ayed. Boosting vision-language models with transduction. *Advances in Neural Information Processing Systems*, 37:62223–62256, 2024b.

Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaption of clip for few-shot classification. In *European conference on computer vision*, pp. 493–510. Springer, 2022.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16816–16825, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 15659–15669, 2023.

Imtiaz Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian regularized few-shot learning. In *International conference on machine learning*, pp. 11660–11670. PMLR, 2020.

APPENDIX

A PROOFS OF THE PROPOSITIONS

A.1 PROOF OF PROPOSITION 1

Proof. The full objective function in Eq. (2) can be expanded and reformulated as

$$\begin{aligned}\min_{\mathbf{W},\boldsymbol{\alpha}} \quad & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log p_{ik}+\sum_{k=1}^{K}\widehat{p}_k\log\widehat{p}_k \\ & -\frac{\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}p_{ik}\log p_{ik}-\frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}p_{ik}\log\hat{y}_{ik}.\end{aligned} \tag{13}$$

The initial optimization problem can be reformulated using the Alternating Direction Method (ADM) approach by introducing auxiliary variables $\boldsymbol{q}=[q_{ik}]\in\mathbb{R}^{|\mathcal{Q}|\times K}$ and enforcing equality constraints $\boldsymbol{q}=\boldsymbol{p}$, with $\boldsymbol{p}=[p_{ik}]\in\mathbb{R}^{|\mathcal{Q}|\times K}$, in addition to pointwise simplex constraints:

$$\begin{aligned}\min_{\mathbf{W},\,\boldsymbol{\alpha}} \quad & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log(p_{ik})+\sum_{k=1}^{K}\widehat{q}_k\log\widehat{q}_k \\ & -\frac{\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log(p_{ik})-\frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log\hat{y}_{ik}, \\ \text{s.t.} \quad & q_{ik}=p_{ik},\quad i\in\mathcal{Q},\quad k\in\{1,\cdots,K\} \\ & \sum_{k=1}^{K}q_{ik}=1,\quad i\in\mathcal{Q} \\ & q_{ik}\geq 0,\quad i\in\mathcal{Q},\quad k\in\{1,\cdots,K\}.\end{aligned} \tag{14}$$

We then solve the constrained problem (14) with the penalty-based approach, by using the Kullback-Leibler(KL) divergence.

$$\mathcal{D}_{KL}(\boldsymbol{q}_i\parallel\boldsymbol{p}_i)=\sum_{k=1}^{K}q_{ik}\log\frac{q_{ik}}{p_{ik}} \tag{15}$$

The KL divergence encourages equality constraints $\boldsymbol{q}_i=\boldsymbol{p}_i$, where $\boldsymbol{q}_i=[q_{i1},\cdots,q_{iK}]$ are the auxiliary pointwise predictions, and $\boldsymbol{p}_i=[p_{i1},\cdots,p_{iK}]$ are our model's posteriors. Thus, the constrained problem becomes:

$$\begin{aligned}\min_{\mathbf{W},\,\boldsymbol{\alpha},\,\boldsymbol{q}} \quad & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log(p_{ik})+\sum_{k=1}^{K}\widehat{q}_k\log\widehat{q}_k\ -\frac{\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log(p_{ik}) \\ & -\frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log\hat{y}_{ik}+\frac{\delta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\mathcal{D}_{KL}(\boldsymbol{q}_i\parallel\boldsymbol{p}_i), \\ \text{s.t.} \quad & \sum_{k=1}^{K}q_{ik}=1,\quad i\in\mathcal{Q} \\ & q_{ik}\geq 0,\quad i\in\mathcal{Q},\quad k\in\{1,\cdots,K\},\end{aligned} \tag{16}$$

where δ is a weight parameter. Expanding the KL penalty $\mathcal{D}_{KL}(\boldsymbol{q}_i\parallel\boldsymbol{p}_i)=\sum_k q_{ik}\log q_{ik}-\sum_k q_{ik}\log p_{ik}$ and grouping the $\log p_{ik}$ terms yields the following objectives, as shown in Proposi-

tion 1 of the main paper.

$$\begin{aligned}\min_{\mathbf{W},\,\boldsymbol{\alpha},\,\boldsymbol{q}} \quad & -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\log(p_{ik}) + \sum_{k=1}^{K}\widehat{q}_k\log\widehat{q}_k \; - \frac{\beta+\delta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log(p_{ik}) \\ & - \frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log\hat{y}_{ik} + \frac{\delta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log q_{ik} \\ \text{s.t.} \quad & \sum_{k=1}^{K}q_{ik}=1, \quad i\in\mathcal{Q} \\ & q_{ik}\geq 0, \quad i\in\mathcal{Q}, \quad k\in\{1,\cdots,K\}.\end{aligned} \tag{17}$$

□

A.2 Proof of Proposition 2

Proof. When $\mathbf{W}$ and $\boldsymbol{\alpha}$ are fixed, the objective function (4) is convex (sum of linear and convex functions) w.r.t auxiliary variable $\boldsymbol{q}$, and the simplex constraints are affine. Therefore, we can minimize this constrained convex problem for each $\boldsymbol{q}_i$ by solving the Karush-Kuhn-Tucker (KKT) conditions. Note that, $\sum_{k=1}^{K}q_{ik}\log q_{ik}$ in the KL penalty term handles implicitly non-negativity constraints $\boldsymbol{q}_i\geq 0$. So we only consider to handle the probability simplex constraints $\sum_{k=1}^{K}q_{ik}=1$. Let $\epsilon_i\in\mathbb{R}$ denote the Lagrangian multiplier corresponding to this constraint.

Collecting all terms that involve $\boldsymbol{q}$ from Eq. (4):

$$\begin{aligned}\mathcal{L}_q = & \sum_{k=1}^{K}\widehat{q}_k\log\widehat{q}_k + \frac{\delta}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log q_{ik} \\ & - \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log(p_{ik}) - \frac{\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\log\hat{y}_{ik}.\end{aligned} \tag{18}$$

Then the Lagrangian function can be expressed as:

$$\mathcal{L} = \mathcal{L}_q + \sum_{i\in\mathcal{Q}}\epsilon_i\Big(\sum_{k=1}^{K}q_{ik}-1\Big) \tag{19}$$

Then set the gradient of the Lagrangian function to zero, we have

$$\frac{\partial\mathcal{L}}{\partial q_{ik}} = \frac{1}{|\mathcal{Q}|}(\log\widehat{q}_k+1) + \frac{\delta}{|\mathcal{Q}|}(\log q_{ik}+1) - \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\log(p_{ik}) - \frac{\gamma}{|\mathcal{Q}|}\log\hat{y}_{ik} + \epsilon_i = 0 \tag{20}$$

we can get

$$q_{ik} = \frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}}\exp\Big(\frac{-\epsilon_i|\mathcal{Q}|-1-\delta}{\delta}\Big) \tag{21}$$

Applying the constraints $\sum_{k=1}^{K}q_{ik}=1$ into Eq. (21), Lagrangian multiplier ϵ_i can be achieved:

$$\exp\Big(\frac{-\epsilon_i|\mathcal{Q}|-1-\delta}{\delta}\Big) = \Big(\sum_{j=1}^{K}\frac{p_{ij}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ij}^{\frac{\gamma}{\delta}}}{\widehat{q}_j^{\frac{1}{\delta}}}\Big)^{-1} \tag{22}$$

Plugging Eq. (22) into Eq. (21) yields:

$$q_{ik} = \Big(\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}}\Big)\Big/\Big(\sum_{j=1}^{K}\frac{p_{ij}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot\hat{y}_{ij}^{\frac{\gamma}{\delta}}}{\widehat{q}_j^{\frac{1}{\delta}}}\Big) \tag{23}$$

Till now, we have obtained the update formula of q_{ik}. Note that, $\widehat{q}_k=\frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}$, so Eq. (23) does not admit a direct closed form. To obtain the closed-form update in our algorithm, we relax

this formula by treating the per-sample normalization constant C_i as approximately constant across samples i. Under this relaxation,

$$\begin{aligned}\widehat{q}_k &= \frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} q_{ik} = \frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\Big[\Big(\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}}\Big)\Big/ C_i\Big] \\ &\approx \frac{1}{|\mathcal{Q}|}\cdot\frac{1}{C}\sum_{i\in\mathcal{Q}}\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}} \propto \sum_{i\in\mathcal{Q}}\frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\widehat{q}_k^{\frac{1}{\delta}}},\end{aligned} \tag{24}$$

where $C_i = \sum_{j=1}^{K}\frac{p_{ij}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ij}^{\frac{\gamma}{\delta}}}{\widehat{q}_j^{\frac{1}{\delta}}}$, and C is introduced as a sample-independent approximation to the normalization constant C_i. Then we can get

$$\widehat{q}_k \propto \Big(\sum_{i\in\mathcal{Q}} p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}\Big)^{\frac{\delta}{\delta+1}}. \tag{25}$$

Plugging this back in Eq. (23), we can get:

$$q_{ik} \propto \frac{p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\Big(\sum_{i\in\mathcal{Q}} p_{ik}^{\frac{\delta+\beta-\gamma}{\delta}}\cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}\Big)^{\frac{1}{\delta+1}}} \tag{26}$$

Clearly, $q_{ik} \geq 0$ because all factors p_{ik} and $\hat{y}_{ik}$ are non-negative. □

A.3 Proof of Proposition 3

We first derive the Hessian with respect to each variable block and then establish the corresponding block Lipschitz constants.

Lemma 1. *Suppose that $\boldsymbol{\alpha}$ and $\boldsymbol{q}$ are fixed. The Hessian of the objective function in Eq. (7) with respect to the block $\mathbf{W} \in \mathbb{R}^{KD}$, denoted by $H_{\mathbf{W}} \in \mathbb{R}^{KD\times KD}$, is*

$$H_{\mathbf{W}} = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big] + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big]. \tag{27}$$

Here, $\otimes$ denotes the Kronecker product, $\mathrm{diag}(\boldsymbol{p}_i)$ is the diagonal $K\times K$ matrix with diagonal entries p_{ik}, and $\boldsymbol{p}_i = (p_{i1},\ldots,p_{iK})^\top \in \mathbb{R}^K$.

Proof. Suppose that $\boldsymbol{\alpha}$ and $\boldsymbol{q}$ are fixed, the gradient of the objective function (7) with respect to the class prototype $\boldsymbol{w}_j$ is

$$\nabla_{\boldsymbol{w}_j} L = \frac{\mu\tau}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ij}-\tilde{y}_{ij})\boldsymbol{f}_i + \frac{(\delta+\beta-\gamma)\tau}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ij}-q_{ij})\boldsymbol{f}_i, \tag{28}$$

where p_{ij} is defined in Eq. (1).

For $j = l$, we have

$$\nabla_{\mathbf{w}_l} p_{ij} = \tau\,(p_{ij} - p_{ij}^2)\boldsymbol{f}_i. \tag{29}$$

Hence,

$$\nabla^2_{\mathbf{w}_j\mathbf{w}_l} L = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ij}-p_{ij}^2)\boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ij}-p_{ij}^2)\boldsymbol{f}_i\boldsymbol{f}_i^\top. \tag{30}$$

For $j \neq l$, we have

$$\nabla_{\mathbf{w}_l} p_{ij} = -\tau\, p_{ij}p_{il}\boldsymbol{f}_i, \tag{31}$$

which gives

$$\nabla^2_{\mathbf{w}_j\mathbf{w}_l} L = -\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} p_{ij}p_{il}\boldsymbol{f}_i\boldsymbol{f}_i^\top - \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} p_{ij}p_{il}\boldsymbol{f}_i\boldsymbol{f}_i^\top \tag{32}$$

So the Block-coordinatewise $KD \times KD$ Hessian matrix $H_{\mathbf{W}}$ can be summarized as

$$\begin{aligned} H_{\mathbf{W}} = &\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big] \\ &+ \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big], \end{aligned} \tag{33}$$

where $\otimes$ denotes the Kronecker matrix product, and $\mathrm{diag}(\boldsymbol{p}_i)$ represents a diagonal $K \times K$ matrix whose elements are given by p_{ik}, and $\boldsymbol{p}_i = (p_{ik})_{1\le k\le K}$. □

Lemma 2. *Let λ be an eigenvalue of $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top$. Then $\lambda \in [0, 1/2]$. Consequently, $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top$ is positive semidefinite, with its largest eigenvalue satisfying*

$$\lambda_{\max}(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i{\boldsymbol{p}_i}^\top) \le \frac{1}{2}. \tag{34}$$

Proof. This lemma corresponds to Lemma 5.1 in LP++ (Huang et al., 2024). We omit the proof here and refer the reader to the original paper for details. □

Lemma 3. *For $\pi_1 \ge 2$, the quantity $\gamma_{\mathbf{W}}$ defined below is a valid block Lipschitz constant for the gradient of the objective function in Eq. (7) with respect to $\mathbf{W}$:*

$$\gamma_{\mathbf{W}} = \frac{\pi_1\tau^2}{4}\lambda_{\max}\Big(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\boldsymbol{f}_i\boldsymbol{f}_i^\top\Big), \tag{35}$$

Here, $\lambda_{\max}(\mathbf{A})$ denotes the largest eigenvalue of matrix $\mathbf{A}$, and $\delta + \beta - \gamma > 0$ should hold. Furthermore, for $\pi_1 \ge 1$, the same quantity $\gamma_{\mathbf{W}}$ serves as an approximate block Lipschitz constant when the Hessian is replaced by its diagonal approximation.

Proof. From Lemma 2, we can easily conclude $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top - \frac{1}{2}\mathbf{I}_K$ is negative semi-definite. Furthermore, as $\boldsymbol{f}_i\boldsymbol{f}_i^\top$ is positive semi-definite, the kronecker multiplication $\big(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top - \frac{1}{2}\mathbf{I}_K\big)\otimes\big(\boldsymbol{f}_i\boldsymbol{f}_i^\top\big)$ is also negative semi-definite. Hence, if $\delta+\beta-\gamma > 0$, then the matrix

$$\begin{aligned} &H_{\mathbf{W}} - \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\frac{1}{2}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top) - \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\frac{1}{2}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top) \\ =&\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top - \frac{1}{2}\mathbf{I}_K)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)] \\ &+ \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top - \frac{1}{2}\mathbf{I}_K)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)] \end{aligned} \tag{36}$$

is negative semi-definite. Therefore,

$$\boldsymbol{x}^\top H_{\mathbf{W}}\boldsymbol{x} \le \boldsymbol{x}^\top\big[\frac{\mu\tau^2}{2|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top) + \frac{(\delta+\beta-\gamma)\tau^2}{2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big]\boldsymbol{x}. \tag{37}$$

This leads to

$$\max_{\|\boldsymbol{x}\|=1}\boldsymbol{x}^\top H_{\mathbf{W}}\boldsymbol{x} \le \max_{\|\boldsymbol{x}\|=1}\boldsymbol{x}^\top\big[\frac{\mu\tau^2}{2|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top) + \frac{(\delta+\beta-\gamma)\tau^2}{2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\mathbf{I}_K\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big]\boldsymbol{x}. \tag{38}$$

According to the variational principle $\lambda_{\max}(H) = \max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^\top H \boldsymbol{x}$, we can obtain

$$\lambda_{\max}(H_{\mathbf{W}}) \leq \lambda_{\max}\big(\frac{\mu\tau^2}{2|\mathcal{S}|}\sum_{i\in\mathcal{S}} \mathbf{I}_K \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^\top) + \frac{(\delta+\beta-\gamma)\tau^2}{2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \mathbf{I}_K \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big). \tag{39}$$

Using the linearity of the Kronecker product, this becomes

$$\lambda_{\max}(H_{\mathbf{W}}) \leq \lambda_{\max}\big(\mathbf{I}_K \otimes \big[\frac{\mu\tau^2}{2|\mathcal{S}|}\sum_{i\in\mathcal{S}} (\boldsymbol{f}_i\boldsymbol{f}_i^\top) + \frac{(\delta+\beta-\gamma)\tau^2}{2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} (\boldsymbol{f}_i\boldsymbol{f}_i^\top)\big]\big). \tag{40}$$

Note that, for symmetric matrices A and B, the eigenvalues of $A\otimes B$ are all pairwise products of the eigenvalues of A and B. Since all eigenvalues of $\mathbf{I}_K$ are equal to 1, it follows that $\lambda_{\max}(\mathbf{I}_K \otimes A) = \lambda_{\max}(A)$. Hence,

$$\lambda_{\max}(H_{\mathbf{W}}) \leq \frac{\tau^2}{2}\lambda_{\max}\big(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \boldsymbol{f}_i\boldsymbol{f}_i^\top\big). \tag{41}$$

Finally, define

$$\gamma_{\mathbf{W}} = \frac{\pi_1\tau^2}{4}\lambda_{\max}\left(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \boldsymbol{f}_i\boldsymbol{f}_i^\top\right). \tag{42}$$

Since $\pi_1 \geq 2$ implies $\pi_1/4 \geq 1/2$, we have

$$\gamma_{\mathbf{W}} \geq \frac{\tau^2}{2}\lambda_{\max}\left(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \boldsymbol{f}_i\boldsymbol{f}_i^\top\right). \tag{43}$$

Hence, $\gamma_{\mathbf{W}}$ is an upper bound on $\lambda_{\max}(H_{\mathbf{W}})$ if $\pi_1 \geq 2$.

Notablly, $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top$ is diagonally dominant. Besides, $\boldsymbol{p}_i = (p_{ik})_{1\leq k\leq K}$ is a probability vector with one element approximate to 1 and other elements approximate to 0. So $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top$ can be approximated by a $K \times K$ matrix $\tilde{\boldsymbol{D}}_i$, where $\tilde{\boldsymbol{D}}_i$ is diagonal and its diagonal elements are given by the diagonal elements of $\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top$, namely, $p_{ik} - p_{ik}^2, k = 1, \cdots, K$. Thus, the Hessian matrix $H_{\mathbf{W}}$ can be approximated by

$$H_{\mathbf{W}} \approx \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} [\tilde{\boldsymbol{D}}_i \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^\top)] + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} [\tilde{\boldsymbol{D}}_i \otimes (\boldsymbol{f}_i\boldsymbol{f}_i^\top)] \triangleq \tilde{H}_{\mathbf{W}}. \tag{44}$$

Since $p_{ik} - p_{ik}^2 = -(p_{ik} - \frac{1}{2})^2 + \frac{1}{4} \leq \frac{1}{4}$, then $\lambda_{max}(\tilde{\boldsymbol{D}}_i) \leq \frac{1}{4}$. We can then conclude $\tilde{\boldsymbol{D}}_i - \frac{1}{4}\mathbf{I}_K$ is negative semi-definite. Following the same arguments as before, we can get:

$$\lambda_{\max}(\tilde{H}_{\mathbf{W}}) \leq \frac{\tau^2}{4}\lambda_{\max}\big(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \boldsymbol{f}_i\boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \boldsymbol{f}_i\boldsymbol{f}_i^\top\big). \tag{45}$$

Hence, if $\pi_1 \geq 1$, then Eq. (42) provides an approximate Lipschitz constant for the block of variables in $\mathbf{W}$. □

Lemma 4. *Suppose that $\mathbf{W}$ and $\boldsymbol{q}$ are fixed. The Hessian matrix of the objective function with respect to block $\boldsymbol{\alpha} \in \mathbb{R}^K$, denoted as $H_{\boldsymbol{\alpha}} \in \mathbb{R}^{K\times K}$, is*

$$H_{\boldsymbol{\alpha}} = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top) \odot \mathbf{T}_i\big] + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} \big[(\mathrm{diag}(\boldsymbol{p}_i) - \boldsymbol{p}_i\boldsymbol{p}_i^\top) \odot \mathbf{T}_i\big], \tag{46}$$

where $\odot$ denotes the Hadamard product, and $\mathbf{T}_i \in \mathbb{R}^{K\times K}$ is a matrix whose element at $(j,l) \in [1,2,\cdots,K]^2$ is given by $(\boldsymbol{f}_i^\top\boldsymbol{t}_l)(\boldsymbol{f}_i^\top\boldsymbol{t}_j)$.

Proof. Suppose that $\mathbf{W}$ and $\boldsymbol{q}$ are fixed. The first-order derivatives of L in Eq. (7) with respect to $\boldsymbol{\alpha}$ is

$$\nabla_{\alpha_j} L = \frac{\mu\tau}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ij}-\tilde{y}_{ij})\boldsymbol{f}_i^\top \boldsymbol{t}_j + \frac{(\delta+\beta-\gamma)\tau}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ij}-q_{ij})\boldsymbol{f}_i^\top \boldsymbol{t}_j, \tag{47}$$

where p_{ij} is defined in Eq. (1). Then, the second-order derivatives of L with respect to $\boldsymbol{\alpha}$, denoted as $\nabla^2_{\alpha_j\alpha_l} L$, can be calculated in the following two situations.
(1) if $l = j$, then

$$\nabla_{\alpha_l} p_{ij} = \tau\,(p_{ij}-p_{ij}^2)\boldsymbol{f}_i^\top \boldsymbol{t}_j, \tag{48}$$

thus,

$$\nabla^2_{\alpha_j\alpha_l} L = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ij}-p_{ij}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_j)^2 + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ij}-p_{ij}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_j)^2. \tag{49}$$

(2) if $l \neq j$, then

$$\nabla_{\alpha_l} p_{ij} = -\tau\, p_{ij}p_{il}\boldsymbol{f}_i^\top \boldsymbol{t}_l, \tag{50}$$

thus,

$$\nabla^2_{\alpha_j\alpha_l} L = -\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} p_{ij}p_{il}(\boldsymbol{f}_i^\top \boldsymbol{t}_l)(\boldsymbol{f}_i^\top \boldsymbol{t}_j) - \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} p_{ij}p_{il}(\boldsymbol{f}_i^\top \boldsymbol{t}_l)(\boldsymbol{f}_i^\top \boldsymbol{t}_j). \tag{51}$$

So the $K\times K$ Hessian matrix $H_{\boldsymbol{\alpha}}$ can be summarized as

$$H_{\boldsymbol{\alpha}} = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left[(\mathrm{diag}(\boldsymbol{p}_i)-\boldsymbol{p}_i\boldsymbol{p}_i^\top)\odot \mathbf{T}_i\right] + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\left[(\mathrm{diag}(\boldsymbol{p}_i)-\boldsymbol{p}_i\boldsymbol{p}_i^\top)\odot \mathbf{T}_i\right], \tag{52}$$

where $\odot$ denotes the Hadamard product, and $\mathbf{T}_i$ is the $K\times K$ matrix whose element at $(j,l)\in[1,2,\cdots,K]^2$ is given by $(\boldsymbol{f}_i^\top \boldsymbol{t}_l)(\boldsymbol{f}_i^\top \boldsymbol{t}_j)$. $\mathbf{p}_i = (p_{ik})_{1\le k\le K}$, and $\mathrm{diag}(\mathbf{p}_i)$ represents a diagonal $K\times K$ matrix whose elements are given by p_{ik}. □

Lemma 5. *For $\pi_2 \ge 1$, the objective function in Equation (7) has the following block Lipschitz constant for the set of variables in $\boldsymbol{\alpha}$:*

$$\gamma_{\boldsymbol{\alpha}} = \frac{\pi_2\tau^2}{4}\cdot\max_k\Big(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2\Big) \tag{53}$$

Proof. We can get an approximate matrix of the Hessian matrix $H_{\boldsymbol{\alpha}}$ by omitting the off-diagonal elements in Eq. (46). The approximate Hessian matrix, denoted as $\tilde{H}_{\boldsymbol{\alpha}}$, is a $K\times K$ diagonal matrix with its diagonal elements given by

$$\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2, \quad k=1,2,\cdots,K. \tag{54}$$

Since $p_{ik}-p_{ik}^2 = -(p_{ik}-\frac{1}{2})^2+\frac{1}{4}\le\frac{1}{4}$, then we can obtain for $\pi_2\ge 1$,

$$\lambda_{\max}(\tilde{H}_{\boldsymbol{\alpha}}) \le \frac{\pi_2\tau^2}{4}\cdot\max_k\Big(\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2\Big). \tag{55}$$

So Eq. (55) gives an approximate Lipschitz constant. □

B INITIALIZATION OF THE VARIABLES

In this subsection, we describe the initialization strategy for the classifier parameters $\mathbf{W}$ and $\boldsymbol{\alpha}$ and the auxiliary variable $\boldsymbol{q}$ in the alternating optimization procedure. We further discuss the choices of ρ_1 and ρ_2 for initializing $\mathbf{W}$ and $\boldsymbol{\alpha}$.

B.1 Initialization of the Auxiliary Variable

The auxiliary variable $\boldsymbol{q} = [q_{ik}] \in \mathbb{R}^{|\mathcal{Q}|\times K}$ represents the class-assignment probabilities of the query samples. We initialize $\boldsymbol{q}$ using the zero-shot predictions of the pretrained CLIP model. Specifically, for each query sample $\boldsymbol{x}_i$, we compute

$$q_{ik}^{(0)} = \frac{\exp\left(\eta \boldsymbol{f}_i^\top \boldsymbol{t}_k\right)}{\sum_{j=1}^{K} \exp\left(\eta \boldsymbol{f}_i^\top \boldsymbol{t}_j\right)}, \tag{56}$$

where η is the CLIP temperature, $\boldsymbol{f}_i$ denotes the visual feature of the i-th query sample $\boldsymbol{x}_i$, and $\boldsymbol{t}_k$ is the text embedding of class k. Thus, the initial query assignments incorporate the semantic knowledge encoded by the pretrained VLM before transductive adaptation.

B.2 Initialization of the classifier parameters

The parameters $\mathbf{W}$ and $\boldsymbol{\alpha}$ are initialized in a data-driven manner by approximating the minima of the sub-objective function $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ in Eq. (7), based on the initial estimate of $\boldsymbol{q}$. The following proposition provides a principled approach for this initialization.

Proposition 4. *The sub-objective function $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ in Eq. (7) can be viewed as the sum of two convex functions, i.e., $L = g_1 + g_2$. For each class $k \in [1, 2, \cdots, K]$, the minimum of g_1 and g_2 w.r.t $\boldsymbol{w}_k$ can be viewed as weighted means of the feature vectors belonging to class k:*

$$\arg\min_{\boldsymbol{w}_k} g_1 \propto \frac{\mu\tau}{\rho_1} \cdot \frac{\sum_{i\in\mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i}{\sum_{i\in\mathcal{S}} \tilde{y}_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_1} \cdot \frac{\sum_{i\in\mathcal{Q}} q_{ik} \boldsymbol{f}_i}{\sum_{i\in\mathcal{Q}} q_{ik}}, \tag{57}$$

$$\arg\min_{\boldsymbol{w}_k} g_2 \propto \frac{\mu\tau}{\rho_1} \cdot \frac{\sum_{i\in\mathcal{S}} p_{ik} \boldsymbol{f}_i}{\sum_{i\in\mathcal{S}} p_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_1} \frac{\sum_{i\in\mathcal{Q}} p_{ik} \boldsymbol{f}_i}{\sum_{i\in\mathcal{Q}} p_{ik}}. \tag{58}$$

Here, ρ_1 satisfies the following inequality to ensure the convexity of g_2:

$$\rho_1 \leq \min_k \lambda_{\min}\Big(\frac{\mu\tau^2}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} (p_{ik} - p_{ik}^2) \boldsymbol{f}_i \boldsymbol{f}_i^\top + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|} \sum_{i\in\mathcal{Q}} (p_{ik} - p_{ik}^2) \boldsymbol{f}_i \boldsymbol{f}_i^\top\Big), \tag{59}$$

where $\lambda_{\min}(\cdot)$ denotes the smallest eigen value of matrix. We initialize $\boldsymbol{w}_k$ using the minimizer of g_1 as given in Eq. (57).

Similarly, $L(\mathbf{W}, \boldsymbol{\alpha}; \boldsymbol{q})$ can be viewed as the sum of two convex functions, i.e., $L = h_1 + h_2$, such that, $\forall k \in [1, 2, \cdots, K]$, the minimum of h_1 and h_2 w.r.t α_k can be viewed as the (hard and soft, or soft and soft) weighted cosine similarities between the image and text embeddings:

$$\arg\min_{\alpha_k} h_1 \propto \frac{\mu\tau}{\rho_2} \cdot \frac{\sum_{i\in\mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{S}} \tilde{y}_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_2} \cdot \frac{\sum_{i\in\mathcal{Q}} q_{ik} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{Q}} q_{ik}}, \tag{60}$$

$$\arg\min_{\alpha_k} h_2 \propto \frac{\mu\tau}{\rho_2} \cdot \frac{\sum_{i\in\mathcal{S}} p_{ik} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{S}} p_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_2} \frac{\sum_{i\in\mathcal{Q}} p_{ik} \mathbf{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{Q}} p_{ik}} \tag{61}$$

where

$$\rho_2 \leq \min_k \Big[\frac{\mu\tau^2}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} (p_{ik} - p_{ik}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|} \sum_{i\in\mathcal{Q}} (p_{ik} - p_{ik}^2)(\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2\Big]. \tag{62}$$

We can simply initialize α_k as $\arg\min_{\alpha_k} h_1$, as shown in Eq. (60).

Proof. **(1) Initialization of $\mathbf{w}_k$.**
Consider the following expression of g_1 and g_2, for some $\rho_1 > 0$:

$$\begin{aligned} g_1 =& -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\big[\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)\big] \\ &-\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\big[\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)\big]+\frac{\rho_1}{2}\sum_{k=1}^{K}\|\boldsymbol{w}_k\|^2 \\ g_2 =& \frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big) \\ &+\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big)-\frac{\rho_1}{2}\sum_{k=1}^{K}\|\boldsymbol{w}_k\|^2 \end{aligned} \tag{63}$$

Obviously, $L(\mathbf{W},\boldsymbol{\alpha};\boldsymbol{q}) = g_1 + g_2$. Take the gradients of g_1 and g_2 w.r.t $\boldsymbol{w}_k$, $k\in[1,2,\cdots,K]$, we have:

$$\begin{aligned} \frac{\partial g_1}{\boldsymbol{w}_k} &= -\frac{\mu\tau}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\boldsymbol{f}_i-\frac{(\delta+\beta-\gamma)\tau}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}\boldsymbol{f}_i+\rho_1\boldsymbol{w}_k \\ \frac{\partial g_2}{\boldsymbol{w}_k} &= \frac{\mu\tau}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}p_{ik}\boldsymbol{f}_i+\frac{(\delta+\beta-\gamma)\tau}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}p_{ik}\boldsymbol{f}_i-\rho_1\boldsymbol{w}_k \end{aligned} \tag{64}$$

For $\rho_1 > 0$, g_1 is convex w.r.t each $\boldsymbol{w}_k$, $k\in[1,2,\cdots,K]$. By setting the gradient of g_1 w.r.t $\boldsymbol{w}_k$ to 0, we can get the minimum as follows:

$$\begin{aligned} \arg\min_{\mathbf{w}_k} g_1 &= \frac{\mu\tau}{\rho_1|\mathcal{S}|}\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\boldsymbol{f}_i+\frac{(\delta+\beta-\gamma)\tau}{\rho_1|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}\boldsymbol{f}_i \\ &\propto \frac{\mu\tau}{\rho_1}\cdot\frac{\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\boldsymbol{f}_i}{\sum_{i\in\mathcal{S}}\tilde{y}_{ik}}+\frac{(\delta+\beta-\gamma)\tau}{\rho_1}\cdot\frac{\sum_{i\in\mathcal{Q}}q_{ik}\boldsymbol{f}_i}{\sum_{i\in\mathcal{Q}}q_{ik}}. \end{aligned} \tag{65}$$

As for g_2, the Hessian matrix is as follows,

$$\mathbf{H}_{g_2} = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mathbf{H}_i+\frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\mathbf{H}_i-\rho_1\mathbf{I}, \tag{66}$$

where $\mathbf{I}$ denotes the $KD\times KD$ identity matrix. $\mathbf{H}_i$ is defined as follows:

$$\mathbf{H}_i = (\mathrm{diag}(\boldsymbol{p}_i)-\boldsymbol{p}_i\boldsymbol{p}_i^\top)\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top)\approx\tilde{\boldsymbol{D}}_i\otimes(\boldsymbol{f}_i\boldsymbol{f}_i^\top), \tag{67}$$

where $\otimes$ denotes the Kronecker matrix product, and $\mathrm{diag}(\boldsymbol{p}_i)$ represents a diagonal $K\times K$ matrix whose elements are given by p_{ik}, and $\boldsymbol{p}_i = (p_{ik})_{1\le k\le K}$. $\tilde{\boldsymbol{D}}_i$ is diagonal and its elements are given by the diagonal elements of $\mathrm{diag}(\boldsymbol{p}_i)-\boldsymbol{p}_i\boldsymbol{p}_i^\top$, namely, $p_{ik}-p_{ik}^2, k=1,\cdots,K$. Let $\lambda_{\min}(\cdot)$ denotes the smallest eigen value of matrix, if we set ρ_1 as

$$\rho_1 \le \min_k \lambda_{\min}\Big(\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ik}-p_{ik}^2)\boldsymbol{f}_i\boldsymbol{f}_i^\top+\frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ik}-p_{ik}^2)\boldsymbol{f}_i\boldsymbol{f}_i^\top\Big), \tag{68}$$

then the Hessian matrix above is positive semi-definite(PSD), meaning that g_2 is convex. So setting the gradients of g_2 to 0 lead to the minimum:

$$\begin{aligned} \arg\min_{\boldsymbol{w}_k} g_2 &= \frac{\mu\tau}{\rho_1|\mathcal{S}|}\sum_{i\in\mathcal{S}}p_{ik}\boldsymbol{f}_i+\frac{(\delta+\beta-\gamma)\tau}{\rho_1|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}p_{ik}\boldsymbol{f}_i \\ &\propto \frac{\mu\tau}{\rho_1}\cdot\frac{\sum_{i\in\mathcal{S}}p_{ik}\boldsymbol{f}_i}{\sum_{i\in\mathcal{S}}p_{ik}}+\frac{(\delta+\beta-\gamma)\tau}{\rho_1}\frac{\sum_{i\in\mathcal{Q}}p_{ik}\boldsymbol{f}_i}{\sum_{i\in\mathcal{Q}}p_{ik}}, \end{aligned} \tag{69}$$

where ρ_1 is given in Eq. (68). Note that, since $p_{ik} = \mathrm{softmax}\big(\tau\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)\big)$ depends on $\boldsymbol{w}$ itself, this is not a closed form, and the posterior p is not available at initialization time. Hence we initialize $\boldsymbol{w}_k$ with the weighted-average minimizer of g_1 (Eq. (65)) only.

(2) Initialization of α_k.
Similarly, consider the following expression of h_1 and h_2, for some $\rho_2 > 0$:

$$\begin{aligned} h_1 =& -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\sum_{k=1}^{K}\tilde{y}_{ik}\big[\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)\big] \\ &-\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\sum_{k=1}^{K}q_{ik}\big[\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k)\big]+\frac{\rho_2}{2}\sum_{k=1}^{K}\alpha_k^2 \\ h_2 =& \frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big) \\ &+\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\log\Big(\sum_{k=1}^{K}\exp(\tau\,\boldsymbol{f}_i^\top(\boldsymbol{w}_k+\alpha_k\boldsymbol{t}_k))\Big)-\frac{\rho_2}{2}\sum_{k=1}^{K}\alpha_k^2 \end{aligned} \tag{70}$$

Obviously, $L(\mathbf{W},\boldsymbol{\alpha};\boldsymbol{q}) = h_1 + h_2$. Take the gradients of h_1 and h_2 w.r.t α_k, $k \in [1, 2, \cdots, K]$, we have:

$$\begin{aligned} \frac{\partial h_1}{\alpha_k} &= -\frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\tau\,\boldsymbol{f}_i^\top\boldsymbol{t}_k-\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}\tau\,\boldsymbol{f}_i^\top\boldsymbol{t}_k+\rho_2\alpha_k \\ \frac{\partial h_2}{\alpha_k} &= \frac{\mu}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}p_{ik}\tau\,\boldsymbol{f}_i^\top\boldsymbol{t}_k+\frac{\delta+\beta-\gamma}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}p_{ik}\tau\,\boldsymbol{f}_i^\top\boldsymbol{t}_k-\rho_2\alpha_k \end{aligned} \tag{71}$$

For $\rho_2 > 0$, h_1 is convex w.r.t each α_k, $k \in [1, 2, \cdots, K]$. By setting the gradient of h_1 w.r.t α_k to 0, we can get the minimum as follows:

$$\begin{aligned} \arg\min_{\alpha_k} h_1 &= \frac{\mu\tau}{\rho_2|\mathcal{S}|}\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\boldsymbol{f}_i^\top\mathbf{t}_k+\frac{(\delta+\beta-\gamma)\tau}{\rho_2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}q_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k \\ &\propto \frac{\mu\tau}{\rho_2}\cdot\frac{\sum_{i\in\mathcal{S}}\tilde{y}_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k}{\sum_{i\in\mathcal{S}}\tilde{y}_{ik}}+\frac{(\delta+\beta-\gamma)\tau}{\rho_2}\cdot\frac{\sum_{i\in\mathcal{Q}}q_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k}{\sum_{i\in\mathcal{Q}}q_{ik}}. \end{aligned} \tag{72}$$

As for h_2, the Hessian matrix is as follows,

$$\mathbf{H}_{h_2} = \frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mathbf{H}_i^{'}+\frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}\mathbf{H}_i^{'}-\rho_2\mathbf{I}, \tag{73}$$

where $\mathbf{I}$ denotes the identity matrix, and $\mathbf{H}_i^{'}$ is defined as follows:

$$\mathbf{H}_i^{'} = (\mathrm{diag}(\boldsymbol{p}_i)-\boldsymbol{p}_i\boldsymbol{p}_i^\top)\odot\mathbf{T}_i, \tag{74}$$

where $\odot$ denotes the Hadamard product, and $\mathbf{T}_i$ is the $K\times K$ matrix whose element at $(j,l)\in[1,2,\cdots,K]^2$ is given by $(\boldsymbol{f}_i^\top\boldsymbol{t}_l)(\boldsymbol{f}_i^\top\boldsymbol{t}_j)$. We can get an approximate of the Hessian matrix $\mathbf{H}_{h_2}$ by omitting the off-diagonal elements. Then the approximate Hessian matrix is a $K\times K$ diagonal matrix with its diagonal elements given by

$$\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top\boldsymbol{t}_k)^2+\frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top\boldsymbol{t}_k)^2-\rho_2,\quad k=1,2,\cdots,K. \tag{75}$$

Thus, if we set ρ_2 as

$$\rho_2 \le \min_k\Big[\frac{\mu\tau^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top\boldsymbol{t}_k)^2+\frac{(\delta+\beta-\gamma)\tau^2}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}(p_{ik}-p_{ik}^2)(\boldsymbol{f}_i^\top\boldsymbol{t}_k)^2\Big], \tag{76}$$

then the Hessian matrix above is positive semi-definite(PSD), meaning that h_2 is convex. So setting the gradients of h_2 to 0 lead to the minimum:

$$\begin{aligned} \arg\min_{\alpha_k} h_2 &= \frac{\mu\tau}{\rho_2|\mathcal{S}|}\sum_{i\in\mathcal{S}}p_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k+\frac{(\delta+\beta-\gamma)\tau}{\rho_2|\mathcal{Q}|}\sum_{i\in\mathcal{Q}}p_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k \\ &\propto \frac{\mu\tau}{\rho_2}\cdot\frac{\sum_{i\in\mathcal{S}}p_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k}{\sum_{i\in\mathcal{S}}p_{ik}}+\frac{(\delta+\beta-\gamma)\tau}{\rho_2}\frac{\sum_{i\in\mathcal{Q}}p_{ik}\boldsymbol{f}_i^\top\boldsymbol{t}_k}{\sum_{i\in\mathcal{Q}}p_{ik}}, \end{aligned} \tag{77}$$

where ρ_2 satisfies Eq. (76). Similarly, the minimization of h_2 depends on p_{ik}, thus we initialize α_k with the weighted-average minimizer of $h1$ (Eq. (72)) only.

□

Choice of ρ_1 and ρ_2. Since the minimizers of g_2 and h_2 involve the posterior p_{ik}, which is unavailable at initialization, we initialize $\mathbf{W}$ and $\boldsymbol{\alpha}$ with the closed-form minimizers of g_1 and h_1 given in Eqs.(57) and(60). These initializers depend on the auxiliary parameters ρ_1 and ρ_2 only through the overall scaling factors $1/\rho_1$ and $1/\rho_2$. These factors scale the magnitude of $\boldsymbol{w}_k^{(0)}$ and $\alpha_k^{(0)}$ but leave their directions (determined by the support labels and the initial query assignments $\boldsymbol{q}$) unchanged. Since we subsequently rescale each class prototype $\boldsymbol{w}_k^{(0)}$ to L_2-norm 1.0 and $\boldsymbol{\alpha}^{(0)}$ to $\max_k |\alpha_k^{(0)}| = 1.0$, these global scale factors are exactly cancelled, and the resulting initialization is identical for any fixed choice of ρ_1, ρ_2. Therefore, without loss of generality, we set $\rho_1 = \rho_2 = 1$, avoiding the computationally expensive, posterior-dependent eigenvalue bounds of Proposition 4. This normalization also keeps the initial logits within a reasonable range, preventing excessively confident near-one-hot predictions in the early optimization stages.

C Additional Experiments

C.1 Hyperparameter tuning strategy

In all experiments, we first perform hyperparameter tuning using TransLP++ with Lipschitz-guided GD. For efficient hyperparameter search, we set $T = 200$ outer iterations, with $\text{iter}_{\mathbf{W}} = 20$ inner iterations for updating the classifier parameters $\mathbf{W}$ and $\text{iter}_{\boldsymbol{\alpha}} = 1$ inner iteration for updating $\boldsymbol{\alpha}$. We set $\pi_1 = 2$ and $\pi_2 = 4$ for the corresponding block Lipschitz constants.

TransLP++ involves several hyperparameters, including the support supervision weight μ, the query-set mutual information weight β, the CLIP-guided regularization weight γ, the auxiliary-variable regularization coefficient δ, the label-smoothing factor ϵ, the temperature η for CLIP zero-shot predictions, and the temperature τ used in the adapted classifier. We search over $\mu \in \{1.0, 2.0, 4.0\}$, $\beta \in \{0.5, 1.0\}$, $\gamma \in \{0.1, 0.25, 0.5\}$, $\delta \in \{1.0, 2.0\}$, $\epsilon \in \{0.05, 0.1, 0.15, 0.2\}$, $\eta \in \{5, 10, 30, 60, 100\}$, and $\tau \in \{5, 15, 30, 60, 100\}$. Only configurations satisfying $\delta + \beta - \gamma > 0$ are considered, as required by the stability condition in our theoretical analysis. We select the configuration with the best average validation accuracy across the 11 datasets under the 4-shot setting with random seed 1, and use the selected configuration consistently across all reported experiments.

The selected hyperparameters are $\mu = 1.0$, $\beta = 0.5$, $\gamma = 0.25$, $\delta = 1.0$, $\epsilon = 0.1$, $\eta = 60$, and $\tau = 30$. Under this configuration, $\delta + \beta - \gamma = 1.25 > 0$, satisfying the required stability condition. The same hyperparameter configuration is used for both Lipschitz-guided GD and momentum-based optimization. For the final experiments, we increase the optimization budget to improve convergence. Specifically, Lipschitz-guided GD uses $T = 2000$ outer iterations with $\text{iter}_{\mathbf{W}} = 40$ and $\text{iter}_{\boldsymbol{\alpha}} = 1$, whereas momentum-based optimization uses $T = 1000$ outer iterations with $\text{iter}_{\mathbf{W}} = 20$ and $\text{iter}_{\boldsymbol{\alpha}} = 1$. All other experimental settings are kept identical.

C.2 Comparison of Lipschitz-guided GD and Momentum

We compare the Lipschitz-guided GD and momentum-based optimization strategies for TransLP++. All other components and hyperparameters are kept identical unless otherwise specified, including the initialization, objective function, and alternating update scheme. For Lipschitz-guided GD, the learning rates are determined by the Lipschitz constants derived in Proposition 3. For momentum optimization, a fixed learning rate of 10^{-3} is used for both $\mathbf{W}$ and $\boldsymbol{\alpha}$. After each update of $\mathbf{W}$, each classifier weight $\boldsymbol{w}_k$ is normalized to unit ℓ_2 norm. Results are averaged over 11 datasets and three random seeds.

As shown in Table 5 and Table 6, the two optimization strategies achieve very similar classification performance across all few-shot settings and backbones. Notably, the momentum-based variant requires substantially fewer optimization updates. Specifically, it employs 1,000 outer iterations with $\text{iter}_{\mathbf{W}} = 20$, compared with the 2,000 outer iterations and $\text{iter}_{\mathbf{W}} = 40$ required by Lipschitz-guided GD. This reduction in optimization updates leads to a substantial decrease in inference time.

Algorithm 1 TransLP++ with Lipschitz-guided GD or momentum-based optimization.

Input: Image embeddings $\{\boldsymbol{f_i}\}$, text embeddings $\{\boldsymbol{t}_k\}$, and optimizer $\texttt{opt} \in \{\texttt{GD}, \texttt{Momentum}\}$.

function TRANSLP++($\boldsymbol{f}, \boldsymbol{t}, \texttt{opt}$)

// Calculate $\{\hat{y}_{ik}\}$, Initialize $\boldsymbol{q} \in \mathbb{R}^{|\mathcal{Q}|\times K}$, $\mathbf{W} \in \mathbb{R}^{d\times K}$, and $\boldsymbol{\alpha} \in \mathbb{R}^K$.

$\hat{y}_{ik} = \text{softmax}(\eta \boldsymbol{f}_i^\top \boldsymbol{t}_k), \quad \forall i \in \mathcal{Q},\ k \in [1, 2, \cdots, K].$

$q_{ik}^{(0)} = \hat{y}_{ik}, \quad \forall i \in \mathcal{Q},\ k \in [1, 2, \cdots, K],$

$\mathbf{w}_k^{(0)} = \frac{\mu\tau}{\rho_1} \cdot \frac{\sum_{i\in\mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i}{\sum_{i\in\mathcal{S}} \tilde{y}_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_1} \cdot \frac{\sum_{i\in\mathcal{Q}} q_{ik}^{(0)} \boldsymbol{f}_i}{\sum_{i\in\mathcal{Q}} q_{ik}^{(0)}}, \quad \forall k \in [1, 2, \cdots, K],$ // Eq. (57)

$\alpha_k^{(0)} = \frac{\mu\tau}{\rho_2} \cdot \frac{\sum_{i\in\mathcal{S}} \tilde{y}_{ik} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{S}} \tilde{y}_{ik}} + \frac{(\delta+\beta-\gamma)\tau}{\rho_2} \cdot \frac{\sum_{i\in\mathcal{Q}} q_{ik}^{(0)} \boldsymbol{f}_i^\top \boldsymbol{t}_k}{\sum_{i\in\mathcal{Q}} q_{ik}^{(0)}}, \quad \forall k \in [1, 2, \cdots, K],$ // Eq. (60)

Normalize each $\boldsymbol{w}_k^{(0)}$ to unit ℓ_2 norm and rescale $\boldsymbol{\alpha}^{(0)}$ such that $\max_k |\alpha_k^{(0)}| = 1$.

// Initialization other parameters.

if $\texttt{opt} = \texttt{GD}$ **then**

$\gamma_{\mathbf{W}} = \frac{\pi_1\tau^2}{4} \lambda_{\max}\big(\frac{\mu}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} \boldsymbol{f}_i \boldsymbol{f}_i^\top + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|} \sum_{i\in\mathcal{Q}} \boldsymbol{f}_i \boldsymbol{f}_i^\top\big),$ // Eq. (10)

$\gamma_{\boldsymbol{\alpha}} = \frac{\pi_2\tau^2}{4} \cdot \max_k \Big(\frac{\mu}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} (\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2 + \frac{\delta+\beta-\gamma}{|\mathcal{Q}|} \sum_{i\in\mathcal{Q}} (\boldsymbol{f}_i^\top \boldsymbol{t}_k)^2\Big),$ // Eq. (11)

$\text{lr}_{\mathbf{W}} = \frac{1}{\gamma_{\mathbf{W}}}, \ \text{lr}_{\boldsymbol{\alpha}} = \frac{1}{\gamma_{\boldsymbol{\alpha}}}$

else

$\mathbf{v}_{\mathbf{W}} = \mathbf{0}, \quad \mathbf{v}_{\boldsymbol{\alpha}} = \mathbf{0},$ // initialize momentum buffers

$\nu = 0.9$ // momentum parameter

$\text{lr}_{\mathbf{W}} = 1e-3, \ \text{lr}_{\boldsymbol{\alpha}} = 1e-3$ // learning rate

end if

// Iterative procedure.

for $t = 0, 1, \cdots, T-1$ **do**

// 1). Update the classifier weights $\mathbf{W}$.

for $l_1 = 0, 1, \cdots, \text{iter}_{\mathbf{W}} - 1$ **do**

if $\texttt{opt} = \texttt{GD}$ **then**

$\mathbf{W}^{(t,l_1+1)} = \mathbf{W}^{(t,l_1)} - \text{lr}_{\mathbf{W}} \nabla L_{\mathbf{W}}(\mathbf{W}^{(t,l_1)}, \boldsymbol{\alpha}^{(t,0)})$ // Use Eq. (28) for $\nabla L_{\mathbf{W}}$.

else

$\mathbf{v}_{\mathbf{W}} \leftarrow \nu \mathbf{v}_{\mathbf{W}} - \text{lr}_{\mathbf{W}} \nabla L_{\mathbf{W}}(\mathbf{W}^{(t,l_1)}, \boldsymbol{\alpha}^{(t,0)})$

$\mathbf{W}^{(t,l_1+1)} \leftarrow \mathbf{W}^{(t,l_1)} + \mathbf{v}_{\mathbf{W}}$

Normalize each $\boldsymbol{w}_k$ to unit ℓ_2 norm.

end if

end for

$\mathbf{W}^{(t+1,0)} = \mathbf{W}^{(t,\ \text{iter}_{\mathbf{W}})}$

// 2). Update the mixing coefficient $\boldsymbol{\alpha}$.

for $l_2 = 0, 1, \cdots, \text{iter}_{\boldsymbol{\alpha}} - 1$ **do**

if $\texttt{opt} = \texttt{GD}$ **then**

$\boldsymbol{\alpha}^{(t,l_2+1)} = \boldsymbol{\alpha}^{(t,l_2)} - \text{lr}_{\boldsymbol{\alpha}} \nabla L_{\boldsymbol{\alpha}}(\mathbf{W}^{(t+1,0)}, \boldsymbol{\alpha}^{(t,l_2)})$ // Use Eq. (47) for $\nabla L_{\boldsymbol{\alpha}}$.

else

$\mathbf{v}_{\boldsymbol{\alpha}} \leftarrow \nu \mathbf{v}_{\boldsymbol{\alpha}} - \text{lr}_{\boldsymbol{\alpha}} \nabla L_{\boldsymbol{\alpha}}(\mathbf{W}^{(t+1,0)}, \boldsymbol{\alpha}^{(t,l_2)})$

$\boldsymbol{\alpha}^{(t,l_2+1)} \leftarrow \boldsymbol{\alpha}^{(t,l_2)} + \mathbf{v}_{\boldsymbol{\alpha}}$

end if

end for

$\boldsymbol{\alpha}^{(t+1,0)} = \boldsymbol{\alpha}^{(t,\ \text{iter}_{\boldsymbol{\alpha}})}$

// 3). Update the auxiliary variable $\boldsymbol{q}$.

$q_{ik}^{(t+1)} \propto \frac{\big(p_{ik}^{(t+1)}\big)^{\frac{\delta+\beta-\gamma}{\delta}} \cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}}{\Big[\sum_{i\in\mathcal{Q}} \Big(\big(p_{ik}^{(t+1)}\big)^{\frac{\delta+\beta-\gamma}{\delta}} \cdot \hat{y}_{ik}^{\frac{\gamma}{\delta}}\Big)\Big]^{\frac{1}{\delta+1}}}$

end for

return $\mathbf{W}$, $\boldsymbol{\alpha}$, and $\boldsymbol{q}$.

end function

Table 6: **Comparison of optimization strategies for TransLP++.** Average accuracy and standard deviation are reported over 11 datasets and three random seeds across different backbone.

Method	1-shot	2-shot	4-shot	8-shot	16-shot
CLIP-ResNet-50			58.68		
TransLP++(Lipschitz-guided GD)	67.46 ± 0.53	70.08 ± 0.81	71.80 ± 0.93	73.71 ± 0.38	75.92 ± 0.47
TransLP++(Momentum)	67.69 ± 0.47	70.18 ± 0.81	72.00 ± 0.77	73.83 ± 0.43	76.00 ± 0.47
CLIP-ResNet-101			59.51		
TransLP++(Lipschitz-guided GD)	70.35 ± 0.59	72.26 ± 0.95	74.00 ± 0.57	76.26 ± 0.35	78.19 ± 0.39
TransLP++(Momentum)	70.51 ± 0.74	72.46 ± 0.85	74.10 ± 0.48	76.21 ± 0.43	78.09 ± 0.45
CLIP-ViT-B/32			61.88		
TransLP++(Lipschitz-guided GD)	70.68 ± 1.10	72.69 ± 0.58	74.35 ± 0.61	76.21 ± 0.34	78.17 ± 0.44
TransLP++(Momentum)	70.98 ± 1.00	72.77 ± 0.62	74.46 ± 0.48	76.23 ± 0.32	78.13 ± 0.39
CLIP-ViT-B/16			65.28		
TransLP++(Lipschitz-guided GD)	75.91 ± 0.74	77.13 ± 0.63	78.83 ± 0.59	80.35 ± 0.52	82.03 ± 0.33
TransLP++(Momentum)	75.88 ± 0.59	77.25 ± 0.53	78.73 ± 0.55	80.36 ± 0.44	82.01 ± 0.35
CLIP-ViT-L/14			72.58		
TransLP++(Lipschitz-guided GD)	81.73 ± 0.84	82.88 ± 0.58	84.13 ± 0.49	85.07 ± 0.41	86.48 ± 0.33
TransLP++(Momentum)	81.76 ± 0.82	82.89 ± 0.42	84.10 ± 0.46	85.10 ± 0.44	86.41 ± 0.29

Table 7: **Sensitivity analysis of key hyperparameters in TransLP++.** Each entry reports the accuracy (%) corresponding to the five parameter values listed in the header, in the same order. Results are averaged over 11 datasets and three random seeds.

Shots	$\mu:\ 0.1/0.5/\mathbf{1}/1.5/2.0$	$\beta:\ 0.1/0.25/\mathbf{0.5}/1/2$	$\gamma:\ 0.05/0.1/0.2/\mathbf{0.25}/0.5$
1-shot	74.70/75.71/**75.88**/75.81/75.82	71.61/73.95/**75.88**/75.54/74.19	74.34/74.82/75.79/**75.88**/73.50
4-shot	76.57/78.41/78.73/78.86/**78.86**	75.44/77.16/78.73/**78.76**/77.66	77.94/78.31/**78.80**/78.73/76.51
16-shot	77.65/81.50/82.01/82.13/**82.17**	80.74/81.09/**82.01**/81.70/80.08	81.62/81.82/**82.02**/82.01/80.70
Shots	$\delta:\ 0.1/0.5/\mathbf{1.0}/1.5/2.0$	$\tau:\ 5/15/\mathbf{30}/60/100$	$\eta:\ 5/15/30/\mathbf{60}/100$
1-shot	74.37/**76.13**/75.88/75.67/75.54	60.68/73.64/**75.88**/74.92/73.05	72.25/73.69/75.03/**75.88**/75.80
4-shot	77.74/**79.04**/78.73/78.49/78.40	72.64/77.37/**78.73**/78.41/77.14	76.77/77.68/78.38/**78.73**/78.41
16-shot	81.43/**82.17**/82.01/81.81/81.74	77.07/80.84/**82.01**/81.59/80.69	81.17/81.48/81.92/**82.01**/81.64

For example, under the 16-shot setting on `ImageNet` with the ViT-B/16 backbone, where $d = 512$, $K = 1000$, and $|\mathcal{Q}| = 50{,}000$, replacing Lipschitz-guided GD with momentum reduces the adaptation time from approximately 480 s to 118 s while maintaining comparable classification performance. These results show that the efficiency gain of momentum primarily comes from faster convergence, enabling TransLP++ to achieve comparable performance at a much higher inference speed.".

C.3 Hyperparameter Sensitivity

To assess the sensitivity of TransLP++ to its hyperparameters, we evaluate six key parameters, including the support supervision weight μ, the query-set mutual information weight β, the CLIP-guided regularization weight γ, the auxiliary-variable regularization coefficient δ, and the temperature parameters τ and η. We conduct experiments on all 11 public datasets under the 1-, 4-, and 16-shot settings. Table 7 reports the corresponding average accuracies over 11 datasets and three random seeds. Overall, TransLP++ exhibits stable performance across a broad range of hyperparameter values. For the loss weights, $\mu = 1$, $\beta = 0.5$, and $\gamma = 0.25$ provide consistently strong performance across different shot settings. Although slightly larger μ values yield marginally higher accuracy in the 4- and 16-shot settings, the differences from $\mu = 1$ remain small. Similarly, the performance varies smoothly around the selected values of β and γ, indicating that the method is not overly sensitive to the relative weighting of the support supervision, query-set mutual information, and CLIP-guided regularization. For the optimization-related parameters, $\delta = 0.5$ performs best or near-best across all shot settings, while $\tau = 30$ and $\eta = 60$ provide consistently strong results. In particular, overly small τ substantially degrades performance in the 1-shot setting, whereas increasing τ beyond 30 does not provide further gains. The performance under different η values is comparatively stable, with $\eta = 60$ achieving the best accuracy in all three shot settings. These results suggest that TransLP++ maintains robust performance over a reasonably wide range of hyperparameter choices, with the default settings providing a good balance across different few-shot regimes.

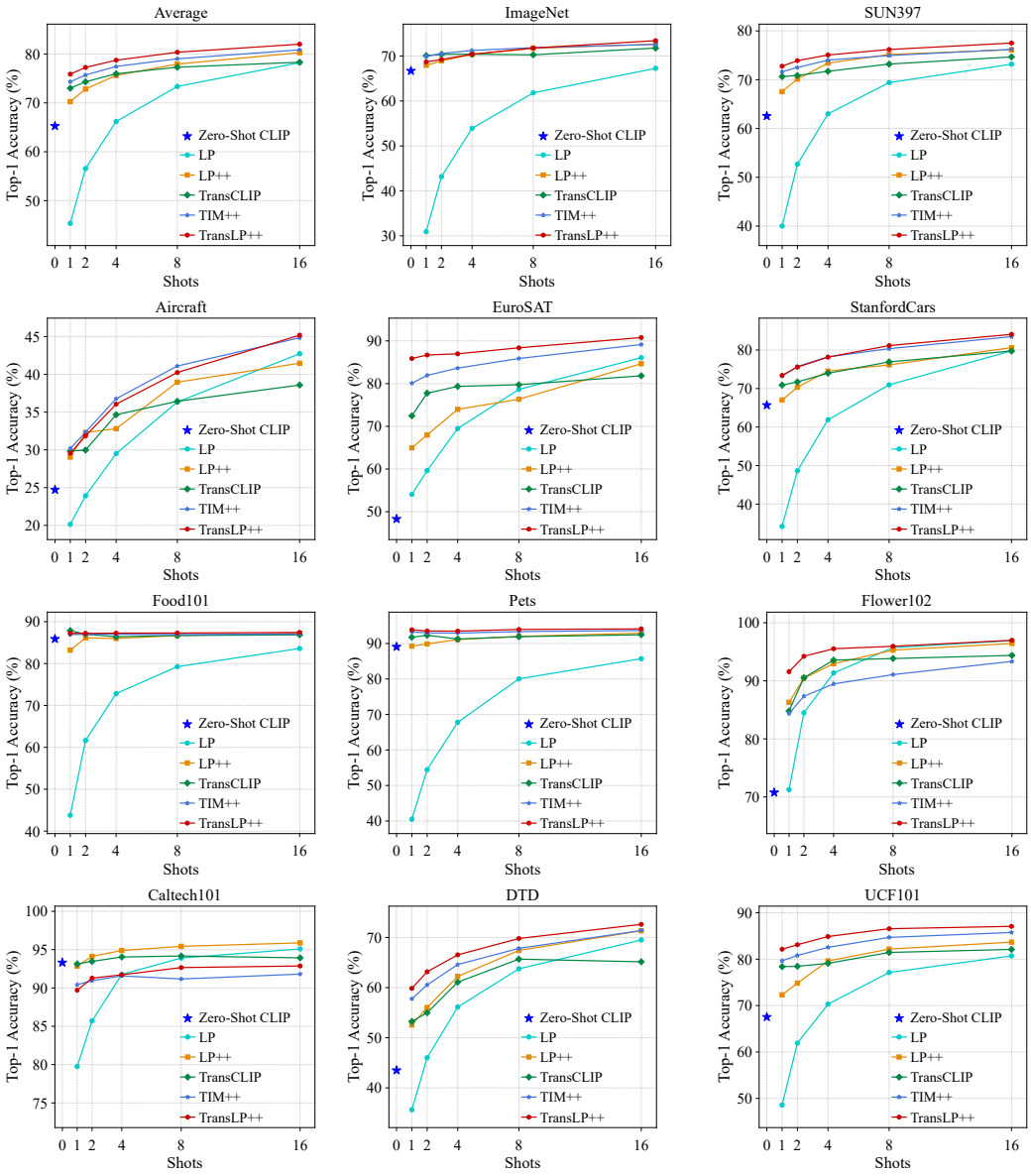


Figure 3: **Top-1 Accuracy on 11 public datasets across different shot settings.** Results on ViT-B/16 are reported under 1-, 2-, 4-, 8-, and 16-shot settings and averaged over three random seeds.