Pro-Tuning: Unified Prompt Tuning for Vision Tasks

Xing Nie, Bolin Ni, Jianlong Chang, Gaofeng Meng, Senior Member, IEEE, Chunlei Huo, Member, IEEE, Shiming Xiang, and Qi Tian, Fellow, IEEE

Abstract—In computer vision, fine-tuning is the de-facto approach to leverage pre-trained vision models to perform downstream tasks. However, deploying it in practice is quite challenging, due to adopting parameter inefficient global update and heavily relying on high-quality downstream data. Recently, prompt-based learning, which adds the task-relevant prompt to adapt the pre-trained models to downstream tasks, has drastically boosted the performance of many natural language downstream tasks. In this work, we extend this notable transfer ability benefited from prompt into vision models as an alternative to fine-tuning. To this end, we propose parameter-efficient Prompt tuning (Pro-tuning) to adapt diverse frozen pre-trained models to a wide variety of downstream vision tasks. The key to Pro-tuning is prompt-based tuning, i.e., learning task-specific vision prompts for downstream input images with the pre-trained model frozen. By only training a small number of additional parameters, Pro-tuning can generate compact and robust downstream models both for CNN-based and transformer-based network architectures. Comprehensive experiments evidence that the proposed Pro-tuning outperforms fine-tuning on a broad range of vision tasks and scenarios, including image classification (under generic objects, class imbalance, image corruption, natural adversarial examples, and out-of-distribution generalization), and dense prediction tasks such as object detection and semantic segmentation.

Index Terms—Prompt-based learning, representation learning, task-specific knowledge, transfer learning.

I. INTRODUCTION

FINE-TUNING is the dominant technique for using pre-trained vision models to adapt to downstream tasks, which has contributed to significant advances in various computer vision problems [1], [2], [3], [4], [5]. Its standard practice is first to pre-train a general-purpose model on a large-scale dataset (e.g., ImageNet-1K [6]), then fine-tune the entire pre-trained model on downstream tasks. Although the achievements in the literature are brilliant, the fine-tuned model is still quite intractable to handle practical downstream vision applications for two main reasons. First, it requires updating and storing all model parameters separately for each downstream task, which is prohibitively expensive for the current ever-increasing vision model capacity. Second, its effectiveness heavily relies on high-quality downstream data, however, realistic data encountered in practical scenarios probably contains various negative perturbations. In Fig. 1, we provide an empirical study of fine-tuning versus other popular tuning strategies. Though achieving decent accuracy on CIFAR-100 [7], fine-tuning encounters devastating performance damages on a wide variety of pre-trained vision models under distribution perturbations, e.g., 17.7%–25.2% accuracy drops on long-tailed CIFAR-100 [8] compared with its accuracy on original CIFAR-100.

Recently, prompt-based learning has made waves in the natural language processing (NLP) community by demonstrating astounding transfer performance on myriad downstream tasks [9], [10], [11], [12], [13]. Typically, prompt is a task-relevant description prepended to the downstream input to induce the downstream task to the pre-trained model. Its key idea is to reformulate the downstream task aided by an appropriate prompt design, making it close to those solved during the original pre-training, rather than fine-tuning pre-trained models to adapt to downstream tasks. Following this ideology, vision-language pre-training [14], [15], [16], [17] has been gradually developed to put visual categories into a text input as prompt, in order to diamentrically generate visual concepts from natural language. Though achieving remarkable performance on many downstream vision tasks even without fine-tuning the language model, their prompt tuning strategies are tailored for the multimodal model, inapplicable to the pre-trained vision model. A question naturally arises: can we devise a lightweight prompt tuning paradigm specifically for the pre-trained vision models to boost downstream vision tasks?

In this spirit, we introduce Pro-tuning, a unified and parameter-efficient tuning paradigm that adapts frozen vision
As a plug-and-play technique, it is a capable and unified tuning framework with minor additional parameters.

- Extensive experiments demonstrate that Pro-tuning exceeds fine-tuning on fifteen challenging benchmarks across three vision tasks, including image classification under five scenarios, and dense prediction tasks such as object detection and semantic segmentation.

II. RELATED WORK

A. Model Fine-Tuning

In computer vision, a series of techniques have been proposed to adapt vision models to downstream tasks, where canonical examples include partial fine-tuning [20], bias tuning [21], and Side-tuning [22]. Among them, ATL [23] introduces the orthogonal projection matrix and the weight coefficient vector to reduce the discrepancy between source and target domains. MTLSC [24] develops a model-based transfer strategy with rectification-based sparse coding for partial face recognition. FSPI [25] firstly leverages privileged information to achieve multi-source transfer to mitigate the data sparsity of feature selection in the target domain. LT [26] encourages vision foundation models to combine capabilities learned from different loss functions by transferring the knowledge from one loss function to another. Despite the promising performances, most dominant techniques are based on the moderately-sized CNNs with high-quality downstream data and require update the total parameters of vision foundation models. On the other hand, parameter-efficient fine-tuning techniques have enabled remarkable progress for a variety of NLP tasks [9], [13], [27], [28]. Among them, PET [13] introduces a set of learnable tokens to reformulate input examples as cloze-style phrases to help language models handle few-shot text classification and natural language inference. Adapter [27] inserts lightweight linear layers into layers of a language model for text classification and question answering tasks. Notably, the flagship systems like GPT-3 [9] (composed of 175 billion parameters) achieve remarkable performance across a wide range of downstream tasks even without requiring additional task-specific parameters. Recent studies [15], [29], [30], [31], [32], [33] empirically validate that larger pre-trained models have a tendency to achieve better transfer performance, thus an important ingredient in leading to the difference in fine-tuning methods between computer vision and NLP could be the evident increasing scale of pre-trained language models. Inspired by fine-tuning techniques in NLP, we develop a conceptually simple yet effective tuning paradigm for diverse downstream vision tasks with a broad distribution under various vision models, including CNNs and vision transformers. Different from existing fine-tuning methods for NLP tasks, we focus on efficiently modeling the 2D spatial layout of images, which is proved to be a critical capability for vision understanding to adapt pre-trained models to multiple downstream vision tasks. Our work is to explore how far we can push without the advantages of large-scale pre-trained models.

B. Prompt-Based Learning

Prompt-based learning is proposed and fueled by the GPT series [9], [11], [12] in the field of NLP. It originally refers to models to a broad distribution of downstream tasks with robustness. The key to Pro-tuning is prompt-based tuning, i.e., only learning task-specific vision prompts for downstream input images while freezing the pre-trained vision model. Specifically, Pro-tuning constructs lightweight prompt prompt blocks to generate task-specific discriminative prompts for each downstream input image. By blending intermediate feature maps of each input with the learned prompts, Pro-tuning can generate a compact and robust downstream model to adapt tuning demands on various vision tasks, while only training lightweight additional parameters per downstream task.

We demonstrate that Pro-tuning is a generic tuning framework for convolutional neural networks (CNNs) and vision transformers, which can surpass fine-tuning on various vision tasks of object detection, semantic segmentation, and image classification under diverse practical scenarios, including generic objects, class imbalance, image corruption, natural adversarial examples, and out-of-distribution generalization. Fig. 1 illustrates the results of Pro-tuning and other tuning paradigms on two classification datasets. In particular, on the original CIFAR-100 [7], Pro-tuning achieves superior performance compared with fine-tuning under DeiT-S [18], which brings +0.5% accuracy gain with only 25.5x fewer trainable parameters. On the more challenging long-tailed CIFAR-100 with the imbalance ratio of 100 [8], Pro-tuning brings significant accuracy gains by +2.1% and +0.9% over fine-tuning under ResNet-101 [19] and DeiT-B [18], which reduces the trainable parameters by up to 11x and 27x, respectively.

The key contributions can be summarized as follow:

- A parameter-efficient vision tuning paradigm, i.e., Pro-tuning, is proposed. This prompt-based tuning specifically for vision models can adapt frozen pre-trained models to diverse downstream tasks with a broad distribution.
- The proposed Pro-tuning can cooperate with various vision models including CNNs and vision transformers.
adding the task-relevant description to the downstream input to help a language model tackle downstream tasks, rather than only adapting the pre-trained model to downstream tasks by fine-tuning. GPT-3 pioneers this route by treating each downstream task as a masked language modeling problem, where the model directly generates a textual response in the prompt. Subsequently, numerous studies are devoted to devising efficient prompt strategies for mining knowledge from pre-trained models [10, 25, 34, 35, 36, 37]. For example, Prefix-Tuning [35] directly prepends a sequence of continuous task-specific vectors to the input as an upstream prefix for steering the downstream language model via gradient optimization. LoRA [37] injects trainable rank decomposition matrices into each layer of the transformer-based model to train some dense layers indirectly, which can greatly reduce the trainable parameters for downstream tasks. Motivated by the success in NLP, recent vision-language models such as CLIP [15] and BLIP [17] diametrically generate vision concepts from natural language by training a large contrastive learning model over a significant amount of image-text pairs, where the key is to place visual categories into a text input as prompt. For example, SubPT [38] eliminates the overfitting problem in prompt tuning of vision-language models and enhances the generalization ability of prompts towards novel categories. These methods achieve impressive performance on a wide range of vision tasks without any fine-tuning. However, prompt-based tuning methods in vision-language models are tailored for the multimodal model, and inapplicable to the pre-trained vision model. To address this issue, DAM-VP [39] clusters downstream datasets into multiple small homogeneity subsets to reduce the optimization difficulty and improve the downstream classification performance. VPT [40] follows the technical line of prompt tuning [34] in NLP, thus it concatenates several trainable tokens with the original image tokens as prompts, specifically for the frozen transformer-based vision model. AdaptFormer [41] inserts linear layers into the feed-forward network of a vision transformer model for improving transfer performance on image tasks, similar to Adapter [27] in NLP. Different from these methods, our work aims to develop a generally applicable prompt tuning paradigm for various vision foundation models, while covering a diverse set of visual recognition tasks.

C. Other Related Topic

1) Continual Learning: Continual Learning aims to foster the network to train on a sequence of tasks incrementally and perform well on all learned tasks to prevent forgetting. Series [42] and Parallel [43] insert $1 \times 1$ convolutional layers after each convolution of CNN-based vision foundation models. Though performing well in continual learning, they are not competitive enough in terms of training efficiency and performance in transfer learning. Experimental results show that our proposed Pro-tuning is more suitable for image transfer learning tasks, since it employs lightweight convolutional layers as prompt blocks, with enough spatial modeling capability and great computational efficiency.

III. PRO-TUNING

In this section, we describe the proposed Pro-tuning in terms of architecture and optimization. Our approach is generally applicable to diverse vision models, including CNN-based and transformer-based network architectures.

A. Overall Architecture

1) Basic Primitives: For vision systems, the ability to transfer a frozen pre-trained model to new tasks from practical downstream data even with negative perturbations, is a desirable and often key requirement. To achieve this, we develop a new unified vision prompt tuning paradigm, namely, Pro-tuning, to achieve explicit reasoning about 2D spatial relationships that is crucial for general-purpose visual modeling. Technically, we introduce lightweight prompt blocks to learn task-specific prompts for each downstream input, while freezing the pre-trained model. We follow three essential design principles – low parameter overheads, powerful representations, and deployment friendliness. That is, the extra learnable parameters should be sufficiently small for parameter efficiency while owing strong representation capability to help frozen models rapidly adapt to diverse new tasks. Additionally, it can be implemented with highly optimized functions and is hardware friendly, making the whole system easy to run on a wide variety of platforms and achieve actual speed-up.

Based on the above analysis, we realize Pro-tuning with commonly-used operation primitives that support well-optimized Cudnn functions. Concretely, we introduce a prompt block with three sequential lightweight convolutional layers: $1 \times 1$ convolution, $5 \times 5$ depthwise convolution [44], and $1 \times 1$ convolution. This design has two natural advantages in efficiently conducting spatial relationship reasoning. First, the depthwise convolution provides enough spatial modeling capability for guiding the deviated downstream data to adapt to the pre-trained model, while enjoying great computational efficiency. Second, it can perform the whole computation in a structured sparse way to keep high parameter efficiency, which can be achieved by group convolutions (see Section IV-B for details) to drastically reduce computational costs [45], [46].

2) Integrated Vision System: As pointed out in [20], [47], and [48], visual feature representations of different levels contribute to the generalization performance of the network, especially for low-level and mid-level representations. To sufficiently exploit semantic information from diverse levels to enrich the feature space, we build multiple stage-wise prompt blocks, which can be simply plugged into a given frozen pre-trained vision model for adapting to new tasks. Intuitively, Pro-tuning incorporates the prompt blocks into multiple stages of a frozen pre-trained model, as illustrated in Fig. 2. Notably, vision models typically divide network architectures into multiple sub-layers as stages. To ensure the generality of the proposed Pro-tuning, we follow their original concepts of stage partitioning, for either CNN-based or transformer-based models.

3) Modeling Framework: As a result, the whole computation can be described through four simple steps: 1) feeding each downstream input image into the frozen pre-trained vision model; 2) producing task-specific vision prompts by multiple stage-wise prompt blocks; 3) blending the learned vision prompt with the corresponding feature map as input to the next stage of the frozen pre-trained model; 4) appending a specialized output head to top layers of the pre-trained
model for the final prediction. In this way, the task-specific vision prompts can be obtained, maximally mitigating the gap between pre-training and solving downstream tasks.

Generally, a neural network $F$ consists of a backbone $G$ and a head $H$, denoted by $F = H \circ G$. Given the $n$-stage backbone $G = \{S_i\}^n_{i=1}$ and an input image $x^1$ on a downstream dataset $D = \{(x^j, y^j)\}^m_{j=1}$, we define the output of intermediate stages of $G(x^j)$ as $\{x^j_1, x^j_2, \cdots, x^j_n\}$. Suppose that the prompt block appended to the stage $S_i$ is denoted by $P_i$, the prompt-based blending representation $\tilde{x}^j_i$ is obtained by aggregating the output $x^j_i$ of the stage $S_i$ and the learned vision prompt $P_i(x^j_i)$. Specifically, this process can be computed as

$$\begin{align*}
x^j_i &= S_i(x^j) \\
\tilde{x}^j_i &= x^j_i + \beta \cdot P_i(x^j_i), \quad i = 1, \cdots, n \\
x^j_{i+1} &= S_{i+1}(\tilde{x}^j_i), \quad i = 1, \cdots, n - 1
\end{align*}$$

where $\beta$ denotes a learnable parameter to balance the two terms. In this way, task-specific knowledge can be sufficiently distilled from the pre-trained parameters and multi-level feature representations of downstream data. Note that, the prompt-based blending representation encourages a balance between the intermediate feature map and the learned prompt when pondering semantics of different stages, enabling adaptive control. Quite intuitively, Pro-tuning is a generic plug-and-play technique to re-purpose a frozen pre-trained vision model for rapidly adapting to a new downstream task via minor additional computational costs, without any architectural modification.

Particularly, since the input image is flattened into 1D token sequences by the embedding layer in partial transformer-based models (such as DeiT [18]), the output of each network stage is first reshaped into the 2D structure before feeding into the prompt block. After aggregating with the learned vision prompt, the prompt-based blending representation is reshaped back into 1D token embeddings as input for the next stage.

During training, we only update a few parameters $\theta$ with the pre-trained model parameters $\phi$ frozen. Training only the parameters $\theta$ makes our Pro-tuning modular – it can use an off-the-shelf pre-trained vision model while removing the need for modifying or retraining, merely adding a small number of additional parameters per task. This parameter sharing of frozen pre-trained models is significantly appealing in practical applications. A vivid example is that Pro-tuning can gear toward common application scenarios such as cloud services, where a diverse set of downstream tasks need to be solved by a given vision model. Formally, the objective of the proposed Pro-tuning can be formulated as

$$\theta^* = \arg \min_{\theta} \frac{1}{|D|} \sum_{j=1}^{|D|} \ell \left( H^\theta(\tilde{x}^j_i), y^j \right),$$

where $\tilde{x}^j_i = S^\phi_i(\tilde{x}^j_{i-1}) + \beta \cdot P^\theta_i \left( S^\phi_i(\tilde{x}^j_{i-1}) \right), \quad i = 2, \cdots, n$

$$\tilde{x}^j_i = S^\phi_i(x^j_i) + \beta \cdot P^\theta_i \left( S^\phi_i(x^j_i) \right). \quad (4)$$

During training, we simply minimize the specialized loss $\ell(\cdot, \cdot)$ (e.g., the cross-entropy loss for image classification task) with respect to the prompt blocks and network head, while keeping the pre-trained model parameters frozen. By only training a few additional parameters, the gradients can be back-propagated all the way through the pre-trained model, distilling the rich knowledge encoded in pre-trained parameters for learning the task-specific prompt.

IV. EXPERIMENT

In this section, we systematically verify the capability of the proposed Pro-tuning. Firstly, we introduce experimental settings and implementation details. Secondly, we perform extensive experiments on a wide range of challenging benchmarks to evaluate the effectiveness of Pro-tuning. Then, we provide comprehensive ablation studies to analyze Pro-tuning thoroughly, including three model robustness evaluations and the effects of different design choices. We also show the powerful scalability of Pro-tuning to object detection and semantic segmentation. Moreover, we perform further analysis on Pro-tuning in long-tailed classes. Finally, we provide visualization and failure case analysis in the appendix.

A. Experimental Settings

1) Datasets: As is common practice [4], [18], [29], we adopt seven generic object classification datasets covering a wide range of domains, including superordinate-level object classification (CIFAR-10 [7], CIFAR-100 [7], Clothing1M [4], Caltech-101 [49]) and fine-grained object classification (Stanford Dogs [50], Oxford-IIT Pets [51], Oxford 102 Flowers [52]). Moreover, we use the long-tailed versions of CIFAR-10 and CIFAR-100 with controllable degrees of data imbalance on the class imbalance transferability, as practiced in [8]. Next, we use CIFAR-10-C and CIFAR-100-C [53] for image corruption robustness, ImageNet-A [54] for adversarial robustness, and ImageNet-R [55] for out-of-distribution generalization, respectively. More details are in the appendix.

For the used abbreviations, “IR”: imbalance ratio, “C-10-LT”: long-tailed CIFAR-10, “C-100-LT”: long-tailed CIFAR-100.
| Backbone   | Method         | Params† (M) | C-10-LT IR100 | C-10-LT IR50 | C-10-100 IR100 | C-10-100 IR50 | C-10 | C-100 | PET | Flower | DOG | CIM | C-101 |
|------------|----------------|-------------|----------------|--------------|----------------|--------------|------|------|-----|--------|-----|-----|-------|
| Scratch    |                | 23.75       | 79.6           | 84.4         | 42.6           | 50.1         | 96.4 | 80.4 | 78.2| 77.8  | 66.8| 77.0| 72.2  |
| Linear     |                | 0.25        | 83.6           | 83.0         | 56.1           | 60.2         | 86.1 | 69.7 | 91.8| 93.0  | 88.9| 66.5| 90.0  |
| Partial-1 [20] |              | 4.71        | 86.5           | 88.9         | 57.9           | 62.5         | 95.2 | 79.3 | 91.0| 96.8  | 87.4| 75.3| 91.5  |
| Partial-2 [20] |              | 9.17        | 87.3           | 89.5         | 56.8           | 62.1         | 96.1 | 80.9 | 91.4| 97.3  | 85.8| 76.5| 92.2  |
| Bias [21]  |                | 0.27        | 88.8           | 90.7         | 57.8           | 63.5         | 95.4 | 80.8 | 91.9| 94.9  | 86.2| 75.7| 91.2  |
| Side-tuning [22] |            | 7.05        | 84.0           | 83.0         | 56.4           | 60.8         | 86.3 | 70.0 | 91.8| 93.4  | 89.0| 66.6| 90.3  |
| Adapter-1 [27] |              | 2.26        | 85.8           | 88.8         | 52.5           | 58.2         | 96.6 | 82.6 | 89.7| 95.1  | 78.9| 77.4| 89.4  |
| Adapter-2 [27] |              | 4.26        | 88.9           | 91.2         | 58.6           | 63.8         | 96.8 | 83.0 | 91.6| 96.9  | 79.5| 77.6| 92.1  |
| Fine-tuning |                | 23.75       | 89.7           | 92.0         | 60.5           | 65.7         | 97.5 | 84.5 | 92.5| 98.3  | 85.6| 78.7| 93.4  |
| Pro-tuning  |                | 3.86        | 89.9           | 92.2         | 61.2           | 66.0         | 97.8 | 84.8 | 92.9| 98.0  | 89.2| 77.0| 92.2  |

CIFAR-100, “C-10”: CIFAR-10, “C-100”: CIFAR-100, “PET”: Oxford-IIIT Pets, “Flower”: Oxford 102 Flowers, “DOG”: Stanford Dogs, “C1M”: Clothing1M, “C-101”: Caltech-101, “C-10-C”: CIFAR-10-C, “C-100-C”: CIFAR-100-C, “IM-A”: ImageNet-A, “IM-R”: ImageNet-R.

2) Evaluation Metrics: We adopt the widely used metrics of the top-1 accuracy and trainable parameters, as is common practice [21], [22], [27].

3) Baseline Methods: To put our results in perspective, we compare Pro-tuning with a series of commonly-adopted protocols: 1) fine-tuning, which fully updates all the model parameters when adapting to downstream tasks; 2) partial fine-tuning [20], which only updates the head and last several layers of the pre-trained model while freezing the remaining parts, including two variants “Partial-1” and “Partial-2” according to the number of trainable parameters; 3) linear probing, denoted by “Linear”, which freezes the model parameters except the head; trains the whole model from random initialization; 4) Bias tuning [21], denoted by “Bias”, which freezes the model parameters except the head and the bias terms; 5) Side-tuning [22], which trains a lightweight side network to be fused with the frozen pre-trained model via summation; 7) Adapter [27], which inserts MLP modules into the pre-trained model, including two variants “Adapter-1” and “Adapter-2” with the intermediate dimensions as 512 and 256 respectively.

For vision foundation models, we employ a series of representative backbones, including CNN-based models (ResNet-50 [19], ResNet-101 [19], RegNetX-32G [56], ConvNeXt-XL [57]) and vision transformers (DeiT-S [18], Swin-S [58], DeiT-B [18], Swin-B [58], Swin-L [58]).

B. Implementation Details

In our prompt block, two 1 × 1 convolutions share the same configurations. Each 1 × 1 convolution contains stride 1, zero padding, and group 4. The 5 × 5 depthwise convolution contains stride 1 and padding 2. All convolutions do not change the channel dimension. Moreover, a SE module [59] is appended to the 5 × 5 depthwise convolution for learning the inter-dependency between channels. The learnable parameter $\beta$ in Eq. (2) is initialized as $1 \times 10^{-5}$. We directly adopt the ImageNet-1K and ImageNet-21K supervised pre-trained models, MAE [60] and MoCo v3 [61] self-supervised models. More details are in the appendix.

C. Comparisons With State-of-the-Arts

1) Class Imbalance Transferability: We conduct extensive experiments on long-tailed CIFAR-10 and CIFAR-100 [8]. Table I and Table II show the results of our proposed Pro-tuning and other tuning paradigms. Under three pre-trained CNNs, Pro-tuning achieves strong performance on various imbalanced scenarios. Specifically, on long-tailed CIFAR-100 with imbalance ratio $\rho = 100$, Pro-tuning brings +2.1% validation accuracy improvement and reduces 11.1x fewer trainable parameters over fine-tuning under ResNet-101 [19]. Regarding the larger-capacity RegNetX-32G [56], Pro-tuning brings a substantial performance gain of +4.5% validation accuracy over Partial-2 [20] with the reduction of trainable parameters by 41.2% on long-tailed CIFAR-10 with imbalance ratio $\rho = 100$.

1To keep computationally efficient, we adopt the group convolution and observe no obvious performance degradation.
TABLE II
THE PERFORMANCE OF OUR PROPOSED PRO-TUNING AND OTHER TUNING PARADIGMS UNDER FOUR TRANSFORMER-BASED PRE-TRAINED MODELS ON NINE BENCHMARKS. "PARAMS†" DENOTES THE MAXIMUM NUMBER OF TRAINABLE PARAMETERS. THE BEST AND NEXT BEST RESULTS ARE BOLDED AND UNDERLINED, RESPECTIVELY.

| Backbone   | Method             | Params† | C-10-LT IR100 | C-10-LT IR50 | C-10 | C-100 | PET | Flower | DOG | CIM | C-101 |
|------------|--------------------|---------|----------------|--------------|------|-------|-----|--------|-----|-----|-------|
|            |                    | (M)     | C-10-LT IR100  | C-10-LT IR50 | C-10 | C-100 | PET | Flower | DOG | CIM | C-101 |
|            |                    |         |                |              |      |       |     |        |     |     |       |
| DeiT-B [18]| Scratch            | 21.71   | 58.9 67.9      | 33.5 37.9    | 95.3 | 74.1  | 41.8| 68.9   | 34.4| 66.2| 50.1  |
|            | Linear             | 0.05    | 83.6 86.9      | 63.1 66.8    | 92.6 | 77.0  | 92.8| 89.0   | 92.7| 76.0| 91.5  |
|            | Partial-1 [20]     | 1.23    | 87.7 89.2      | 62.1 66.5    | 95.6 | 80.7  | 92.2| 92.7   | 92.7| 76.9| 91.5  |
|            | Partial-2 [20]     | 1.82    | 89.3 91.6      | 64.6 68.9    | 96.9 | 83.5  | 93.2| 95.6   | 92.2| 74.6| 92.3  |
|            | Bias [21]          | 0.10    | 90.4 92.0      | 65.6 70.5    | 97.2 | 84.7  | 93.1| 96.1   | 92.4| 74.3| 92.5  |
|            | Side-tuning [22]   | 3.25    | 81.3 85.7      | 62.2 66.6    | 92.6 | 77.0  | 92.9| 87.5   | 92.8| 68.3| 90.4  |
|            | Adapter-1 [27]     | 1.04    | 91.1 92.5      | 67.7 72.2    | 97.4 | 86.8  | 93.0| 96.3   | 97.2| 76.5| 92.7  |
|            | Adapter-2 [27]     | 2.02    | 91.3 92.8      | 68.0 72.6    | 97.6 | 86.9  | 93.3| 96.5   | 91.4| 76.8| 93.3  |
|            | Fine-tuning        | 21.71   | 92.0 94.0      | 69.4 73.6    | 98.5 | 87.5  | 93.8| 98.3   | 87.5| 78.5| 94.8  |
|            | Pro-tuning         | 0.85    | 92.6 94.2      | 70.0 74.4    | 98.4 | 86.0  | 94.0| 98.5   | 93.0| 78.3| 94.3  |

For state-of-the-art transformer-based architectures, Pro-tuning consistently surpasses other tuning methods in various imbalanced scenarios. In particular, on long-tailed CIFAR-100 with imbalance ratio \( \rho = 50 \), Pro-tuning brings +1.9\% validation accuracy gain over Adapter-2 [27] under DeiT-B [18] while reducing trainable parameters by 21.5\%. On long-tailed CIFAR-10 with imbalance ratio \( \rho = 50 \) under Swin-B [58], Pro-tuning brings a gain of +8.3\% validation accuracy over Side-tuning [22] using 1.6x fewer trainable parameters.

2) Generic Object Transferability: The quantitative results on seven downstream datasets of generic objects are reported in Table I and Table II. One can observe that Pro-tuning achieves superior performance over other tuning methods under various pre-trained models in most scenarios. Specifically, Pro-tuning obtains 97.8\% top-1 accuracy under ResNet-50 on CIFAR-10 [7], which is +1.7\% higher than that of Partial-2 while only using 2.4x fewer trainable parameters. Under three pre-trained CNNs, Pro-tuning brings gains of +1.1\%~+2.1\% validation accuracy over Adapter-2 on Oxford-IIIT Pets [51] while obtaining superior performance compared with fine-tuning.

For advanced transformer-based architectures, Pro-tuning brings +5.0\% accuracy improvement over fine-tuning with up to 27x fewer trainable parameters under DeiT-B on Stanford Dogs [50]. Moreover, Pro-tuning obtains +10.2\% validation accuracy gain over Partial-1 while reducing the trainable parameters by 37.1\% under Swin-S on Clothing1M [4].

D. Ablation Study
In this subsection, extensive ablation studies are performed to analyze the proposed Pro-tuning systematically. If not stated particularly, the experimental settings inherit directly from the statements in Section IV-A and Section IV-B. More details are included in the appendix.

1) Image Corruption Robustness: We evaluate Pro-tuning on the image corruption robustness on CIFAR-10-C and CIFAR-100-C [53] in Table III. Notably, Pro-tuning consistently performs better than other methods on two datasets under three pre-trained models. On CIFAR-10, Pro-tuning brings a gain of +2.7\% over fine-tuning, with only 6.2x fewer trainable parameters under ResNet-50. Compared with Partial-2, Pro-tuning achieves better transfer performance with +3.3\%~+6.2\% validation accuracy gains with the significant reduction of trainable parameters by 2.2x~3.1x under three backbones.

On CIFAR-100-C, Pro-tuning brings +1.0\% validation accuracy gain over Adapter-2 with up to 35.0\% drop in trainable parameters under ResNet-50. Compared with Partial-1, Pro-tuning achieves better transfer performance with +3.3\%~+6.2\% validation accuracy gains with the significant reduction of trainable parameters by 2.2x~3.1x under three backbones.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
and +13.8% validation accuracy gains over bias tuning [21] and Side-tuning under DeIT-S, respectively. The dramatic improvements demonstrate the capability of our proposed Pro-tuning to defend against image corruption.

2) Natural Adversarial Examples: We employ ImageNet-A [54] to evaluate Pro-tuning on natural adversarial examples in Table IV. The performances of all tuning methods are relatively low on ImageNet-A, as this task is quite difficult. Even though, our Pro-tuning also outperforms other tuning methods under four advanced pre-trained models. For example, Pro-tuning surpasses fine-tuning by bringing in +4.8% performance gain under DeiT-S, while requiring only 24.7x fewer trainable parameters. Moreover, Pro-tuning brings +0.9% accuracy gain over Adapter-2 with the reduction of trainable parameters by 33.9% under Swin-S. Regarding Swin-B, Pro-tuning brings a significant accuracy gain of +4.1% over fine-tuning with the reduction of trainable parameters by 96.8%.

3) Out-of-Distribution Generalization: The validation accuracy and trainable parameters on ImageNet-R [55] are listed in Table IV to investigate the out-of-distribution generalization of Pro-tuning. Note that, Pro-tuning outperforms fine-tuning under DeIT-S and Swin-S, with gains of +1.0% and +1.2% top-1 accuracy using up to 24x and 30x fewer trainable parameters, respectively. Regarding DeIT-B, Pro-tuning obtains 44.6% top-1 accuracy, +1.5% higher than Partial-2 while reducing trainable parameters by 55.2%. Pro-tuning also brings +0.8% validation accuracy gain over Adapter-2 with 1.5x fewer trainable parameters under Swin-S. Moreover, Pro-tuning surpasses Side-tuning by +1.1% validation accuracy with the reduction of trainable parameters by 38.1% under Swin-B.

4) Collaboration With Fine-Tuning: To reveal the influence of different tuning settings, we compare with one important additional baseline (denoted by Pro-tuning FT), which updates all model parameters including our proposed prompt blocks and the pre-trained model. As shown in Fig. 3 (a), joint training of prompt blocks and the pre-trained model achieves a superior top-1 accuracy (85.7%) compared with individually updating prompt blocks (84.8%) or fine-tuning (84.5%) under ResNet-50 on CIFAR-100. This shows that more trainable parameters could further improve the transferability of the whole vision model for the proposed Pro-tuning.

5) Impact of Training Data Scale: To further evaluate the proposed Pro-tuning on training data efficiency, we show the validation accuracy under different scales of training examples under DeiT-B on CIFAR-100. As shown in Fig. 3 (b), Pro-tuning surpasses fine-tuning on various data scales, especially when the training data size is limited. A vivid example is that Pro-tuning brings +2.3% accuracy gain than fine-tuning (83.3% vs. 81.0%) with 20% training data. This adequately validates that the proposed Pro-tuning has a strong transferability to rapidly adapt the pre-trained model to a new task, even with only a small number of training examples.

### Table III

| Backbone     | Method     | Params\(^1\) (M) | CIFAR-10-C | CIFAR-100-C |
|--------------|------------|-----------------|------------|-------------|
| ResNet-50 [19] | Scratch    | 23.71           | 77.0       | 55.2        |
|              | Linear     | 0.20            | 62.2       | 43.6        |
|              | Partial-1 [20] | 4.67           | 76.0       | 53.5        |
|              | Partial-2 [20] | 9.13           | 77.3       | 55.4        |
|              | Bias [21]  | 0.23            | 80.2       | 56.3        |
|              | Side-tuning [22] | 7.01           | 61.7       | 43.8        |
|              | Adapter-1  | 2.21            | 81.1       | 56.7        |
|              | Adapter-2  | 4.22            | 80.9       | 57.8        |
|              | Fine-tuning | 23.71           | 80.8       | 57.7        |
|              | Pro-tuning  | 3.82            | 83.5       | 58.6        |

### Table IV

| Backbone     | Method     | Params\(^1\) (M) | ImageNet-A | ImageNet-R |
|--------------|------------|-----------------|------------|------------|
| Swin-S [58]  | Scratch    | 21.74           | 6.5        | 24.9       |
|              | Linear     | 0.08            | 19.2       | 40.7       |
|              | Partial-1 [20] | 1.26           | 18.5       | 39.8       |
|              | Partial-2 [20] | 1.85           | 19.0       | 40.2       |
|              | Bias [21]  | 0.13            | 19.3       | 40.7       |
|              | Side-tuning [22] | 3.27           | 18.9       | 40.5       |
|              | Adapter-1 [27] | 1.07           | 19.5       | 40.4       |
|              | Adapter-2 [27] | 2.05           | 19.7       | 40.9       |
|              | Fine-tuning | 21.74           | 19.6       | 40.6       |
|              | Pro-tuning  | 0.88            | 20.4       | 41.6       |

Note: "Params\(^1\)" denotes the number of trainable parameters. The best and next best results are BOLDED and UNDERLINED, respectively.

---

TABLE III

The Performance of Pro-Tuning and Other Tuning Paradigms on CIFAR-10-C and CIFAR-100-C. "Params\(^1\)" Denotes the Number of Trainable Parameters. The Best and Next Best Results Are BOLDED and UNDERLINED, Respectively.

| Backbone     | Method     | Params\(^1\) (M) | CIFAR-10-C | CIFAR-100-C |
|--------------|------------|-----------------|------------|-------------|
| ResNet-50 [19] | Scratch    | 21.74           | 6.5        | 24.9       |
|              | Linear     | 0.08            | 19.2       | 40.7       |
|              | Partial-1 [20] | 1.26           | 18.5       | 39.8       |
|              | Partial-2 [20] | 1.85           | 19.0       | 40.2       |
|              | Bias [21]  | 0.13            | 19.3       | 40.7       |
|              | Side-tuning [22] | 3.27           | 18.9       | 40.5       |
|              | Adapter-1 [27] | 1.07           | 19.5       | 40.4       |
|              | Adapter-2 [27] | 2.05           | 19.7       | 40.9       |
|              | Fine-tuning | 21.74           | 19.6       | 40.6       |
|              | Pro-tuning  | 0.88            | 20.4       | 41.6       |

Note: "Params\(^1\)" denotes the maximal number of trainable parameters. The best and next best results are BOLDED and UNDERLINED, respectively.

---

TABLE IV

The Performance of Pro-Tuning and Other Tuning Paradigms on ImageNet-A and ImageNet-R. "Params\(^1\)" Denotes the Number of Trainable Parameters. The Best and Next Best Results Are BOLDED and UNDERLINED, Respectively.

| Backbone     | Method     | Params\(^1\) (M) | ImageNet-A | ImageNet-R |
|--------------|------------|-----------------|------------|------------|
| Swin-S [58]  | Scratch    | 21.74           | 6.5        | 24.9       |
|              | Linear     | 0.08            | 19.2       | 40.7       |
|              | Partial-1 [20] | 1.26           | 18.5       | 39.8       |
|              | Partial-2 [20] | 1.85           | 19.0       | 40.2       |
|              | Bias [21]  | 0.13            | 19.3       | 40.7       |
|              | Side-tuning [22] | 3.27           | 18.9       | 40.5       |
|              | Adapter-1 [27] | 1.07           | 19.5       | 40.4       |
|              | Adapter-2 [27] | 2.05           | 19.7       | 40.9       |
|              | Fine-tuning | 21.74           | 19.6       | 40.6       |
|              | Pro-tuning  | 0.88            | 20.4       | 41.6       |

Note: "Params\(^1\)" denotes the number of trainable parameters. The best and next best results are BOLDED and UNDERLINED, respectively.
Fig. 3. Ablation studies of the proposed Pro-tuning. (a) The trade-off between trainable parameters and accuracy. “Params†” denotes the number of trainable parameters. “Pro-tuning FT” updates all model parameters including our prompt blocks and the pre-trained model. (b) Performance comparison obtained with different training data scales. The performance gains brought by Pro-tuning are marked in the gray background. (c) Performance of Pro-tuning obtained with different ratios $\beta$ in Eq. (2), where “Ci” denotes that $\beta$ equals the fixed constant $i$, otherwise a learnable coefficient denoted by “Lea.”. (d) Effect of different inserting positions of the prompt block. (e) Influence of different capacities of prompt blocks. (f) Performance of Pro-tuning under different kernel sizes. (g) Performance comparison on different imbalance ratios varying from 0, 50, and 100 on CIAFR-100. Particularly, the imbalance ratio of the original CIFAR-100 is 0. (h) Effect of the SE module. Notably, Pro-tuning consistently outperforms Adapter-2 on diverse scenarios evolved from CIFAR-100, regardless of adding the SE module. All means and variances are calculated in five independent runs. Best viewed in color.

6) Influence of the Pre-Trained Scale: We conduct experiments under ImageNet-21K pre-trained Swin-B to study the influence of the larger pre-training. ImageNet-21K is a superset of ImageNet-1K, comprised of 21K classes and 14M images. Particularly, we add an advanced baseline VPT [40], which is specifically designed for vision transformers on image classification with ImageNet-21K pre-training. We exactly follow the training settings in the original paper. Additionally, we provide a lower-capacity variant of our Pro-tuning, denoted by Pro-tuning*, which is set to learn without the SE module while reducing the channel dimension of prompt blocks by 25%. As listed in Table V, one can observe that the proposed Pro-tuning consistently obtains superior performance over other tuning methods on a wide variety of datasets including diverse negative perturbations. Particularly, the lower-capacity variant of our Pro-tuning achieves excellent parameters-accuracy trade-offs: Pro-tuning* brings a gain of +0.5% mean validation accuracy over VPT while
reducing trainable parameters by up to 11.1% on various scenarios.

7) Impact of the Foundation Model Capacity: We evaluate the proposed Pro-tuning on current state-of-the-art transformer-based and CNN-based large vision foundation models: Swin-L [58] (195M) and ConvNeXt-XL [57] (350M), as listed in Table VI and Table VII, respectively. For a fair comparison, we follow the experimental setting as in the original paper of Swin-L and ConvNeXt-XL. In addition, we provide a lower-capacity variant of our Pro-tuning, denoted by “Pro-tuning∗”, which learns without the SE module and reduce the channel dimension of prompt blocks to 25% of the original. Specifically, as listed in Table VI, Pro-tuning outperforms fine-tuning on eight of the ten challenging benchmarks with up to 50x fewer trainable parameters under Swin-L. Pro-tuning∗ brings +0.5% average accuracy gains over VPT with the reduction of trainable parameters by 14.9% under Swin-L. Regarding ConvNeXt-XL, Pro-tuning also exhibits excellent performance. For example, as shown in Table VII, Pro-tuning consistently outperforms Adapter-2 on ten challenging benchmarks while reducing trainable parameters by 25.5%.

8) Analysis of Training Efficiency: We further discuss the training efficiency, including training time, training memory cost, and others. For example, as shown in Table VIII, Pro-tuning consistently outperforms Adapter-2 on ten challenging benchmarks while reducing trainable parameters by 25.5%.

8) Analysis of Training Efficiency: We further discuss the training efficiency, including training time, training memory cost, and others. For example, as shown in Table VIII, Pro-tuning consistently outperforms Adapter-2 on ten challenging benchmarks while reducing trainable parameters by 25.5%.
TABLE X

| Method          | Params $^1$ (M) | Time (h) | Mem. (MB) | C-10-LT | C-100-LT | C-100-LT | C-10-CL | C-100-CL | CIM   | C-101 | C-10-CL | C-100-CL | IM-A  | IM-R  |
|-----------------|----------------|----------|-----------|---------|---------|---------|---------|---------|-------|-------|---------|---------|-------|-------|
| ResNet-101      |                |          |           |         |         |         |         |         |       |       |         |         |       |       |
| Series [42]     | 43.20          | 36.2     | 17122     | 90.2    | 62.5    | 86.2    | 93.4    | 78.6    | 93.0  | 82.6  | 59.8    | 21.4    | 43.0  |       |
| Parallel [43]   | 21.05          | 26.9     | 6557      | 90.5    | 62.9    | 86.4    | 93.5    | 78.8    | 93.0  | 83.1  | 60.3    | 21.8    | 43.1  |       |
| Fine-tuning     | 42.91          | 24.0     | 6829      | 90.8    | 61.8    | 87.0    | 93.7    | 78.5    | 93.2  | 83.7  | 60.7    | 21.3    | 43.4  |       |
| Pro-tuning      | 4.02           | 20.6     | 7345      | 91.0    | 63.6    | 86.6    | 93.9    | 79.0    | 93.8  | 83.7  | 61.0    | 22.4    | 44.2  |       |
| ResNet-26       |                |          |           |         |         |         |         |         |       |       |         |         |       |       |
| Series [42]     | 0.76           | 21.82    | 7644      | 69.0    | 44.0    | 73.4    | 87.6    | 77.3    | 84.3  | 66.2  | 44.1    | 12.7    | 24.0  |       |
| Parallel [43]   | 0.70           | 18.52    | 4533      | 69.3    | 44.4    | 73.8    | 87.7    | 77.5    | 84.4  | 66.6  | 44.2    | 14.1    | 25.4  |       |
| Fine-tuning     | 5.87           | 20.46    | 4524      | 69.7    | 45.1    | 74.4    | 88.0    | 78.1    | 84.6  | 67.0  | 44.6    | 13.1    | 25.0  |       |
| Pro-tuning      | 0.12           | 18.87    | 7638      | 69.0    | 44.7    | 73.9    | 88.2    | 77.8    | 84.8  | 67.1  | 45.0    | 13.9    | 25.8  |       |

Results of pro-tuning and other tuning paradigms under ResNet-101 and ResNet-26. “Params” denotes the maximum number of trainable parameters. “Time” denotes the average training time over the used benchmarks. “Mem.” denotes the training memory cost. The best and next best results are bolded and underlined, respectively.

cost, and parameter efficiency. Concretely, Table VIII shows the experimental results under two representative vision foundation models, i.e., ResNet-101 [19] and Swin-B [58]. One can observe that Pro-tuning achieves the best speed-accuracy trade-off among these methods on multiple challenging benchmarks. For example, Pro-tuning surpasses fine-tuning with +0.7% accuracy gains on average while reducing the training time, training memory cost, trainable parameters by 29.5%, 20.1%, 96.9% respectively on eight challenging benchmarks under Swin-B. These results adequately prove the effectiveness of Pro-tuning in training efficiency and performance.

9) Effect of Convolution in Prompt Block: We compare Pro-tuning with six additional baselines, i.e., Prefix-tuning [35], LoRA [37], VPT [40], AdaptFormer [41], Series [42], and Parallel [43]. We also provide a lower-capacity variant of our LoRA [37], VPT [40], AdaptFormer [41], Series [42], and cost of +1.2% with comparable numbers of parameters under MAE pre-training. For MoCo v3, Pro-tuning performs better than Adapter-2 with +0.7% average accuracy gains and the reduction of trainable parameters by 21.0% on ten challenging benchmarks.

11) Effect of Adaptive Prompt: To evaluate the effect of adaptive prompt, we compare Pro-tuning under different settings of the ratio $\beta$ in prompt blocks in Eq. (2) under ResNet-50 on CIFAR-100. Specifically, we set $\beta$ as a fixed constant or a learnable parameter (our default setting). As shown in Fig. 3 (c), the learnable $\beta$ achieves superior performance compared with all fixed constants, while facilitating accelerating the convergence speed. This adequately validates the effectiveness of our adaptive prompt. Interestingly, when setting $\beta$ as fixed constants, we can find that decreasing $\beta$ generally boosts performance, but significantly reduces the validation accuracy when it is less than 0.1. A considerable reason is that properly restricting the feature representations from prompt blocks could facilitate transferability, but a quite small and fixed $\beta$ causes the learned prompt to have little effect since it is consistently smaller (e.g., 3 or 5 orders of magnitude) than the corresponding output of the network stage.

12) Investigation of Inserting Positions: We explore the impact of different inserting positions of prompt blocks under DeiT-B on CIFAR-100. Concretely, we compare Pro-tuning that uses the multi-stage inserting strategy with its variants that append all prompt blocks after a single position of network stages, including the embedding layer, the first transformer layer, the third transformer layer, the sixth transformer layer, the ninth transformer layer, and the twelfth transformer layer. For a fair comparison, these inserting ways have the same parameters as our adopted multi-stage inserting strategy. As shown in Fig. 3 (d), we can observe that our used multi-stage inserting strategy outperforms other strategies, with the validation accuracy gains of +1.0%~7.1%. Interestingly, one can observe that inserting after the lower layer (e.g., 3rd or 6th) typically obtains better transfer performance than inserting after the last layer. This is reasonable since low-level and mid-level feature representations could be more significant for the transfer compared with high-level feature representations, which has been pointed out in recent studies [47], [63].

13) Sensitivity to the Capacity of Prompt Blocks: To investigate the impact of the capacity of prompt blocks, we compare
TABLE XI
RESULTS OF PRO-TUNING AND OTHER TUNING PARADIGMS UNDER VI-T-BASE [58] PRE-TRAINED ON MAE [60] AND MoCo v3 [61]. “PARAMS†” DENOTES THE MAXIMUM NUMBER OF TRAINABLE PARAMETERS. THE BEST AND NEXT BEST RESULTS ARE BOLDED AND UNDERLINED, RESPECTIVELY.

| Method                          | Params† (M) | C-10-LT | C-100-LT | C-10-LT | C-10-LT | PET     | C-101 | C-10-C | C-100-C | IM-A | IM-R |
|--------------------------------|-------------|---------|----------|---------|---------|---------|-------|--------|---------|------|------|
| MAE                            |             |         |          |         |         |         |       |        |         |      |      |
| Linear                         | 0.15        | 38.5    | 16.9     | 72.2    | 47.0    | 62.1    | 73.7  | 54.1   | 33.0    | 2.5  | 23.5 |
| Partial-1 [20]                 | 4.88        | 62.1    | 38.5     | 86.9    | 67.4    | 86.4    | 91.5  | 67.6   | 46.6    | 4.4  | 30.0 |
| Partial-2 [20]                 | 7.24        | 84.6    | 51.6     | 96.0    | 81.5    | 89.9    | 92.6  | 84.5   | 62.9    | 13.8 | 31.3 |
| Bias [21]                      | 0.26        | 74.7    | 47.3     | 95.7    | 80.1    | 89.8    | 92.2  | 82.8   | 61.9    | 8.8  | 31.7 |
| Side-tuning [22]               | 5.66        | 38.8    | 16.8     | 72.1    | 46.9    | 61.9    | 74.0  | 54.3   | 33.1    | 2.6  | 23.5 |
| Adapter-1 [27]                 | 2.13        | 90.6    | 57.8     | 97.5    | 86.8    | 90.3    | 93.5  | 88.5   | 65.3    | 10.4 | 34.2 |
| Adapter-2 [27]                 | 4.10        | 90.2    | 57.9     | 97.7    | 86.6    | 90.6    | 93.6  | 88.0   | 65.7    | 10.8 | 34.4 |
| VPT [40]                       | 0.61        | 87.3    | 51.4     | 84.1    | 78.9    | 83.7    | 80.8  | 82.1   | 61.0    | 10.6 | 28.4 |
| AdapFormers [41]               | 1.34        | 91.1    | 59.1     | 98.0    | 85.9    | 90.3    | 92.7  | 85.0   | 65.2    | 11.7 | 34.6 |
| Fine-tuning†                   | 85.95       | 97.0    | 59.8     | 98.7    | 87.4    | 90.8    | 93.5  | 90.3   | 66.2    | 13.2 | 37.0 |
| Pro-tuning†                    | 1.35        | **91.8**| **60.4** | 98.3    | **87.2**| **91.9**| **94.1**| **88.5**| **66.2**| **11.6**| **35.6**|
| Pro-tuning                     | 3.24        | 91.8    | 60.3     | 98.3    | 87.5    | 91.1    | 93.8  | 90.5   | 67.1    | **13.3**| **36.6**|

MoCo v3

| Method                          | Params† (M) | C-10-LT | C-100-LT | C-10-LT | C-10-LT | PET     | C-101 | C-10-C | C-100-C | IM-A | IM-R |
|--------------------------------|-------------|---------|----------|---------|---------|---------|-------|--------|---------|------|------|
| Linear                         | 0.15        | 61.9    | 28.1     | 93.6    | 74.7    | 86.2    | 88.8  | 77.8   | 54.8    | 9.8  | 37.6 |
| Partial-1 [20]                 | 4.88        | 91.0    | 46.5     | 97.2    | 85.4    | 91.4    | 93.1  | 83.1   | 64.5    | 13.8 | 40.5 |
| Partial-2 [20]                 | 7.24        | 92.4    | 66.5     | 98.2    | 87.9    | 91.7    | 93.9  | 88.8   | 70.7    | 13.7 | 44.0 |
| Bias [21]                      | 0.26        | 91.3    | 49.1     | 97.7    | 86.6    | 89.6    | 91.8  | 89.0   | 71.4    | 16.4 | 43.1 |
| Side-tuning [22]               | 5.66        | 61.7    | 28.3     | 93.5    | 74.7    | 86.2    | 88.8  | 77.8   | 54.8    | 9.9  | 37.7 |
| Adapter-1 [27]                 | 2.13        | 92.4    | 66.6     | 98.4    | 89.4    | 91.8    | 94.5  | 91.4   | 71.7    | 18.3 | 42.8 |
| Adapter-2 [27]                 | 4.10        | 92.9    | 67.4     | 98.4    | 89.7    | 91.7    | 94.4  | 90.9   | 72.3    | 19.1 | 43.2 |
| VPT [40]                       | 0.61        | 93.8    | 69.5     | 98.2    | 87.7    | 91.3    | 93.7  | 90.3   | 72.0    | 18.8 | 42.7 |
| AdapFormers [41]               | 1.34        | 94.0    | 69.7     | 98.5    | 88.1    | 91.5    | 94.1  | 90.5   | 72.3    | 18.7 | 41.9 |
| Fine-tuning†                   | 85.95       | 93.0    | 68.7     | 98.9    | 89.6    | 92.5    | 95.6  | 91.3   | 72.4    | 19.8 | 43.8 |
| Pro-tuning†                    | 1.35        | **94.2**| **70.1** | **98.7**| **89.8**| **92.1**| **95.2**| **90.7**| **72.2**| **19.9**| **44.3**|
| Pro-tuning                     | 3.24        | 93.5    | 69.3     | **98.9**| **90.2**| **91.2**| **94.7**| **91.7**| **73.0**| **20.4**| **44.2**|

Pro-tuning with its variants that use multiple prompt blocks appended to each stage of pre-trained models under ResNet-50 on CIFAR-100. The results in Fig. 3 (e) indicate that the increase of prompt blocks brings little help to transferability. This shows that the proposed Pro-tuning suffices to achieve excellent performance using only one prompt block after each stage of pre-trained models.

14) Sensitivity to the Kernel Size of Prompt Blocks: We evaluate Pro-tuning under the different kernel sizes $k$ of the depthwise convolution in prompt blocks. The results in Fig. 3 (f) indicate that the performance generally improves with the increase of kernel size, then the performance will be saturated when $k \geq 5$ is satisfied. A considerable reason is that the increase of receptive field size typically enhances the representation ability. Nevertheless, a quite large kernel size could result in difficult optimization [57], hence there are few performance gains from further increasing. Particularly, we investigate the effect of kernel sizes in corrupted data in Table XII. Under DeiT-S, Pro-tuning generally performs better with the kernel size increasing on CIFAR-10-C and CIFAR-100-C. We think this could be attributed to the information exchange within a local region resulting from convolution in prompt blocks, which contributes to the robustness of Pro-tuning in corrupted image data. This similar phenomenon has also been observed in [64]. Thus, we set the kernel size as 5 in our experiments unless specified otherwise. Additionally, in Table XIII, we add an additional baseline
parameters by 9.5% and 26.3%, respectively. With $\rho = 50$ and $\rho = 100$, and CIFAR-100-C. The results indicate that Pro-tuning works better than modeling with no SE module. Note that, our Pro-tuning and its variant both surpass Adapter-2 with $+0.5\% \sim 2.6\%$ accuracy gains while reducing trainable parameters by 9.5% and 26.3%, respectively.

15) Impact of the Imbalance Ratio: To observe the influence of different degrees of class imbalance on various tuning methods, an intuitive comparison under ResNet-50 on CIFAR-100 and long-tailed CIFAR-100 is illustrated in Fig. 3 (g), where the results are from Table I. Particularly, the imbalance ratio of the original CIFAR-100 is 0. As shown in Fig. 3 (g) and Table I, one can find that the increase in imbalance ratios degrades the performance of all tuning methods, since the distribution perturbations from class imbalance result in domain gaps between pre-training and solving downstream tasks. Even in this case, Pro-tuning can achieve superior performance over other methods including fine-tuning, while the accuracy gains brought by Pro-tuning generally grow with the increasing of imbalance ratios. The results adequately validate that the proposed Pro-tuning exhibits the notable ability to the imbalanced classification tasks, not requiring updating all model parameters.

16) Analysis of Few-Shot Learning: We perform few-shot transfer under ResNet-50 on CIFAR-10, CIFAR-100, and Caltech-101. Following the representative few-shot protocol in [15], [65], and [66], we adopt 1, 2, 4, 8, and 16 shots for training and the full test sets for testing, as illustrated in Fig. 4. One can find that Pro-tuning is a strong few-shot learner, requiring only 4 shots to obtain a significant margin with $+2.1\%$ accuracy improvement over fine-tuning on CIFAR-10. Pro-tuning also achieves a superior result over fine-tuning and Adapter-2 with 8 shots, which brings $+2.9\%$ and $+2.0\%$ validation accuracy improvements on CIFAR-100 while reducing trainable parameters by 83.9% and 9.5% respectively. Moreover, Pro-tuning brings gains of $+7.5\%$ and $+4.8\%$ validation accuracy over fine-tuning and Partial-2 with 2 shots on Caltech-101, with 6.2x and 2.4x fewer trainable parameters, respectively.

17) Effectiveness of the SE Module: To analyze the effect of the SE module in prompt blocks, we realize Pro-tuning without the SE module, which is denoted by “w/o SE”. Fig. 3 (h) shows the comparisons of validation accuracy on CIFAR-100, long-tailed CIFAR-100 with imbalance ratios $\rho = 50$ and $\rho = 100$, and CIFAR-100-C. The results indicate that Pro-tuning works better than modeling with no SE module. Note that, our Pro-tuning and its variant both surpass Adapter-2 with $+0.5\% \sim 2.6\%$ accuracy gains while reducing trainable parameters by 9.5% and 26.3%, respectively.

### Table XIII

| Method       | Params\(^{\dagger}\) (M) | C-10-LT | C-100 | DOG | PET |
|--------------|--------------------------|--------|-------|-----|-----|
| Pro-tuning   | 63.62                    | 94.7   | 90.9  | 95.4| 94.5|
| Pro-tuning  \(_{\text{w/o SE}}\) | 3.86                     | 94.2   | 90.8  | 95.8| 95.3|

### Table XIV

| Methods | Params\(^{\dagger}\) | Epochs | AP  | AP\(_{50}\) | AP\(_{75}\) | AP\(_S\) | AP\(_M\) | AP\(_L\) |
|---------|----------------------|-------|-----|------------|------------|---------|---------|---------|
| Cascade Mask R-CNN [2] |                      |       |     |            |            |         |         |         |
| Linear  | $5.8M$               | 12    | 38.5| 57.9      | 41.6       | 22.5    | 41.4    | 51.5    |
| Partial-1 [20] | $5.6M$               | 12    | 38.8| 58.5      | 41.6       | 22.3    | 41.6    | 51.0    |
| Partial-2 [20] | $5.6M$               | 12    | 38.9| 58.4      | 42.0       | 22.5    | 41.5    | 51.6    |
| Bias [21]   | $5.6M$               | 12    | 38.9| 58.4      | 42.0       | 22.5    | 41.6    | 51.4    |
| Side-tuning [22] | $5.8M$               | 12    | 39.3| 58.8      | 42.7       | 22.9    | 42.3    | 52.4    |
| Adapter-1 [27] | $5.8M$               | 12    | 60.0| 59.7      | 43.5       | 23.7    | 42.9    | 53.4    |
| Adapter-2 [27] | $5.8M$               | 12    | 60.0| 59.7      | 43.5       | 23.7    | 42.9    | 53.4    |
| Fine-tuning | $5.8M$               | 12    | 60.1| 59.7      | 43.5       | 23.7    | 42.9    | 53.4    |
| Pro-tuning  | $5.7M$               | 12    | 61.2| 59.7      | 43.5       | 23.7    | 42.9    | 53.4    |

### Table XV

| Method       | Params\(^{\dagger}\) | Steps | mIoU |
|--------------|----------------------|-------|------|
| **FCN [3]**  |                      |       |      |
| Linear       | $26.1M$              | 80    | 26.7%|
| Partial-1 [20] | $30.6M$              | 80    | 27.0%|
| Partial-2 [20] | $30.6M$              | 80    | 27.3%|
| Bias [21]    | $26.1M$              | 80    | 27.1%|
| Side-tuning [22] | $32.9M$              | 80    | 27.9%|
| Adapter-1 [27] | $28.1M$              | 80    | 32.5%|
| Adapter-2 [27] | $30.1M$              | 80    | 32.9%|
| Fine-tuning  | $49.6M$              | 80    | 35.9%|
| Pro-tuning   | $29.7M$              | 80    | 36.2%|

### Table XVI

| Method       | Params\(^{\dagger}\) |                      |      |      |      |      |      |      |
|--------------|----------------------|----------------------|------|------|------|------|------|------|
| **UpperNet [68]** |                      |                      |      |      |      |      |      |      |
| Linear       | $43.0M$              | 80                  | 37.7%|
| Partial-1 [20] | $47.5M$              | 80                  | 38.4%|
| Partial-2 [20] | $51.9M$              | 80                  | 38.6%|
| Bias [21]    | $43.0M$              | 80                  | 37.8%|
| Side-tuning [22] | $49.8M$              | 80                  | 38.7%|
| Adapter-1 [27] | $45.0M$              | 80                  | 38.3%|
| Adapter-2 [27] | $47.0M$              | 80                  | 38.9%|
| Fine-tuning  | $66.5M$              | 80                  | 40.7%|
| Pro-tuning   | $46.0M$              | 80                  | 41.3%|

E. Further Analysis

To evaluate the transferability of Pro-tuning to dense prediction tasks, we perform experiments on COCO object detection [69] and ADE20K semantic segmentation [70]. Additionally, we further analyze Pro-tuning in long-tailed classes.

1) Object Detection: We evaluate Pro-tuning under ResNet-50 on two typical object detection frameworks – Cascade Mask R-CNN [2] and HTC [67]. Table XIV lists the object detection results. Without bells and whistles, Pro-tuning brings gains of $+1.9\% \sim 2.9\%$ box AP over Adapter-2 with fewer trainable parameters. Pro-tuning also outperforms Side-tuning with a significant performance gain of $+4.4\%$ box AP while reducing the trainable parameters by 5.3% under Cascade...
Task 2: Partial-2 while reducing more than 8% trainable parameters + tuning can obtain higher than Partial-1 with fewer trainable parameters. Regarding HTC, the result of Pro-tuning is Mask R-CNN, while reducing the trainable parameters by 25.6%. Regarding HTC, the result of Pro-tuning is.

Table XV shows the quantitative comparisons, indicating that the proposed Pro-tuning achieves the best performance. In particular, Pro-tuning surpasses Partial-2 by +8.9 mIoU with a large reduction of 15.1% in trainable parameters (29.7M vs. 35.0M) under FCN. Regarding UperNet, Pro-tuning is +0.6 mIoU higher than fine-tuning while significantly reducing the trainable parameters by 29.9%. Pro-tuning also brings a significant gain of +2.4 mIoU over Adapter-2 with fewer trainable parameters under UperNet.

3) Analysis of Long-Tailed Classes: We provide per-class results and the visualization of attention maps on long-tailed CIFAR-10 with imbalance ratio of 100. In Table XVI, we provide the validation accuracy of each class, with classes further to the right having fewer training samples. The last column is the average validation accuracy.

| Method     | airplane | automobile | bird | cat | deer | dog | frog | horse | ship | truck | overall |
|------------|----------|------------|------|-----|------|-----|------|-------|------|-------|---------|
| Adapter-1  | 99.9     | 99.7       | 99.2 | 96.5| 97.5 | 92.7| 96.2 | 94.8  | 86.5 | 78.9  | 94.2    |
| Adapter-2  | 99.2     | 99.8       | 99.3 | 96.3| 97.6 | 92.9| 96.3 | 96.0  | 89.0 | 82.0  | 94.9    |
| VPT [40]   | 99.7     | 99.4       | 98.7 | 96.7| 97.6 | 92.4| 96.1 | 97.5  | 89.9 | 84.6  | 95.3    |
| AdapFormer [41] | 99.8 | 99.6       | 98.7 | 96.6| 97.6 | 92.4| 96.4 | 97.7  | 90.1 | 84.3  | 95.3    |
| Fine-tuning| 99.8     | 99.3       | 99.1 | 96.5| 97.8 | 92.7| 96.9 | 98.0  | 91.2 | 86.0  | 95.7    |

Pro-tuning 99.7 99.6 99.3 96.8 98.0 93.0 97.1 98.5 92.2 87.2 96.1

Fig. 5. Visualization of example attention maps of Pro-tuning and other tuning methods under ImageNet-21K pre-trained ViT-Base on long-tailed CIFAR-10 with imbalance ratio of 100. The classes of samples from top to bottom are three tail classes – truck, ship, and horse, respectively.

Mask R-CNN. Meanwhile, Pro-tuning reaches the identical validation accuracy of fine-tuning under Cascade Mask R-CNN, while reducing the trainable parameters by 25.6%. Regarding HTC, the result of Pro-tuning is +3.4 box AP higher than Partial-1 with fewer trainable parameters. Pro-tuning can obtain +3.3~4.7 box AP gains compared to Partial-2 while reducing more than 8% trainable parameters under two frameworks.

2) Semantic Segmentation: For the base frameworks, we adopt FCN [3] and UperNet [68] to evaluate Pro-tuning under ImageNet-1K pre-trained ResNet-50 on ADE20K. Table XV shows the quantitative comparisons, indicating that the proposed Pro-tuning achieves the best performance. In particular, Pro-tuning surpasses Partial-2 by +8.9 mIoU with a large reduction of 15.1% in trainable parameters (29.7M vs. 35.0M) under FCN. Regarding UperNet, Pro-tuning is +0.6 mIoU higher than fine-tuning while significantly reducing the trainable parameters by 29.9%. Pro-tuning also brings a significant gain of +2.4 mIoU over Adapter-2 with fewer trainable parameters under UperNet.

3) Analysis of Long-Tailed Classes: We provide per-class results and the visualization of attention maps on long-tailed CIFAR-10 with imbalance ratio of 100. In Table XVI, we provide the validation accuracy of each class, with classes further to the right having fewer training samples, as is common practice [8]. One can observe that Pro-tuning brings significant gains on tail classes. For example, Pro-tuning outperforms AdapFormer with +0.8%, +2.1%, and +2.9% accuracy gains on horse, ship, and truck, respectively. To further understand the effect of Pro-tuning, we compare the visualizations of the learned attention maps. In Fig. 5, we visualize the attention maps of Pro-tuning and other tuning methods using Attention Rollout [71] on long-tailed CIFAR-10 with imbalance ratio of 100. We can find that Pro-tuning can capture richer information and attend well to the objects in the tail class, including truck, ship, and horse. This validates that Pro-tuning can adapt to downstream tasks well even in limited training samples.

V. CONCLUSION

In this work, we present Pro-tuning, a parameter-efficient vision tuning paradigm by extending the prompt-based philosophy to vision models. The main idea of Pro-tuning is to adapt frozen vision models to diverse downstream tasks by learning task-specific vision prompts. For this purpose, stage-wise prompt blocks are proposed to distill the rich knowledge encoded in the pre-trained parameters and multi-level feature representations of downstream data. Pro-tuning can integrate with existing vision models, including diverse CNNs and vision transformers. By only updating a handful of additional parameters, Pro-tuning can generate compact and robust downstream models. Extensive experiments on various vision tasks have demonstrated the effectiveness of Pro-tuning.

REFERENCES

[1] Y. Wang, D. Chang, Z. Fu, J. Wen, and Y. Zhao, “Incomplete multiview clustering via cross-view relation transfer,” IEEE Trans. Circuits Syst. Video Technol., vol. 33, no. 1, pp. 367–378, Jan. 2023.
[2] Z. Cai and N. Vasconcelos, “Cascade R-CNN: High-quality object detection and instance segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 5, pp. 1483–1498, May 2021.
[3] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3431–3440.
[4] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, “Learning from massive noisy labeled data for image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 2691–2699.
[5] M. Cheng, H. Wang, and Y. Long, “Meta-learning-based incremental few-shot object detection,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 4, pp. 2158–2169, Apr. 2022.
[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 2247–2250.
[7] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009-09.
[8] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, “Learning imbalanced datasets with label-distribution-aware margin loss,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2019, pp. 1877–1901.
[9] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, “Learning from massive noisy labeled data for image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 2691–2699.
[10] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009-09.
[11] T. Brown et al., “Language models are few-shot learners,” in Proc. NIPS, 2020, pp. 1877–1901.
[12] T. Gao, A. Fisch, and D. Chen, “Making pre-trained language models better few-shot learners,” in Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process., 2021, pp. 3816–3830.
[13] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” OpenAI Blog, 2018, pp. 1–12. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf
[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and J. Sutskever, “Language models are unsupervised multitask learners,” OpenAI Blog, vol. 1, no. 8, p. 9, 2019.

[13] T. Schick and H. Schütze, “Exploiting close questions for few shot text classification and natural language inference,” in Proc. EACL, 2020, pp. 255–269.

[14] C. Jia et al., “Scaling up visual and vision-language representation learning with noisy text supervision,” in Proc. ICML, 2021, pp. 4904–4916.

[15] Radford et al., “Learning transferable visual models from natural language supervision,” in Proc. ICML, 2021, pp. 8748–8763.

[16] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, “Conditional prompt learning for vision-language models,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 16795–16804.

[17] J. Li, D. Li, C. Xiong, and S. Hoi, “Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation,” in Proc. ICML, 2022, pp. 12888–12900.

[18] A. Todor, M. Chat, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, “Training data-efficient image transformers & distillation through attention,” in Proc. ICML, 2021, pp. 10347–10357.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[20] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in Proc. NIPS, 2014, pp. 3320–3328.

[21] H. Cai, C. Gan, L. Zhu, and S. Han, “TinyTL: Reduce memory, not parameters for efficient on-device learning,” in Proc. NIPS, 2020, pp. 11285–11297.

[22] J. Zhang, A. Sax, A. Zamir, L. Guibas, and J. Malik, “Side-tuning: A baseline for network adaptation via additive side networks,” in Proc. ECCV, 2020, pp. 698–714.

[23] Z. Peng, W. Zhang, N. Han, X. Fang, P. Kang, and L. Teng, “Active transfer learning,” IEEE Trans. Circuits Syst. Video Technol., vol. 30, no. 4, pp. 1022–1036, Apr. 2020.

[24] X. Shan, Y. Lu, Q. Li, and Y. Wen, “Model-based transfer learning and feature selection with multi-source transfer,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 11, pp. 4611–4625, Dec. 2020.

[25] N. Houlsby et al., “Parameter-efficient transfer learning for NLP,” in Proc. ICML, 2019, pp. 2790–2799.

[26] Z. Lin, A. Madotto, and P. Fung, “Exploring versatile generative language model via parameter-efficient transfer learning,” in Proc. Findings Assoc. Comput. Linguistics, EMNLP, 2020, pp. 441–459.

[27] S. Kornblith, J. Shlens, and Q. V. Le, “Do better ImageNet models transfer better?” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2656–2666.

[28] H. Zhou, S. Lai, H. Jin, X. Qian, and T. Mei, “Dual transfer deep hashing for efficient social image retrieval,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 2, pp. 742–753, Feb. 2021.

[29] L. Zhu, H. Cui, Z. Cheng, J. Li, and Z. Zhang, “Dual-level semantic transfer deep hashing for efficient social image retrieval,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 4, pp. 1478–1489, Apr. 2021.

[30] M. Tsimpoukelli, J. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill, “Multimodal few-shot learning with frozen language models,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 10966–11076.

[31] X. Wang, L. Gao, Y. Zhou, J. Song, and M. Wang, “KTN: Knowledge transfer network for learning multiperson 2D-3D correspondences,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 11, pp. 7732–7745, Nov. 2022.

[32] D. Hendrycks et al., “Benchmarking neural network robustness to common corruptions and perturbations,” in Proc. ICLR, 2019, pp. 1–16.

[33] D. Hendrycks and T. Dietterich, “A broad study on the transferability of visual representations with contrastive learning,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 8320–8329.

[34] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, “Designing network design spaces,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 15257–15266.

[35] D. Hendrycks et al., “The many faces of robustness: A critical analysis of out-of-distribution generalization,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 8119–8128.

[36] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 15977–16000.

[37] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, “AutoPrompt: Eliciting knowledge from language models with automatically generated prompts,” in Proc. Conf. Empirical Methods Natural Lang. Process. EMNLP, 2020, pp. 4222–4235.

[38] E. J. Hu et al., “Lora: Low-rank adaptation of large language models,” in Proc. ICLR, 2022, pp. 1–26.
X. Mao et al., “Towards robust vision transformer,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2022, pp. 12032–12041.

P. Gao et al., “CLIP-adapter: Better vision-language models with feature adapters,” 2021, arXiv:2110.04544.

K. Zhou, J. Yang, C. C. Loy, and Z. Liu, “Learning to prompt for vision-language models,” Int. J. Comput. Vis., vol. 130, no. 9, pp. 2337–2348, Sep. 2022.

K. Chen et al., “Hybrid task cascade for instance segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4969–4978.

T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, “Unified perceptual parsing for scene understanding,” in Proc. ECCV, 2018, pp. 418–434.

T. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. ECCV, 2014, pp. 740–755.

B. Zhou et al., “Semantic understanding of scenes through the ADE20K dataset,” Int. J. Comput. Vis., vol. 127, no. 3, pp. 302–321, Mar. 2019.

S. Abnar and W. Zuidema, “Quantifying attention flow in transformers,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 4190–4197.

Xing Nie received the B.S. degree in electronic and information engineering from Xidian University, Xi’an, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, and the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. His current research interests include pattern recognition, computer vision, and architecture designing.

Bolin Ni received the B.S. degree in software engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, and the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. His research interests include computer vision, video understanding, and architecture designing.

Jianlong Chang received the B.S. degree from the School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, China, in 2015, and the Ph.D. degree from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2020. He is currently a Research Scientist with Huawei Cloud and AI. His current research interests include relation-based DL, pre-trained model, auto ML, graph networks, and unsupervised learning.

Gaofeng Meng (Senior Member, IEEE) received the B.S. degree in applied mathematics from Northwestern Polytechnical University in 2002, the M.S. degree in applied mathematics from Tianjin University in 2005, and the Ph.D. degree in control science and engineering from Xi’an Jiaotong University in 2009. In the same year, he joined the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, as an Assistant Professor. From May 2015 to July 2017, he was a Visiting Scholar with the Delft University of Technology, The Netherlands. From 2016 to 2017, he was a Visiting Scholar with Northwestern University, Evanston, IL, USA. He is currently a Professor with the Department of State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences. His research interests include document image processing, computer vision, and machine learning. He serves as an Associate Editor for Neurocomputing and Image and Vision Computing.

Chunlei Huo (Member, IEEE) received the B.S. degree in applied mathematics from Hebei Normal University, Shijiazhuang, China, in 1999, the M.S. degree in applied mathematics from Xidian University, Xi’an, China, in 2002, and the Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009. He is currently a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include remote sensing image processing, computer vision, and pattern recognition.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1993, the M.S. degree from Chongqing University, Chongqing, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2004. From 1996 to 2001, he was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a Post-Doctorate Candidate with the Department of Automation, Tsinghua University, Beijing, until 2006. He is currently a Professor with the State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences.

Qi Tian (Fellow, IEEE) received the Ph.D. degree in ECE from the University of Illinois at Urbana-Champaign (UIUC) in 2002. He was the Chief Scientist of computer vision with the Huawei Noah’s Ark Laboratory from 2018 to 2020. He is currently the Chief Scientist of artificial intelligence with Huawei Cloud and AI. Before he joined Huawei Cloud and AI, he was a Full Professor with the Department of Computer Science, The University of Texas at San Antonio (UTSA), from 2002 to 2019.

Dr. Tian has served as a Founding Member for ICMR, from 2009 to 2014, and ACM MM, from 2009 to 2012, and the International Steering Committee Member for ACM MIR, from 2006 to 2010, ACM ICIMCS 2013, ICME 2006 and 2009, PCM 2012, and the IEEE International Symposium on Multimedia 2011. He is a Academician of the International Eurasian Academy of Sciences (IEAS) in 2021. He was listed in the Top Ten of the 2016 Most Influential Scholars in Multimedia by Aminer.org. He received the 2017 UTSA President Distinguished Award for Research Achievement, the 2016 UTSA Innovation Award in the first category, the 2014 Research Achievement Awards from the College of Science, UTSA, and the 2010 Google Faculty Research Award. He served as the Chair for the ACM Multimedia 2015. He is an Associate Editor of IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, ACM Transactions on Multimedia Computing, Communications, and Applications, MMSJ, and Machine Vision and Applications.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.