# A Implementation Details

We use PyTorch [35] to implement all experiments on NVIDIA A100-40GB GPUs.

## A.1 Classification Experiments

**VPT.** We use `val` set of each dataset to find best prompt length $p$, see Sec. 3.2. The prompt length is the only VPT-specific hyper-parameter that we tune. For Transformer backbones, the range of $p$ is $\{1, 5, 10, 50, 100, 200\}$ and $\{1, 5, 10, 50\}$ for ViT and Swin, respectively. The maximum choice of $p$ is approximately close to the number of image patch tokens within each MSA for both architectures (ViT: 196, Swin: 49). We also apply a dropout of 0.1 for VPT-DEEP. For ConvNets, the range of $p$ is $\{1, 3, 5, 7, 9, 11\}$. Each prompt is randomly initialized with xavier uniform initialization scheme [11]. We follow the original backbone' design choices, such as the existence of the classification tokens `[CLS]`, or whether or not to use the final `[CLS]` embeddings for the classification head input.

**Adapter.** Adapters [16] insert extra lightweight modules inside each Transformer layer. One adapter module generally consists of a linear down-projection (with a reduction rate $r$), followed by a nonlinear activation function, and a linear up-projection, together with a residual connection. [36,37] exhaustively searched all possible configurations and found that only inserting adapters after the FFN "Add & LayerNorm" sub-layer works the best. Therefore we also use this setup in our own implementation. We sweep the reduction rate $r$ in $\{8, 64, 256\}$.

**Augmentation and other hyper-parameters.** We adopt standard image augmentation strategy during training: normalize with ImageNet means and standard deviation, randomly resize crop to 224×224 and random horizontal flip for five FGVC datasets, and resize to 224×224 for the VTAB-1k suite.[1]

**Table 1.** Implementation details for each fine-tuning method evaluated. ⋆: we observe that for VPT-SHALLOW sometimes benefit from a larger base LR for 6 out of 24 tasks evaluated, where we search from $\{1000.0, 500.0, 250.0, 100.0\}$

| | FULL,PARTIAL,BIAS,ADAPTER | LINEAR,SIDETUNE, MLP, VPT |
|---|---|---|
| Optimizer | AdamW [29] | SGD |
| Optimizer momentum | - | 0.9 |
| *base_lr* range | {0.001, 0.0001, 0.0005, 0.005} | {50., 25., 10., 5., 2.5, 1.,0.5, 0.25, 0.1, 0.05}⋆ |
| Weight decay range | $\{0.01, 0.001, 0.0001, 0.0\}$ | |
| Learning rate schedule | cosine decay | |
| Warm up epochs | 10 | |
| Total epochs | 100 (ViT-B, Swin-B), 50 (ViT-L/H) | |

[1] Following the default settings in VTAB, we don't adopt other augmentations

**Table 2.** Specifications of the various datasets evaluated. ⋆: we randomly sampled the `train` and `val` sets since there are no public splits available

| Dataset | Description | # Classes | Train | Val | Test |
|---|---|---|---|---|---|
| Fine-grained visual recognition tasks (FGVC) | | | | | |
| CUB-200-2011 [40] | Fine-grained bird species recognition | 200 | 5,394⋆ | 600⋆ | 5,794 |
| NABirds [38] | Fine-grained bird species recognition | 55 | 21,536⋆ | 2,393⋆ | 24,633 |
| Oxford Flowers [33] | Fine-grained flower species recognition | 102 | 1,020 | 1,020 | 6,149 |
| Stanford Dogs [19] | Fine-grained dog species recognition | 120 | 10,800⋆ | 1,200⋆ | 8,580 |
| Stanford Cars [9] | Fine-grained car classification | 196 | 7,329⋆ | 815⋆ | 8,041 |
| Visual Task Adaptation Benchmark (VTAB-1k) [44] | | | | | |
| CIFAR-100 [21] | Natural | 100 | 800/1000 | 200 | 10,000 |
| Caltech101 [24] | | 102 | | | 6,084 |
| DTD [4] | | 47 | | | 1,880 |
| Flowers102 [33] | | 102 | | | 6,149 |
| Pets [34] | | 37 | | | 3,669 |
| SVHN [32] | | 10 | | | 26,032 |
| Sun397 [43] | | 397 | | | 21,750 |
| Patch Camelyon [39] | Specialized | 2 | 800/1000 | 200 | 32,768 |
| EuroSAT [15] | | 10 | | | 5,400 |
| Resisc45 [3] | | 45 | | | 6,300 |
| Retinopathy [18] | | 5 | | | 42,670 |
| Clevr/count [17] | Structured | 8 | 800/1000 | 200 | 15,000 |
| Clevr/distance [17] | | 6 | | | 15,000 |
| DMLab [1] | | 6 | | | 22,735 |
| KITTI/distance [10] | | 4 | | | 711 |
| dSprites/location [31] | | 16 | | | 73,728 |
| dSprites/orientation [31] | | 16 | | | 73,728 |
| SmallNORB/azimuth [22] | | 18 | | | 12,150 |
| SmallNORB/elevation [22] | | 9 | | | 12,150 |

**Table 3.** Specifications of different pre-trained backbones used in the paper. Parameters (M) are of the feature extractor. "Batch size" column reports the batch size for LINEAR / PARTIAL / {FULL, BIAS, ADAPTER} / VPT ($p < 100$) / VPT ($p \geq 100$). All backbones are pre-trained on ImageNet [6] with resolution 224×224

| Backbone | Pre-trained Objective | Pre-trained Dataset | # params (M) | Feature dim $d$ | Batch Size | Pre-trained Model |
|---|---|---|---|---|---|---|
| ViT-B/16 [7] | Supervised | ImageNet-21k | 85 | 768 | 2048 / 1280 / 128 / 128 / 64 | checkpoint |
| ViT-L/16 [7] | | | 307 | 1024 | 2048 / 640 / 64 / 64 / 32 | checkpoint |
| ViT-H/14 [7] | | | 630 | 1280 | 1024 / 240 / 28 / 28 / 14 | checkpoint |
| ViT-B/16 [7] | MoCo v3 [2] | ImageNet-1k | 85 | 768 | 2048 / 1280 / 128 / 128 / 64 | checkpoint |
| ViT-B/16 [7] | MAE [13] | | | | | checkpoint |
| Swin-B [27] | Supervised | ImageNet-21k | 88 | 1024 | 1024 / 1024 / 128 / 80 / - | checkpoint |
| ConvNeXt-Base [28] | Supervised | ImageNet-21k | 88 | 1024 | 1024 / 1024 / 128 / 128 / - | checkpoint |
| ResNet-50 [14] | Supervised | ImageNet-1k | 23 | 2048 | 2048 / 2048 / 384 / 256 / - | checkpoint |

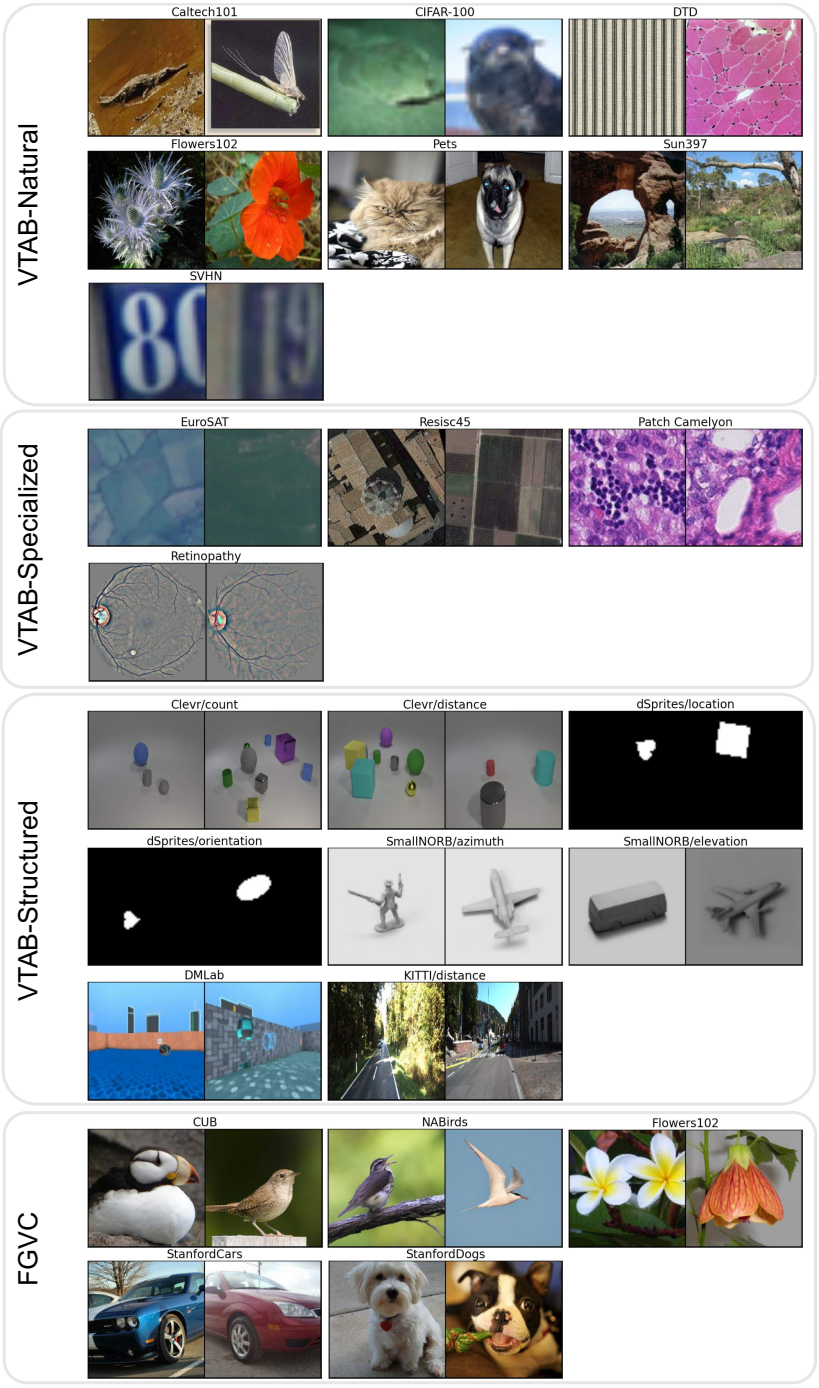


**Fig. 1.** Dataset examples for all classification tasks evaluated

Tab. 1 summarizes the optimization configurations we used. Following [30], we conduct grid search to find the tuning-specific hyper-parameters, learning rate, and weight decay values using `val` set of each task. Following the linear scaling rule [20,12,2,13], the learning rate is set as $base\_lr \times b/256$, where $b$ is the batch size used for the particular model, and $base\_lr$ is chosen from the range specified in Tab. 1. The optimal hyper-parameter values for each experiment can be found in Appendix D.

**Datasets and pre-trained backbones specifications.** Tabs. 2 and 3 summarize the statistics and details of the evaluated classification datasets and all the pre-trained backbones used in the paper. Fig. 1 includes image examples of all 24 classification tasks evaluated.

### A.2 Semantic Segmentation Experiments

ADE20K [46] is a challenging scene parsing benchmark with 150 fine-grained labels. The training and validation sets contain 20,210 and 2,000 images respectively. We utilize the public codebase MMSegmentation [5] in our implementation.[2] The ViT-L backbone is supervisely pre-trained on ImageNet-21k.[3]

SETR [45] is a competitive segmentation framework using ViT as the encoder. PUP is a progressive upsampling strategy consisting of consecutive convolution layers and bilinear upsampling operations. Among multiple decoder choices, PUP works the best according to MMSegmentation's reproduction therefore we also use it as in our implementation.[4]

When applying VPT to SETR-PUP, we only insert prompts into SETR's ViT encoder backbone. For the decoder, only image patch embeddings are used as inputs and prompt embeddings are discarded. Same as recognition tasks, only the PUP decoder head and prompts are learned during training and the ViT backbone is frozen.

For full fine-tuning, we use the same hyper-parameters as in MMSegmentation. For HeadOnly, Bias, and VPT, we use the hyper-parameter sweep on learning rate {0.05, 0.005, 0.0005, 0.001}. The optimal learning rate is 0.005 for all methods. We sweep prompt length $p \in \{1, 5, 10, 50, 100, 200\}$. For VPT, we also change the learning rate multiplier to 1.0 instead of the default 10.0, so the decoder head and prompts share the same learning rate. Other hyper-parameters remain the same as full fine-tuning.

## B Extended Analysis

**Effect of expanding input sequence length.** As shown in Tab. 1, by expanding the input sequence with learnable prompts, VPT achieves better performance than Full on the 20 out of 24 tasks evaluated. To investigate whether the

[2] See the MMSegmentation GitHub page
[3] ViT-L/16 checkpoint
[4] MMSegmentation's reproduction on SETR

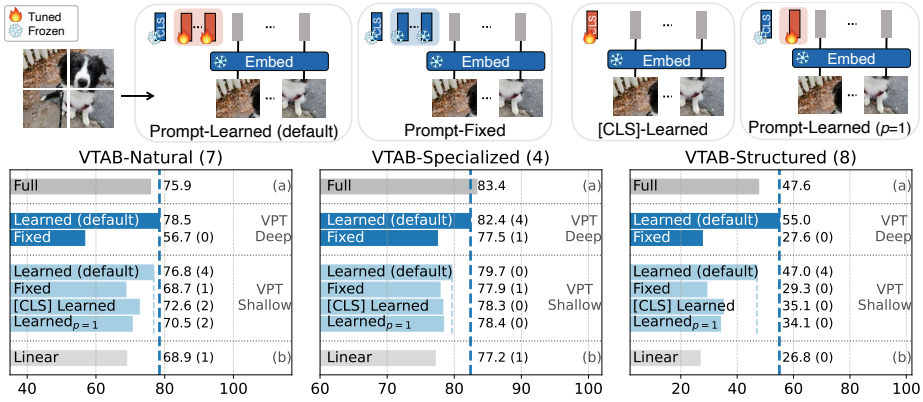


**Fig. 2.** Effect of expanding input sequence. Illustration of different strategies is included at top, and results of those are presented at the bottom section. For easy comparison, two dark and light blue lines represent the performance of default VPT-DEEP and VPT-SHALLOW, respectively

advantage of VPT is due to its enlarged input sequence length, we experiment on two more variants: (1) the prompts are kept frozen during fine-tuning stage (`Prompt-Fixed`). (2) only tuning the `[CLS]` token (`[CLS]-Learned`). From Fig. 2 we can see that, updating prompt embeddings (`Prompt-Learned`) offers significant gains, while `Prompt-Fixed` yields comparable results w.r.t. LINEAR. This suggests that the final performance of VPT is mainly contributed by the learned prompt embeddings instead of the enlarged sequence length. Updating the `[CLS]` token performs similarly as updating 1 prompt (`[CLS]` *vs*. $\texttt{Learned}_{p=1}$), but still lags behind the default setting where we manually select the best number of prompt tokens based on the `val` set.

**Sharing prompts.** We examine the effect of sharing parameters of prompts in Fig. 3 by setting the same prompt embedding *within* Transformer layers (`Shared-intra`), *among* all layers (`Shared-inter`), and for all prompts inserted in the Transformer (`Shared-all`). We can observe that: (1) Sharing prompts within layer (`Shared-intra`) performs competitively or slightly outperforms the performance of using one prompt ($\texttt{Default}_{p=1}$), further demonstrating the value of expanding input sequence. (2) Although `Shared-intra` under-performs `Default` in general, surprisingly, `Shared-inter` slightly outperforms our default VPT-DEEP while using similar number of trainable parameters (total number of parameters for all VTAB tasks: 1.14× *vs*. 1.13× for `Shared-inter` *vs*. `Default`, respectively). Closer examination reveals that the optimal prompt length $p$ for `Shared-inter` is in general larger than `Default`, *i.e*., average prompt length on all VTAB tasks: 64.58 *vs*. 60.94, for `Shared-inter` *vs*. `Default`, respectively. (3) Sharing the same prompt embedding both among and within layers

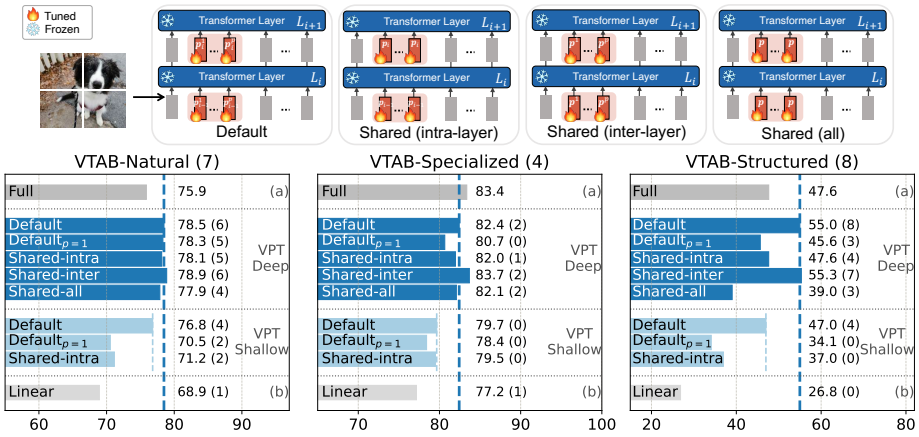


**Fig. 3.** Effect of sharing prompts. Illustration of different strategies is included at top, and results of those are presented at the bottom section. For easy comparison, the blue dashed line represents the performance of default VPT-DEEP

(`Shared-all`) deteriorates performance, but still surpass the linear probing results across three VTAB subgroups.

**Prompt initialization.** In NLP, prompt tuning could benefit from more sophisticated prompt initialization, as shown in [23]. We investigate if this is the case for visual prompting as well. We utilize prototype representations for down-stream target classes so that the prompts are initialized with embeddings that enumerate the output space. Since we want the model to produce an output embedding that is close to one of these prototype representations given a test example, initializing prompts in this manner might give the model some hints about the target categories thus help improve the optimization process.

Concretely, we use the averaged final `[CLS]` embeddings whithin each target class of the down-stream dataset `train` split. Given the pre-trained ViT with $N$ layers, and the down-stream `train` set with $c$ target classes, for each training example, we compute the final `[CLS]` embeddings, $\mathbf{x}_N \in \mathbb{R}^d$. Then we average these embeddings within each target class to get $\{\hat{\mathbf{x}}_N^k \in \mathbb{R}^d \mid k \in \mathbb{N}, 1 \leq k \leq c\}$.[5] Setting prompt length $p = c$,[6] we initialize $\mathbf{P}$ with $\{\hat{\mathbf{x}}_N^k\}_{k=1}^{k=c}$ for VPT-SHALLOW , and initialize each $\mathbf{P}_i$ with $\{\hat{\mathbf{x}}_N^k\}_{k=1}^{k=c}$, where $i = 0, 1, \ldots, N-1$, for VPT-DEEP.

We compare the fine-tuning performance using the above initialization strategy (`CLS`) against the default random initialization (`Random`) in Fig. 4. We also report results when we fix the prompts during the fine-tuning stage (·`-fixed`).

[5] if $c > 200$, we further apply $k$-means ($k = 200$) to class-averaged embeddings and use the corresponding 200 centroid embeddings as $\{\hat{\mathbf{x}}_N^k \in \mathbb{R}^d\}_{k=1}^{k=200}$.

[6] if $c > 200$, we set $p = 200$ so that prompt length won't be too large. In fact, for VTAB, only the Sun397 task in the *Natural* subgroup has over 200 classes. See Tab. 2.

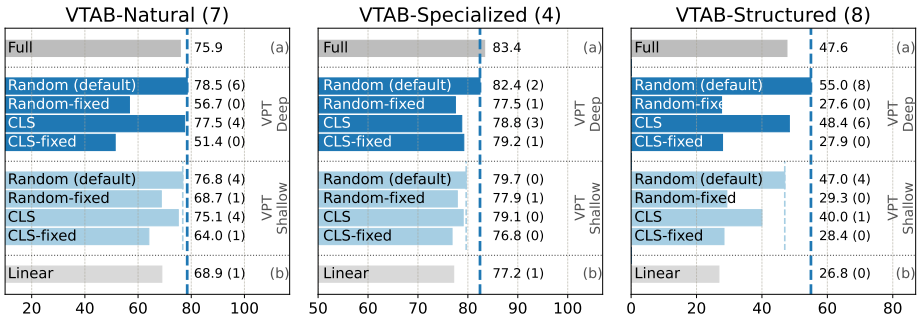


**Fig. 4.** Effect of prompt initialization. For easy comparison, the two blue dashed line represents the performance of default VPT-DEEP and VPT-SHALLOW, respectively

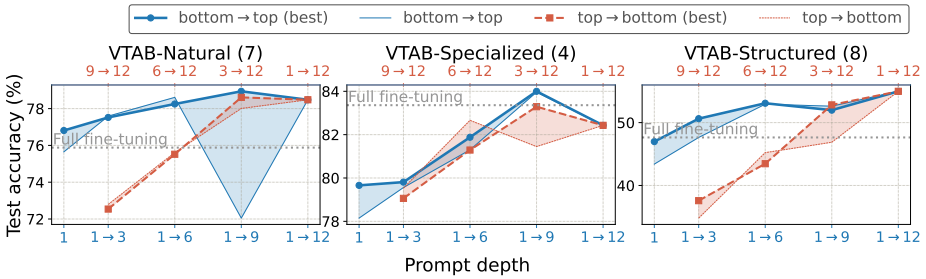


**Fig. 5.** Sensitivity to prompt length for the prompt depth experiments. We select the best prompt length for each variant with `val` sets. We also include the same prompt length for all depth choices. $i \rightarrow j$ indicates the Transformer layer indices that prompts are inserted into. The 1-st layer refers to the one closest to input. ViT-B has a total of 12 layers

As shown in Fig. 4, it's quite surprising that our default random initialization (`Random`) works the best in general, consistently across different subgroups of VTAB without extra pre-processing steps described above (`CLS`). `CLS` works comparably in *Natural* and *Specialized* subgroups.[7]

**Prompt depth *vs*. prompt length.** In Fig.7, we ablate the number of layers we insert prompts in. For each prompt depth variant, Fig.7 reports the results using the best prompt length for *each* task ("$\cdot \rightarrow \cdot$ (best)" in Fig. 5). Here we adopt another setting where the best prompt length from $1 \rightarrow 12$ are used for *all* other prompt depth variants. Comparing both "$\cdot \rightarrow \cdot$ (best)" and "$\cdot \rightarrow \cdot$", we

[7]Utilizing the per-class averaged `[CLS]` features, we also tried several other different implementation variants, including using per-layer `[CLS]` embeddings for VPT-DEEP instead of only the final output `[CLS]` vector. They perform either the same as or even much worse than the `CLS` strategy above, and none of them is able to out-perform the default `Random`.

**Table 4.** Combining VPT with BIAS with a pre-trained ViT-B in Sec.4.2. For each method and each downstream task group, we report the average test accuracy score and `number of wins in (·)` compared to FULL. The difference between the hybrid methods and their VPT counterpart are color coded

| | BIAS | VPT-SHALLOW | VPT-SHALLOW + BIAS | VPT-DEEP | VPT-DEEP + BIAS |
|---|---|---|---|---|---|
| **VTAB-Natural** | 73.30 (3) | 76.81 (4) | 79.78 (5) ↑2.97 | 78.48 (6) | 77.64 (6) ↓0.84 |
| **VTAB-Specialized** | 78.25 (0) | 79.66 (0) | 81.38 (0) ↑1.72 | 82.43 (2) | 82.22 (2) ↓0.21 |
| **VTAB-Structured** | 44.09 (2) | 46.98 (4) | 45.89 (3) ↓1.09 | 54.98 (8) | 53.87 (6) ↓1.11 |

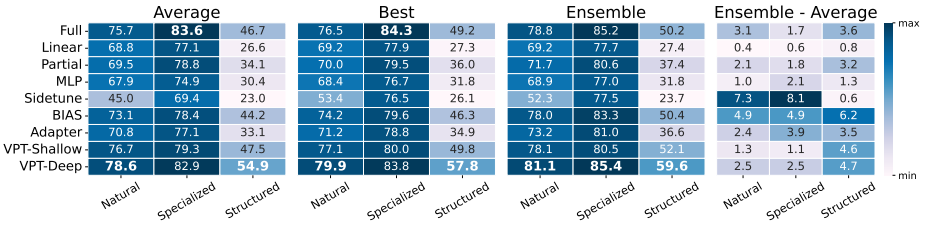


**Fig. 6.** Performance of a five-run ensemble. We report the averaged, the best among five runs as well. Best performance is bolded in each column

observe that there are varied sensitivities to prompt length for different depths, especially if we insert prompts in nine layers only (3 →12, 12 → 3).

**Combine VPT with Bias Tuning.** Our experiments in the main paper reveal that BIAS is a competitive parameter-efficient tuning baseline (*e.g.*, Tab.1`(c)`). Based on this observation, we explore another protocol where we update both prompts and the bias terms of the pre-trained backbone, keeping everything else in the backbone frozen (VPT+BIAS). As shown in Tab. 4, to our surprise, incorporating BIAS with VPT does not yield superior results in general, even undermines VPT-DEEP for all 3 task subgroups. This suggests that these two methods are not necessarily complementary to each other.

**Prompt ensembling.** [23] demonstrated prompt's efficiency in the context of model ensembling. For an ensemble of $k$ models, we only need to store the learnt prompt vectors instead of $k$ copies of the whole fine-tuned model parameters (*e.g.*, $k\times$2.5GB for ViT-H). Furthermore, given one test example during inference time, *only one* forward pass is executed with a specially-designed batch with replicated original data but varied prompts.

Given such advantages, we also investigate VPT's effectiveness on prompt ensembling. We train 5 different prompts for each VTAB task with different random seeds, using the same pre-trained ViT-B backbone and hyper-parameters as in Tab.1. Fig. 6 shows that the ensembled VPT-DEEP outperforms the average

**Table 5.** Non-parametric paired one-tailed $t$-test (the Wilcoxon signed-rank test) on whether VPT-DEEP's performance is greater than other methods on 19 VTAB tasks. Results show that, VPT-DEEP is indeed statistically significantly better than other fine-tuning protocols ($p < 0.05$)

| | (a) FULL | (b) LINEAR | (b) MLP-3 | (b) PARTIAL-1 | (c) SIDETUNE | (c) BIAS | (c) ADAPTER | (ours) VPT-SHALLOW |
|---|---|---|---|---|---|---|---|---|
| Is VPT-DEEP statistically significantly better? | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $p$-value | 1.2e-03 | 2.7e-05 | 1.9e-06 | 1.9e-05 | 1.9e-06 | 1.9e-06 | 3.8e-06 | 2.7e-05 |

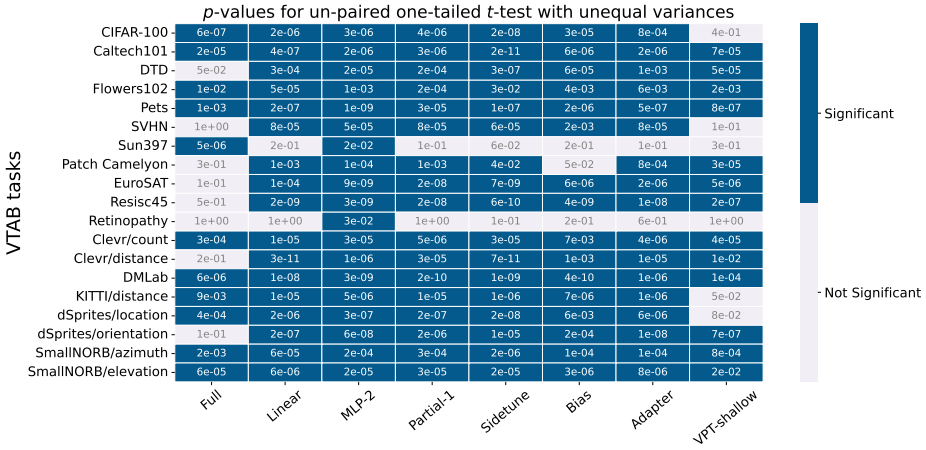


**Fig. 7.** Un-paired one-tailed $t$-test with unequal variances (Welch's $t$-test) on whether VPT-DEEP's performance is greater than other methods for each VTAB task. Results show that, VPT-DEEP is statistically significantly better than other fine-tuning protocols ($p < 0.05$) in most instances

or even the best single-prompt counterparts, as well as other ensembled fine-tuning methods including FULL.

**Test of statistical significance.** We conduct non-parametric paired one-tailed $t$-test (the Wilcoxon signed-rank test [42]) on whether VPT-DEEP's performance is greater than other fine-tuning methods across 19 VTAB tasks (the null hypothesis $H_0$ states that the mean VTAB performance difference between VPT-DEEP and alternate baseline method is zero. The alternative hypothesis $H_1$ states that VPT-DEEP outperforms the baseline method on VTAB). Tab. 5 presents the $p$-values of each test, with the number of observations equal to 19 for each method compared (we use the averaged accuracy scores among 5 runs for 19 VTAB tasks and all fine-tuning methods). For all of the fine-tuning protocols compared, VPT-DEEP's improvements are statistically significant ($p < 0.05$).

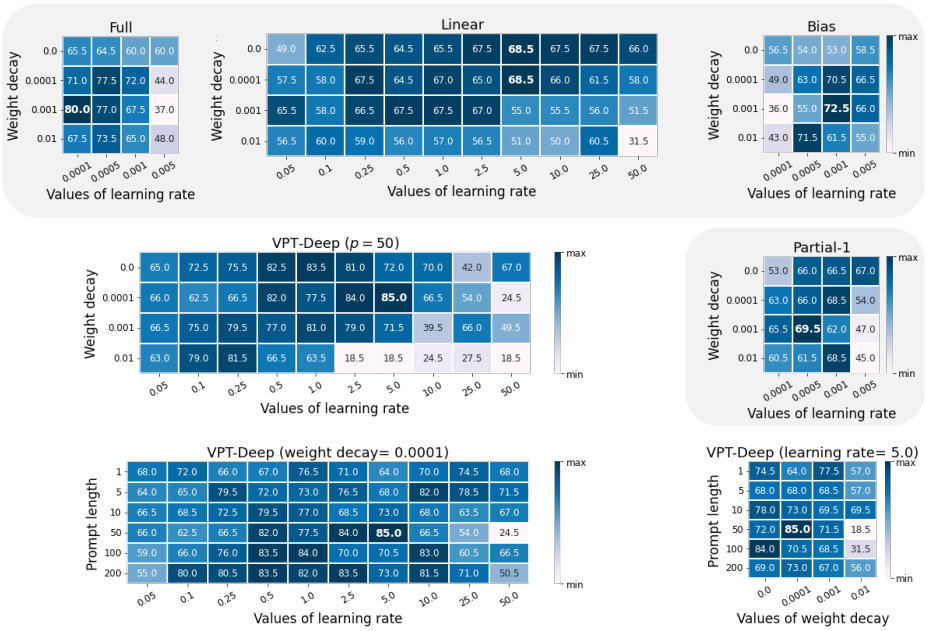


**Fig. 8.** Effect of different fine-tuning hyperparameters. Evaluated on the VTAB-*Specialized*: KITTI/Distance task. Other tuning methods are shaded in gray

We also conduct un-paired one-tailed $t$-test with unequal variances (Welch's $t$-test [41]), comparing the individual runs (the number of observations = 5) for each VTAB task ($H_0$ states that VPT-DEEP and the other baseline perform the same for a specific VTAB task, while $H_1$ states that VPT-DEEP outperforms the other baseline for a specific VTAB task). Fig. 7 presents the $p$-values for each <VPT-DEEP, baseline method> pair on each task. We reject $H_0$ on 127 out of $19 \times 8 = 152$ cases ($p < 0.05$). Compared to FULL, VPT-DEEP achieves statistically significant better performance on 11 out of 19 tasks.

**Effect of different fine-tuning hyper-parameters.** In Fig. 8, we present different tuning protocol's performance on different fine-tuning hyper-parameters including learning rate and weight decay. For our proposed VPT-DEEP, we also ablate different choices of prompt length $p$, which is the only hyper-parameter that needs to be manually tuned. All experiments are evaluated on the `val` set of KITTI/Distance task (VTAB-*Specialized*). We observe different behaviors between LINEAR and VPT. Both methods freeze backbone parameters during fine-tuning stage. Linear probing is more sensitive to weight decay values in general, whereas VPT is influenced by both learning rate and weight decay values. VPT with larger prompt length is also less sensitive to the choice of learning rate.

**Table 6.** ViT-B/16 pre-trained on supervised ImageNet-21k, fine-tuned with resolution 384×384. We also include VPT with image resolution 224×224, $p = 380$, so the effective image resolution is 384×384. For each method and each downstream task group, we report the average test accuracy score and `number of wins in (·)` compared to FULL. "Total params" denotes total parameters needed for all 24 downstream tasks. Best results among all methods except FULL are **bolded**

| | ViT-B/16 (85.8M) | Fine-tune Resolution | Total params | VTAB-1k Natural | VTAB-1k Specialized | VTAB-1k Structured |
|---|---|---|---|---|---|---|
| | Total # of tasks | | | 7 | 4 | 8 |
| (a) | FULL | 384 | 19.07× | 72.57 | 83.05 | 50.86 |
| | FULL | 224 | 19.07× | 75.88 | 83.36 | 47.64 |
| (b) | LINEAR | 384 | 1.01× | 66.30 (2) | 76.77 (0) | 27.86 (0) |
| | MLP-3 | | 1.27× | 66.45 (3) | 77.77 (0) | 38.03 (0) |
| | PARTIAL-1 | | 2.58× | 67.91 (4) | 76.94 (0) | 37.16 (0) |
| (c) | SIDETUNE | 384 | 3.12× | 47.08 (1) | 40.34 (0) | 24.18 (0) |
| | BIAS | | 1.03× | 70.30 (4) | 76.06 (0) | 45.35 (1) |
| | ADAPTER | | 1.11× | 69.42 (6) | 77.11 (0) | 30.62 (0) |
| (ours) | VPT-SHALLOW ($p \in \{1, 5, 10, 50, 100, 200\}$) | 384 | 1.02× | 75.30 (4) | 78.50 (0) | 46.56 (2) |
| | VPT-DEEP ($p \in \{1, 5, 10, 50, 100, 200\}$) | 384 | 1.19× | **79.37 (6)** | **82.86 (2)** | **56.36 (7)** |
| | VPT-SHALLOW ($p = 380$) | 224 | 1.07× | 75.07 (3) | 79.03 (0) | 46.21 (2) |
| | VPT-DEEP ($p = 380$) | 224 | 1.78× | 74.20 (4) | 82.30 (2) | 54.50 (6) |

**Effect of image resolution.** The original ViT paper [7] found that fine-tuning with higher image resolutions (384×384) is beneficial to downstream recognition tasks. All recognition experiments presented in the main paper are fine-tuned on 224×224 resolution. As shown in Tab. 6, we re-run the VTAB experiments with the same setup as in Tab.1 but in the 384 resolution instead of the default 224. We can see that, VPT-DEEP still achieves the best performance among all parameter-efficient tuning protocols, and even outperforms full fine-tuning on 15 out of 19 tasks. Although the increase of image resolutions doesn't lead to better full fine-tuning performance in general, it indeed slightly boosts VPT-DEEP's performance.

Another interesting observation from Tab. 6 is that with 224 fine-tune resolution and a larger value of $p = 380$, VPT could achieve similar or better performance compared to FULL with 384 resolution, while using the same input sequence length yet significantly less trainable parameters.

**Empirical computational cost.** One possible limitation of VPT is the extra input sequence length for Transformers. In theory the complexity of MSA is quadratic w.r.t. the input sequence length, but this might not be the case for real-world speed due to hardware details like lane widths and cache sizes [7]. In Tab. 7 and Fig. 9, we study the empirical computational cost, *i.e.*, latency, and peak GPU memory usage at both training and inference times, for all the fine-tuning protocols studied. All experiments use the same A100 GPU with a batch size 64 for both training and inference. We can see that the theoretical

**Table 7.** Cost analysis using a ViT-B/16 pre-trained on supervised ImageNet-21k. For each method and each downstream task group, we report the latency (**ms/img**) and peak GPU memory usage (**GB**) at both training and inference time. "Tuned params" denotes the fraction of learnable parameters needed. "Scope" denotes the tuning scope of each method. "Extra params" denotes the presence of additional parameters besides the pre-trained backbone and linear head. All experiments use the same A100 GPU

| | ViT-B/16 (85.8M) | Tuned params | Scope | | | Extra params | Train | | Test | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Input | Backbone | Head | | Latency (ms/img) | Memory (GB) | Latency (ms/img) | Memory (GB) |
| (a) | FULL | 100% | | ✓ | ✓ | | 358.7 | 11.7 | 69.7 | 0.87 |
| (b) | LINEAR | 0.09% | | | | | 148.9 | 0.9 | 64.4 | 0.87 |
| | PARTIAL-1 | 8.35% | | | ✓ | | 193.2 | 1.4 | 66.1 | 0.87 |
| | MLP-3 | 1.45% | | | | ✓ | 164.3 | 0.9 | 64.4 | 0.87 |
| (c) | SIDETUNE | 10.09% | | | | ✓ | 164.6 | 1.2 | 66.9 | 0.91 |
| | BIAS | 0.21% | | | | | 296.9 | 10.1 | 65.6 | 0.87 |
| | ADAPTER ($r = 8$) | 2.12% | | ✓ | | ✓ | 293.4 | 9.9 | 68.2 | 0.87 |
| | ADAPTER ($r = 64$) | 0.36% | | | | ✓ | 294.4 | 9.8 | 68.3 | 0.87 |
| | ADAPTER ($r = 256$) | 0.17% | | | | ✓ | 271.4 | 9.8 | 68.0 | 0.87 |
| (ours) | VPT-SHALLOW ($p = 1$) | 0.09% | | | | | 205.9 | 10.3 | 68.1 | 0.88 |
| | VPT-DEEP ($p = 1$) | 0.10% | ✓ | | | ✓ | 213.6 | 10.3 | 69.4 | 0.88 |
| | VPT-SHALLOW ($p = 200$) | 0.27% | | | | | 350.6 | 25.8 | 138.8 | 1.84 |
| | VPT-DEEP ($p = 200$) | 2.19% | | | | | 360.1 | 25.8 | 140.8 | 1.85 |

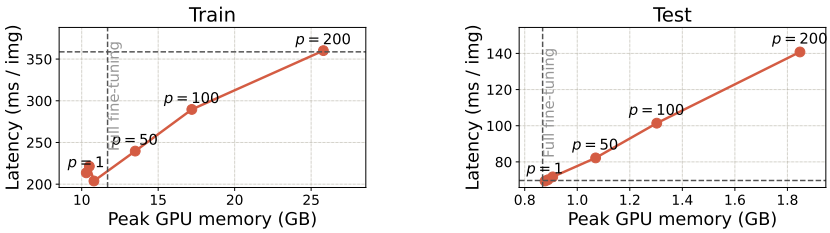


**Fig. 9.** Peak GPU memory and latency (ms/img) during both training (left) and inference time (right). For easy comparison, the gray dashed lines represent latency and memory of full fine-tuning

quadratic scaling w.r.t. sequence length barely happens to VPT. For instance, doubling the length ($p = 200$ *vs*. $m = 198$) basically only lead to 2× (instead of 4×) inference latency and peak GPU memory w.r.t. full fine-tuning. For training, the latency would be largely reduced with less number of prompts.

An equivalent implementation of VPT during test time is directly prepend the parameters to the key and value arrays inside the self-attention module of Transformer [25] (VPT-prefix). While we found that such implementation does not lead to accuracy improvement on VTAB datasets, it reduces the computation cost during inference. Figure 10 shows the comparison with different values of $p$. VPT-prefix reduces test-time latency and peak GPU memory with a large margin especially when $p$ becomes large.

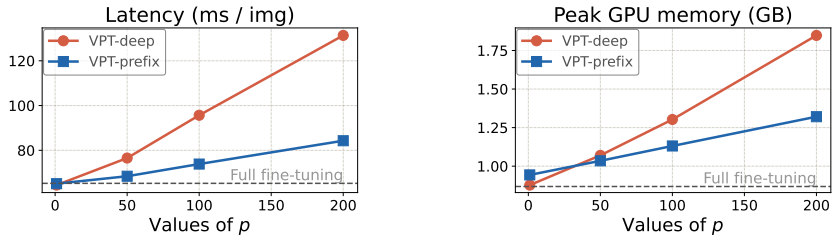


**Fig. 10.** VPT-deep *vs*. VPT-prefix: peak GPU memory (left) and latency (right) during inference time. For easy comparison, the gray dashed lines represent latency and memory of full fine-tuning

## C Further Discussion

**VPT *vs*. Adversarial Reprogramming (AR).** The differences are: (1) the number of learnt parameters injected in the input space in AR literature [8] is nearly 20 times larger than ours (264k *vs*. 13k). VPT is significantly more parameter-efficient; (2) AR has shown its effectiveness in ConvNet, while VPT can be applied to broader architectures, including ViT, Swin. Furthermore, VPT is more general with the option of diving into deeper layers of pre-trained backbone (Fig. 2), whereas AR strictly applies to the *first* input layer of ConvNets. (3) another distinction is that our setting update both prompts and classification head, while AR [8] directly use the pre-trained classification head. Our setup is more general and could be applied to models with a broader range of pre-training objectives (*e.g.*, MAE [13], which does not include a pre-trained classification head) and broader vision tasks (*e.g.*, segmentation).

**Visual prompt *vs*. textual prompt.** Our paper also discover discrepancies between visual and textual prompts: we show that VPT could even outperform full-model fine-tuning on 20 out of 24 cases, which is in contract to the NLP's related work [23]. We also found that random initialized prompts works better (Fig. 4), and prompts at earlier layers matters more (Figs. 7 and 14), which are also different from observation on the NLP side [23,26]. These discrepancies indicate that visual prompting might be fundamentally different from text prompts thus in need of further investigation.

## D Supplementary Results

**Numerical results of Table 1.** Tabs. 8 and 9 present per-task results for 24 classification tasks evaluated in Tab. 1.

**Per-task results on training data ablations.** Fig. 11 presents the per-task results for five FGVC datasets. We observe a similar trend in Fig.3: while all

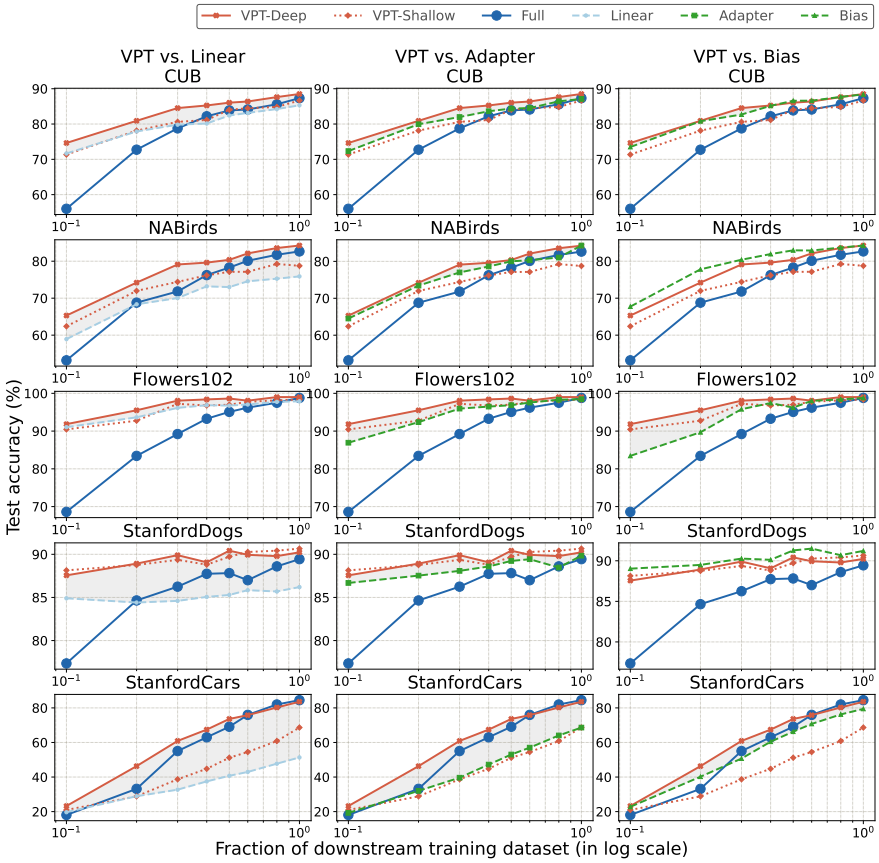


**Fig. 11.** Effect of downstream data size, for each of FGVC tasks. The size of markers are proportional to the percentage of tunable parameters in log scale

parameter-efficient methods outperform full fine-tuning in small-to-medium data regime, VPT-DEEP consistently surpasses FULL across data scales for five FGVC tasks.

**More t-SNE visualizations.** In Fig. 12, We presents more t-SNE visualizations, similar to Fig.9, for all VTAB datasets with less than or equal to 20 target classes.

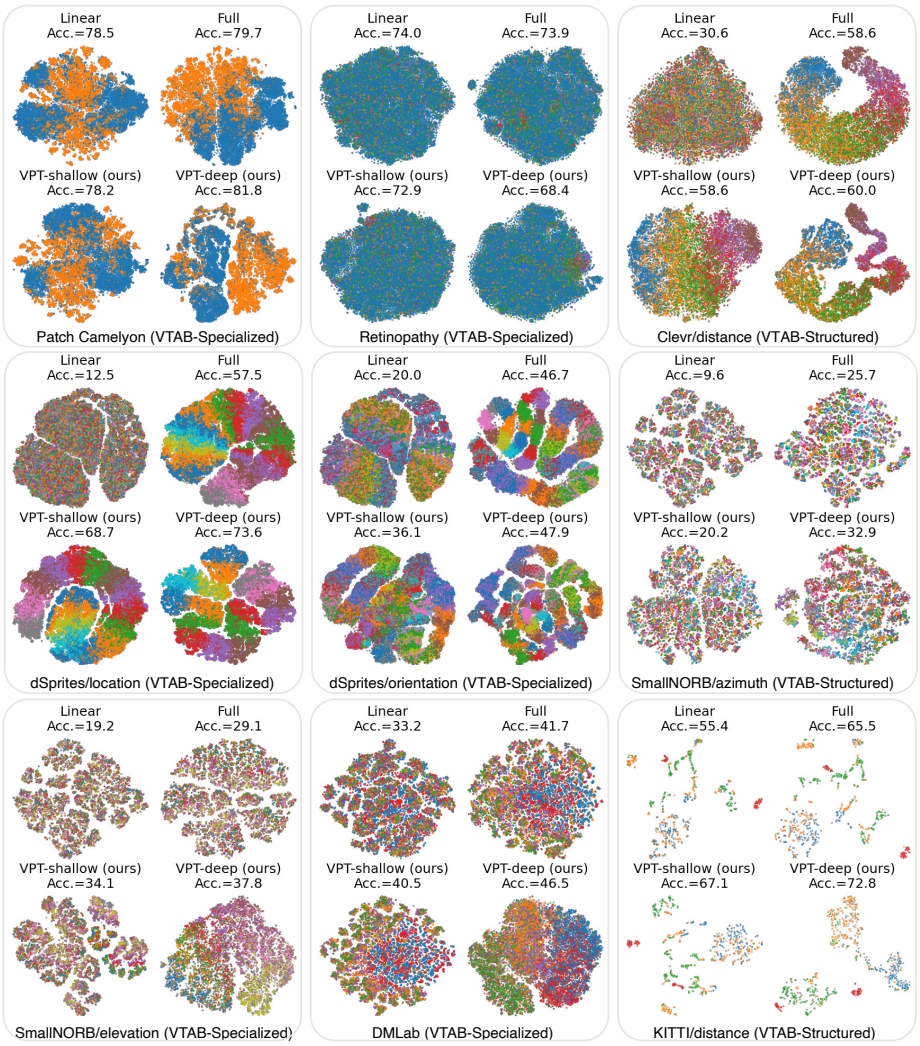


**Fig. 12.** More t-SNE visualization of the final `[CLS]` embedding $\mathbf{x}_N$ of more VTAB tasks. We include tasks that have less or equal to 20 target classes for visualization

**Table 8.** Per-task fine-tuning results from Tab.1 for VTAB-1k with a pre-trained ViT-B/16

| | | CIFAR-100 | Caltech101 | DTD | Flowers102 | Pets | SVHN | Sun397 | Mean | Patch Camelyon | EuroSAT | Resisc45 | Retinopathy | Mean | Clevr/count | Clevr/distance | DMLab | KITTI/distance | dSprites/location | dSprites/orientation | SmallNORB/azimuth | SmallNORB/elevation | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | FULL | 68.9 | 87.7 | 64.3 | 97.2 | 86.9 | 87.4 | 38.8 | 75.88 | 79.7 | 95.7 | 84.2 | 73.9 | 83.36 | 56.3 | 58.6 | 41.7 | 65.5 | 57.5 | 46.7 | 25.7 | 29.1 | 47.64 |
| *Head-oriented* | | | | | | | | | | | | | | | | | | | | | | | |
| (a) | LINEAR | 63.4 | 85.0 | 63.2 | 97.0 | 86.3 | 36.6 | 51.0 | 68.93 (1) | 78.5 | 87.5 | 68.6 | 74.0 | 77.16 (1) | 34.3 | 30.6 | 33.2 | 55.4 | 12.5 | 20.0 | 9.6 | 19.2 | 26.84 (0) |
| | PARTIAL-1 | 66.8 | 85.9 | 62.5 | 97.3 | 85.5 | 37.6 | 50.6 | 69.44 (2) | 78.6 | 89.8 | 72.5 | 73.3 | 78.53 (0) | 41.5 | 34.3 | 33.9 | 61.0 | 31.3 | 32.8 | 16.3 | 22.4 | 34.17 (0) |
| | MLP-2 | 63.2 | 84.8 | 60.5 | 97.6 | 85.9 | 34.1 | 47.8 | 67.70 (2) | 74.3 | 88.8 | 67.1 | 73.2 | 75.86 (0) | 45.2 | 31.6 | 31.8 | 55.7 | 30.9 | 24.6 | 16.6 | 23.3 | 32.47 (0) |
| | MLP-3 | 63.8 | 84.7 | 62.3 | 97.4 | 84.7 | 32.5 | 49.2 | 67.80 (2) | 77.0 | 88.0 | 70.2 | 56.1 | 72.83 (0) | 47.8 | 32.8 | 32.3 | 58.1 | 12.9 | 21.2 | 15.2 | 24.8 | 30.62 (0) |
| | MLP-5 | 59.3 | 84.4 | 59.9 | 96.1 | 84.4 | 30.9 | 46.8 | 65.98 (1) | 73.7 | 87.2 | 64.8 | 71.5 | 74.31 (0) | 50.8 | 32.3 | 31.5 | 56.4 | 7.5 | 20.8 | 14.4 | 20.4 | 29.23 (0) |
| | MLP-9 | 53.1 | 80.5 | 53.9 | 95.1 | 82.6 | 24.4 | 43.7 | 61.90 (1) | 78.5 | 83.0 | 60.2 | 72.3 | 73.49 (0) | 47.5 | 27.9 | 28.9 | 54.0 | 6.2 | 17.7 | 10.8 | 16.2 | 26.15 (0) |
| *Backbone-oriented* | | | | | | | | | | | | | | | | | | | | | | | |
| (b) | SIDETUNE | 60.7 | 60.8 | 53.6 | 95.5 | 66.7 | 34.9 | 35.3 | 58.21 (0) | 58.5 | 87.7 | 65.2 | 61.0 | 68.12 (0) | 27.6 | 22.6 | 31.3 | 51.7 | 8.2 | 14.4 | 9.8 | 21.8 | 23.41 (0) |
| | BIAS | 72.8 | 87.0 | 59.2 | 97.5 | 85.3 | 59.9 | 51.4 | 73.30 (3) | 78.7 | 91.6 | 72.9 | 69.8 | 78.25 (0) | 61.5 | 55.6 | 32.4 | 55.9 | 66.6 | 40.0 | 15.7 | 25.1 | 44.09 (2) |
| | ADAPTER-256 | 74.1 | 86.1 | 63.2 | 97.7 | 87.0 | 34.6 | 50.8 | 70.50 (4) | 76.3 | 88.0 | 73.1 | 70.5 | 76.98 (0) | 45.7 | 37.4 | 31.2 | 53.2 | 30.3 | 25.4 | 13.8 | 22.1 | 32.39 (0) |
| | ADAPTER-64 | 74.2 | 85.8 | 62.7 | 97.6 | 87.2 | 36.3 | 50.9 | 70.65 (4) | 76.3 | 87.5 | 73.7 | 70.9 | 77.10 (0) | 42.9 | 39.9 | 30.4 | 54.5 | 31.9 | 25.6 | 13.5 | 21.4 | 32.51 (0) |
| | ADAPTER-8 | 74.2 | 85.7 | 62.7 | 97.8 | 87.2 | 36.4 | 50.7 | 70.67 (4) | 76.9 | 89.2 | 73.5 | 71.6 | 77.80 (0) | 45.2 | 41.8 | 31.1 | 56.4 | 30.4 | 24.6 | 13.2 | 22.0 | 33.09 (0) |
| *Visual-Prompt Tuning* | | | | | | | | | | | | | | | | | | | | | | | |
| (ours) | VPT-SHALLOW | 77.7 | 86.9 | 62.6 | 97.5 | 87.3 | 74.5 | 51.2 | 76.81 (4) | 78.2 | 92.0 | 75.6 | 72.9 | 79.66 (0) | 50.5 | 58.6 | 40.5 | 67.1 | 68.7 | 36.1 | 20.2 | 34.1 | 46.98 (4) |
| | Prompt length ($p$) | 100 | 5 | 1 | 200 | 50 | 200 | 1 | 79.4 | 5 | 50 | 50 | 10 | 28.7 | 100 | 200 | 100 | 100 | 100 | 100 | 200 | 200 | 137.5 |
| | Tuned / Total (%) | 0.18 | 0.10 | 0.04 | 0.27 | 0.08 | 0.19 | 0.36 | 0.17 | 0.01 | 0.05 | 0.09 | 0.01 | 0.04 | 0.10 | 0.18 | 0.09 | 0.09 | 0.10 | 0.10 | 0.19 | 0.19 | 0.13 |
| | VPT-DEEP | 78.8 | 90.8 | 65.8 | 98.0 | 88.3 | 78.1 | 49.6 | 78.48 (6) | 81.8 | 96.1 | 83.4 | 68.4 | 82.43 (2) | 68.5 | 60.0 | 46.5 | 72.8 | 73.6 | 47.9 | 32.9 | 37.8 | 54.98 (8) |
| | Prompt length ($p$) | 10 | 10 | 10 | 1 | 1 | 50 | 5 | 12.4 | 100 | 100 | 10 | 1 | 52.8 | 50 | 200 | 100 | 50 | 10 | 50 | 200 | 200 | 107.5 |
| | Tuned / Total (%) | 0.20 | 0.20 | 0.15 | 0.10 | 0.04 | 0.54 | 0.41 | 0.23 | 1.06 | 1.07 | 0.15 | 0.02 | 0.57 | 0.54 | 2.11 | 1.07 | 0.54 | 0.12 | 0.55 | 2.12 | 2.11 | 1.14 |

**Table 9.** Per-task fine-tuning results from Tab.1 for five FGVC tasks, with a pre-trained ViT-B/16

| | | CUB-200-2011 | NABirds | Oxford Flowers | Stanford Dogs | Stanford Cars | Mean |
|---|---|---|---|---|---|---|---|
| (a) | FULL | 87.3 | 82.7 | 98.8 | 89.4 | 84.5 | 88.54 |
| *Head-oriented* | | | | | | | |
| (a) | LINEAR | 85.3 | 75.9 | 97.9 | 86.2 | 51.3 | 79.32 (0) |
| | PARTIAL-1 | 85.6 | 77.8 | 98.2 | 85.5 | 66.2 | 82.63 (0) |
| | MLP-2 | 85.7 | 77.2 | 98.2 | 85.4 | 54.9 | 80.28 (0) |
| | MLP-3 | 85.1 | 77.3 | 97.9 | 84.9 | 53.8 | 79.80 (0) |
| | MLP-5 | 84.2 | 76.7 | 97.6 | 84.8 | 50.2 | 78.71 (0) |
| | MLP-9 | 83.2 | 76.0 | 96.2 | 83.7 | 47.6 | 77.31 (0) |
| *Backbone-oriented* | | | | | | | |
| (b) | SIDETUNE | 84.7 | 75.8 | 96.9 | 85.8 | 48.6 | 78.35 (0) |
| | BIAS | 88.4 | 84.2 | 98.8 | 91.2 | 79.4 | 88.41 (3) |
| | ADAPTER-256 | 87.2 | 84.3 | 98.5 | 89.9 | 68.6 | 85.70 (2) |
| | ADAPTER-64 | 87.1 | 84.3 | 98.5 | 89.8 | 68.6 | 85.67 (2) |
| | ADAPTER-8 | 87.3 | 84.3 | 98.4 | 88.8 | 68.4 | 85.46 (1) |
| *Visual-Prompt Tuning* | | | | | | | |
| (ours) | VPT-SHALLOW | 86.7 | 78.8 | 98.4 | 90.7 | 68.7 | 84.62 (1) |
| | Prompt length ($p$) | 100 | 50 | 100 | 100 | 100 | 90 |
| | Tuned / Total (%) | 0.31 | 0.54 | 0.23 | 0.20 | 0.26 | 0.31 |
| | VPT-DEEP | 88.5 | 84.2 | 99.0 | 90.2 | 83.6 | 89.11 (4) |
| | Prompt length ($p$) | 10 | 50 | 5 | 100 | 200 | 73 |
| | Tuned / Total (%) | 0.29 | 1.02 | 0.14 | 1.17 | 2.27 | 0.98 |

## References


1. Beattie, C., Leibo, J.Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al.: Deepmind lab. arXiv preprint arXiv:1612.03801 (2016) 2
2. Chen*, X., Xie*, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021) 2, 4
3. Cheng, G., Han, J., Lu, X.: Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE (2017) 2
4. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., , Vedaldi, A.: Describing textures in the wild. In: CVPR (2014) 2
5. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020) 4
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009) 2
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020) 2, 11
8. Elsayed, G.F., Goodfellow, I., Sohl-Dickstein, J.: Adversarial reprogramming of neural networks. In: ICLR (2019) 13
9. Gebru, T., Krause, J., Wang, Y., Chen, D., Deng, J., Fei-Fei, L.: Fine-grained car detection for visual census estimation. In: AAAI (2017) 2
10. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (2013) 2
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010) 1
12. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017) 4
13. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR. pp. 16000–16009 (2022) 2, 4, 13
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 2
15. Helber, P., Bischke, B., Dengel, A., Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (2019) 2
16. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: ICML. pp. 2790–2799. PMLR (2019) 1
17. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: CVPR (2017) 2
18. Kaggle, EyePacs: Kaggle diabetic retinopathy detection (July 2015) 2
19. Khosla, A., Jayadevaprakash, N., Yao, B., Fei-Fei, L.: Novel dataset for fine-grained image categorization. In: First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition. Colorado Springs, CO (June 2011) 2
20. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997 (2014) 4

21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 2
22. LeCun, Y., Huang, F.J., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: CVPR (2004) 2
23. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 3045–3059. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (Nov 2021) 6, 8, 13
24. Li, F.F., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE TPAMI (2006) 2
25. Li, X.L., Liang, P.: 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). pp. 4582–4597. Association for Computational Linguistics, Online (Aug 2021) 12
26. Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., Tang, J.: P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602 (2021) 13
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021) 2
28. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. CVPR (2022) 2
29. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 1
30. Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., Van Der Maaten, L.: Exploring the limits of weakly supervised pretraining. In: ECCV (2018) 4
31. Matthey, L., Higgins, I., Hassabis, D., Lerchner, A.: dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/ (2017) 2
32. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011 (2011) 2
33. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. pp. 722–729. IEEE (2008) 2
34. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: CVPR (2012) 2
35. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NeurIPS Autodiff Workshop (2017) 1
36. Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I.: Adapterfusion: Non-destructive task composition for transfer learning. arXiv preprint arXiv:2005.00247 (2020) 1
37. Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., Gurevych, I.: Adapterhub: A framework for adapting transformers. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations. pp. 46–54. Association for Computational Linguistics, Online (2020) 1

38. Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., Belongie, S.: Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In: CVPR. pp. 595–604 (2015) 2
39. Veeling, B.S., Linmans, J., Winkens, J., Cohen, T., Welling, M.: Rotation equivariant cnns for digital pathology. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (2018) 2
40. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 2
41. Welch, B.L.: The generalization of 'student's'problem when several different population varlances are involved. Biometrika **34**(1-2), 28–35 (1947) 10
42. Wilcoxon, F.: Individual comparisons by ranking methods. In: Breakthroughs in statistics, pp. 196–202. Springer (1992) 9
43. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: CVPR (2010) 2
44. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867 (2019) 2
45. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR. pp. 6881–6890 (2021) 4
46. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. IJCV **127**(3), 302–321 (2019) 4