arXiv:2306.02240v2 [cs.CV] 28 Mar 2024

ProTeCt: Prompt Tuning for Taxonomic Open Set Classification

Tz-Ying Wu* Chih-Hui Ho* Nuno Vasconcelos
University of California, San Diego
{tzw001, chh279, nvasconcelos}@ucsd.edu

Abstract

Visual-language foundation models, like CLIP, learn generalized representations that enable zero-shot open-set classification. Few-shot adaptation methods, based on prompt tuning, have been shown to further improve performance on downstream datasets. However, these methods do not fare well in the taxonomic open set *(TOS) setting, where the classifier is asked to make prediction from label set across different levels of semantic granularity. Frequently, they infer incorrect labels at coarser taxonomic class levels, even when the inference at the leaf level (original class labels) is correct. To address this problem, we propose a prompt tuning technique that calibrates the hierarchical consistency of model predictions. A set of metrics of hierarchical consistency, the Hierarchical Consistent Accuracy (HCA) and the Mean Treecut Accuracy (MTA), are first proposed to evaluate TOS model performance. A new* Prompt Tuning for Hierarchical Consistency *(ProTeCt) technique is then proposed to calibrate classification across label set granularities. Results show that ProTeCt can be combined with existing prompt tuning methods to significantly improve TOS classification without degrading the leaf level classification performance. The code is available at* https://github.com/gina9726/ProTeCt.

1. Introduction

Vision-language foundation models (FMs) have opened up new possibilities for image classification. They are large models, trained on large corpora, to learn aligned representations of images and text. For example, CLIP [32] combines text and image encoders trained with 400M image-text pairs in an open vocabulary fashion, using a contrastive loss [3, 4, 36, 37]. Zero-shot classification can then proceed by leveraging the feature alignments. Each class name is first converted to a text prompt, e.g., "a photo of [CLASS]," which is fed to the text encoder. The resulting text feature is then used as the parameter vector of a softmax classifier of image feature vectors. Since the training does not emphasize any particular classes, CLIP supports open set classification. Several works [17, 44, 47, 48] have shown that classification accuracy can be enhanced by fine-tuning the FM on the few-shot setting (i.e. few examples per class). To adapt the model and maintain image-text alignment, these works augment the FM with a few learnable prompts [17, 44, 47, 48]. The model parameters are then frozen and only the prompts are optimized. This process is known as **prompt tuning** and can outperform zero-shot performance, on the dataset of interest.

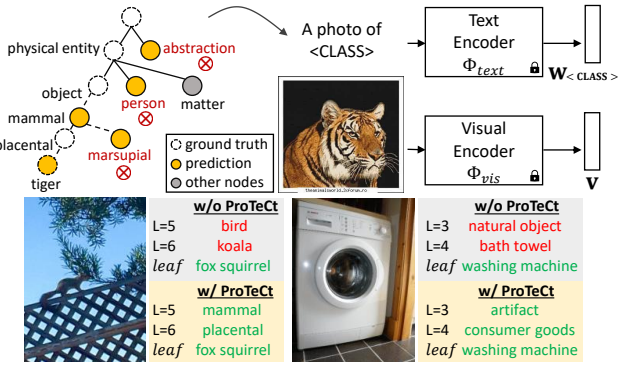


Figure 1. (Top) An example of class hierarchy, where CLIP predicts the tiger image as "person" at the internal hierarchy level. (Bottom) Correct/incorrect model predictions (green/red) of CoOp w/ and w/o ProTeCt on ImageNet variants. L denotes the tree level.

Method	Acc_{leaf}	HCA	MTA
CLIP [32]	68.36	3.32	48.21
CoOp [48]	71.23	2.99	46.98
MaPLe [17]	70.70	4.15	48.29

Table 1. TOS classification performance of CLIP-based classifiers.

While prompting enables classifiers to be designed for virtually any classes with minimal dataset curation effort, it should not compromise the open set nature and generality of the FM representation. In this work, we consider the setting where "open set" means the ability to refer to concepts at different levels of granularity. Consider, for example, an educational application in biology. While at grade school level it will teach students to classify animals into ("cat", "dog", "lizard"), at the high-school level the **exact same images** should be classified into much more detailed classes, e.g. ("iguana", "anole", "komodo", etc.) for lizards. A classifier that classifies an image as a "komodo" lizard for high schoolers but "dog" for gradeschoolers is not useful and trustworthy. Advanced biology students should

even learn about the taxonomic relations between different species. This requires a representation that supports hierarchical classification [20, 30, 41, 43], where the classifier understands the relations between the superclasses and subclasses that compose a class hierarchy, and provides correct predictions *across* hierarchy levels.

Fig. 1 shows an example hierarchy built from ImageNet [5] classes, according to the WordNet [11]. When faced with a tiger image, the classifier should provide a correct prediction under the label sets $\mathcal{Y}_1$ =("dog", "cat", "**tiger**"), $\mathcal{Y}_2$ = ("person", "**animal**", "insect") or $\mathcal{Y}_3$ =("**physical entity**", "abstraction"), where the correct one is shown in bold. Note that, given a classifier with this property, teachers have the ability to define different classification problems, for many levels of granularity, tailoring the same app to different uses. We refer to this setting as **taxonomic open set** (TOS) classification. In many real-world applications, support for this restricted form of open set classification is much more important than support for unbounded open set classification. In the example above, biology teachers do not really care if the classifier can still discriminate between cars and trucks, or soda cans and wine cups. Hence, these classes are irrelevant to the app developer.

In principle, TOS should be trivially supported by FMs. Even at zero-shot level, it should suffice to specify [CLASS] names at the desired levels of granularity. However, our experiments show that this does not work because the representation of most FMs fails to capture taxonmic relations. This is illustrated for CLIP in Fig. 1. While the model knows that the object is a tiger, it fails to know that it is "a physical entity" and not an "abstraction" or that it is a "placental mammal" and not a "marsupial," indicating that it only understands class relations locally. It can perform well for the leaf class label set $\mathcal{Y}_1$, but cannot reason across abstraction levels, and can thus not support TOS classification. To enable TOS, we introduce the notion of **hierarchical consistency**, and a new *hierarchical consistency accuracy* (HCA) metric, where classification is defined with respect to a taxonomic tree and its success requires the correct prediction of all superclasses (e.g., mammal, object and physical entity) of each ground truth leave class (e.g., tiger). This is complemented by the notion of **TOS classification**, where classifiers can have any set of nodes in the class hierarchy as the label set, and a new *mean treecut accuracy* (MTA) metric, which estimates classification accuracy in this setting.

Our experiments show that neither CLIP nor existing prompt tuning methods [17, 47, 48] perform well under the HCA and MTA metrics of the TOS setting. Fig. 1 illustrates the problem and the *inconsistent* CLIP class predictions (orange dots) across hierarchy levels. Table 1 compares the standard (leaf) accuracy of the model with HCA/MTA, under both the zero-shot and two prompt-tuning settings. While the leaf accuracy is quite reasonable, hierarchical consistency is very poor. To address this problem, we propose a novel prompt-tuning procedure, denoted *Prompt Tuning for Hierarchical Consistency* (ProTeCt), that explicitly targets the TOS setting. Given a dataset of interest, a class hierarchy is extracted from the associated metadata, a generic public taxonomy (e.g. WordNet [11]), or a special purpose taxonomy related to the application (e.g. scientific taxonomies). Since FMs support classification with open vocabulary, any node in the hierarchy can be used in the label set of the classifier. Prompts are then learned with the help of two new regularization losses that encourage hierarchical consistency. A *dynamic treecut loss* (DTL) encourages correct classification at all tree levels by sampling random tree cuts during training. A *node-centric loss* (NCL) contributes additional supervision to each internal tree node to increase classification robustness for all granularities of the hierarchy.

Experiments show that ProTeCt significantly improves the performance of prompt tuning methods, like CoOp [48] and MaPLe [17], under TOS setting. Fig. 1 shows the predictions of CoOp at different hierarchy levels before/after adding ProTeCt. Under the HCA/MTA metrics, the improvement can be more than 15/25 points on Cifar100, SUN and ImageNet datasets. Following [17, 47, 48], we show that these gains hold for zero-shot domain generalization to several variants of ImageNet [14, 15, 33, 38], showing that hierarchical consistency transfers across datasets. Furthermore, ablations show that ProTeCt can be used with different CLIP architectures, parameter tuning methods and taxonomies.

Overall, this work makes four contributions. First, we introduce the TOS setting, including two novel metrics (HCA and MTA) that evaluate the consistency of hierarchical classification. Second, we show that neither zero-shot CLIP nor existing prompting methods fare well in this setting. Third, we propose a novel prompt-tuning method for the TOS setting, ProTeCt, which improves hierarchical consistency by combining DTL and NCL losses. The former relies on a dynamic stochastic sampling of label sets involving multiple levels of the hierarchy, while the latter regularizes the classification of every node in the hierarchy. Finally, ProTeCt is shown to outperform vanilla prompt tuning methods on three datasets with different hierarchies. Extensive ablations demonstrate that ProTeCt is applicable to different parameter tuning methods, CLIP architectures, taxonomies and the learned hierarchical consistency transfers to unseen datasets from different image domains.

2. Related Work

Prompt Tuning of Vision-Language Models. Many large vision-language FMs have been proposed recently [10, 39, 45]. Despite their promising zero-shot performance, several works [16, 17, 47, 48] have shown that their few-shot finetuning with a dataset from the target application can further improve performance. Unlike conventional finetuning methods that optimize the entire model, these methods are

designed to (a) be parameter efficient and (b) maintain the general purpose feature representation of the FM. Several such tuning methods have been proposed for CLIP [32]. Inspired by prompt tuning techniques from the language literature [21, 23, 24], CoOp [48] inserts learnable prompts at the CLIP text input. CoCoOp [47] further learns a meta-network to generate an image-conditioned prompt. The idea of connecting image and text prompts is further extended by UPT [44] and MaPLe [17]. The former learns a unified transformer for generating an image and text prompt, the latter learns a coupling function to generate image prompts from text prompts. LASP [2] proposed a text-to-text cross-entropy loss to regularize the distribution shift when different prompts are used. Unlike these works, we investigate the TOS problem, where labels can be drawn from any level in a class taxonomy, and propose prompting techniques to improve hierarchical classification consistency. This is shown to be compatible with several of the above prompt-tuning methods without degrading their leaf classification accuracy.

Hierachical Classifiers. Hierarchical classification aims to predict labels at different levels of a class hierarchy. Early works [6, 7, 30, 34, 35, 46] date back to the era before deep learning and are not directly applicable to deep learning-based models. Several works [1, 13, 18, 25, 43, 49] propose hierarchical classifiers for CNN-based deep models. For example, [13, 25, 49] use additional convolutional modules to learn a hierarchical feature space. It is unclear how these approaches generalize to the recent transformer-based architectures [8, 26, 27]. Furthermore, prior works [1, 13, 25, 41, 43, 49] finetune the entire model, which requires substantial data and computation, especially at the FM scale. In this work, we study the problem of hierarchical consistency for foundational vision-language models (e.g., CLIP). While CLIP-based classifiers [32, 47, 48] have outstanding zero/few-shot performance, we show that they produce inconsistent predictions for label sets of different granularity and cannot be used in the TOS setting. We propose an efficient prompt tuning method to address this.

3. Preliminaries

Foundation Models (FMs). Visual-language FMs are composed by a text Φ_{text} and a visual Φ_{vis} encoder, which extract features from text and images, respectively. The two encoders are optimized by contrastive training [3, 4, 36, 37] to create a joint representation for the two modalities. Since the encoders are learned from a large-scale corpus of image-text pairs, the features are general and support various downstream tasks, e.g., image classification [17, 44, 47, 48] and segmentation [28, 40]. While in this work we use the CLIP [32], ProTeCt should generalize to other FMs.

Image Classification with FMs. Given a label set $\mathcal{Y} = \{t_y\}_{y=1}^{C}$, a zero-shot classifier can be designed in the FM representation space by introducing a weight vector $\mathbf{w}_y$ per class y. These weight vectors are obtained by simply using the class name t_y (e.g., "dog") as a text encoder prompt, i.e., $\mathbf{w}_y = \Phi_{text}(Emb_t(t_y)) \in \mathbb{R}^k$, where $Emb_t(\cdot)$ is a word embedding. Given these weight vectors, an image classifier of label set $\mathcal{Y}$ can be implemented by computing class posterior probabilities with

$$p(t_y|\mathbf{x};\mathcal{Y}) = \frac{\exp\left(\cos(\mathbf{w}_y, \mathbf{v})/\tau\right)}{\sum_{t_j \in \mathcal{Y}} \exp\left(\cos(\mathbf{w}_j, \mathbf{v})/\tau\right)}, \quad (1)$$

where $p(t_y|\mathbf{x};\mathcal{Y})$ is the probability of class label t_y given image $\mathbf{x}$, $\mathbf{v} = \Phi_{vis}(Emb_v(\mathbf{x})) \in \mathbb{R}^k$ the visual feature vector, $Emb_v(\cdot)$ an image embedding, $cos(\cdot,\cdot)$ the cosine similarity metric, and τ a temperature hyperparameter. Classification performance can usually be improved by inferring the classifier parameters $\mathbf{w}_y$ from multiple text prompts, e.g. by including context words such as a prompt prefix p ="a photo of", or p ="a drawing of", computing $\mathbf{w}_y = \Phi_{text}(Emb_t(\{p, t_y\}))$, and ensembling the vectors $\mathbf{w}_y$ obtained from multiple prompts [32, 48]. This, however, requires multiple forward passes through Φ_{text} during inference and can be undesirable for downstream applications.

More efficient inference can be achieved with prompt tuning [17, 44, 47, 48], which leverages a set of learnable parameters $\{\mathbf{c}_m^t\}_{m=1}^{M}$ as context features. These are prepended to each class name embedding $Emb_t(t_y)$ as text prompts, to produce the weight vectors $\mathbf{w}_y = \Phi_{text}(\{\mathbf{c}_1^t, \dots \mathbf{c}_M^t, Emb_t(t_y)\})$. Note that each $\mathbf{c}_i^t$ has the same dimension as the word embedding. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, context features can be end-to-end optimized with the cross-entropy loss

$$L_{\mathcal{Y}}(\mathbf{C}^t) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t_j \in \mathcal{Y}} -\mathbb{1}(t_j = t_{y_i}) \log p(t_j|\mathbf{x}_i;\mathcal{Y},\mathbf{C}^t) \quad (2)$$

for the classifier of (1), where $\mathbb{1}(\cdot)$ is the indicator function, and $\mathbf{C}^t$ the matrix of context features. Similarly, learnable prompts $\mathbf{c}_i^v$ can be inserted into the image branch, i.e. $\mathbf{v} = \Phi_{vis}(\{\mathbf{c}_1^v, \dots, \mathbf{c}_M^v, Emb_v(\mathbf{x})\})$, for better visual adaptation [16, 17, 44]. To prevent compromising the generalization of the FM embeddings, the parameters of the two encoders (i.e., Φ_{text}, Φ_{vis}) are frozen in the few-shot setting. In this paper, we consider two prompt tuning variants, CoOp [48] and MaPLe [17], the former using learnable prompts in the text branch, and the latter on both branches.

Class Taxonomy. A class taxonomy $\mathcal{Y}^{tax}$ organizes classes into a tree where classes of similar semantics are recursively assembled into superclasses, at each graph node (e.g. "dog" is a superclass of "Chihuahua" and "Corgi"). For a tree hierarchy, $\mathcal{T}$, each node $n \in \mathcal{N}$ has a single parent and multiple child nodes $Chd(n)$, where $\mathcal{N}$ is the set of tree nodes. Given a set of classes $\{t_y\}_{y=1}^{C}$, a tree hierarchy

$\mathcal{T}$ can be built by treating $\{t_y\}_{y=1}^{C}$ as leaf nodes (where $Chd(t_y) = \emptyset$), i.e., $Leaf(\mathcal{T}) = \{t_y\}_{y=1}^{C}$, and recursively grouping classes in a bottom-up manner until a single root node is created, according to the similarity relationships defined by the taxonomy $\mathcal{Y}^{tax}$. For example, ImageNet [5] classes are organized into a tree of 1,000 leaf nodes derived from the WordNet [11] taxonomy. Nodes that are not at the leaves are denoted as internal nodes $\mathcal{N}^{int} = \mathcal{N} \setminus Leaf(\mathcal{T})$.

4. Taxonomic Open Set Classification

Definition. A significant advantage of FMs for practical applications is their support for open set classification. Since the classifier of (1) can be implemented with any class names t_y, and the FM is trained with an open vocabulary, it is possible to perform classification for virtually any class. Prompting methods improve the classification of the classes defined by the label set $\mathcal{Y}$, but attempt to maintain this generality. However, for most applications "open set" does not mean the ability to recognize "any possible word." On the contrary, the whole point of prompt tuning is to enhance the FM for a given application *context*. This context defines what "open set" truly means for the application. In practice, it frequently means "all the possible ways" to refer to the classes in $\mathcal{Y}$.

One important component of this requirement is the ability to describe classes at different levels of granularity. For example, while user A (a car mechanic) may need to know if an image depicts a "Fan Clutch Wrench" or a "Box-Ended Wrench," user B (a retail store worker) may need to know if the exact same image depicts a "a mechanic's tool" or a "plumber's tool." A FM-based classification app should be deployable in both the car garage or the retail store. However, because the app is a tool classification app, the prompted model does not need to be good at recognizing "lollipops," which are beyond the context of the app. On the other hand, it is undesirable to have to prompt-tune the app for every specific use or user group. Ideally, it should be possible to prompt tune the FM *once,* with respect to the *entire* class taxonomy $\mathcal{Y}^{tax}$ of tools. The app can then be deployed to each user base without any retraining, by simply drawing the most suitable class names t_y from $\mathcal{Y}^{tax}$. We refer to this problem as **Taxonomic Open Set** (TOS) classification and introduce a formal definition in the remainder of this section.

Datasets. Most existing classification dataset can be used to study the TOS problem, since the very nature of taxonomies is to group objects or concepts into semantic classes of different levels of granularity. Hence, most vision datasets are already labeled taxonomically or adopt classes defined by a public taxonomy, usually WordNet [11]. We consider three popular datasets: Cifar100 [19], SUN [42] and ImageNet [5]. ImageNet is complemented by the ImageNetv2 [33], ImageNet-S [38], ImageNet-A [15] and ImageNet-R [14] to enable the study of generalization across image domains. For each dataset, the K-shot setting is considered, where K images per class are sampled for training. We consider $K = \{1, 2, 4, 8, 16\}$.

Label sets. Given a dataset $\mathcal{D}$ and class hierarchy $\mathcal{Y}^{tax}$ a label set $\mathcal{Y}$ is defined at each level of granularity, according the latter. The leaf label set $\mathcal{Y}_{leaf}$ is defined as the set of classes of $\mathcal{D}$ and the class hierarchy $\mathcal{T}$ is build recursively, denoting by $\mathcal{Y}_n = Chd(n)$ the set of class labels for the children of node n. In our experiments, we adopt the default hierarchy of the SUN dataset and use WordNet [11] to build the hierarchy for Cifar100 and ImageNet. The resulting class hierarchies are as follows. Cifar100 [19] contains 100 leaf nodes and 48 internal nodes. SUN contains 324 leaf nodes and 19 internal nodes (after pruning 73 leaf classes that have confusing superclasses). ImageNet [5], ImageNetv2 [33] and ImageNet-S [38] share a class hierarchy of 1,000 leaf nodes and 368 internal nodes. ImageNet-A [15] and ImageNet-R [14] only contain 200 subclasses and the corresponding internal nodes from the ImageNet hierarchy.

Metrics: Given a classifier

$$\hat{y}(\mathbf{x}; \mathcal{Y}) = \arg\max_{t_y \in \mathcal{Y}} p(t_y|\mathbf{x}; \mathcal{Y}) \tag{3}$$

using a label set $\mathcal{Y}$, several metrics are proposed to evaluate TOS performance.

Leaf Accuracy is defined as

$$Acc_{leaf} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[\hat{y}(\mathbf{x}_i; \mathcal{Y}_{leaf}) = t_{y_i}] \tag{4}$$

and measures the classification accuracy at the leaves of the taxonomic tree (usually defined as the "dataset classes"). This enables comparison of hierarchical classifiers to standard, or *flat*, classifiers which only consider the leaf classes.

Hierarchical Consistent Accuracy (HCA) is defined as

$$HCA = \frac{1}{N} \sum_{i=1}^{N} (\mathbb{1}[\hat{y}(\mathbf{x}_i; \mathcal{Y}_{leaf}) = t_{y_i}] \prod_{n \in \mathcal{A}(t_{y_i})} \mathbb{1}[\hat{y}(\mathbf{x}_i; \mathcal{Y}_n) \in \mathcal{A}(t_{y_i}) \cup \{t_{y_i}\}]), \tag{5}$$

where $\mathcal{A}(n)$ denotes all the ancestors of node n, and t_{y_i} is the leaf node corresponding to class label y_i. While Acc_{leaf} considers successful any correct classification at the leaf level of the tree, the *HCA* is stricter. It declares a success only when all the ancestors of the leaf node are correctly classified. In other words, each sample needs to be classified correctly at each tree level to be viewed as correctly classified under the *HCA*. Acc_{leaf} is an upper bound for the *HCA*.

Mean Treecut Accuracy (MTA) estimates the expected accuracy under the TOS classification setting. It computes the average accuracy over a set of treecuts $\mathcal{T}_c \in \Omega$,

$$MTA = \frac{1}{|\Omega|} \sum_{\mathcal{T}_c \in \Omega} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[\hat{y}(\mathbf{x}_i; \mathcal{Y}_{\mathcal{T}_c}) = t_{y_i}] \,, \tag{6}$$

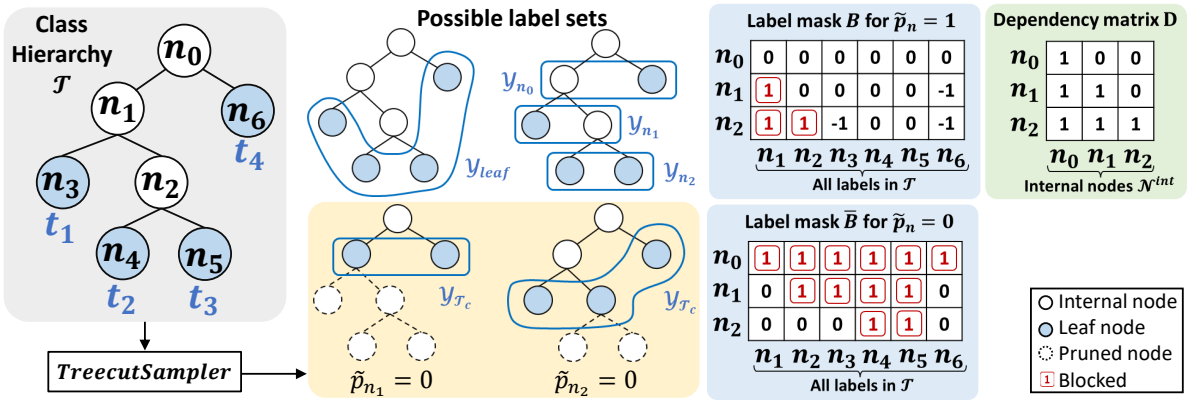


Figure 2. (Left) Multiple possible label sets are available in a class hierarchy. The label set can cover nodes at same level or across different hierarchy levels. (Right) Predefined matrices for efficient treecut sampling used in Algorithm 1.

where $\mathcal{Y}_{\mathcal{T}_c} = Leaf(\mathcal{T}_c)$. However, as shown by the following lemma (see appendix for proof), the set of all possible tree cuts in the hierarchy $\mathcal{T}$ is usually very large.

Lemma 4.1. *For a balanced M-ary tree with depth L (root node is excluded and is at depth 0), the number of all valid treecut is* $L + \sum_{l=2}^{L} \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{l-1}}$.

For example, a tree with $M = 2$ and $L = 6$ has more than 4 billion treecuts. For a dataset like ImageNet ($L = 15$), this number is monumental. Thus, we randomly sampled $|\Omega| = 25$ treecuts from $\mathcal{T}$ in all experiments and showed that it is already fairly stable.

State-of-the-art. To test TOS performance of the CLIP with existing prompting techniques, we performed an experiment on ImageNet. Table 1 summarizes the performance of the different methods under the three metrics. Two conclusions are possible. First, the sharp drop from Acc_{leaf} to HCA shows that none of the methods make consistent predictions across the class hierarchy. Second, the low MTAs show that the expected accuracy of TOS classification is dramatically smaller than that of flat classification (leaf classes).

5. Prompt Tuning for Hierarchical Consistency

To enhance TOS performance of FMs, we propose *Prompt Tuning for Hierarchical Consistency* (ProTeCt). ProTeCt can be implemented with many existing prompt tuning methods (e.g., CoOp, MaPLe). These methods optimize context prompts using the cross-entropy loss of (2) with leaf label set $\mathcal{Y}_{leaf}$. While this optimizes leaf accuracy Acc_{leaf}, it is not robust to label set changes, even for label sets comprised of superclasses of $\mathcal{Y}_{leaf}$. A simple generalization would be to replace (2) with $\mathcal{L}(\mathbf{C}^t) = \sum_{\mathcal{Y}_p \in \mathcal{T}} L_{\mathcal{Y}_p}(\mathbf{C}^t)$, i.e., to consider all the partial label sets $\mathcal{Y}_p$ of the tree $\mathcal{T}$. However, for sizeable taxonomies, this involves a very large number of label sets and is not feasible. ProTeCt avoids the problem by dynamically sampling label sets from $\mathcal{T}$ during training, with a combination of two learning objectives, a *node-centric loss* (NCL) and a *dynamic tree-cut loss* (DTL).

Node-Centric Loss (NCL). NCL is the aggregate cross-entropy loss of (2) over all node-centric label sets $\mathcal{Y}_n = Chd(n)$ defined by each internal node $n \in \mathcal{N}^{int}$ of the hierarchy, i.e.,

$$\mathcal{L}_{NCL}(\mathbf{C}^t) = \frac{1}{|\mathcal{N}^{int}|} \sum_{n \in \mathcal{N}^{int}} L_{\mathcal{Y}_n}(\mathbf{C}^t) \,. \tag{7}$$

NCL optimization encourages prompts that robustify the classification at different granularities. For example, "Corgi" should be classified as "mammal" within the animal label set $\mathcal{Y}_{n_1}$ ={mammal, reptile, bird}, as a "dog" in the mammal label set $\mathcal{Y}_{n_2}$ ={dog, cat, elephant, tiger}, and so forth.

Dynamic Treecut Loss (DTL). While NCL calibrates node classification, guaranteeing consistency within each node, the label sets of TOS classification can also span different sub-trees of the hierarchy, including nodes at different levels, e.g., $\mathcal{Y}$ ={dog, cat, elephant, tiger, reptile, bird}. DTL seeks to calibrate such label sets, by aggregating the cross-entropy loss of (2) dynamically, i.e., on an example basis, over randomly sampled label sets $\mathcal{Y}_{\mathcal{T}_c} = Leaf(\mathcal{T}_c)$ comprised of the leaves of the tree cuts $\mathcal{T}_c$ (sub-trees) of $\mathcal{T}$. At each training iteration, a random tree cut $\mathcal{T}_c$ is sampled with the $TreeCutSampler$ procedure of Algorithm 1, as illustrated on the middle of Fig. 2, to define the loss

$$\mathcal{L}_{DTL}(\mathbf{C}^t) = L_{\mathcal{Y}_{\mathcal{T}_c}}(\mathbf{C}^t) \quad \mathcal{T}_c \sim TreecutSampler(\mathcal{T}, \beta), \tag{8}$$

where $\beta \in [0, 1]$ is a rate of tree dropout. For this, a Bernoulli random variable $P_n \sim Bernoulli(\beta)$ of dropout rate β is defined for each internal node $n \in \mathcal{N}^{int} \setminus n_0$. The algorithm descends the tree $\mathcal{T}$, sampling a binary drop-out variable p_n at each node. If $p_n = 1$, node n is kept in the pruned tree $\mathcal{T}_c$. Otherwise, the sub-tree of $\mathcal{T}$ rooted with n is dropped from $\mathcal{T}_c$. The parameter β controls the degree of pruning. Larger β induces the pruning of more tree nodes, while $\beta = 0$ guarantess that $\mathcal{Y}_{\mathcal{T}_c} = \mathcal{Y}_{leaf}$. The root node n_0 is excluded, as $p_{n_0} = 0$ would imply discarding the whole $\mathcal{T}$.

The $TreeCutSampler$ algorithm is an efficient procedure to sample tree cuts $\mathcal{T}_c$ from $\mathcal{T}$. It starts by sampling a vector

Algorithm 1 Treecut Sampler

Input: The tree hierarchy $\mathcal{T}$ of the dataset, tree dropout rate β
Output: The treecut label set $\mathcal{Y}_{\mathcal{T}_c}$

```
// sampling p for internal nodes; prune the
   sub-tree rooted at n if p_n = 0
p_{n_0} ← 1 ;                    // always keep the root node
for n ∈ N^int \ n_0 do
    p_n ← Bernoulli(β)
p ← (p_{n_1^int}, ..., p_{n_K^int})
// correct p based on the node dependency
p̃ ← p ⊗ 𝟙[Dp = D1]
// obtain blocked labels with predefined masks
   and the sampled p̃
b ← min(B, 0)^T p̃ + B̄^T (1 − p̃)
// gather available (unblocked) labels as the
   sampled label set
Y_{T_c} ← {n_j : n_j ∈ N \ n_0, b_j = 0}
return Y_{T_c}
```

$\mathbf{p} = (p_{n_1^{int}}, ..., p_{n_K^{int}})$, where n_i^{int} denotes the i-th internal node and $K = |\mathcal{N}^{int}|$, containing pruning flags p_n for all internal nodes $n \in \mathcal{N}^{int}$. The next step is to enforce consistency between these flags, according to the tree structure. If any node in $\mathcal{A}(n)$ is pruned, then node n should be pruned even if $p_n = 1$. This is efficiently enforced across all the flags by defining a dependency matrix $\mathbf{D} \in \{0,1\}^{K \times K}$ where $\mathbf{D}_{ij} = \mathbb{1}[n_j^{int} \in \mathcal{A}(n_i^{int}) \cup \{n_i^{int}\}]$ indicates whether the i-th internal node n_i^{int} is a child of the j-th internal node n_j^{int}. An example is provided on the right of Fig. 2 for the tree on the left. The sampled flags are then corrected by computing $\tilde{\mathbf{p}} = \mathbf{p} \otimes \mathbb{1}[\mathbf{Dp} = \mathbf{D1}]$, where $\mathbf{1}$ is the vector of K ones and $\otimes$ the Hadamard product. Note that both $\mathbf{D}$ and $\mathbf{D1}$ are pre-computed, making the complexity of this step roughly that of one matrix-vector multiplication.

To identify the leaves of the sampled treecut ($\mathcal{Y}_{\mathcal{T}_c} = Leaf(\mathcal{T}_c)$) efficiently, a mask $\mathbf{B} \in \{0, 1, -1\}^{K \times |\mathcal{N} \setminus \{n_0\}|}$ is defined, where each row corresponds to an internal node, and the columns contain all possible labels in $\mathcal{T}$, i.e., all nodes except the root n_0. Entry B_{ij} flags that n_j cannot appear in the sampled label set, given that $n_i \in \mathcal{N}^{int}$ has not been pruned (i.e., $\tilde{p}_{n_i^{int}} = 1$), as follows

$$B_{ij} = \begin{cases} 1, \text{ if } n_j \in \mathcal{A}(n_i^{int}) \cup \{n_i^{int}\} & (n_j \textit{ is an ancestor of } n_i^{int}) \\ 0, \text{ if } n_i^{int} \in \mathcal{A}(n_j) & (n_j \textit{ is a descendant of } n_i^{int}) \\ -1, \text{ otherwise} & (n_j \textit{ is outside of the sub-tree rooted at } n_i^{int}) \end{cases} \tag{9}$$

Similarly, a matrix $\bar{\mathbf{B}}$, of entries $\bar{B}_{ij} = 1 - |B_{ij}|$, is defined to flag that n_j cannot appear in the label set, given that $n_i \in \mathcal{N}^{int}$ has been pruned, i.e. $\tilde{p}_{n_i^{int}} = 0$. A mask of the nodes unavailable to the label set is then computed by accumulating the masks corresponding to the values of $\tilde{\mathbf{p}}$,

$$\mathbf{b} = min(\mathbf{B}, 0)^T \tilde{\mathbf{p}} + \bar{\mathbf{B}}^T (\mathbf{1} - \tilde{\mathbf{p}}) \,, \tag{10}$$

where the mask in $min(\mathbf{B}, 0)$ is selected if $\tilde{p}_n = 1$, and that in $\bar{\mathbf{B}}$ if $\tilde{p}_n = 0$. Note that $min(\mathbf{B}, 0)$ clips $B_{ij} = -1$ to 0. The mask $\mathbf{b}$ can then be used to obtain $\mathcal{Y}_{\mathcal{T}_c} = Leaf(\mathcal{T}_c) = \{n_j : n_j \in \mathcal{N} \setminus n_0, b_j = 0\}$. Fig. 2 gives an example. When $\tilde{\mathbf{p}} = (\tilde{p}_{n_0}, \tilde{p}_{n_1}, \tilde{p}_{n_2}) = (1, 0, 0)$, then $\mathbf{b} = min(\mathbf{B}_1, 0) + \bar{\mathbf{B}}_2 + \bar{\mathbf{B}}_3 = (0, 1, 1, 2, 2, 0)$, signaling that only n_1 and n_6 are available to the label set (as $b_1, b_6 = 0$), resulting in $\mathcal{Y}_{\mathcal{T}_c} = \{n_1, n_6\}$. More detailed examples are given in the appendix.

Optimization. The overall loss used for prompt tuning is a combination of the two losses

$$\mathcal{L}(\mathbf{C}^t) = \mathcal{L}_{DTL}(\mathbf{C}^t) + \lambda \mathcal{L}_{NCL}(\mathbf{C}^t) \tag{11}$$

where λ is a hyperparameter. Note that, like previous prompting approaches, ProTeCt optimizes the learnable prompts $\{\mathbf{c}_m\}_{m=1}^M$ while keeping the parameters of Φ_{text}, Φ_{vis} frozen.

6. Experiments

In this section, we discuss experiments for evaluating the effectiveness of ProTeCt. To demonstrate that ProTeCt is a plug-an-play method, it was applied to two SOTA prompt tuning methods: CoOp [48] and MaPLe [17]. Each experiment is averaged over 3 runs and full tables with error bars are shown in the appendix for brevity. All experiments were conducted on a single Nvidia A10 GPU, using Pytorch [31]. Please see the appendix for more training details and results. ProTeCt code builds on the publicly available codebases for CoOp and MaPLe and will be released upon publication.

Metrics: Acc_{leaf} of (4), HCA of (5) and MTA of (6) are considered. MTA uses 5 tree dropout rates ($\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$) to sample treecuts of various granularities. For each β, T treecuts are sampled without repetition to obtain a total of $5T$ treecuts. MTA($5T$) indicates the result is averaged over these $5T$ treecuts. We ablate $T = 5$ and $T = 20$ on Cifar100 and use $T = 5$ for all datasets by default.

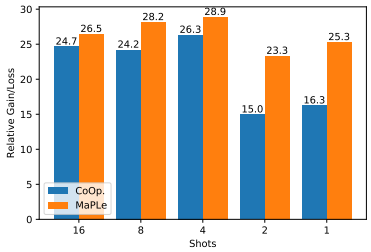
Training Details: All vanilla prompt-tuning and their ProTeCt counterparts are trained under the same setting. The following configuration is used unless noted. All experiments use SGD optimizer and the learning rate is set to 0.02 with a cosine learning rate scheduler. By default, a pretrained ViT-B/16 CLIP model is used as initialization. For Cifar100 and SUN, we train both CoOp and MaPLe prompts for 200 epochs, using a batch size of 128 and 32, respectively. For ImageNet, CoOp is trained for 30 epochs with a batch size of 8, while MaPLe is trained for 10 epochs with a batch size of 2. Note that the setting is slightly different from the original paper due to our GPU availability.

6.1. TOS Classification Performance

Table 2 shows that vanilla CoOp and MaPLe have reasonable leaf accuracy for both 1-shot and 16-shot classification on Cifar100, SUN, and ImageNet. However, their very low HCA shows that their predictions are not consistent over the class hierarchy. As a result, their TOS classification performance (MTA) is much weaker than their leaf accuracy.

Method	K-Shot	w/ ProTeCt	Cifar100 Acc_{leaf}	Cifar100 HCA	Cifar100 MTA (25)	Cifar100 MTA (100)	SUN Acc_{leaf}	SUN HCA	SUN MTA (25)	ImageNet Acc_{leaf}	ImageNet HCA	ImageNet MTA (25)
CoOp	16		72.88	10.04	50.64	51.14	73.82	38.28	52.99	71.23	2.99	46.98
	16	✓	72.94	56.85	87.69	87.30	74.59	62.94	83.51	69.92	37.74	88.61
			(+0.06)	(+46.81)	(+37.05)	(+36.16)	(+0.77)	(+24.66)	(+30.52)	(-1.31)	(+34.75)	(+41.63)
	1		65.03	7.81	41.78	44.17	63.65	33.36	51.20	63.67	1.59	40.52
	1	✓	66.88	41.01	81.64	81.01	63.79	49.62	76.25	66.11	25.79	86.14
			(+1.85)	(+33.2)	(+39.86)	(+36.84)	(+0.14)	(+16.26)	(+25.05)	(+2.44)	(+24.2)	(+45.62)
MaPLe	16		75.01	17.54	52.21	50.82	71.86	33.25	54.29	70.70	4.15	48.29
	16	✓	75.34	61.15	88.04	88.33	72.17	59.71	82.27	69.52	31.24	87.87
			(+0.33)	(+43.61)	(+35.83)	(+37.51)	(+0.31)	(+26.46)	(+27.98)	(-1.18)	(+27.09)	(+39.58)
	1		68.75	4.65	50.60	54.99	63.98	25.15	50.31	68.91	2.97	48.16
	1	✓	69.33	48.10	83.36	83.78	64.29	50.45	76.73	66.16	20.44	85.18
			(+0.58)	(+43.45)	(+32.76)	(+28.79)	(+0.31)	(+25.30)	(+26.42)	(-2.75)	(+17.47)	(+37.02)

Table 2. TOS performance w/ and w/o ProTeCt on Cifar100 ($\lambda = 0.5$), SUN ($\lambda = 0.5$) and ImageNet ($\lambda = 1$). $\beta = 0.1$ for all datasets.



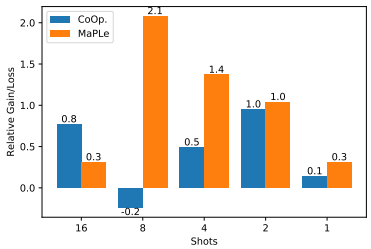
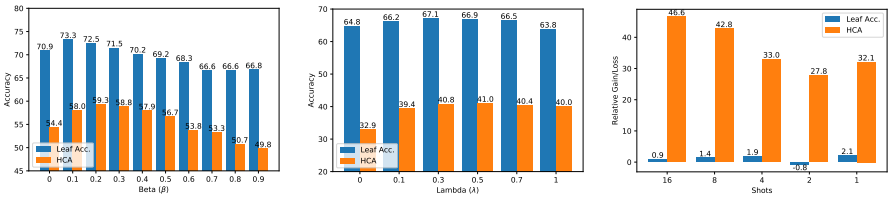


Figure 3. Relative gain/loss after adding ProTeCt to CoOp and MaPle, respectively. (Top) HCA ; (Bottom) Acc_{leaf}.



(a) (b) (c)

Figure 4. Ablation of (a) tree dropout rate β, (b) NCL strength λ and (c) CLIP ViT B32 architecture.

(a): [Boxer, Person] (b): [Trolleybus, Animal] (c):[Lion, Koala] (d): [Cheeseburger, Ice cream]

Figure 5. ProTeCt correctly predicts examples from ImageNet (a,b) and its variants (c,d) at all levels. [GT, Prediction] shows the groundtruth and incorrect prediction by vanilla prompt tuning.

For example, 16-shot classification with CoOp on ImageNet has a leaf accuracy of 71.23, but expected TOS accuracy of 46.98. This is explained by the very low HCA of 2.99. Similar observations hold for different few-shot configurations. In all cases, ProTeCt (results on rows with a checkmark) significantly improves HCA and MTA(25). For example, it boosts the HCA of 16-shot classification with CoOp on ImageNet by 34.75 (2.99 vs 37.74), leading to an increase of MTA(25) of 41.63 (46.98 to 88.61).

Note that, in all cases, MTA(25) after ProTeCt training is *higher* than leaf accuracy. This is expected for a well-calibrated classifier, since decisions at intermediate levels of the tree are coarser-grained than those at the leaves, which can require very fine class distinctions. These results show that ProTeCt robustifies the model for use in the TOS classification setting. The table also shows that ProTeCt maintains leaf accuracies comparable to those of the vanilla methods. Furthermore, the MTA results when 25 and 100 treecuts are sampled (corresponding to $T = 5$ and $T = 20$), are compared on Cifar100. It can be seen that the performances are similar, showing that sampling 25 treecuts is sufficient to achieve good estimation. Fig. 3 compares the **relative** gains in HCA and leaf accuracy of training with ProTeCt, as compared to vanilla prompt tuning. These gains are shown for both CoOp and MaPLe, under several few shot configurations, on SUN dataset. In all cases, ProTeCt increases HCA by more than 15 points, while maintaining a leaf accuracy comparable to that of vanilla CoOp/MaPLe. Similar results for Cifar100 and ImageNet can be found in appendix.

6.2. Domain Generalization of TOS Classification

We investigate whether TOS classification performance generalizes across datasets, following the domain generalization setting of [17, 44, 47, 48]. The CLIP model with ProTeCt prompts trained on ImageNet (source) is applied to 4 ImageNet variants (target) with visual domain shift: ImageNetv2 [33], ImageNet-Sketch [38], ImageNet-A [15] and ImageNet-R [14]. Table 3 summarizes the three metrics on these datasets for CoOp and MaPLe. Similarly to Table 2, ProTeCt enables significant gains in HCA and MTA(25) over the baselines for all datasets. Note that since ImageNet-A and ImageNet-R only contain 200 ImageNet subclasses, their

Method	K-Shot	w/ ProTeCt	ImageNetv2 [33]			ImageNet-S [38]			ImageNet-A [15]			ImageNet-R [14]		
			Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)
CoOp	16		64.01	2.31	43.74	47.82	1.39	38.58	50.28	2.97	52.56	75.83	18.49	64.13
	16	✓	62.60	32.84	86.66	46.80	20.73	82.60	49.08	22.45	78.21	74.94	31.18	75.59
			(-1.41)	(+30.53)	(+42.92)	(-1.02)	(+19.34)	(+44.02)	(-1.20)	(+19.48)	(+25.65)	(-0.89)	(+12.69)	(+11.40)
	1		56.43	1.51	38.27	41.38	1.11	33.61	45.92	1.76	47.54	69.84	11.74	55.31
	1	✓	60.16	22.95	84.38	44.75	13.88	80.64	48.95	20.52	76.95	74.26	27.46	76.48
			(+3.73)	(+21.44)	(+46.11)	(+3.37)	(+12.77)	(+47.03)	(3.03)	(+18.76)	(+29.41)	(+4.42)	(+15.72)	(+21.17)
MaPLe	16		64.15	1.97	45.93	48.97	1.58	43.37	50.61	2.31	54.88	76.61	20.67	63.06
	16	✓	62.77	27.86	86.14	47.47	17.77	82.52	47.41	19.75	77.46	75.70	32.58	77.99
			(-1.38)	(+25.89)	(+40.21)	(-1.50)	(+16.19)	(+39.15)	(-3.20)	(+17.44)	(+22.58)	(-0.91)	(+11.91)	(+14.93)
	1		61.78	2.18	45.50	46.79	1.70	45.26	47.55	3.52	55.48	74.55	18.85	62.48
	1	✓	59.14	17.89	83.27	44.92	11.24	79.94	47.15	16.03	76.81	74.60	25.20	75.72
			(-2.64)	(+15.71)	(+37.77)	(-1.87)	(+9.54)	(+34.68)	(-0.40)	(+12.51)	(+21.33)	(+0.05)	(+6.35)	(+13.24)

Table 3. The gain of hierarchical consistency after adding ProTeCt generalizes across datasets in unseen domains. All methods are fine-tuned on ImageNet and evaluated on its 4 variants.

DTL	NCL	16-shot			1-shot		
		Acc_{Leaf}	HCA	MTA (25)	Acc_{Leaf}	HCA	MTA (25)
		72.88	10.04	50.64	65.03	7.81	41.78
✓		72.81	47.97	87.32	64.77	32.93	81.38
	✓	64.20	51.69	79.44	61.22	38.02	62.16
✓	✓	**72.94**	**56.85**	**87.69**	**66.88**	**41.01**	**81.64**

Table 4. Loss ablation with CoOp on Cifar100 dataset. Both losses improve the hierarchical consistency.

K-Shot	w/ ProTeCt	CLIP-Adapter [12]			CLIP+LORA [9]		
		Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)
16		71.96	5.59	42.93	70.45	4.57	47.19
16	✓	72.47	57.15	87.67	70.64	51.06	77.29
		(+0.51)	(+51.56)	(+44.83)	(+0.19)	(+46.49)	(+30.10)
1		65.35	8.35	48.25	63.57	2.89	38.63
1	✓	67.29	36.21	78.49	63.62	24.66	56.42
		(+1.94)	(+27.86)	(+30.24)	(+0.05)	(+21.8)	(+17.79)

Table 5. ProTeCt also improves adapter-based methods, including CLIP-Adapter [12] and CLIP+LORA [9] (dataset: Cifar100).

hierarchy is different from that of ImageNet. These results demonstrate the flexibility and robustness of ProTeCt, even when transferring the model to a target domain whose class hierarchy is different from that of the source domain.

6.3. Ablation Study and Visualization

In this section, we discuss the ablations of ProTeCt components and visualize the predictions (more in the appendix).

Tree Dropout Rate β: Fig. 4 (a) plots Cifar100 Acc_{leaf} and HCA as a function of the drop-out rate β, for 16-shot CoOp+ProTeCt training ($\lambda = 1$). Larger values of β reduce the likelihood of sampling the leaf nodes of the tree, resulting in shorter trees and weaker regularization. Hence, both leaf accuracy and HCA degrade for large β. However, always using the full tree ($\beta = 0$) also achieves sub-optimal results. The two metrics peak at $\beta = 0.1$ and $\beta = 0.2$, respectively. $\beta = 0.1$ is selected for all experiments.

Loss: Fig. 4(b) ablates the strength of NCL loss (i.e. λ) for ProTeCt+CoOp using 1-shot setting on Cifar100 and $\beta = 0.1$. The introduction of NCL improves leaf accuracy/HCA from 64.8/32.9 ($\lambda = 0$) to 66.9/41 ($\lambda = 0.5$). We adopt $\lambda = 0.5$ for CIFAR100 and SUN. For ImageNet, $\lambda = 0.5$ and $\lambda = 1$ have similar performance. Table 4 further summarizes the CoOp+ProTeCt performance with and without the two losses of (11). Both losses improve TOS performance individually and there is a large additional gain when they are combined. Using NCL alone can degrade leaf performance, due to the lack of regularization across different levels of the hierarchy. The combination of the two losses overcomes this problem.

Architecture: Fig. 4 (c) shows that the gains for CoOp+ProTeCt in Fig. 3 with CLIP ViT B16 also hold for ViT B32, showing the plug-and-play properties of ProTeCt.

Adapter-based tuning methods: We further use the ProTeCt losses to train the CLIP adapter of [12] and the CLIP+LORA method of [9] to test the generation of ProTeCt. Table 5 shows that this again produces large consistency gains on the TOS setting, indicating that ProTeCt losses generalize to both prompt-based and adapter-based methods.

Visualization: Fig. 5 shows examples from ImageNet (a,b) and its variants (c,d). While ProTeCt correctly classifies these examples at all hierarchy levels, vanilla prompt tuning fails at certain levels. More examples are in the appendix.

7. Conclusion

In this work, we formulated the TOS classification setting, including datasets, performance metrics, and experiments. Given a dataset, a class hierarchy is built by assigning dataset classes to leaf nodes and superclasses to internal nodes. The TOS classifier is then expected to support classification with label sets drawn throughout the taxonomy. We have shown that existing FMs and prompting methods fail under this setting and proposed ProTeCt training to enhance the TOS performance of FMs, as a plug-and-play method. ProTeCt includes two losses. A dynamic treecut loss, based on an efficient treecut sampler, dynamically regularizes labels of varying granularity. A node-centric loss encourages correct predictions at all hierarchy levels. Experiments show that ProTeCt enhances TOS performance of existing prompt-tuning techniques, and the gain generalizes across unseen domains. Finally, we show that ProTeCt is applicable to various architectures, hierarchies, and parameter-tuning methods.

Acknowledgement This work was partially funded by NSF awards IIS-2303153, and a gift from Qualcomm. We also acknowledge and thank the use of the Nautilus platform for some of the experiments discussed above.

References

[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[2] Adrian Bulat and Georgios Tzimiropoulos. Lasp: Text-to-text optimization for language-aware soft prompting of vision&language models. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.org, 2020. 1, 3

[4] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *ArXiv*, abs/2003.04297, 2020. 1, 3

[5] 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*, pages 248–255, 2009. 2, 4

[6] Jia Deng, Jonathan Krause, Alexander C. Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 3

[7] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)*, 2014. 3

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3

[9] Sivan Doveh, Assaf Arbelle, Sivan Harary, Rameswar Panda, Roei Herzig, Eli Schwartz, Donghyun Kim, Raja Giryes, Rogério Schmidt Feris, Shimon Ullman, and Leonid Karlinsky. Teaching structured vision & language concepts to vision & language models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2657–2668, 2022. 8

[10] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models. In *International Joint Conference on Artificial Intelligence*, 2022. 2

[11] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998. 2, 4, 16

[12] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Jiao Qiao. Clip-adapter: Better vision-language models with feature adapters. *ArXiv*, abs/2110.04544, 2021. 8

[13] Wonjoon Goo, Juyong Kim, Gunhee Kim, and Sung Ju Hwang. Taxonomy-regularized semantic deep convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[14] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Lixuan Zhu, Samyak Parajuli, Mike Guo, Dawn Xiaodong Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8320–8329, 2020. 2, 4, 7, 8

[15] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021. 2, 4, 7, 8

[16] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 3

[17] Muhammad Uzair khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1, 2, 3, 6, 7

[18] Hyo Jin Kim and Jan-Michael Frahm. Hierarchy of alternating specialists for scene recognition. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, Citeseer*, 2009. 4

[20] Kibok Lee, Kimin Lee, Kyle Min, Yuting Zhang, Jinwoo Shin, and Honglak Lee. Hierarchical novelty detection for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[21] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. 3

[22] Haifeng Li, Xin Dou, Chao Tao, Zhixiang Wu, Jie Chen, Jian Peng, Min Deng, and Ling Zhao. Rsi-cb: A large-scale remote sensing image classification benchmark using crowdsourced data. *Sensors*, 20(6):1594, 2020. 11, 16

[23] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, 2021. Association for Computational Linguistics. 3

[24] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, 2022. Association for Computational Linguistics. 3

[25] Yuntao Liu, Yong Dou, Ruochun Jin, and Peng Qiao. Visual tree convolutional neural network in image classification. In *International Conference on Pattern Recognition (ICPR)*, 2018. 3

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer:

Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[27] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[28] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, 2022. 3

[29] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *ArXiv*, abs/1306.5151, 2013. 11, 16

[30] Marcin Marszałek and Cordelia Schmid. Semantic hierarchies for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2, 3

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 1, 3

[33] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 2019. 2, 4, 7, 8

[34] Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 3

[35] Babak Shahbaba and Radford M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis, 2(1):221–238*, 2007. 3

[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016. 1, 3

[37] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018. 1, 3

[38] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019. 2, 4, 7, 8

[39] Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiaoyong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *ArXiv*, abs/2302.10035, 2023. 2

[40] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. Cris: Clip-driven referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 3

[41] Tz-Ying Wu, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos. Solving long-tailed recognition with deep realistic taxonomic classifier. In *European Conference on Computer Vision (ECCV)*, 2020. 2, 3

[42] 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*, pages 3485–3492, 2010. 4, 16

[43] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015. 2, 3

[44] Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. Unified vision and language prompt learning. *ArXiv*, abs/2210.07225, 2022. 1, 3, 7

[45] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *ArXiv*, abs/2304.00685, 2023. 2

[46] Bin Zhao, Li Fei-Fei, and Eric P. Xing. Large-scale category structure aware image categorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011. 3

[47] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3, 7

[48] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision (IJCV)*, 2022. 1, 2, 3, 6, 7, 16

[49] Xinqi Zhu and Michael Bain. B-cnn: Branch convolutional neural network for hierarchical classification. *CoRR*, abs/1709.09890, 2017. 3

ProTeCt: Prompt Tuning for Taxonomic Open Set Classification

Supplementary Material

The appendix is organized as follows. Section 8 provides more training details for ProTeCt. Section 9 shows the complete proof of Lemma 4.1. Section 10 shows more examples for explaining the implementation of treecut sampler. Section 11, Section 12, and Section 13 shows the complete results conducted on Cifar100, Sun and ImageNet, respectively. Section 13 further shows the complete domain generalization results by applying the model trained on ImageNet to its 4 variants in a zero-shot fashion. We also test the robustness of ProTeCt on additional hierarchies in Section 14 with the FGVC Aircraft [29] dataset and the RSI-CB [22] satellite dataset. Ablations of different ProTeCt components are shown in Section 15 and more visualizations of incorrect predictions from existing prompt tuning methods are illustrated in Section 16.

8. Additional Training Details

In addition to the training details provided in the main paper, we list the url links that are used for training and evaluating ProTeCt. For CoOp and CoCoOp baselines, we adopt the code from https://github.com/KaiyangZhou/CoOp. For MaPLe, we adopt the code from https://github.com/muzairkhattak/multimodal-prompt-learning.

9. Treecut size of a balanced M-ary tree

Lemma 4.1. *For a balanced M-ary tree with depth L (root node is excluded and is at depth 0), the number of all valid treecut is* $L + \sum_{l=2}^{L} \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{l-1}}$

Proof. This can be proved by induction. Given an M-ary tree with depth L, the number of treecuts is denoted as f_L. The idea is that when adding the depth L, we only need to recompute the additional possible treecuts between depth $L-1$ and L. Since there are $N = M^{L-1}$ nodes in layer $L-1$, the possible treecuts after adding layer L is $1 + \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}}$, where 1 indicates the treecut that covers all nodes at depth L and $\sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}}$ means k nodes are covered in layer $L-1$. Below is the proof.

- When $L = 1$, $f_1 = 1$.
- When $L = 2$, $f_2 = 1 + f_1 + \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}}$. Consider the binary case, where $M = 2$ and $N = M^{L-1} = 2$, then $f_2 = 1 + f_1 + \frac{2!}{1!(2-1)!} = 1 + 1 + 2 = 4$
- Similarly, $f_3 = 1 + f_2 + \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}}$
- $$\begin{aligned} f_L &= 1 + f_{L-1} + \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}} \\ &= 1 + 1 + f_{L-2} + \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-2}} \\ &+ \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{L-1}} \\ &= L + \sum_{l=2}^{L} \sum_{k=1}^{N-1} \frac{N!}{k!(N-k)!}|_{N=M^{l-1}} \end{aligned}$$

□

10. Additional Examples for Treecut Sampler

In this section, we provide more detailed examples of the proposed Treecut sampler (i.e. Algorithm 1 in the main paper). Given the class hierarchy $\mathcal{T}$ on the left of Figure 6, three possible treecuts can be sampled by $\mathcal{T}$, i.e., $\mathcal{Y}_{\mathcal{T}_c} = \{n_1, n_6\}$ (see Figure 6), $\mathcal{Y}_{\mathcal{T}_c} = \{n_2, n_3, n_6\}$ (see Figure 7), and $\mathcal{Y}_{\mathcal{T}_c} = \{n_3, n_4, n_5, n_6\}$ (see Figure 8), depending on the sampled values p_n at each internal node $n \in \mathcal{N}^{int} = \{n_0, n_1, n_2\}$. Note that p_{n_0} is always set to 1 to ensure that the tree is not entirely pruned. As described in the paper, we use a dependency matrix $\mathbf{D}$ to correct $\mathbf{p}$ as $\tilde{\mathbf{p}}$, which is aligned with the dependency relationship among the internal nodes. For example, in the example shown in Figure 6, $p_{n_2} = 1$ is corrected as $\tilde{p}_{n_2} = 0$, since n_2 depends on n_1 and $p_{n_1} = 0$. A mask $\mathbf{b}$, flagging the unavailable labels, is then computed according to the values of $\tilde{\mathbf{p}}$. More specifically, the corresponding row in $min(\mathbf{B}, 0)$ is fetched when $\tilde{p}_n = 1$, and that in $\bar{\mathbf{B}}$ is used when $\tilde{p}_n = 0$, for each internal node $n \in \mathcal{N}^{int}$. These masks are accumulated into the final mask $\mathbf{b}$, as shown on the right of each figure, where entries of 0 indicate the available labels for the sampled label set. For example, in Figure 7, the sampled label set contains $\{n_2, n_3, n_6\}$, because b_2, b_3 and b_6 are 0s. Note that since the proposed Treecut Sampler maintains the node dependency with pre-computed matrices defined by the given hierarchy $\mathcal{T}$, it does not require any recursive traversal over the tree, and thus it is very efficient for the *on-the-fly* treecut sampling.

11. Complete Table of Cifar100 Experiments

In this section, we report the complete experiment results conducted on Cifar100. Table 6, Table 7 and Table 8 shows the results of vanilla CoOp and its results after adding ProTeCt. The CLIP features from ViT B16, ViT B32 and ViT

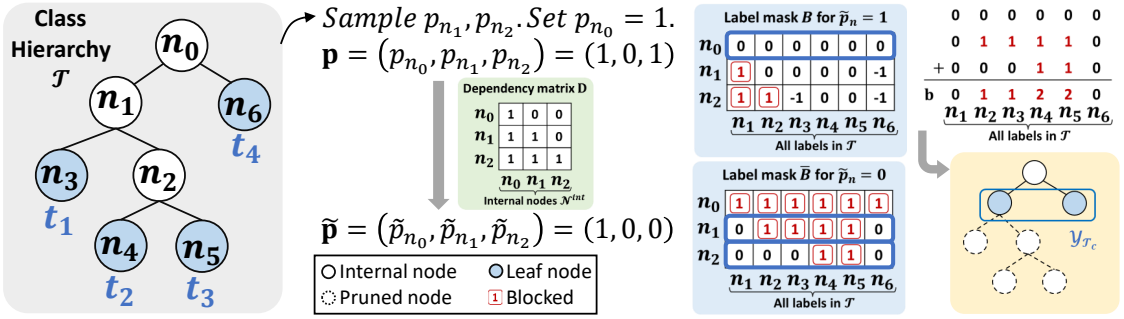


Figure 6. Treecut example of $\mathcal{Y}_{\mathcal{T}_c} = \{n_1, n_6\}$.

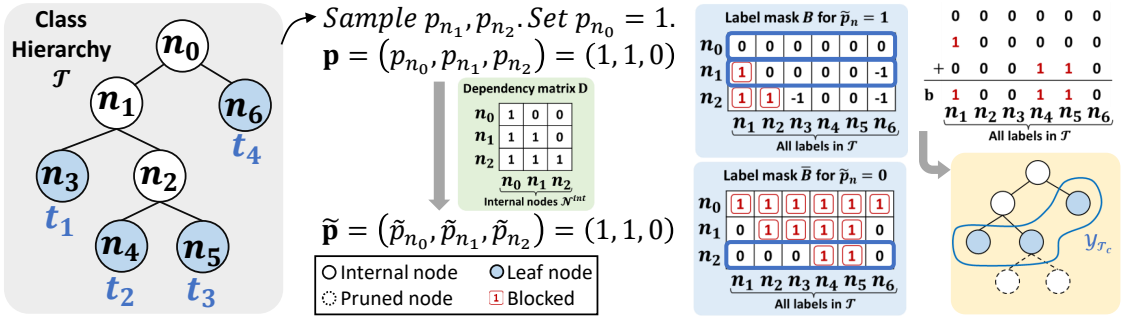


Figure 7. Treecut example of $\mathcal{Y}_{\mathcal{T}_c} = \{n_2, n_3, n_6\}$.

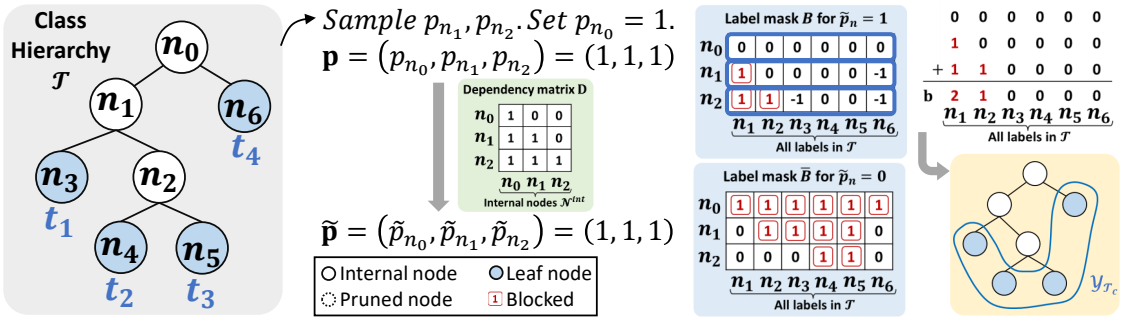


Figure 8. Treecut example of $\mathcal{Y}_{\mathcal{T}_c} = \{n_3, n_4, n_5, n_6\}$.

L14 are considered in Table 6, Table 7 and Table 8, respectively. While it is known that CLIP ViT L14 has a more powerful representation than ViT B32 and ViT B16 (also reflected in the leaf accuracy between three tables), all of them perform equally poor in terms of HCA (10.04/4.95/11.14 for 16-shot CoOp using CLIP B16/B32/L14 feature). This shows that simply using a stronger CLIP feature does not address the problem of hierarchical classification and does not improve hierarchical consistency. Furthermore, Table 6 contains the result of ProTeCt without using the treecut sampler ($\beta = 0$; Block 2 and Block 3) and without using NCL loss of (7) ($\lambda = 0$; Block 4) under multiple low-shot settings. For example, when 16-shot is considered, adding both NCL loss and treecut sampler ($\lambda = 0.5$ and $\beta = 0.1$) gives the result of 56.85 for HCA. Removing the tree dropout ($\lambda = 0.5$ and $\beta = 0$) yields 51.99 and removing the NCL loss ($\lambda = 0$ and $\beta = 0.1$) yields 47.97. This shows that both the NCL loss and the treecut sampler are important and lead to a significant gain over vanilla CoOp (HCA=10.04). Table 9 shows similar results when adding ProTeCt on MaPLe. Furthermore, we sampled T treecuts for each dropout rate $\beta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, where $T = 5$ and $T = 20$, resulting in 25 and 100 treecuts, respectively. Table 10 demonstrates ProTeCt can improve the MTA metric for both CoOp and MaPLe for both 25 and 100 randomly sampled treecuts. Table 11 further shows that ProTeCt can generalize to ResNet-based architectures.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B16	16		N/A	N/A	72.88 ± 0.62	10.04 ± 1.11
CoOp	ViT B16	8		N/A	N/A	70.84 ± 0.85	6.03 ± 0.64
CoOp	ViT B16	4		N/A	N/A	69.47 ± 0.90	6.15 ± 1.04
CoOp	ViT B16	2		N/A	N/A	68.17 ± 0.57	4.19 ± 0.81
CoOp	ViT B16	1		N/A	N/A	65.03 ± 0.56	7.81 ± 0.14
CoOp	ViT B16	16	✓	0.5	0	72.08 ± 0.38	51.99 ± 0.24
CoOp	ViT B16	8	✓	0.5	0	68.94 ± 0.52	49.01 ± 0.54
CoOp	ViT B16	4	✓	0.5	0	66.38 ± 1.18	45.24 ± 0.93
CoOp	ViT B16	2	✓	0.5	0	63.96 ± 0.57	42.78 ± 1.49
CoOp	ViT B16	1	✓	0.5	0	62.01 ± 0.80	34.90 ± 1.08
CoOp	ViT B16	16	✓	1	0	70.86 ± 0.59	54.39 ± 0.68
CoOp	ViT B16	8	✓	1	0	68.76 ± 0.90	52.14 ± 0.32
CoOp	ViT B16	4	✓	1	0	66.92 ± 0.20	47.63 ± 0.54
CoOp	ViT B16	2	✓	1	0	64.87 ± 1.28	40.74 ± 0.87
CoOp	ViT B16	1	✓	1	0	62.57 ± 0.06	38.97 ± 1.29
CoOp	ViT B16	16	✓	0	0.1	72.81 ± 0.31	47.97 ± 0.70
CoOp	ViT B16	8	✓	0	0.1	70.94 ± 0.18	48.53 ± 0.02
CoOp	ViT B16	4	✓	0	0.1	69.10 ± 0.92	45.20 ± 0.25
CoOp	ViT B16	2	✓	0	0.1	68.85 ± 0.11	42.28 ± 1.57
CoOp	ViT B16	1	✓	0	0.1	64.77 ± 1.37	32.93 ± 0.42
CoOp	ViT B16	16	✓	0.5	0.1	72.94 ± 0.83	56.85 ± 1.60
CoOp	ViT B16	8	✓	0.5	0.1	71.10 ± 1.06	52.27 ± 0.62
CoOp	ViT B16	4	✓	0.5	0.1	69.46 ± 0.58	48.71 ± 0.13
CoOp	ViT B16	2	✓	0.5	0.1	68.63 ± 0.67	46.03 ± 0.24
CoOp	ViT B16	1	✓	0.5	0.1	66.88 ± 0.21	41.01 ± 1.18
CoOp	ViT B16	16	✓	1	0.1	73.26 ± 0.66	58.01 ± 0.43
CoOp	ViT B16	8	✓	1	0.1	70.10 ± 0.08	52.81 ± 0.05
CoOp	ViT B16	4	✓	1	0.1	68.41 ± 0.50	49.59 ± 0.89
CoOp	ViT B16	2	✓	1	0.1	67.73 ± 1.25	45.27 ± 0.28
CoOp	ViT B16	1	✓	1	0.1	63.84 ± 1.51	40.05 ± 1.48

Table 6. Performance of few-shot CoOp on Cifar100 under ViT B16. Ablations cover both NCL strengths λ and tree dropout rate β.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B32	16		N/A	N/A	68.13 ± 0.19	4.95 ± 0.61
CoOp	ViT B32	8		N/A	N/A	65.52 ± 0.15	5.82 ± 0.29
CoOp	ViT B32	4		N/A	N/A	63.42 ± 1.40	8.56 ± 0.72
CoOp	ViT B32	2		N/A	N/A	63.65 ± 0.60	10.25 ± 0.88
CoOp	ViT B32	1		N/A	N/A	59.53 ± 0.60	3.43 ± 0.86
CoOp	ViT B32	16	✓	0	0.1	68.42 ± 0.91	47.79 ± 0.54
CoOp	ViT B32	8	✓	0	0.1	66.39 ± 0.48	44.47 ± 0.98
CoOp	ViT B32	4	✓	0	0.1	64.73 ± 0.17	31.72 ± 0.33
CoOp	ViT B32	2	✓	0	0.1	64.55 ± 0.44	30.78 ± 0.66
CoOp	ViT B32	1	✓	0	0.1	60.91 ± 0.42	34.64 ± 0.55
CoOp	ViT B32	16	✓	0.5	0.1	68.87 ± 1.09	51.55 ± 0.65
CoOp	ViT B32	8	✓	0.5	0.1	66.85 ± 0.32	48.39 ± 1.35
CoOp	ViT B32	4	✓	0.5	0.1	65.41 ± 0.74	41.63 ± 0.39
CoOp	ViT B32	2	✓	0.5	0.1	62.86 ± 0.81	38.13 ± 0.61
CoOp	ViT B32	1	✓	0.5	0.1	61.59 ± 0.80	35.65 ± 0.19
CoOp	ViT B32	16	✓	1	0.1	68.93 ± 0.22	51.67 ± 0.58
CoOp	ViT B32	8	✓	1	0.1	65.54 ± 0.54	48.36 ± 0.63
CoOp	ViT B32	4	✓	1	0.1	64.28 ± 0.07	42.78 ± 1.04
CoOp	ViT B32	2	✓	1	0.1	61.68 ± 0.67	40.53 ± 0.42
CoOp	ViT B32	1	✓	1	0.1	58.98 ± 0.88	36.59 ± 0.76

Table 7. Performance of few-shot CoOp on Cifar100 under ViT B32. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT L14	16		N/A	N/A	79.98 ± 0.97	11.14 ± 0.47
CoOp	ViT L14	8		N/A	N/A	79.37 ± 0.90	6.91 ± 0.67
CoOp	ViT L14	4		N/A	N/A	77.34 ± 0.78	7.78 ± 0.82
CoOp	ViT L14	2		N/A	N/A	76.63 ± 0.65	5.21 ± 0.87
CoOp	ViT L14	1		N/A	N/A	73.26 ± 0.95	4.87 ± 0.15
CoOp	ViT L14	16	✓	0	0.1	81.17 ± 0.34	63.40 ± 0.30
CoOp	ViT L14	8	✓	0	0.1	80.00 ± 0.98	62.11 ± 0.81
CoOp	ViT L14	4	✓	0	0.1	79.05 ± 0.68	57.19 ± 0.26
CoOp	ViT L14	2	✓	0	0.1	78.53 ± 0.69	40.59 ± 0.68
CoOp	ViT L14	1	✓	0	0.1	76.48 ± 0.52	45.11 ± 0.68
CoOp	ViT L14	16	✓	0.5	0.1	80.95 ± 0.38	68.92 ± 0.77
CoOp	ViT L14	8	✓	0.5	0.1	79.87 ± 0.11	64.05 ± 0.57
CoOp	ViT L14	4	✓	0.5	0.1	79.18 ± 0.51	51.88 ± 0.45
CoOp	ViT L14	2	✓	0.5	0.1	76.76 ± 0.24	51.96 ± 0.06
CoOp	ViT L14	1	✓	0.5	0.1	73.89 ± 0.62	50.31 ± 1.02
CoOp	ViT L14	16	✓	1	0.1	80.45 ± 0.90	70.15 ± 0.98
CoOp	ViT L14	8	✓	1	0.1	79.25 ± 0.93	65.75 ± 0.69
CoOp	ViT L14	4	✓	1	0.1	78.37 ± 0.13	47.30 ± 0.20
CoOp	ViT L14	2	✓	1	0.1	75.21 ± 0.33	54.78 ± 0.66
CoOp	ViT L14	1	✓	1	0.1	74.93 ± 0.15	52.08 ± 1.04

Table 8. Performance of few-shot CoOp on Cifar100 under both ViT L14. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
MaPLe	ViT B16	16		N/A	N/A	75.01 ± 0.37	17.54 ± 0.83
MaPLe	ViT B16	8		N/A	N/A	73.93 ± 0.46	9.44 ± 1.13
MaPLe	ViT B16	4		N/A	N/A	72.68 ± 0.47	20.29 ± 1.07
MaPLe	ViT B16	2		N/A	N/A	71.37 ± 1.39	12.15 ± 0.25
MaPLe	ViT B16	1		N/A	N/A	68.75 ± 0.96	4.65 ± 1.52
MaPLe	ViT B16	16	✓	0	0.1	75.82 ± 0.10	58.63 ± 0.43
MaPLe	ViT B16	8	✓	0	0.1	74.29 ± 0.91	57.31 ± 0.79
MaPLe	ViT B16	4	✓	0	0.1	72.92 ± 0.42	54.12 ± 1.56
MaPLe	ViT B16	2	✓	0	0.1	71.09 ± 1.35	47.78 ± 0.64
MaPLe	ViT B16	1	✓	0	0.1	68.32 ± 0.20	39.43 ± 0.25
MaPLe	ViT B16	16	✓	0.5	0.1	75.34 ± 0.39	61.15 ± 0.53
MaPLe	ViT B16	8	✓	0.5	0.1	74.30 ± 0.29	60.24 ± 0.82
MaPLe	ViT B16	4	✓	0.5	0.1	71.35 ± 0.61	56.03 ± 0.35
MaPLe	ViT B16	2	✓	0.5	0.1	70.24 ± 1.01	52.56 ± 0.48
MaPLe	ViT B16	1	✓	0.5	0.1	69.33 ± 0.81	48.10 ± 0.26
MaPLe	ViT B16	16	✓	1	0.1	76.30 ± 0.56	62.04 ± 0.97
MaPLe	ViT B16	8	✓	1	0.1	73.60 ± 0.69	61.20 ± 0.77
MaPLe	ViT B16	4	✓	1	0.1	72.06 ± 0.34	56.51 ± 1.24
MaPLe	ViT B16	2	✓	1	0.1	69.95 ± 1.30	53.53 ± 0.67
MaPLe	ViT B16	1	✓	1	0.1	70.44 ± 0.10	46.94 ± 0.85

Table 9. Performance of few-shot MaPLe on Cifar100. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	MTA (25)	MTA (100)
CoOp	ViT B32	16		N/A	N/A	52.33	54.58
CoOp	ViT B32	8		N/A	N/A	46.09	47.20
CoOp	ViT B32	4		N/A	N/A	53.35	54.30
CoOp	ViT B32	2		N/A	N/A	53.13	53.81
CoOp	ViT B32	1		N/A	N/A	38.80	40.16
CoOp	ViT B32	16	✓	0.5	0.1	86.26	85.73
CoOp	ViT B32	8	✓	0.5	0.1	85.05	84.57
CoOp	ViT B32	4	✓	0.5	0.1	81.01	80.61
CoOp	ViT B32	2	✓	0.5	0.1	79.95	79.98
CoOp	ViT B32	1	✓	0.5	0.1	78.08	76.95
CoOp	ViT B16	16		N/A	N/A	50.64	51.14
CoOp	ViT B16	8		N/A	N/A	47.95	50.41
CoOp	ViT B16	4		N/A	N/A	43.77	46.29
CoOp	ViT B16	2		N/A	N/A	40.81	42.95
CoOp	ViT B16	1		N/A	N/A	41.78	44.17
CoOp	ViT B16	16	✓	0.5	0.1	87.69	87.30
CoOp	ViT B16	8	✓	0.5	0.1	86.28	86.01
CoOp	ViT B16	4	✓	0.5	0.1	84.52	83.79
CoOp	ViT B16	2	✓	0.5	0.1	83.49	83.18
CoOp	ViT B16	1	✓	0.5	0.1	81.64	81.01
CoOp	ViT L14	16		N/A	N/A	58.81	60.89
CoOp	ViT L14	8		N/A	N/A	40.49	43.20
CoOp	ViT L14	4		N/A	N/A	44.71	47.39
CoOp	ViT L14	2		N/A	N/A	39.44	43.22
CoOp	ViT L14	1		N/A	N/A	52.32	54.90
CoOp	ViT L14	16	✓	0.5	0.1	90.83	90.48
CoOp	ViT L14	8	✓	0.5	0.1	89.39	89.16
CoOp	ViT L14	4	✓	0.5	0.1	84.48	84.79
CoOp	ViT L14	2	✓	0.5	0.1	85.57	85.29
CoOp	ViT L14	1	✓	0.5	0.1	83.65	83.52
MaPLe	ViT B16	16		N/A	N/A	52.21	50.82
MaPLe	ViT B16	8		N/A	N/A	58.56	61.48
MaPLe	ViT B16	4		N/A	N/A	66.14	67.06
MaPLe	ViT B16	2		N/A	N/A	55.98	57.59
MaPLe	ViT B16	1		N/A	N/A	50.60	54.99
MaPLe	ViT B16	16	✓	0.5	0.1	88.04	88.33
MaPLe	ViT B16	8	✓	0.5	0.1	87.65	88.13
MaPLe	ViT B16	4	✓	0.5	0.1	86.72	87.04
MaPLe	ViT B16	2	✓	0.5	0.1	85.03	85.39
MaPLe	ViT B16	1	✓	0.5	0.1	83.36	83.78

Table 10. Performance of MTA for both few-shot CoOp and MaPLe on Cifar100. 25 ($T = 5$) and 100 ($T = 20$) treecuts are sampled for MTA evaluation.

Method	Encoder	K-shot	w/ ProTeCt	Acc_{leaf}	HCA	MTA (25)
CoOp	ResNet-50	16		52.61	5.72	41.97
CoOp	ResNet-50	16	✓	52.83	33.34	79.06
CoOp	ResNet-101	16		56.97	5.58	53.43
CoOp	ResNet-101	16	✓	57.64	39.93	81.76

Table 11. CoOp 16-shot results on Cifar100 with ResNets.

12. Complete Table of SUN Experiments

In this section, we report the complete experiment result conducted on SUN. Table 12 and Table 13 show the results of vanilla CoOp and MaPLe, and their results after adding ProTeCt. When comparing the HCA results of the vanilla prompt tuning with that of Cifar100 and ImageNet, the HCA result on SUN is much higher and the gap between HCA and Acc_{leaf} is much smaller. This is due to the shallow hierarchy of SUN dataset, indicating SUN is a much simpler dataset for hierarchical classification. However, we still see that ProTeCt achieves consistent improvement over the vanilla prompt tuning methods. Table 14 further compares the MTA result of vanilla CoOp and MaPLe, and their ProTeCt counterpart.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B16	16		N/A	N/A	73.82 ± 0.12	38.28 ± 0.46
CoOp	ViT B16	8		N/A	N/A	71.77 ± 0.67	33.95 ± 0.08
CoOp	ViT B16	4		N/A	N/A	69.31 ± 0.51	30.51 ± 0.71
CoOp	ViT B16	2		N/A	N/A	66.34 ± 0.33	36.85 ± 0.67
CoOp	ViT B16	1		N/A	N/A	63.65 ± 1.42	33.36 ± 0.21
CoOp	ViT B16	16	✓	0	0.1	74.95 ± 0.69	60.95 ± 0.91
CoOp	ViT B16	8	✓	0	0.1	72.31 ± 0.18	57.61 ± 1.31
CoOp	ViT B16	4	✓	0	0.1	69.53 ± 0.77	54.79 ± 0.12
CoOp	ViT B16	2	✓	0	0.1	67.01 ± 1.10	50.78 ± 0.03
CoOp	ViT B16	1	✓	0	0.1	64.45 ± 0.96	47.75 ± 0.11
CoOp	ViT B16	8	✓	0.5	0.1	74.59 ± 0.41	62.94 ± 0.15
CoOp	ViT B16	4	✓	0.5	0.1	71.53 ± 0.67	58.17 ± 0.33
CoOp	ViT B16	2	✓	0.5	0.1	69.80 ± 0.98	56.85 ± 0.41
CoOp	ViT B16	16	✓	0.5	0.1	67.29 ± 1.32	51.82 ± 1.20
CoOp	ViT B16	1	✓	0.5	0.1	63.79 ± 1.16	49.62 ± 1.40
CoOp	ViT B16	16	✓	1	0.1	74.31 ± 0.23	62.96 ± 0.61
CoOp	ViT B16	8	✓	1	0.1	71.27 ± 0.42	58.74 ± 0.98
CoOp	ViT B16	4	✓	1	0.1	68.81 ± 0.71	55.90 ± 0.09
CoOp	ViT B16	2	✓	1	0.1	67.66 ± 0.51	50.94 ± 1.31
CoOp	ViT B16	1	✓	1	0.1	63.95 ± 1.19	50.99 ± 1.21

Table 12. Performance of few-shot CoOp on SUN. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
MaPLe	ViT B16	16		N/A	N/A	71.86 ± 0.11	33.25 ± 1.31
MaPLe	ViT B16	8		N/A	N/A	68.96 ± 0.51	29.63 ± 0.19
MaPLe	ViT B16	4		N/A	N/A	67.27 ± 0.45	25.97 ± 0.53
MaPLe	ViT B16	2		N/A	N/A	65.33 ± 1.21	29.79 ± 0.13
MaPLe	ViT B16	1		N/A	N/A	63.98 ± 0.99	25.15 ± 0.76
MaPLe	ViT B16	16	✓	0	0.1	72.89 ± 0.77	56.52 ± 0.88
MaPLe	ViT B16	8	✓	0	0.1	71.24 ± 0.76	55.49 ± 1.05
MaPLe	ViT B16	4	✓	0	0.1	69.24 ± 0.41	51.88 ± 1.22
MaPLe	ViT B16	2	✓	0	0.1	66.98 ± 0.44	51.60 ± 0.55
MaPLe	ViT B16	1	✓	0	0.1	63.80 ± 1.51	47.93 ± 0.31
MaPLe	ViT B16	16	✓	0.5	0.1	72.17 ± 1.20	59.71 ± 0.04
MaPLe	ViT B16	8	✓	0.5	0.1	71.04 ± 0.09	57.78 ± 1.22
MaPLe	ViT B16	4	✓	0.5	0.1	68.64 ± 0.61	54.86 ± 1.08
MaPLe	ViT B16	2	✓	0.5	0.1	66.37 ± 0.62	53.13 ± 0.39
MaPLe	ViT B16	1	✓	0.5	0.1	64.29 ± 1.23	50.45 ± 0.40
MaPLe	ViT B16	16	✓	1	0.1	71.03 ± 0.99	59.92 ± 0.06
MaPLe	ViT B16	8	✓	1	0.1	69.66 ± 0.16	57.60 ± 0.81
MaPLe	ViT B16	4	✓	1	0.1	66.96 ± 0.31	53.61 ± 0.55
MaPLe	ViT B16	2	✓	1	0.1	66.74 ± 0.36	53.54 ± 0.76
MaPLe	ViT B16	1	✓	1	0.1	63.46 ± 0.14	50.49 ± 1.01

Table 13. Performance of few-shot MaPLe on SUN. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	MTA
CoOp	ViT B16	16		N/A	N/A	52.99
CoOp	ViT B16	8		N/A	N/A	55.24
CoOp	ViT B16	4		N/A	N/A	49.48
CoOp	ViT B16	2		N/A	N/A	51.94
CoOp	ViT B16	1		N/A	N/A	51.20
CoOp	ViT B16	16	✓	0.5	0.1	83.51
CoOp	ViT B16	8	✓	0.5	0.1	81.34
CoOp	ViT B16	4	✓	0.5	0.1	80.30
CoOp	ViT B16	2	✓	0.5	0.1	76.59
CoOp	ViT B16	1	✓	0.5	0.1	76.25
MaPLe	ViT B16	16		N/A	N/A	54.29
MaPLe	ViT B16	8		N/A	N/A	53.24
MaPLe	ViT B16	4		N/A	N/A	55.79
MaPLe	ViT B16	2		N/A	N/A	51.30
MaPLe	ViT B16	1		N/A	N/A	50.31
MaPLe	ViT B16	16	✓	0.5	0.1	82.27
MaPLe	ViT B16	8	✓	0.5	0.1	80.71
MaPLe	ViT B16	4	✓	0.5	0.1	79.10
MaPLe	ViT B16	2	✓	0.5	0.1	77.55
MaPLe	ViT B16	1	✓	0.5	0.1	76.73

Table 14. Performance of MTA for both few-shot CoOp and MaPLe on Sun.

13. Complete Table of ImageNet Experiments

In this section, we report the complete experiment result conducted on ImageNet. Table 15 first show the performance of CLIP and CoCoOp as a complement of Table 1 in the main paper. Note that none of the CLIP features (e.g. ViT B32, ViT B16, RN50, RN101) nor existing prompt tuning methods help the HCA metric. Table 16 and Table 17 show the results of vanilla CoOp and MaPLe, and their results after adding ProTeCt. Table 18 further compares the MTA result of vanilla CoOp and MaPLe, and their ProTeCt counterpart. Clearly, adding ProTeCt boosts both HCA and MTA. Furthermore, we apply the model trained on ImageNet to its four variants. Table 19, Table 20, Table 21 and Table 22 report the domain generalization results on ImageNetV2, ImageNet-sketch, ImageNet-A and ImageNet-R datasets for Acc_{leaf}, HCA and MTA. All four tables show that ProTeCt can not only improves the hierarchical consistency on the seen dataset, but also unseen datasets from other image domains.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CLIP	ViT-B32	0		N/A	N/A	63.31	4.29
CLIP	ViT-B16	0		N/A	N/A	68.36	3.32
CLIP	RN50	0		N/A	N/A	59.81	4.16
CLIP	RN101	0		N/A	N/A	62.30	2.03
CoCoOp	ViT-B16	16		N/A	N/A	71.20 ± 0.13	2.92 ± 1.23

Table 15. Performance of zero-shot CLIP and 16-shot CoCoOp on ImageNet.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B16	16		N/A	N/A	71.23 ± 0.67	2.99 ± 1.04
CoOp	ViT B16	8		N/A	N/A	69.40 ± 0.52	3.00 ± 0.58
CoOp	ViT B16	4		N/A	N/A	68.06 ± 0.42	2.95 ± 0.62
CoOp	ViT B16	2		N/A	N/A	65.46 ± 0.77	1.56 ± 0.17
CoOp	ViT B16	1		N/A	N/A	63.67 ± 0.85	1.59 ± 0.43
CoOp	ViT B16	16	✓	0	0.1	70.47 ± 0.22	27.81 ± 0.71
CoOp	ViT B16	8	✓	0	0.1	70.03 ± 0.14	26.17 ± 0.52
CoOp	ViT B16	4	✓	0	0.1	69.32 ± 0.11	21.99 ± 0.10
CoOp	ViT B16	2	✓	0	0.1	68.09 ± 0.23	20.92 ± 1.02
CoOp	ViT B16	1	✓	0	0.1	67.26 ± 0.65	18.69 ± 1.12
CoOp	ViT B16	8	✓	0.5	0.1	70.27 ± 0.36	34.63 ± 0.33
CoOp	ViT B16	4	✓	0.5	0.1	69.65 ± 0.41	31.84 ± 0.35
CoOp	ViT B16	2	✓	0.5	0.1	68.09 ± 0.16	27.05 ± 0.27
CoOp	ViT B16	16	✓	0.5	0.1	67.24 ± 0.24	26.09 ± 0.53
CoOp	ViT B16	1	✓	0.5	0.1	66.69 ± 0.15	23.79 ± 0.15
CoOp	ViT B16	16	✓	1	0.1	69.92 ± 0.21	37.74 ± 0.12
CoOp	ViT B16	8	✓	1	0.1	69.34 ± 0.17	34.66 ± 0.55
CoOp	ViT B16	4	✓	1	0.1	68.06 ± 0.44	30.87 ± 0.32
CoOp	ViT B16	2	✓	1	0.1	67.12 ± 0.35	26.34 ± 1.1
CoOp	ViT B16	1	✓	1	0.1	66.11 ± 0.50	25.79 ± 0.06

Table 16. Performance of few-shot CoOp on ImageNet. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
MaPLe	ViT B16	16		N/A	N/A	70.70 ± 0.11	4.15 ± 1.05
MaPLe	ViT B16	8		N/A	N/A	70.44 ± 0.06	4.32 ± 0.90
MaPLe	ViT B16	4		N/A	N/A	70.20 ± 0.06	2.95 ± 0.87
MaPLe	ViT B16	2		N/A	N/A	69.74 ± 0.25	4.27 ± 1.32
MaPLe	ViT B16	1		N/A	N/A	68.91 ± 0.13	2.97 ± 1.08
MaPLe	ViT B16	16	✓	0	0.1	70.08 ± 0.26	23.38 ± 1.43
MaPLe	ViT B16	8	✓	0	0.1	69.00 ± 0.26	21.71 ± 0.64
MaPLe	ViT B16	4	✓	0	0.1	68.50 ± 0.41	19.03 ± 0.21
MaPLe	ViT B16	2	✓	0	0.1	67.45 ± 0.32	17.54 ± 0.52
MaPLe	ViT B16	1	✓	0	0.1	67.03 ± 0.11	16.54 ± 0.32
MaPLe	ViT B16	16	✓	0.5	0.1	69.59 ± 0.25	27.74 ± 1.31
MaPLe	ViT B16	8	✓	0.5	0.1	69.06 ± 0.49	25.25 ± 0.52
MaPLe	ViT B16	4	✓	0.5	0.1	68.13 ± 0.01	25.25 ± 0.12
MaPLe	ViT B16	2	✓	0.5	0.1	67.45 ± 0.43	20.14 ± 1.07
MaPLe	ViT B16	1	✓	0.5	0.1	66.80 ± 0.26	20.62 ± 0.65
MaPLe	ViT B16	16	✓	1	0.1	69.52 ± 0.71	31.24 ± 1.02
MaPLe	ViT B16	8	✓	1	0.1	68.48 ± 0.06	26.92 ± 0.42
MaPLe	ViT B16	4	✓	1	0.1	68.59 ± 0.17	26.28 ± 0.31
MaPLe	ViT B16	2	✓	1	0.1	67.12 ± 0.11	22.96 ± 0.05
MaPLe	ViT B16	1	✓	1	0.1	66.16 ± 0.88	20.44 ± 0.77

Table 17. Performance of few-shot MaPLe on ImageNet. Ablations cover different NCL strengths λ.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	MTA
CoOp	ViT B16	16		N/A	N/A	46.98
CoOp	ViT B16	8		N/A	N/A	46.04
CoOp	ViT B16	4		N/A	N/A	42.57
CoOp	ViT B16	2		N/A	N/A	44.89
CoOp	ViT B16	1		N/A	N/A	40.52
CoOp	ViT B16	16	✓	0.5	0.1	88.61
CoOp	ViT B16	8	✓	0.5	0.1	87.86
CoOp	ViT B16	4	✓	0.5	0.1	87.37
CoOp	ViT B16	2	✓	0.5	0.1	86.14
CoOp	ViT B16	1	✓	0.5	0.1	86.14
MaPLe	ViT B16	16		N/A	N/A	48.29
MaPLe	ViT B16	8		N/A	N/A	45.84
MaPLe	ViT B16	4		N/A	N/A	51.84
MaPLe	ViT B16	2		N/A	N/A	48.17
MaPLe	ViT B16	1		N/A	N/A	48.16
MaPLe	ViT B16	16	✓	0.5	0.1	87.87
MaPLe	ViT B16	8	✓	0.5	0.1	87.26
MaPLe	ViT B16	4	✓	0.5	0.1	86.85
MaPLe	ViT B16	2	✓	0.5	0.1	85.93
MaPLe	ViT B16	1	✓	0.5	0.1	85.18

Table 18. Performance of MTA for both few-shot CoOp and MaPLe on ImageNet.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA	MTA
CoOp	ViT B16	16		N/A	N/A	64.01	2.31	43.74
CoOp	ViT B16	8		N/A	N/A	62.20	2.62	43.30
CoOp	ViT B16	4		N/A	N/A	61.51	2.48	40.68
CoOp	ViT B16	2		N/A	N/A	58.68	1.35	42.84
CoOp	ViT B16	1		N/A	N/A	56.43	1.51	38.27
CoOp	ViT B16	16	✓	1	0.1	62.60	32.84	86.66
CoOp	ViT B16	8	✓	1	0.1	62.15	30.65	85.84
CoOp	ViT B16	4	✓	1	0.1	61.24	26.85	85.52
CoOp	ViT B16	2	✓	1	0.1	60.42	23.22	84.38
CoOp	ViT B16	1	✓	1	0.1	60.16	22.95	84.38
MaPLe	ViT B16	16		N/A	N/A	64.15	1.97	45.93
MaPLe	ViT B16	8		N/A	N/A	62.76	1.99	43.98
MaPLe	ViT B16	4		N/A	N/A	63.45	2.51	49.41
MaPLe	ViT B16	2		N/A	N/A	61.75	2.81	45.92
MaPLe	ViT B16	1		N/A	N/A	61.78	2.18	45.50
MaPLe	ViT B16	16	✓	1	0.1	62.77	27.86	86.14
MaPLe	ViT B16	8	✓	1	0.1	61.42	23.45	85.51
MaPLe	ViT B16	4	✓	1	0.1	61.89	22.92	85.17
MaPLe	ViT B16	2	✓	1	0.1	60.43	20.10	84.23
MaPLe	ViT B16	1	✓	1	0.1	59.14	17.89	83.27

Table 19. Domain generalization on ImageNetv2 dataset using CoOp and MaPLe.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA	MTA
CoOp	ViT B16	16		N/A	N/A	47.82	1.39	38.58
CoOp	ViT B16	8		N/A	N/A	45.93	2.10	42.56
CoOp	ViT B16	4		N/A	N/A	44.60	1.41	36.52
CoOp	ViT B16	2		N/A	N/A	42.17	0.96	36.01
CoOp	ViT B16	1		N/A	N/A	41.38	1.11	33.61
CoOp	ViT B16	16	✓	1	0.1	46.80	20.73	82.60
CoOp	ViT B16	8	✓	1	0.1	46.91	19.71	82.11
CoOp	ViT B16	4	✓	1	0.1	46.53	17.69	82.07
CoOp	ViT B16	2	✓	1	0.1	45.40	15.49	80.82
CoOp	ViT B16	1	✓	1	0.1	44.75	13.88	80.64
MaPLe	ViT B16	16		N/A	N/A	48.97	1.58	43.37
MaPLe	ViT B16	8		N/A	N/A	47.55	1.66	45.26
MaPLe	ViT B16	4		N/A	N/A	48.20	2.45	53.31
MaPLe	ViT B16	2		N/A	N/A	46.86	1.01	42.55
MaPLe	ViT B16	1		N/A	N/A	46.79	1.70	45.26
MaPLe	ViT B16	16	✓	1	0.1	47.47	17.77	82.52
MaPLe	ViT B16	8	✓	1	0.1	46.60	15.31	82.04
MaPLe	ViT B16	4	✓	1	0.1	47.23	14.95	81.67
MaPLe	ViT B16	2	✓	1	0.1	45.95	13.32	80.87
MaPLe	ViT B16	1	✓	1	0.1	44.92	11.24	79.94

Table 20. Domain generalization on ImageNet-sketch dataset using CoOp and MaPLe.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA	MTA
CoOp	ViT B16	16		N/A	N/A	50.28	2.97	52.56
CoOp	ViT B16	8		N/A	N/A	48.08	4.19	45.05
CoOp	ViT B16	4		N/A	N/A	48.43	2.97	41.20
CoOp	ViT B16	2		N/A	N/A	46.56	1.95	52.47
CoOp	ViT B16	1		N/A	N/A	45.92	1.76	47.54
CoOp	ViT B16	16	✓	1	0.1	49.08	22.45	78.21
CoOp	ViT B16	8	✓	1	0.1	49.29	24.00	79.47
CoOp	ViT B16	4	✓	1	0.1	48.39	18.11	76.95
CoOp	ViT B16	2	✓	1	0.1	48.81	20.00	78.11
CoOp	ViT B16	1	✓	1	0.1	48.95	20.52	76.95
MaPLe	ViT B16	16		N/A	N/A	50.61	2.31	54.88
MaPLe	ViT B16	8		N/A	N/A	48.41	5.31	55.97
MaPLe	ViT B16	4		N/A	N/A	50.23	4.95	57.07
MaPLe	ViT B16	2		N/A	N/A	48.49	9.80	59.90
MaPLe	ViT B16	1		N/A	N/A	47.55	3.52	55.48
MaPLe	ViT B16	16	✓	1	0.1	47.41	19.75	77.46
MaPLe	ViT B16	8	✓	1	0.1	46.15	16.49	75.88
MaPLe	ViT B16	4	✓	1	0.1	47.35	17.39	77.64
MaPLe	ViT B16	2	✓	1	0.1	49.15	16.23	77.71
MaPLe	ViT B16	1	✓	1	0.1	47.15	16.03	76.81

Table 21. Domain generalization on ImageNet-A dataset using CoOp and MaPLe.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA	MTA
CoOp	ViT B16	16		N/A	N/A	75.83	18.49	64.13
CoOp	ViT B16	8		N/A	N/A	74.79	5.91	49.56
CoOp	ViT B16	4		N/A	N/A	73.99	14.85	61.40
CoOp	ViT B16	2		N/A	N/A	70.94	16.32	56.67
CoOp	ViT B16	1		N/A	N/A	69.84	11.74	55.31
CoOp	ViT B16	16	✓	1	0.1	74.94	31.18	75.59
CoOp	ViT B16	8	✓	1	0.1	75.51	37.96	81.11
CoOp	ViT B16	4	✓	1	0.1	74.23	29.69	75.54
CoOp	ViT B16	2	✓	1	0.1	74.86	28.67	78.17
CoOp	ViT B16	1	✓	1	0.1	74.26	27.46	76.48
MaPLe	ViT B16	16		N/A	N/A	76.61	20.67	63.06
MaPLe	ViT B16	8		N/A	N/A	76.48	18.92	67.30
MaPLe	ViT B16	4		N/A	N/A	76.83	21.06	64.30
MaPLe	ViT B16	2		N/A	N/A	75.85	19.84	60.86
MaPLe	ViT B16	1		N/A	N/A	74.55	18.85	62.48
MaPLe	ViT B16	16	✓	1	0.1	75.70	32.58	77.99
MaPLe	ViT B16	8	✓	1	0.1	75.98	30.97	77.57
MaPLe	ViT B16	4	✓	1	0.1	76.31	29.28	78.52
MaPLe	ViT B16	2	✓	1	0.1	75.01	23.94	72.73
MaPLe	ViT B16	1	✓	1	0.1	74.60	25.20	75.72

Table 22. Domain generalization on ImageNet-R dataset using CoOp and MaPLe.

14. Additional Taxonomies

To investigate the robustness of ProTeCt across hierarchies, we consider the FGVC Aircraft [29] dataset and the RSICB [22] satellite dataset. These datasets have their built-in hierarchies, which beyond differing from those of SUN [42] and WordNet [11], are a technical hierarchy of fine-grained aircraft classes and satellite image classes, respectively. Table 23 and Table 24 summarize the CoOp results for these experiments, showing that ProTeCt improves performance under all metrics. This illustrates its taxonomy robustness.

K-shot	w/ ProTeCt	Acc_{leaf}	HCA	MTA (25)
16		41.88	17.82	21.11
16	✓	42.00	29.94	32.95
		(+0.12)	(+12.12)	(+11.84)
1		23.61	11.55	16.77
1	✓	27.30	16.47	24.67
		(+3.69)	(+4.92)	(+7.90)

Table 23. Comparison of CoOp with/without ProTeCt on FGVC Aircraft [29] dataset.

K-shot	w/ ProTeCt	Acc_{leaf}	HCA	MTA (25)
16		91.79	43.50	64.49
16	✓	93.21	85.21	91.44
		(+1.42)	(+41.71)	(+26.95)
1		63.93	32.29	52.17
1	✓	65.00	48.36	67.05
		(+1.07)	(+16.07)	(+14.88)

Table 24. Comparison of CoOp with/without ProTeCt on RSICB [22] satellite dataset.

15. Complete Ablation Results

This section complements Figure 4(a) and 4(b) in the main paper with the error bar. Table 25 and Table 26 show how the NCL strength λ and tree dropout rates β affect the Acc_{leaf} and HCA. Please refer to Section 6.3 of the main paper for more discussion.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B16	1	✓	0	0.1	64.77 ± 1.37	32.93 ± 0.42
CoOp	ViT B16	1	✓	0.1	0.1	66.18 ± 0.51	39.41 ± 0.78
CoOp	ViT B16	1	✓	0.3	0.1	67.13 ± 0.66	40.82 ± 0.33
CoOp	ViT B16	1	✓	0.5	0.1	66.88 ± 0.21	41.01 ± 1.18
CoOp	ViT B16	1	✓	0.7	0.1	66.50 ± 0.71	40.39 ± 0.37
CoOp	ViT B16	1	✓	1	0.1	63.84 ± 1.51	40.05 ± 1.48

Table 25. Ablation of different NCL strength λ on Cifar100 using CoOp 1 shot setting.

Method	Encoder	K-shot	w/ ProTeCt	λ	β	Leaf Acc.	HCA
CoOp	ViT B16	16	✓	1	0	70.86 ± 0.59	54.39 ± 0.68
CoOp	ViT B16	16	✓	1	0.1	73.26 ± 0.66	58.01 ± 0.43
CoOp	ViT B16	16	✓	1	0.2	72.48 ± 0.57	59.32 ± 0.21
CoOp	ViT B16	16	✓	1	0.3	71.49 ± 0.36	58.82 ± 0.12
CoOp	ViT B16	16	✓	1	0.4	70.15 ± 0.75	57.93 ± 0.50
CoOp	ViT B16	16	✓	1	0.5	69.22 ± 0.32	56.66 ± 0.41
CoOp	ViT B16	16	✓	1	0.6	68.35 ± 0.78	53.75 ± 0.85
CoOp	ViT B16	16	✓	1	0.7	66.58 ± 0.45	53.27 ± 0.42
CoOp	ViT B16	16	✓	1	0.8	66.62 ± 0.38	50.74 ± 1.05
CoOp	ViT B16	16	✓	1	0.9	66.77 ± 0.38	49.76 ± 1.02

Table 26. Ablation of different tree dropout rates β on Cifar100 using CoOp 16 shot setting.

16. Visualizations

This section illustrates some misclassified examples of prior prompt tuning methods in ImageNet and its variants (i.e. ImageNetv2, ImageNet-S, ImageNet-A, ImageNet-R). Note that the hierarchy of these variants may differ from the one of ImageNet. The misclassification can occur in both coarse or fine-grained levels of the hierarchy. Note that ProTeCt can successfully classify all the illustrated examples at **every** hierarchy level in the examples shown in Figure 9-13. Figure 9 presents the correct/incorrect predictions of CoOp and its ProTeCt counterpart at multiple tree levels on ImageNet. CoOp [48] fails to generate consistent predictions at different hierarchy levels, and even predicts incorrectly at coarser hierarchy levels when the predictions at the leaf level are correct. More examples of the predictions on ImageNet variants are shown in Figure 10-13, where [GT, Prediction] shows the groundtruth and incorrect prediction by vanilla prompt tuning.

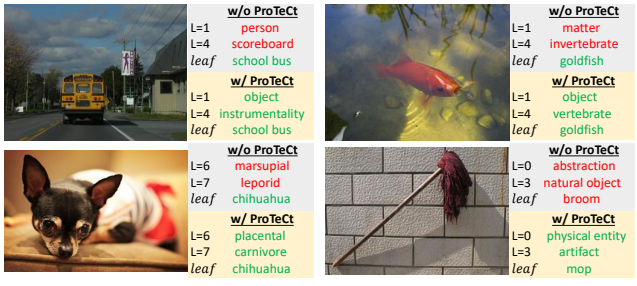


Figure 9. ImageNet visual examples at multiple hierarchy levels. Correct/incorrect model predictions (green/red) of CoOp w/ and w/o ProTeCt, respectively. L denotes the tree level.

(a) (b) (c)

Figure 10. ImageNetv2 visual examples: (a): [Taxicab, Teddy bear], (b): [Washing machine, Bath towel], (c):[Grey fox, Marsupial].

(a) (b) (c) (d)

Figure 11. ImageNet-S visual examples: (a): [Water bottle, Soap dispenser], (b): [Umbrella, Lampshade], (c):[Folding chair, Baby bed], (d):[Pembroke Welsh Corgi, Marsupial].

(a) (b) (c)

Figure 12. ImageNet-A visual examples: (a): [Chihuahua, Cottontail rabbit], (b): [Fox squirrel, Bird], (c):[American black bear, Koala].

(a) (b) (c)

Figure 13. ImageNet-R visual examples: (a): [Killer whale, Person], (b): [Wine bottle, Fruit], (c):[Cheeseburger, Ice cream].