ATTENTION OUTPUT PROJECTION LAYERS ARE STRONG TARGETS FOR ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Effective adaptation need not update every component or add parameters. Tuning the right subset of existing parameters can be both more effective and more efficient. But where should a pretrained transformer be adapted? We conduct systematic comparisons of transformer components and identify the attention output projection as the strongest individual adaptation target. We attribute this to its role: attention output projections determine how attention features enter the residual stream, so when pretrained features are rich, recombining them can suffice. Our method, Attention Projection Layer Adaptation (APLA), tunes the output projection in each transformer block, or a fixed random subset of its columns, together with the task head. It adds zero parameters, requires no architectural changes, and adds no inference computation. On 21 classification datasets, we achieve the highest mean accuracy of 91.5% with DINOv2 ViT-B, outperforming 20 baselines (the strongest reaching 90.9%) including full fine-tuning (89.7%). Evaluation on 49 vision tasks in total supports these findings under low-data regimes, distribution shift, segmentation, and detection. With ViT-g, APLA reduces training memory by 53% and training time by 43% relative to full fine-tuning. Our finding extends beyond vision: output projections are the best attention component to adapt on six of eight GLUE language tasks. Lastly, we also find that parameter-adding methods such as LoRA and AdaptFormer benefit from placement at the output projection over their default location.

1 INTRODUCTION

Adapting a pretrained transformer requires choosing where to intervene. Many parameter-efficient fine-tuning (PEFT) methods introduce adapters, prompts, or low-rank factors into frozen foundation models (Chen et al., 2022; Jia et al., 2022; Hu et al., 2022), placed by architectural convention or by sensitivity analysis (He et al., 2023). Other approaches tune existing parameters, ranging from biases (Ben Zaken et al., 2022) and sparse weight subsets (Zhang et al., 2024; Shen et al., 2024) to whole attention modules (Touvron et al., 2022). Regardless of the adaptation method, the choice of which parameters to update is a primary design decision.

Our main finding is that the *attention output projection*, W_O, is a particularly effective target. Tuning it across all blocks can outperform even full fine-tuning and leading PEFT methods, with no added parameters or architectural changes.

We establish this through systematic comparisons, tuning each transformer component in isolation across all blocks together with the task head. W_O achieves the highest mean accuracy among six attention and MLP matrices across 21 classification datasets (Table 1). A broader four-dataset comparison also places it ahead of embeddings, normalization, scaling, and whole attention and MLP blocks (Figure 1). The same holds in language: on RoBERTa-base, W_O leads the four attention projections on six of eight GLUE tasks (Table 2).

Why the W_O layer? W_O combines the contextualized features of all attention heads and maps them into the residual stream. We posit that, when the pretrained features already contain what a task needs, adaptation can succeed by changing how those features contribute downstream rather than changing the features themselves. Our analysis characterizes how updating each component alters a block's computation and shows that, at a fixed block input, adapting W_O changes only how attention features are combined, leaving the attention probabilities and value features unchanged. We test this

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

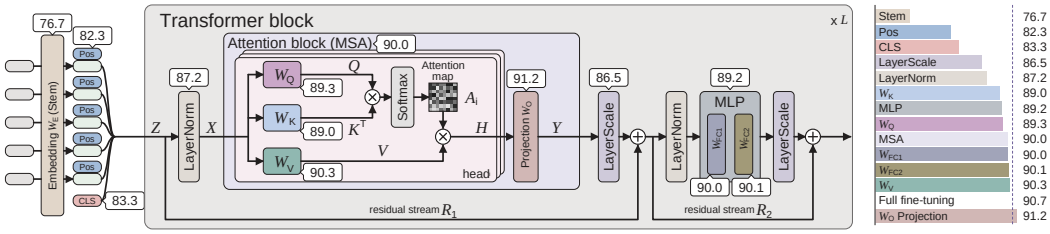

Figure 1: **Which transformer component to adapt?** Mean accuracy over four classification datasets (NABirds, Stanford Cars, AID, and ISIC2019) when tuning each component with the task head. The attention output projection W_O combines the contextualized features of all heads into $Y = HW_O$, which enters the residual stream. It achieves the highest mean accuracy (91.2%), exceeding full fine-tuning (90.7%, dashed line). Tables 1(a) and 2 extend the component comparison to 21 vision datasets and eight language tasks.

empirically, in vision and language tasks, including experiments varying model capacity, distribution shift, and low-data regimes. Interestingly, we also find that relocating other PEFT methods such as LoRA (Hu et al., 2022) and AdaptFormer (Chen et al., 2022) from their default location to W_O improves their average performance (Appendix Table 12).

How much of the projection must change? We find that updating a fixed subset of W_O's columns can retain and even improve the performance compared to adapting the whole projection layer (Table 1(b)). **Which columns should be selected?** We compare selection by gradient-, activation-, and magnitude-based strategies along with random selection. Random selection matches or exceeds these strategies without computing importance scores (Table 1(b)), so effective adaptation need not spend compute to identify important columns to update in advance.

We call the resulting method *Attention Projection Layer Adaptation* (APLA), defined in Section 3.2. We evaluate it on 49 vision and 9 language tasks with 10 vision and 3 language models (Section 4). APLA has the highest mean accuracy against 20 baselines on 21 classification datasets and the highest VTAB-1k averages on ViT-B and Swin-B (Tables 1 and 3); it performs competitively with full fine-tuning and established PEFT methods on GLUE and MMLU (Tables 2 and 27); and it trains with 41% less memory than full fine-tuning on ViT-B and 53% less on ViT-g (Table 5; Figure 2).

2 Related Work

Parameter-selective tuning updates a subset of existing model parameters. Fisher information is used to select parameters in FISH (Sung et al., 2021), Diff pruning learns sparse parameter changes (Guo et al., 2021), and GPS selects weights using gradients (Zhang et al., 2024). SPT combines sensitivity-based selection with unstructured and structured updates (He et al., 2023), while SNELL makes sparse tuning memory-efficient through a low-rank parameterization (Shen et al., 2024). Other methods target components such as biases (Ben Zaken et al., 2022) or selected rows and columns (Kowsher et al., 2025). At the architecture level, Partial-k tunes the final blocks (Jia et al., 2022), surgical fine-tuning selects layer subsets based on distribution shifts (Lee et al., 2023), and ALaST budgets adaptation costs across layers and tokens (Devoto et al., 2024). Attention-only tuning updates the MSA layers (Touvron et al., 2022; Jin et al., 2025), while ProLIP adapts only the final projector from CLIP (Fahes et al., 2026). Closest to our work, RoCoFT updates selected rows or columns of every weight matrix in an LLM (Kowsher et al., 2025), and Hu et al. (2022) compared LoRA target matrices in GPT-3, finding W_O alone close to W_Q, W_V. Neither compares adaptation locations across all block components, tunes the output projection directly, or asks whether restricting its columns helps; we do all three, across domains and tasks.

Low-rank and factorized adaptation methods learn weight changes using fewer trainable parameters. LoRA expresses each change as a product of two smaller matrices (Hu et al., 2022); FacT and RLRR also use factorized updates for ViTs (Jie & Deng, 2023; Dong et al., 2024b). Householder-based adaptation learns reflections (Dong et al., 2024a), while LoRA-Pro improves how low-rank updates are optimized (Wang et al., 2025). LoRA-FA trains only one of LoRA's two matrices (Zhang et al., 2023); VeRA trains scaling vectors while keeping shared random matrices fixed (Kopiczko et al., 2024). APLA instead trains selected columns of the existing W_O matrix: changing r columns gives an update of rank at most r, without adding trainable matrices.

Adapters and prompts insert lightweight modules or tokens into a frozen backbone (Houlsby et al., 2019): adapters in parallel with the MLP (Chen et al., 2022) or shared across blocks (Dong et al., 2023), feature scaling and shifting (Lian et al., 2022), adapters for dense prediction (Yin et al., 2023; 2025), and prompt tokens (Jia et al., 2022; Han et al., 2023); NOAH searches over their combinations (Zhang et al., 2022). Appendix Table 7 lists all methods we compare against.

3 METHODS

3.1 WHAT MAKES THE ATTENTION OUTPUT PROJECTION SPECIAL?

Following Figure 1, let $Z \in \mathbb{R}^{n\times d}$ denote the token representations entering a transformer block (Vaswani et al., 2017), and let $X = \mathrm{LN}_1(Z)$ be the normalized attention input. For each attention head i,

$$A_i(X) = \mathrm{softmax}\left(\frac{(XW_{Q_i})(XW_{K_i})^\top}{\sqrt{d_h}}\right), \qquad H_i(X) = A_i(X)XW_{V_i}, \tag{1}$$

where d_h is the head dimension. $A_i(X)$ specifies how strongly each token attends to the others, and $H_i(X)$ contains the resulting contextualized features. The output projection combines the head outputs $H(X) = [H_1(X), \ldots, H_i(X), \ldots, H_h(X)]$:

$$Y = H(X)W_O = \sum_{i=1}^{h} H_i(X)W_{O_i}, \tag{2}$$

where W_{O_i} contains the rows of W_O associated with head i. The output of the attention module Y contributes one d-dimensional vector per token to the residual stream. In a sequential pre-normalized block,

$$R_1 = Z + \mathrm{LS}_1(Y), \qquad R_2 = R_1 + \mathrm{LS}_2(\mathrm{MLP}(\mathrm{LN}_2(R_1))), \tag{3}$$

where R_1 and R_2 are the residual stream after the attention and MLP sub-blocks, and LS denotes LayerScale, or the identity when LayerScale is absent.

Our hypothesis *is that, especially when data is limited in the target domain, adaptation benefits from reusing useful pretrained features while changing how they contribute to the residual stream.* W_O directly recombines the contextualized features of all heads while retaining the parameters that generate them. If these features already support the task, this restriction can even improve generalization compared with full fine-tuning.

W_O adapts how pretrained features contribute. Assume the hypothesis holds. To understand W_O's role in combining features from attention heads into the residual stream, consider the scenario where the input X and all other parameters W_Q, W_K, and W_V are fixed. Adapting W_O has a direct linear effect on the residual stream as the attention maps $A_i(X)$, LayerScale, and contextualized features $H_i(X)$ are fixed[1], giving

$$\Delta Y = H(X)\Delta W_O, \qquad \Delta R_1 = \mathrm{LS}_1(\Delta Y). \tag{4}$$

Analysis of different adaptation locations. When data for the new task is limited, a useful adaptation location must provide flexibility while preserving useful pretrained structure. We compare how each component changes the residual stream: whether its effect is linear in the parameter update, whether it changes the pre-trained features, and whether its output is constrained by passing through a frozen projection. Eq. (5) isolates each component with the remaining parameters and X fixed.

sub-block	matrix	change to residual stream*	linear in ΔW?	changes pretrained features?	constrained by frozen	
attn.	W_O	$\Delta R_1 = \sum_i H_i(X)\,\Delta W_{O_i}$	yes	no	–	(5a)
	W_V	$\Delta R_1 = \sum_i A_i(X)X\Delta W_{V_i}W_{O_i}$	yes	yes	W_{O_i}	(5b)
	W_Q, W_K	$\Delta R_1 = \sum_i \Delta A_i X W_{V_i}W_{O_i}$	no, softmax	can, via A_i	$W_{V_i}W_{O_i}$	(5c)
MLP	W_{FC2}	$\Delta R_2 = \phi(X_2W_{\mathrm{FC1}})\,\Delta W_{\mathrm{FC2}}$	yes	no	–	(5d)
	W_{FC1}	$\Delta R_2 = \Delta\phi\, W_{\mathrm{FC2}}$	no, ϕ	yes	W_{FC2}	(5e)

[1] LS_1 is a fixed diagonal scaling, thus $\Delta R_1 = H(X)\,\Delta W_O\,\mathrm{diag}(\gamma_1)$; LS is omitted below for clarity.

*These expressions omit LayerScale and projection biases. With LayerScale present, apply the fixed map LS_1 to each attention expression and LS_2 to each MLP expression. Here X is fixed and all other matrices are frozen; $X_2 = \mathrm{LN}_2(R_1)$ is the MLP input, $\Delta A_i = A_i'(X) - A_i(X)$ the change in the attention map after a query or key update, and $\Delta\phi = \phi(X_2(W_{\mathrm{FC1}} + \Delta W_{\mathrm{FC1}})) - \phi(X_2 W_{\mathrm{FC1}})$.

Eq. (5) sorts the locations into three groups. W_O and W_{FC2} are pure feature recombination: linear in the update, no new features, no frozen matrix between the update and the residual stream. Of the two, W_O is closer to what the hypothesis singles out, since it recombines the contextualized features $H(X)$ of all attention heads directly and before the MLP, whereas W_{FC2} recombines only what the token-wise MLP has made of them; their ordering is empirical (Section 4.2). W_V, W_{FC1}, W_Q and W_K all learn new features, which should not be needed under our hypothesis, and all must pass through a frozen matrix. Among them W_V is the least restricted: its effect is linear in the update and passes through a single frozen matrix. W_{FC1} acts through ϕ and W_Q, W_K only re-weight XW_{V_i} through the softmax. We further illustrate these restrictions with a toy example in Appendix A.6.

If the hypothesis only partly holds, and some features must be relearned, W_V offers the best compromise, which may explain why it ranks second after W_O in our empirical results. Note that our analysis concerns what each location can express, not how easily it is learned. Under the feature-reuse hypothesis, it leads us to expect W_O to adapt best, followed by W_V and W_{FC2}, with W_Q, W_K the worst. Results in Section 4.2 support this.

3.2 Low-rank updates through selected columns

With limited data for the new task, changing too many parameters risks overfitting. LoRA (Hu et al., 2022) addresses this by constraining the weight change to rank r, following the observation that adaptation updates have low intrinsic rank (Aghajanyan et al., 2021). We obtain the same constraint without added matrices by updating only r columns of W_O, each of which combines the features of all heads into one entry of Y: the update then has at most r nonzero columns and rank at most r. For a fixed block input, this changes only the r selected entries of Y and leaves the others at their pretrained values, a coordinate restriction that LoRA does not impose.

Let $W_O^{\mathrm{pre}} \in \mathbb{R}^{d\times d}$ denote the pretrained projection, and let J be a fixed set of r selected column indices. Then

$$W_O = W_O^{\mathrm{pre}} + \Delta W_O, \qquad \Delta W_O[:, j] = 0 \quad \text{for } j \notin J, \qquad \mathrm{rank}(\Delta W_O) \le r. \tag{6}$$

Here $\Delta W_O[:, j]$ is column j of the update. Keeping J fixed preserves the rank bound throughout training; it bounds the update, not W_O itself. Selecting all d columns recovers full-projection tuning.

Our approach. We call our method *Attention Projection Layer Adaptation* (APLA). It tunes the attention output projection W_O in each transformer block, or a fixed random subset of r of its columns (dr trainable weights per block), together with the task head, while freezing the remaining backbone parameters. The columns are drawn at random once before training and held fixed. APLA adds zero new parameters, requires no architectural changes, and introduces no additional inference computation. In Section 4.2 we validate APLA empirically.

4 Experiments and Results

4.1 Experimental Setup

We benchmark APLA and 34 other adaptation methods on 49 vision tasks (21 classification datasets, the 19 low-data tasks of VTAB-1k, ImageNet distribution shift, ADE20K segmentation, MS COCO detection, and the MONA protocol) and 9 language tasks (eight GLUE tasks and 5-shot MMLU after Alpaca fine-tuning), using 10 vision backbones (DINOV2 ViT-S/B/L/g by default, plus ImageNet-21K, ImageNet-1K, and OpenCLIP ViT and Swin models) and 3 language models (RoBERTa-base; Llama 3.2 1B and 3.1 8B). Appendix Tables 6 and 7 list every dataset, backbone, and baseline with citations, and state which baselines are our runs under a common protocol and which are published results under their original protocols (VTAB, MONA, and the GLUE literature rows). Each dataset uses its standard metric (Appendix Table 6), accuracy for 17 of the 21 classification datasets; we call the mean over datasets mean accuracy for brevity. Hyperparameters, including the column count r, are selected on validation data and results are reported on the test split; the default protocol is AdamW for 100 epochs with cosine decay (all protocols: Appendix B).

Table 1: **Image classification, 21 datasets, DINOv2 ViT-B. (a)** Adapting a single component (plus the task head): W_O is best or tied on 16/21 datasets and is the only component whose mean exceeds full fine-tuning; dataset by dataset the two tie (10/10/1). **(b)** Column-selection strategies; random selection is best or tied on 16/21. **(c)** Against 20 adaptation methods, APLA has the best mean and is best or second best on 18/21. Details: Tables 8–11.

	Fine-grained						General								Scene	Satellite	Medical					
	CUB-200	NABirds	Birdsnap	Stanford Dogs	Stanford Cars	Aircraft	Caltech-256	Caltech-101	CIFAR-100	CIFAR-10	Oxford Pets	DTD	MIT Indoor	SUN397	AID	RSSCN7	ISIC2019	APTOS2019	DDSM	Colorectal	Pneumonia	**Mean**
(a) Which component to adapt? (each full component tuned in isolation, plus the task head, $r = 768$ for projection)																						
[CLS] token	–	83.9	–	–	89.9	–	–	–	–	–	–	–	–	–	92.9	–	66.4	–	–	–	–	–
Position embeddings	–	85.1	–	–	89.6	–	–	–	–	–	–	–	–	–	92.9	–	61.4	–	–	–	–	–
Patch embedding W_E	–	85.7	–	–	88.2	–	–	–	–	–	–	–	–	–	87.9	–	45.1	–	–	–	–	–
LayerNorm	–	83.7	–	–	91.5	–	–	–	–	–	–	–	–	–	94.6	–	79.0	–	–	–	–	–
LayerScale	–	85.5	–	–	91.2	–	–	–	–	–	–	–	–	–	93.9	–	75.4	–	–	–	–	–
W_Q	86.2	85.5	78.9	84.3	91.8	85.4	93.0	95.9	91.6	98.3	93.3	79.4	87.0	74.2	94.6	80.1	85.1	89.5	94.1	96.5	97.2	88.7
W_K	86.3	85.8	78.5	84.5	91.8	85.9	92.8	96.2	91.7	98.1	93.1	79.9	86.3	74.5	94.5	79.8	83.8	90.1	94.5	96.7	97.4	88.7
W_V	86.8	85.8	79.2	84.8	93.2	86.6	93.7	96.5	92.3	98.7	93.9	**80.8**	87.9	75.1	**96.0**	81.3	86.9	90.8	96.1	**97.8**	98.2	89.6
$\boldsymbol{W_O}$	**88.7**	**86.5**	**79.8**	**85.1**	**94.0**	**86.7**	94.0	**97.3**	92.8	99.2	**94.4**	**80.8**	**88.3**	**75.4**	**96.0**	**82.2**	**88.2**	90.6	**96.2**	97.6	**98.4**	**90.1**
MSA block	–	85.0	–	–	93.8	–	–	–	–	–	–	–	–	–	94.7	–	86.5	–	–	–	–	–
W_{FC1}	86.0	84.7	78.6	84.0	93.5	85.1	92.6	96.1	91.9	98.4	93.6	79.2	86.5	74.3	95.0	80.7	86.9	89.9	95.2	96.9	97.8	88.9
W_{FC2}	86.6	84.6	79.3	84.6	93.4	85.8	**94.3**	**97.3**	**93.2**	**99.5**	93.8	79.6	87.2	74.7	94.7	81.6	87.7	**91.2**	95.5	97.1	98.1	89.5
MLP block	–	82.4	–	–	93.6	–	–	–	–	–	–	–	–	–	94.3	–	86.4	–	–	–	–	–
(b) Which columns of W_O to update? (best average r value selected per dataset)																						
Gradient	**90.2**	**88.1**	**82.2**	88.2	**94.0**†	**86.7**†	95.3	97.4	93.2	98.9	95.9	82.5	87.9	78.3	**96.0**†	85.9	**88.2**†	91.8	**97.2**	97.6	99.1	91.2
Activation	89.9	87.8	82.1	**88.6**	**94.0**†	**86.7**†	95.6	97.7	93.1	99.1	**96.1**	82.6	89.1	78.5	**96.0**†	86.0	**88.2**†	91.7	96.7	98.2	99.2	91.3
Magnitude	89.8	87.8	81.8	**88.6**	**94.0**†	**86.7**†	95.5	97.8	93.0	99.1	96.0	82.7	89.3	**78.6**	**96.0**†	85.9	**88.2**†	91.3	97.0	98.1	99.1	91.3
Random (Ours)	89.6	88.0	81.9	88.5	**94.0**†	**86.7**†	**95.7**	**98.0**	**93.4**	**99.3**	**96.1**	**83.0**	**90.4**	78.3	**96.0**†	**86.2**	**88.2**†	**92.1**	**97.2**	**98.4**	**99.5**	**91.5**
(c) Comparison with other adaptation methods																						
Full fine-tuning	88.9	85.2	78.7	86.0	94.4	**87.5**	93.9	97.3	92.4	98.7	94.7	81.9	87.8	75.6	95.4	73.3	87.7	90.8	95.5	97.8	99.4	89.7
LINEAR	89.1	86.6	79.4	87.6	88.4	76.5	94.9	97.0	88.9	98.0	95.8	81.1	88.9	76.4	91.2	77.1	55.3	90.4	89.4	94.0	97.9	86.9
MLP	89.1	86.4	78.9	87.8	88.3	77.7	94.4	97.5	89.2	98.4	96.0	80.9	88.6	76.2	91.6	76.4	71.9	90.7	93.4	95.8	97.9	88.0
PARTIAL	88.8	86.5	78.6	87.4	88.1	76.6	94.9	96.9	89.0	98.0	96.0	80.6	88.5	76.3	90.9	77.7	56.1	90.6	89.5	94.2	97.6	86.8
BITFIT	89.4	87.9	80.7	87.8	92.5	83.2	95.2	97.6	93.1	**99.3**	95.7	82.2	89.3	77.7	95.2	83.5	79.0	90.1	96.4	97.2	99.1	90.1
ADAPTER	89.6	**88.4**	80.0	87.8	93.5	86.1	95.0	97.6	93.5	**99.3**	95.9	81.8	89.5	77.9	95.0	84.2	84.3	89.6	96.1	98.0	98.5	90.6
ADAPTFORMER	**89.7**	**88.4**	80.5	88.1	93.1	85.4	95.6	**98.1**	93.2	**99.3**	95.9	82.8	89.9	78.1	95.4	85.3	85.6	90.8	**97.3**	97.4	98.6	90.9
VPT-SHALLOW	88.8	86.7	79.1	87.4	90.6	73.2	95.1	97.2	92.4	99.1	96.0	80.4	89.4	76.6	91.6	70.4	76.5	89.3	96.2	96.4	98.7	88.1
VPT-DEEP	89.1	87.3	79.9	87.2	91.5	81.7	95.3	97.8	92.7	99.1	95.7	80.5	90.0	77.0	94.4	78.0	79.6	91.0	96.2	97.6	98.8	89.5
E^2VPT	88.3	86.6	79.7	87.4	91.2	81.0	94.5	96.9	92.7	99.2	95.4	79.6	88.7	76.3	93.7	72.7	80.9	90.6	96.2	96.8	98.6	88.9
SSF	89.4	88.1	80.5	87.7	92.7	83.7	95.3	97.8	93.2	99.2	95.7	82.1	88.9	77.4	95.3	82.6	80.7	90.5	96.4	97.2	99.0	90.2
LORA	88.7	87.5	79.3	86.3	93.4	86.4	94.3	97.1	93.0	99.0	94.0	80.4	88.5	76.4	95.4	81.4	86.5	91.1	95.1	97.4	98.9	90.0
SPT-ADAPTER	89.4	88.1	80.6	87.7	93.1	86.2	**95.8**	97.5	93.1	99.2	95.8	82.7	89.5	78.1	95.6	84.7	82.1	90.4	96.1	97.2	99.0	90.6
SPT-LORA	89.2	87.9	80.6	87.5	92.8	86.3	**95.8**	97.7	92.6	99.2	95.7	82.3	89.9	77.7	95.4	84.2	82.2	89.3	96.2	97.4	99.1	90.4
FACT-TK	88.8	87.8	80.5	87.5	93.0	85.4	95.3	97.6	92.8	99.2	95.5	81.5	88.9	77.4	95.4	79.9	85.1	91.5	96.3	97.2	97.9	90.2
FACT-TT	88.8	87.6	79.7	87.1	92.9	84.3	94.9	97.5	92.4	99.2	95.5	81.7	89.4	77.1	94.5	80.6	81.5	90.9	97.0	97.4	98.6	89.9
CONSOLIDATOR	**89.7**	87.4	81.5	87.1	93.0	83.4	94.5	97.3	92.7	99.0	95.8	81.8	89.5	77.0	94.6	78.0	81.9	91.1	96.9	96.2	99.4	89.9
ARC	89.4	88.2	80.7	88.1	92.6	84.1	**95.8**	97.8	92.9	99.2	96.0	82.8	89.8	78.2	95.6	**86.2**	82.5	90.7	**97.3**	97.0	98.8	90.7
GPS	89.1	86.4	80.7	86.6	**94.7**	85.5	94.8	97.6	**94.0**	**99.3**	94.4	77.9	88.4	76.9	94.9	61.6	87.7	90.8	96.7	97.4	99.3	89.3
RLRR	88.9	87.9	80.8	87.6	92.4	83.7	95.2	97.4	93.1	99.2	95.8	82.0	89.5	77.6	95.0	81.2	81.7	90.4	96.3	96.6	98.6	90.0
APLA (Ours)	89.6	88.0	**81.9**	**88.5**	94.0	86.7	95.7	98.0	93.4	**99.3**	**96.1**	**83.0**	**90.4**	**78.3**	**96.0**	**86.2**	**88.2**	**92.1**	97.2	**98.4**	**99.5**	**91.5**

† Optimal r=768 (full projection layer), all selection strategies are equivalent. Δ vs. full fine-tuning: ≤ -3 $\geq +3$

4.2 WHERE SHOULD A TRANSFORMER BE ADAPTED?

Comparing adaptation locations. Under the feature-reuse hypothesis in Section 3, we expect W_O to perform best on average, followed by W_V, with W_Q and W_K performing worst. We also expect feature-combination layers (W_O and W_{FC2}) to be strong adaptation targets. These are hypotheses about empirical performance, driven by but not guaranteed by the analysis.

In Table 1(a) we compare tuning individual ViT components across transformer blocks, together with the task head, while freezing the remaining backbone parameters. The four attention projections have equal parameter counts (0.59M per block), so their comparison controls for trainable capacity; the MLP matrices are four times larger and are compared at their native size. Across 21 classification datasets with DINOV2 ViT-B, W_O achieves the highest mean accuracy among the six attention and MLP matrices, 90.1% versus 89.7% for full fine-tuning (10 wins, 10 losses, and 1 tie across the 21 datasets), and is best or tied on 16 of 21 datasets. W_V ranks second at 89.6%, followed by W_{FC2} at 89.5%, while W_Q and W_K rank last at 88.7%; per dataset, W_O beats W_V on 17 of 21 (signed-rank $p < 0.001$; test details in Appendix B.3). Full fine-tuning is tuned on validation data like every

Table 2: **Main results on language.** RoBERTa-base (Liu et al., 2019) on GLUE (Wang et al., 2019). Top: tuning different components across all blocks and the task head; W_O leads on six of eight tasks. Bottom: our full fine-tuning run and published baselines using their original protocols. Metrics: the overall (matched and mismatched) accuracy for MNLI, Matthews correlation for CoLA, Pearson for STS-B, and accuracy otherwise; averages are computed from the displayed scores. Scores are on the GLUE development splits; further language results and settings are in Appendix B.6 (Tables 22–27).

	CoLA	STS-B	MRPC	RTE	SST-2	MNLI	QNLI	QQP	**Avg.**
W_Q	57.5	83.3	92.6	71.5	93.3	85.3	91.2	91.0	83.2
W_K	58.1	84.2	91.5	72.2	93.7	85.6	91.4	91.0	83.5
W_V	63.3	**91.3**	93.0	**81.6**	94.8	87.0	93.0	91.5	86.9
W_O **(Ours)**	**66.7**	90.9	**93.6**	79.4	**95.2**	87.1	**93.2**	91.6	**87.2**
Full fine-tuning	63.6	91.1	93.2	81.2	95.0	**87.7**	93.0	**92.0**	87.1
BitFit†	62.0	90.8	92.7	81.5	93.7	84.7	91.8	84.0	85.2
Adapter†	62.6	90.3	88.4	75.9	94.7	87.3	93.0	90.6	85.4
LoRA†	63.4	91.5	89.7	86.6	95.1	87.5	93.3	90.8	87.2
GaLore‡	60.1	90.8	92.0	79.8	94.4	87.2	92.2	91.1	85.9

Δ vs. our full fine-tuning: ≤ -3 $\geq +3$

method, yet a linear probe beats it on nine of the 21 datasets (Table 1(c)), the feature distortion documented by Kumar et al. (2022); this is the regime in which preserving pretrained features should pay.

Figure 1 broadens the comparison on four datasets (NABirds, Stanford Cars, AID, and ISIC2019) to include embedding, normalization, and block-level adaptation. Tuning W_O alone achieves 91.2%, exceeding full fine-tuning (90.7%), the entire attention block containing it (90.0%) and the whole MLP (89.2%). Thus, tuning more parameters does not necessarily improve adaptation.

In Table 2 we extend the analysis to language using RoBERTa-base (Liu et al., 2019) on eight GLUE tasks (Wang et al., 2019), adapting W_Q, W_K, W_V, or W_O across transformer blocks (two-decimal scores and selected configurations in Appendix Tables 22 and 23). W_O again achieves the highest mean score, 87.2 versus 86.9 for W_V, although most per-task differences are small. W_V leads on STS-B and RTE, suggesting that output adaptation is not always preferable and that new features may need to be learned on these tasks (Appendix A.6).

Across the vision and language component comparisons, W_O achieves the highest mean performance, followed by W_V, with W_Q and W_K ranking below both. Beyond the attention block locations, W_{FC2} ranks third. These rankings support our hypotheses in Section 3.

Applying other adaptation methods at W_O. To test whether the location finding extends to existing methods, we apply LoRA (Hu et al., 2022), AdaptFormer (Chen et al., 2022), and FacT (Jie & Deng, 2023) at W_O in DinoV2 ViT-B and compare with their standard placements. All three improve their mean accuracy across NABirds, Cars, AID, and ISIC, although not on every dataset (Appendix Table 12). On VTAB-1k the effect is small and mixed: LoRA and AdaptFormer gain 0.3 and 0.7 points on their averages, whereas FacT loses 0.4 (Appendix Table 13). APLA nevertheless has the highest four-task mean (91.6 versus 91.1 for the best relocated method) and VTAB average (79.0 versus 78.5 for the best relocated method and 78.9 for the best default one, FacT), and is best on Cars, AID, and ISIC.

4.3 How much of W_O, and which columns?

How many columns? Selecting the column count r on validation data improves mean test accuracy from 90.1% with full-projection tuning to 91.5% with random column subsets (Table 1(a–b)); full-projection tuning is the $r = 768$ point of the same grid, so this gain is that of selecting r on validation data. The selected configurations improve accuracy on 17 of 21 datasets (signed-rank $p < 0.001$) and match full-projection tuning on the remaining four, where all columns are selected. Improvements include 85.1% to 88.5% on Dogs, 75.4% to 78.3% on SUN397, and 82.2% to 86.2% on RSSCN7. Updating fewer columns can improve performance, rather than merely preserve it, consistent with the expectation in Section 3.2 that restricting the update limits overfitting when data is limited.

Table 3: **Low-data adaptation.** Accuracy for each task group on VTAB-1k (natural, specialized, structured) and the overall average, computed as the mean of the three group averages, over 19 tasks with 1,000 training images each, using ImageNet-21K-pretrained ViT-B/16 and Swin-B. Baselines are published results under the VPT protocol; APLA is our run.

		Full fine-tuning	Linear	MLP-3	Partial-1	BitFit	Adapter	AdaptFormer	VPT-shallow	VPT-deep	E^2VPT	SSF	LoRA	SPT-Adapter	SPT-LoRA	NOAH	Consolidator	FacT-TK	FacT-TT	ARC	RLRR	GPS	DA-VPT+	MIST	CaRA	DyT	HTA	SNELL	PYRA	FreqFit	**APLA (Ours)**
ViT-B	Natural	75.9	68.9	67.8	69.4	73.3	79.0	80.6	76.8	78.5	80.0	81.6	79.5	82.0	81.9	80.2	82.4	80.6	80.6	81.8	83.7	83.7	82.0	81.4	81.6	82.8	83.7	82.7	79.1	81.9	**84.6**
	Specialized	83.4	77.2	72.8	78.5	78.3	84.1	85.4	79.7	82.4	84.4	86.6	84.6	85.8	85.9	84.9	86.3	85.3	85.0	87.0	87.3	86.8	86.5	86.4	86.0	86.9	87.6	86.1	84.4	84.8	**88.5**
	Structured	47.6	26.9	30.6	34.2	44.1	58.5	58.8	47.0	55.0	57.4	59.0	60.5	61.4	61.3	61.3	60.9	60.7	60.5	61.4	61.8	61.9	60.0	62.4	61.7	**63.0**	62.8	61.8	60.6	57.4	62.7
	Mean	69.0	57.7	57.1	60.7	65.2	73.9	74.9	67.8	72.0	73.9	75.7	74.8	76.4	76.4	75.5	76.5	75.5	75.3	76.7	77.6	77.5	76.2	76.7	76.5	77.6	78.0	76.9	74.7	74.7	**78.6**
Swin-B	Natural	79.1	73.5	73.6	73.1	74.2	81.7	–	79.9	76.8	83.3	–	81.7	83.0	83.1	–	–	–	83.1	79.0	81.3	–	–	–	82.9	–	–	83.3	–	–	**84.4**
	Specialized	86.2	80.8	75.2	81.7	80.1	87.3	–	82.5	84.5	85.0	–	87.2	87.3	87.4	–	–	–	86.9	86.6	86.7	–	–	–	87.7	–	–	87.7	–	–	**87.8**
	Structured	59.7	33.5	35.7	35.0	42.4	61.2	–	37.8	53.4	57.4	–	60.1	62.1	60.4	–	–	–	62.1	59.9	59.0	–	–	–	63.0	–	–	61.4	–	–	**65.9**
	Mean	75.0	62.6	61.5	63.3	65.6	76.7	–	66.7	71.6	75.2	–	76.3	77.5	77.2	–	–	–	77.4	75.2	75.7	–	–	–	77.9	–	–	77.5	–	–	**79.4**

Δ vs. full fine-tuning: ≤ −10 ≥ +10

Appendix C.5 reports the distribution of selected column counts across vision datasets. On GLUE the selected configurations use 512 or all 768 columns (Table 23), the two counts evaluated.

Which columns? We compare random selection with three informed strategies that rank the columns of W_O in each block by the magnitude of their gradient, their output activation, or their pretrained weights, and keep the r largest; the selected columns are then trained exactly as in APLA. Across 21 classification datasets, random selection achieves the highest mean accuracy, 91.5%, compared with 91.2–91.3% for the informed strategies (Table 1(b)). Excluding the four full-projection cases, where the strategies coincide, random selection is best or tied on 12 of the remaining 17 datasets and beats each informed strategy on 12 to 14 of the datasets where they differ (signed-rank $p \leq 0.02$). The margins are small, so we conclude only that importance scoring does not help, and adopt random selection, which needs no scores.

Appendix C.4 shows that adapting more transformer blocks improves performance, and later blocks are more effective than the same number of early blocks, which supports applying APLA throughout the backbone.

4.4 How does APLA compare?

We benchmark APLA against other methods on vision and language tasks described in Section 4.1.

Image classification. In Table 1(c) APLA achieves the highest mean accuracy 91.5% across 21 classification datasets, exceeding AdaptFormer at 90.9% and full fine-tuning at 89.7%. It ranks first or second on 18 of 21 datasets, spanning fine-grained recognition, scene classification, satellite imagery, and medical imaging, and beats full fine-tuning on 19 and AdaptFormer on 16 (signed-rank $p < 0.001$ and $p = 0.001$). Full fine-tuning retains an advantage on Cars and Aircraft even when APLA tunes all 768 columns of W_O: 94.4% versus 94.0% and 87.5% versus 86.7%, respectively.

Low-data transfer. In Table 3 APLA has the highest VTAB-1k average (19 tasks with 1,000 training examples each) with both backbones. With ViT-B, it reaches 78.6%, exceeding HTA at 78.0% and RLRR and DyT at 77.6%; with Swin-B, it reaches 79.4%, ahead of CaRA at 77.9%. With Swin-B it leads every task group and the overall average.

Out-of-distribution robustness. Table 4(a) tests generalization beyond the adaptation distribution. An ImageNet-21K-pretrained ViT-B is adapted on ImageNet-1K (Deng et al., 2009) and evaluated on its validation set and on three shifted test sets: ImageNet-A (adversarially filtered natural images) (Hendrycks et al., 2021c), ImageNet-R (renditions) (Hendrycks et al., 2021a), and ImageNet-C (synthetic corruptions, scored by mean corruption error, mCE, lower is better) (Hendrycks & Dietterich, 2019). APLA achieves the highest accuracy on ImageNet-1K and ImageNet-A and the lowest mCE on ImageNet-C; its mCE of 32.9 improves on the strongest baseline, SSF at 41.5, by 8.6 points, while on ImageNet-R it trails GPS and SSF (55.5 versus 57.0 and 56.8). It beats full fine-tuning on all three shifted sets; the baselines are published results under their own recipes (Appendix Table 7).

Table 4: **Distribution shift and dense prediction.** **(a)** ImageNet-21K-pretrained ViT-B adapted on ImageNet-1K; mCE: lower is better. **(b)** ADE20K segmentation with SETR-PUP (ImageNet-21K ViT-L; single-/multi-scale mIoU) and COCO detection and instance segmentation with Mask R-CNN (ImageNet-1K Swin-T; box/mask AP).

(a) Distribution shift

		Full fine-tuning	LINEAR	BITFIT	ADAPTER	VPT-SHALLOW	VPT-DEEP	SSF	GPS	**APLA (Ours)**
Accuracy	IN-1K	83.6	82.0	82.7	82.7	82.1	82.5	83.1	83.9	**84.0**
	IN-A	34.5	33.9	42.1	42.2	30.9	39.1	45.9	46.1	**46.9**
	IN-R	51.3	52.9	55.9	54.1	53.7	53.5	56.8	**57.0**	55.5
mCE ↓	IN-C	46.5	46.9	41.9	42.7	46.9	43.1	41.5	42.0	**32.9**

Δ vs. full fine-tuning: ≤ -3 $\geq +3$

(b) Segmentation and detection

		Full fine-tuning	BITFIT	ADAPTER	ADAPTFORMER	VPT-DEEP	SSF	LORA	SPT-ADAPTER	SPT-LORA	**APLA (Ours)**
ADE20K mIoU	SS	48.6	43.4	44.4	44.3	42.1	45.6	43.9	45.2	45.4	**48.3**
	MS	50.1	45.3	46.6	46.2	44.1	47.4	45.9	47.2	47.5	**49.5**
COCO box AP	AP	43.7	33.7	37.6	35.1	33.8	34.9	37.1	–	–	**38.1**
	AP_{50}	66.6	57.8	61.1	59.1	57.6	58.9	60.9	–	–	**61.8**
	AP_{75}	47.7	35.0	40.2	36.9	35.3	36.1	39.5	–	–	**40.9**
COCO mask AP	AP	39.8	32.7	35.6	33.8	32.5	33.5	35.2	–	–	**35.9**
	AP_{50}	63.3	54.7	58.2	56.0	54.5	55.8	57.7	–	–	**58.7**
	AP_{75}	42.7	33.9	37.8	35.6	33.9	34.7	37.2	–	–	**37.9**

Table 5: **Comparing different pretraining strategies, model capacities, and training costs.** Mean accuracy over NABirds, Stanford Cars, AID and ISIC2019 for ViT-B pretrained with supervised ImageNet-21K or OpenCLIP, and for DINOv2 ViT-S to ViT-g. The last two rows give peak training memory and training time for DINOv2 ViT-B (batch size 64): APLA uses 41% less memory and 35% less time than full fine-tuning. Per-dataset results in Tables 14–19; cost measurements in Table 20.

		Full fine-tuning	LINEAR	MLP	PARTIAL	BITFIT	ADAPTER	ADAPTFORMER	VPT-SHALLOW	VPT-DEEP	E^2VPT	SSF	LORA	SPT-ADAPTER	SPT-LORA	FACT-TK	FACT-TT	ARC	RLRR	**APLA (Ours)**
Pretraining strategy	ImageNet-21K	85.7	64.9	68.8	67.9	81.8	80.9	83.3	74.6	82.9	83.6	86.1	85.8	84.0	84.2	83.7	82.6	86.3	86.1	**88.7**
	OpenCLIP	88.6	79.4	82.8	79.6	85.7	88.3	88.2	79.6	79.5	85.6	86.7	87.8	85.7	85.8	87.5	86.5	87.2	87.6	**88.9**
Model capacity	ViT-S	86.2	76.1	80.9	75.8	84.8	87.9	87.2	82.8	84.0	84.2	84.7	87.1	87.2	87.6	86.9	85.4	86.0	84.4	**88.0**
	ViT-B	90.7	80.4	84.6	80.4	88.7	90.3	90.6	86.4	88.2	88.1	89.2	90.8	89.7	89.6	90.3	89.1	89.7	89.3	**91.6**
	ViT-L	92.1	83.6	86.2	83.6	90.0	91.9	91.7	88.3	90.2	90.9	91.0	92.1	90.0	90.5	91.5	91.3	91.1	91.8	**92.9**
	ViT-g	92.6	85.5	87.8	85.2	91.8	92.3	92.8	88.9	91.9	91.7	91.9	92.6	89.5	89.4	92.0	91.6	91.9	92.3	**93.4**
Training cost (ViT-B)	Memory (GB)	15.9	**1.9**	**1.9**	–	9.6	10.5	9.6	11.7	12.7	13.7	11.8	12.4	14.7	15.3	13.6	13.4	10.4	15.0	9.4
	Time (ms/batch)	959	**285**	**285**	–	643	689	656	771	789	789	756	753	705	763	915	910	684	867	628

Δ vs. full fine-tuning: ≤ -3 $\geq +3$

Segmentation and detection. APLA leads the compared PEFT methods on ADE20K segmentation and COCO detection and instance segmentation (Table 4(b)). On ADE20K it reaches 48.3/49.5 single-/multi-scale mIoU against 48.6/50.1 for full fine-tuning. The gap is larger on COCO: 38.1 box and 35.9 mask AP against 43.7 and 39.8 for full fine-tuning. Under the protocol of MONA (Yin et al., 2025) (Table 21), APLA leads the three classification columns but trails MONA on dense prediction.

Language. In Table 2 we compare adaptation methods on GLUE tasks (settings provided in Table 26 and per-task scores in Table 25). APLA achieves a mean GLUE score of 87.2 against 87.1 for our full fine-tuning, with four wins and four losses. Published means are 85.2 for BitFit, 85.4 for Adapter, and 87.2 for LoRA (Hu et al., 2022), and 85.9 for GaLore (Zhao et al., 2024a).

On five-shot MMLU after Alpaca fine-tuning of Llama 3.2 (1B) and Llama 3.1 (8B) (Grattafiori et al., 2024), APLA scores 31.3% and 65.4%, against 31.3% and 65.3% for full fine-tuning, 30.4% and 65.2% for LoRA, and 31.5% (1B only) for GaLore (Table 27).

Pretraining and model capacity. Table 5 varies the pretraining strategy and the capacity of the backbone. Across its four datasets, APLA achieves the highest mean accuracy with both supervised ImageNet-21K pretraining (88.7%) and OpenCLIP image–text pretraining (88.9%). It also achieves the highest mean across ViT-S, ViT-B, ViT-L, and ViT-g (DINOV2 pretraining). With the Swin-B results in Table 3, our finding extends across pretraining, capacity, and architecture.

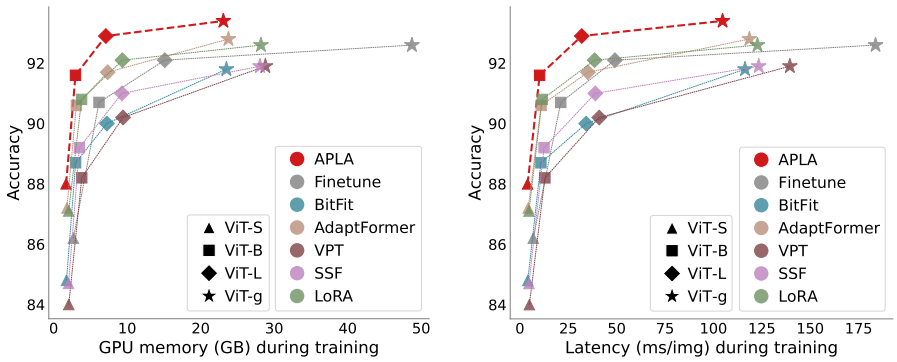

Figure 2: **Performance vs. compute cost.** Mean accuracy over four datasets against training GPU memory (left) and latency (right) for ViT-S/ViT-B/ViT-L/ViT-g and other adaptation methods.

4.5 What is the computational cost?

With DINOV2 ViT-B and batch size 64, APLA reduces peak training memory from 15.9 to 9.4 GB (41%) and training time by 35% relative to full fine-tuning, and uses less of both than BitFit, AdaptFormer, and LoRA (Table 5, bottom; Appendix Table 20). Across model scales (Figure 2), the reductions for ViT-g reach 53% in training memory and 43% in training time.

At inference, APLA adds no computation and matches the memory and time of full fine-tuning. Training savings come from frozen weights: their gradients are never computed, so autograd stores neither the inputs saved for their weight gradients nor gradients and optimizer states, although backpropagation through them remains necessary; restricting the columns of W_O further has little effect on memory or time across the tested r (Appendix Figure 4), so its benefit is accuracy, not cost.

5 Discussion and Conclusion

Adaptation location is a key design choice: tuning W_O can outperform larger trainable components and full fine-tuning, and relocating existing methods to W_O can improve them. The ranking predicted by our hypothesis holds across the vision and language comparisons, although we do not measure feature reuse directly. Adapting W_O must be seen across the whole backbone: at a single block it only recombines existing features (Appendix A.6), but across blocks one block's recombination changes what later blocks read, so new features arise by composition.

Validation-selected column subsets beat full-projection tuning, and random selection achieves the highest mean of the tested strategies without any scoring.

Our results are consistent with the hypothesis that adapting feature combinations may preserve useful pretrained structure while reducing the need to learn from limited examples. APLA's strong VTAB results support this interpretation. Still, some tasks may benefit more from changing feature extraction. Cars and Aircraft require subtle distinctions between visually similar categories, potentially relying on details that pretraining does not preserve in attention features. Likewise, the advantage of W_V on STS-B and RTE may reflect a benefit from adapting features for semantic similarity and entailment. Dense prediction requires spatial detail for boundaries and localization, which may help explain why all the compared PEFT methods (save MONA) trail full fine-tuning, particularly on COCO. These are possible explanations rather than demonstrated mechanisms; small gaps may also reflect noise.

Limitations. We do not systematically evaluate combinations of adaptation locations, such as jointly tuning W_O and W_V, and column restriction was applied only to W_O; whether rows or sparse masks of W_O, or columns of W_V or W_{FC2}, would behave the same is untested. The evidence is stronger in vision than in language, where APLA is competitive but not consistently superior; compute also limited our evaluation of larger language models. On dense prediction, APLA and the other PEFT methods trail full fine-tuning, particularly on COCO.

AI USE STATEMENT

We used GPT-6 Astra and Claude Fable to assist with writing and editing the manuscript. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

Implementation details, data splits, training protocols, and selection rules for all experiments are given in Appendix B, and all main-text results are backed by complete per-dataset tables in Appendices B and C. Our code is available in anonymized form at `https://anonymous.4open.science/r/attention-projection-adaptation-iclr2027-9082/`; it contains the APLA module which replaces a standard attention block, together with the configuration files of the classification, segmentation, and detection experiments and the fixed column-index files used for the reported runs. The code will be released publicly upon acceptance.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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. 7319–7328, 2021.

Asia Pacific Tele-Ophthalmology Society. APTOS 2019 blindness detection, 2019. URL `https://kaggle.com/competitions/aptos2019-blindness-detection`.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 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)*, pp. 1–9, 2022.

Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L Alexander, David W Jacobs, and Peter N Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2011–2018, 2014.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. TinyTL: Reduce memory, not parameters for efficient on-device learning. *Advances in Neural Information Processing Systems*, 33:11285–11297, 2020.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open MMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. AdaptFormer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, 2014.

Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 168–172. IEEE, 2018.

Marc Combalia, Noel CF Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Cristina Carrera, Alicia Barreiro, Allan C Halpern, Susana Puig, et al. BCN20000: Dermoscopic lesions in the wild. *arXiv preprint arXiv:1908.02288*, 2019.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. The efficiency misnomer. In *International Conference on Learning Representations*, 2022.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Alessio Devoto, Federico Alvetreti, Jary Pomponi, Paolo Di Lorenzo, Pasquale Minervini, and Simone Scardapane. Adaptive layer selection for efficient vision transformer fine-tuning. *arXiv preprint arXiv:2408.08670*, 2024.

Wei Dong, Dawei Yan, Zhijun Lin, and Peng Wang. Efficient adaptation of large vision transformer via adapter re-composing. *Advances in Neural Information Processing Systems*, 36, 2023.

Wei Dong, Yuan Sun, Yiting Yang, Xing Zhang, Zhijun Lin, Qingsen Yan, Haokui Zhang, Peng Wang, Yang Yang, and Hengtao Shen. Efficient adaptation of pre-trained vision transformer via Householder transformation. In *Advances in Neural Information Processing Systems*, volume 37, 2024a.

Wei Dong, Xing Zhang, Bihui Chen, Dawei Yan, Zhijun Lin, Qingsen Yan, Peng Wang, and Yang Yang. Low-rank rescaled vision transformer fine-tuning: A residual design approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16101–16110, 2024b.

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.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Mohammad Fahes, Tuan-Hung Vu, Andrei Bursuc, Patrick Pérez, and Raoul de Charette. CLIP's visual embedding projector is a few-shot cornucopia. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3254–3264, 2026.

Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical Report CNS-TR-2007-001, California Institute of Technology, 2007.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 4884–4896, 2021.

Cheng Han, Qifan Wang, Yiming Cui, Zhiwen Cao, Wenguan Wang, Siyuan Qi, and Dongfang Liu. E^2VPT: An effective and efficient approach for visual prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17491–17502, 2023.

Tianxiang Hao, Hui Chen, Yuchen Guo, and Guiguang Ding. Consolidator: Mergable adapter with group connections for visual adaptation. In *International Conference on Learning Representations*, 2023.

Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11825–11835, 2023.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021b.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021c.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, 2021. URL `https://doi.org/10.5281/zenodo.5143773`.

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*, pp. 709–727. Springer, 2022.

Shibo Jie and Zhi-Hong Deng. FacT: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 1060–1068, 2023.

Ruochen Jin, Bojian Hou, Jiancong Xiao, Weijie Su, and Li Shen. Fine-tuning attention modules only: Enhancing weight disentanglement in task arithmetic. In *International Conference on Learning Representations*, 2025.

Jakob Nikolas Kather, Cleo-Aron Weis, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner. Multi-class texture analysis in colorectal cancer histology. *Scientific Reports*, 6:27988, 2016.

Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

Md Kowsher, Tara Esmaeilbeig, Chun-Nam Yu, Chen Chen, Mojtaba Soltanalian, and Niloofar Yousefi. RoCoFT: Efficient finetuning of large language models with row-column updates. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26659–26678, 2025.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, Kanae Kawai Miyake, Mia Gorovoy, and Daniel L Rubin. A curated mammography data set for use in computer-aided detection and diagnosis research. *Scientific Data*, 4:170177, 2017.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Chelsea Finn, and Percy Liang. Surgical fine-tuning improves adaptation to distribution shifts. In *International Conference on Learning Representations*, 2023.

Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35: 109–123, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Son Thai Ly and Hien V Nguyen. Enhancing parameter-efficient fine-tuning of vision transformers through frequency-based adaptation. *arXiv preprint arXiv:2411.19297*, 2024.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505. IEEE, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Li Ren, Chen Chen, Liqiang Wang, and Kien Hua. DA-VPT: Semantic-guided visual prompt tuning for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4353–4363, 2025.

Shufan Shen, Junshu Sun, Xiangyang Ji, Qingming Huang, and Shuhui Wang. Expanding sparse tuning for low memory usage. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24193–24205, 2021.

Ningyuan Tang, Minghao Fu, and Jianxin Wu. Minimal interaction separated tuning: A new paradigm for visual adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25208–25217, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. In *European Conference on Computer Vision*, pp. 497–515. Springer, 2022.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5:180161, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604, 2015.

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.

Lokesh Veeramacheneni, Moritz Wolter, Hilde Kuehne, and Juergen Gall. Canonical rank adaptation: An efficient fine-tuning strategy for vision transformers. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61108–61125, 2025.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, 2025.

Gui-Song Xia, Jingwen Hu, Fan Hu, Baoguang Shi, Xiang Bai, Yanfei Zhong, Liangpei Zhang, and Xiaoqiang Lu. AID: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492. IEEE, 2010.

Yizhe Xiong, Hui Chen, Tianxiang Hao, Zijia Lin, Jungong Han, Yuesong Zhang, Guoxin Wang, Yongjun Bao, and Guiguang Ding. PYRA: Parallel yielding re-activation for training-inference efficient task adaptation. In *European Conference on Computer Vision*, pp. 455–473. Springer, 2024.

Dongshuo Yin, Yiran Yang, Zhechao Wang, Hongfeng Yu, Kaiwen Wei, and Xian Sun. 1% vs 100%: Parameter-efficient low rank adapter for dense predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20116–20126, 2023.

Dongshuo Yin, Leiyi Hu, Bin Li, Youqun Zhang, and Xue Yang. 5%>100%: Breaking performance shackles of full fine-tuning on visual recognition tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20071–20081, 2025.

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.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023.

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

Zhi Zhang, Qizhe Zhang, Zijun Gao, Renrui Zhang, Ekaterina Shutova, Shiji Zhou, and Shanghang Zhang. Gradient-based parameter selection for efficient fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28566–28577, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143, 2024a.

Wangbo Zhao, Jiasheng Tang, Yizeng Han, Yibing Song, Kai Wang, Gao Huang, Fan Wang, and Yang You. Dynamic tuning towards parameter and inference efficiency for ViT adaptation. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6881–6890, 2021.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641, 2017.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.

Qin Zou, Lihao Ni, Tong Zhang, and Qian Wang. Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2321–2325, 2015.

Appendix

We first present the analysis, followed by additional experimental details and results.

- Section A analyzes the changes expressible through different adaptation locations, the conditions under which output adaptation is sufficient, and the restriction imposed by selecting projection columns.
- Section B provides experimental details:
  - Sections B.1 and B.2: all datasets (Table 6) and all benchmarks and baselines (Table 7).
  - Section B.3: image classification, including hyperparameter selection and column selection.
  - Sections B.4 and B.5: semantic segmentation, object detection and instance segmentation, and the MONA protocol.
  - Section B.6: GLUE and MMLU.
  - Section B.7: cost measurement.
- Section C provides additional results:
  - Section C.1: classification performance and computational cost across model scales.
  - Section C.2: training and inference costs of APLA and competing methods.
  - Section C.3: the effect of rank on training costs for low-rank methods applied to W_O.
  - Section C.4: adapting increasing numbers of ViT blocks.
  - Section C.5: the number of projection columns selected for adaptation.
  - Section C.6: visualizations of features produced by different adaptation methods.

A Theoretical Analysis of Adaptation Locations

This appendix makes precise the informal argument of Section 3.1, starting from the per-location changes of Eq. (5), and the rank statement of Section 3.2. We use the notation of Section 3.1: Z is the block input, $X = \mathrm{LN}_1(Z)$ the normalized attention input, $A_i(X)$ and $H_i(X)$ the attention map and contextualized features of Eq. (1), $Y = \sum_i H_i(X) W_{O_i}$ the attention output of Eq. (2), and $R_1 = Z + \mathrm{LS}_1(Y)$ and R_2 the residual stream after the attention and MLP sub-blocks (Eq. (3)). Throughout, one location is adapted while all other parameters, including normalization and LayerScale, keep their pretrained values, and the input X is held fixed[2]. The analysis concerns expressivity only: which changes each adaptation location can produce at a fixed input. It does not rank locations by optimization or generalization; that is what Table 1(a) measures.

The appendix has six parts. Appendix A.1 derives the six per-location changes of Eq. (5). Appendix A.2 characterizes exactly which head matrices $M_i = W_{V_i} W_{O_i}$, the per-token map applied by head i (Eq. (8)), adapting W_O and adapting W_V can reach, generalizing the example of Appendix A.6, and Appendix A.3 does the same for W_Q and W_K. Appendix A.4 extends the comparison to W_{FC1} and W_{FC2}, which Table 1(a) also adapts individually, and shows where in the block each update enters. Appendix A.5 turns to the column-restricted update of Eq. (6): its gradient, why its rank bound holds for the whole of training, and why training cost is measured rather than inferred from it. Appendix A.6 illustrates the restrictions with a two-head example.

A.1 Derivation of the per-location changes, Eq. (5)

With Z and all other parameters fixed, Eq. (3) gives $\Delta R_1 = \mathrm{LS}_1(\Delta Y)$ when an attention matrix is adapted, and $\Delta R_2 = \mathrm{LS}_2(\Delta F)$ with R_1 unchanged when an MLP matrix is adapted; we drop LS as stated above. Substituting Eq. (1) into Eq. (2),

$$Y = \sum_i A_i(X)\, X\, W_{V_i} W_{O_i}, \tag{7}$$

[2]For simplicity, biases are omitted, and LayerScale is dropped from the equations: it is a fixed diagonal scaling, so none of the statements below depend on it.

and each row of Eq. (5) follows by changing one factor while the others stay fixed. For W_O only the last factor changes, $\Delta Y = \sum_i A_i(X) X W_{V_i}\, \Delta W_{O_i} = \sum_i H_i(X)\, \Delta W_{O_i}$. For W_V only the third, $\Delta Y = \sum_i A_i(X)\, X\, \Delta W_{V_i}\, W_{O_i}$. For W_Q or W_K only the attention map changes, $A_i(X) \to A_i'(X)$, so $\Delta Y = \sum_i (A_i'(X) - A_i(X))\, X\, W_{V_i} W_{O_i}$; this row is not linear in the parameter change, because A_i' depends on it through the softmax. For the MLP, $F(X_2) = \phi(X_2 W_{\mathrm{FC1}}) W_{\mathrm{FC2}}$ with $X_2 = \mathrm{LN}_2(R_1)$ fixed gives $\Delta F = \phi(X_2 W_{\mathrm{FC1}})\, \Delta W_{\mathrm{FC2}}$ for W_{FC2} and $\Delta F = \Delta\phi\, W_{\mathrm{FC2}}$ with $\Delta\phi = \phi(X_2(W_{\mathrm{FC1}} + \Delta W_{\mathrm{FC1}})) - \phi(X_2 W_{\mathrm{FC1}})$ for W_{FC1}, the latter nonlinear through ϕ. All six identities are exact rather than linearizations, because in each case only the varied factor changes.

A.2 Adapting W_O versus W_V: what a head reads and where it writes

Eq. (7) expresses each head's contribution through a single $d \times d$ matrix,

$$Y(X) = \sum_i A_i(X)\, X\, M_i, \qquad M_i = W_{V_i} W_{O_i}, \tag{8}$$

so at a fixed attention map the head is determined by its head matrix M_i: its column space is what the head reads from each token, and its row space is where the head writes in the residual stream. Adapting W_{O_i} or W_{V_i} changes M_i by

$$\Delta M_i^{(O)} = W_{V_i}\, \Delta W_{O_i}, \qquad \Delta M_i^{(V)} = \Delta W_{V_i}\, W_{O_i}. \tag{9}$$

Since ΔW_{O_i} ranges over all $d_h \times d$ matrices, adapting W_{O_i} can reach a target head matrix $M_i^\star$ if and only if $\mathrm{col}(M_i^\star) \subseteq \mathrm{col}(W_{V_i})$, that is, if and only if $M_i^\star = W_{V_i} B_i$ for some B_i; setting $W_{O_i} = B_i$ then realizes $M_i^\star$ with the attention maps unchanged. Symmetrically, adapting W_{V_i} can reach $M_i^\star$ if and only if $\mathrm{row}(M_i^\star) \subseteq \mathrm{row}(W_{O_i})$. In words, adapting W_{O_i} keeps the input features the head reads and frees the directions it writes to, while adapting W_{V_i} keeps the output directions and frees the features it reads; neither set of reachable head matrices contains the other. For head 1 of the two-slot example in Appendix A.6, $\mathrm{col}(W_{V_1})$ is the x_1 axis and $\mathrm{row}(W_{O_1})$ is slot 1, so adapting W_{O_1} reaches exactly the heads that read x_1 and write to any combination of slots, and adapting W_{V_1} exactly the heads that read any combination of x_1 and x_2 and write to slot 1.

A.3 Adapting W_Q or W_K: changing the attention map

With W_{V_i} and W_{O_i} fixed, adapting W_{Q_i} or W_{K_i} leaves every M_i unchanged and alters only the attention maps, $A_i(X) \to A_i'(X)$:

$$\Delta Y_{Q,K}(X) = \sum_i (A_i'(X) - A_i(X))\, X\, M_i. \tag{10}$$

It changes the attention map, that is, which tokens are averaged and with what weights, and hence the contextualized features $H_i(X)$; but the per-token readout M_i is untouched, so what each head reads from a token and where it writes stay fixed. Adapting W_O is the complementary case: at a fixed input it keeps the attention map and changes where the averaged features are written.

A.4 Adapting W_{FC1} or W_{FC2}

Table 1(a) also adapts the two MLP matrices individually, so we state the analogous distinction, using the last two rows of Eq. (5). As with W_V and W_O, adapting W_{FC1} changes the features before the nonlinearity and adapting W_{FC2} recombines nonlinear features that already exist. The MLP acts token-wise, but its input already carries context from attention.

The two direct locations, W_O and W_{FC2}, enter the block at different points (Eq. (5)). A W_O change alters R_1 and therefore also the input of the frozen MLP, so that

$$\Delta R_2 = \Delta R_1 + F(\mathrm{LN}_2(R_1 + \Delta R_1)) - F(\mathrm{LN}_2(R_1)), \tag{11}$$

that is, it reaches the block output directly through the residual path and indirectly through the MLP. A W_{FC2} change leaves R_1 fixed and enters only at the second residual addition. These are structural differences, not an ordering; Table 1(a) is what ranks the locations in the evaluated settings.

A.5 Column-restricted updates: gradient and rank bound

Section 3.2 defines the column-restricted update of Eq. (6), $W_O = W_O^{\text{pre}} + \Delta W_O$ with $\Delta W_O[:, j] = 0$ for $j \notin J$, and states $\text{rank}(\Delta W_O) \leq r$. Here we explain the details behind that statement: the gradient that training follows, why the rank bound holds for the accumulated update over all of training rather than per step, and what the bound does not imply for the activations and for training cost.

Let $S \in \{0, 1\}^{d \times r}$ be the selection matrix whose columns are the standard basis vectors e_j for $j \in J$, so that every admissible update can be written $\Delta W_O = US^\top$ with $U \in \mathbb{R}^{d \times r}$ holding the selected columns. An update to W_O has the form $\Delta W_O = US^\top$ if and only if its unselected columns are zero, so column restriction is a further restriction of adapting the full W_O: sufficiency of a full W_O update does not by itself guarantee sufficiency of a given subset of columns, which is why the number and choice of columns are tested empirically (Table 1(b), Appendix C.5). With $G = \partial \mathcal{L} / \partial Y$, the gradient with respect to the trainable block is

$$\nabla_U \mathcal{L} = H(X)^\top G\, S, \tag{12}$$

the columns of the full gradient $H(X)^\top G$ at the selected positions. Because the support S is fixed before training, every step and hence the accumulated change ΔW_O has at most r nonzero columns, so the rank bound of Eq. (6) holds for the complete learned update and not only per step; a method whose per-step gradient is low rank, such as gradient projection (Zhao et al., 2024a), does not obtain this bound on the accumulated change. Two things the bound does not imply. Unselected coordinates of the activations are not held fixed, since later layers couple all coordinates. And restricting the trainable weights does not remove backpropagation through the frozen operations between adapted blocks; freezing can reduce gradient and optimizer storage, but the savings depend on how the trainable parameters are allocated, and gradient masking alone need not reduce optimizer state. Training memory and time are therefore measured (Tables 5 and 20) rather than inferred from the rank bound.

A.6 A Toy Example of Adaptation-Location Restrictions

Suppose adaptation must make an existing contextualized feature contribute to a different entry ("slot") of the residual stream. Alternatively, it may require a head to read a feature it previously ignored. These goals illustrate the different freedoms offered by W_O and W_V adaptation.

Consider one block at a fixed input $X \in \mathbb{R}^{n \times 2}$, ignoring biases and LayerScale, and write A_i for the attention map $A_i(X)$. Let $x_1 = X_{:,1}$ and $x_2 = X_{:,2}$ collect the two input slots across tokens. There are two heads, each with one feature ($d_h = 1$):

$$W_{V_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad W_{O_1} = [1 \quad 0], \qquad W_{V_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad W_{O_2} = [0 \quad 1].$$

Head 1 reads slot 1, combines its values across tokens using A_1, and writes the result to slot 1. Head 2 similarly reads and writes slot 2 using A_2. Thus,

$$H_1(X) = A_1 x_1 =: u, \qquad H_2(X) = A_2 x_2 =: v, \qquad Y = [u, v].$$

Here, u and v each contain one scalar per token. Also define $w = A_2 x_1$: the feature head 2 would produce if it read slot 1 instead. To make the restrictions unambiguous, assume u, v, and w are linearly independent across tokens. This excludes accidental equivalences between the features; different attention maps alone do not guarantee this condition.

(A) Reuse an existing feature in another slot. Suppose adaptation requires

$$Y_A = [u, v + u].$$

The goal is to add head 1's existing feature u to slot 2 while preserving both original contributions. Setting $W_{O_1} = (1, 1)$ achieves this directly, without changing either head's contextualized feature or attention map.

Value adaptation alone cannot achieve this target with the output projection frozen. Head 1 still writes only to slot 1. Head 2 can read any combination of x_1 and x_2, but its fixed attention map makes the resulting feature a combination of w and v, which cannot supply the missing u.

(B) Read an additional feature within a head. Now suppose adaptation instead requires

$$Y_B = [u, v + w].$$

The additional feature uses slot 1, but combines it across tokens using head 2's attention map. Setting $W_{V_2} = (1, 1)^\top$ achieves this:

$$H_2^{\text{new}}(X) = A_2(x_1 + x_2) = w + v.$$

Output adaptation alone cannot achieve this target because the retained contextualized features are $H(X) = [u, v]$. Changing W_O can combine u and v, but cannot produce the independent feature w.

Desired change	Output adaptation	Value adaptation
(A) Add u to slot 2	$W_{O_1} = (1, 1)$	Head 1 confined to slot 1; head 2 yields only w, v
(B) Add w to slot 2	w is outside the span of $H(X)$	$W_{V_2} = (1, 1)^\top$

The distinction is between changing how the contextualized features $H(X)$ contribute and changing which features a head reads. With W_O frozen, each head's contribution remains in the row space of its W_{O_i}. Even though the two heads together can write to both slots, their different attention maps prevent compensation in case (A). With $H(X)$ frozen, output adaptation is instead restricted to combinations of those features, as case (B) shows. Neither form of adaptation therefore subsumes the other.

Query and key adaptation changes A_i but not what is read or where it is written: with W_V and W_O frozen, slot 2 remains a convex combination of x_2 across tokens, so neither target is reachable for general X.

Our feature-reuse hypothesis motivates output adaptation when the required features are already available, as in case (A); value adaptation offers a complementary freedom when they are not, as in case (B). Their relative performance remains an empirical question. These restrictions concern one block at a fixed input: earlier adapted blocks can change that input, and subsequent blocks can further transform its output.

B Experimental Details

This appendix lists every dataset (Table 6) and every benchmark (Table 7) used in Section 4, and states the training and selection protocol behind each result table.

B.1 Datasets and Metrics

Splits. For the classification datasets we use the official train/validation/test splits where they exist and otherwise the splits of Jia et al. (2022). Where no validation split is provided, one is held out from the training data with a fixed seed (for example 30% of the NABirds training set); ISIC2019 and APTOS2019, whose test labels are not public, are split by holding out 20% and 30% of the labelled data, respectively, and dividing that portion equally into validation and test. VTAB-1k provides its splits. For distribution shift, models are adapted on the ImageNet-1K training set and evaluated on the ImageNet-1K validation set and on ImageNet-A, -R, and -C. ADE20K and MS COCO use their standard training and validation sets. For GLUE, models are trained on the official training splits and evaluated on the official development splits, which are used as the reported results. Table 6 gives the metric reported for each dataset; except for GLUE, reported test metrics use checkpoints selected on validation data.

B.2 Benchmarks and Baselines

Baselines. The traditional baselines are full fine-tuning, a linear head on frozen features (Linear), a three-layer MLP head (MLP-3), and tuning the last block together with the head (Partial-1), following Jia et al. (2022). The adaptation methods are BitFit (Ben Zaken et al., 2022), Adapter (Houlsby et al., 2019), AdaptFormer (Chen et al., 2022), VPT-shallow and VPT-deep (Jia et al., 2022), E^2VPT (Han et al., 2023), SSF (Lian et al., 2022), LoRA (Hu et al., 2022), SPT-Adapter and SPT-LoRA (He et al., 2023), NOAH (Zhang et al., 2022), FacT-TK and FacT-TT (Jie & Deng, 2023), Consolidator

Table 6: All datasets used in this paper, grouped by the experiment they serve. Classes: number of target classes (ADE20K and COCO: semantic and object categories). Metric: the score reported in the tables; for classification datasets it is also the model-selection metric.

Dataset	Domain	Classes	Metric
Image classification (21 datasets; Tables 1, 5, 8–11)			
CUB-200-2011 (Wah et al., 2011)	fine-grained, birds	200	accuracy
NABirds (Van Horn et al., 2015)	fine-grained, birds	555	accuracy
Birdsnap (Berg et al., 2014)	fine-grained, birds	500	accuracy
Stanford Dogs (Khosla et al., 2011)	fine-grained, dogs	120	accuracy
Stanford Cars (Krause et al., 2013)	fine-grained, cars	196	accuracy
FGVC-Aircraft (Maji et al., 2013)	fine-grained, aircraft	100	mean per-class accuracy
Caltech-101 (Fei-Fei et al., 2006)	general objects	101	mean per-class accuracy
Caltech-256 (Griffin et al., 2007)	general objects	257	mean per-class accuracy
CIFAR-10 (Krizhevsky, 2009)	general objects	10	accuracy
CIFAR-100 (Krizhevsky, 2009)	general objects	100	accuracy
Oxford-IIIT Pets (Parkhi et al., 2012)	fine-grained, pets	37	mean per-class accuracy
DTD (Cimpoi et al., 2014)	textures	47	accuracy
MIT Indoor (Quattoni & Torralba, 2009)	scenes	67	mean per-class accuracy
SUN397 (Xiao et al., 2010)	scenes	397	accuracy
AID (Xia et al., 2017)	aerial scenes	30	accuracy
RSSCN7 (Zou et al., 2015)	aerial scenes	7	mean per-class accuracy
ISIC2019 (Tschandl et al., 2018; Codella et al., 2018; Combalia et al., 2019)	dermoscopy	8	macro recall
APTOS2019 (Asia Pacific Tele-Ophthalmology Society, 2019)	retinal fundus	5	quadratic-weighted kappa
DDSM (Lee et al., 2017)	mammography	2	ROC-AUC
Colorectal (Kather et al., 2016)	histology	8	accuracy
Pneumonia (Kermany et al., 2018)	chest X-ray	2	ROC-AUC
Low-data adaptation, VTAB-1k (Zhai et al., 2019): 19 tasks, 1,000 training images each (Table 3)			
Natural (7)	CIFAR-100 (100), Caltech101 (102), DTD (47), Flowers102 (102), Pets (37), SVHN (10), SUN397 (397)	10–397	accuracy
Specialized (4)	Patch Camelyon (2), EuroSAT (10), Resisc45 (45), Retinopathy (5)	2–45	accuracy
Structured (8)	Clevr-Count (8), Clevr-Dist (6), DMLab (6), KITTI-Dist (4), dSprites-Loc (16), dSprites-Ori (16), SmallNORB-Azim (18), SmallNORB-Elev (9)	4–18	accuracy
Distribution shift (Table 4a)			
ImageNet-1K (Deng et al., 2009)	adaptation set and in-distribution test	1,000	accuracy
ImageNet-A, ImageNet-R (Hendrycks et al., 2021c;a)	natural adversarial examples; renditions	200	accuracy
ImageNet-C (Hendrycks & Dietterich, 2019)	corruptions	1,000	mCE (lower is better)
Dense prediction (Table 4b)			
ADE20K (Zhou et al., 2019; 2017)	semantic segmentation	150	mIoU (single-/multi-scale)
MS COCO (Lin et al., 2014)	detection, instance segmentation	80	box AP, mask AP
MONA protocol (Yin et al., 2025) (Table 21); Oxford-IIIT Pets, ADE20K, and MS COCO as above			
Flowers102 (Nilsback & Zisserman, 2008)	fine-grained, flowers	102	accuracy
Pascal VOC (Everingham et al., 2010)	classification	20	accuracy
Language (Tables 2 and 27)			
GLUE (Wang et al., 2019): CoLA	linguistic acceptability	2	Matthews correlation
STS-B	semantic similarity	regression	Pearson correlation
MRPC, QQP	paraphrase	2	accuracy
RTE, QNLI	entailment	2	accuracy
MNLI	entailment	3	accuracy
SST-2	sentiment	2	accuracy
Alpaca (Taori et al., 2023)	instruction tuning (training only)	–	–
MMLU (Hendrycks et al., 2021b)	57-subject multiple choice	4 options	5-shot accuracy

Table 7: All benchmarks in this paper: backbone and pretraining, the methods compared, whether baseline numbers are our runs or taken from the literature, and where the results appear. Method abbreviations follow the tables.

Benchmark	Backbone (pretraining)	Compared methods	Baselines	Results
Component comparison, 21 classification datasets	DINOv2 ViT-B	Each component in isolation plus the task head: [CLS] token, position embeddings, patch embedding, LayerNorm, LayerScale, W_Q, W_K, W_V, W_O, MSA block, W_{FC1}, W_{FC2}, MLP block	ours	Table 1a
Column selection, 21 classification datasets	DINOv2 ViT-B	Gradient, Activation, Magnitude, Random	ours	Table 1b
Relocation, 4 classification datasets and VTAB	DINOv2 ViT-B	LoRA, AdaptFormer, FacT at their default location and at W_O	ours	Tables 12, 13
Classification, 21 datasets	DINOv2 ViT-B	Full fine-tuning, Linear, MLP-3, Partial-1, BitFit, Adapter, AdaptFormer, VPT-shallow, VPT-deep, E^2VPT, SSF, LoRA, SPT-Adapter, SPT-LoRA, FacT-TK, FacT-TT, Consolidator, ARC, GPS, RLRR	ours	Table 1c; Tables 8–11
Pretraining strategy, 4 datasets	ViT-B: supervised ImageNet-21K, OpenCLIP	Full fine-tuning, Linear, MLP-3, Partial-1, BitFit, Adapter, AdaptFormer, VPT-shallow, VPT-deep, E^2VPT, SSF, LoRA, SPT-Adapter, SPT-LoRA, FacT-TK, FacT-TT, ARC, RLRR	ours	Table 5; Tables 14, 15
Model capacity, 4 datasets	DINOv2 ViT-S, ViT-B, ViT-L, ViT-g	as for pretraining strategy	ours	Table 5; Tables 16–19
Training cost	DINOv2 ViT-B, batch size 64	as for classification	ours	Table 5; Table 20; Figures 2, 3
Low-data, VTAB-1k	ImageNet-21K ViT-B/16 and Swin-B	Full fine-tuning, Linear, MLP-3, Partial-1, BitFit, Adapter, AdaptFormer, VPT-shallow, VPT-deep, E^2VPT, SSF, LoRA, SPT-Adapter, SPT-LoRA, NOAH, Consolidator, FacT-TK, FacT-TT, ARC, RLRR, GPS, DA-VPT+, MIST, CaRA, DyT, HTA, SNELL, PYRA, FreqFit	reported	Table 3
Distribution shift	ImageNet-21K ViT-B, adapted on ImageNet-1K	Full fine-tuning, Linear, BitFit, Adapter, VPT-shallow, VPT-deep, SSF, GPS	ours	Table 4a
Semantic segmentation, ADE20K	ImageNet-21K ViT-L, SETR-PUP	Full fine-tuning, BitFit, Adapter, AdaptFormer, VPT-deep, SSF, LoRA, SPT-Adapter, SPT-LoRA	ours	Table 4b
Detection and instance segmentation, MS COCO	ImageNet-1K Swin-T, Mask R-CNN	as for semantic segmentation	ours	Table 4b
MONA protocol	ImageNet-21K Swin-L (Flowers, Pets, VOC, ADE20K), Swin-B (MS COCO)	Full fine-tuning, Fixed, NormTuning, Partial-1, BitFit, Adapter, LoRA, AdaptFormer, LoRand, MONA	reported	Table 21
GLUE, component comparison	RoBERTa-base	W_Q, W_K, W_V, W_O, each in all blocks	ours	Table 2 (top); Tables 22, 23
GLUE, method comparison	RoBERTa-base	Full fine-tuning (ours); BitFit, Adapter, LoRA, GaLore	mixed	Table 2 (bottom); Tables 25, 26
Instruction tuning, MMLU	Llama 3.2 (1B), Llama 3.1 (8B)	Full fine-tuning, LoRA, GaLore	ours	Table 27

(Hao et al., 2023), ARC (Dong et al., 2023), GPS (Zhang et al., 2024), RLRR (Dong et al., 2024b), DA-VPT (Ren et al., 2025), CaRA (Veeramacheneni et al., 2025), and MIST (Tang et al., 2025); on VTAB-1k additionally DyT (Zhao et al., 2024b), HTA (Dong et al., 2024a), SNELL (Shen et al., 2024), PYRA (Xiong et al., 2024), and FreqFit (Ly & Nguyen, 2024); under the MONA protocol MONA (Yin et al., 2025) and LoRand (Yin et al., 2023) together with the frozen-backbone (Fixed), NormTuning, and Partial-1 baselines of that protocol; and on GLUE and MMLU GaLore (Zhao et al., 2024a).

Provenance. The VTAB-1k and MONA-protocol tables report the numbers published for each baseline under the VPT protocol (Jia et al., 2022) and the MONA protocol (Yin et al., 2025), respectively, as stated in their captions; APLA is our run under the same protocol. The published GLUE baselines are those reported by Hu et al. (2022) for BitFit, Adapter, and LoRA and by Zhao et al. (2024a) for GaLore, and are separated from our runs in Table 2. All other baselines were run by us under the common protocol described below, following the public implementations of Jia et al. (2022); Dong et al. (2023; 2024b); Jie & Deng (2023); Han et al. (2023); Chen et al. (2022); Hu et al. (2022).

B.3 IMAGE CLASSIFICATION

Training recipe. Unless a table states otherwise, classification models are trained with AdamW (Loshchilov & Hutter, 2019) (weight decay 10^{-5}), batch size 64, and mixed precision for 100 epochs, with gradient clipping at norm 1.0 and a cosine learning-rate schedule (final learning rate 10^{-6}) after a 500-iteration linear warm-up. Training images are resized to 256×256 and randomly cropped to 224×224 (area scale 0.8–1.2) with horizontal flips ($p = 0.5$) and colour jitter (brightness and contrast 0.2, saturation and hue 0.1, $p = 0.8$); validation and test images are resized to 256 and centre-cropped to 224. The released ImageNet-1K configuration (DINOV2 ViT-B) adds TrivialAugment, random erasing ($p = 0.25$), mixup ($\alpha = 0.8$) and CutMix ($\alpha = 1$) applied with probability 0.4, and label smoothing 0.1. All models were implemented in PyTorch (Paszke et al., 2019) and trained on NVIDIA A100 GPUs. DINOv2 backbones keep their LayerScale parameters, which is why LayerScale appears as a component in Table 1a.

Hyperparameter selection. Hyperparameters are selected by grid search on the validation split of each dataset; the validation metric is computed every half epoch and the best checkpoint is evaluated once on the test split. Full fine-tuning follows the same procedure: its learning rate is selected on validation data and its best validation checkpoint is tested. For APLA the grid covers the learning rate and the column count r: for ViT-S, $r \in \{8, 16, 128, 256, 384\}$; for ViT-B, $r \in \{8, 16, 128, 512, 768\}$; for ViT-L, $r \in \{8, 16, 128, 512, 1024\}$; and for ViT-g, $r \in \{8, 16, 128, 1024, 1536\}$. The size of this grid is comparable to the searches in Jie & Deng (2023); Chen et al. (2022); Hu et al. (2022); Jia et al. (2022). Paired comparisons in Section 4 count per-dataset wins across the 21 classification datasets of Table 1, and the p-values are two-sided Wilcoxon signed-rank tests on the per-dataset differences, with each dataset as one paired observation and ties excluded. This is the procedure recommended by Demšar (2006) for comparing methods over multiple datasets; it treats run-to-run variation within a dataset as part of the paired difference, so under the null hypothesis of equivalent methods such variation would produce wins and losses symmetrically rather than the one-sided counts reported.

Component comparison and column selection. In Table 1a each component is tuned in full in every block, together with the task head, with all other parameters frozen; for the four attention projections this corresponds to $r = 768$. APLA selects the columns J of W_O once per block at initialisation by drawing a random permutation of the d output indices and keeping the first r; the indices are stored with the model and remain fixed for the whole run. The bias entries of the selected columns are trained as well, so a block has $r(d+1)$ trainable parameters and APLA trains $12\,r(d+1)$ existing backbone parameters on ViT-B, in addition to the task head, for example 0.07M at $r = 8$ and 1.2M at $r = 128$, against 86M for full fine-tuning; every W_Q, W_K, W_V, or W_O is 0.59M per block (7.1M over 12 blocks) and every W_{FC1} or W_{FC2} is 2.4M per block (28M over 12 blocks). The informed strategies in Table 1b (Gradient, Activation, Magnitude) rank the columns of W_O by a per-column score and keep the top r; the selected columns are then trained exactly as in APLA.

Relocation. In Tables 12 and 13, LoRA, AdaptFormer, and FacT are applied either at their default location or at W_O only, with all other settings unchanged; each method uses its own most beneficial rank or bottleneck width, selected on validation data, at both locations.

B.4 SEMANTIC SEGMENTATION

We follow Jia et al. (2022); He et al. (2023) and use the SETR-PUP framework (Zheng et al., 2021) with a ViT-L (Dosovitskiy et al., 2021) backbone pretrained on IMAGENET-21K (Deng et al., 2009), implemented in *mmsegmentation* (MMSegmentation Contributors, 2020). The released configuration is the library default: 512×512 crops, the 160k-iteration schedule with an effective batch size of 16, AdamW with learning rate 10^{-3}, no weight decay, and a $10\times$ learning-rate multiplier on the decoder head; single-scale results use sliding-window inference with 512×512 windows. APLA trains the full output projection (weights and biases) of every attention block in the backbone, that is, all columns of W_O, and freezes the rest of the backbone; the decoder is trained as usual. We report single- and multi-scale mIoU following Jia et al. (2022); He et al. (2023).

B.5 Object Detection and Instance Segmentation

We follow Lian et al. (2022); Liu et al. (2021) and use Mask R-CNN (He et al., 2017) with a Swin-T (Liu et al., 2021) backbone pretrained on ImageNet-1K (Deng et al., 2009), implemented in *mmdetection* (Chen et al., 2019). The released configuration is the library's Swin-T $1\times$ schedule: multi-scale training with the shorter side in 480–800 pixels, AdamW with learning rate 10^{-4} and weight decay 0.05, 12 epochs with learning-rate steps after epochs 8 and 11, and an effective batch size of 16. As for segmentation, APLA trains the full output projection of every attention block and freezes the rest of the backbone, while the neck and heads are trained as usual. We report box and mask AP averaged over IoU thresholds, and at IoU 0.5 and 0.75, following Lin et al. (2014).

MONA protocol. Table 21 follows the evaluation protocol of Yin et al. (2025): ImageNet-21K-pretrained Swin-L for classification (Flowers102, Oxford-IIIT Pets, Pascal VOC) and for ADE20K with UperNet, and Swin-B for MS COCO with Cascade Mask R-CNN, with the training recipe of that paper. Baseline numbers are those reported by Yin et al. (2025); APLA was run under the same recipe.

B.6 Language Tasks

GLUE. We adapt RoBERTa-base (Liu et al., 2019) on the eight GLUE tasks (Wang et al., 2019) listed in Table 6. For the component comparison a single projection (W_Q, W_K, W_V, or W_O) is tuned in all blocks together with the classification head; the learning rate and the column count $r \in \{512, 768\}$ are selected per task on the development set, which, as in the published GLUE baselines we compare against, serves both for selection and for reporting. Table 24 gives the learning rates searched for each task, and the selected configurations are listed in Table 23. Our full fine-tuning baseline uses the same selection procedure. Metrics are the overall (matched and mismatched) accuracy for MNLI, Matthews correlation for CoLA, Pearson correlation for STS-B, and accuracy otherwise.

Instruction tuning and MMLU. We fine-tune the Llama 3.2 (1B) and Llama 3.1 (8B) base checkpoints (Grattafiori et al., 2024) on the Alpaca instruction data (Taori et al., 2023) using full fine-tuning, LoRA, GaLore, or APLA, and evaluated by 5-shot accuracy on MMLU (Hendrycks et al., 2021b). The exact Hugging Face model identifiers are `meta-llama/Llama-3.2-1B` and `meta-llama/Llama-3.1-8B`, respectively. For APLA, the selected configuration is $r = 2048$ with learning rate 3×10^{-5} for Llama 3.2 (1B), and $r = 128$ with learning rate 5×10^{-5} for Llama 3.1 (8B).

MMLU is evaluated using the standard 5-shot protocol. For each of the 57 subjects, the first five examples from the official development split are used as demonstrations, formatted as multiple-choice questions with options A–D and their answers, followed by the test question without its answer. Predictions are obtained by selecting the option with the highest next-token logit among A, B, C, and D. If the prompt exceeds 2048 tokens, demonstrations are removed until it fits within the context limit.

B.7 Cost Measurement

Training and inference costs (Table 5, Table 20, Figures 2 and 3) are measured for DINOv2 ViT-B with a batch size of 64 on A100 GPUs: peak GPU memory and time per mini-batch of 64 images (reported per image in the figures), for training (forward and backward pass with update) and for inference. Trainable-parameter counts in Figure 3 are averaged over the classification datasets, since the selected r varies by dataset.

C Additional Experimental Results

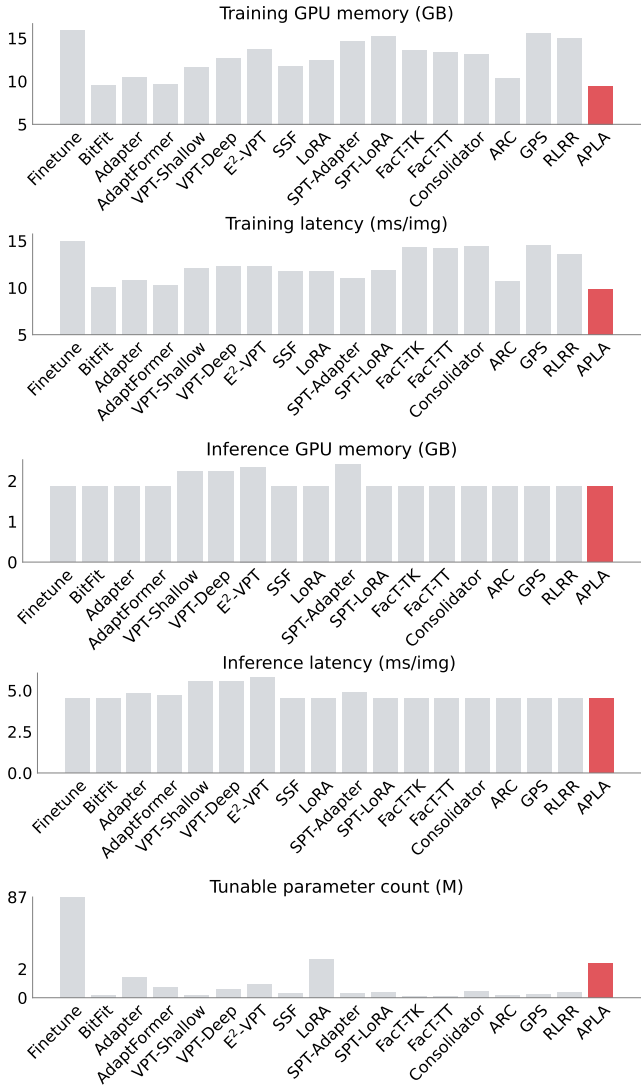

Figure 3: *Computational costs.* We report memory footprint and latency of various adaptation methods during training (top) and inference (middle) for ViT-B with a batch size of 64. Additionally, we provide the number of tunable parameter count for each method (bottom), averaged across all the datasets.

Table 8: Complete classification results: finegrained. All compared methods are retained; bold marks the highest reported score in each column.

Method	CUB	NABirds	Birdsnap	Dogs	Cars	Aircraft
Full FT	88.9	85.2	78.7	86.0	94.4	**87.5**
Linear	89.1	86.6	79.4	87.6	88.4	76.5
MLP head	89.1	86.4	78.9	87.8	88.3	77.7
Partial	88.8	86.5	78.6	87.4	88.1	76.6
BitFit	89.4	87.9	80.7	87.8	92.5	83.2
Adapter	89.6	**88.4**	80.0	87.8	93.5	86.1
AdaptFormer	**89.7**	**88.4**	80.5	88.1	93.1	85.4
VPT-shallow	88.8	86.7	79.1	87.4	90.6	73.2
VPT-deep	89.1	87.3	79.9	87.2	91.5	81.7
E^2VPT	88.3	86.6	79.7	87.4	91.2	81.0
SSF	89.4	88.1	80.5	87.7	92.7	83.7
LoRA	88.7	87.5	79.3	86.3	93.4	86.4
SPT-Adapter	89.4	88.1	80.6	87.7	93.1	86.2
SPT-LoRA	89.2	87.9	80.6	87.5	92.8	86.3
FacT-TK	88.8	87.8	80.5	87.5	93.0	85.4
FacT-TT	88.8	87.6	79.7	87.1	92.9	84.3
Consolidator	**89.7**	87.4	81.5	87.1	93.0	83.4
ARC	89.4	88.2	80.7	88.1	92.6	84.1
GPS	89.1	86.4	80.7	86.6	**94.7**	85.5
RLRR	88.9	87.9	80.8	87.6	92.4	83.7
APLA	89.6	88.0	**81.9**	**88.5**	94.0	86.7

Table 9: Complete classification results: general. All compared methods are retained; bold marks the highest reported score in each column.

Method	Cal-256	Cal-101	CIFAR-100	CIFAR-10	Pet	DTD
Full FT	93.9	97.3	92.4	98.7	94.7	81.9
Linear	94.9	97.0	88.9	98.0	95.8	81.1
MLP head	94.4	97.5	89.2	98.4	96.0	80.9
Partial	94.9	96.9	89.0	98.0	96.0	80.6
BitFit	95.2	97.6	93.1	**99.3**	95.7	82.2
Adapter	95.0	97.6	93.5	**99.3**	95.9	81.8
AdaptFormer	95.6	**98.1**	93.2	**99.3**	95.9	82.8
VPT-shallow	95.1	97.2	92.4	99.1	96.0	80.4
VPT-deep	95.3	97.8	92.7	99.1	95.7	80.5
E^2VPT	94.5	96.9	92.7	99.2	95.4	79.6
SSF	95.3	97.8	93.2	99.2	95.7	82.1
LoRA	94.3	97.1	93.0	99.0	94.0	80.4
SPT-Adapter	**95.8**	97.5	93.1	99.2	95.8	82.7
SPT-LoRA	**95.8**	97.7	92.6	99.2	95.7	82.3
FacT-TK	95.3	97.6	92.8	99.2	95.5	81.5
FacT-TT	94.9	97.5	92.4	99.2	95.5	81.7
Consolidator	94.5	97.3	92.7	99.0	95.8	81.8
ARC	**95.8**	97.8	92.9	99.2	96.0	82.8
GPS	94.8	97.6	**94.0**	**99.3**	94.4	77.9
RLRR	95.2	97.4	93.1	99.2	95.8	82.0
APLA	95.7	98.0	93.4	**99.3**	**96.1**	**83.0**

Table 10: Complete classification results: scene and satellite. All compared methods are retained; bold marks the highest reported score in each column.

Method	MIT-Indoor	SUN397	AID	RSSCN7
Full FT	87.8	75.6	95.4	73.3
Linear	88.9	76.4	91.2	77.1
MLP head	88.6	76.2	91.6	76.4
Partial	88.5	76.3	90.9	77.7
BitFit	89.3	77.7	95.2	83.5
Adapter	89.5	77.9	95.0	84.2
AdaptFormer	89.9	78.1	95.4	85.3
VPT-shallow	89.4	76.6	91.6	70.4
VPT-deep	90.0	77.0	94.4	78.0
E^2VPT	88.7	76.3	93.7	72.7
SSF	88.9	77.4	95.3	82.6
LoRA	88.5	76.4	95.4	81.4
SPT-Adapter	89.5	78.1	95.6	84.7
SPT-LoRA	89.9	77.7	95.4	84.2
FacT-TK	88.9	77.4	95.4	79.9
FacT-TT	89.4	77.1	94.5	80.6
Consolidator	89.5	77.0	94.6	78.0
ARC	89.8	78.2	95.6	**86.2**
GPS	88.4	76.9	94.9	61.6
RLRR	89.5	77.6	95.0	81.2
APLA	**90.4**	**78.3**	**96.0**	**86.2**

Table 11: Complete classification results: medical. All compared methods are retained; bold marks the highest reported score in each column.

Method	ISIC2019	APTOS2019	DDSM	Colorectal	Pneumonia
Full FT	87.7	90.8	95.5	97.8	99.4
Linear	55.3	90.4	89.4	94.0	97.9
MLP head	71.9	90.7	93.4	95.8	97.9
Partial	56.1	90.6	89.5	94.2	97.6
BitFit	79.0	90.1	96.4	97.2	99.1
Adapter	84.3	89.6	96.1	98.0	98.5
AdaptFormer	85.6	90.8	**97.3**	97.4	98.6
VPT-shallow	76.5	89.3	96.2	96.4	98.7
VPT-deep	79.6	91.0	96.2	97.6	98.8
E^2VPT	80.9	90.6	96.2	96.8	98.6
SSF	80.7	90.5	96.4	97.2	99.0
LoRA	86.5	91.1	95.1	97.4	98.9
SPT-Adapter	82.1	90.4	96.1	97.2	99.0
SPT-LoRA	82.2	89.3	96.2	97.4	99.1
FacT-TK	85.1	91.5	96.3	97.2	97.9
FacT-TT	81.5	90.9	97.0	97.4	98.6
Consolidator	81.9	91.1	96.9	96.2	99.4
ARC	82.5	90.7	**97.3**	97.0	98.8
GPS	87.7	90.8	96.7	97.4	99.3
RLRR	81.7	90.4	96.3	96.6	98.6
APLA	**88.2**	**92.1**	97.2	**98.4**	**99.5**

Table 12: Applying LoRA, AdaptFormer, and FacT at their default locations or at W_O, using DINOv2 ViT-B. Underlining marks the better result within each method pair; ties are unmarked. Bold marks the best result in each column. Mean is the average of the four tasks; the corresponding VTAB categories appear in Table 13.

Method / location	Birds	Cars	AID	ISIC	Mean
LoRA / default	87.5	93.4	95.4	86.5	90.7
LoRA / W_O	87.7	93.5	95.1	87.9	91.1
AdaptFormer / default	**88.4**	93.1	95.4	85.6	90.6
AdaptFormer / W_O	**88.4**	93.6	95.3	87.1	91.1
FacT / default	87.8	93.0	95.4	85.1	90.3
FacT / W_O	88.0	93.8	95.3	86.5	90.9
APLA	88.0	**94.0**	**96.0**	**88.2**	**91.6**

Table 13: VTAB results when applying competing methods at their default locations or at W_O. Underlining marks the better result within each method pair; ties are unmarked. Bold marks the best result in each column. The overall average is the mean of the three group averages, as in Table 3.

Method / location	Natural	Specialized	Structured	Avg.
LoRA / default	83.4	86.5	63.1	77.7
LoRA / W_O	83.7	87.6	62.6	78.0
AdaptFormer / default	84.0	87.2	59.8	77.0
AdaptFormer / W_O	84.2	87.2	61.6	77.7
FacT / default	84.7	87.4	**64.5**	78.9
FacT / W_O	84.4	87.2	63.8	78.5
APLA	**85.0**	**88.2**	63.9	**79.0**

Table 14: **Supervised ImageNet-21K pretraining.** ViT-B pretrained with supervised ImageNet-21K (Deng et al., 2009), per dataset and mean over the four datasets; the means are summarized in Table 5. **Bold**: best; underline: second best per column. Baselines as in Table 1.

Method	Birds	Cars	AID	ISIC	Mean
Full fine-tuning	82.7	84.5	91.7	84.0	85.7
LINEAR	75.9	51.3	81.0	51.2	64.9
MLP	77.3	54.9	81.2	61.7	68.8
PARTIAL	77.8	66.2	81.1	46.6	67.9
BITFIT	84.2	79.4	90.5	73.0	81.8
ADAPTER	84.3	68.6	90.0	80.6	80.9
ADAPTFORMER	78.8	83.1	90.1	81.2	83.3
VPT-SHALLOW	78.8	68.7	85.9	65.0	74.6
VPT-DEEP	84.2	83.6	89.0	74.8	82.9
E^2VPT	84.6	82.8	88.4	78.6	83.6
SSF	**85.7**	89.2	90.9	78.4	86.1
LORA	85.6	83.2	91.0	83.5	85.8
SPT-ADAPTER	83.3	86.2	90.8	75.7	84.0
SPT-LORA	83.4	87.3	90.0	76.2	84.2
FACT-TK	80.3	84.0	91.1	79.3	83.7
FACT-TT	79.2	82.4	90.9	77.7	82.6
ARC	**85.7**	89.5	90.8	79.2	86.3
RLRR	85.3	90.4	91.1	77.5	86.1
APLA (Ours)	85.2	**90.5**	**94.3**	**84.9**	**88.7**

Table 15: **OpenCLIP pretraining.** ViT-B pretrained with OpenCLIP (Ilharco et al., 2021), per dataset and mean over the four datasets; the means are summarized in Table 5. **Bold**: best; underline: second best per column. Baselines as in Table 1.

Method	Birds	Cars	AID	ISIC	Mean
Full fine-tuning	79.0	94.7	95.6	84.9	88.6
LINEAR	73.7	94.5	95.0	54.2	79.4
MLP	73.6	93.8	95.0	68.7	82.8
PARTIAL	73.8	94.4	95.2	54.9	79.6
BITFIT	79.2	95.0	95.8	72.7	85.7
ADAPTER	79.2	95.0	95.2	83.9	88.3
ADAPTFORMER	**80.0**	95.0	95.5	82.4	88.2
VPT-SHALLOW	73.8	94.7	95.1	54.6	79.6
VPT-DEEP	73.6	94.5	95.1	54.6	79.5
E^2VPT	77.7	95.1	**95.9**	73.6	85.6
SSF	79.9	94.9	95.7	76.1	86.7
LORA	79.0	94.1	95.0	83.2	87.8
SPT-ADAPTER	76.0	94.8	95.4	76.4	85.7
SPT-LORA	75.9	94.9	95.1	77.4	85.8
FACT-TK	79.2	94.9	95.0	80.7	87.5
FACT-TT	78.5	94.9	95.4	77.2	86.5
ARC	79.5	95.1	95.1	79.0	87.2
RLRR	**80.0**	94.9	95.8	79.5	87.6
APLA (Ours)	79.1	**95.2**	**95.9**	**85.5**	**88.9**

C.1 DETAILED RESULTS FOR DIFFERENT PRETRAINING STRATEGIES AND MODEL SCALES

To examine if APLA scales well with model size, we utilize ViT models of varying sizes (ViT-S, ViT-B, ViT-L, and ViT-g), pre-trained with DINOV2 (Oquab et al., 2023). Table 5 in the main text shows the average results of different adaptation methods across model scales, while Table 16 provides detailed per-dataset results. APLA's advantage persists as model capacity increases. We further present a performance-efficiency trade-off comparison in terms of GPU memory consumption

Table 16: Detailed classification results for DINOv2 ViT-S. The corresponding ViT-B, ViT-L, and ViT-g panels appear in Tables 17–19. Bold marks the highest reported score in each column.

Method	Birds	Cars	AID	ISIC	Mean
Full FT	77.5	**91.9**	91.7	83.8	86.2
Linear	81.3	83.1	88.5	51.5	76.1
MLP head	80.6	83.1	88.8	71.1	80.9
Partial	81.1	83.1	88.3	50.6	75.8
BitFit	83.1	89.7	93.1	73.2	84.8
Adapter	83.2	91.3	93.3	83.8	87.9
AdaptFormer	**83.6**	90.6	92.8	81.9	87.2
VPT-shallow	81.7	86.3	91.3	72.0	82.8
VPT-deep	80.1	86.8	92.5	76.7	84.0
E^2VPT	80.0	87.6	91.9	77.3	84.2
SSF	81.7	89.5	92.7	74.9	84.7
LoRA	80.8	91.0	93.3	83.2	87.1
SPT-Adapter	83.0	91.0	93.3	81.4	87.2
SPT-LoRA	82.8	91.1	93.2	83.2	87.6
FacT-TK	82.5	90.6	92.9	81.7	86.9
FacT-TT	82.3	89.7	91.6	78.0	85.4
ARC	82.9	89.4	93.2	78.5	86.0
RLRR	82.4	89.5	93.3	72.2	84.4
APLA	82.4	91.4	**93.5**	**84.5**	**88.0**

Table 17: Scaling: ViT-B. Complete reported four-task comparison.

Method	Birds	Cars	AID	ISIC	Mean
Full FT	85.2	**94.4**	95.4	87.7	90.7
Linear	86.6	88.4	91.2	55.3	80.4
MLP head	86.4	88.3	91.6	71.9	84.6
Partial	86.5	88.1	90.9	56.1	80.4
BitFit	87.9	92.5	95.2	79.0	88.7
Adapter	**88.4**	93.5	95.0	84.3	90.3
AdaptFormer	**88.4**	93.1	95.4	85.6	90.6
VPT-shallow	86.7	90.6	91.6	76.5	86.4
VPT-deep	87.3	91.5	94.4	79.6	88.2
E^2VPT	86.6	91.2	93.7	80.9	88.1
SSF	88.1	92.7	95.3	80.7	89.2
LoRA	87.9	93.4	95.4	86.5	90.8
SPT-Adapter	88.1	93.1	95.6	82.1	89.7
SPT-LoRA	87.9	92.8	95.4	82.2	89.6
FacT-TK	87.8	93.0	95.4	85.1	90.3
FacT-TT	87.6	92.9	94.5	81.5	89.1
ARC	88.2	92.6	95.6	82.5	89.7
RLRR	87.9	92.4	95.0	81.7	89.3
APLA	88.0	94.0	**96.0**	**88.2**	**91.6**

and latency during training across different model sizes in Figure 2. As model size increases, APLA keeps the highest four-task mean while its training memory and latency remain among the lowest of the methods that adapt the backbone.

C.2 Computational Costs of Adaptation Methods

We analyze the computational costs of adaptation methods by measuring GPU memory footprint and latency during training and inference. Results are shown in Figure 3. During training, APLA uses the

Table 18: Scaling: ViT-L. Complete reported four-task comparison.

Method	Birds	Cars	AID	ISIC	Mean
Full FT	88.2	94.9	95.9	**89.2**	92.1
Linear	89.1	89.8	93.4	62.1	83.6
MLP head	88.9	89.9	92.9	73.0	86.2
Partial	89.1	89.8	93.3	62.0	83.6
BitFit	90.4	93.8	95.8	80.1	90.0
Adapter	90.4	94.8	95.8	86.4	91.9
AdaptFormer	**90.8**	94.2	95.8	86.0	91.7
VPT-shallow	89.0	91.6	93.0	79.7	88.3
VPT-deep	89.1	93.4	95.7	82.6	90.2
E^2VPT	89.5	93.8	95.8	84.3	90.9
SSF	90.6	94.0	95.9	83.5	91.0
LoRA	89.9	94.8	95.9	87.9	92.1
SPT-Adapter	90.6	93.4	95.6	80.5	90.0
SPT-LoRA	89.6	93.8	95.5	82.9	90.5
FacT-TK	90.6	94.5	96.3	84.4	91.5
FacT-TT	90.5	94.6	96.3	83.7	91.3
ARC	90.6	94.3	95.5	83.9	91.1
RLRR	90.6	94.0	95.7	86.7	91.8
APLA	90.6	**95.1**	**96.5**	**89.2**	**92.9**

Table 19: Scaling: ViT-g. Complete reported four-task comparison.

Method	Birds	Cars	AID	ISIC	Mean
Full FT	90.0	95.2	96.5	88.8	92.6
Linear	90.2	91.0	93.6	67.3	85.5
MLP head	89.9	91.0	93.1	77.1	87.8
Partial	90.3	91.0	93.6	65.7	85.2
BitFit	90.8	94.5	95.9	85.8	91.8
Adapter	91.1	95.1	96.3	86.8	92.3
AdaptFormer	91.5	94.9	96.0	88.9	92.8
VPT-shallow	89.8	92.1	95.1	78.7	88.9
VPT-deep	91.1	94.5	96.2	85.9	91.9
E^2VPT	91.2	94.5	95.9	85.3	91.7
SSF	91.0	94.5	95.9	86.3	91.9
LoRA	89.8	94.8	96.7	88.9	92.6
SPT-Adapter	90.7	93.2	95.0	79.0	89.5
SPT-LoRA	90.3	93.5	95.2	78.5	89.4
FacT-TK	91.6	95.1	96.1	85.3	92.0
FacT-TT	91.4	94.7	96.0	84.1	91.6
ARC	91.5	94.7	95.7	85.5	91.9
RLRR	91.6	94.6	95.7	87.2	92.3
APLA	**91.7**	**95.4**	**96.8**	**89.5**	**93.4**

least GPU memory and has the lowest latency among the methods that adapt the backbone (head-only methods such as linear probing are cheaper), and it adds no cost at inference. While APLA appears to tune more parameters than other adaptation methods, one should note that parameter count alone does not necessarily reflect the true computational costs. This inconsistency has been previously emphasized by other studies (Dehghani et al., 2022; Cai et al., 2020).

Table 20: Training and inference cost measurements for ViT-B with batch size 64. Memory is peak GPU memory in GB; time is milliseconds per mini-batch of 64 images (divide by 64 for the per-image latency shown in Figures 2 and 3).

Method	Train mem.	Train time	Infer mem.	Infer time
Full FT	15.9	959	1.86	287
Linear	1.87	285	1.86	287
MLP head	1.87	285	1.86	288
BitFit	9.55	643	1.86	287
Adapter	10.5	689	1.87	310
AdaptFormer	9.64	656	1.87	300
VPT-shallow	11.7	771	2.23	353
VPT-deep	12.7	789	2.23	353
E^2VPT	13.7	789	2.34	369
SSF	11.8	756	1.86	287
LoRA	12.4	753	1.86	287
SPT-Adapter	14.7	705	2.40	313
SPT-LoRA	15.3	763	1.86	287
FacT-TK	13.6	915	1.86	287
FacT-TT	13.4	910	1.86	287
Consolidator	13.2	922	1.86	287
ARC	10.4	684	1.86	287
GPS	15.6	930	1.86	287
RLRR	15.0	867	1.86	287
APLA	9.43	628	1.86	287

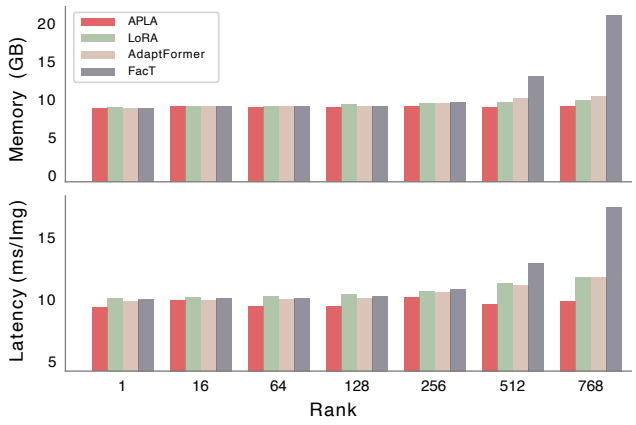

Figure 4: *Computational requirements of different adaptation methods during training for varying r values.* For LoRA and FacT, r denotes the rank, for APLA the number of tuned columns, and for AdaptFormer, the size of the bottleneck dimension.

C.3 Computational Costs of Low-rank Adaptation Methods When Increasing r

In Table 12, we investigated the impact of applying other low-rank adaptation methods on W_O, isolating the impact of the low-rank adaptation strategy. Using the same setup, here we analyze their computational costs with respect to the choice of rank r, considering GPU memory and latency during training. As shown in Figure 4, APLA is the only method that only minimally impacts memory and latency during training, whereas other methods are affected to a larger extent as r grows (*e.g.* FacT). Across the tested values of r, APLA's memory and latency stay nearly constant and at or below those of the compared methods at the same r, because it adds no trainable modules; in this configuration the choice of r is an accuracy decision rather than a cost one (Section 4.5).

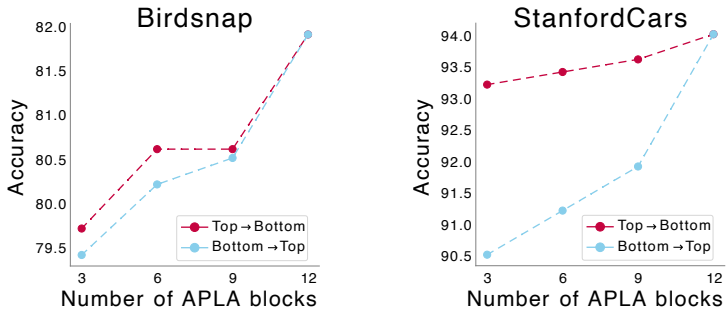

Figure 5: *Classification performance when applying APLA to an increasing number of attention blocks.* "Top-bottom" means applying APLA starting from the last ViT block and moving toward the first layer, while "bottom-top" refers to applying it in the opposite direction.

C.4 APLA on Increasing Number of ViT Blocks

We investigate the impact of applying APLA to increasing number of ViT blocks, when starting from the first layer and moving towards the last layers ("Bottom $\rightarrow$ Top") and the opposite direction ("Top $\rightarrow$ Bottom"), and present the results in Figure 5. Applying APLA to more blocks monotonically improves performance. Applying APLA to later transformer blocks leads to greater performance improvements than applying it to the early ViT layers.

C.5 The Effect of Choice of r

In APLA, the hyperparameter r is used to specify how many columns of the weight matrix W_O are tuned. In Figure 6, we present the selected values of r for general classification tasks (left), datasets with limited data (middle), and all datasets together (right).

C.6 Feature Visualization

As a last sanity check, we evaluate the quality of learned representations when using APLA and compare them with those obtained from other adaptation methods, similarly to (Jia et al., 2022; Lian et al., 2022; Chen et al., 2022). In Figure 7 we use t-SNE (Van der Maaten & Hinton, 2008) to visualize the final representations derived from the [CLS] token of the last ViT block for various datasets from VTAB. Similar to other adaptation methods, APLA generates well-separated clusters for different classes, with data points from the same class positioned closely together.

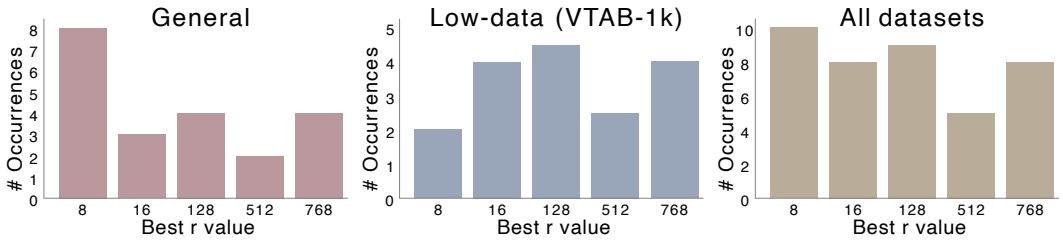

Figure 6: *Selected r values across different datasets.* We report the optimal r values, determined by grid searches, for general classification tasks (left), datasets with limited data (middle), and all datasets (right).

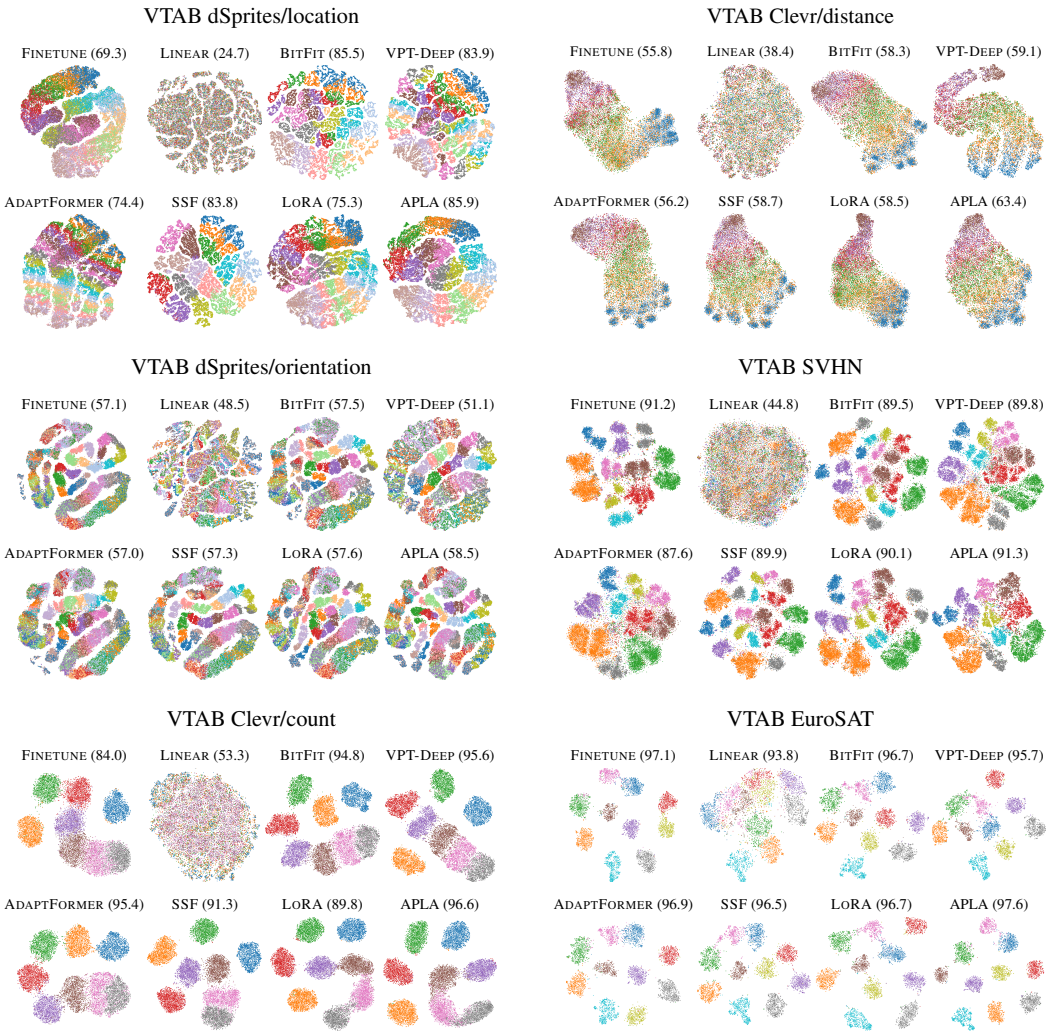

Figure 7: *t-SNE visualizations.* We plot the t-SNE visualizations of the output [CLS] embeddings on VTAB using ViT-B models pre-trained with DINOV2. All models have been adapted for each task. The numbers in parentheses indicate each adaptation method's classification performance for the task.

Table 21: Comparison under the MONA evaluation protocol (Yin et al., 2025): Swin-L (classification, VOC, ADE20K) and Swin-B (COCO) pretrained on ImageNet-21K with MONA's training recipe. Baseline numbers are those reported by Yin et al. (2025) (Fixed, NormTuning and Partial-1 as defined there; BitFit (Ben Zaken et al., 2022), Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), AdaptFormer (Chen et al., 2022), LoRand (Yin et al., 2023)); APLA was run under the same protocol. Flowers, Pets and VOC are top-1 classification accuracy; COCO is Cascade Mask R-CNN box/mask AP; ADE20K is UperNet mIoU. This setup differs from the dense-prediction comparison in Table 4.

Method	Flowers	Pet	VOC	Box AP	Mask AP	ADE20K
Full FT	99.58	94.66	84.13	52.40	45.10	51.18
Fixed	99.30	94.22	85.02	48.00	41.60	46.84
NormTuning	99.53	95.23	85.52	50.10	43.50	47.89
Partial-1	99.66	95.39	84.94	50.60	43.70	47.44
BitFit	99.58	95.34	85.60	50.10	43.60	48.37
Adapter	99.59	95.34	87.04	52.10	45.00	50.78
LoRA	99.54	95.15	85.70	50.40	43.90	50.34
AdaptFormer	99.56	95.26	86.29	51.70	44.60	50.83
LoRAND	99.57	95.35	86.65	51.90	44.70	50.93
MONA	99.68	95.48	86.97	**53.40**	**46.00**	**51.36**
APLA	**99.72**	**95.63**	**87.31**	52.20	44.80	51.02

Table 22: RoBERTa-base attention-component adaptation on eight GLUE tasks (two-decimal values behind Table 2). Scores are reported on the official GLUE development splits; bold marks the best of the four targets per task. Selected learning rates and column counts appear in Table 23.

Target	CoLA	STS-B	MRPC	RTE	SST-2	MNLI	QNLI	QQP
W_Q	57.54	83.35	92.63	71.48	93.35	85.32	91.20	91.01
W_K	58.08	84.24	91.46	72.20	93.69	85.57	91.42	90.98
W_V	63.32	**91.28**	92.98	**81.59**	94.84	87.05	92.95	91.53
W_O	**66.71**	90.92	**93.59**	79.42	**95.18**	**87.14**	**93.23**	**91.62**

Output adaptation leads on six tasks; value adaptation leads on STS-B and RTE. Column counts vary across configurations; this is not an established equal-budget comparison.

Table 23: Selected configurations for the RoBERTa-base component comparison. Each cell gives learning rate / column count. Scores for the corresponding configurations appear in Table 22.

Task	W_Q	W_K	W_V	W_O
CoLA	1e-04 / 768	1e-04 / 768	1e-04 / 768	3e-04 / 768
STS-B	3e-04 / 768	3e-04 / 768	1e-04 / 768	5e-04 / 512
MRPC	3e-04 / 512	5e-04 / 768	7e-05 / 768	3e-04 / 512
RTE	5e-04 / 512	7e-05 / 768	7e-05 / 768	3e-04 / 768
SST-2	1e-04 / 512	5e-05 / 768	3e-05 / 768	1e-04 / 512
MNLI	3e-05 / 768	3e-05 / 768	5e-06 / 768	1e-05 / 512
QNLI	7e-05 / 768	7e-05 / 512	5e-05 / 512	5e-05 / 768
QQP	1e-04 / 768	1e-04 / 768	5e-05 / 768	5e-05 / 768

Table 24: Learning rates searched for the RoBERTa-base GLUE experiments. "All eight" refers to the GLUE tasks listed in Table 6.

Search	Learning rates	Tasks
Initial, small	10^{-6}, 5×10^{-6}	All eight
Initial, medium	10^{-5}, 5×10^{-5}, 10^{-4}, 5×10^{-4}	All eight
Initial, large	10^{-3}, 5×10^{-3}	All eight
Additional, small	3×10^{-6}, 7×10^{-6}	QNLI, QQP
Additional, medium	3×10^{-5}, 7×10^{-5}, 3×10^{-4}, 7×10^{-4}	All eight
Additional, large	3×10^{-3}, 7×10^{-3}, 10^{-2}	All except QNLI

Table 25: RoBERTa-base GLUE: authors' runs of full fine-tuning and APLA (output projection only), two-decimal scores.

Method	CoLA	STS-B	MRPC	RTE
Full FT (ours)	63.58	91.07	93.22	81.23
APLA (output only)	66.71	90.92	93.59	79.42

Method	SST-2	MNLI	QNLI	QQP	Mean
Full FT (ours)	94.95	87.71	92.99	91.96	87.09
APLA (output only)	95.18	87.14	93.23	91.62	87.23

Scores are reported on the official GLUE development splits. Mean is an unweighted average of the eight recorded task scores, not an official aggregate claim.

Table 26: RoBERTa-base GLUE results reported in the LoRA (Hu et al., 2022) and GaLore (Zhao et al., 2024a) papers, obtained under their original training protocols. GaLore is the rank-8 configuration (the one with the highest reported average); the means are recomputed from the displayed task scores so that all methods are averaged the same way.

Method	CoLA	STS-B	MRPC	RTE
LoRA (literature)	63.40	91.50	89.70	86.60
GaLore (literature)	60.06	90.82	92.01	79.78

Method	SST-2	MNLI	QNLI	QQP	Mean
LoRA (literature)	95.10	87.50	93.30	90.80	87.24
GaLore (literature)	94.38	87.17	92.20	91.11	85.94

LoRA uses an MNLI checkpoint to initialize STS-B, MRPC, and RTE; the authors' runs initialize directly from the pretrained model. Our full fine-tuning baseline (Table 25) achieves a higher mean score than the published full fine-tuning references for RoBERTa-base (87.1 versus 86.4 in the RoBERTa paper (Liu et al., 2019), also used as the reference in the LoRA paper, and 86.3 in the GaLore paper).

Table 27: **Decoder LLMs: 5-shot MMLU accuracy (%) after fine-tuning on Alpaca.** Llama 3.2 (1B) and Llama 3.1 (8B) (Grattafiori et al., 2024); hyperparameters were selected on validation data and test scores are reported. Dashes: not run. **Bold**/underline: highest/second-highest score per model.

Method	Llama 3.2 (1B)	Llama 3.1 (8B)
Full fine-tuning	<u>31.3</u>	<u>65.3</u>
LoRA	30.4	65.2
GaLore	**31.5**	–
APLA (Ours)	<u>31.3</u>	**65.4**

No unadapted baseline is reported.