Visual Query Tuning: Towards Effective Usage of Intermediate Representations for Parameter and Memory Efficient Transfer Learning

Cheng-Hao Tu* Zheda Mai* Wei-Lun Chao
The Ohio State University, {tu.343, mai.145, chao.209}@osu.edu

Abstract

Intermediate features of a pre-trained model have been shown informative for making accurate predictions on downstream tasks, even if the model backbone is kept frozen. The key challenge is how to utilize these intermediate features given their gigantic amount. We propose visual query tuning (VQT), a simple yet effective approach to aggregate intermediate features of Vision Transformers. Through introducing a handful of learnable "query" tokens to each layer, VQT leverages the inner workings of Transformers to "summarize" rich intermediate features of each layer, which can then be used to train the prediction heads of downstream tasks. As VQT keeps the intermediate features intact and only learns to combine them, it enjoys memory efficiency in training, compared to many other parameter-efficient fine-tuning approaches that learn to adapt features and need back-propagation through the entire backbone. This also suggests the complementary role between VQT and those approaches in transfer learning. Empirically, VQT consistently surpasses the state-of-the-art approach that utilizes intermediate features for transfer learning and outperforms full fine-tuning in many cases. Compared to parameter-efficient approaches that adapt features, VQT achieves much higher accuracy under memory constraints. Most importantly, VQT is compatible with these approaches to attain even higher accuracy, making it a simple add-on to further boost transfer learning. Code is available at `https://github.com/andytu28/VQT`.

1. Introduction

Transfer learning by adapting large pre-trained models to downstream tasks has been a de facto standard for competitive performance, especially when downstream tasks have limited data [37, 59]. Generally speaking, there are two ways to adapt a pre-trained model [15, 27]: updating the model backbone for new feature embeddings (the output of the penultimate layer) or recombining the existing feature embeddings, which correspond to the two prevalent approaches, *fine-tuning* and *linear probing*, respectively. *Fine-tuning*, or more specifically, *full fine-tuning*, updates all the model parameters end-to-end based on the new dataset. Although *fine-tuning* consistently outperforms *linear probing* on various tasks [54], it requires running gradient descent for all parameters and storing a separate fine-tuned model for each task, making it computationally expensive and parameter inefficient. These problems become more salient with Transformer-based models whose parameters grow exponentially [17, 26, 46]. Alternatively, *linear probing* only trains and stores new prediction heads to recombine features while keeping the backbone frozen. Despite its computational and parameter efficiency, *linear probing* is often less attractive due to its inferior performance.

Several recent works have attempted to overcome such a dilemma in transfer learning. One representative work is by Evci *et al.* [15], who attributed the success of *fine-tuning* to leveraging the "intermediate" features of pre-trained models and proposed to directly allow *linear probing* to access the intermediate features. Some other works also demonstrated the effectiveness of such an approach [14,15]. Nevertheless, given numerous intermediate features in each layer, most of these methods require pooling to reduce the dimensionality, which likely would eliminate useful information before the prediction head can access it.

To better utilize intermediate features, we propose **Visual Query Tuning (VQT)**, a simple yet effective approach to aggregate the intermediate features of Transformer-based models like Vision Transformers (ViT) [13]. A Transformer usually contains multiple Transformer layers, each starting with a Multi-head self-attention (MSA) module operating over the intermediate feature tokens (often > 100 tokens) outputted by the previous layer. The MSA module transforms each feature token by querying all the other tokens, followed by a weighted combination of their features.

Taking such inner workings into account, **VQT** introduces a handful of *learnable* "query" tokens to each layer, which, through the MSA module, can then "summarize" the intermediate features of the previous layer to reduce the dimensionality. The output features of these query tokens af-

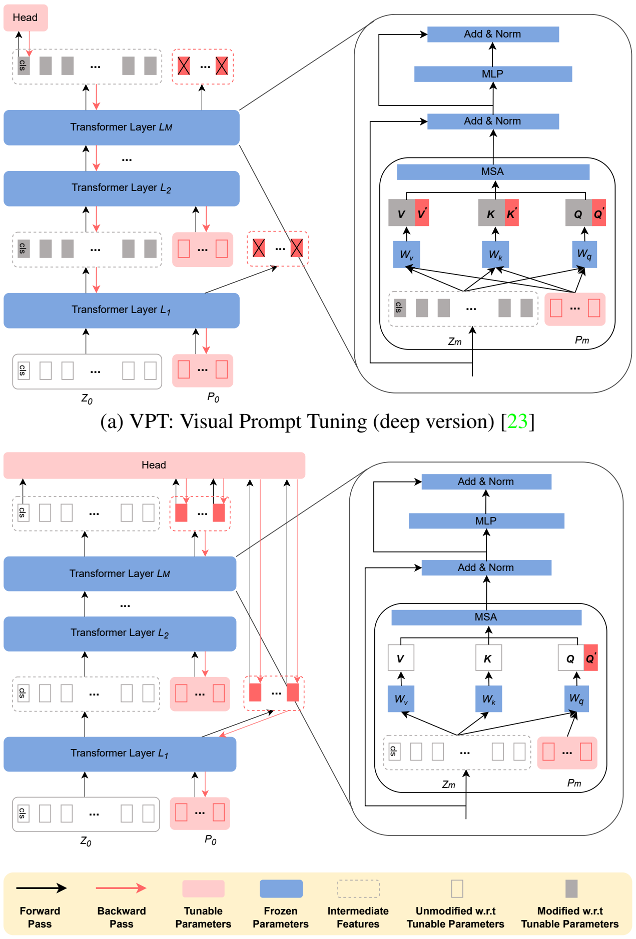
[*] Equal contributions.

ter each layer can then be used by *linear probing* to make predictions. Compared to pooling which simply averages the features over tokens, **VQT** performs a weighted combination whose weights are adaptive, conditioned on the features and the learned query tokens, and is more likely to capture useful information for the downstream task.

At first glance, **VQT** may look superficially similar to Visual Prompt Tuning (VPT) [23], a recent transfer learning method that also introduces additional learnable tokens (*i.e.*, prompts) to each layer of Transformers, but they are fundamentally different in two aspects. First, our **VQT** only uses the additional tokens to generate queries, not keys and values, for the MSA module. Thus, it does not change the intermediate features of a Transformer at all. In contrast, the additional tokens in VPT generate queries, keys, and values, and thus can be queried by other tokens and change their intermediate features. Second, and more importantly, while our **VQT** leverages the corresponding outputs of the additional tokens as summarized intermediate features, VPT in its Deep version disregards such output features entirely. *In other words, these two methods take fundamentally different routes to approach transfer learning:* ***VQT*** *learns to leverage the existing intermediate features, while VPT aims to adapt the intermediate features.* As will be demonstrated in section 4, these two routes have complementary strengths and can be compatible to further unleash the power of transfer learning. It is worth noting that most of the recent methods towards parameter-efficient transfer learning (PETL), such as Prefix Tuning [30] and AdaptFormer [10], all can be considered adapting the intermediate features [19]. Thus, the aforementioned complementary strengths still apply.

Besides the difference in how to approach transfer learning, another difference between **VQT** and many other PETL methods, including VPT, is memory usage in training. While many of them freeze (most of) the backbone model and only learn to adjust or add some parameters, the fact that the intermediate features are updated implies the need of a full back-propagation throughout the backbone, which is memory-heavy. In contrast, **VQT** keeps all the intermediate features intact and only learns to combine them. Learning the query tokens thus bypasses many paths in the standard back-propagation, reducing the memory footprint by 76% compared to VPT.

We validate **VQT** on various downstream visual recognition tasks, using a pre-trained ViT [13] as the backbone. **VQT** surpasses the SOTA method that utilizes intermediate features [15] and *full fine-tuning* in most tasks. We further demonstrate the robust and mutually beneficial compatibility between **VQT** and existing PETL approaches using different pre-trained backbones, including self-supervised and image-language pre-training. Finally, **VQT** achieves much higher accuracy than other PETL methods in a low-memory regime, suggesting that it is a more memory-efficient method.



(a) VPT: Visual Prompt Tuning (deep version) [23]

(b) **Our VQT: Visual Query Tuning**

Figure 1. **Our Visual Query Tuning (VQT) vs. Visual Prompt Tuning (VPT) [23].** Our **VQT** allows *linear probing* to directly access the intermediate features of a frozen Transformer model for parameter-efficient transfer learning. The newly introduced query tokens in **VQT** (marked by the red empty boxes in the red shaded areas) only append additional columns (*i.e.*, $\boldsymbol{Q'}$) to the Query features $\boldsymbol{Q}$, not to the Value features $\boldsymbol{V}$ and the Key features $\boldsymbol{K}$. Thus, VQT keeps the intermediate features intact (gray empty boxes), enabling it to bypass expensive back-propagation steps in training (hence memory efficient). In contrast, VPT modifies the intermediate features (gray solid boxes) and needs more memory to learn its prompts. Please see section 3 for details.

To sum up, our key contributions are

1. We propose **VQT** to aggregate intermediate features of Transformers for effective linear probing, featuring parameter and memory efficient transfer learning.
2. **VQT** is compatible with other PETL methods that adapt intermediate features, further boosting the performance.
3. **VQT** is robust to different pre-training setups, including self-supervised and image-language pre-training.

2. Related Work

Transformer. The splendent success of Transformer models [46] in natural language processing (NLP) [48] has sparked a growing interest in adopting these models in vision and multi-modal domains [26]. Since the proposal of the Vision Transformer (ViT) [13], Transformer-based methods have demonstrated impressive advances in various vision tasks, including image classification [35, 44, 51], image segmentation [40, 49], object detection [7, 58], video understanding [2, 36], point cloud processing [16, 56], and several other use cases [9, 50]. As Transformer models assume minimal prior knowledge about the structure of the problem, they are often pre-trained on large-scale datasets [8, 11, 39]. Given that the Transformer models are notably larger than their convolutional neural network counterparts, e.g., ViT-G (1843M parameters) [53] vs. ResNet-152 (58M parameters) [21], how to adapt the pre-trained Transformers to downstream tasks in a parameter and memory efficient way remains a crucial open problem.

PETL. The past few years have witnessed the huge success of parameter-efficient transfer learning (PETL) in NLP, aiming to adapt large pretrained language models (PLMs) to downstream tasks [6, 25]. Typically, PETL methods insert small learnable modules into PLMs and fine-tune these modules with downstream tasks while freezing the pre-trained weights of PLMs [3,5,19,22,29,33,38,41,43,47,57]. The current dominance of Transformer models in the vision field has urged the development of PETL methods in ViT [10, 23, 24, 31, 34, 55]. Recently, Visual Prompt Tuning (VPT) [23] was proposed to prepend learnable prompts to the input embeddings of each Transformer layer. AdaptFormer [10] inserts a bottleneck-structured fully connected layers parallel to the MLP block in a Transformer layer. Convpass [24] inserts a convolutional bottleneck module while NOAH [55] performs a neural architecture search on existing PETL methods. Unlike all the aforementioned methods that update the output features of each Transformer layer, our VQT focuses on leveraging the frozen intermediate features. Thus, VQT is compatible with most existing PETL methods and enjoys memory efficiency.

Transfer learning with intermediate features. Intermediate features of a pre-trained model contain rich and valuable information, which can be leveraged in various tasks such as object detection [4, 18, 32] and OOD detection [28], etc. Recently, multiple works [12, 14, 15, 42] have demonstrated the effectiveness of these features on transfer learning. On the NLP side, LST [42] trains a lightweight Transformer network that takes intermediate features as input and generates output features for predictions. On the CV side, Evci *et al.* [15] attribute the success of fine-tuning to the ability to leverage intermediate features and proposed Head2Toe to select features from all layers for efficient transfer learning. Eom *et al.* [14] proposed utilizing intermediate features to facilitate transfer learning for multi-label classification. However, due to the massive number of intermediate features, most methods rely on the pooling operation to reduce the dimensionality, which may distort or eliminate useful information. This observation motivates us to introduce VQT, which learns to summarize intermediate features according to the downstream task.

3. Approach

We propose **Visual Query Tuning (VQT)** to adapt pre-trained Transformers to downstream tasks while keeping the backbone frozen. VQT keeps all the intermediate features intact and only learns to "summarize" them for *linear probing* by introducing learnable "query" tokens to each layer.

3.1. Preliminaries

3.1.1 Vision Transformer

Vision Transformers (ViT) [13] adapt the Transformer-based models [46] from NLP into visual tasks, by dividing an image $\boldsymbol{I}$ into a sequence of N fixed-sized patches $\{\boldsymbol{I}^{(n)}\}_{n=1}^{N}$ and treating them as NLP tokens. Each patch $\boldsymbol{I}^{(n)}$ is first embedded into a D-dimensional vector $\boldsymbol{x}_0^{(n)}$ with positional encoding. The sequence of vectors is then prepended with a "CLS" vector $\boldsymbol{x}_0^{(\text{Class})}$ to generate the input $\boldsymbol{Z}_0 = [\boldsymbol{x}_0^{(\text{Class})}, \boldsymbol{x}_0^{(1)}, \cdots, \boldsymbol{x}_0^{(N)}] \in \mathbb{R}^{D\times(1+N)}$ to the ViT. We use superscript/subscript to index token/layer.

Normally, a ViT has M layers, denoted by $\{L_m\}_{m=1}^{M}$. Given the input $\boldsymbol{Z}_0$, the first layer L_1 generates the output $\boldsymbol{Z}_1 = L_1(\boldsymbol{Z}_0) = [\boldsymbol{x}_1^{(\text{Class})}, \boldsymbol{x}_1^{(1)}, \cdots, \boldsymbol{x}_1^{(N)}] \in \mathbb{R}^{D\times(N+1)}$, which is of the same size as $\boldsymbol{Z}_0$. That is, $\boldsymbol{Z}_1$ has $1 + N$ feature tokens, and each corresponds to the same column in $\boldsymbol{Z}_0$. Such layer-wise processing then continues to generate the output of the next layer, $\boldsymbol{Z}_m = L_m(\boldsymbol{Z}_{m-1})$ for $m = 2, \cdots, M$, taking the output of the previous layer as input. Finally, the "CLS" vector $\boldsymbol{x}_M^{(\text{Class})}$ in $\boldsymbol{Z}_M$ is used as the feature for prediction. Taking classification as an example, the predicted label $\hat{y} = \mathsf{Head}(\boldsymbol{x}_M^{(\text{Class})})$ is generated by a linear classifier (*i.e.*, a fully-connected layer).

Details of each Transformer layer. Our approach takes advantage of the inner workings of Transformer layers. In the following, we provide a concise background.

Each Transformer layer consists of a Multi-head Self-Attention (MSA) block, a Multi-Layer Perceptron (MLP) block, and several other operations including layer normalization and residual connections. Without loss of generality, let us consider a single-head self-attention block and disregard those additional operations.

Given the input $\boldsymbol{Z}_{m-1}$ to L_m, the self-attention block first projects it into three matrices, namely Query $\boldsymbol{Q}_m$, Key

$\boldsymbol{K}_m$, and Value $\boldsymbol{V}_m$,

$$\boldsymbol{Q}_m = \boldsymbol{W}_q \boldsymbol{Z}_{m-1}, \quad \boldsymbol{K}_m = \boldsymbol{W}_k \boldsymbol{Z}_{m-1}, \quad \boldsymbol{V}_m = \boldsymbol{W}_v \boldsymbol{Z}_{m-1}. \tag{1}$$

Each of them has $1+N$ columns[1], corresponding to each column (*i.e.*, token) in $\boldsymbol{Z}_{m-1}$. Then, the output of L_m, *i.e.*, $\boldsymbol{Z}_m$, can be calculated by:

$$\boldsymbol{Z}_m = \mathsf{MLP}_m \circ \mathsf{MSA}_m(\boldsymbol{Z}_{m-1}), \tag{2}$$

$$\text{where } \mathsf{MSA}_m(\boldsymbol{Z}_{m-1}) = \boldsymbol{V}_m \times \mathsf{Softmax}(\frac{\boldsymbol{K}_m^\top \boldsymbol{Q}_m}{\sqrt{D}}). \tag{3}$$

The $\mathsf{Softmax}$ is taken over elements of each column; the MLP_m is applied to each column of $\mathsf{MSA}_m(\boldsymbol{Z}_{m-1})$ independently.

3.1.2 Transfer Learning: Linear Probing, Fine-tuning, and Intermediate Feature Utilization

To adapt a pre-trained ViT to downstream tasks, *linear probing* freezes the whole backbone model but the prediction head: it disregards the original Head and learns a new one. *Fine-tuning*, on top of *linear probing*, allows the backbone model to be updated as well.

Several recent works have demonstrated the effectiveness of utilizing intermediate features in transfer learning, by allowing *linear probing* to directly access them [14, 15]. The seminal work HEAD2TOE [15] takes intermediate features from $\boldsymbol{Z}_0$ and four distinct steps in each Transformer layer: features after the layer normalization, after the MSA block, and inside and after the MLP block. Since each of them has $1+N$ tokens, HEAD2TOE groups tokens by their indices and performs average pooling to reduce the dimensionality. The resulting features — over each group, step, and layer — are then concatenated together for *linear probing*. To further reduce dimensionality, HEAD2TOE employs group lasso [1, 52] for feature selection.

We note that while the second dimensionality reduction is driven by downstream tasks, the first (*i.e.*, pooling) is not, which may inadvertently eliminate useful information. This shortcoming motivates us to develop Visual Query Tuning (VQT) for the effective usage of intermediate features.

3.2. Visual Query Tuning (VQT)

We propose to replace the average pooling operation in HEAD2TOE with the intrinsic "summarizing" mechanism in Transformers. We note that the MSA block introduced in Equation 3 essentially performs weighted averages of the Value features $\boldsymbol{V}$ over tokens, in which the weights are determined by the columns of $\boldsymbol{K}^\top \boldsymbol{Q}$. That is, if we can append additional "columns" to $\boldsymbol{K}^\top \boldsymbol{Q}$, the MSA block will output additional weighted combinations of $\boldsymbol{V}$. In the special case that the appended vector to $\boldsymbol{K}^\top \boldsymbol{Q}$ has identical entries (*e.g.*, an all-zero vector), the weighted average reduces to a simple average. In other words, average pooling can be thought of as a special output of the MSA layer.

Taking this insight into account, we propose to learn and append additional columns $\boldsymbol{Q}'$ to $\boldsymbol{Q}$. We realize this idea by introducing a handful of T learnable "query" tokens $\boldsymbol{P}_{m-1} = [\boldsymbol{p}_{m-1}^{(1)}, \cdots, \boldsymbol{p}_{m-1}^{(T)}]$ to the input of each Transformer layer L_m. See Figure 1b for an illustration. Different from the original input $\boldsymbol{Z}_{m-1}$ that undergoes the three projections introduced in Equation 1, $\boldsymbol{P}_{m-1}$ only undergoes the projection by $\boldsymbol{W}_q$,

$$\boldsymbol{Q}'_m = \boldsymbol{W}_q \boldsymbol{P}_{m-1}. \tag{4}$$

By appending $\boldsymbol{Q}'_m$ to $\boldsymbol{Q}_m$ column-wise, we modify the computation of the original MSA block in Equation 3 by

$$\boldsymbol{V}_m \times \mathsf{Softmax}(\frac{\boldsymbol{K}_m^\top [\boldsymbol{Q}_m, \boldsymbol{Q}'_m]}{\sqrt{D}}) = \tag{5}$$
$$[\boldsymbol{V}_m \times \mathsf{Softmax}(\frac{\boldsymbol{K}_m^\top \boldsymbol{Q}_m}{\sqrt{D}}), \boldsymbol{V}_m \times \mathsf{Softmax}(\frac{\boldsymbol{K}_m^\top \boldsymbol{Q}'_m}{\sqrt{D}})].$$

The second half (blue color) corresponds to the newly summarized MSA features by the learnable query tokens $\boldsymbol{P}_{m-1}$. Then after the MLP block MLP_m, these features lead to the newly summarized features $\boldsymbol{Z}'_m \in \mathbb{R}^{D\times T}$ from layer L_m. We can then concatenate these newly summarized features over layers, $\boldsymbol{Z}'_m \in \mathbb{R}^{D\times T}$ for $m = 1, \cdots, M$, together with the final "CLS" vector $\boldsymbol{x}_M^{\text{(Class)}}$, for *linear probing*. We name our approach **Visual Query Tuning (VQT)**, reflecting the fact that the newly added tokens $\boldsymbol{P}_m$ for $m = 0, \cdots, M-1$ only serve for the additional columns in Query matrices.

Properties of VQT. As indicated in Equation 5, the newly introduced query tokens do not change the MSA features the pre-trained ViT obtains (*i.e.*, the first half). This implies that VQT keeps all the original intermediate features (*e.g.*, $\boldsymbol{Z}_m$) intact but only learns to recombine them.

Training of VQT. Given the training data of the downstream task, the query tokens $\{\boldsymbol{P}_m\}_{m=0}^{M-1}$ are learned end-to-end with the new prediction head, which directly accesses the outputs $\{\boldsymbol{Z}'_{m+1}\}_{m=0}^{M-1}$ of these query tokens.

To further reduce the dimensionality of $\{\boldsymbol{Z}'_{m+1}\}_{m=0}^{M-1}$, we optionally employ group lasso, following HEAD2TOE [15]. In detail, we first learn the query tokens without group lasso. We then freeze them and apply group lasso to select useful features from $\{\boldsymbol{Z}'_{m+1}\}_{m=0}^{M-1}$. We also explored various ways for dimension reduction in Appendix C.4.

3.3. Comparison to Related Works

Comparison to HEAD2TOE [15]. We list several key differences between HEAD2TOE and our VQT. First, compared to HEAD2TOE, which takes intermediate features

[1] For brevity, we ignore the layer index m for the projection matrices $\boldsymbol{W}_q, \boldsymbol{W}_k, \boldsymbol{W}_v$, but each layer has its own projection matrices.

from multiple steps in a Transformer layer, VQT only takes the newly summarized intermediate features after each layer. Second, and more importantly, VQT employs a different way to combine intermediate features *across tokens*. Generally speaking, there are two ways to combine a set of feature vectors $\{\boldsymbol{x}^{(n)} \in \mathbb{R}^D\}_{n=1}^N$: concatenation and average pooling. The former assumes that different vectors have different meanings even at the same dimension, which is suitable for features across layers. The latter assumes that the same dimension means similarly to different vectors so they can be compared and averaged, which is suitable for features across tokens. One particular drawback of the former is the dimensionality (*i.e*., inefficiency). For the latter, it is the potential loss of useful information since it combines features blindly to the downstream tasks (*i.e*., ineffectiveness). HEAD2TOE takes a mix of these two ways to combine features over tokens, and likely suffers one (or both) drawbacks. In contrast, VQT leverages the intrinsic mechanism of self-attention to aggregate features adaptively, conditioned on the features and the learnable query tokens, making it a more efficient and effective way to tackle the numerous intermediate features within each layer.

Comparison to Visual Prompt Tuning (VPT). At first glance, VQT may be reminiscent of VPT [23], but they are fundamentally different as highlighted in section 1 and Figure 1. Here, we provide some more details and illustrations.

VPT in its deep version (VPT-Deep) introduces learnable tokens $\boldsymbol{P}_{m-1} = [\boldsymbol{p}_{m-1}^{(1)}, \cdots, \boldsymbol{p}_{m-1}^{(T)}]$ to the input of each Transformer layer L_m, similarly to VQT. However, unlike VQT which uses $\boldsymbol{P}_{m-1}$ only for querying, VPT-Deep treats $\boldsymbol{P}_{m-1}$ the same as other input tokens $\boldsymbol{Z}_{m-1}$ and generates the corresponding Query, Key, and Value matrices,

$$\boldsymbol{Q}'_m = \boldsymbol{W}_q\boldsymbol{P}_{m-1}, \quad \boldsymbol{K}'_m = \boldsymbol{W}_k\boldsymbol{P}_{m-1}, \quad \boldsymbol{V}'_m = \boldsymbol{W}_v\boldsymbol{P}_{m-1}.$$

These matrices are then appended to the original ones from $\boldsymbol{Z}_{m-1}$ (cf. Equation 1) before self attention,

$$\tilde{\boldsymbol{Q}}_m = [\boldsymbol{Q}_m, \boldsymbol{Q}'_m], \ \ \tilde{\boldsymbol{K}}_m = [\boldsymbol{K}_m, \boldsymbol{K}'_m], \ \ \tilde{\boldsymbol{V}}_m = [\boldsymbol{V}_m, \boldsymbol{V}'_m],$$

making the output of the MSA block as

$$\begin{aligned} &\tilde{\boldsymbol{V}}_m \times \mathsf{Softmax}(\frac{\tilde{\boldsymbol{K}}_m^\top \tilde{\boldsymbol{Q}}_m}{\sqrt{D}}) = \qquad (6) \\ &[\tilde{\boldsymbol{V}}_m \times \mathsf{Softmax}(\frac{\tilde{\boldsymbol{K}}_m^\top \boldsymbol{Q}_m}{\sqrt{D}}), \tilde{\boldsymbol{V}}_m \times \mathsf{Softmax}(\frac{\tilde{\boldsymbol{K}}_m^\top \boldsymbol{Q}'_m}{\sqrt{D}})]. \end{aligned}$$

Compared to Equation 3 and Equation 5, the first half of the matrix in Equation 6 changes, implying that all the intermediate features as well as the final "CLS" vector $\boldsymbol{x}_M^{(\text{Class})}$ are updated according to the learnable tokens $\boldsymbol{P}_{m-1}$. In contrast, VQT keeps these (intermediate) features intact.

Perhaps more subtly but importantly, VPT-Deep ends up dropping the second half of the matrix in Equation 6. In other words, VPT-Deep does not exploit the newly summarized features by $\boldsymbol{Q}'_m$ at all, making it conceptually similar to Prefix Tuning [30]. Please see Figure 1 for a side-by-side comparison between VQT and VPT-Deep.

The aforementioned differences suggest an interesting distinction between VQT and VPT: *VQT learns to leverage the existing intermediate features, while VPT learns to adapt the intermediate features.* In subsection 4.3, we demonstrate one particular strength of VQT, which is to transfer self-supervised pre-trained models.

Comparison and Compatibility with PETL methods. In fact, most of the existing PETL approaches that adjust or add a small set of parameters to the backbone model update the intermediate features [19]. Thus, our VQT is likely complementary to them and can be used to boost their performance. In subsection 4.3, we explore this idea by introducing learnable query tokens to these methods.

Memory efficiency in training. As pointed out in [42], when learning the newly added parameters, most PETL methods require storing intermediate back-propagation results, which is memory-inefficient for large Transformer-based models. For VQT, since it keeps all the intermediate features intact and only learns to (i) tune the query tokens (ii) and *linearly probe* the corresponding outputs of them, the training bypasses many expensive back-propagation paths, significantly reducing the memory footprint. See subsection 4.4 for details.

4. Experiments

4.1. Experiment Setup

Dataset. We evaluate the transfer learning performance on the **VTAB-1k** [54], which consists of 19 image classification tasks categorized into three groups: Natural, Specialized, and Structured. The Natural group comprises natural images captured with standard cameras. The Specialized group contains images captured by specialist equipment for remote sensing and medical purpose. The Structured group evaluates the scene structure comprehension, such as object counting and 3D depth estimation. Following [54], we perform an 80/20 split on the **1000** training images in each task for hyperparameter searching. The reported result (top-1 classification accuracy) is obtained by training on the 1000 training images and evaluating on the original test set.

Pre-training setup. We use ViT-B/16 [13] as the backbone. The pre-training setup follows the corresponding compared baselines. When comparing with Head2Toe, we use ImageNet-1K supervised pre-trained backbone. When investigating the compatibility with other PETL methods, ImageNet-21K supervised pre-trained backbone is used. To demonstrate the robustness of VQT to different pre-training setups, we also evaluate VQT on self-supervised (MAE) [20] and image-language (CLIP) pre-trained [39]

	Natural								Specialized					Structured									
Method	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Mean	Camelyon	EuroSAT	Resisc45	Retinopathy	Mean	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Elev	Mean	Overall Mean
Scratch	7.6	19.1	13.1	29.6	6..7	19.4	2.3	14.0	71.0	71.0	29.3	72.0	60.8	31.6	52.5	27.2	39.1	66.1	29.7	11.7	24.1	35.3	32.8
Linear-probing	50.6	85.6	61.4	79.5	86.5	40.8	38.0	63.2	79.7	91.5	71.7	65.5	77.1	41.4	34.4	34.1	55.4	18.1	26.4	16.5	24.8	31.4	52.7
Fine-tuning	44.3	84.5	54.1	84.7	74.7	**87.2**	26.9	65.2	**85.3**	95.0	76.0	70.4	81.7	**71.5**	**60.5**	**46.9**	72.9	**74.5**	38.7	28.5	23.8	**52.2**	63.2
HEAD2TOE	54..4	86.8	64.1	83.4	82.6	78.9	32.1	68.9	81.3	95.4	81.2	73.7	82.9	49.0	57.7	41.5	64.4	52.3	32.8	**32.7**	**39.7**	46.3	62.3
VQT (Ours)	**58.4**	**89.4**	**66.7**	**90.4**	**89.1**	81.1	**33.7**	**72.7**	82.2	**96.2**	**84.7**	**74.9**	**84.5**	50.8	57.6	43.5	**77.2**	65.9	**43.1**	24.8	31.6	49.3	**65.3**

Table 1. Test accuracy on the VTAB-1k benchmark with ViT-B/16 pre-trained on ImageNet-1K. ”Mean” denotes the average accuracy for each category and ”Overall Mean” shows the average accuracy over 19 tasks.

Methods	Natural	Specialized	Structured
	CLIP backbone		
AdaptFormer	82.6	85.1	60.9
AdaptFormer+VQT	82.1 0.5 ↓	85.8 0.7 ↑	62.6 1.7 ↑
VPT	80.4	84.9	50.9
VPT+VQT	81.5 1.1 ↑	86.3 1.4 ↑	57.2 6.3 ↑
	MAE backbone		
AdaptFormer	68.7	81.3	58.3
AdaptFormer+VQT	71.1 2.4 ↑	83.3 2.0 ↑	59.2 0.9 ↑
VPT	63.5	79.1	48.6
VPT+VQT	67.9 4.4 ↑	82.7 3.6 ↑	49.7 1.1 ↑
	Supervised ImageNet-21K backbone		
AdaptFormer	80.1	82.3	50.3
AdaptFormer+VQT	79.6 0.5 ↓	84.3 2.0 ↑	53.0 2.7 ↑
VPT	79.1	84.6	54.4
VPT+VQT	78.9 0.2 ↓	83.7 0.9 ↓	54.6 0.2 ↑

Table 2. **Compatibility of VQT** with AdaptFormer and VPT on MAE, CLIP, and supervised pre-trained backbones.

backbones. Please see Appendix B for more details.

4.2. Effectiveness of VQT

To evaluate the transfer learning performance of VQT, we compare VQT with methods that fix the whole backbone (*linear-probing* and HEAD2TOE) and full *fine-tuning*, which updates all network parameters end to end. For a fair comparison, **we match the number of tunable parameters in VQT with that in HEAD2TOE** (details are included in Appendix B.3). In general, VQT improves over *linear-probing* by 12.6% and outperforms HEAD2TOE and full *fine-tuning* by 3% and 2.1% respectively, on average performance over 19 tasks, which demonstrates **the strength of using intermediate features and the effectiveness of VQT in summarizing them**. In the Natural category, VQT surpasses HEAD2TOE and *fine-tuning* by 2.8% and 7.5%, respectively, and outperforms them in the Specialized category by 1.6% and 2.8%, respectively. As shown in [15, 54], the Natural and Specialized categories have stronger domain affinities with the source domain (ImageNet) since they are all real images captured by cameras. Thus, the pre-trained backbone can generate more relevant intermediate features for similar domains. The only exception is the Structured category consisting of rendered artificial images from simulated environments, which differs significantly from ImageNet. Although VQT continues to improve HEAD2TOE, *fine-tuning* shows 2.9% enhancement over VQT, suggesting that if we need to adapt to a more different targeted domain, we may consider tuning a small part of the backbone to produce updated features for new data before applying our VQT techniques. Appendix C.1 contains more comparisons between HEAD2TOE and VQT.

4.3. Compatibility with PETL Methods in Different Pre-training Methods

As mentioned in subsection 3.3, most existing PETL methods and VQT take fundamentally different routes to approach transfer learning: PETL methods focus on adapting the model to generate updated features, while VQT aims to better leverage features. Building upon this conceptual complementariness, we investigate if they can be combined to unleash the power of transfer learning. Moreover, in order to demonstrate the robustness of the compatibility, we evaluate performance on three different pre-trained backbones: self-supervised pre-trained (MAE with ImageNet-1K) [20], image-language pre-trained (CLIP) [39] and supervised pre-trained (ImageNet-21K).

Specifically, we focus on two recently proposed methods: AdaptFormer [10] and VPT [23]. AdaptFormer inserts fully connected layers in a bottleneck structure parallel to the MLP block in each Transformer layer [10]; VPT adds learnable tokens to the input of every Transformer layer.

To equip AdaptFormer [10] and VPT [23] with our VQT, firstly, we update the pre-trained model with AdaptFormer or VPT so that the model can generate relevant intermediate features for the downstream task. Then we add $T = 1$ query token to the input of every layer to summarize the updated intermediate features. For AdaptFormer, we use the default bottleneck dimension 64; for VPT, we use the best number of added tokens for each task reported in their paper.

We summarize the results in Table 2, where each row shows the results for one pre-trained backbone, and each column shows the results for one data category. Generally speaking, AdaptFormer and VPT benefit from VQT in most of the scenarios across different data categories and pre-

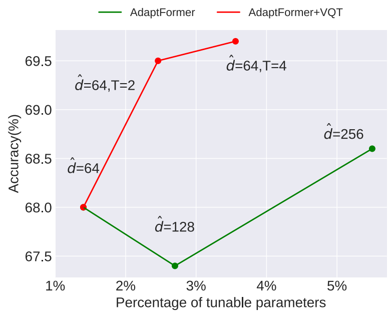


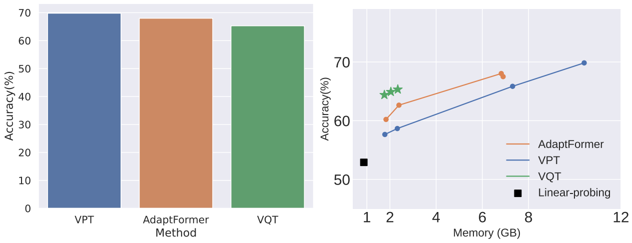
Figure 2. The power of leveraging intermediate features provided by VQT allows AdaptFormer to incorporate additional information from the updated model (red curve), which would not be possible by simply increasing the complexity of the inserted modules (green curve). $\hat{d}$ denotes the bottleneck dimension of AdaptFormer and T represents the number of VQT's query tokens.

Methods	Natural	Specialized	Structured
Linear-probing	18.87	53.72	23.70
Fine-tuning	59.29	79.68	53.82
VPT	63.50	79.15	48.58
VQT (Our)	66.00	82.87	52.64

Table 3. Average accuracy on VTAB-1k using the MAE backbone.

trained backbones. **The improvement is more salient in the MAE backbone.** Since the MAE pre-training uses the reconstruction objective instead of the classification or contrastive one, we hypothesize that some useful intermediate features for classification may not be propagated to the final layer[2]. With the help of VQT, AdaptFormer and VPT can leverage intermediate features in a more concise and effective way. Additionally, **VQT also benefits from AdaptFormer and VPT.** In subsection 4.2, we found that directly applying VQT to the pre-trained backbone may not be effective for the Structured category due to the low domain affinity. With the intermediate features updated by AdaptFormer and VPT, VQT can summarize these more relevant features to improve the results for the Structured group. To sum up, the experiment results illustrate that **VQT and PETL methods are complementary and mutually beneficial**, with the potential to further unleash the power of transfer. We provide detailed results of various pre-trained backbones in Appendix C.6 and the compatibility comparison between HEAD2TOE and VQT in Appendix C.7.

To confirm that the improvement mentioned above does not simply come from the increase of tunable parameters, we enlarge AdaptFormer's added modules by increasing the bottleneck dimension $\hat{d}$ from 64 to 128 and 256 to match the tunable parameter number of AdaptFormer when it is equipped with VQT[3]. As shown in Figure 2, AdaptFormer with VQT significantly outperforms AdaptFormer with larger added modules when the numbers of tunable parameters are similar. This further demonstrates the complementary strength of VQT and AdaptFormer: the improvement by leveraging intermediate features summarized by VQT cannot be achieved by simply increasing the complexity of the inserted modules in AdaptFormer.



(a) (b)

Figure 3. **Comparison under memory constraints.** (a) Without constraints, VPT and AdaptFormer slightly outperform VQT. (b) With constraints, VQT performs the best in low-memory regimes.

4.4. Memory Efficient Training

While many PETL methods reduce the number of tunable parameters, they cannot cut down the memory footprint during training by much, and therefore, the evaluation of PETL methods often ignores memory consumption. In real-world scenarios, however, a model is often required to adapt to new data on edge devices for privacy concerns, necessitating the need for methods that can be trained with limited memory. This motivates us to further analyze the accuracy-memory trade-off for VPT, AdaptFormer, and VQT.

As discussed in subsection 3.3, VPT and AdaptFormer require storing the intermediate back-propagation results to update their added parameters, while VQT bypasses the expensive back-propagation because it keeps all the intermediate features intact. To evaluate their performance in the low-memory regime, we only add their inserted parameters to the last few layers to match the memory usage. Figure 3a shows the performance of VQT, VPT, and AdaptFormer under their **best hyperparameters without memory constraints**; Figure 3b depicts the **accuracy-memory tradeoff** for these methods. When memory is not a constraint, VPT and AdaptFormer slightly outperform VQT, but they consume 3.8x and 5.9x more memory (GB) than VQT, respectively, as we can see in Figure 3b.

When memory is a constraint (left side of Figure 3b), we see drastic accuracy drops of AdaptFormer and VPT. Although they still surpass *linear-probing*, VQT outperforms them significantly, suggesting that VQT is a more memory-efficient method thanks to its query-only mechanism.

4.5. Discussion

Layer importance for each category. As VQT leverages the summarized intermediate features for predictions, we

[2] Table 3 shows the transfer learning results by each method alone, using the MAE backbone. Our VQT notably outperforms other methods.

[3] For VPT, since we already use its best prompt sizes, adding more prompts to it will not improve its performance.

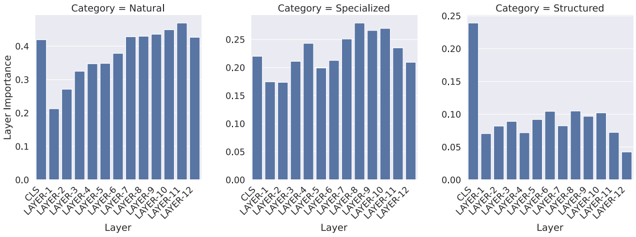


Figure 4. Layer importance for each category in VTAB-1k.

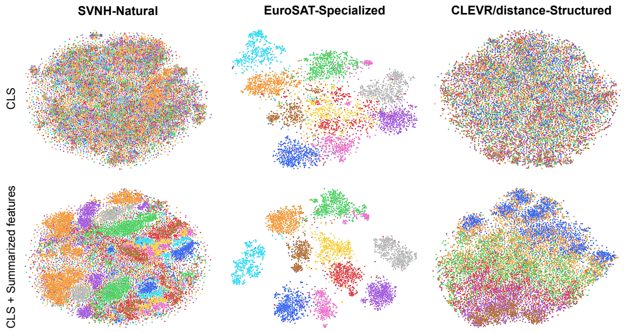
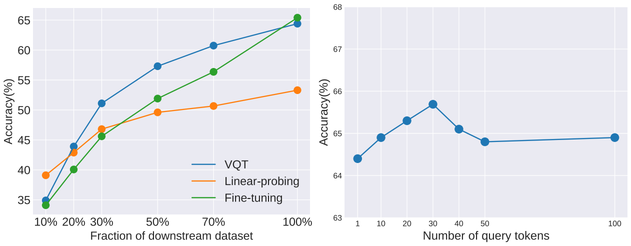


Figure 5. **t-SNE visualization of the CLS tokens alone (top) and CLS tokens plus our summarized features (bottom)** on 3 tasks from each VTAB's category. Adding the summarized intermediate features makes the whole features more separable.



(a)

(b)

Figure 6. (a) Average accuracy over the 19 tasks in VTAB-1k using different training data sizes. For each task, 100% means that we use all the 1000 training images. In the 10% data case, we averagely have only 2 images per class. (b) Average accuracy on VTAB-1k using different numbers of query tokens for VQT.

investigate which layers produce more critical features for each category. In Figure 4, we show each layer's importance score computed by averaging the feature importance in the layer. Features in deeper layers are more important for the Natural category, while features from all layers are almost equally important for the Specialized category. Contrastingly, VQT heavily relies on the CLS token for the Structured category. We hypothesize that the low domain affinity between ImageNet and the Structured category may cause the intermediate features to be less relevant, and the model needs to depend more on the CLS token.

Different downstream data sizes. We further study the effectiveness of VQT under various training data sizes. We reduce the VTAB's training sizes to $\{10\%, 20\%, 30\%, 50\%, 70\%\}$ and compare VQT with Fine-tuning and Linear probing in Figure 6a. Although fine-tuning slightly outperforms VQT on 100% data, VQT consistently performs better as we keep reducing the training data. On the 10% data case, where we only have 2 images per class on average, Linear probing obtains the best accuracy, but its improvement diminishes and performs much worse than VQT when more data become available. These results show that VQT is more favorable in a wide range of training data sizes.

Number of query tokens. As we use only *one* query token for VQT in previous experiments, we now study VQT's performance using more query tokens on VTAB-1k. Figure 6b shows that more query tokens can improve VQT, but the accuracy drops when we add more than 40 tokens. We hypothesize that overly increasing the model complexity causes overfitting due to the limited data in VTAB-1k.

Visualization. Figure 5 shows t-SNE [45] visualization of the CLS token and our summarized features for three tasks (SVHN, EuroSAT, and Clevr-Dist), one from each category. Compared with the CLS token alone, adding summarized features makes the whole features more separable, showing the strength of using intermediate features and the effectiveness of our query tokens in summarizing them. We provide the visualization of other tasks in Appendix C.5.

5. Conclusion

We introduced Visual Query Tuning, a simple yet effective approach to aggregate intermediate features of Vision Transformers. By introducing a set of learnable "query" tokens to each layer, VQT leverages the intrinsic mechanism of Transformers to "summarize" rich intermediate features while keeping the intermediate features intact, which allows it to enjoy a memory-efficient training without back-propagation through the entire backbone. Empirically, VQT surpasses HEAD2TOE, the SOTA method that utilizes intermediate features, and we demonstrate robust and mutually beneficial compatibility between VQT and other PETL methods. Furthermore, VQT is a more memory-efficient approach and achieves much higher performance in a low-memory regime. While VQT only focuses on summarizing features within each layer, we hope our work can pave the way for exploring more effective ways of using features across layers and leveraging intermediate features in transfer learning for other tasks, such as object detection, semantic segmentation and video classification.

Acknowledgments

This research is supported in part by NSF (IIS-2107077, OAC-2118240, and OAC-2112606) and Cisco Research. We are thankful for the computational resources of the Ohio Supercomputer Center. We thank Yu Su (OSU) for the helpful discussion. We thank Menglin Jia and Luming Tang (Cornell) for code sharing.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19, 2006. 4

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021. 3

[3] Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attentional mixtures of soft prompt tuning for parameter-efficient multi-task knowledge sharing. *arXiv preprint arXiv:2205.11961*, 2022. 3

[4] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016. 3

[5] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9. Association for Computational Linguistics, 2022. 3

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 3

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 3

[9] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 3

[10] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*, 2022. 2, 3, 6, 14

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. 3

[12] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, 2020. 3

[13] 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. 1, 2, 3, 5, 12

[14] Seongha Eom, Taehyeon Kim, and Se-Young Yun. Layover intermediate layer for multi-label classification in efficient transfer learning. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022. 1, 3, 4

[15] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2toe: Utilizing intermediate representations for better transfer learning. In *International Conference on Machine Learning*, pages 6009–6033. PMLR, 2022. 1, 2, 3, 4, 6, 12, 13, 14

[16] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 3

[17] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022. 1

[18] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015. 3

[19] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2021. 2, 3, 5

[20] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 5, 6, 12

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[22] Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, et al. Hyperprompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, pages 8678–8690. PMLR, 2022. 3

[23] 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, 5, 6, 12, 14

[24] Shibo Jie and Zhi-Hong Deng. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039*, 2022. 3

[25] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional trans-

formers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019. 3, 12

[26] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022. 1, 3

[27] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019. 1

[28] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018. 3

[29] 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. Association for Computational Linguistics, 2021. 3

[30] 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, 2021. 2, 5

[31] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *arXiv preprint arXiv:2210.08823*, 2022. 3

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[33] 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, 2022. 3

[34] Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *arXiv preprint arXiv:2210.03265*, 2022. 3

[35] 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*, pages 10012–10022, 2021. 3

[36] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3202–3211, 2022. 3

[37] Ying Lu, Lingkun Luo, Di Huang, Yunhong Wang, and Liming Chen. Knowledge transfer in vision recognition: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–35, 2020. 1

[38] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. UniPELT: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264. Association for Computational Linguistics, 2022. 3

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 3, 5, 6, 12

[40] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7262–7272, 2021. 3

[41] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, et al. On transferability of prompt tuning for natural language processing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969, 2022. 3

[42] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *arXiv preprint arXiv:2206.06522*, 2022. 3, 5

[43] Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021. 3

[44] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 3

[45] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 8

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 3

[47] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059. Association for Computational Linguistics, 2022. 3

[48] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020. 3

[49] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and

efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021. 3

[50] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5791–5800, 2020. 3

[51] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022. 3

[52] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006. 4

[53] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022. 3

[54] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 1, 5, 6

[55] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint arXiv:2206.04673*, 2022. 3

[56] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 3

[57] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *arXiv preprint arXiv:2208.10160*, 2022. 3

[58] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020. 3

[59] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 1