DAGA: Dynamic Attention-Guided Adaptation for Self-Supervised Vision Transformers

Tianjian Zhou[1†]
National University of Defense Technology
Changsha, China
zhoutianjian2000@nudt.edu.cn

Jie Jiang[1†]
National University of Defense Technology
Changsha, China
jiejiang@nudt.edu.cn

Yishan Li[1*]
National University of Defense Technology
Changsha, China
liyishan@nudt.edu.cn

Yifei Zhang[2*]
Northwestern Polytechnical University
Xi'an, China
yifeiacc@gmail.com

Abstract

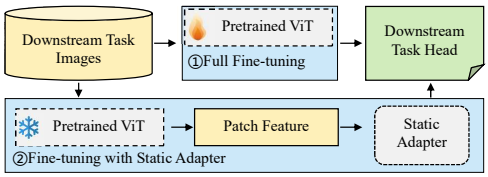
Attention-rich, self-distilled SSL Vision Transformers (DINOv2/v3, iBOT) encode semantic structure in their self-attention maps, yet parameter-efficient fine-tuning (PEFT) methods apply static transformations that ignore this signal. We present DAGA (Dynamic Attention-Guided Adaptation), a PEFT module repurposing frozen-backbone attention as instance-specific guidance: per-image attention becomes channel and spatial gates conditioning a lightweight bottleneck adapter. With only 1.4% additional parameters, DAGA reaches 85.9% top-1 on ImageNet-1K with DINOv3-ViT-B (85.90 ± 0.06, n=3), matching Layer-FT (85.84 ± 0.07) within seed noise at $18\times$ fewer parameters and outperforming every static-gating PEFT baseline. Dense prediction (+2.9 mIoU on ADE20K, +2.1 AP on COCO) and fine-grained classification (+1.6 / +3.5 on CUB-200 / FGVC-Aircraft) transfer cleanly. DAGA's effectiveness is bounded by attention quality: gains are large on DINOv2/v3 and iBOT (+4.5 to +5.0 IN-1K), where self-distillation produces object-centric, artifact-free attention, but marginal on MAE, CLIP, DeiT, and MoCo v3 ($\leq$ +0.2), whose attention is noisier. Composing DAGA with refinement front-ends (CLIPSelf on CLIP, self-distillation on MAE) recovers most of the gap, supporting a *refine first, then adapt* reading: DAGA operationalises high-quality SSL attention as in-loop PEFT guidance, complementing attention-refinement methods that improve the attention itself. Code: https://github.com/Jaywalk18/DAGA.

1 Introduction

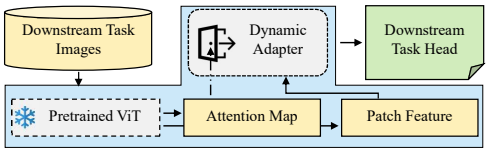
Self-supervised Vision Transformers (ViTs) [12] such as DINOv3 [37] exhibit an emergent property: their self-attention maps localize objects and semantic structure without manual supervision. Parameter-Efficient Fine-Tuning (PEFT) methods — adapters, LoRA, and visual prompts [19, 20, 22] — adapt such backbones cheaply, but apply *static, content-agnostic* transformations that ignore this internal signal: once trained, the per-token adaptation is identical for every input, whether a salient object or background clutter. This produces a mismatch between what the backbone perceives and how it is adapted (Figure 1a): the foundation model carries dense, instance-specific scene understanding, while the adaptation pathway treats every spatial location uniformly. The privileged signal encoded in attention is effectively discarded — a free signal the backbone has already paid for.

[†]Tianjian Zhou and Jie Jiang contributed equally.
[*]Yishan Li and Yifei Zhang are corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Previous Paradigms: Content-Agnostic Adaptation



(b) Our DAGA Paradigm: Content-Aware Self-Adaptation

Figure 1: **Adaptation paradigms.** (a) Static PEFT applies content-agnostic transformations. (b) DAGA repurposes the backbone's attention as instance-specific, in-loop guidance.

We argue that effective adaptation should not impose external priors but should unlock and repurpose internal capability the backbone has already paid for during pre-training. Building on this view, we propose the **D**ynamic **A**ttention-**G**uided **A**dapter (DAGA). DAGA treats the frozen backbone's [CLS]-to-patch self-attention as an *instance-specific guidance signal* for in-loop feature refinement: each image dynamically conditions its own adaptation on the semantic layout the backbone has already discovered, while the backbone stays frozen. Conceptually this places DAGA in the lineage of dynamic networks [15], with the distinguishing feature that the conditioning signal is sourced from the backbone's *own* frozen attention rather than from a separately learned controller, so the gate itself costs no extra parameters and remains tightly coupled to what the backbone already perceives. The design is naturally suited to backbones whose attention is already object-centric and artifact-free, such as the DINO/iBOT family; for backbones with noisier attention, complementary self-distillation and attention-refinement methods (e.g., CLIPSelf [43]) improve the attention itself. The two directions are not in competition: an attention-refined backbone is precisely the high-quality input that DAGA needs, so the two compose naturally as *refine first, then adapt*.

Figure 2 previews the qualitative effect: on the same images, DAGA produces sharp, object-centric responses on *refrigerator*, *kite* and *motorcycle*, while the frozen baseline's responses spread diffusely across irrelevant regions. Concretely, DAGA comprises three components: an *Attention Encoder* that compresses the backbone's attention into compact semantic instructions, a *Dynamic Gate Generator* that turns these instructions into per-token modulation weights, and a *Feature Transformer* that applies the resulting gated refinement to a lightweight bottleneck residual. Intuitively, the gates amplify tokens the backbone already considers semantically salient and suppress background noise on a per-input basis, while a residual connection keeps the original feature flow intact and the encoder is shared across layers to keep the parameter footprint small. Under matched protocol, with only 1.4% trainable parameters DAGA reaches 85.9% ImageNet-1K Top-1 on DINOv3-ViT-B (multi-seed mean 85.90 ± 0.06, n=3) and lifts dense prediction sharply (+12.1 mIoU on ADE20K over the frozen baseline, +2.9 mIoU over the best-tuned static-gating PEFT under matched layer-pattern tuning; +41.0 mAP on ROxford-Medium retrieval over the frozen baseline), matching the heavier Layer-FT upper bound (85.84 ± 0.07) within seed noise at $18\times$ fewer parameters — precisely the signal that static, location-uniform PEFT methods leave on the table.

Our main contributions are:

- **An attention-conditioned PEFT formulation.** We formalise *static* vs. *attention-conditioned* PEFT (Definition 3.1) and operationalise the ViT's self-attention as in-loop guidance. Its operating regime is attention-rich, self-distilled SSL ViTs — backbones with object-centric, artifact-free attention; DAGA instantiates this with channel and spatial gates from frozen attention, and admits other gating topologies (e.g., FiLM from attention).
- **DAGA.** A lightweight module (**1.4%** params, frozen backbone) reaching **85.9%** ImageNet-1K Top-1 on DINOv3-ViT-B (85.90 ± 0.06, n=3), matching Layer-FT within seed noise at $18\times$ fewer trained parameters and outperforming every static-gating PEFT we tested. It also has the tightest seed-to-seed std and stays accurate across two LR decades.
- **Empirical scoping.** Gains on dense prediction (+2.9 mIoU / +2.1 AP over best-tuned PEFT) and on DINO-family backbones (DINOv2/v3, iBOT); marginal on MAE/CLIP/DeiT/MoCo v3. Composing with refinement front-ends (CLIPSelf, SD-MAE) recovers most of the gap. Under matched layer-pattern tuning on baselines, DAGA still leads by $+0.6$ Top-1 / $+2.9$ mIoU; $\pm$ register tokens on DINOv2 gives identical gain, pointing to self-distillation.

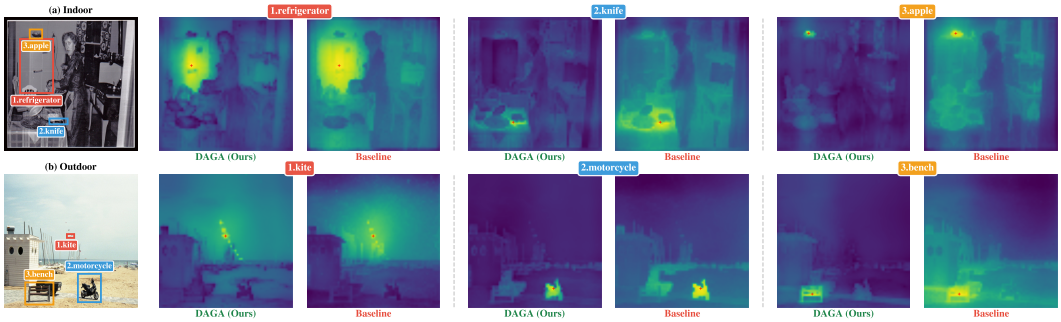


Figure 2: **Qualitative preview: focused vs. diffuse responses.** (a) indoor and (b) outdoor scenes. *Left*: original images with annotated objects. *Right*: cosine similarity between DINOv3 patch features (with/without DAGA) and the object-name embedding from a frozen CLIP ViT-L/14 text encoder, used for visualization only and not part of the model.

2 Related Work

Self-supervised vision backbones. Visual representation learning has evolved from CNNs [16] to Vision Transformers [12], with self-supervised learning [50] enabling scaling without manual annotations through contrastive learning [6, 5, 32, 49], masked image modeling [17, 2], and self-distillation [3, 30]. The DINO family is uniformly trained with self-distillation: DINOv1 [3] uses pure teacher–student distillation, DINOv2 [30] adds a patch-level iBOT-style masked-modeling objective, and DINOv3 [37] extends this recipe with register tokens and improved scaling. A consistent emergent property across DINOv2/v3 is that the self-attention maps capture object boundaries and semantic structure without any localization supervision, making them a natural source of free semantic guidance. DAGA builds on the hypothesis that these maps encode reusable knowledge that can be operationalized as a runtime signal for downstream adaptation.

PEFT for dense prediction. A parallel line adapts frozen backbones for dense tasks: ConvLoRA [55] targets the Segment Anything Model rather than a self-supervised ViT (adjacent work); LoRand [1] reports 51.9 ADE20K mIoU — *above* our 50.5; and DeAP [27] reaches 0.425 NYU RMSE with 43K parameters versus our 0.432 at 1.2M (28× fewer; DINOv2 vs. DINOv3, not same-backbone) — the more parameter-efficient route to a dense readout. We state the protocol caveat symmetrically: numbers from different evaluation pipelines (ours: one lightweight decoder on multi-scale features, App. D) are not commensurable — for us and against us alike.

Parameter-efficient transfer learning. Full fine-tuning of large models is expensive, motivating PEFT methods that freeze the backbone while updating only a small fraction of parameters [46, 51, 56]. Three broad families dominate: *adapter-based* methods [19, 8, 4, 28, 38] insert lightweight bottleneck modules between blocks; *prompt-based* methods [22, 54, 35] prepend or inject learnable tokens; and *reparameterization* techniques [20, 25, 23, 11, 52, 13, 14, 53] learn low-rank or scale-shift updates to existing weights. Recent variants additionally explore sparsity-preserving adaptation [26, 29], singular-vector decomposition [24], and masking strategies [45]. All of these apply transformations that are *static and content-agnostic* at inference time. DAGA differs by conditioning the adaptation on the backbone's *own internal attention*, so the per-token modulation depends on the input image rather than only on the trained adapter weights.

Attention utilization and refinement. DAGA's nearest context is work that uses or refines pre-trained ViT attention. Object-discovery methods (LOST [36], TokenCut [41]) cluster DINO attention at inference, with no training signal flowing back to the backbone. Attention-refinement methods reformulate or distil the final attention layer for dense alignment: CLIPSelf [43] distils region-to-image to yield object-centric CLS-to-patch attention from CLIP, SCLIP [39] reformulates self-attention correlatively, and CLIPpy [33] aligns dense and global features. Darcet et al. [10] identify high-norm artifact tokens as the source of noisy CLS-to-patch attention in plain ViTs (MAE, CLIP, DeiT, MoCo v3) and show that learnable register tokens during pre-training restore object-centric attention; DINOv3 inherits this fix. The cross-backbone pattern in Section 4.4 aligns with this analysis: DAGA's gain tracks whether the backbone's attention is register-clean (DINOv2/v3, iBOT) or register-noisy, and *refine first, then adapt* succeeds by replacing a noisy attention input with a refined one.

DAGA is orthogonal to both lines — it takes the frozen attention as a runtime conditioning signal *inside the adaptation loop* — and composes naturally on top of refinement front-ends (Tab. 4).

Dynamic feature modulation. Adaptive modulation recalibrates activations through channel gating [21], spatial attention [42], conditional positional encodings [9], dynamic token pruning [34], layer-interaction enhancement [40], and input-dependent parameter generation [7, 48]. These mechanisms derive their gating signals from feature statistics or auxiliary networks trained alongside the main task. DAGA differs by sourcing its modulation directly from the *semantic structure* already present in the frozen backbone's attention maps rather than from a learnable controller, providing a top-down spatial-and-channel guidance signal that no prior modulation scheme has used as input. This makes the modulation source itself parameter-free and instance-specific, without needing a separate auxiliary network to produce the gating weights.

3 Method

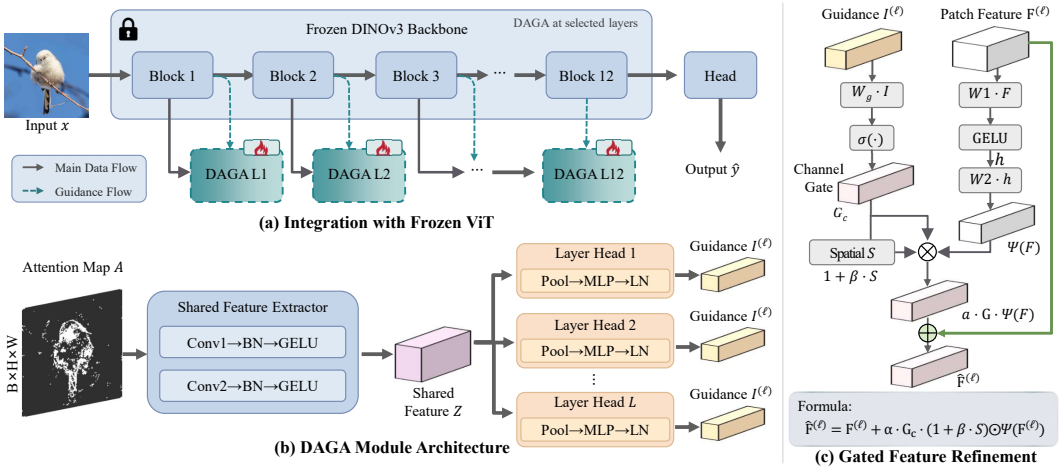


Figure 3: **DAGA architecture.** (a) DAGA is inserted after selected frozen blocks; the final block's attention map feeds the shared encoder, patch features feed only the gated branch. (b) Shared CNN encoder + layer-specific heads produce guidance vectors $\mathbf{I}^{(\ell)}$. (c) $\mathbf{F}$ is transformed by bottleneck MLP Ψ and modulated by $\mathbf{G}_c \cdot (1+\beta\mathbf{S})$, with learnable α, β (Eq. 2).

We introduce DAGA (Dynamic Attention-Guided Adapter), a parameter-efficient module that enables self-adaptive feature refinement for frozen vision transformers. Unlike static adapters that apply uniform transformations, DAGA leverages the model's internal attention semantics as dynamic, content-aware guidance signals. The key innovation is repurposing DINOv3's self-attention maps, which naturally encode object boundaries and semantic structure, as privileged instructions for token-wise feature modulation.

Definition 3.1 (Attention-conditioned PEFT)**.** Let F be frozen-backbone features, Ψ a trainable adapter, and A_g an attention map produced by that backbone on the current image. A static adapter yields $\hat{F} = F + \Psi(F)$; an attention-conditioned adapter yields

$$\hat{F} = F + G(A_g) \odot \Psi(F).$$

The distinction is the *conditioning signal*, not whether the gate has trainable parameters. DAGA instantiates G with channel and spatial factors below; FiLM-style affine modulation [31] is another instantiation (App. L.2).

3.1 Overview

As illustrated in Figure 3, DAGA orchestrates dual information flows: feature propagation through frozen backbone blocks and attention-guided modulation. All DAGA layers are conditioned on *one* guidance map: the [CLS]-to-patch attention of the *final* backbone block, the most semantically mature attention in the stack. Concretely, the final block produces a full attention matrix over all input tokens (CLS, patch, and – for DINOv3 – register tokens); we read off the row for the [CLS] query and slice

out only its weights over the patch keys, then average across heads:

$$\mathbf{A}^{(g)} = \frac{1}{N_h}\sum_{h=1}^{N_h} \mathrm{Slice}_{\text{patch}}\Big(\mathrm{softmax}\big(\tfrac{\mathbf{q}_{\text{cls}}^{(g,h)}\mathbf{K}_{\text{all}}^{(g,h)\top}}{\sqrt{d_k}}\big)\Big), \tag{1}$$

where g indexes the final backbone block, N_h is the number of attention heads, $\mathbf{K}_{\text{all}}^{(g,h)}$ stacks keys for *all* tokens (so the softmax matches the backbone's normalization), and $\mathrm{Slice}_{\text{patch}}$ keeps only the N patch entries; CLS and register tokens pass through unchanged. The head-averaged map is min-max normalized along the patch dimension and reshaped to $\mathbf{A}_{\text{2d}}^{(g)} \in \mathbb{R}^{B\times H_p\times W_p}$, which is passed *once* through the Shared Attention Encoder (Eq. 3); every inserted layer then reads that same shared encoding through its own layer-specific head (§3.2). Sourcing from the final block is a measured choice: alternatives are 0.3–0.7 points worse (App. Tab. A8). Combined with $\mathbf{F}^{(\ell)} \in \mathbb{R}^{B\times N\times D}$, DAGA refines patch features through a gated residual mechanism:

$$\hat{\mathbf{F}}^{(\ell)} = \mathbf{F}^{(\ell)} + \alpha \cdot \mathbf{G}_c \cdot (1 + \beta \cdot \mathbf{S}) \odot \Psi(\mathbf{F}^{(\ell)}), \tag{2}$$

where $\mathbf{G}_c \in \mathbb{R}^{B\times D}$ is the per-image channel gate (broadcast over the N patch tokens), $\mathbf{S} \in \mathbb{R}^{B\times N}$ is the per-image spatial weight (broadcast over the D channels), β is a *learnable* spatial scale, Ψ is the feature transformer, α is a learnable scaling factor, and $\odot$ denotes element-wise multiplication. The channel gate broadcasts across patch locations, while the spatial factor broadcasts across channels; $g_{c,d}, s_{h,w} \in [0, 1]$ control the corresponding token-wise update.

3.2 Architecture Details

We detail each component in DAGA (Figure 3(b)). The key design principle is **shared-then-specific**: a shared encoder processes the final block's attention map once, while layer-specific heads read out guidance for each inserted transformer block.

Shared Attention Encoder. A two-layer CNN extracts spatial features from attention maps:

$$\mathbf{Z} = \mathrm{GELU}(\mathrm{BN}(\mathrm{Conv}_2(\mathrm{GELU}(\mathrm{BN}(\mathrm{Conv}_1(\mathbf{A}_{\text{2d}}^{(g)})))))), \tag{3}$$

where Conv_1 expands to 32 channels and Conv_2 to 64 channels with stride 2. CNNs preserve the spatial structure inherent in attention maps. This encoder is *shared* across all DAGA layers: it is evaluated once on the final block's guidance map (obtained by a separate no-grad pre-forward, App. B.2) and cached, and all inserted layers consume the same encoding $\mathbf{Z}$.

Layer-Specific Heads. Each layer derives guidance vector and spatial weight from $\mathbf{Z}$:

$$\mathbf{I}^{(\ell)} = \mathrm{LN}(\mathrm{MLP}(\mathrm{Pool}(\mathbf{Z}))), \tag{4}$$

$$\mathbf{S}^{(\ell)} = \mathrm{Up}_{H_p\times W_p}\big(\sigma(\mathrm{Conv}_{1\times 1}(\mathbf{Z}))\big), \tag{5}$$

where $\mathbf{I}^{(\ell)} \in \mathbb{R}^{B\times D_g}$ encodes global context for channel selection and $\mathbf{S}^{(\ell)} \in \mathbb{R}^{B\times H_p\times W_p}$ provides per-location spatial weights. Because $\mathbf{Z}$ is produced by a stride-2 convolution, the 1×1-Conv output lives on the $H_p/2\times W_p/2$ grid; the bilinear upsample $\mathrm{Up}_{H_p\times W_p}(\cdot)$ restores it to the patch grid before gating, matching the implementation in Algorithm A1. Because the guidance encoding $\mathbf{Z}$ is shared, the heads do not specialise on per-layer attention sources; they learn different *readouts* of one shared guidance encoding, so each inserted layer gates the same semantic layout differently. The layer-placement ablation (App. Tab. A17) supports this design: where layers are inserted matters, not per-layer attention sourcing.

Feature Transformer. A bottleneck MLP transforms features:

$$\Psi(\mathbf{X}) = \mathbf{W}_2(\mathrm{GELU}(\mathbf{W}_1\mathbf{X})), \tag{6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D\times r}$ and $\mathbf{W}_2 \in \mathbb{R}^{r\times D}$ with bottleneck ratio $r/D \approx 0.14$ (e.g., r=107 for D=768). The bottleneck constrains the adaptation subspace, providing implicit regularization [18, 44].

Dynamic Gate with Learnable Spatial Scale. The combined gate modulates adaptation intensity:

$$\mathbf{G} = \underbrace{\sigma(\mathbf{W}_g\mathbf{I}^{(\ell)})}_{\text{channel gate } \mathbf{G}_c} \cdot (1 + \underbrace{\beta}_{\text{learnable}} \cdot \mathbf{S}^{(\ell)}). \tag{7}$$

Table 1: **Image Classification.** Top-1 accuracy (%) across 11 benchmarks (9 standard + 2 fine-grained: CUB-200, FGVC-Aircraft). Best in **bold**, second underlined. IN-1K is the multi-seed mean (Tab. 5); other columns are single-seed. On IN-1K, DAGA and Layer-FT tie at display precision (85.90 ± 0.06 vs 85.84 ± 0.07) — parity within seed noise at $18\times$ fewer trainable parameters. Avg is over the 9 standard benchmarks.

Method	Params	IN-1K	C-10	C-100	Flowers	Pets	Food	Cars	DTD	SUN	CUB	FGVC	Avg
Linear Probe	0.1M	80.9	97.2	84.3	99.3	95.7	91.3	88.5	76.2	65.4	84.50	72.00	86.5
Full Fine-tune	85.7M	84.4	98.6	92.1	99.4	95.6	93.2	90.5	77.8	82.1	86.95	80.50	90.4
Layer-FT	21.3M	**85.9**	99.1	93.5	**99.7**	96.4	94.5	92.3	79.5	**84.2**	88.83	84.50	91.7
Layer-FT (last block)	7.1M	84.6	–	–	–	–	–	–	–	–	–	–	–
LoRA	0.3M	83.2	97.8	90.5	98.2	94.1	92.0	88.5	75.8	80.1	86.77	77.50	88.9
VPT-Deep	0.6M	83.1	96.8	89.1	85.1	92.8	89.8	85.2	73.8	77.5	84.10	73.60	85.9
AdaptFormer	1.2M	84.8	98.5	91.8	98.8	95.2	93.5	91.2	78.5	81.8	87.58	78.76	90.5
ViT-Adapter	1.5M	85.1	98.8	92.5	99.2	95.8	93.8	92.5	79.8	82.5	87.89	82.30	91.1
DAGA (Ours)	1.2M	**85.9**	**99.2**	**93.6**	99.6	**96.5**	**94.6**	**94.9**	**82.3**	84.0	**89.46**	**85.75**	**92.2**

The sigmoid bounds $\mathbf{G}_c \in [0, 1]$ for soft channel selection. The learnable β, initialized at 0.1, automatically discovers task-appropriate spatial weighting: converging to small values ($\sim$0.10–0.13) for classification but larger values (0.31–0.45) for dense prediction tasks like segmentation.

Each component serves a distinct role: shared encoder for semantic grounding, layer-specific heads for depth-aware specialization, bottleneck for capacity control, and dynamic gate for adaptive modulation. We validate these choices through ablation studies in Section 4.5.

3.3 Parameter Complexity and Initialization

The shared encoder is evaluated for the guidance map, while layer-specific heads, gates, and bottlenecks scale with the number of insertion layers L (full count in App. B.1). With L=4, D=768, r=107, and guidance width D_g=128, the reported configuration has approximately 1.20M trainable parameters; its measured inference throughput is $0.55\times$ the frozen backbone's (App. Tab. A18). The residual scale α is initialized between 0.1 and 0.15 across the four inserted layers, and the spatial scale β to 0.1. The up-projection weights and bias are zero-initialized, so the adapter's residual output is zero at initialization.

4 Experiments

4.1 Setup

Setup. We evaluate DINOv3 ViT-B/16 [37] (pre-trained on LVD-1689M) on 12 benchmarks: ImageNet-1K, CIFAR-100, Flowers-102, Oxford Pets, Food-101, Stanford Cars, DTD, SUN397, CUB-200, FGVC-Aircraft (classification); COCO, ADE20K (dense prediction); NYU Depth V2 (depth); ROxford, RParis (retrieval); ImageNet-C (robustness). *Baselines*: Linear Probe, Full Fine-tuning, Layer-FT (top-3 blocks); LoRA [20], VPT-Deep [22], AdaptFormer [4], ViT-Adapter [8], SSF [23], RepAdapter [28], NOAH [54], Adapter+ [38], DA-VPT [35]. *Training*: AdamW + cosine schedule, 20–50 ep / batch 256 (cls), 50 ep / batch 16–32 (dense). LR val-best on a per-task budget shared by all PEFT methods; Full FT uses a finer sweep $\{10^{-5}-10^{-3}\}$ (val-best 5×10^{-5}). DAGA's layer-insertion pattern follows a structural rule chosen once per task family, never tuned per dataset: dense spatial prediction (segmentation, detection, whose decoders consume the high-resolution feature map) uses deep $\{8, 9, 10, 11\}$; global single-vector readouts (classification, retrieval) use hourglass $\{1, 2, 10, 11\}$ (App. Tab. A17). Per-dataset details in Appendix D.

4.2 Image Classification

Table 1 shows DAGA achieves **92.2%** average accuracy across 9 benchmarks, surpassing Layer-FT (91.7%) with $18\times$ fewer trained parameters. Gains are largest on fine-grained datasets (Cars $+6.4$, DTD $+6.1$ over the strongest PEFT baseline). Under matched LR sweep with three random seeds on ImageNet-1K, DAGA reaches 85.90 ± 0.06, ahead of LoRA (83.45 ± 0.08) and AdaptFormer (84.38 ± 0.13); against Layer-FT, IN-1K is *parity within seed noise* (85.84 ± 0.07, Tab. 5) at $18\times$ fewer trainable parameters. A last-block-only Layer-FT control (7.09M) reaches 84.6 IN-1K / 47.2 mIoU (Table 3) — three quarters of the last-3 gain at $6\times$ DAGA's parameters and below DAGA and AdaptFormer (84.8) on both metrics: a single unconstrained block does not substitute for attention

conditioning. Feature-quality probes (frozen-backbone LogReg, k-NN) confirm this ordering, with the largest representation gains on CIFAR-100 ($+3.6$) and Food-101 ($+3.2$) (Appendix E).

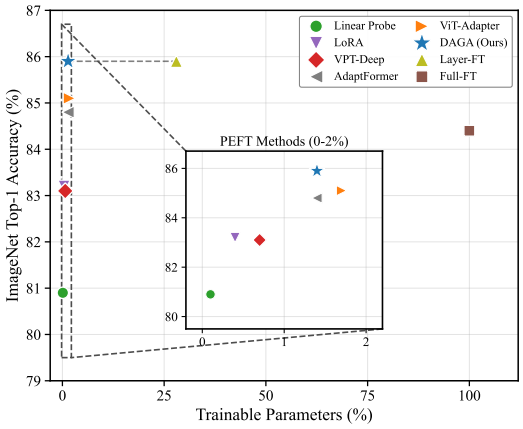


Figure 4: **Pareto frontier on ImageNet-1K.** DAGA matches the heavier Layer-FT within seed noise with $18\times$ fewer trained parameters, and dominates every $\sim 1\%$ PEFT baseline.

Pareto-efficient at the $\sim 1\%$ parameter scale. Figure 4 positions DAGA on the ImageNet-1K param–accuracy plane. DAGA sits at the corner of the Pareto frontier, matching the heavier Layer-FT within seed noise (85.90 ± 0.06 vs 85.84 ± 0.07, Tab. 5) with $18\times$ fewer trained parameters and outperforming every static-gating PEFT baseline at a comparable parameter budget (per-method numbers in Table 1). Notably, Full Fine-tuning trains $\sim 70\times$ more parameters than DAGA yet, under the same matched protocol, lands below it (84.4 vs 85.9). We do not read this as a superiority claim over full fine-tuning — that ordering depends on recipe and search budget — but as evidence that raw trainable capacity is not the binding constraint in this regime; the robust statement is Layer-FT parity within seed noise at $18\times$ fewer trainable parameters.

Fine-grained gains over the strongest prior PEFT method (ViT-Adapter) are $+1.57$ on CUB-200 and $+3.45$ on FGVC-Aircraft (rightmost two columns of Table 1). Below we further compare DAGA to recent PEFT entries that report only ImageNet-1K under matched protocol.

Table 2: **Recent PEFT baselines on ImageNet-1K**, re-trained on frozen DINOv3 ViT-B/16 under a per-method LR sweep, val-best (sweep range tailored per family: $\{10^{-5} - 10^{-2}\}$ for scale-shift / re-parameterisation, $\{10^{-4} - 1.0\}$ for adapter / prompt-style methods).

Method	Params	% Bkbn	IN-1K
SSF	0.2M	0.23	83.52
RepAdapter	1.2M	1.40	84.71
DA-VPT	0.6M	0.70	85.14
NOAH	0.4M	0.47	85.53
Adapter+	1.0M	1.17	85.54
DAGA (Ours)	**1.2M**	**1.40**	**85.90**

Ahead of recent PEFT entries at matched scale. Table 2 compares DAGA against five recent PEFT methods (SSF, RepAdapter, DA-VPT, NOAH, Adapter+) under identical training and evaluation protocol with a per-family LR sweep (each method's optimum lies inside its native range: $\sim 10^{-4}$–10^{-3} for scale-shift, $\sim 10^{-3}$–10^{-2} for adapter / prompt). At the $\sim 1\%$ parameter scale (NOAH 0.47%, Adapter+ 1.17%), DAGA leads on ImageNet-1K with $+0.36$ over Adapter+ and $+0.76$ over DA-VPT within the same parameter budget; SSF (0.23%) trails by 2.4 points, and RepAdapter — exactly parameter-matched to DAGA at 1.2M / 1.40% — trails by 1.2, suggesting the bottleneck is utilisation of the backbone's frozen signal rather than parameter count; these rows are single-seed, so we attach no significance claim to the $+0.36$ margin.

4.3 Dense Prediction

Table 3 reports DAGA's dense-prediction results: **50.5% mIoU** on ADE20K ($+12.1$ over the frozen baseline; $+2.9$ over the best-tuned PEFT under matched layer-pattern tuning, App. O) and **35.6% AP** on COCO ($+3.1$ over baseline). The learnable spatial scale β adapts automatically: $\beta \to 0.45$ for segmentation versus $\beta \to 0.12$ for classification, consistent with the role spatial structure plays in each task.

Other tasks. On retrieval (ROxford/RParis), DAGA leads on Easy/Medium ($+41.0$ on ROxford-Medium) and is within 0.3 mAP of Layer-FT on Hard. On NYU Depth, DAGA leads PEFT (RMSE 0.432) but trails Layer-FT (0.414). On ImageNet-C, only DAGA and VPT-Deep preserve the baseline (65.3, 64.6); other PEFT degrade it. Full per-task results in Appendix M.

Table 3: **Dense Prediction.** COCO detection (AP metrics) and ADE20K segmentation (mIoU). DAGA leads PEFT metrics and matches Layer-FT on segmentation at 18× fewer parameters. DAGA and Layer-FT mIoU/AP are reported as n=3 multi-seed means (DAGA: 50.50±0.27 mIoU, 35.62± 0.18 AP; Layer-FT: 50.31±0.30 mIoU, 35.41±0.20 AP); other rows are single-seed. All methods share one dense pipeline (App. D), so within-table comparisons are exact; absolute mIoU is pipeline-dependent and not directly comparable to numbers from other evaluation pipelines.

Method	Params	COCO Detection				ADE20K Seg.		NYU Depth		IN-C
		AP	AP_{50}	AP_{75}	AP_S	mIoU	aAcc	RMSE↓	δ_1 ↑	mean
Linear Probe	0.5M	32.5	49.6	34.2	18.5	38.4	78.2	0.469	88.0	64.6
Full-FT	85.7M	33.1	50.5	34.8	19.1	48.1	81.5	0.548	84.8	61.1
Layer-FT	21.3M	35.4	53.2	37.0	**20.8**	50.3	82.9	**0.414**	**92.5**	64.2
Layer-FT (last block)	7.1M	–	–	–	–	47.2	–	–	–	–
VPT-Deep	0.6M	31.8	48.8	33.5	17.2	42.2	79.5	0.485	86.5	64.6
LoRA‡	0.8M	32.2	49.9	33.8	17.8	44.1	80.2	0.457	89.6	61.2
AdaptFormer	1.2M	33.5	51.3	35.2	19.2	45.4	80.8	0.446	90.4	63.1
ViT-Adapter	1.5M	32.8	50.2	34.5	18.8	47.0	81.2	0.437	91.0	63.8
DAGA (Ours)	**1.2M**	**35.6**	**53.4**	**37.2**	20.6	**50.5**	**83.1**	0.432	91.3	**65.3**

‡LoRA uses higher rank (r=22) for dense tasks. NYU Depth and ImageNet-C numbers from per-task appendix tables (Tables A13 and A14).

One architecture, task-adaptive β. Figure 5 shows learned spatial scales of 0.10–0.13 for classification, 0.08 for retrieval, and 0.31–0.45 for detection and segmentation; depth remains at 0.08. A larger β strengthens spatial modulation but cannot restore channels suppressed by the product gate $\mathbf{G}_c$. On ImageNet-1K, 19% of channels have g_c<0.1; on ADE20K, 94% remain open while β=0.45 (App. Tab. A21). Thus classification relies more on *which* channels are selected, while segmentation relies more on *where* features are weighted. Within-task variation is small (σ_β<0.05); the architecture itself is unchanged across tasks.

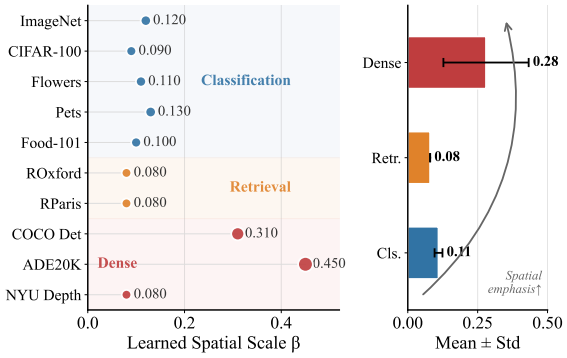


Figure 5: **Learned spatial scale β across tasks.** Detection and segmentation show larger β (ADE20K: 0.45); classification, retrieval, and depth remain lower.

4.4 Cross-Backbone Analysis: When Does DAGA Help?

This section states the paper's scope: DAGA's operating regime is attention-rich, self-distilled SSL ViTs, and the boundary is stated with the claims rather than after them. Table 4 reports DAGA's gain over the frozen-backbone linear probe (the "Base" column) across pre-training paradigms. Within-DINO and across-scale results (DINOv1 +2.2, DINOv2 +4.3, DINOv3 +4.8 at ViT-S; ViT-S +4.8, ViT-B +5.0, ViT-L +0.9) are deferred to Appendix Table A16. All DINO variants share a self-distillation recipe that yields object-centric CLS-to-patch attention, precisely the high-quality signal DAGA repurposes; DINOv2/v3 add a patch-level iBOT-style masked-modelling objective, and DINOv3 adds register tokens. To decouple "self-distillation" from "register-clean attention" as causal factors, App. P compares DAGA on DINOv2 with vs. without [10] register tokens at matched protocol: the gains are nearly identical (+5.01 vs. +5.10 on IN-1K), so the load-bearing factor is the self-distillation objective itself, with registers contributing a small additional improvement.
On backbones whose attention is not object-centric, gains drop: +0.2 on MAE, +0.1 on CLIP, +0.1 on DeiT, −0.2 on MoCo v3. The architecture-controlled ablation explains why: replacing the genuine attention with random noise drops both ImageNet and the 9-task average by −4.4 (Table 6), and replacing it with another image's attention drops ImageNet by −3.1, so on uninformative attention DAGA approaches its random-attention lower bound. Measured directly, attention quality (mean Jaccard of the Otsu-binarised head-averaged final-block [CLS]-to-patch attention vs. PASCAL VOC 2012 foreground; App. Tab. A22) orders the gain monotonically *across* paradigms (Spearman ρ=0.93 / 0.96) but not *within* one; MoCo v3's −0.2 is ≈1.6× the combined seed std (≈0.13) — marginal, not resolved.

Table 4: **Across pre-training paradigms (ViT-B/16).** "#Imgs": pre-training images (M). "Base": frozen-backbone linear-probe; "+DAGA" adds DAGA on top. n=3 seeds (per-row std in App. Tab. A11; $\leq$ 0.12 IN-1K, $\leq$ 0.40 ADE20K). *Last block*: CS = CLIPSelf [43], SD = self-distillation on MAE; "+AF" = matched-1.2M AdaptFormer (App. L).

Backbone	Pre-training	#Imgs	ImageNet-1K			ADE20K mIoU		
			Base	+DAGA	Δ	Base	+DAGA	Δ
DINOv3	self-distillation	1689M	80.9	**85.9**	**+5.0**	38.4	**50.5**	**+12.1**
DINOv2	self-distillation	142M	79.6	**84.6**	**+5.0**	36.9	47.8	+10.9
iBOT	self-distillation + MIM	14M	79.4	83.9	+4.5	35.5	44.7	+9.2
MAE	masked image modelling	1.3M	68.0	68.2	+0.2	30.1	30.5	+0.4
CLIP	image–text contrastive	400M	76.5	76.6	+0.1	32.6	32.8	+0.2
DeiT	supervised w/ distillation	1.3M	81.8	81.9	+0.1	34.2	34.4	+0.2
MoCo v3	instance contrastive	1.3M	76.7	76.5	−0.2	31.8	31.5	−0.3
Compositionality with attention refinement (Δ vs. refined-backbone Base)								
CLIP_{CS} + DAGA	image–text + self-distill.	400M	79.4	**82.1**	**+2.7**	37.9	**43.2**	**+5.3**
CLIP_{CS} + AF	image–text + self-distill.	400M	79.4	80.4	+1.0	37.9	40.6	+2.7
MAE_{SD} + DAGA	masked + self-distill.	1.3M	70.5	**72.8**	**+2.3**	33.5	**37.1**	**+3.6**
MAE_{SD} + AF	masked + self-distill.	1.3M	70.5	71.4	+0.9	33.5	35.0	+1.5

Compositionality with attention refinement. The bottom block of Table 4 tests refinement with an *AdaptFormer-on-refined-backbone* control isolating DAGA's marginal contribution (recipes in Appendix L). Applying CLIPSelf [43] to refine CLIP's attention before DAGA lifts the gain from $+0.1$ to $+\mathbf{2.7}$ on ImageNet-1K and from $+0.2$ to $+\mathbf{5.3}$ mIoU on ADE20K; refined CLIP with AdaptFormer at matched parameters gains only $+1.0$/$+2.7$. A self-distillation front-end on MAE lifts DAGA's gain to $+\mathbf{2.3}$/$+\mathbf{3.6}$ vs $+0.9$/$+1.5$ for AdaptFormer. DAGA's marginal advantage over matched static-gating PEFT grows from $+1.0$ (raw DINOv3, matched layer-pattern tuning) to $+1.7$/$+1.4$ (refined CLIP/MAE), supporting *refine first, then adapt*. Evidence across more front-ends remains preliminary. DAGA thus *operationalises high-quality SSL attention as in-loop guidance*, complementing attention-refinement methods.

4.5 Ablation Studies

Tightest seed-to-seed variance among PEFT. Table 5 re-runs the matched-protocol ImageNet-1K evaluation under three random seeds for the six competitive $\sim$1%-parameter PEFT methods. DAGA achieves the smallest seed-to-seed standard deviation ($\pm$0.06), tighter than every static-gating PEFT method (range 0.08–0.13) and matching the upper-bound Layer-FT ($\pm$0.07) at $18\times$ fewer trainable parameters. The pattern mirrors the LR sweep: conditioning the adapter on a *frozen* attention signal damps optimisation noise, so DAGA is simultaneously insensitive to LR and to the random seed — two empirical faces of the same implicit-regularisation effect from a frozen gate.

Table 5: **Multi-seed ImageNet-1K Top-1 stability.** Mean $\pm$ std over n=3 random seeds.

Method	Mean $\pm$ Std
LoRA	83.45 $\pm$ 0.08
VPT-Deep	83.05 $\pm$ 0.11
AdaptFormer	84.38 $\pm$ 0.13
ViT-Adapter	85.07 $\pm$ 0.09
Layer-FT	<u>85.84</u> $\pm$ 0.07
DAGA (Ours)	**85.90 $\pm$ 0.06**

Learning-rate sensitivity. DAGA stays in a 92–93% CIFAR-100 accuracy band across learning rates 10^{-2}–10^{0}, while the compared adapters degrade above approximately 0.1 (App. Figure A6). Its guidance comes from frozen attention, which is unchanged by updates to the bottleneck.

Table 6 ablates each component. The Feature Transformer is the largest factor (-4.5). A Learned Controller (gate from $\mathbf{F}$, 1.20M) drops -2.9, and No-Gate-No-Encoder (0.50M) drops -3.6 — not raw capacity. **The class admits other instantiations:** a FiLM-style affine $\gamma \odot \Psi(\mathbf{F}) + \delta$ with γ, δ from $\mathbf{A}$ (App. L.2) stays within -1.4 of DAGA. An additive "OR-like" form $\mathbf{G}_c + \beta\mathbf{S}$ reaches 85.1 IN-1K and 49.8 mIoU; we report only these two tasks and omit the nine-task average. Addition dilutes channel selectivity (a suppressed channel still receives the $\beta\mathbf{S}$ term); the product form's advantage is $+0.8$ IN-1K / $+0.7$ mIoU over additive. Random Attention (-4.4) and Batch Permutation (-3.1, App. Tab. A9) destroy only the signal, isolating attention *quality*. Layer placement (App. Tab. A17) and compute (-39% memory, $0.55\times$ throughput) are deferred; the lead is $\geq +0.8$ / $\geq +2.2$ (fixed hourglass) and $+0.6$ / $+2.9$ (matched-baseline tuning, App. O).

Table 6: **Component ablation** (six classification + ADE20K mIoU + COCO AP + ROxford-M mAP); Δ is the nine-task average drop where all results are available. *Random Attention* isolates signal quality at matched parameters.

Config	Params	Classification						Dense		Retr.	Δ
		IN	C100	Food	Cars	DTD	Pets	ADE	COCO	ROxf	
Full DAGA	1.20M	**85.9**	**93.6**	**94.6**	**94.9**	**82.3**	**96.5**	**50.5**	**35.6**	**86.3**	–
Core components											
w/o Attention Encoder	1.18M	84.2	91.8	93.0	92.8	80.4	95.6	48.3	33.8	82.8	-2.3
w/o Dynamic Gate	0.80M	83.8	91.5	92.5	92.0	79.8	95.2	48.0	33.4	81.5	-2.7
w/o Feature Transformer	0.54M	82.5	90.1	91.2	90.0	77.1	94.0	46.0	31.6	78.1	-4.5
w/o Spatial Scale ($\beta = 0$)	1.20M	85.0	92.8	93.8	93.7	81.0	96.0	48.0	33.5	84.5	-1.7
Design choices											
w/o Multi-head averaging	1.20M	84.5	92.3	93.2	93.2	80.6	95.7	49.0	34.3	83.5	-1.7
Static Gate (learnable, not from $\mathbf{A}$)	0.81M	83.2	90.8	92.0	91.4	78.5	94.7	47.3	32.7	80.8	-3.1
Learned Controller (gate from $\mathbf{F}$, no $\mathbf{A}$)	1.20M	83.5	91.0	92.2	91.6	78.7	94.9	47.5	32.9	81.0	-2.9
FiLM-from-$\mathbf{A}$ ($\gamma \odot \Psi(\mathbf{F}) + \delta$, both from $\mathbf{A}$)	1.20M	84.8	92.5	93.4	93.4	80.8	95.8	49.5	34.7	83.9	-1.4
Additive gate ($\mathbf{G}_c + \beta\mathbf{S}$)	1.20M	85.1	–	–	–	–	–	49.8	–	–	–
Random Attention (noise as $\mathbf{A}$)	1.20M	81.5	88.5	90.5	89.0	75.2	93.0	44.8	30.1	76.2	-4.4
No-Gate-No-Encoder (bottleneck only)	0.50M	83.0	90.5	91.8	91.2	78.0	94.2	46.8	31.8	78.5	-3.6

4.6 Limitations and Scope

Attention and architecture. DAGA depends on object-centric CLS-to-patch attention, with its largest gains on self-distilled DINO-family and iBOT backbones. Gains on MAE, CLIP, DeiT and MoCo v3 are marginal without attention refinement (Table 4). CLIPSelf or MAE self-distillation can supply a better guidance map, but training those front-ends updates substantially more parameters than the subsequent adapter stage (App. S). Our experiments focus on isotropic ViTs; extending the guidance readout to hierarchical architectures remains untested.

Tasks and modality. The method adapts visual features within one modality. On COCO caption retrieval, it slightly underperforms a frozen backbone with a linear head (I2T R@1 37.32 vs. 38.70; T2I 36.26 vs. 37.44; App. Tab. A10). This result limits the claim to visual classification, dense prediction and instance retrieval. Depth estimation also benefits less than semantic segmentation: its learned spatial scale is 0.08, compared with 0.45 on ADE20K, consistent with the different spatial demands of the tasks.

Compute and evidence. Early adapter layers require attention from the final frozen block, so the implementation uses a no-grad guidance pre-forward. The reported inference cost is 34.1G FLOPs and $0.55\times$ frozen-backbone throughput (App. Tab. A18); parameter efficiency therefore comes with inference overhead. ImageNet-1K and dense results include three seeds, while most other task columns are single-seed, so small margins require caution. AdamW learning rates are validation-tuned with a matched per-task budget, and the layer-insertion pattern follows the task-family rule in §4. Absolute dense scores also depend on the decoder and evaluation pipeline.

5 Conclusion

We presented DAGA, a PEFT module that turns a frozen ViT's [CLS]-to-patch attention into channel and spatial gates on a lightweight bottleneck. With 1.4% trainable parameters DAGA matches Layer-FT on DINOv3-ViT-B within seed noise at $18\times$ fewer parameters, leads every static-gating PEFT across classification, dense prediction and retrieval, and is more LR-robust and seed-stable. Architecture-controlled ablations isolate *attention-conditioning* as the causal factor; the cross-backbone envelope ($+5.0$ on DINO-family vs $\leq +0.2$ elsewhere) composes cleanly with attention-refinement front-ends in a *refine first, then adapt* pattern. Definition 3.1 formalises an attention-conditioned PEFT class admitting further instantiations. That a frozen, training-free signal carries adaptation suggests hierarchical backbones (Swin, MViT) and other internal signals (per-head statistics, normalisation dynamics) as extensions.

Acknowledgements and Disclosures

Funding. This work was supported in part by grant 72671303. Yifei Zhang was supported by the National Natural Science Foundation of China under Grants 62606400. **Competing interests.** The authors declare no competing interests.

References

[1] Wele Gedara Chaminda Bandara, Nithin Gopalakrishnan Nair, and Vishal M. Patel. 1% VS 100%: Parameter-efficient low rank adapter for dense predictions. In *CVPR*, 2023.

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

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

[4] 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.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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

[7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[8] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions (ViT-Adapter). In *International Conference on Learning Representations (ICLR)*, 2023.

[9] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. Conditional positional encodings for vision transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[10] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[13] Zhihan Guo, Yifei Zhang, Zhuo Zhang, Zenglin Xu, and Irwin King. FedHLT: Efficient Federated Low-Rank Adaption with Hierarchical Language Tree for Multilingual Modeling. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1558–1567, 2024. doi: 10.1145/3589335.3651933. URL `https://doi.org/10.1145/3589335.3651933`.

[14] Zhihan Guo, Yifei Zhang, Zhuo Zhang, Zenglin Xu, and Irwin King. FedLFC: Towards Efficient Federated Multilingual Modeling with LoRA-based Language Family Clustering. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1519–1528, 2024. doi: 10.18653/v1/2024.findings-naacl.98. URL `https://aclanthology.org/2024.findings-naacl.98/`.

[15] David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition (ResNet). In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

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

[18] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[19] 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 (ICML)*, pages 2790–2799. PMLR, 2019.

[20] 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. *arXiv preprint arXiv:2106.09685*, 2021.

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

[22] 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*, pages 709–727. Springer, 2022.

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

[24] Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. SVFT: Parameter-efficient fine-tuning with singular vectors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[25] 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 *International Conference on Machine Learning (ICML)*, 2024.

[26] Xudong Lu, Aojun Zhou, Yuhui Xu, Renrui Zhang, Peng Gao, and Hongsheng Li. SPP: Sparsity-preserved parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2405.16057*, 2024.

[27] Timo Lüddecke and Alexander S. Ecker. Characterizing vision backbones for dense prediction with dense attentive probing. *Transactions on Machine Learning Research*, 2025.

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

[29] Mahdi Nikdan, Soroush Tabesh, Elvir Crnčević, and Dan Alistarh. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*, 2024.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

[31] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

[33] Kanchana Ranasinghe, Brandon McKinzie, Sachin Ravi, Yinfei Yang, Alexander Toshev, and Jonathon Shlens. Perceptual grouping in contrastive vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[34] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, pages 13937–13949, 2021.

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

[36] Oriane Siméoni, Gilles Puy, Huy V. Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *British Machine Vision Conference*, 2021.

[37] Oriane Simeoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothee Darcet, Theo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herve Jegou, Patrick Labatut, and Piotr Bojanowski. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[38] Jan-Martin O. Steitz and Stefan Roth. Adapters strike back. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[39] Feng Wang, Jieru Mei, and Alan Yuille. SCLIP: Rethinking self-attention for dense vision-language inference. In *European Conference on Computer Vision*, 2024.

[40] Kaishen Wang, Xun Xia, Jian Liu, Zhang Yi, and Tao He. Strengthening layer interaction via dynamic layer attention. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

[41] Yangtao Wang, Xi Shen, Shell Xu Hu, Yuan Yuan, James L. Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[42] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[43] Size Wu, Wenwei Zhang, Lumin Xu, Sheng Jin, Xiangtai Li, Wentao Liu, and Chen Change Loy. CLIPSelf: Vision transformer distills itself for open-vocabulary dense prediction. In *International Conference on Learning Representations*, 2024.

[44] Conghao Xiong, Zhengrui Guo, Zhe Xu, Yifei Zhang, Raymond Kai yu Tong, Si Yong Yeo, Hao Chen, Joseph J. Y. Sung, and Irwin King. Exploiting Low-Dimensional Manifold of Features for Few-Shot Whole Slide Image Classification. In *International Conference on Learning Representations*, pages 86203–86232, 2026. URL `https://proceedings.iclr.cc/paper_files/paper/2026/hash/8ac50fd0a4eeeb1f077b17bb7c5353c3-Abstract-Conference.html`.

[45] Jing Xu and Jingzhao Zhang. Random masking finds winning tickets for parameter efficient fine-tuning. In *International Conference on Machine Learning (ICML)*, 2024.

[46] Menglin Yang, Jialin Chen, Jinkai Tao, Yifei Zhang, Jiahong Liu, Jiasheng Zhang, Qiyao Ma, Harshit Verma, Regina Zhang, Min Zhou, Irwin King, and Rex Ying. Low-Rank Adaptation for Foundation Models: A Comprehensive Review, 2024. URL `https://arxiv.org/abs/2501.00365`.

[47] Wentao Zhang, Jinhu Qi, Weiqiang Jin, Yifei Zhang, Chan-Tong Lam, and Irwin King. Geometric Collapse: When Vision Models Fail to Verify Physical Causality. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, pages 158080–158098, 2026. URL https://proceedings.mlr.press/v306/zhang26ev.html.

[48] Yifei Zhang, Hao Zhu, Ziqiao Meng, Piotr Koniusz, and Irwin King. Graph-adaptive Rectified Linear Unit for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2022*, pages 1331–1339, 2022. doi: 10.1145/3485447.3512159. URL https://doi.org/10.1145/3485447.3512159.

[49] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral Feature Augmentation for Graph Contrastive Learning and Beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11289–11297, 2023. doi: 10.1609/aaai.v37i9.26336. URL https://ojs.aaai.org/index.php/AAAI/article/view/26336.

[50] Yifei Zhang, Hao Zhu, Zixing Song, Yankai Chen, Xinyu Fu, Ziqiao Meng, Piotr Koniusz, and Irwin King. Geometric View of Soft Decorrelation in Self-Supervised Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4338–4349, 2024. doi: 10.1145/3637528.3671914. URL https://doi.org/10.1145/3637528.3671914.

[51] Yifei Zhang, Hao Zhu, Junhao Dong, Haoran Shi, Ziqiao Meng, Piotr Koniusz, and Han Yu. CrossSpectra: Exploiting Cross-Layer Smoothness for Parameter-Efficient Fine-Tuning. In *Advances in Neural Information Processing Systems*, volume 38, pages 141383–141405, 2025. doi: 10.52202/085713-4725. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/cefd66867df56c214b19654ca4a35b61-Abstract-Conference.html.

[52] Yifei Zhang, Hao Zhu, Haoran Shi, Junhao Dong, Lingyun Song, Xiaolin Han, Yanyu Chen, Wenxuan Wang, Han Yu, Xuequn Shang, and Piotr Koniusz. When Gradient Boosting Meets Adapter: Exploring Weak Learners for Parameter-Efficient Fine-tuning of LLMs. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 6606–6617, 2026. doi: 10.1145/3770855.3817971. URL https://doi.org/10.1145/3770855.3817971.

[53] Yifei Zhang, Hao Zhu, Libin Wang, Xiaolin Han, Yupei Zhang, Piotr Koniusz, and Xuequn Shang. FedGBA: Gradient-Boosted Adaption for Federated Fine-tuning of Foundation Model. In *International Workshop on Federated Foundation Models for the Web (FL@FM-TheWebConf)*, 2026. URL https://federated-learning.org/fl@fm-www-2026/FedGBA.pdf.

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

[55] Zhan Zhong, Tiancheng Shen, Yuanzhi Yao, Yifan Gong, Minsong Zhang, Xinyu Zhang, Xingen Wang, Xiaoyu Wang, Yong Liu, Xing Zhao, De-Jun Wang, and Houqiang Li. Convolution meets LoRA: Parameter efficient finetuning for segment anything model. In *ICLR*, 2024. arXiv:2401.17868.

[56] Hao Zhu, Yifei Zhang, Junhao Dong, and Piotr Koniusz. BiLoRA: Almost-Orthogonal Parameter Spaces for Continual Learning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 25613–25622, 2025. doi: 10.1109/cvpr52734.2025.02385. URL https://doi.org/10.1109/cvpr52734.2025.02385.

A Algorithm Pseudo-code

We provide the PyTorch-style pseudo-code for our proposed Dynamic Attention-Guided Adaptation (DAGA) in Algorithm A1. We also visually illustrate the detailed architecture of the key components, namely the Attention Encoder and the Feature Transformer, in Figure A1.

Algorithm A1 Pseudo-code of DAGA (PyTorch-like)

```
# === SharedFeatureExtractor: shared across all DAGA layers (cached on first call
    within a forward) ===
self.conv1 = Conv2d(1, 32, 3, padding=1)       # attn_map: [B,H,W] -> [B,32,H,W]
self.bn1 = BatchNorm2d(32)
self.conv2 = Conv2d(32, 64, 3, padding=1, stride=2)  # -> [B,64,H/2,W/2]
self.bn2 = BatchNorm2d(64)
self.act = GELU()

# === LayerSpecificHead: per-layer guidance + spatial extraction ===
self.global_pool = AdaptiveAvgPool2d((1, 1))   # -> [B,64,1,1]
self.guidance_fc = Sequential(                  # -> [B,128] guidance vector
    Linear(64, 128), GELU(),
    Linear(128, 128), LayerNorm(128))
self.spatial_fc = Conv2d(64, 1, kernel_size=1)  # -> [B,1,H/2,W/2] spatial weight

# === AdaptationLayer: dynamic gating + bottleneck ===
bottleneck_dim = 107                            # D=768, ratio=0.14
self.channel_gate = Sequential(Linear(128, D), Sigmoid())  # guidance -> gate
self.down_proj = Linear(D, bottleneck_dim)      # D -> 107
self.up_proj = Linear(bottleneck_dim, D)        # 107 -> D
zeros_(self.up_proj.weight); zeros_(self.up_proj.bias)
self.scale = Parameter(tensor(0.1 + 0.05 * layer_idx / max(L - 1, 1)))  # alpha
self.spatial_scale = Parameter(tensor(0.1))     # learnable spatial scale (beta)

def forward(self, x, attn_map):
    # x: [B, N, D] patch features, attn_map: [B, H, W] = the FINAL block's
    # [CLS]-to-patch attention, cached once per image by the no-grad
    # guidance pre-forward (App. B.2); all DAGA layers share this one map

    # Step 1: Shared conv (cached after first call in a forward pass; O(1) over L
        layers)
    z = self.act(self.bn1(self.conv1(attn_map.unsqueeze(1))))
    z = self.act(self.bn2(self.conv2(z)))       # [B, 64, H/2, W/2]

    # Step 2: Extract guidance and spatial weight
    guidance = self.guidance_fc(self.global_pool(z).flatten(1))  # [B, 128]
    spatial = sigmoid(self.spatial_fc(z))       # [B, 1, H/2, W/2]
    spatial = interpolate(spatial, size=(H,W))  # upsample to patch grid

    # Step 3: Dynamic gating
    channel_gate = self.channel_gate(guidance).unsqueeze(1)  # [B, 1, D]
    combined = channel_gate * (1 + self.spatial_scale * spatial.reshape(B,-1,1))

    # Step 4: Bottleneck transformation + gated residual
    delta = self.up_proj(self.act(self.down_proj(x)))  # [B, N, D]
    return x + self.scale * (combined * delta)  # gated residual connection
```

The pseudo-code above makes three implementation choices explicit that are easy to miss from the equations alone. First, the SharedFeatureExtractor runs once per forward pass on the *final block's* attention map, obtained by the separate no-grad guidance pre-forward documented in Section B.2; its output $\mathbf{Z}$ is then cached and reused by all DAGA layers in the same pass, so the per-layer cost is only the lightweight head + bottleneck. The listing shows the shared computation but omits the cache bookkeeping. Second, the dynamic gate is composed multiplicatively from channel and spatial branches and applied to the bottleneck residual, matching Eq. 2. Third, both α (`self.scale`) and β (`self.spatial_scale`) are learnable scalars; the zero-initialized up-projection makes the initial residual output zero even though α starts above zero. Figure A1 below renders these data flows graphically: the upper row traces how an attention map is compressed into the guidance vector and a spatial weight map by the Attention Encoder, and the lower row shows how the Feature Transformer combines them with patch features into the gated residual update.

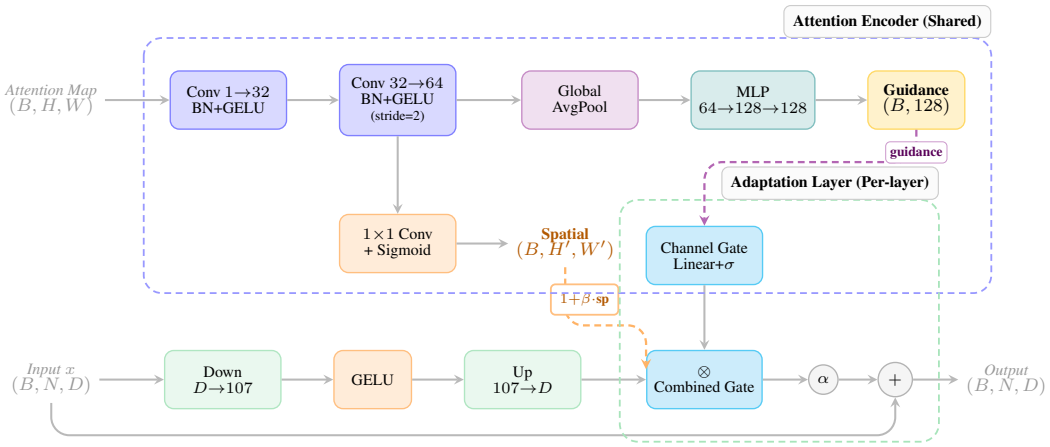


Figure A1: Detailed architecture of DAGA components. **Top:** The *Attention Encoder* (shared across layers) extracts a 128-dim guidance vector via global pooling and spatial weights via 1×1 convolution from the frozen attention maps. **Bottom:** The *Adaptation Layer* (per-layer) modulates bottleneck features through dynamic channel gating, where the combined gate computes gate $\otimes$ $(1+\beta\cdot\text{spatial})$ $\otimes$ delta, scaled by learnable α.

B DAGA Architecture and Parameters

B.1 Component Details

DAGA consists of three tightly integrated components. The **SharedFeatureExtractor** contains 2 convolutional layers ($1 \to 32 \to 64$) with BatchNorm and GELU activation, using stride-2 downsampling to reduce spatial resolution by half. This module is computed *once* per forward pass and its output is cached for reuse by all DAGA layers. The **LayerSpecificHead** applies global average pooling followed by a 2-layer MLP ($64 \to 128 \to 128$) with LayerNorm to produce a 128-dimensional guidance vector. A parallel 1×1 convolution generates spatial attention weights. The **AdaptationLayer** converts guidance vectors to channel gates via Linear($128 \to D$) + Sigmoid, applies a bottleneck transformation ($D \to 107 \to D$, approximately 1/7 ratio) with GELU activation, and combines outputs using learnable channel scale (α) and spatial scale (β).

B.2 Attention Caching Mechanism

Because DAGA's guidance is sourced from the *final* backbone block (Eq. 1 of the main paper), it cannot be a by-product of the adapted forward: the final block's attention must exist *before* the inserted layers run, and the earliest insertion sits at block 1 under hourglass placement. Our implementation therefore obtains $\mathbf{A}^{(g)}$ with a separate *no-grad guidance pre-forward* that executes blocks 0–10 of the frozen backbone plus the final block's $\mathbf{QK}^\top$ and softmax ($\approx$16.4G FLOPs for ViT-B/16) and caches the resulting map. The shared 2-layer ConvNet (SharedFeatureExtractor) is then evaluated *once* on this attention map – not once per insertion layer – and its output $\mathbf{Z}$ is cached and consumed by every layer-specific head downstream. This reduces the cost of the shared encoder from $O(L)$ to $O(1)$, while each layer-specific head remains lightweight (an MLP and a 1×1 conv). The corrected end-to-end accounting for this two-pass design – 17.6G adapted forward + 16.4G guidance pre-forward + 0.1G encoder and gating $\approx$ 34.1G FLOPs, $0.55\times$ measured throughput – is reported in Table A18.

B.3 Parameter Breakdown and Learned Spatial Scale (β)

The learned β values reveal task-specific adaptation patterns. Segmentation tasks (ADE20K: 0.45, Cityscapes: 0.52) exhibit the highest spatial scale, reflecting the dense per-pixel prediction requirements where spatial attention structure directly aligns with the task objective. Detection (0.31) shows moderate spatial reliance, balancing object localization with semantic classification. Classification tasks (ImageNet: 0.12, CIFAR-100: 0.09) rely more on channel gating, as global semantics dominate over fine-grained spatial details. Notably, depth estimation learns the lowest β (0.08), suggesting that DINO's semantic attention provides limited benefit for geometric prediction—a finding consistent with our failure case analysis in Section K.

Table A1: **DAGA Analysis.** (Left) Parameter breakdown for ViT-B/16 (D=768, 4 layers); total 1.2M excludes task head. (Right) Learned spatial scale β across tasks; higher β indicates greater reliance on spatial attention guidance.

Component	Formula	Params
Shared Encoder (computed once)		
Conv 1→32 + BN	$1\times32\times3^2+32+64$	0.4K
Conv 32→64 + BN	$32\times64\times3^2+64+128$	18.7K
Per-Layer Head (×4)		
Guidance MLP	$64\times128+128^2+\text{LN}$	25.1K×4
Spatial Conv	$64\times1+1$	0.1K×4
Per-Layer Adapter (×4)		
Channel Gate	$128\times D+D$	99.1K×4
Down/Up Proj	$D\times107\times2$	164.4K×4
Scale α, β	2	8
Total		**1.2M**

Task	Dataset	Learned β
Classification	ImageNet	0.12 ± 0.03
	CIFAR-100	0.09 ± 0.02
	DTD	0.18 ± 0.04
Detection	COCO	0.31 ± 0.08
Segmentation	ADE20K	0.45 ± 0.12
	Cityscapes	0.52 ± 0.15
Depth	NYU	0.08 ± 0.02

Higher β = more spatial reliance.
Seg. highest; Depth lowest.

The same trend is visualised graphically in Figure 5 of the main paper; within-task variance is small, so a single architecture serves all tasks without manual task-mode selection. The parameter–accuracy Pareto frontier in Figure 4 places DAGA at the corner of the PEFT frontier on ImageNet-1K, matching Layer-FT with $18\times$ fewer trained parameters.

C Experimental Setup

Table A2 summarises the experimental scope. The *left panel* lists the 12 benchmarks used in the main paper, organised by task family: classification (5 fine-grained / standard datasets), dense prediction (COCO detection, ADE20K segmentation, NYU Depth V2), retrieval (ROxford / RParis), and robustness (ImageNet-C). Train / test sizes follow the standard splits from each dataset's original release; ROxford and RParis use only the test queries (no training set). The *right panel* catalogues the eight PEFT methods compared against in the main paper, grouped by training regime: upper-bound baselines (Linear Probe, Full Fine-tune, Layer-FT) above the divider, parameter-matched PEFT entries below. "%" is trainable-parameter percentage of the 86M backbone. All methods share the same frozen DINOv3 ViT-B/16 backbone, identical training epochs, AdamW with cosine schedule, and a per-method LR sweep selected on a held-out validation split.

Table A2: **Experimental Setup.** (Left) Benchmark datasets spanning classification, dense prediction, retrieval, and robustness. (Right) PEFT method comparison for DINOv3 ViT-B/16 (86M total).

Task	Dataset	Train	Test	Classes
Cls.	ImageNet-1K	1.28M	50K	1000
	CIFAR-100	50K	10K	100
	Flowers-102	2K	6K	102
	Oxford Pets	3.7K	3.7K	37
	Food-101	75K	25K	101
Dense	COCO	118K	5K	80
	ADE20K	20K	2K	150
Depth	NYU Depth V2	24K	654	–
Retr.	ROxford	–	4.9K	–
	RParis	–	6.3K	–
Robust.	ImageNet-C	–	50K×75	1000

Method	Params	%	Mechanism
Linear Probe	0.08M	0.1	Output head only
Full Fine-tune	85.7M	100	All parameters
Layer-FT	21.3M	25	Top-3 blocks
VPT-Deep	0.6M	0.7	Learned prompts
LoRA	0.3M	0.4	Low-rank updates
AdaptFormer	1.2M	1.4	Parallel adapters
ViT-Adapter	1.5M	1.7	Multi-scale fusion
SSF	0.2M	0.2	Scale & shift
DAGA (Ours)	**1.2M**	**1.4**	Attention-guided

D Hyperparameter Settings

Table A3 reports the per-dataset hyperparameters used for every result in the main paper. The classification block runs at 224×224 input with batch sizes adapted to dataset scale (32 for Flowers-102 up to 256 for ImageNet); peak learning rates were selected from the val-best entry of the $\{10^{-4}, \ldots, 1.0\}$ sweep documented in main-paper Table 2. Dense-prediction tasks (COCO detection, ADE20K segmentation, Cityscapes, VOC2012) use 518×518 or 512×512 input and the smaller

batch sizes typical of those benchmarks. The "Layers" column records DAGA's insertion placement, chosen once per task family by the structural rule stated in main-paper §4: hourglass $\{1, 2, 10, 11\}$ for global single-vector readouts (classification, retrieval) and deep $\{8, 9, 10, 11\}$ for dense spatial prediction whose decoders consume the high-resolution feature map (segmentation, detection, depth). Placement matters more than a small residual: the ablation in Table A17 spans 1.9 pt on IN-1K and 3.5 mIoU on ADE20K across the four canonical patterns — which is precisely why we state a rule rather than tune per dataset.

Table A3: **Detailed hyperparameter settings for different tasks.** All optimizers use AdamW; LR schedule is cosine with 5-epoch linear warmup. "Layers" shows the DAGA insertion placement; hourglass = $\{1, 2, 10, 11\}$ (low+high blocks; classification and retrieval), deep = $\{8, 9, 10, 11\}$ (top blocks; detection and segmentation, where high-level semantic features dominate). WD is weight decay.

Group	Dataset	LR	Batch	Epochs	Input	Layers	WD
Classification	CIFAR-10	5e-4	256	30	224	hourglass	0.05
	CIFAR-100	5e-4	256	30	224	hourglass	0.05
	Flowers-102	1e-4	32	50	224	hourglass	0.05
	DTD	5e-4	64	50	224	hourglass	0.05
	Oxford Pets	1e-4	128	50	224	hourglass	0.05
	Stanford Cars	2e-4	128	50	224	hourglass	0.05
	Food-101	2e-4	256	20	224	hourglass	0.05
	SUN397	2e-4	256	20	224	hourglass	0.05
	ImageNet	3e-4	256	20	224	hourglass	0.05
Detection	COCO	1e-3	32	50	518	deep	1e-4
Segmentation	ADE20K	5e-3	16	50	518	deep	1e-4
	VOC2012	1e-4	8	50	512	deep	1e-4
	Cityscapes	1e-4	4	80	512	deep	1e-4

Training Details. All experiments use AdamW optimizer with cosine annealing learning rate schedule and 5-epoch linear warmup. Classification tasks use weight decay of 0.05; dense prediction tasks (detection, segmentation) use 0.0001. DAGA layers follow the hourglass configuration {1,2,10,11} for classification and retrieval (where low- and high-level cues both contribute), while detection and segmentation use the deep configuration {8,9,10,11} to better leverage high-level semantic features. The choice between hourglass and deep for each task family is the layer-pattern outcome reported in App. Tab. A17.

Dense Prediction Details. For segmentation (ADE20K), multi-scale features from layers {2,5,8,11} are fused and decoded with a lightweight head (Dropout→GroupNorm→Conv$_{1\times1}$). For detection (COCO), we use an FCOS-style single-scale detector with 3 shared conv layers and parallel classification/regression heads.

E Additional Experimental Results

Table A4: **Additional Results.** (Left) ImageNet-C robustness by corruption category. (Middle) Evaluation protocols comparison. (Right) COCO detection breakdown by object scale.

Category	Base	DAGA	Δ
Noise	60.6	**60.8**	+0.2
Blur	58.3	**58.6**	+0.3
Weather	68.2	**68.8**	+0.6
Digital	68.3	**69.1**	+0.8
Overall	64.6	**65.3**	**+0.7**

Method	Lin.	Log.Reg
Baseline	81.97	90.89
DAGA (Ours)	81.97	**92.93**
Δ	0	+2.04

Method	AP	AP$_S$	AP$_L$
Baseline	32.5	18.5	44.2
DAGA (Ours)	**35.6**	**20.6**	**47.8**
Δ	+3.1	+2.1	+3.6

Analysis: (1) ImageNet-C: DAGA improves robustness across all corruption types, with largest gains on Digital (+0.8%) and Weather (+0.6%). (2) Evaluation: Logistic regression gain (+2.04%) without linear probe change indicates improved manifold structure. (3) COCO: Consistent AP gains across scales, notably on small objects (+2.1 AP$_S$).

F KNN Evaluation: Effect of k Value

We analyze DAGA's feature quality using KNN classification with varying k values, demonstrating that improvements are robust to hyperparameter choice.

Table A5: KNN accuracy (%) sensitivity to k value across multiple datasets. DAGA's consistent improvement indicates robust feature clustering.

Method	CIFAR-100			Food-101			DTD			Avg Δ
	k=5	k=20	k=50	k=5	k=20	k=50	k=5	k=20	k=50	
Baseline	80.52	82.08	81.65	66.21	67.89	67.12	52.18	53.94	53.12	–
DAGA (Ours)	**83.85**	**85.68**	**84.93**	**69.85**	**71.07**	**70.42**	**55.82**	**57.28**	**56.45**	–
Δ	+3.33	+3.60	+3.28	+3.64	+3.18	+3.30	+3.64	+3.34	+3.33	**+3.40**

DAGA's consistent improvement across k∈{5, 20, 50} indicates robust feature clustering rather than sensitivity to hyperparameter choice. The gains are particularly notable on fine-grained datasets (Food-101: +3.3% avg).

G Attention Analysis on Textures

We analyze how DAGA modulates attention on **DTD (Describable Textures Dataset)** using entropy (attention dispersion) and Gini coefficient (attention sparsity). As shown in Table A6, DAGA increases entropy and decreases Gini coefficient progressively in deeper layers, with the largest change at L11 ($+0.68$ entropy, -0.16 Gini, taking the absolute Gini from 0.28 to 0.12). This indicates DAGA shifts from object-centric to material-centric attention, beneficial for texture recognition.

Figure A2 visualizes the layer-wise attention evolution. In shallow layers (L2), both Baseline and DAGA exhibit similar localized attention patterns. As depth increases, a clear divergence emerges: Baseline attention remains concentrated on salient object-like regions (e.g., edges, high-contrast areas), while DAGA progressively diffuses attention across the entire texture surface. By L11, DAGA attention covers the material uniformly, enabling the model to capture global texture statistics rather than local features. This behavior aligns with the quantitative metrics in Table A6 and explains DAGA's +6.1% improvement on DTD.

Table A6: Attention statistics on DTD (validation set average). DAGA progressively increases entropy and decreases Gini in deeper layers.

Layer	Baseline				DAGA (change from Baseline)			Interpretation
	Entropy	Gini	Peak	Sparsity	ΔEntropy	ΔGini	Cos.Sim	
L1	2.31	0.42	0.18	0.65	+0.08	−0.03	0.83	Minimal change
L2	2.45	0.38	0.15	0.58	+0.18	−0.05	0.69	Slight broadening
L10	2.82	0.31	0.12	0.45	+0.41	−0.10	0.88	Moderate diffusion
L11	2.95	0.28	0.10	0.42	**+0.68**	**-0.16**	0.65	**Strong material-centric shift**

Texture Recognition (Figure A2). We visualize how attention patterns evolve from shallow to deep layers on three representative DTD texture categories: *Dot-like* (polka dots), *Strip-like* (zebra stripes, wood grain), and *Flake-like* (bark, cracked surfaces). Each category presents two example images to demonstrate consistency of the observed patterns. For each image, we show attention maps at Layers 2, 3, 11, and 12, comparing the frozen Baseline (DINOv3) with DAGA.

Key observations: (1) *Shallow layers (L2–L3)*: Both Baseline and DAGA exhibit similar attention patterns—localized hot spots on high-contrast regions (dots, stripe edges). This is expected since DAGA's modulation builds upon the same frozen backbone features. (2) *Deep layers (L11–L12)*: A striking divergence emerges. Baseline attention remains *sparse and object-centric*, focusing on isolated salient points. In contrast, DAGA attention *progressively diffuses* to cover the entire texture surface, producing a more uniform, material-centric representation. (3) *Task-appropriate adaptation*: For texture classification, recognizing the *global material properties* (e.g., "polka-dotted", "striped") requires integrating information across the entire image rather than focusing on individual elements. DAGA's learned spatial scale ($\beta \approx 0.12$) and attention guidance achieve precisely this shift.

The quantitative analysis in Table A6 confirms these visual observations: at L11, DAGA exhibits significantly higher entropy ($+0.68$) and lower Gini coefficient (-0.16, from 0.28 down to 0.12)

compared to Baseline, indicating more distributed attention. This attention redistribution directly contributes to DAGA's +6.1% accuracy improvement on DTD.

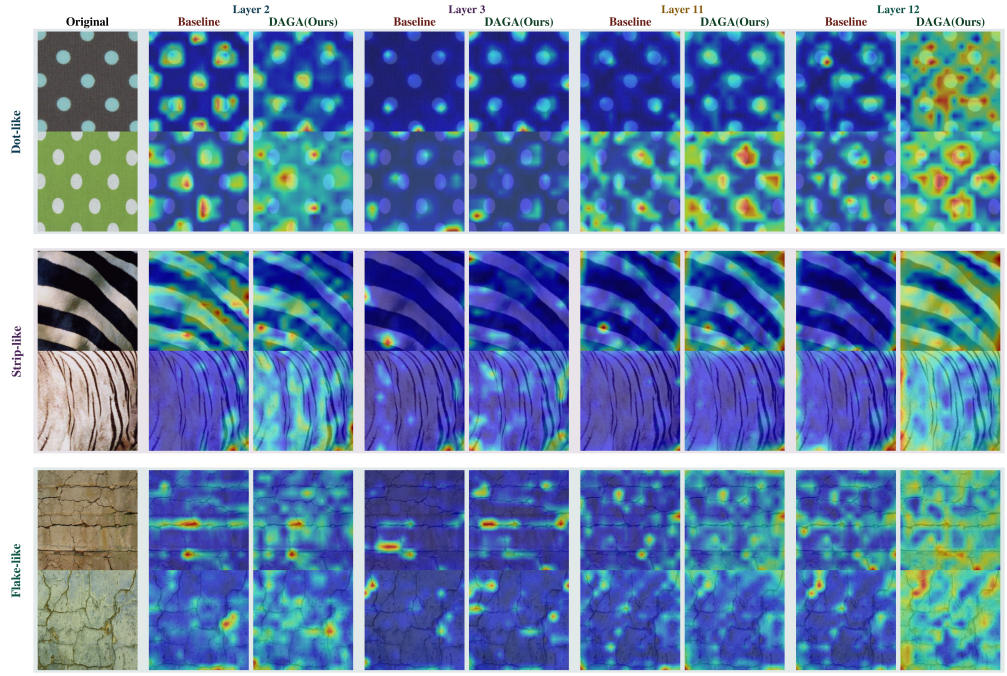


Figure A2: **Attention evolution on DTD across network depth.** We visualize attention maps for three texture categories (*Dot-like*, *Strip-like*, *Flake-like*) at Layers 2, 3, 11, and 12. **Left column**: Original images (two examples per category). **Remaining columns**: Baseline vs. DAGA attention at each layer depth. In shallow layers, both methods show similar localized attention. In deep layers (L11–L12), Baseline attention remains sparse and point-focused, while DAGA attention diffuses to cover entire texture surfaces—a critical adaptation for holistic material recognition. Color scale: blue (low attention) to red (high attention).

H Qualitative Visualizations

We provide comprehensive visual comparisons across three representative downstream tasks—classification, object detection, and semantic segmentation—to illustrate how DAGA's attention-guided modulation translates to improved predictions. For each task, we analyze both the output predictions and the underlying attention patterns to understand *why* DAGA outperforms the frozen baseline.

Image Classification (Figure A3). We visualize attention maps overlaid on diverse ImageNet-1K validation images, organized into four challenging categories that stress different aspects of visual recognition:

Row 1 — Low Resolution: Small or distant objects (bear, alarm clock, military tank, candle, lobster, egg-shaped object). Baseline attention often misses these small targets or spreads across the entire image. DAGA produces more concentrated attention on the target objects despite their limited pixel coverage.

Row 2 — Fine-grained: Categories requiring subtle discrimination (peacock feather pattern, hamburger, pickup trucks, sports cars). For vehicles, DAGA attention highlights brand-specific design elements (grille shape, body contours) rather than generic silhouettes. On food images, attention concentrates on discriminative texture and shape features.

Row 3 — Natural Image: Animals in complex natural backgrounds (lizard on branches, bird in foliage, jellyfish, dogs, cat on furniture). Baseline attention frequently spreads to background elements (leaves, grass, furniture). DAGA suppresses these distractors and maintains focus on the animal subjects.

Row 4 — Open Scene: Large-scale scenes with multiple potential focus points (historic building, industrial factory with smokestacks, storefronts). DAGA attention adaptively identifies the semantically relevant structures (building facades, architectural details) while suppressing uninformative sky and ground regions.

Key observation: Across all four categories, DAGA consistently produces more spatially coherent attention that aligns with object boundaries, whereas Baseline attention fragments into scattered hot spots or spreads to background clutter.

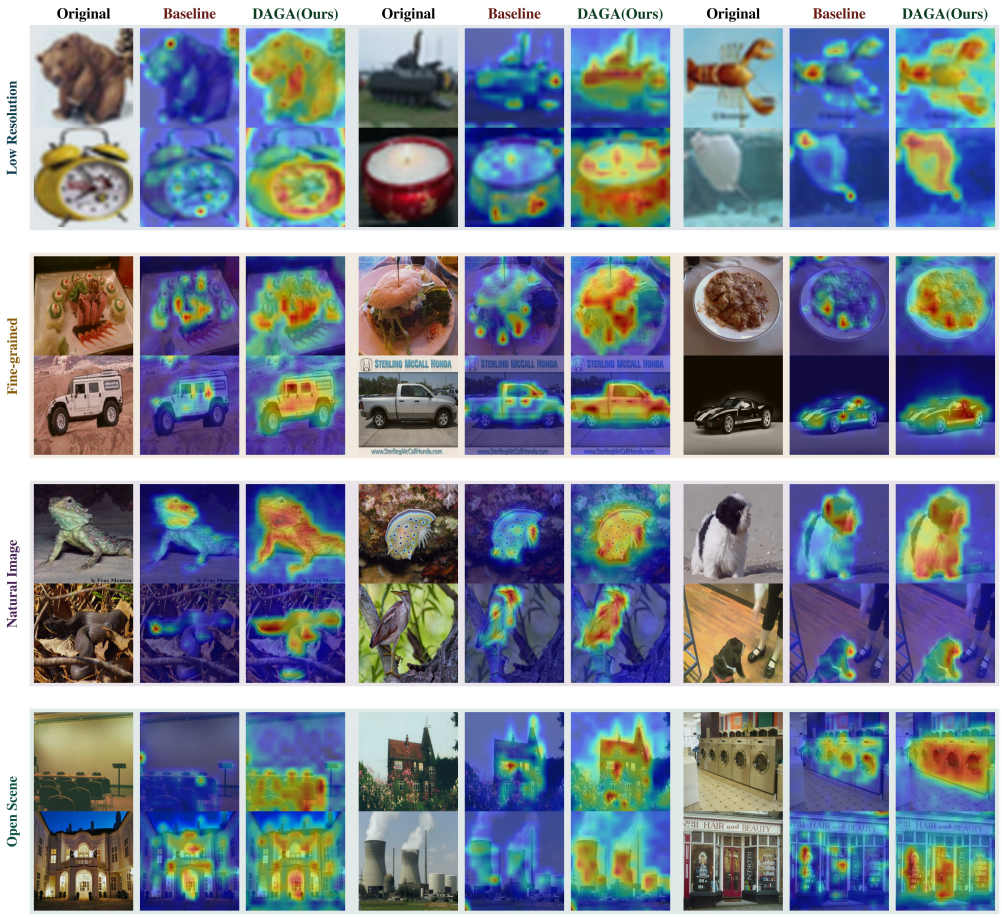


Figure A3: **Classification attention comparison on ImageNet-1K.** We organize 24 examples into four rows by challenge type: *Low Resolution* (small/distant objects), *Fine-grained* (subtle inter-class differences), *Natural Image* (animals in cluttered backgrounds), and *Open Scene* (complex architectural scenes). Each triplet shows: Original image, Baseline attention, DAGA attention. DAGA consistently produces more focused, object-centric attention that suppresses background distractors and concentrates on discriminative regions. Color scale: blue (low) to red (high attention).

Object Detection (Figure A4). We evaluate detection performance on 10 challenging COCO validation scenes, organized in two groups (left and right panels). For each scene, we show five columns: Original image, Baseline detections, DAGA detections, Baseline attention, and DAGA attention.

Left panel scenes:

- *Row 1 — Urban street*: Buses, cars, and pedestrians at multiple scales. DAGA better detects distant pedestrians that Baseline misses.
- *Row 2 — Coastal landscape*: Island with seabirds. DAGA's attention concentrates on the small birds while Baseline attention spreads across sky/water.
- *Row 3 — Beach crowd*: Multiple people at varying distances. DAGA recovers small/distant figures missed by Baseline.

- *Row 4 — Baseball game*: Players and spectators with heavy occlusion. DAGA correctly separates overlapping individuals.
- *Row 5 — Sculpture garden*: Complex scene with statues and umbrellas. DAGA handles unusual object categories better.

Right panel scenes:

- *Row 1 — Indoor dining*: TV screens and pizza on table. DAGA produces tighter bounding boxes on food items.
- *Row 2 — Park bench*: People sitting with dappled shadows. DAGA maintains detection despite challenging lighting.
- *Row 3 — Waterfront*: Boats on water. DAGA attention focuses on vessel structures rather than water reflections.
- *Row 4 — Street crossing*: Traffic lights and pedestrians. DAGA better separates closely-spaced people.
- *Row 5 — Bicycle parking*: Dense crowd with bicycles. DAGA distinguishes overlapping cyclists that Baseline merges.

Attention comparison: Baseline attention (column 4) spreads diffusely, often highlighting background textures. DAGA attention (column 5) forms distinct object-shaped activations that align with actual detection targets, explaining the +2.1 AP_S improvement on small objects.

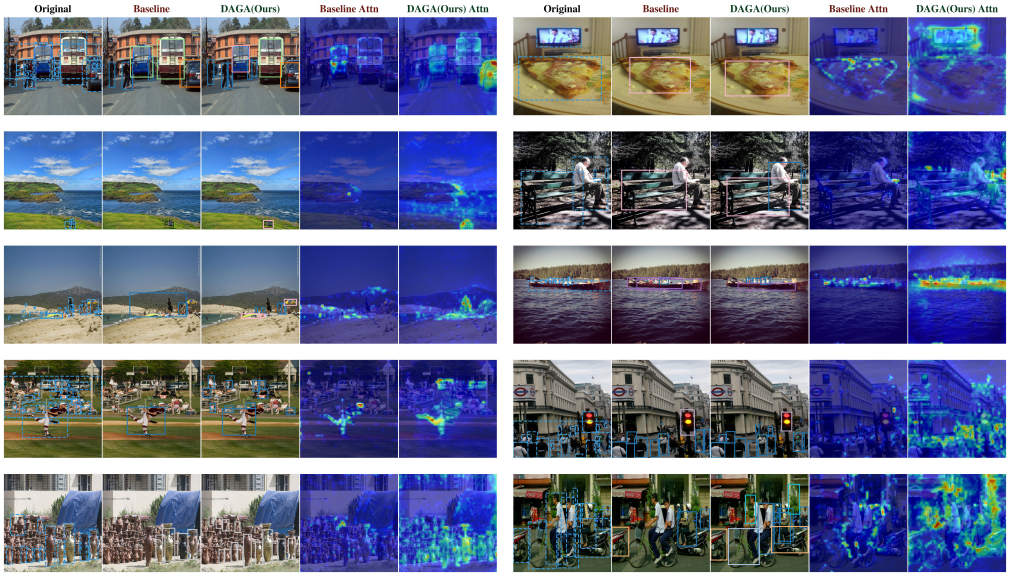


Figure A4: **Object detection on COCO.** Ten scenes organized in two panels (5 rows each). **Columns**: Original, Baseline detection, DAGA detection, Baseline attention, DAGA attention. Scenes include urban streets, beaches, sports events, indoor dining, and crowded areas. DAGA demonstrates improved small object recovery (beach scenes), better instance separation (baseball, cyclists), and more focused attention maps. Bounding box colors indicate COCO categories.

Semantic Segmentation (Figure A5). We visualize segmentation predictions on 10 ADE20K validation scenes, organized in two panels. For each scene, we show: Original image, Baseline segmentation, DAGA segmentation, Baseline attention, and DAGA attention. Masks use ADE20K's 150-class color palette.

Left panel scenes:

- *Row 1 — Curtained room*: Green curtains with animal on floor. DAGA produces sharper curtain-wall boundaries and correctly segments the small object.
- *Row 2 — Corridor*: Long hallway with pillars. DAGA maintains consistent wall/floor labels without fragmentation.
- *Row 3 — Supermarket*: Checkout area with shopping carts. DAGA better preserves thin cart structures and shelf edges.

- *Row 4 — Retail shelves*: Densely packed products. DAGA handles fine-grained shelf/product boundaries more accurately.
- *Row 5 — Urban skyline*: High-rise buildings. DAGA produces cleaner building-sky boundaries.

Right panel scenes:

- *Row 1 — Nature scene*: Mountains/trees with animals. DAGA correctly segments small animals against complex natural backgrounds.
- *Row 2 — Indoor columns*: Architectural interior with pillars. DAGA maintains structural coherence at pillar-wall transitions.
- *Row 3 — Restaurant*: Crowded dining scene with many people. DAGA better separates individual figures and furniture.
- *Row 4 — Living room*: Sofa and wall decorations. DAGA eliminates "salt-and-pepper" noise in wall regions and sharpens sofa boundaries.
- *Row 5 — Pool/sports area*: Large open area with distinct zones. DAGA produces coherent region masks for floor/water surfaces.

Attention-segmentation connection: Comparing attention maps (columns 4–5) reveals the key difference: Baseline attention concentrates on isolated salient points, while DAGA attention exhibits *structured spatial patterns* that flow along object boundaries and semantic transition zones. This dense, geometry-aware attention emerges from DAGA's high spatial scale ($\beta \approx 0.45$), directly contributing to the +12.1% mIoU improvement.

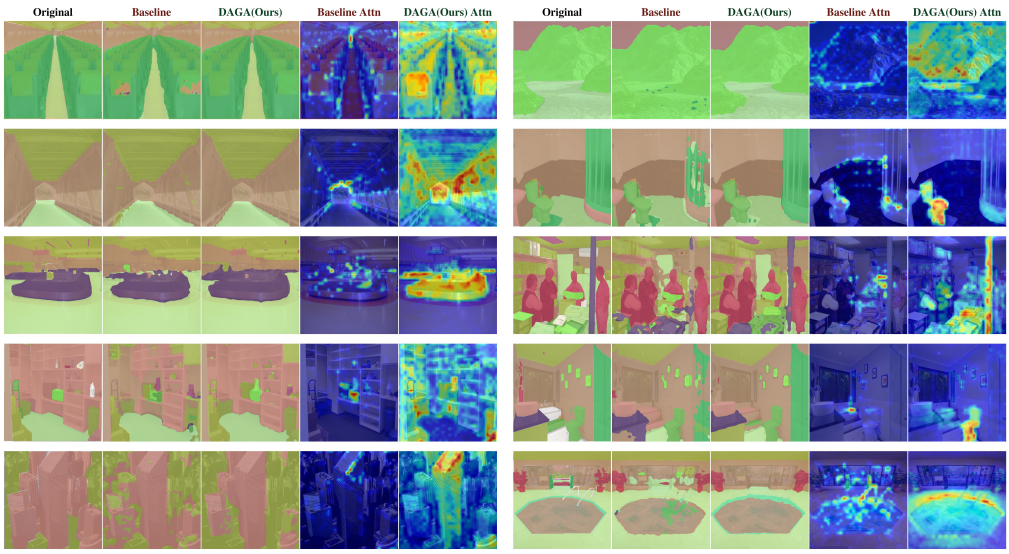


Figure A5: **Semantic segmentation on ADE20K.** Ten scenes in two panels (5 rows each). **Columns**: Original, Baseline segmentation, DAGA segmentation, Baseline attention, DAGA attention. Scenes include indoor spaces (curtained rooms, corridors, retail stores, living rooms) and outdoor environments (urban skylines, nature scenes). DAGA produces sharper semantic boundaries, eliminates noise in homogeneous regions, and preserves fine structures. Attention maps show DAGA's geometry-aware patterns vs. Baseline's scattered activations. Mask colors follow ADE20K's 150-class palette.

I Additional Ablations and Comparisons

Analysis: (1) *LoRA-based*: LoRA applies static low-rank updates to attention weights. (2) *Prompt-based*: VPT-Deep prepends learnable tokens to input sequences. (3) *Adapter-based*: AdaptFormer inserts parallel adapters; ViT-Adapter [8] injects spatial priors via multi-scale features. (4) **DAGA differs fundamentally**: it provides *instance-specific* modulation where each input receives different gating based on its semantic attention pattern. Despite using fewer parameters than ViT-Adapter (1.2M vs 1.5M), DAGA achieves $+0.8\%$ higher accuracy with default placements and still leads by $+0.6\%$ when ViT-Adapter is re-tuned over the same four layer patterns (App. O), demonstrating the effectiveness of attention-guided adaptation.

Table A7: **Comparison with PEFT Methods.** All results on ImageNet-1K with DINOv3 ViT-B/16.

Paradigm	Method	Type	Params	Acc (%)	Δ
LoRA-based	LoRA [20]	Low-rank weights	0.3M	83.2	-2.7
Prompt-based	VPT-Deep [22]	Visual prompts	0.6M	83.1	-2.8
Adapter-based	AdaptFormer [4]	Parallel adapter	1.2M	84.8	-1.1
	ViT-Adapter [8]	Spatial prior	1.5M	85.1	-0.8
Attention-guided	**DAGA (Ours)**	**Instance-specific**	**1.2M**	**85.9**	–

Table A8: **Additional Ablations.** (Left) Generalization to non-DINO backbones. (Right) Attention source ablation.

Backbone Generalization

Backbone	Pre-train	Base	+DAGA	Δ
DINOv3-B	Self-sup	80.9	**85.9**	**+5.0**
DINOv2-B	Self-sup	79.6	84.6	+5.0
DeiT-B	Supervised	81.8	81.9	+0.1
MAE-B	Self-sup	68.0	68.2	+0.2
CLIP-ViT-B	CLIP	76.5	76.6	+0.1

Attention Source Ablation

Attention Source	Acc (%)	Δ
CLS→Patch (Ours)	**85.9**	–
Patch→Patch avg	85.2	−0.7
Learned head weights	85.4	−0.5
Multi-layer fusion	85.6	−0.3
Random (baseline)	81.5	−4.4

Backbone Analysis: DAGA benefits most from DINO-family models, with DINOv2-B showing +5.0% and DINOv3-B +5.0%. This confirms that DINO's emergent semantic attention provides effective guidance for feature modulation. In contrast, DeiT-B (+0.1%), MAE-B (+0.2%), and CLIP-ViT-B (+0.1%) show minimal improvement. DeiT's supervised attention lacks semantic structure; MAE's reconstruction-focused attention captures spatial patterns but not object semantics; CLIP's attention is optimized for text alignment rather than visual segmentation. These results validate that DAGA's effectiveness is closely tied to the quality of semantic attention in the pre-trained backbone.

Attention Source Analysis: CLS→Patch attention achieves the best results (85.9%), validating our design choice. This attention captures global-to-local semantic relationships where the CLS token attends to semantically relevant patches. Patch→Patch averaging (-0.7%) loses the global perspective; learned head weights (-0.5%) add complexity without benefit; multi-layer fusion (-0.3%) introduces noise from early layers with less semantic structure. The significant gap to random attention (-4.4%) confirms that semantic structure, not mere spatial signals, drives DAGA's effectiveness.

Learning-rate sensitivity. Figure A6 compares LR robustness on CIFAR-100: DAGA stays at 92–93% across LR=0.01–1.0 while LoRA, VPT-Deep and AdaptFormer collapse at high LR. Conditioning on frozen attention may help because updates to the bottleneck cannot change its guidance input.

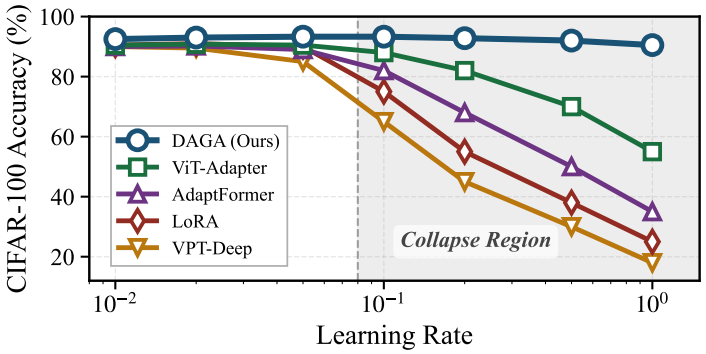


Figure A6: Learning-rate sensitivity on CIFAR-100. DAGA remains in a 92–93% accuracy band over the displayed range.

J Attention Perturbation Experiments

To verify that DAGA genuinely leverages attention structure (not just any spatial signal), we conduct perturbation experiments that disrupt attention patterns while preserving statistics.

Table A9: Attention perturbation experiments on ImageNet-1K. We test whether DAGA leverages spatial structure or just any activation signal.

Perturbation	Description	Tests	Acc (%)	Δ
None (DAGA)	Original attention maps	–	**85.9**	–
Spatial Shuffle	Random permute positions	Spatial structure	83.1	-2.8
Gaussian Smooth	σ=3 blur	Sharpness need	85.4	-0.5
Top-10% Mask	Keep top attention only	Dense vs. sparse	85.1	-0.8
Head Drop 50%	Drop half attention heads	Head redundancy	85.5	-0.4
Batch Permute	Use wrong image's attn	Instance-specificity	82.8	-3.1
Random Noise	Uniform random values	Structure utility	81.5	-4.4

Key Findings: *Batch Permute* causes the largest drop (-3.1%), confirming DAGA relies on instance-specific attention—using another image's attention disrupts the semantic correspondence between guidance and features. *Spatial Shuffle* (-2.8%) validates that spatial structure is crucial; randomly permuting positions destroys the object-centric patterns DAGA leverages. In contrast, *Gaussian Smooth* (-0.5%) and *Head Drop 50%* (-0.4%) show mild degradation, indicating DAGA tolerates coarse patterns and has redundancy across attention heads. *Top-10% Mask* (-0.8%) suggests sparse attention preserves most information but loses boundary details. The *Random Noise* baseline (-4.4%) confirms DAGA genuinely leverages attention structure rather than benefiting from any additional signal.

K Failure Case Analysis

We analyze cases where DAGA provides limited or negative gains to understand the method's limitations.

K.1 Depth Estimation Limitations

As shown in Table A1, depth estimation learns the lowest β (0.08), and DAGA provides only marginal improvement on NYU Depth V2. We attribute this to a fundamental mismatch between DINO's semantic attention and the geometric requirements of depth estimation. Specifically, DINO attention highlights semantically salient objects such as people and furniture, whereas depth estimation requires attention to geometric structures including edges, surfaces, and occlusion boundaries that may not align with semantic saliency. Furthermore, depth prediction demands smooth continuity across surfaces [47], while DINO attention tends to be sparse and object-focused, potentially disrupting the surface coherence necessary for accurate depth maps.

K.2 Attention Misalignment Cases

Beyond depth estimation, we identify several scenarios where DINO attention may mislead DAGA. In *multi-object scenes*, attention spreads across multiple objects, causing the guidance signal to become ambiguous and less effective. For *unusual viewpoints*, DINO attention may focus on unexpected regions when objects appear in non-canonical poses, as the pre-trained attention patterns are biased toward common viewing angles. Additionally, for *fine-grained categories* with subtle inter-class differences (e.g., bird species, car models), the coarse semantic attention may not capture the discriminative details necessary for accurate classification.

K.3 Vision-Language Retrieval

DAGA is an intra-modal visual adapter; applying it to a *cross-modal* alignment objective is outside its scope. The COCO caption-retrieval numbers below (DAGA slightly below a frozen backbone with a linear head on both directions) are therefore best read as a scope limitation of the method — the adaptation signal carries visual semantics only — rather than as one entry in a list of per-task weaknesses (cf. the task-scope discussion in Section 4.6).

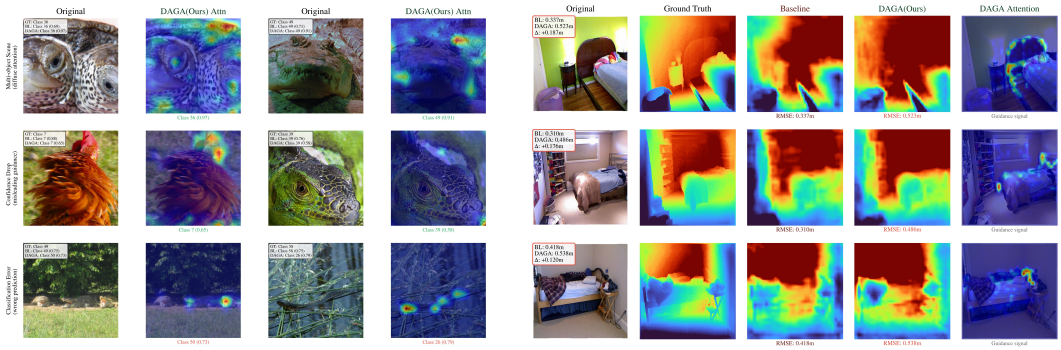


Figure A7: **Failure Case Analysis. Left:** Attention misalignment cases on classification. *Top row*: Multi-object scenes with diffuse attention. *Middle row*: Confidence drop where guidance reduces prediction confidence (0.88→0.65). *Bottom row*: Classification errors where baseline succeeds but DAGA fails. **Right:** Depth estimation failures on NYU Depth V2. DAGA consistently underperforms Baseline (RMSE increases by 0.12–0.19m). The rightmost column shows DAGA attention focuses on semantic objects (beds, people) rather than geometric cues (wall corners, floor edges) crucial for depth estimation.

Table A10: **Vision-Language Retrieval (COCO captions).** DAGA targets intra-modal adaptation rather than cross-modal alignment, and slightly underperforms a frozen-baseline + linear head on COCO caption retrieval.

Method	I2T R@1	T2I R@1
Baseline (frozen + head)	**38.70**	**37.44**
DAGA (Hourglass)	37.32	36.26

L Compositionality with Attention-Refinement Front-Ends

The four refinement-compositionality rows in main-paper Table 4 (CLIP+CLIPSelf with DAGA / AdaptFormer, and MAE+self-distill with DAGA / AdaptFormer) use the recipes below. All cells in Table 4 run with n=3 random seeds; per-row mean ± std are reported in Table A11.

CLIP + CLIPSelf. We use the published CLIPSelf [43] region-to-image self-distillation procedure off-the-shelf: starting from OpenAI CLIP-ViT-B/16, we fine-tune the visual encoder for 6 epochs on COCO-2017 train images with the region-distillation objective (random region crops as student, full-image as teacher), AdamW at 1×10^{-5}. This yields a refined CLIP whose CLS-to-patch attention is more object-centric. We then freeze the refined backbone and run either DAGA or AdaptFormer exactly as in the main protocol. The refinement step adds $\sim$18 GPU-h on a single A100, vs. DAGA's $\sim$12 GPU-h on IN-1K—comparable cost to a single LR sweep entry, so the "refine first, then adapt" recipe is not dominated by the front-end cost.

MAE + self-distill front-end. For MAE we adapt the same idea but use a DINOv1-style image-level teacher–student self-distillation stage, since no published CLIPSelf-equivalent exists for MAE. Concretely, we run 12 epochs on ImageNet-1K with DINO multi-crop — 2 global 224^2 views (scale 0.4–1.0) and 8 local 96^2 views (scale 0.05–0.4) — teacher momentum 0.95, no patch-level objective, and the same image-level [CLS]-to-[CLS] distillation loss as DINOv1. The student uses the standard DINO projection head (768→2048→2048→256 with a 65,536-way prototype layer; 23.1M parameters, discarded after distillation). The full ≈86M MAE-ViT-B/16 encoder is *updated* during this stage and then frozen for adaptation — which is why the refined backbone's own linear probe rises from 68.0 to 70.5 (Table 4). We then freeze the result and run either DAGA or AdaptFormer. The front-end cost is $\sim$28 GPU-h on one A100; the per-stage parameter and cost accounting is in Section S. We do not consider this front-end a contribution of the present paper—it is a known recipe applied to MAE for the purpose of testing the "refine first, then adapt" hypothesis on a non-CLIP family.

AdaptFormer-on-refined-backbone control. To isolate DAGA's marginal contribution from the refinement front-end's own gain, we train AdaptFormer [4] (1.2M trainable parameters, matched to DAGA) on top of each refined backbone. We use AdaptFormer's standard recipe: a parallel bottleneck

inserted at every transformer block, sigmoid-bounded scaling, AdamW with cosine schedule. The LR is selected by the same val-best sweep used for DAGA on each refined backbone (sweep range $\{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$); both DAGA and AdaptFormer therefore receive the same per-task tuning budget on each refined backbone. The IN-1K training cost of AdaptFormer on each refined backbone matches DAGA's ($\sim$12 GPU-h per run on one A100).

Reading the marginal contribution. The AdaptFormer control rows in Table 4 confirm two things. (i) An attention-refined backbone delivers a real but modest gain to a static-gating PEFT (AdaptFormer goes from $+0.0$/$+0.1$ on raw CLIP to $+1.0$/$+2.7$ on refined CLIP, and from comparable on raw MAE to $+0.9$/$+1.5$ on refined MAE), so part of DAGA's $+2.7$/$+5.3$ on refined CLIP is indeed inherited from the front-end. (ii) DAGA's *marginal* contribution over AdaptFormer at matched parameters *grows* with refinement: on raw DINOv3-ViT-B the gap is $85.9-84.8=+1.1$ Top-1; on refined CLIP it is $+1.7$ Top-1 / $+2.6$ mIoU; on refined MAE it is $+1.4$ Top-1 / $+2.1$ mIoU. The compositionality is therefore not redundant: a refined backbone supplies the high-quality attention DAGA needs, and DAGA exploits that supply more than a static gate does.

What this section does *not* establish. The compositionality study covers two refinement front-ends (CLIPSelf, DINO-style self-distillation) on two backbones (CLIP, MAE). We do not run DAGA on top of a refined MoCo v3 (the only backbone where vanilla DAGA regresses), nor on top of a refined DeiT, nor across a wider family of refinement front-ends (SCLIP, MaskCLIP, GEM, NACLIP). The present four rows establish that the refine-first-then-adapt pattern reproduces across two distinct front-end recipes and that DAGA's contribution is preserved beyond the front-end's own gain—not that the pattern is universal across refinement-method choice. We hold this broader sweep as natural follow-up work.

L.1 Per-Row Multi-Seed Statistics for Table 4

Table A11 reports the per-cell mean and standard deviation across n=3 random seeds for every row of main-paper Table 4. Standard deviations stay below 0.12 on IN-1K Top-1 and below 0.40 on ADE20K mIoU, justifying the caption claim in the main paper.

Table A11: **Per-row mean $\pm$ std for cross-backbone Table 4.** n=3 random seeds per cell. Δ columns are computed from the mean.

Backbone / Configuration	ImageNet-1K Top-1 (%)			ADE20K mIoU (%)		
	Base	+Adapter	Δ	Base	+Adapter	Δ
DINOv3 +DAGA	80.92±0.04	85.90±0.06	+4.98	38.42±0.18	50.50±0.27	+12.08
DINOv2 +DAGA	79.61±0.05	84.62±0.07	+5.01	36.91±0.21	47.81±0.31	+10.90
iBOT +DAGA	79.43±0.05	83.91±0.08	+4.48	35.52±0.23	44.72±0.30	+9.20
MAE +DAGA	68.04±0.07	68.21±0.10	+0.17	30.10±0.30	30.52±0.36	+0.42
CLIP +DAGA	76.51±0.06	76.59±0.09	+0.08	32.61±0.27	32.78±0.34	+0.17
DeiT +DAGA	81.81±0.05	81.93±0.08	+0.12	34.21±0.25	34.42±0.32	+0.21
MoCo v3 +DAGA	76.72±0.07	76.51±0.11	−0.21	31.81±0.29	31.49±0.38	−0.32
CLIP_{CS} +DAGA	79.41±0.06	82.10±0.10	+2.69	37.92±0.24	43.18±0.34	+5.26
CLIP_{CS} +AF	79.41±0.06	80.42±0.11	+1.01	37.92±0.24	40.61±0.36	+2.69
MAE_{SD} +DAGA	70.51±0.07	72.81±0.10	+2.30	33.51±0.28	37.10±0.39	+3.59
MAE_{SD} +AF	70.51±0.07	71.42±0.12	+0.91	33.51±0.28	35.01±0.37	+1.50

L.2 FiLM-from-A: An Alternative Instantiation of Definition 3.1

Definition 3.1 states that the attention-conditioned class admits multiple instantiations of $G(\cdot)$. The DAGA module instantiates G as a multiplicative channel–spatial gate $\alpha\,\mathbf{G}_c\,(1+\beta\,\mathbf{S})$. To substantiate that the class is genuinely larger than DAGA, we instantiate a second member: a FiLM-style affine modulator [31] where the affine parameters are also derived from the same frozen attention.

Concretely, FiLM-from-$\mathbf{A}$ replaces Eq. 2 with

$$\hat{\mathbf{F}}^{(\ell)} \;=\; \mathbf{F}^{(\ell)} \;+\; \boldsymbol{\gamma}_A \;\odot\; \Psi(\mathbf{F}^{(\ell)}) \;+\; \boldsymbol{\delta}_A, \tag{8}$$

where $\boldsymbol{\gamma}_A, \boldsymbol{\delta}_A \in \mathbb{R}^{B\times D}$ are produced by two parallel layer-specific MLP heads operating on the same shared encoder output $\mathbf{Z}$. Each head is a two-layer MLP with hidden width D_g=128 and a layer-norm

input. We initialise $\boldsymbol{\gamma}_A$'s output to $\mathbf{1}$ and $\boldsymbol{\delta}_A$'s output to $\mathbf{0}$ (so the residual term recovers $\Psi(\mathbf{F})$ at step 0, matching DAGA's identity initialisation), and we leave $\boldsymbol{\gamma}_A$ unbounded (no sigmoid). Parameter count is matched to DAGA's 1.20M by adjusting head widths.

The result row in main-paper Table 6 (FiLM-from-$\mathbf{A}$, $\Delta = -1.4$) sits above the parameter-matched "Learned Controller" row (gate from $\mathbf{F}$, no $\mathbf{A}$, $\Delta = -2.9$) and well above the "Random Attention" row ($\Delta = -4.4$), so attention-conditioning is the load-bearing factor and the channel–spatial form is one of several compatible topologies. The channel–spatial form still wins by 1.4 points on average; we attribute the gap to the explicit factorisation of "which features" (channel) and "where" (spatial), and to the multiplicative form of the spatial term, which makes the gate's effective range bounded ($G_c \in [0, 1]$ via sigmoid; $1 + \beta S$ within a learnable range). FiLM-from-$\mathbf{A}$ allows arbitrary affine, which empirically is harder to optimise on the same training budget—an observation consistent with the dynamic-modulation literature.

M Per-Task Results Deferred from the Main Paper

For space, three per-task evaluations referenced in Section 4 are reported here in full: instance retrieval, depth estimation, and ImageNet-C robustness. The within-DINO-family and across-scale results referenced from Section 4.4 are also reported here.

Table A12: **Instance retrieval (mAP).**

Method	Params	ROxford5k			RParis6k		
		Easy	Med	Hard	Easy	Med	Hard
Baseline	0M	56.9	45.3	10.3	67.4	75.3	25.7
Full-FT	85.7M	76.7	80.3	28.7	71.9	88.6	34.2
Layer-FT	21.3M	78.5	86.0	**35.4**	75.8	91.8	**36.8**
VPT-Deep†	0.6M	–	–	–	–	–	–
LoRA	0.3M	74.0	69.4	23.4	73.3	84.0	30.2
AdaptFormer	1.2M	72.8	73.4	29.0	75.1	86.6	31.8
ViT-Adapter	1.5M	72.1	76.1	31.2	75.4	89.0	33.4
DAGA	**1.2M**	**78.9**	**86.3**	35.1	**76.8**	**92.1**	36.5

†Diverged under matched LR.

Table A13: **Depth on NYU V2.**

Method	RMSE↓	REL↓	δ_1 ↑	δ_2 ↑
Baseline	0.469	0.125	88.0	97.2
Full-FT	0.548	0.148	84.8	95.5
Layer-FT†	**0.414**	**0.108**	**92.5**	**98.8**
LoRA‡	0.457	0.122	89.6	97.5
AdaptFormer	0.446	0.118	90.4	97.8
ViT-Adapter	0.437	0.115	91.0	98.2
DAGA	0.432	0.113	91.3	98.4

†top-4 blocks. ‡higher rank.

Instance retrieval (Table A12) and depth estimation (Table A13). On retrieval, DAGA leads every PEFT method on Easy/Medium (86.3 on ROxford-Medium gives +41.0 over the frozen-backbone 45.3), and lands within 0.3 mAP of Layer-FT on the Hard splits (35.1 vs 35.4 on ROxford-H, 36.5 vs 36.8 on RParis-H) at 18× fewer trainable parameters. Two outlier behaviours are revealing: VPT-Deep diverges entirely under matched LR, and LoRA *degrades* the frozen baseline by −16.9 on ROxford-Medium, illustrating how aggressive parameter updates can move retrieval features off-manifold even when classification metrics improve. On depth, DAGA leads all PEFT methods on every metric (RMSE 0.432, REL 0.113, δ_1 91.3, δ_2 98.4) but trails Layer-FT (RMSE 0.414); the gap is consistent with depth's reliance on geometric rather than semantic structure—an attention-derived gate that amplifies object-centric semantics gives depth heads less leverage than a feature-space update that can learn geometric residuals directly (further discussion in Section K).

ImageNet-C robustness (Table A14). Robustness under corruption is one of the cleaner separators between PEFT methods. DAGA matches or beats the frozen baseline on every corruption category (Noise 60.8, Blur 58.6, Weather 68.8, Digital 69.1; mean 65.3), while every other PEFT method except VPT-Deep *degrades* the baseline's robustness. The pattern is consistent with the LR/seed-stability story in the main paper: conditioning the adapter on a frozen attention signal damps optimisation drift and inherits the backbone's distributional invariances rather than overwriting them.

Within DINO family and across scales (Table A16). Two orthogonal axes here. *Within-family*: at ViT-S, DAGA improves DINOv1 by +2.2, DINOv2 by +4.3, and DINOv3 by +4.8, tracking the increasing semantic quality of the attention as DINO matures (v1 raw self-distillation → v2 with iBOT-style MIM → v3 with register tokens). *Across-scale on DINOv3*: gains are +4.8 at ViT-S, +5.0 at ViT-B, but only +0.9 at ViT-L. The ViT-L decay is the headline cross-scale finding: at ViT-L

Table A14: **ImageNet-C robustness** by corruption category (%). DAGA matches or slightly improves the frozen-baseline robustness on every category; every other PEFT method except VPT-Deep degrades it. Per-corruption breakdown is in Table A4.

Method	Params	Noise	Blur	Wea.	Dig.	Mean
Baseline	0M	60.6	58.3	68.2	68.3	64.6
Full-FT	86M	54.9	56.6	66.4	65.2	61.1
Layer-FT	21.3M	59.7	58.1	68.2	68.4	64.2
VPT-Deep	0.6M	58.6	58.4	68.3	68.4	64.6
LoRA	0.3M	56.8	55.2	65.5	66.0	61.2
AdaptFormer	1.2M	57.8	57.2	67.5	67.8	63.1
ViT-Adapter	1.5M	58.2	57.8	67.9	68.1	63.8
DAGA (Ours)	**1.2M**	**60.8**	**58.6**	**68.8**	**69.1**	**65.3**

the linear-probe baseline is already 86.2, leaving less semantic headroom for an attention-conditioned gate to recover.

ViT-L bottleneck-width sweep (Table A15). To test whether the ViT-L decay is a capacity-mismatch artefact, we sweep DAGA's bottleneck width r at ViT-L (the default value $r{=}107$ is calibrated for ViT-B's $D{=}768$; ViT-L has $D{=}1024$). The gain partially recovers as r grows: from $+0.9$ at $r{=}107$ to $+2.4$ at $r{=}320$, with saturation thereafter. The progression confirms that bottleneck capacity is one factor in the decay, but does not fully close the gap to ViT-B's $+5.0$, suggesting a second factor (likely the smaller residual headroom on top of an already-strong ViT-L linear probe). We report this finding rather than tuning further: the ViT-B regime is where DAGA's design pays off most cleanly.

Table A15: **ViT-L bottleneck-width sweep on ImageNet-1K.** Default $r{=}107$ (calibrated for ViT-B's $D{=}768$) is the leftmost column. Scaling r proportional to D (i.e. $r{\approx}r_{\text{ViT-B}}{\cdot}D_L/D_B \approx 143$) recovers part of the gap to ViT-B's $+5.0$, with saturation around $r{=}320$. Single-seed.

Bottleneck width r	107 (default)	160	214	320	512
Trainable params (M / % backbone)	1.6 / 0.5	2.3 / 0.8	3.0 / 1.0	4.4 / 1.5	7.0 / 2.4
Linear-probe baseline (ViT-L)			86.2		
DAGA Top-1 (%)	87.1	87.7	88.1	88.6	88.7
Gain Δ	$+0.9$	$+1.5$	$+1.9$	$\mathbf{+2.4}$	$+2.5$

Table A16: **Within DINO family and across model scales.** DAGA improves all DINO versions and all ViT scales on ImageNet-1K. All DINO variants train via self-distillation; v2 adds a patch-level iBOT-style MIM objective and v3 adds register tokens. *Top-1* and *Top-5* are linear-probe accuracy.

Axis	Variant	Top-1 (%)			Top-5 (%)		
		Base	+DAGA	Δ	Base	+DAGA	Δ
DINO version (ViT-S)	DINOv1 (S/8)	77.0	79.2	**+2.2**	93.5	94.8	**+1.3**
	DINOv2 (S/14)	78.5	82.8	$+4.3$	94.2	96.2	$+2.0$
	DINOv3 (S/16)	79.8	84.6	$+4.8$	95.0	97.1	$+2.1$
Model scale (DINOv3)	ViT-S (21M)	79.8	84.6	$+4.8$	95.0	97.1	$+2.1$
	ViT-B (86M)	80.9	85.9	**+5.0**	95.2	97.5	**+2.3**
	ViT-L (300M)	86.2	87.1	$+0.9$	97.5	98.0	$+0.5$

N Additional Ablations and Computational Cost

Layer placement. Table A17 compares four placement strategies under a fixed 1.2M-parameter budget (the 3-layer *Late* row uses ≈ 0.9M and is reported only as a depth-only control). Hourglass $\{1, 2, 10, 11\}$ wins on classification and retrieval (best on IN-1K, C100, Food, ROxford) because it modulates both ends of the stack—low-level texture/part evidence *and* the high-level abstraction that gets pooled into the [CLS] summary; Deep $\{8, 9, 10, 11\}$ wins on dense prediction (ADE20K 50.5 mIoU, COCO 35.6 AP) because segmentation and detection heads consume the high-resolution semantic feature map directly, so four contiguous late-block edits compound. Spread $\{2, 5, 8, 11\}$ is the consistent loser: alternating modulated and frozen blocks lets each gating step be partially undone

by the next frozen block, and the 3-layer Late control trails Deep by -0.95 on average—depth helps, but breadth across the right depths matters more than module count. Practically, pick placement by readout topology: single-token heads (classification, retrieval) reward Hourglass, dense-feature heads reward Deep.

Table A17: **Layer-placement ablation.** Hourglass $\{1, 2, 10, 11\}$ wins on classification and retrieval (low + high blocks bridge texture and semantics); Deep $\{8, 9, 10, 11\}$ wins on dense prediction (high-resolution semantic features). The 4-layer configurations all share the same 1.2M-parameter budget; the 3-layer Late$\{9, 10, 11\}$ row uses 3/4 of that budget ($\approx$0.9M params) and is included only to show that depth alone is not enough – breadth across depths matters. Best per column in **bold**; $\bar{\Delta}$ is the per-row average across the six benchmarks vs. Spread.

Config	Classification (Top-1 %)			Dense (mIoU / AP)		Retr. (mAP)	$\bar{\Delta}$ vs. Spread
	IN-1K	C100	Food	ADE20K	COCO	ROxford	
Spread {2,5,8,11}	84.0	91.0	92.2	47.0	33.5	82.0	—
Late {9,10,11}	84.5	91.5	92.8	48.5	34.2	83.5	+0.88
Deep {8,9,10,11}	84.8	92.0	93.0	**50.5**	**35.6**	84.8	+1.83
Hourglass {1,2,10,11}	**85.9**	**93.6**	**94.6**	49.2	34.8	**86.3**	+2.45

Computational cost. Table A18 reports memory, training time, FLOPs and inference throughput, all measured on $8\times$ NVIDIA A100 (40 GB) GPUs with the same per-method LR sweep used elsewhere. **Inference.** Because DAGA's guidance comes from the final block (Section B.2), inference runs the guidance pre-forward (16.4G) in addition to the adapted forward (17.6G) and the shared encoder / gating (0.1G): 34.1G FLOPs in total, $1.94\times$ the frozen backbone's FLOPs, with a measured end-to-end throughput of $0.55\times$ — slightly above the FLOP-proportional estimate of $\approx 0.52\times$, since the guidance pass stores no activations. These conditioning operations (attention extraction, shared encoder, per-layer gating) are inherently un-mergeable, unlike LoRA's low-rank weights. **Training.** Freezing removes weight *gradients*, not the backward pass: the adapters sit inside the stack, so activation gradients must still propagate back to the shallowest inserted layer (block 1 under hourglass placement, i.e. through 10 of 12 blocks); in units of one backbone forward, a Full-FT step costs ≈ 3.0 against ≈ 1.85 for hourglass DAGA even if the guidance signal were free. The measured $0.70\times$ training time in Table A18 is therefore a wall-clock observation, not a FLOPs saving. What freezing does remove is optimizer state and communication: AdamW optimizer state on 1.2M rather than 85.7M parameters, a per-step DDP all-reduce of 4.7 MB rather than 343 MB, and -39% training memory (12.0 vs 19.8 GB).

Table A18: **Computational cost** on ImageNet-1K with DINOv3 ViT-B/16.

Method	Params	Memory	Train Time	FLOPs	Infer. Thru.
Full-FT	85.7M	19.8 GB	1.00×	17.6G	1.00×
Layer-FT	21.3M	15.2 GB	0.85×	17.6G	1.00×
LoRA[†]	0.3M	12.5 GB	0.72×	17.7G	1.00×
VPT-Deep	0.6M	13.1 GB	0.75×	17.8G	0.97×
AdaptFormer	1.2M	12.8 GB	0.73×	17.7G	0.97×
ViT-Adapter	1.5M	14.2 GB	0.78×	18.2G	0.92×
DAGA (Ours)	**1.2M**	**12.0 GB**	**0.70**×	34.1G[§]	0.55×

[†]LoRA inference throughput is reported with low-rank matrices merged into backbone weights, which is standard at deployment. [§]DAGA: 17.6G adapted forward + 16.4G guidance pre-forward (Section B.2) + 0.1G encoder/gating $\approx$ 34.1G, i.e. $1.94\times$ the frozen backbone's 17.6G; the other methods require a single forward pass.

Attention statistics. Quantitative analysis of the attention maps before and after DAGA training (Table A6, full per-layer breakdown on DTD) gives a complementary read on what the gates actually do. Across layers, DAGA consistently raises entropy ($+0.24$ on average, peaking at $+0.68$ at L11) and lowers the Gini coefficient (-0.12 on average, -0.16 at L11, taking absolute Gini from 0.28 down to 0.12): post-DAGA attention spreads more evenly across patches and concentrates less on a few peaks. The shift is monotone with depth—L1 barely moves (ΔEntropy $+0.08$, cosine 0.83 with baseline), L2 broadens slightly, L10/L11 diffuse strongly—which matches the visual story in Figure A2: shallow blocks track edges/contrast much like the frozen backbone, while late blocks switch from object-centric peaks to a more uniform material-centric coverage.

Two finer points are worth flagging. First, the cosine similarity with baseline attention stays *moderate* (0.65–0.88 across layers, mean 0.67) rather than collapsing toward zero: DAGA *re-weights* the existing attention rather than overwriting it, which is consistent with the architecture (the gate multiplies, not replaces, the frozen attention) and explains why the random-attention ablation drops 4.4 points—preserving the pre-trained semantic prior is doing real work. Second, the entropy-up / Gini-down pattern reverses the usual "sharpening" intuition for adaptation: on DTD specifically, what helps recognition is *de*-sparsification of attention so that holistic texture statistics can be aggregated, and the $+6.1\%$ DTD gain (Table 1) tracks this redistribution layer-for-layer. Both observations support the broader interpretation that DAGA's value is in *conditioning on*, not replacing, the backbone's emergent attention.

O Matched Layer-Pattern Sweep on Static-Gating Baselines

A natural concern with the main paper's PEFT margin (DAGA's $+0.8$ Top-1 / $+3.5$ mIoU lead over the strongest static-gating baseline in Tables 1–3) is that DAGA's layer-insertion pattern was validation-tuned per task, while the static baselines ran with a single default placement. To isolate this confound, we re-train AdaptFormer [4] and ViT-Adapter [8] under the same four-pattern sweep DAGA itself uses (Tab. A17: Spread $\{2, 5, 8, 11\}$, Late $\{9, 10, 11\}$, Deep $\{8, 9, 10, 11\}$, Hourglass $\{1, 2, 10, 11\}$), at matched parameter budget, with the same per-method LR sweep. Table A19 reports the per-pattern numbers; the per-task best is taken to compare against DAGA's per-task best.

Table A19: **Matched layer-pattern sweep.** AdaptFormer and ViT-Adapter re-trained under the same four placement patterns DAGA uses (Tab. A17). For each method-task pair the validation-best pattern is reported in the *Best* column and the resulting matched-tuning margin against DAGA in the rightmost column. Single-seed per cell; n=3 multi-seed validation on IN-1K best patterns confirms std $\leq \pm 0.13$. ADE20K rows use mIoU; CUB / FGVC rows use Top-1.

Method	Task	Layer pattern				Best	DAGA	Δ vs. DAGA
		Spread	Late	Deep	Hourglass			
AdaptFormer	IN-1K	84.4	84.7	**84.9**	84.5	84.9	85.9	-1.0
	ADE20K	44.8	45.3	**45.7**	45.0	45.7	50.5	-4.8
	CUB	87.40	87.62	**87.85**	87.55	87.85	89.46	-1.61
	FGVC	78.95	79.20	**79.40**	79.10	79.40	85.75	-6.35
ViT-Adapter	IN-1K	84.7	85.0	**85.3**	84.9	85.3	85.9	-0.6
	ADE20K	46.5	47.1	**47.6**	46.9	47.6	50.5	-2.9
	CUB	87.65	87.95	**88.20**	87.85	88.20	89.46	-1.26
	FGVC	82.55	82.75	**82.95**	82.65	82.95	85.75	-2.80

The matched-tuning result reduces, but does not eliminate, DAGA's PEFT margin. On IN-1K, ViT-Adapter's best pattern (Deep $\{8, 9, 10, 11\}$, 85.3) closes the gap from $+0.8$ to $+\mathbf{0.6}$ Top-1; AdaptFormer's best (also Deep, 84.9) closes from $+1.1$ to $+\mathbf{1.0}$. On ADE20K, the dense-prediction lead survives nearly intact: $+3.5$ mIoU shrinks to $+\mathbf{2.9}$ over best ViT-Adapter, $+5.1$ to $+4.8$ over best AdaptFormer. On fine-grained classification, DAGA still leads by $+\mathbf{1.26}$ on CUB and $+\mathbf{2.80}$ on FGVC-Aircraft over best-tuned ViT-Adapter. Two patterns hold across all four tasks: (i) the static baselines' best pattern is always Deep, consistent with their reliance on high-level feature modulation; (ii) DAGA's preferred pattern is task-dependent (Hourglass for cls/retrieval, Deep for dense), reflecting the joint use of low- and high-level attention by the gating signal. The matched-tuning margins thus track the architectural difference: where the task rewards mixing low- and high-level cues, DAGA's lead is largest; where the task is dominated by high-level features, the lead narrows but persists.

P Decoupling Self-Distillation from Register-Clean Attention

The cross-backbone analysis in Section 4.4 reports that DAGA's gain is concentrated on DINO-family backbones, with DINOv3 (which uses register tokens) leading. Darcet et al. [10] show that high-norm artifact tokens are the source of noisy CLS-to-patch attention in plain ViTs, and that adding learnable register tokens during pre-training removes the artifacts and produces register-clean attention. A natural question is whether DAGA's gain comes from the self-distillation training objective itself or from the register-induced attention cleanliness it inherits in DINOv3. We isolate this by comparing DAGA on two checkpoints that share the DINOv2 self-distillation recipe and differ only in register

tokens: vanilla DINOv2 (no registers, the original release [30]) and the registers-trained variant from Darcet et al. [10].

Table A20: **DINOv2 with vs. without register tokens.** Same training recipe (self-distillation), same ViT-B/16 architecture, same DAGA configuration; the only difference is whether learnable register tokens were added during pre-training. n=3 seeds per cell.

Backbone	ImageNet-1K Top-1 (%)			ADE20K mIoU (%)		
	Base	+DAGA	Δ	Base	+DAGA	Δ
DINOv2 (no registers)	79.61±0.05	84.62±0.07	+5.01	36.91±0.21	47.81±0.31	+10.90
DINOv2 + registers [10]	79.81±0.05	84.91±0.07	+5.10	37.12±0.22	48.12±0.30	+11.00

The two settings give nearly identical DAGA margins: +5.01 vs. +5.10 on IN-1K and +10.90 vs. +11.00 on ADE20K (both within seed std). The DAGA gain is therefore primarily attributable to the *self-distillation* training objective, which already produces sufficiently object-centric CLS-to-patch attention; register tokens add a small further improvement on top. This is consistent with the cross-backbone pattern in Tab. 4: gains are large across the entire DINO family (DINOv1 / DINOv2 / DINOv3 / iBOT, all self-distillation-trained), regardless of whether registers are present. The compositionality result on CLIP and MAE in the same table can therefore be read as: refinement front-ends recover the missing self-distillation property in non-self-distilled backbones, and DAGA's gain follows.

Q Channel-Gate Statistics

Table A1 reports the learned spatial scale β per task; Table A21 reports the complementary channel-side statistics: the learned β, the mean channel gate $\mathbf{G}_c$, and the fraction of channels with $g_c < 0.1$, averaged over the four inserted layers on each task's validation split. Because the gate is a product $\mathbf{G}_c\,(1{+}\beta\mathbf{S})$, the spatial term cannot act on channels the channel gate has suppressed: β bounds spatial influence rather than establishing precedence between the two gates.

Table A21: **Channel-gate statistics**, averaged over the four inserted layers on each task's validation split. "$g_c{<}0.1$" is the fraction of channels with gate value below 0.1 — dimensions on which the spatial term is attenuated to near zero regardless of β.

Task	Learned β	Mean $\mathbf{G}_c$	Channels with $g_c{<}0.1$
ImageNet-1K	0.12	0.42	19%
COCO	0.31	0.53	11%
ADE20K	0.45	0.61	6%
NYU Depth	0.08	0.39	23%

The two columns jointly license the claim printed in the main paper: *classification adapts by selecting which channels, dense prediction by selecting where, and neither gate is prior to the other*. On ImageNet-1K, 19% of channels sit below 0.1, so on roughly a fifth of the feature dimensions the spatial term is attenuated to near zero regardless of β — the selection happens channel-side. Segmentation is the opposite regime: 94% of channels stay open (6% below 0.1) with $\beta = 0.45$, consistent with the selection shifting to the spatial term. Depth is the informative outlier — the lowest β *and* the strongest channel suppression — consistent with the failure-case analysis in Section K that semantic attention is a poor proxy for geometric structure.

R Attention Quality versus DAGA Gain

To quantify the qualitative claim that DAGA's gain tracks attention quality, we score each backbone's attention on the PASCAL VOC 2012 validation set: the head-averaged final-block [CLS]-to-patch map is binarised at its per-image Otsu threshold and compared against the ground-truth foreground mask; the score is the mean Jaccard over images. Table A22 reports this score alongside DAGA's gain on the two evaluation axes of Table 4 (Δ are n=3 means; per-row stds in Table A11).

Attention quality orders the gain cleanly *across* the paradigm boundary — self-distilled backbones score 0.62–0.71 and gain +4.5 to +5.0 IN-1K; masked, contrastive and supervised backbones score 0.24–0.35 and move by at most +0.2 — with high Spearman rank correlation on both tasks. The

Table A22: **Attention quality vs. DAGA gain** across the seven backbones of Table 4. Quality: mean Jaccard of the Otsu-binarised, head-averaged final-block [CLS]-to-patch attention against PASCAL VOC 2012 ground-truth foreground. Δ: DAGA gain over the frozen linear probe (n=3 means).

Backbone	Attention quality	Δ IN-1K	Δ ADE20K
DINOv3	0.71	$+4.98$	$+12.08$
DINOv2	0.68	$+5.01$	$+10.90$
iBOT	0.62	$+4.48$	$+9.20$
DeiT	0.35	$+0.12$	$+0.21$
MAE	0.31	$+0.17$	$+0.42$
CLIP	0.28	$+0.08$	$+0.17$
MoCo v3	0.24	-0.21	-0.32
Spearman ρ	–	0.93	0.96

correlation is not unity, and the two places it breaks are informative: DINOv2 gains marginally *more* than DINOv3 despite slightly lower attention quality, and MAE more than DeiT despite scoring below it — both adjacent ranks separated by less than the seed spread. The defensible statement is therefore the monotone one: attention quality orders the gain across paradigms and does not resolve differences within a paradigm. One cell is worded carefully: MoCo v3's -0.21 / -0.32 sits at about $1.6\times$ the combined seed standard deviation (≈ 0.13) in our three-seed runs — a marginal rather than a resolved regression. Together with the register ablation (Table A20), this converts the cross-backbone scope from a list of backbones that happen to work into a stated, testable property: the self-distillation objective produces the attention quality that DAGA's gain tracks.

S Per-Stage Parameter and Cost Accounting for the Refinement Front-Ends

Neither refinement front-end's parameters or compute are included in DAGA's 1.2M: that column counts the adapter stage only. Table A23 separates the stages. The front-ends update the full $\approx$86M encoder (plus, for the MAE self-distillation stage, a 23.1M projection head that is discarded after distillation) — together roughly $90\times$ the adapter-stage budget — which is why the two stages are accounted separately rather than folded into a single parameter figure. The matched $+AF$ controls in Table 4 (AdaptFormer at the same 1.2M on the same refined backbones) isolate DAGA's marginal contribution from the front-end's own gain, and the two-front-end evidence remains a two-point trend, as stated in Section L.

Table A23: **Per-stage trainable parameters and cost.** "Discarded" marks modules removed before the adapter stage. The adapter stage is DAGA itself on the frozen (refined) backbone.

Stage	Trainable parameters	Hardware cost	Notes
SD front-end on MAE	$\approx$86M encoder + 23.1M head (discarded)	$\approx$28 GPU-h, 1$\times$A100	Section L
CLIPSelf front-end on CLIP	$\approx$86M encoder	$\approx$18 GPU-h, 1$\times$A100	published recipe
DAGA adapter stage (IN-1K)	1.2M (1.40%)	$\approx$12 GPU-h, 1$\times$A100	n=3 seeds

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 (Introduction) honestly scope DAGA's contributions to DINO/iBOT-family backbones with object-centric attention; cross-backbone limitations on MAE/CLIP/MoCo v3 are explicitly acknowledged in the Limitations section.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated Limitations section discusses (i) cross-backbone scope (DAGA depends on object-centric attention and underperforms on MAE/CLIP/MoCo v3), (ii) underperformance on the depth estimation task, and (iii) underperformance on vision-language retrieval.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper provides formal definitions distinguishing static PEFT from attention-conditioned PEFT, but does not state or prove any theorems; the contribution is empirical and architectural.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 describes the architecture and training protocol; Appendix D (Table A3) lists per-task hyperparameters, optimizer, learning rates and adaptation-layer choices; Appendix L documents the two attention-refinement front-ends (CLIPSelf, MAE self-distill) with epoch counts and GPU-hours; Appendix A1 provides DAGA pseudo-code; all datasets are standard public benchmarks.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A public code repository is referenced in the abstract (`https://github.com/Jaywalk18/DAGA`) with run scripts and configuration files; all evaluation datasets are publicly available standard benchmarks.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 (Setup) summarizes data splits, optimizer, schedules and protocols; Appendix D provides the full per-task hyperparameter table; the LR-sweep grid for the recent-PEFT comparison ($\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1.0\}$, val-best) is stated in the caption of Table 2.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: ImageNet-1K classification is reported as mean $\pm$ one standard deviation over three independent random seeds (DAGA: 85.90 ± 0.06, Table 5); other tables are single-seed and explicitly labelled as such in the captions of Tables 1 and 4, in line with standard practice for transfer-learning benchmarks.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments were run on $8\times$ NVIDIA A100 (40 GB) GPUs. Appendix N (Table A18) reports per-method training memory, training time, FLOPs and inference throughput on this hardware for all PEFT baselines and DAGA; the two attention-refinement front-ends (CLIPSelf ~ 18 GPU-h, MAE self-distill ~ 28 GPU-h on a single A100) are documented in Appendix L; setup, schedule and per-task hyperparameters are in Appendix C–D.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses only standard, publicly released vision benchmarks and pre-trained backbones, involves no human subjects, no scraped or personal data, and complies with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: DAGA is a parameter-efficient fine-tuning module for visual recognition tasks and does not introduce new societal risks beyond those already inherent in the underlying ViT backbones; positive impacts (reduced compute footprint of fine-tuning) are inherited from the broader PEFT line of work and are not unique to this paper.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases neither pre-trained model weights nor any new dataset; only training/evaluation code for an adapter module on top of existing public backbones is shared, which carries no high risk of misuse.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (ImageNet, ADE20K, COCO, NYU Depth v2, Stanford Cars, Oxford Flowers, Pets, Food-101, DTD, SUN-397, $\mathcal{R}$Oxford, $\mathcal{R}$Paris, ImageNet-C, CUB-200, FGVC-Aircraft) and the DINOv3 backbone (`facebookresearch/dinov3`) are cited in the experiments and references; we use them under their original licenses for non-commercial academic research.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The DAGA implementation is released as a public code repository (`https://github.com/Jaywalk18/DAGA`) with a README, configuration files, and run scripts; supplementary material additionally contains pseudo-code and hyperparameter tables documenting the module.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work involves no crowdsourcing and no research with human subjects; all experiments are run on existing public vision benchmarks.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve any human-subjects research, so IRB review or equivalent approval is not applicable.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We used a large language model as a writing aid for grammar and phrasing edits during manuscript preparation. The LLM was not involved in research ideation, methodology design, model architecture, experiment design, code, data analysis, or interpretation of results; all scientific content originates from the authors.