MSLoRA: Multi-Scale Low-Rank Adaptation via Attention Reweighting

Anonymous authors
Paper under double-blind review

Abstract

Parameter-efficient fine-tuning (PEFT) of vision backbones is usually posed as a question of *how many* parameters to train. We argue that for dense prediction the more consequential question is *where* a fixed budget should be spent. We study two ways of spending a small budget on a frozen backbone: an additive low-rank residual over intermediate feature maps, and the same residual modulated by an input-dependent, spatially varying multiplicative gate. At budgets of the same order, the multiplicative form is consistently better for localization-heavy tasks and roughly neutral for global classification, which is the pattern one expects if the gate supplies spatial selectivity that a channel-wise low-rank update cannot express. We instantiate this finding as MSLoRA, an adapter that gates a grouped low-rank projection with a multi-scale depth-wise branch. Because MSLoRA acts on feature maps rather than on architecture-specific weight matrices, the same feature-space formulation applies across convolutional, hierarchical-transformer, hybrid, and plain ViT backbones; a separate adapter is still trained per backbone and task, as for other PEFT methods. On COCO with Cascade Mask R-CNN, MSLoRA reaches 42.9 box AP / 38.4 mask AP on ResNet-50 while training 0.7M parameters (2.7% of the backbone), and is competitive with backbone-specialized PEFT methods on Swin-B at roughly half their trainable-parameter count. We report the full cost profile, including the cases where MSLoRA is slower than full fine-tuning, and we audit the strength of the full fine-tuning baselines we compare against.

1 Introduction

Adapting a pretrained vision backbone (Deng et al., 2009; He et al., 2016) to a downstream task by updating all of its parameters is expensive and, when target data are limited, prone to overfitting. Parameter-efficient fine-tuning (PEFT) addresses this by restricting adaptation to a small trainable set (Hu et al., 2022; Houlsby et al., 2019; He et al., 2022), either by selecting existing parameters such as biases or normalization statistics (Zaken et al., 2022; Giannou et al., 2023) or by inserting lightweight modules around a frozen network (Chen et al., 2022a; 2023a; Liu et al., 2022a).

Most of this literature is organized around a single axis: the size of the trainable set. Methods are compared by how few parameters they train and how close they come to full fine-tuning. We think a second axis matters at least as much for dense prediction, and has received far less attention: given a fixed budget, what *kind* of function should those parameters buy?

Low-rank adaptation answers this implicitly. LoRA (Hu et al., 2022) and its descendants spend the budget on an *additive* low-rank update to a weight matrix. Applied to a vision feature map F, the analogous operation is an additive low-rank residual $\hat{F} = F + P(F)$, where P factors through a narrow channel bottleneck. Such an update is, by construction, a channel recombination applied identically at every spatial location. It can reweight what a feature means; it cannot express *where* that reweighting should apply.

That limitation is invisible for global classification, where a single pooled descriptor is all that is consumed downstream, and it is exactly the wrong limitation for detection and segmentation, where the task is defined by spatially distributed evidence over multiple objects and regions. This suggests a concrete alternative: spend part of the same budget on a term whose job is to decide *where* the low-rank residual is expressed.

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.

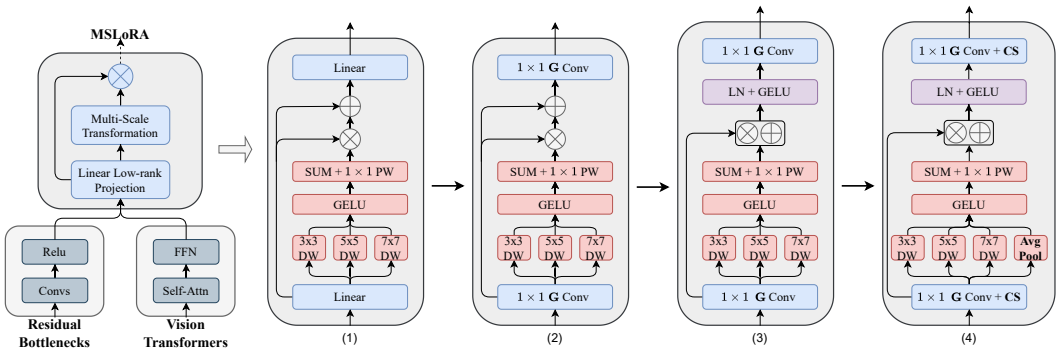


Figure 1: **MSLoRA.** A grouped low-rank projection branch $P(F)$ is gated by a multi-scale depth-wise branch $A(F)$ computed in the same reduced space; the product is added back as a residual to a frozen feature map. Because both branches consume and produce feature maps, the module inserts into convolutional bottlenecks, hierarchical transformer blocks, and, after reshaping patch tokens to their 2D grid, into plain ViT blocks, without touching architecture-specific weights.

Hypothesis. For a small adaptation budget on a frozen vision backbone, an input-dependent multiplicative spatial gate over a low-rank residual is a better allocation than spending the whole budget on the low-rank residual, and the advantage grows with how localization-dependent the downstream task is.

This hypothesis makes a task-dependent prediction rather than a universal one. If spatial selectivity is the relevant difference, the gated update should provide little additional benefit when spatial evidence is ultimately pooled, but a larger benefit when spatially resolved features are consumed directly. A comparable gain in both settings would provide little support for this interpretation.

We test this by comparing the update forms at parameter budgets of the same order: $\hat{F} = F + P(F)$ versus $\hat{F} = F + A(F) \odot P(F)$, where A is a multi-scale depth-wise branch computed in the same low-rank space. The pattern in the data matches the hypothesis. On Food101 classification with Swin-B the two forms are within 0.2 top-1 of one another (92.7 vs. 92.9). On COCO detection and instance segmentation with ResNet-50 the gated form is better by 1.6 box AP and 1.3 mask AP (40.3/36.5 $\rightarrow$ 41.9/37.8), for 0.4M additional parameters. The task-dependent contrast is consistent with spatial selectivity being more valuable when spatially resolved features are consumed downstream.

We instantiate the gated form as **MSLoRA**. The projection branch uses grouped 1×1 convolutions so that its cost does not scale with the full channel width; the gating branch uses parallel depth-wise convolutions at several kernel sizes, so that the gate can be driven by evidence at more than one receptive field. Both branches operate on intermediate feature maps rather than on weight matrices, which is what allows one operator to be used across backbone families (Fig. 1).

Scope. MSLoRA uses a separately trained adapter for each backbone/task pair, as in standard PEFT. The cross-architecture claim concerns the operator definition: because it acts on feature maps rather than selecting architecture-specific weight matrices, the same formulation can be applied across backbone families.

Contributions.

- A budget-allocation finding: at comparable small parameter budgets, multiplicative spatial gating of a low-rank residual provides a substantially larger benefit on dense prediction than on global classification (Sec. 4.2).
- MSLoRA, a feature-space instantiation of that finding, evaluated with the same feature-space formulation across ResNet (He et al., 2016), ResNeXt (Xie et al., 2017), RegNet (Radosavovic et al., 2020), ConvNeXt (Liu et al., 2022c), Swin (Liu et al., 2021; 2022b), and ViT (Dosovitskiy et al., 2020) backbones on classification, detection, instance segmentation, and semantic segmentation (Secs. 4.3–4.4).

- A full cost profile covering trainable parameters, profiled compute, throughput, and peak training memory, together with an analysis of full-fine-tuning baseline strength and the training-regime differences relevant to their interpretation (Secs. 4.5–4.6).

2 Related Work

Parameter-efficient adaptation. PEFT methods restrict adaptation to a small trainable set. Selective methods tune existing parameters such as biases or normalization layers (Zaken et al., 2022; Giannou et al., 2023). Others rescale or shift existing activations (Lian et al., 2022). Additive methods insert trainable modules around a frozen network: prompt tokens (Jia et al., 2022), bottleneck adapters (Houlsby et al., 2019; Chen et al., 2022a;b), convolutional bypasses (Jie & Deng, 2022), and low-rank weight updates (Hu et al., 2022; Si et al., 2024), whose more recent variants refine the factorization itself by decomposing magnitude from direction (Liu et al., 2024), sharing frozen random bases (Kopiczko et al., 2024), or tensorizing the update across layers (Jie & Deng, 2023). Surveys cover the growing range of vision-specific variants (Xin et al., 2024). What these share is that the trainable function is additive in the representation and, in the low-rank case, spatially uniform: the refinements above change how the low-rank factors are parameterized, not the fact that the update is applied identically at every spatial position. Our question is orthogonal to how large that trainable set is: we ask whether the same budget buys more when part of it is spent on an input-dependent gate.

Architecture-specific PEFT. Several strong methods are designed around a particular backbone family. Mona (Yin et al., 2025) adapts Swin transformers by modifying self-attention and MLP blocks and reports backbone experiments on Swin-B and Swin-L. LoRand (Yin et al., 2023) and AdaptFormer (Chen et al., 2022a) likewise target transformer blocks, while ConvAdapter (Chen et al., 2024) targets convolutional layers, and mixture-of-adapter methods specialize to segmentation foundation models (Sahay & Savakis, 2025). A separate line builds adapters that restore the spatial priors plain ViTs lack so that they can drive dense-prediction heads (Chen et al., 2023b), and RepAdapter (Luo et al., 2023) shows that an adapter can be made structurally re-parameterizable, i.e. folded back into the frozen weights at inference. Our design gives up that property, and we return to it in Sec. 4.5. These methods are competitive and, in Mona's case, remain the strongest Swin-B result we are aware of (Table 2). Our aim is not to displace them but to ask whether an operator defined on feature maps, and therefore reusable across families, can get close to specialized designs at a comparable or smaller budget.

Multiplicative gating in vision. Multiplicative modulation of features is not new, and the closest relatives of our gating branch come from backbone design rather than from PEFT. Channel attention rescales channels by a pooled descriptor (Hu et al., 2018); spatial context modules aggregate evidence at multiple receptive fields for dense prediction (Zhao et al., 2017; Chen et al., 2014); and large-kernel and multi-scale convolutional attention build an attention map from decomposed or parallel depth-wise convolutions and multiply it into the features (Guo et al., 2023; 2022). Depth-wise convolutions at several kernel sizes are likewise a standard efficient building block (Sandler et al., 2018; Zhang et al., 2018; Yang et al., 2022). These primitives are standard; the distinction here is their role. In prior work, the gate modulates the full feature map and forms part of the backbone computation. In MSLoRA, the backbone remains frozen and the gate acts only on a rank-D adaptation residual. The additional trainable capacity is therefore confined to the adapter, where it makes the residual input- and position-dependent. Our question is whether allocating part of a small adaptation budget to this spatial modulation changes the accuracy–parameter trade-off relative to an additive low-rank residual. Sec. 4.2 tests this directly.

3 Method

3.1 Gated low-rank feature adaptation

Let $F \in \mathbb{R}^{C\times H\times W}$ be an intermediate feature map of a frozen backbone. MSLoRA replaces it with

$$\hat{F} = F + A(F) \odot P(F), \tag{1}$$

where P is a low-rank projection branch, A is a multi-scale gating branch, and $\odot$ is element-wise (broadcast-free) multiplication. Setting $A \equiv \mathbf{1}$ recovers the additive low-rank residual $\hat{F} = F + P(F)$;

this is the control we compare against throughout. The backbone is frozen, so the trainable function is exactly the map $F \mapsto A(F) \odot P(F)$.

Low-rank projection branch. P is a symmetric bottleneck of width $D \ll C$:

$$P(F) = \mathrm{GConv}_{\uparrow}\big(\mathrm{GConv}_{\downarrow}(F)\big), \tag{2}$$

with $\mathrm{GConv}_{\downarrow} : C \to D$ and $\mathrm{GConv}_{\uparrow} : D \to C$ grouped 1×1 convolutions with G groups (Xie et al., 2017; Zhang et al., 2018). Grouping matters because a dense 1×1 projection dominates the budget at large C: on Swin-L it accounts for 6.9M of 7.5M adapter parameters at $G{=}1$, against 1.7M of 2.3M at $G{=}4$ (Appendix A).

Multi-scale gating branch. Let $Z = \mathrm{GConv}_{\downarrow}(F) \in \mathbb{R}^{D\times H\times W}$ be the shared low-rank representation. The gate is computed in that same space:

$$A(F) = \mathrm{PWConv}\Big(\textstyle\sum_{k\in\mathcal{K}} \mathrm{GELU}\big(\mathrm{DWConv}_k(Z)\big)\Big), \tag{3}$$

where DWConv_k is a depth-wise convolution with kernel size k and PWConv is a 1×1 convolution mixing channels. Parallel kernels let the gate respond to evidence at several receptive fields; computing them on Z rather than on F is what keeps the branch cheap. After aggregating and mixing the multi-scale responses, we apply LayerNorm followed by GELU before the multiplicative interaction. The post-gate nonlinearity is part of the default MSLoRA configuration used in the final task comparisons. We use $\mathcal{K} = \{3, 5, 7\}$ by default. Because A is a function of the input and varies over (h, w), Eq. 1 is an input-dependent, spatially varying modulation of the residual, which is the property the additive form lacks.

3.2 Why the gate should matter more for localization

The additive residual $P(F)$ is a rank-D channel recombination applied with the same coefficients at every spatial position. Composed with a frozen backbone, it can change which directions in feature space are emphasized, but the change is position-independent: if it amplifies a texture channel, it amplifies it in the background as well as on the object. A global classification head pools over positions, so position-independent reweighting is close to sufficient. A detector or segmentation head consumes the spatial map directly, so the same reweighting propagates its errors into localization.

The gate $A(F)$ removes that constraint at low cost. It does not add representational capacity to the backbone, and cannot, since the backbone is frozen and A only scales an existing rank-D residual. What it decides is how much of that residual is applied, per position and per channel. Under a fixed budget this should be worth more when the downstream loss is spatially resolved, and close to worthless when it is not. Sec. 4.2 reports that this is what happens.

3.3 Cost and exact configuration

For one module with input width C, bottleneck width D, and G groups, the two grouped 1×1 projections contribute $2CD/G$ parameters; the depth-wise kernels contribute $\sum_{k\in\mathcal{K}} k^2 D = 83D$ for $\mathcal{K} = \{3, 5, 7\}$; and the mixing 1×1 contributes D^2. Hence

$$P_{\text{MSLoRA}} = \frac{2CD}{G} + D^2 + 83D. \tag{4}$$

The first term is the only one that grows with backbone width, which is why G controls how the budget divides between the two branches.

Configuration used for all main results. Unless a table states otherwise, MSLoRA uses $G{=}4$, $D{=}128$, and $\mathcal{K} = \{3, 5, 7\}$, together with the post-gate LayerNorm+GELU described above. For detection and segmentation, we additionally use three lightweight refinements: a global-average-pooled context path, a learned scalar gate for each depth-wise path, and channel shuffle after the grouped projection. Classification omits these three dense-prediction refinements. Table 6 separates the contribution of the post-gate nonlinearity from these additional refinements.

Table 1: **Budget allocation on a frozen backbone.** Additive low-rank residual (*linear*, $A \equiv \mathbf{1}$), gate-only (*nonlinear*), and the gated product (*both*). Budgets are of the same order but not exactly matched; see the caveat in Sec. 4.2. Classification is Food101 / Swin-B; detection and segmentation are COCO / ResNet-50 Cascade Mask R-CNN, $1\times$. The gate is worth little for pooled classification and clearly worth its cost for spatially resolved tasks.

	Food101 (Swin-B)			COCO (ResNet-50)		
Form of the update	#p	top-1	top-5	#p	AP^{bbox}	AP^{mask}
additive $F + P(F)$	1.1M	92.7	99.1	0.3M	40.3	36.5
gate only $F + A(F)$	1.7M	92.6	99.1	0.6M	41.6	37.6
gated product $F + A(F) \odot P(F)$	1.8M	**92.9**	**99.2**	0.7M	**41.9**	**37.8**

4 EXPERIMENTS

4.1 SETUP

MSLoRA is optimized with AdamW (weight decay 0.05, β_1=0.9, β_2=0.999). The backbone is frozen, including BatchNorm statistics where present (Ioffe & Szegedy, 2015); DropPath remains active (Huang et al., 2016). For pre-norm architectures we prepend LayerNorm (Ba et al., 2016) to the MSLoRA input.

Classification. Oxford Pets (Parkhi et al., 2012), Flower102 (Nilsback & Zisserman, 2008), and Food101 (Bossard et al., 2014). Training uses `RandomResizedCrop`(224) and horizontal flip; evaluation uses `Resize`(256) + `CenterCrop`(224). Final comparisons train 100 epochs (ablations 50) at batch size 16, learning rate 3×10^{-5}, cosine decay, 20-epoch warmup.

Dense prediction. COCO (Lin et al., 2014) with Cascade Mask R-CNN (Cai & Vasconcelos, 2019), Pascal VOC (Everingham et al., 2015) with RetinaNet (Lin et al., 2017), and ADE20K (Zhou et al., 2017) with UPerNet (Xiao et al., 2018). Learning rate is 2×10^{-4} for CNN backbones and 1×10^{-4} for transformer backbones; all other hyperparameters follow the corresponding full fine-tuning configuration. Swin-B on COCO uses a $3\times$ schedule (36 epochs); all other COCO, VOC, and ADE20K runs use $1\times$ (12 epochs). PEFT baselines use the same pretrained initialization, detector heads, data pipeline, and evaluation protocol.

Reporting precision. All results are from single runs, and we do not fix the random seed. In our COCO experiments, results are stable to one decimal place; we therefore report AP to one decimal and avoid drawing conclusions from differences smaller than 0.1 AP. On the saturated classification datasets, differences below roughly 0.2 top-1 are likewise treated as inconclusive. Several entries below fall within these ranges and are discussed as ties.

4.2 WHERE THE BUDGET SHOULD GO

Table 1 is the paper's central experiment. Holding the backbone, schedule, and optimizer fixed within each setting, we compare three update forms at parameter budgets of the same order.

On Food101 with Swin-B, the three variants span 92.6–92.9 top-1. The additive residual reaches 92.7 with 1.1M parameters, while the gated product reaches 92.9 with 1.8M. Thus, in this pooled classification setting, adding spatial gating changes accuracy only slightly.

The effect is substantially larger on COCO with ResNet-50. The additive residual reaches 40.3 box / 36.5 mask AP, while the gated product reaches 41.9 / 37.8, corresponding to $+1.6$ box and $+1.3$ mask AP for 0.4M additional parameters. The gate-only variant reaches 41.6 / 37.6 with 0.6M parameters, recovering most of this improvement. This pattern is consistent with spatial selectivity contributing more strongly to dense prediction than to pooled classification.

The parameter budgets are not exactly matched: on COCO, the additive residual uses 0.3M parameters and the gated product 0.7M. The comparison therefore does not attribute the entire accuracy gap to update form alone. In addition, the classification and dense-prediction experiments use different backbones and datasets; their contrast should be interpreted at the task-regime level—pooled versus spatially resolved prediction—rather than as an isolated causal effect of readout type.

Table 2: **COCO, Cascade Mask R-CNN.** #p is trainable backbone-adaptation parameters, % the fraction of the backbone. MSLoRA* uses G=1. Parentheses give the difference from full fine-tuning with the same backbone. Swin-B uses a 3× schedule, all others 1×; Sec. 4.6 discusses why that distinction matters. **Bold** marks the best value in each backbone block. PEFT baselines were run for ResNet-50 and Swin-B only.

Backbone	Method	#p	%	AP^{bbox}	AP^{mask}
ResNet-50	fine-tune	23.5M	92%	41.2	35.9
	frozen (no adapter)	0	0%	14.0	12.7
	norm-tune (Giannou et al., 2023)	45K	<1%	29.2	26.0
	bias-tune (Zaken et al., 2022)	23K	<1%	35.8	31.7
	LoRA (Hu et al., 2022)	0.1M	<1%	38.6	35.1
	ConvAdapter (Chen et al., 2024)	5.8M	24%	38.8	35.1
	MSLoRA	0.7M	2.7%	**42.9**(+1.7)	**38.4**(+2.5)
ResNet-101	fine-tune	41.2M	95%	42.9	37.3
	MSLoRA	1.6M	3.7%	**44.0**(+1.1)	**39.2**(+1.9)
ResNeXt-101	fine-tune	41.2M	95%	44.3	38.3
	MSLoRA	1.6M	3.7%	**44.8**(+0.5)	**39.9**(+1.6)
RegNetX-4.0	fine-tune	22.2M	95%	**44.4**	38.6
	MSLoRA	1.7M	7.6%	44.3(-0.1)	**39.7**(+1.1)
ConvNeXt-S	fine-tune	50M	100%	47.9	41.6
	MSLoRA	2.2M	4.4%	**48.5**(+0.6)	**41.9**(+0.3)
Swin-B	fine-tune	88M	100%	52.4	45.1
	LoRA (Hu et al., 2022)	3.0M	3.4%	50.4	43.9
	AdaptFormer (Chen et al., 2022a)	1.6M	1.7%	51.7	44.6
	LoRand (Yin et al., 2023)	4.6M	5.2%	51.9	44.7
	Adapter (Chen et al., 2022b)	3.1M	3.5%	52.1	45.0
	Mona (Yin et al., 2025)	4.1M	4.6%	**53.4**	46.0
	MSLoRA	1.8M	2.0%	52.7(+0.3)	45.9(+0.8)
	MSLoRA*	5.2M	5.9%	53.2(+0.8)	**46.1**(+1.0)

4.3 Dense prediction

COCO. Table 2 reports detection and instance segmentation across six backbones. On the CNN backbones MSLoRA exceeds full fine-tuning on mask AP in every case and on box AP in all but RegNetX-4.0, while training 2.7–7.6% of backbone parameters. Against PEFT baselines on ResNet-50, the margin is large ($+4.1$ box over ConvAdapter at $\sim 8\times$ fewer trainable parameters, $+4.3$ box over LoRA).

On Swin-B, Mona (Yin et al., 2025) obtains the highest box AP in the table (53.4). Default MSLoRA reaches 52.7 / 45.9 with 1.8M trainable parameters, 56% fewer than Mona, while MSLoRA* reaches 53.2 / 46.1 with 5.2M parameters. Mona is designed specifically for Swin transformers, with adaptation modules inserted into the attention and MLP blocks. MSLoRA instead uses the same feature-space formulation across the backbone families evaluated in this paper. Thus, the Swin-B result shows that a backbone-agnostic design can approach the performance of a Swin-specific adapter while retaining a substantially smaller parameter budget in its default setting. PEFT baselines are available for ResNet-50 and Swin-B; for the other four COCO backbones, Table 2 compares MSLoRA with full fine-tuning.

Pascal VOC and ADE20K. Table 3 extends the evaluation to a second detector and to semantic segmentation. MSLoRA exceeds the corresponding full fine-tuning result in all six comparisons. The ADE20K margins for ResNet-101 and Swin-L are small ($+0.15$ and $+0.25$ mIoU, respectively), while the remaining differences are larger.

4.4 Classification

Table 4 reports top-1 accuracy on three fine-grained datasets. Flower102 is highly saturated across methods, so the more informative comparisons come from Oxford Pets and Food101. The same

Table 3: **Pascal VOC (RetinaNet, AP) and ADE20K (UPerNet, mIoU)**, $1\times$ schedule. Parentheses report the difference from full fine-tuning with the same backbone.

Backbone	Method	#p	%	Pascal VOC	ADE20K
ResNet-50	fine-tune	23.5M	92%	77.26	40.70
	MSLoRA	0.7M	2.7%	**77.83**(+0.57)	**41.62**(+0.92)
ResNet-101	fine-tune	41.2M	95%	78.10	42.91
	MSLoRA	1.6M	3.7%	**78.82**(+0.72)	**43.06**(+0.15)
Swin-L	fine-tune	197M	100%	83.40	50.13
	MSLoRA	2.3M	1.4%	**84.49**(+1.09)	**50.38**(+0.25)

Table 4: **Classification, top-1 accuracy.** Avg. is the mean over the three datasets. Flower102 is saturated across methods and therefore provides little discrimination.

Backbone	Method	#p	%	Oxford Pets	Flower102	Food101	Avg.
Swin-B	fine-tune	88M	100%	94.00	99.57	92.76	95.44
	MSLoRA	1.8M	2.0%	**94.93**	**99.72**	**93.37**	**96.00**
Swin-L	fine-tune	197M	100%	94.73	99.69	93.84	96.09
	LoRA	4.5M	2.2%	95.14	99.54	93.19	95.96
	Adapter	4.6M	2.3%	95.33	99.59	93.23	96.05
	LoRand	5.2M	2.6%	95.35	99.57	93.27	96.06
	Mona	5.0M	2.5%	95.47	99.67	93.95	96.36
	MSLoRA	2.3M	1.4%	**95.55**	**99.72**	**94.19**	**96.49**
ViT-B/16	fine-tune	86M	100%	93.68	99.02	90.67	94.46
	MSLoRA	1.2M	1.4%	**94.54**	**99.30**	**91.33**	**95.06**

feature-space formulation also extends to a plain ViT-B/16 (Dosovitskiy et al., 2020): patch tokens are reshaped to their $H\times W$ grid, processed by Eq. 1, and reshaped back, with the class token excluded from the spatial transformation. MSLoRA exceeds full fine-tuning on all three datasets while training 1.4% of the backbone parameters. This result shows that the multi-scale operation is supplied by the adapter itself and does not require a hierarchical backbone. Each backbone/task pair uses a separately trained adapter; the cross-architecture result concerns the shared formulation rather than transfer of a trained adapter.

4.5 What MSLoRA costs

Parameter count alone does not capture practical efficiency, so Table 5 also reports adapted-path compute, training throughput, and peak memory.

MSLoRA reaches 1.2 iter/s, compared with 1.6 for full fine-tuning, 1.5 for LoRA, and 1.3 for ConvAdapter. Its peak training memory is 3999 MB, below full fine-tuning (5512 MB) and ConvAdapter (4434 MB) but above LoRA (3565 MB). The additional cost comes from the parallel spatial branches and from grouped and depth-wise operators, which are less hardware-efficient than dense convolutions in our implementation.

Relative to full fine-tuning, MSLoRA trains 0.7M rather than 23.5M backbone-adaptation parameters and reduces peak memory by 27.5%, while reaching 42.9 / 38.4 box/mask AP. Among the PEFT methods in Table 5, it is the only one that exceeds the corresponding full fine-tuning baseline. Relative to ConvAdapter, it uses approximately $8\times$ fewer trainable parameters and 435 MB less peak memory while improving box/mask AP by $+4.1/+3.3$. Because the gate is input-dependent, MSLoRA cannot be algebraically folded into the frozen weights at inference, unlike a purely linear low-rank update or a structurally re-parameterizable adapter (Luo et al., 2023).

4.6 Validity of the full fine-tuning comparison

Because MSLoRA exceeds full fine-tuning in several settings, we examine the strength of the reference baselines and the protocol differences relevant to this comparison.

Table 5: **Cost profile**, ResNet-50 COCO Cascade Mask R-CNN, NVIDIA A5000, identical batch size, resolution, and mixed-precision settings. GFLOPs are profiled for the adapted/trainable computation path, not for the complete detector. Lower is better for all cost columns.

Method	Trainable #p	GFLOPs (adapted path)	iter/s	Peak mem.	AP^{bbox}/AP^{mask}
fine-tune	23.5M	739.00	**1.6**	5512 MB	41.2 / 35.9
LoRA	**0.1M**	**4.53**	1.5	**3565 MB**	38.6 / 35.1
ConvAdapter	5.8M	37.26	1.3	4434 MB	38.8 / 35.1
MSLoRA	0.7M	38.57	1.2	3999 MB	**42.9 / 38.4**

Baseline strength. The ResNet-50 Cascade Mask R-CNN baseline (41.2 box / 35.9 mask) matches the official MMDetection (Chen et al., 2019) R50-FPN $1\times$ result. The Swin-B baseline reaches 52.4 / 45.1, compared with 51.9 / 45.0 reported by the original Swin Transformer implementation under its $3\times$ multi-scale recipe. Full fine-tuning and MSLoRA share pretrained initialization, detector heads, data pipeline, schedule length, and evaluation protocol within each backbone. Appendix B shows that the full fine-tuning trajectories plateau before the end of training.

Variation across settings. The margin over full fine-tuning is not uniform across backbones. RegNetX-4.0 full fine-tuning is 0.1 box AP higher than MSLoRA, while the Swin-B difference is $+0.3$ box AP and several CNN settings show larger gains. The comparison therefore supports no uniform accuracy advantage across all architectures.

Training regimes. The COCO CNN experiments use $1\times$ schedules, whereas Swin-B uses $3\times$. Within each backbone, MSLoRA and full fine-tuning use the same schedule, data pipeline, and evaluation protocol. The magnitude of the full-fine-tuning gap should therefore not be compared across backbone families as if the training regime were fixed; the within-backbone comparisons remain the relevant quantities in Table 2.

4.7 Design choices

Table 6 examines projection grouping, bottleneck width, kernel set, post-gate nonlinearity, and the dense-prediction refinements. Unless stated otherwise, experiments start from the base gated configuration (G=4, D=128, $\mathcal{K}$=$\{3, 5, 7\}$) and vary one factor at a time; the dense-refinement rows are cumulative. Classification uses Food101 with Swin-B, while detection and instance segmentation use COCO with ResNet-50 Cascade Mask R-CNN at $1\times$.

Projection grouping. Reducing the number of projection groups increases both capacity and accuracy. G=1 gives the highest result in the sweep, reaching 93.7 top-1 on Food101 and 42.5/38.1 box/mask AP on COCO, compared with 92.9 and 41.9/37.8 at G=4. The latter, however, reduces the adapter from 5.2M to 1.8M parameters on Swin-B and from 1.8M to 0.7M on ResNet-50. We therefore use G=4 as the default parameter–accuracy trade-off; G=1 is retained as the higher-capacity MSLoRA* configuration in Table 2.

Bottleneck width. Increasing D from 64 to 128 to 256 raises Food101 top-1 from 92.5 to 92.9 to 93.1, while the D^2 term in Eq. 4 increases the parameter cost more rapidly. A depth-adaptive width of $0.25C$ reaches the same 92.9 top-1 as D=128 with a slightly larger adapter (1.9M versus 1.8M parameters). We use D=128 as the fixed default.

Kernel set. The multi-scale branch benefits from increasing the receptive-field range. A single 3×3 kernel reaches 41.4/37.4 box/mask AP, while $\{3, 5, 7\}$ reaches 41.9/37.8. Extending the set to $\{3, 5, 7, 9\}$ increases the adapter from 0.7M to 0.9M parameters for a further $+0.1$ box AP with unchanged mask AP. Using the larger $\{9, 11, 13\}$ set increases the parameter count to 1.3M without improving over $\{3, 5, 7\}$. We therefore use $\{3, 5, 7\}$ as the default.

Post-gate nonlinearity and dense refinements. LayerNorm+GELU after the multi-scale transformation is part of the default MSLoRA formulation across tasks. In the ablation setting, it changes Food101 top-1 from 92.9 to 93.0 and raises COCO performance from 41.9/37.8 to 42.5/38.2. For

Table 6: **MSLoRA design ablations.** G: projection groups; #p: trainable parameters; %: fraction of the backbone. Classification is Food101 / Swin-B; detection and segmentation are COCO / ResNet-50 Cascade Mask R-CNN. Subtable (f) is cumulative; all others vary one factor.

G	#p	%	top-1	top-5	#p	%	*bbox*	*mask*
ft.	88M	100%	92.7	98.6	23M	92%	41.2	35.9
1	5.2M	5.9%	93.7	99.2	1.8M	7.0%	42.5	38.1
2	2.9M	3.2%	92.2	99.1	1.1M	4.2%	41.6	37.3
4	1.8M	2.0%	92.9	99.2	0.7M	2.7%	41.9	37.8
8	1.2M	1.3%	92.6	99.1	0.6M	2.3%	41.3	37.4

(a) Projection groups. G=1 is most accurate; G=4 is the parameter-budget choice.

D	#p	%	top-1	top-5
64	0.8M	0.9%	92.5	98.9
128	1.8M	2.0%	92.9	99.2
256	4.3M	4.8%	93.1	99.1
0.25C	1.9M	2.1%	92.9	99.2

(b) Bottleneck width.

$\mathcal{K}$	#p	%	*bbox*	*mask*
[3]	0.6M	2.3%	41.4	37.4
[3,5]	0.6M	2.3%	41.7	37.6
[3,5,7]	0.7M	2.7%	41.9	37.8
[3,5,7,9]	0.9M	3.4%	42.0	37.8
[9,11,13]	1.3M	5.0%	41.9	37.8

(c) Kernel set.

update form	#p	top-1	top-5	#p	*bbox*	*mask*
additive	1.1M	92.7	99.1	0.3M	40.3	36.5
gate only	1.7M	92.6	99.1	0.6M	41.6	37.6
gated prod.	1.8M	92.9	99.2	0.7M	41.9	37.8

(d) Update form (= Table 1).

method	top-1	top-5	*bbox*	*mask*
baseline	92.9	99.2	41.9	37.8
+nonlin.	93.0	99.1	42.5	38.2

(e) Post-gate nonlinearity.

cumulative	top-1	top-5	*bbox*	*mask*
+ pool	93.0	99.1	42.5	38.2
+ gate	93.1	99.2	42.7	38.5
+ shuffle	93.2	99.2	42.9	38.4

(f) Dense refinements.

dense prediction, the remaining refinements are added cumulatively: global pooling leaves the result at 42.5/38.2, path gating reaches 42.7/38.5, and channel shuffle gives the final 42.9/38.4 configuration. Thus, the post-gate nonlinearity accounts for the largest single improvement beyond the base gated formulation, while the three dense refinements provide smaller complementary gains.

Taken together, the ablations expose two complementary uses of the adaptation budget. G and D control how much capacity is assigned to channel projection, whereas the multi-scale gate and post-gate nonlinearity determine how that adaptation is conditioned on the input and spatial location. The selected configuration is therefore not the highest-capacity point in the sweep; it preserves the spatially selective component while keeping the projection budget compact.

5 Limitations

Budget matching. The additive and gated controls in Table 1 are of the same order but not exactly parameter-matched. The gate-only control provides complementary evidence for the contribution of spatial modulation, but the full margin between the additive and gated-product variants cannot be attributed to update form alone.

No weight merging. The input-dependent gate cannot be folded into the frozen backbone weights, and MSLoRA has lower training throughput than the alternatives in Table 5. Its efficiency advantage is therefore primarily in trainable parameters and checkpoint storage rather than latency.

6 Conclusion

We studied how a small adaptation budget should be allocated when a pretrained vision backbone is frozen. At comparable budgets, replacing a purely additive low-rank residual with input-dependent spatial modulation changes pooled classification only slightly but produces a substantially larger improvement on dense prediction: on ResNet-50 COCO, the gated formulation improves the additive control by $+1.6$ box and $+1.3$ mask AP for 0.4M additional parameters.

MSLoRA instantiates this allocation in feature space through a grouped low-rank projection and a nonlinear multi-scale gate. Because the formulation does not depend on selecting architecture-specific weight matrices, it applies across convolutional, hierarchical-transformer, hybrid, and plain-ViT backbones. Across the evaluated dense-prediction settings, it uses 1.4–7.6% of backbone parameters while remaining competitive with or exceeding full fine-tuning, and on Swin-B it approaches a Swin-specific adapter with a substantially smaller default parameter budget.

The broader result is a budget-allocation principle rather than a new gating primitive: when the downstream task preserves spatial structure, spending part of limited adaptation capacity on input- and position-dependent modulation can be more effective than using the entire budget for a spatially shared low-rank update.

Reproducibility Statement

Sec. 4.1 gives optimizer, learning rates, schedules, augmentation, and freezing policy for every experiment; Sec. 3.3 states the exact module configuration used for each table; Eq. 4 gives the parameter count analytically. All datasets and detection frameworks are public. Upon acceptance, we will publicly release the source code, including the complete training and evaluation scripts, configuration files, pretrained model weights and checkpoints, and the scripts required to reproduce the main experiments reported in this paper.

Ethics Statement

This work studies efficient adaptation of pretrained vision backbones on public benchmarks. It introduces no new data collection and no new task capability. The usual concerns attaching to general-purpose detection and segmentation models, surveillance in particular, apply to the backbones being adapted rather than to the adaptation mechanism, but are not diminished by making adaptation cheaper.

AI use statement

AI was used for copy-editing and clarity of exposition. It was not used to generate research ideas, design or run experiments, analyze results, or produce any number reported here. All technical content and claims are the authors'.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: High quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1483–1498, 2019.

Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1551–1561, 2024.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

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, 2022a.

Tianrun Chen, Lanyun Zhu, Chaotao Deng, Runlong Cao, Yan Wang, Shangzhan Zhang, Zejian Li, Lingyun Sun, Ying Zang, and Papa Mao. Sam-adapter: Adapting segment anything in underperformed scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 3367–3375, 2023a.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022b.

Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In *International Conference on Learning Representations*, 2023b.

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.

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. *arXiv preprint arXiv:2010.11929*, 2020.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1): 98–136, 2015.

Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the norm layers. *arXiv preprint arXiv:2302.07937*, 2(4), 2023.

Meng-Hao Guo, Cheng-Ze Lu, Qibin Hou, Zhengning Liu, Ming-Ming Cheng, and Shi-Min Hu. Segnext: Rethinking convolutional attention design for semantic segmentation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1140–1156, 2022.

Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *Computational Visual Media*, 9(4):733–752, 2023.

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

Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient fine-tuning for vision transformers. *arXiv preprint arXiv:2203.16329*, 2022.

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. *International Conference on Learning Representations*, 2022.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

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. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039*, 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.

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

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

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 32100–32121, 2024.

Yen-Cheng Liu, Chih-Yao Ma, Junjiao Tian, Zijian He, and Zsolt Kira. Polyhistor: Parameter-efficient multi-task adaptation for dense vision tasks. *Advances in Neural Information Processing Systems*, 35:36889–36901, 2022a.

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.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12009–12019, 2022b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022c.

Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guangnan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*, 2023.

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.

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.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10428–10436, 2020.

Rajat Sahay and Andreas Savakis. Mopeft: A mixture-of-pefts for the segment anything model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2025. Workshop: Domain Generalization: Evolution, Breakthroughs, and Future Horizons.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Chongjie Si, Xuehui Wang, Xue Yang, Zhengqin Xu, Qingyun Li, Jifeng Dai, Yu Qiao, Xiaokang Yang, and Wei Shen. Flora: Low-rank core space for n-dimension. *arXiv preprint arXiv:2405.14739*, 10, 2024.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434, 2018.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Yi Xin, Siqi Luo, Haodi Zhou, Junlong Du, Xiaohong Liu, Yue Fan, Qing Li, and Yuntao Du. Parameter-efficient fine-tuning for pre-trained vision models: A survey. *arXiv preprint arXiv:2402.02242*, 2024. v5.

Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. *Advances in Neural Information Processing Systems*, 35:4203–4217, 2022.

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 Computer Vision and Pattern Recognition Conference*, pp. 20071–20081, 2025.

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

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

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.

A PARAMETER ANALYSIS

Table 7 shows how the group count G divides the adapter budget between the projection branch and the gating branch, summed over all insertion points of a backbone. The transformation (gating) count is constant within a backbone because D is fixed; it differs between backbones because they have different numbers of insertion points. The ratio $r = \#\text{proj.}/\#\text{trans.}$ measures how much of the budget the projection consumes. At G=1 on Swin-L the projection is 11.5× the gate; at G=4 it is 2.8×.

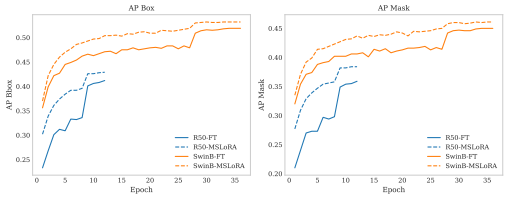
Table 7: Adapter parameter counts by projection group count G, summed over insertion points. *proj.* and *trans.* are the projection and gating branches; $r = \#\text{proj.}/\#\text{trans.}$. The gating count is fixed within each backbone because D is fixed, and differs across backbones because the number of insertion points differs. Rows beyond G=8 are omitted for ResNet-50, where the projection count rounds to 0.1M.

G	Swin-L			ResNet-50		
	#proj.	#trans.	r	#proj.	#trans.	r
1	6.9M	0.6M	11.5	1.4M	0.4M	3.50
2	3.4M	0.6M	5.6	0.7M	0.4M	1.75
4	1.7M	0.6M	2.8	0.3M	0.4M	0.75
8	0.8M	0.6M	1.3	0.1M	0.4M	0.25
16	0.4M	0.6M	0.6	n/a	n/a	n/a

B CONVERGENCE AND QUALITATIVE BEHAVIOR

Fig. 2(a) plots COCO training metrics for full fine-tuning and MSLoRA under the ResNet-50 1× setting. The full fine-tuning curves flatten before the end of the schedule, which is the evidence referred to in Sec. 4.6 that the comparison is not against an under-trained checkpoint.

Fig. 2(b) and Fig. 3 provide qualitative examples of the activation patterns produced by the two adaptation strategies. In the examples shown, MSLoRA produces responses distributed more broadly over object extents, whereas full fine-tuning concentrates on smaller discriminative regions. These figures are illustrative and are not used as quantitative evidence for the paper's claims.



(a) COCO training metrics.



(b) Activation maps: full fine-tuning (top) vs. MSLoRA (bottom).

Figure 2: Convergence and qualitative comparison. Panel (b) is illustrative only.

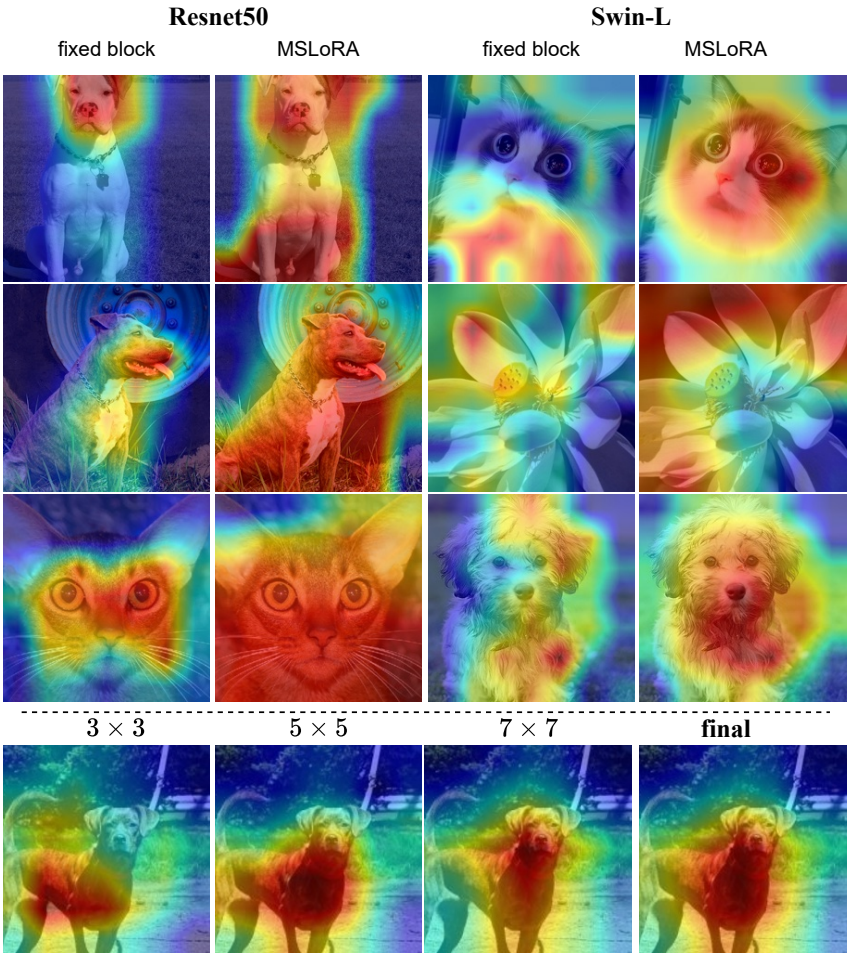


Figure 3: *Top:* a frozen backbone block compared with the same block under MSLoRA reweighting. *Bottom:* the effect of filters at different spatial scales within the gating branch. Illustrative examples; see the caveat above.