DECLAREVIT: DECOUPLING EARLY CLASSIFICATION FROM FEATURE REFINEMENT WITH EXIT QUERY FOR EFFICIENT VITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision Transformers achieve remarkable recognition performance but incur substantial computational cost during inference, limiting their use in latency- and compute-constrained deployment scenarios. Early-exit inference offers a promising solution by allowing easy samples to terminate at intermediate layers. However, existing approaches typically couple early classification with the shared backbone, requiring intermediate representations to support both early classification and subsequent feature refinement. Under joint training, the corresponding optimization objectives jointly update shared backbone parameters, which can lead to conflicting gradient directions. To address this issue, we introduce DeclareViT, a framework that decouples early classification from feature refinement. Each exit uses a dedicated query to read intermediate features through one-way cross-attention without modifying the backbone token stream. The query branches are optimized for early classification while the backbone and final classifier remain frozen. Across ViT, DeiT, and Swin on CIFAR-100, ImageNet-1K, and Food-101, DeclareViT consistently establishes a state-of-the-art accuracy–efficiency trade-off: It achieves $1.58\times$–$2.57\times$ multiply–accumulate (MAC)-based speed-up with only a 0.39–1.91 percentage-point accuracy drop. Wall-clock measurements on CPU and GPU further demonstrate practical inference acceleration. Code is available at `https://anonymous.4open.science/r/declarevit`.

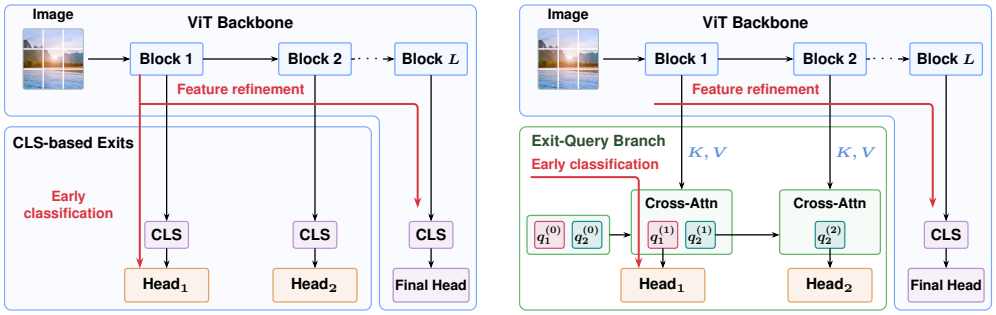
1 INTRODUCTION

Vision Transformers (ViTs) have achieved impressive performance across diverse visual-recognition tasks (Dosovitskiy et al., 2021; Touvron et al., 2021; Liu et al., 2021). However, standard ViTs process every input through all Transformer blocks, which can be inefficient in latency-sensitive and compute-constrained deployment scenarios. *Early-exit* techniques address this limitation by inserting classifiers at intermediate layers and terminating computation once a sufficiently confident prediction is reached (Teerapittayanon et al., 2016; Huang et al., 2018). This enables efficient inference by allocating less computation to easy samples while retaining full model capacity for difficult ones.

Recent work improves early-exit prediction through specialized heads (Bakhtiarnia et al., 2021; Xu et al., 2023), self-distillation and structured dropout (Shen et al., 2023; Dong et al., 2026), prediction reuse (Zhou et al., 2020; Zhang et al., 2022; Wang et al., 2024; Bajpai & Hanawal, 2025), and routing-aware supervision (Han et al., 2022; He et al., 2025). Other designs combine early termination with adaptive spatial granularity or model capacity (Wang et al., 2021; Chen et al., 2023; Hu et al., 2024; Hojjat et al., 2026).

Despite these advances, the interaction between early classification and the backbone representations required for deeper feature refinement remains largely coupled. In particular, many early-exit ViTs directly classify the intermediate class token, denoted as [CLS], and jointly optimize the intermediate exit classifiers and the final classifier as shown in Figure 1a (Bakhtiarnia et al., 2021; Xu et al., 2023; Dong et al., 2026). This design assigns two roles to the same representation: the intermediate [CLS] token must support both early classification and feature refinement by subsequent Transformer blocks. We refer to this coupling as *[CLS] role coupling*. Under joint training, the corresponding optimization objectives jointly update shared backbone parameters, which can lead to

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



(a) Conventional [CLS]-based early exits. (b) DeclareViT.

Figure 1: Comparison between conventional [CLS]-based early exits and DeclareViT. (a) Conventional [CLS]-based early exits using the shared backbone [CLS] representation for early prediction. (b) DeclareViT with read-only queries $\mathbf{q}_1$ and $\mathbf{q}_2$ progressively updated across backbone blocks.

conflicting gradient directions. Existing decoupling approaches alleviate this problem through separate classification branches (Han et al., 2023), feature partitioning (Gong et al., 2024), or adapters and bypass predictors (Zhang et al., 2024b; Liu et al., 2026; Luo et al., 2025). However, none of these decoupling methods establishes a fully read-only and gradient-isolated interface between early classification and shared feature refinement.

To address the challenge above, we introduce **DeclareViT**, a framework for **DE**coupling early **CLA**ssification from feature **RE**finement in early-exit Vision Transformers. As shown in Figure 1b, in DeclareViT, each exit branch maintains an independent learnable *exit query* outside the backbone token sequence. As an input propagates through the backbone, each query reads intermediate features through one-way cross-attention and accumulates information through residual updates. At each exit, a feed-forward network (FFN) refines the corresponding query, and a linear head produces the class logits. After an exit produces its prediction, the corresponding query is discarded. Training proceeds in two stages. We first optimize the backbone and final head using only the full-depth training objective. Then we freeze the backbone and final classifier and optimize only the exit-query branches. DeclareViT decouples early classification from feature refinement in both directions: in the forward pass, the queries read the intermediate features in the backbone but never modify them; in the backward pass, gradients from the exit-query losses do not propagate into the backbone.

In summary, our contributions are threefold:

- We characterize [CLS] role coupling in early-exit ViTs and empirically quantify optimization competition among prediction objectives through gradient-conflict analysis.
- We propose **DeclareViT**, a framework for **DE**coupling early **CLA**ssification from feature **RE**finement. It combines independent exit queries, one-way cross-attention, and two-stage training to separate early classification from backbone feature refinement in both the forward and backward passes.
- We systematically evaluate DeclareViT with ViT, DeiT, and Swin on CIFAR-100, ImageNet-1K, and Food-101. Across all combinations, DeclareViT outperforms state-of-the-art early-exit methods in the accuracy–efficiency trade-off, achieving $1.58\times$–$2.57\times$ multiply–accumulate (MAC)-based speed-up, with an accuracy drop of 0.39–1.91 percentage points. Wall-clock measurements on GPU and CPU further demonstrate its practical computational efficiency.

2 Related Work

Exit-Head Design and Information Reuse. Bakhtiarnia et al. (2021) investigate several ViT early-exit architectures, including MLP heads and convolutional, Transformer, MLP-Mixer, and ResMLP branches. Bakhtiarnia et al. (2022) attach single-layer Transformer branches to CNN backbones to improve early-exit accuracy with less computational overhead. LGViT (Xu et al., 2023) adopts heterogeneous heads, using local-perception heads at shallow exits and global-aggregation heads at deeper exits. For prediction reuse, ZTW (Wołczyk et al., 2021) feeds previous-layer classifier logits into subsequent classifiers and combines their predictions through geometric ensembling.

BEEM (Bajpai & Hanawal, 2025) accumulates confidence scores from consecutive exits that predict the same label and terminates inference when the accumulated score reaches a predefined threshold. Our DeclareViT instead reads backbone features into dedicated exit queries, accumulating feature-level information across depth rather than reusing earlier predictions.

Training and Optimization of Multi-Exit Models. Knowledge distillation transfers knowledge from deeper to shallower exits. BYOT (Zhang et al., 2019) trains shallow classifiers to match the deepest classifier's predictions and features. MEViT (Shen et al., 2023) trains the final and earliest exits together with a sampled subset of intermediate exits, which learn from the final prediction through self-distillation. EnViT (Dong et al., 2026) uses exit-aware structured dropout to create student subnetworks and distills knowledge from the full model into them, aiming to reduce interference between exits. Sample weighting instead addresses the mismatch between training and dynamic routing. L2W-DEN (Han et al., 2022) uses meta-learning to predict sample-specific loss weights for each exit, emphasizing easy samples at shallow exits and harder samples at deeper exits. COSEE (He et al., 2025) simulates early exit under multiple thresholds during training and assigns larger loss weights to classifiers near each sample's simulated exit. DeclareViT also uses distillation, but differs from joint-training approaches by confining early-exit optimization to the query branches.

Decoupling Early Classification from Feature Refinement. Dynamic Perceiver (Han et al., 2023) maintains a feature-extraction branch and a latent classification branch. Bidirectional cross-attention allows the classification branch to read visual features and send information back to the feature branch. Dyn-Adapter (Zhang et al., 2024b) freezes pretrained backbone weights while training adaptation modules and early heads. DFS (Gong et al., 2024) partitions features at each exit depth so that the local exit loss updates the exit-specific portion, while deeper objectives update the shared portion. MET (Liu et al., 2026) partially decouples different exits by inserting exit-specific adapters into the shared backbone, which transform intermediate features into separate [CLS] representations for individual exits. DMPO (Luo et al., 2025) decouples the role coupling through bypass modules: the original intermediate features are preserved for propagation to deeper stages, while bypass-transformed features are separately used for early classification. Related query-based tuning methods such as VQT (Tu et al., 2023) and SynQT (Zhang et al., 2024a) read intermediate features from frozen ViTs without modifying the backbone, but target parameter- or memory-efficient transfer learning rather than multi-exit inference. However, prior early-exit decoupling methods still let early-exit learning influence the feature-refinement path through feature feedback or trainable adaptation modules. In contrast, DeclareViT uses read-only exit queries and two-stage training to keep early classification separate from backbone feature refinement in both forward and backward passes.

3 [CLS] Role Coupling Analysis

3.1 Preliminaries

Vision Transformer. Given an input image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, a ViT (Dosovitskiy et al., 2021) projects N image patches into D-dimensional embeddings $\mathbf{P}^{(0)} \in \mathbb{R}^{D\times N}$ and prepends a learnable [CLS] token $\mathbf{x}_{\text{cls}}^{(0)} \in \mathbb{R}^{D\times 1}$, forming the input sequence $\mathbf{X}^{(0)} = [\mathbf{x}_{\text{cls}}^{(0)}, \mathbf{P}^{(0)}] \in \mathbb{R}^{D\times(N+1)}$. An L-block backbone produces the final logits as

$$\mathbf{X}^{(\ell)} = \mathcal{F}_\ell\Big(\mathbf{X}^{(\ell-1)}\Big), \quad \ell = 1,\dots,L, \qquad \mathbf{z}_{\text{f}} = \mathbf{W}_{\text{f}}\,\text{LN}_{\text{f}}\Big(\mathbf{x}_{\text{cls}}^{(L)}\Big) + \mathbf{b}_{\text{f}} \in \mathbb{R}^C, \tag{1}$$

where $\mathcal{F}_\ell$ denotes Transformer block ℓ, LN denotes LayerNorm, f identifies the final classifier, and C denotes the number of classes.

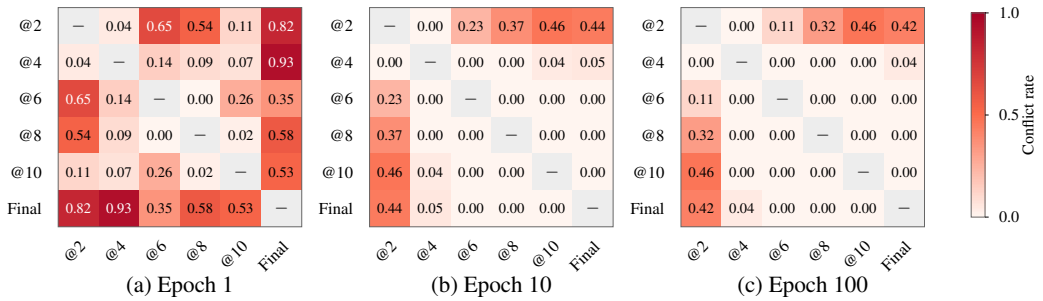
[CLS]-based early exits. Consider E early exits indexed by $e = 1,\dots,E$. Exit e is attached after backbone block d_e, where $1 \le d_1 < \cdots < d_E < L$. Thus, e identifies an exit, whereas d_e specifies its backbone depth; for example, five exits at blocks $2, 4, 6, 8, 10$ correspond to $(d_1,\dots,d_5) = (2, 4, 6, 8, 10)$. As shown in Figure 1a, each exit classifies the intermediate [CLS] token:

$$\mathbf{h}_e = \mathbf{x}_{\text{cls}}^{(d_e)}, \qquad \mathbf{z}_e = \mathcal{H}_e(\mathbf{h}_e), \qquad e = 1,\dots,E, \tag{2}$$

where $\mathcal{H}_e$ is a nonlinear exit head. Joint training minimizes

$$\mathcal{L}_{\text{joint}} = \lambda_{\text{f}}\mathcal{L}_{\text{f}} + \sum_{e=1}^{E} \lambda_e \mathcal{L}_e, \tag{3}$$

where $\mathcal{L}_{\text{f}}$ and $\mathcal{L}_e$ are the full-depth and exit-specific training objectives, with weights λ_{f} and λ_e.



(a) Epoch 1 (b) Epoch 10 (c) Epoch 100

Figure 2: Pairwise gradient conflict rates on backbone block 2 of our [CLS]-based SDN adaptation at training epochs 1, 10, and 100, using ViT-B/16 on ImageNet-1K. @2–@10 denote exit depths and Final denotes the full-depth objective. Early and deeper prediction objectives can favor opposing updates to the shared backbone even at a late training checkpoint.

3.2 [CLS] Role Coupling

Coupled roles and optimization. The same representation $\mathbf{h}_e$ is used for early classification at exit e and is subsequently propagated through the remaining backbone blocks for further refinement. We refer to this shared use as [CLS] role coupling. During joint training, prediction objectives at different depths can impose competing demands on the shared backbone parameters.

To characterize this optimization coupling, let $\boldsymbol{\theta}_\ell$ denote the parameters of backbone block ℓ. For early exit e, we consider gradients on blocks $1 \le \ell \le d_e$; the final objective involves all blocks $1 \le \ell \le L$. We use the individual objective gradients:

$$\mathbf{g}_{e,\ell} = \nabla_{\boldsymbol{\theta}_\ell} \mathcal{L}_e, \qquad \mathbf{g}_{\mathrm{f},\ell} = \nabla_{\boldsymbol{\theta}_\ell} \mathcal{L}_{\mathrm{f}}. \tag{4}$$

For distinct objectives that backpropagate through the same block, we compute the cosine similarity:

$$c_{ij}^{(\ell)} = \frac{\langle \mathbf{g}_{i,\ell}, \mathbf{g}_{j,\ell} \rangle}{\|\mathbf{g}_{i,\ell}\|_2 \, \|\mathbf{g}_{j,\ell}\|_2 + \epsilon}, \qquad i, j \in \{1, \ldots, E, \mathrm{f}\}, \quad i \neq j. \tag{5}$$

Here, $\epsilon = 10^{-12}$. A negative cosine indicates opposing objective-gradient directions (Yu et al., 2020). The *conflict rate* is the fraction of valid sampled training minibatches with negative cosine.

Gradient-conflict analysis. For analysis, we adapt SDN (Kaya et al., 2019) to ViTs on ImageNet-1K by attaching a LayerNorm followed by a linear classifier to the intermediate [CLS] token at blocks $\{2, 4, 6, 8, 10\}$. The backbone, early-exit heads, and final classifier are jointly trained using hard-label cross-entropy losses. Figure 2 reports pairwise gradient conflict rates on backbone block 2, through which all five early-exit losses and the final loss backpropagate. The @2–Final and @4–Final conflict rates are 0.82 and 0.93 at epoch 1. Although many objective pairs become aligned later, @2–Final still has conflict rates of 0.44 and 0.42 at epochs 10 and 100. Thus, early and deeper prediction objectives can favor opposing updates to the shared backbone even at a late training checkpoint. These observations reveal persistent optimization competition in the jointly trained [CLS]-based model and motivate an early-classification pathway that does not update the backbone. Additional analyses are provided in Appendix G.

4 Decoupling Early Classification from Feature Refinement

Motivated by the shared-backbone competition in Section 3, DeclareViT combines read-only exit queries with two-stage training to decouple early classification from feature refinement (Figure 3). We next describe the query architecture, training procedure, and dynamic inference.

4.1 Read-Only Exit Queries

Query initialization. Each exit e has a learnable initial query $\mathbf{q}_e^{(0)} \in \mathbb{R}^{D \times 1}$ serving as its dedicated early-classification state outside the backbone token sequence. Stacking these queries gives $\mathbf{Q}^{(0)} = [\mathbf{q}_1^{(0)}, \ldots, \mathbf{q}_E^{(0)}] \in \mathbb{R}^{D \times E}$. $\mathbf{q}_e^{(\ell)}$ denotes the state of query e after reading backbone block ℓ. Each query is updated until its exit depth d_e.

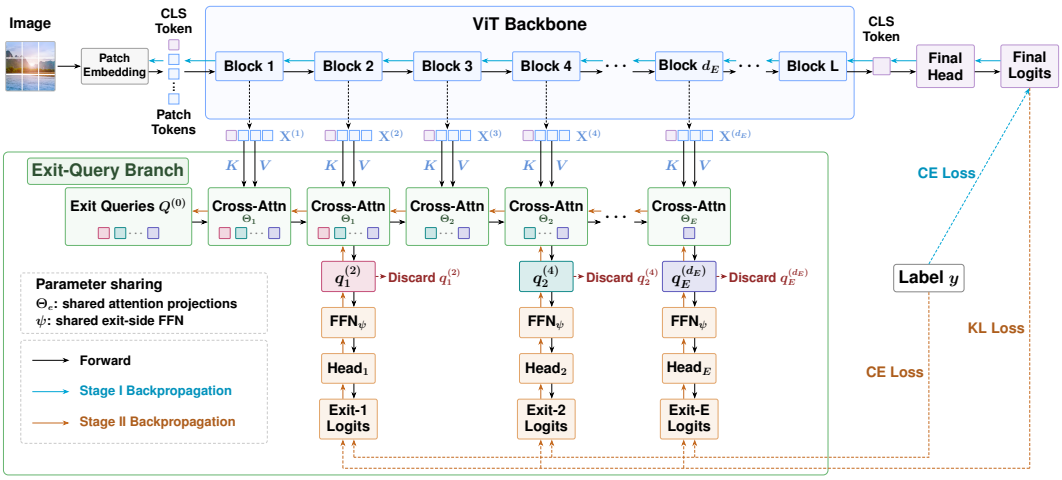


Figure 3: DeclareViT with progressively updated read-only exit queries, discarded after prediction. Stage I trains the backbone and final classifier with cross-entropy; Stage II freezes them and trains only the exit branches with cross-entropy and KL distillation from the final prediction.

Extracting and accumulating information. At backbone block ℓ, each active query e with $\ell \leq d_e$ reads the intermediate features $\mathbf{X}^{(\ell)}$. Its query, key, and value projections are

$$\mathbf{Q}_e^{(\ell)} = \mathbf{W}_{Q,\ell}\,\mathrm{LN}_{q,\ell}\left(\mathbf{q}_e^{(\ell-1)}\right),\ \mathbf{K}^{(\ell)} = \mathbf{W}_{K,\ell}\,\mathrm{LN}_{b,\ell}\left(\mathbf{X}^{(\ell)}\right),\ \mathbf{V}^{(\ell)} = \mathbf{W}_{V,\ell}\,\mathrm{LN}_{b,\ell}\left(\mathbf{X}^{(\ell)}\right). \quad (6)$$

Here, $\mathrm{LN}_{q,\ell}$ and $\mathrm{LN}_{b,\ell}$ are separate LayerNorm operations for the query and backbone features, respectively. The matrices $\mathbf{W}_{Q,\ell}$, $\mathbf{W}_{K,\ell}$, and $\mathbf{W}_{V,\ell} \in \mathbb{R}^{D\times D}$ are the corresponding projection matrices. For a standard ViT, $\mathbf{Q}_e^{(\ell)}$ has shape $D \times 1$, $\mathbf{K}^{(\ell)}$ and $\mathbf{V}^{(\ell)}$ each have shape $D \times (N+1)$.

Each query accumulates information through a residual cross-attention update:

$$\mathbf{q}_e^{(\ell)} = \mathbf{q}_e^{(\ell-1)} + \mathrm{MHA}_\ell\left(\mathbf{Q}_e^{(\ell)}, \mathbf{K}^{(\ell)}, \mathbf{V}^{(\ell)}\right), \qquad 1 \leq \ell \leq d_e, \quad (7)$$

where MHA_ℓ denotes multi-head cross-attention (Vaswani et al., 2017) without applying additional query, key, or value projections. Throughout this process, the query remains read-only: it attends to $\mathbf{X}^{(\ell)}$ without modifying the backbone features. For hierarchical backbones such as Swin (Liu et al., 2021), where repeatedly reading high-resolution intermediate features would incur substantial overhead, we use an exit-local non-accumulative reader; details are provided in Appendix A.2.

Exit readout and discard. When the backbone reaches block $\ell = d_e$, query e produces its prediction through an exit-side FFN and an exit-specific linear classifier:

$$\widetilde{\mathbf{q}}_e = \mathbf{q}_e^{(d_e)} + \mathrm{FFN}_\psi\left(\mathrm{LN}_{\mathrm{ffn}}\left(\mathbf{q}_e^{(d_e)}\right)\right), \qquad \mathbf{z}_e = \mathbf{W}_e\widetilde{\mathbf{q}}_e + \mathbf{b}_e. \quad (8)$$

Here, ψ denotes the FFN parameters, while $\mathbf{W}_e$ and $\mathbf{b}_e$ form the linear classifier. After producing $\mathbf{z}_e$, query $\mathbf{q}_e^{(d_e)}$ is discarded, i.e., excluded from the query matrices used by subsequent readers.

4.2 Two-Stage Decoupled Training

Stage I: full-depth training. This stage learns the backbone trajectory without early-exit supervision. For an image–label pair $(\mathbf{I}, y)$, Stage I optimizes only the full-depth classification objective:

$$(\theta^\star, \omega^\star) = \arg\min_{\theta,\omega}\ \mathbb{E}_{(\mathbf{I},y)}\left[\mathcal{L}_{\mathrm{CE}}\left(\mathbf{z}_{\mathrm{f}}(\mathbf{I};\theta,\omega), y\right)\right], \quad (9)$$

where θ and ω denote the parameters of the backbone and final classifier, respectively.

Stage II: exit-branch training. We then freeze $\theta^\star$ and $\omega^\star$ and optimize only the exit-query branch parameters φ, including the learnable queries, cross-attention projections, exit-side FFN, and exit classifiers. The loss for exit e combines ground-truth supervision with self-distillation from the frozen final classifier:

$$\mathcal{L}_e = (1-\alpha)\mathcal{L}_{\mathrm{CE}}\left(\mathbf{z}_e, y\right) + \alpha T^2 \mathcal{L}_{\mathrm{KL}}\left(\mathbf{p}_{\mathrm{f}}^{(T)} \,\middle\|\, \mathbf{p}_e^{(T)}\right), \quad (10)$$

where $\mathcal{L}_{\mathrm{KL}}$ denotes the Kullback–Leibler divergence, $\mathbf{p}_e^{(T)} = \mathrm{softmax}(\mathbf{z}_e/T)$ and $\mathbf{p}_{\mathrm{f}}^{(T)} = \mathrm{softmax}(\mathbf{z}_{\mathrm{f}}/T)$. Here, $\alpha \in [0,1]$ controls the distillation weight and $T > 0$ is the temperature.

The Stage II objective is

$$\varphi^\star = \arg\min_{\varphi} \ \mathbb{E}_{(\mathbf{I},y)} \left[\sum_{e=1}^{E} \lambda_e \mathcal{L}_e \right], \qquad \theta = \theta^\star, \quad \omega = \omega^\star. \tag{11}$$

4.3 Dynamic Inference and Parameter Sharing

Dynamic inference. DeclareViT evaluates exits in depth order. For $\mathbf{p}_e(\mathbf{I}) = \mathrm{softmax}(\mathbf{z}_e(\mathbf{I}))$, prediction uncertainty is measured by normalized entropy (Teerapittayanon et al., 2016):

$$H_e(\mathbf{I}) = -\frac{1}{\log C} \sum_{c=1}^{C} p_{e,c}(\mathbf{I}) \log p_{e,c}(\mathbf{I}), \ H_e(\mathbf{I}) \in [0,1]. \tag{12}$$

At exit e, inference terminates when $H_e(\mathbf{I}) \le \tau_e$. Otherwise, computation continues to the next exit. If no early exit meets its threshold, the final classifier produces the prediction. Here, τ_e denotes the entropy threshold for exit e.

Parameter sharing. To reduce parameter overhead introduced by the exit branches, all exits share an FFN and its LayerNorm, while query states and classifiers remain exit-specific. We further adopt inter-exit segment sharing for cross-attention. With $d_0 = 0$, segment e contains backbone blocks $\ell = d_{e-1}+1, \ldots, d_e$, for $e = 1, \ldots, E$. All readers within each segment share the projection matrices W_Q, W_K, W_V, and W_O. The query-side and memory-side LayerNorms are layer-specific and shared by all active exit queries at the same block.

5 Experiments

5.1 Experimental Setup

Datasets and backbones. We evaluate on CIFAR-100 (Krizhevsky, 2009), ImageNet-1K (Deng et al., 2009), and Food-101 (Bossard et al., 2014), using ViT-B/16 (Dosovitskiy et al., 2021), distilled DeiT-B/16 (Touvron et al., 2021), and Swin-B (Liu et al., 2021). We refer to these backbones as ViT, DeiT, and Swin. ViT and DeiT have 12 blocks, and Swin has 24.

Metrics. We report Top-1 accuracy, MAC-based speed-up $\mathbf{S}_{\mathrm{MAC}}$, and wall-clock speed-up $\mathbf{S}_{\mathrm{wall}}$. Following prior work (Dong et al., 2026), we define $\mathbf{S}_{\mathrm{MAC}} = N_s C_{\mathrm{base}} / \sum_j n_j C_j$ and $\mathbf{S}_{\mathrm{wall}} = T_{\mathrm{base}}/T_{\mathrm{dyn}}$. Here, j indexes all early-exit and final prediction paths, n_j is the number of samples terminating at path j, and $N_s = \sum_j n_j$ is the total number of samples. C_j is the cumulative MAC cost of path j, including all backbone and exit-branch operations executed before termination, while C_{base} denotes the MAC cost of the unbranched reference. T_{base} and T_{dyn} are measured on the same hardware under a unified benchmarking protocol.

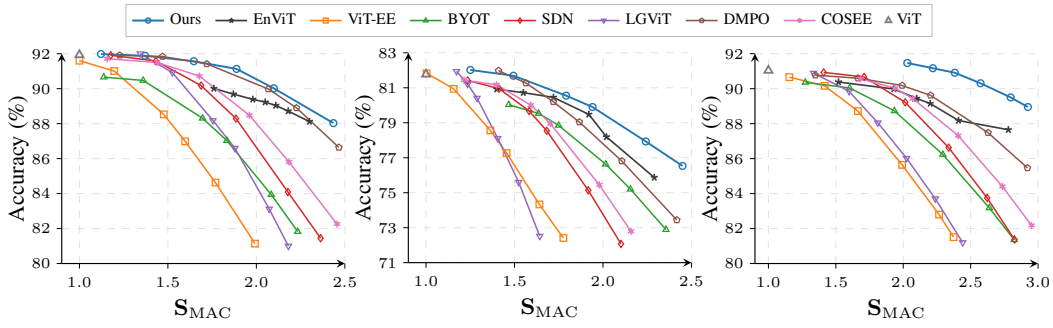
Implementation details. All inputs are processed at 224×224 resolution. DeclareViT uses five early exits at blocks $\{2,4,6,8,10\}$ for ViT and DeiT and global blocks $\{4,8,12,16,20\}$ for Swin, with one query per exit. Thresholds are calibrated on a held-out training subset and applied to the final evaluation split. Further dataset, backbone, architecture, training, hardware, and calibration details are provided in Appendix A.

5.2 Main Accuracy–Efficiency Trade-off

Table 1 compares six locally evaluated baselines across nine dataset–backbone settings and includes EnViT as an external ViT reference. Following prior work (Xu et al., 2023; Dong et al., 2026), it provides representative operating points as a compact summary of the overall accuracy–efficiency trade-off. At the reported operating points, DeclareViT attains higher accuracy and larger MAC-based speed-up than the locally evaluated early-exit baselines across all nine settings. Relative to the corresponding full-depth references, it provides $1.58\times$–$2.57\times$ $\mathbf{S}_{\mathrm{MAC}}$ with accuracy reductions of 0.39–1.91 percentage points. Figure 4 complements Table 1 with the corresponding ViT threshold-sweep curves. Appendix B provides calibration-targeted budget comparisons, multi-seed results, and additional DeiT/Swin trade-off curves.

Table 1: Comparison of early-exit methods on CIFAR-100, ImageNet-1K, and Food-101 using ViT, DeiT, and Swin backbones with seed 2024. Top-1 accuracy and $\mathbf{S}_{\mathrm{MAC}}$ are reported. **Bold** and underlining mark the best and second-best results within each dataset–backbone setting; full-depth references are excluded from this ranking. † denotes externally reported EnViT results.

Backbone	Method	CIFAR-100		ImageNet-1K		Food-101	
		Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑	Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑	Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
ViT	ViT (Dosovitskiy et al., 2021)	91.93	1.00×	81.77	1.00×	91.04	1.00×
	SDN (Kaya et al., 2019)	88.29	1.89×	79.20	1.63×	89.22	2.01×
	BYOT (Zhang et al., 2019)	88.31	1.70×	79.53	1.64×	88.73	1.93×
	ViT-EE (Bakhtiarnia et al., 2021)	88.53	1.48×	78.56	1.36×	88.72	1.66×
	LGViT (Xu et al., 2023)	88.18	1.75×	79.21	1.36×	89.85	1.60×
	COSEE (He et al., 2025)	88.48	1.96×	79.38	1.66×	89.44	2.07×
	DMPO (Luo et al., 2025)	89.98	2.07×	80.52	1.68×	89.62	2.20×
	EnViT† (Dong et al., 2026)	89.23	2.05×	80.45	1.72×	89.15	2.20×
	Ours	**90.02**	**2.10×**	**80.55**	**1.79×**	**90.31**	**2.57×**
DeiT	DeiT (Touvron et al., 2021)	89.90	1.00×	83.38	1.00×	89.96	1.00×
	SDN (Kaya et al., 2019)	88.53	1.61×	78.50	1.57×	89.01	1.68×
	BYOT (Zhang et al., 2019)	87.06	1.51×	78.03	1.54×	88.35	1.74×
	ViT-EE (Bakhtiarnia et al., 2021)	88.19	1.24×	79.13	1.20×	88.53	1.42×
	LGViT (Xu et al., 2023)	88.46	1.40×	78.85	1.36×	88.77	1.51×
	COSEE (He et al., 2025)	88.61	1.55×	78.70	1.58×	88.80	1.70×
	DMPO (Luo et al., 2025)	88.87	1.55×	80.99	1.35×	88.89	1.67×
	Ours	**88.95**	**1.69×**	**81.96**	**1.59×**	**89.57**	**2.15×**
Swin	Swin (Liu et al., 2021)	92.47	1.00×	83.59	1.00×	93.68	1.00×
	SDN (Kaya et al., 2019)	90.65	1.23×	79.31	1.57×	91.07	1.72×
	BYOT (Zhang et al., 2019)	87.39	1.20×	78.40	1.56×	89.83	1.50×
	ViT-EE (Bakhtiarnia et al., 2021)	90.61	1.42×	80.51	1.52×	90.42	2.07×
	LGViT (Xu et al., 2023)	90.55	1.38×	81.23	1.50×	91.75	1.84×
	COSEE (He et al., 2025)	88.25	1.35×	79.66	1.57×	90.04	1.67×
	DMPO (Luo et al., 2025)	90.62	1.62×	81.59	1.44×	91.39	2.20×
	Ours	**90.81**	**1.73×**	**82.71**	**1.58×**	**92.37**	**2.49×**



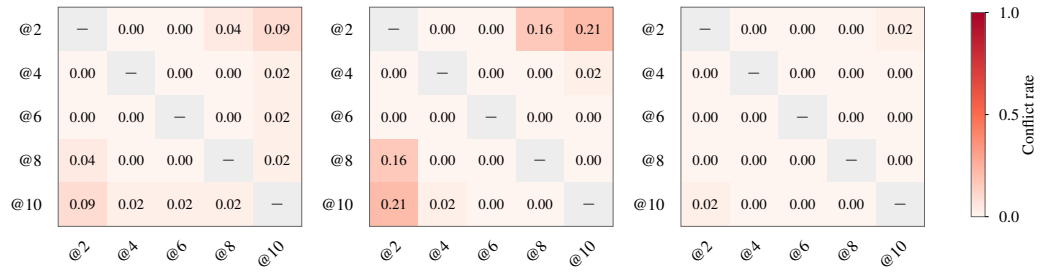
(a) CIFAR-100 (b) ImageNet-1K (c) Food-101

Figure 4: Accuracy–$\mathbf{S}_{\mathrm{MAC}}$ trade-offs for ViT across three datasets.

5.3 Decoupling Analysis in DeclareViT

We analyze DeclareViT with ViT on ImageNet-1K. During Stage II, the backbone and final classifier remain frozen. Trainable parameters shared across exits are therefore confined to exit-side modules, including the segment-shared cross-attention projections and the shared exit-side FFN. We examine the remaining inter-exit gradient interactions on the cross-attention projections shared across reader blocks 1–2, using the diagnostic defined in Section 3.2.

Figure 5 shows that inter-exit conflict rates are already low during epoch 1, ranging from 0.00 to 0.09 in the measured shared attention group. Although a few exit pairs exhibit higher rates during epoch 10, including 0.21 for @2–@10, most pairs remain at or near zero. During epoch 100, all reported rates range from 0.00 to 0.02. Thus, low conflict rates are already observed early in



(a) Epoch 1 (b) Epoch 10 (c) Epoch 100

Figure 5: Pairwise gradient conflict rates between DeclareViT exit objectives on the cross-attention projections shared across reader blocks 1–2 during Stage II. @2–@10 denote exit depths.

Table 2: Wall-clock measurement on ImageNet-1K with ViT. Wall-clock times are reported as the mean ± standard deviation over three independent runs, and $\mathbf{S}_{\text{wall}}$ is measured relative to the unbranched ViT on each device. **Bold** marks the best results for each metric among early-exit methods.

Method	**Acc. (%)** ↑	$\mathbf{S}_{\text{MAC}}$ ↑	**GPU**		**CPU**	
			Wall-clock time ↓	$\mathbf{S}_{\text{wall}}$ ↑	**Wall-clock time** ↓	$\mathbf{S}_{\text{wall}}$ ↑
ViT	81.77	1.00×	14.62 ± 0.04 ms	1.00×	246.42 ± 3.85 ms	1.00×
SDN	79.20	1.63×	9.43 ± 0.02 ms	1.55×	152.40 ± 1.47 ms	1.62×
BYOT	79.53	1.64×	9.41 ± 0.02 ms	1.55×	152.60 ± 1.48 ms	1.61×
ViT-EE	78.56	1.36×	11.19 ± 0.01 ms	1.31×	184.33 ± 2.00 ms	1.34×
LGViT	79.21	1.36×	12.17 ± 0.00 ms	1.20×	196.38 ± 1.79 ms	1.25×
COSEE	79.38	1.66×	9.29 ± 0.01 ms	1.57×	149.37 ± 0.63 ms	1.65×
DMPO	80.52	1.68×	9.85 ± 0.03 ms	1.48×	151.41 ± 0.32 ms	1.63×
Ours	**80.55**	**1.79×**	**9.20 ± 0.00** ms	**1.59×**	**144.63 ± 0.70** ms	**1.70×**

training. These measurements characterize residual gradient competition within the shared exit-side parameters, while the frozen, read-only design prevents exit learning from altering the backbone feature-refinement path. Appendix G extends the analysis to additional shared modules, while Appendix H provides qualitative comparisons of [CLS] and exit-query attention.

5.4 Wall-clock speed-up

Table 2 reports inference latency on ImageNet-1K with ViT at a batch size of 1, measured on an NVIDIA Quadro RTX 6000 GPU and an Intel Xeon Platinum 8480+ CPU using a single CPU thread. DeclareViT achieves the highest accuracy and the largest wall-clock speed-up among the evaluated early-exit methods on both GPU and CPU, consistent with its advantage in MAC-based efficiency. Further measurement details and results are provided in Appendix C.

5.5 Ablation Study

We ablate DeclareViT with ViT on ImageNet-1K. Appendix D provides further studies.

Exit-head design. Table 3 compares readouts with the same frozen backbone, exit locations, and training objective. Exit queries improve block-2 accuracy from 29.70% to 60.10% over [CLS]+MLP and outperform Transformer-encoder heads with 25M fewer parameters. Because these variants share the same frozen backbone, the advantage cannot be attributed to backbone freezing alone.

Query-branch components. Table 4(a) compares three ablated variants with backbone and final-classifier parameters frozen. Removing the exit-side FFN reduces accuracy from 80.55% to 80.41% and $\mathbf{S}_{\text{MAC}}$ from 1.79× to 1.60×. The pooling readout replaces exit queries and cross-attention with mean pooling over normalized detached [CLS] and patch tokens, reaching 80.53% at 1.48×. The token-injection variant inserts learnable exit tokens into the backbone sequence and lets them interact with backbone tokens through standard self-attention; it achieves 79.82% at 1.46×. These results favor the read-only query design with an exit-side FFN.

Query accumulation. Compared with reading features only at the designated exit, progressive accumulation improves accuracy from 80.44% to 80.55% and $\mathbf{S}_{\text{MAC}}$ from 1.71× to 1.79× (Table 4(b)). This supports integrating information across backbone depths.

Table 3: Comparison of early-exit head designs. All variants use the same frozen backbone, exit locations, and training objective. Per-exit accuracy evaluates each exit independently; Acc. denotes accuracy under dynamic exiting. **Bold** and underlining mark the best and second-best results.

Exit Head	Per-Exit Accuracy (%)						Acc. (%) ↑	S_{MAC} ↑	#Params. ↓
	@2	@4	@6	@8	@10	Final			
[CLS] + Linear	19.05	42.56	53.49	62.57	71.67	81.77	78.40	1.30×	**90**M
[CLS] + MLP	29.70	50.33	60.32	67.12	73.83	81.77	79.01	1.43×	96M
Transformer Encoder	60.02	69.75	75.02	78.29	80.56	81.77	79.66	1.56×	132M
Patch Tokens + CNN	51.29	61.92	69.26	74.83	79.60	81.77	79.63	1.48×	158M
Ours	**60.10**	**70.08**	**75.71**	**79.03**	**81.20**	81.77	**80.55**	**1.79**×	107M

Table 4: Ablations on ImageNet-1K with ViT: (a) query-branch components, (b) query accumulation, (c) parameter sharing, and (d) training protocol. **Bold** marks the best result within each panel.

(a) Components

Variant	Acc. (%) ↑	S_{MAC} ↑
w/o FFN	80.41	1.60×
Pooling readout	80.53	1.48×
Token injection	79.82	1.46×
Ours	**80.55**	**1.79**×

(b) Query accumulation

Readout	Acc. (%) ↑	S_{MAC} ↑
One-shot	80.44	1.71×
Accumulative (ours)	**80.55**	**1.79**×

(c) Parameter sharing

Sharing	#Params. ↓	Acc. (%) ↑	S_{MAC} ↑
None	138M	**80.79**	**1.83**×
Cross-attn.	126M	80.66	**1.83**×
FFN	119M	80.65	1.79×
Both (ours)	**107**M	80.55	1.79×

(d) Training protocol

Protocol	Acc. (%) ↑	S_{MAC} ↑
Joint-attached	80.12	1.67×
Joint-detached	80.23	1.72×
Two-stage (ours)	**80.55**	**1.79**×

Parameter sharing. Sharing attention projections within inter-exit segments and the FFN across exits reduces parameters from 138M to 107M, with modest changes in accuracy (80.79% to 80.55%) and S_{MAC} ($1.83\times$ to $1.79\times$; Table 4(c)). We adopt this compact configuration by default. Notably, the unshared variant indicates additional performance gains when the parameter budget is relaxed.

Training protocol. We compare two joint-training variants initialized from the same backbone and final classifier. All variants use the same exit-query architecture, initialization, exit objective, distillation settings, and exit-training budget, differing only in how the backbone and exit branches are optimized. In *Joint-attached*, exit losses backpropagate through the intermediate features and update the backbone together with the main classification objective, yielding 80.12% accuracy at $1.67\times$. In *Joint-detached*, stop-gradient is applied at the backbone–exit interface, so exit losses no longer update the backbone, while the backbone and exits are still optimized simultaneously. This improves performance to 80.23% at $1.72\times$. Our two-stage protocol additionally freezes the trained backbone and final classifier before exit optimization, exposing the exit modules to a stationary representation space. It further improves the result to 80.55% at $1.79\times$ (Table 4(d)). These results support gradient isolation and are consistent with additional benefits from training the exits on fixed backbone representations.

6 Conclusion

DeclareViT decouples early classification from feature refinement using independent read-only exit queries and two-stage training, preserving the backbone trajectory and full-depth prediction. Experiments across three backbones and three datasets demonstrate improved accuracy–efficiency trade-offs and practical GPU/CPU acceleration. These results show that effective early classification can be learned without reshaping the backbone for deeper predictions.

Limitations. DeclareViT introduces additional exit-query parameters beyond the backbone. Parameter sharing substantially reduces this overhead, although the model remains larger than the unbranched backbone. In addition, DeclareViT currently uses manually specified exit locations. Automatically adapting the number and placement of exits to different backbones and deployment budgets remains a promising direction for future work.

AI USE STATEMENT

We used generative AI tools to assist with language editing, proofreading, and the presentation of the manuscript, as well as with code-level assistance such as debugging, refactoring, and improving implementation clarity. Generative AI tools were not used to generate experimental data or results, and all scientific claims, experimental designs, analyses, and conclusions were reviewed and validated by the authors. All AI-assisted text was manually inspected and revised by the authors. AI-assisted code was reviewed, executed, and tested for correctness of the intended implementation. References, factual statements, and comparisons to prior work were checked by the authors. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the implementation of DeclareViT and all locally evaluated baselines in the accompanying anonymous code repository `https://anonymous.4open.science/r/declarevit` to support reproducibility. The main paper specifies the model architecture, training objectives, dynamic inference rule, and evaluation metrics. Appendix A provides detailed architecture, optimization, dataset split, preprocessing, and threshold-calibration settings; Appendix E documents the implementation provenance and training protocols of the reproduced baselines; and Appendix F describes the MAC-counting convention and gives an explicit analytical example. Additional calibration-targeted budget and multi-seed results are reported in Appendix B, while Appendix C specifies the wall-clock evaluation protocol.

The released repository contains the dataset and training configurations, DeclareViT and baseline model implementations, threshold calibration, dynamic evaluation, and MAC-accounting code used for the main locally evaluated accuracy–efficiency experiments. For auxiliary analyses and ablation studies, the paper and appendices provide the corresponding experimental protocols and implementation details.

REFERENCES

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. BEEM: Boosting performance of early exit DNNs using multi-exit classifiers as experts. In *ICLR*, 2025.

Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Multi-exit vision transformer for dynamic inference. In *BMVC*, 2021.

Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Single-layer vision transformers for more accurate early exits with less overhead. *Neural Networks*, 153:461–473, 2022.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *ECCV*, pp. 446–461, 2014.

Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. CF-ViT: A general coarse-to-fine method for vision transformer. In *AAAI*, pp. 7042–7052, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Yonghao Dong, Qiang He, Penghong Rui, Zhenzhe Zheng, Zhao Li, Feifei Chen, Hai Jin, and Yun Yang. En-ViT: Enhancing the performance of early-exit vision transformers via exit-aware structured dropout-enabled self-distillation. In *AAAI*, pp. 20852–20860, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Cheng Gong, Yao Chen, Qiuyang Luo, Ye Lu, Tao Li, Yuzhi Zhang, Yufei Sun, and Le Zhang. Deep feature surgery: Towards accurate and efficient multi-exit networks. In *ECCV*, volume 15107, pp. 435–451, 2024.

Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *ECCV*, volume 13671, pp. 362–378, 2022.

Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *ICCV*, pp. 5992–6002, 2023.

Jianing He, Qi Zhang, Hongyun Zhang, Xuanjing Huang, Usman Naseem, and Duoqian Miao. COSEE: Consistency-oriented signal-based early exiting via calibrated sample weighting mechanism. In *AAAI*, pp. 24023–24031, 2025.

Ali Hojjat, Janek Haberer, Sören Pirk, and Olaf Landsiedel. ThinkingViT: Matryoshka thinking vision transformer for elastic inference. In *CVPR*, pp. 41923–41933, 2026.

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

Youbing Hu, Yun Cheng, Anqi Lu, Zhiqiang Cao, Dawei Wei, Jie Liu, and Zhijun Li. LF-ViT: Reducing spatial redundancy in vision transformer for efficient image recognition. In *AAAI*, pp. 2274–2284, 2024.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *ICML*, pp. 3301–3310, 2019.

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

Piotr Kubaty, Filip Szatkowski, Grzegorz Choczyński, Eric Nalisnick, and Bartosz Wójcik. Rethinking calibration for early-exit neural networks. In *ICML*, 2026.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pp. 10012–10022, 2021.

Zheng Liu, Jinchao Zhu, Nannan Li, and Gao Huang. Multiple-exit tuning: Towards inference-efficient adaptation for vision transformer. *IEEE Trans. Circuits Syst. Video Technol.*, 36(2):2123–2136, 2026.

Liwei Luo, Shuaitengyuan Li, Dongwei Ren, Qilong Wang, Pengfei Zhu, and Qinghua Hu. Decoupled multi-predictor optimization for inference-efficient model tuning. In *ICCV*, pp. 3628–3638, 2025.

Tianyi Shen, Chonghan Lee, and Vijaykrishnan Narayanan. Multi-exit vision transformer with custom fine-tuning for fine-grained image recognition. In *ICIP*, pp. 2830–2834, 2023.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *ICPR*, pp. 2464–2469, 2016.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pp. 10347–10357, 2021.

Cheng-Hao Tu, Zheda Mai, and Wei-Lun Chao. Visual query tuning: Towards effective usage of intermediate representations for parameter and memory efficient transfer learning. In *CVPR*, pp. 7725–7735, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Jingcun Wang, Bing Li, and Grace Li Zhang. Early-exit with class exclusion for efficient inference of neural networks. In *AICAS*, pp. 263–267, 2024.

Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *NeurIPS*, volume 34, pp. 11960–11973, 2021.

Ross Wightman. Pytorch Image Models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T. Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzciński. Zero time waste: Recycling predictions in early exit neural networks. In *NeurIPS*, volume 34, pp. 2516–2528, 2021.

Guanyu Xu, Jiawei Hao, Li Shen, Han Hu, Yong Luo, Hui Lin, and Jialie Shen. LGViT: Dynamic early exiting for accelerating vision transformer. In *ACM MM*, pp. 9103–9114, 2023.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *NeurIPS*, volume 33, pp. 5824–5836, 2020.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, pp. 3713–3722, 2019.

Taolin Zhang, Jiawang Bai, Zhihe Lu, Dongze Lian, Genping Wang, Xinchao Wang, and Shu-Tao Xia. Parameter-efficient and memory-efficient tuning for vision transformer: A disentangled approach. In *ECCV*, volume 15103, pp. 346–363, 2024a.

Yurong Zhang, Honghao Chen, Xinyu Zhang, Xiangxiang Chu, and Li Song. Dyn-Adapter: Towards disentangled representation for efficient visual recognition. In *ECCV*, volume 15129, pp. 433–448, 2024b.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. PCEE-BERT: Accelerating BERT inference via patient and confident early exiting. In *Findings of ACL: NAACL*, pp. 327–338, 2022.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. BERT loses patience: Fast and robust inference with early exit. In *NeurIPS*, volume 33, pp. 18330–18341, 2020.

Table 5: Dataset partitions under the released configuration. Local training and selection/calibration subsets partition the official training set; the counts follow the released splitting rules on complete official datasets.

Dataset	Classes	Local training	Selection/calibration	Final evaluation
CIFAR-100	100	45,000	5,000	10,000 (test)
ImageNet-1K	1,000	1,153,050	128,117	50,000 (validation)
Food-101	101	68,175	7,575	25,250 (test)

Table 6: Backbone and exit-reader settings. Block indices are one-based; Swin indices count Transformer blocks globally across stages, excluding patch-merging operations. All variants use one query per exit, eight reader attention heads, an FFN expansion ratio of four, and a shared exit-side FFN.

Setting	ViT	DeiT	Swin
Backbone depth	12	12	$(2, 2, 18, 2)$ by stage
Backbone width	768	768	$(128, 256, 512, 1024)$ by stage
Exit blocks	$\{2, 4, 6, 8, 10\}$	$\{2, 4, 6, 8, 10\}$	$\{4, 8, 12, 16, 20\}$
Query width D_q	768	768	384
Reader memory	Patches + [CLS]	Patches + [CLS] + distillation token	Spatial tokens
Query updates	Every block up to its exit	Every block up to its exit	Once at its exit block
Attention sharing	Inter-exit segments	Inter-exit segments	None across readers

A Further Implementation Details

A.1 Datasets and Data Protocol

Datasets and splits. We use the classification datasets introduced in Section 5.1: CIFAR-100 contains low-resolution natural images, ImageNet-1K provides large-scale visual categories, and Food-101 contains food categories. Table 5 summarizes their class counts and evaluation splits. We reserve 10% of each official training set for checkpoint selection and threshold calibration. The remaining samples are used for local training.

For ImageNet-1K, the holdout is excluded from local exit-branch training; this does not imply that it was excluded from pretraining of the public backbone checkpoint. The official validation set is the final evaluation split, distinct from this training-set holdout.

Preprocessing. All models use 224×224 inputs. Training applies random resized cropping, random horizontal flipping, and model-specific normalization. Evaluation and calibration use the resize, crop, interpolation, and normalization settings resolved from the selected `timm` checkpoint.

A.2 Backbones and Backbone-Specific Exit Branches

Backbones and initialization. We use ViT-B/16, distilled DeiT-B/16, and Swin-B, denoted as ViT, DeiT, and Swin. ViT and DeiT maintain a constant-width token sequence, whereas Swin uses hierarchical spatial features. Table 6 collects the backbone and reader settings. The corresponding `timm` checkpoint (Wightman, 2019) identifiers are:

ViT: `vit_base_patch16_224.orig_in21k_ft_in1k`
DeiT: `deit_base_distilled_patch16_224.fb_in1k`
Swin: `swin_base_patch4_window7_224.ms_in1k`.

ViT implementation. ViT follows the readout architecture in the main paper.

DeiT adaptation. The distilled DeiT backbone retains both its class and distillation tokens, and both are included in the reader memory. The progressive query updates are otherwise unchanged. The final prediction averages the logits from the class and distillation heads. This averaged-logit prediction also provides the frozen teacher for exit training; the early-exit heads are not duplicated for the two prefix tokens.

Swin adaptation. At each configured exit, we flatten the intermediate spatial feature map into tokens and apply an exit-specific trainable linear projection into the common query space:

$$\widehat{\mathbf{X}}_e = \mathrm{Proj}_e\Big(\mathrm{sg}\Big(\mathrm{Flatten}\Big(\mathbf{X}^{(d_e)}\Big)\Big)\Big) \in \mathbb{R}^{D_q \times N_e}, \tag{13}$$

where N_e is the number of spatial tokens at exit e, D_q is the query dimension, and sg denotes stop-gradient. Detachment is applied before the projection so that its parameters remain trainable. Each reader attends to all projected spatial tokens.

Unlike ViT, Swin maintains hierarchical spatial representations. For Swin-B at 224×224 resolution, the first two blocks operate on $56 \times 56 = 3136$ tokens with feature dimension 128. If progressive accumulation were applied, all five active queries would read these high-resolution features. With query dimension $D_q = 384$, the reader cost at each of these blocks is approximately

$$3136(128)(384) + 2(3136)(384^2) + 2(5)(384^2) + 2(5)(3136)(384) \approx 1.09 \text{ GMAC}.$$

This already exceeds the cost of a first-stage Swin-B block. We therefore use non-accumulative, exit-local reading for Swin: each exit-specific query is updated only once, after its assigned block $d_e \in \{4, 8, 12, 16, 20\}$.

Exit-branch initialization. Query embeddings use a truncated-normal initializer with standard-deviation parameter 0.02. Exit modules use their own trainable projections and normalizations; they do not reuse the backbone's attention parameters.

A.3 Training and Checkpoint Selection

Training schedule. We use the objectives defined in the main paper, with optimization settings in Table 7. When Stage I is run, the final classifier is initialized for the target label space. The selected Stage-I backbone and classifier weights initialize Stage II, while the exit branches are initialized afresh. During Stage II, the backbone and final classifier are kept in evaluation mode; the Swin feature projections remain trainable as part of the exit branch.

Optimizer and checkpoint selection. Stage I selects the checkpoint with the highest full-depth validation accuracy; Stage II selects the checkpoint with the lowest validation exit-training loss. Both use the calibration subset defined in Appendix A.1. The same held-out subset is subsequently reused for threshold calibration. Stage II teacher logits are detached.

Randomness and repeated runs. We repeat training with seeds 2024, 2025, and 2026. For CIFAR-100 and Food-101, each run starts from the Stage-I checkpoint trained with the corresponding seed; ImageNet-1K runs use the same public pretrained checkpoint. Each seed controls the initialization of newly constructed exit-branch parameters, minibatch ordering, data partition and stochastic data augmentation. Thresholds are recalibrated for each trained checkpoint, and operating points are selected using the same rule across seeds.

A.4 Threshold Calibration and Evaluation

We follow the dynamic-threshold protocol of Kubaty et al. (2026). For E early exits and one final classifier, the target path proportions are

$$\pi_e(\gamma) = \frac{\exp(\gamma(e-1))}{\sum_{j=1}^{E+1} \exp(\gamma(j-1))}, \qquad e = 1, \ldots, E+1. \tag{14}$$

We use 101 logarithmically spaced magnitudes from 10^{-2} to 10^{2} with both signs, excluding zero, for 202 candidates before deduplicating identical routing outcomes.

Table 7: DeclareViT optimization settings. Panel (a) lists shared settings; panel (b) lists peak learning rates. CIFAR-100 and Food-101 are grouped because their schedules are identical. A dash denotes skipping Stage I and retaining the public pretrained backbone and original final classifier.

(a) Shared settings

Optimizer / weight decay	AdamW / 0.05
Epochs per executed stage	100
Global batch size	1,024 (256 per GPU)
Schedule	Cosine decay with 10 epochs of linear warmup
Minimum learning rate	10^{-6}
Precision	BF16 mixed precision
Distillation	$\alpha = 0.5, T = 4$
Exit loss weights	$\lambda_e = 1$
Training hardware	4 NVIDIA H100 80GB GPUs

(b) Peak learning rates

Dataset	Stage I			Stage II
	ViT	DeiT	Swin	All backbones
CIFAR-100 / Food-101	4×10^{-5}	8×10^{-5}	4×10^{-5}	4×10^{-4}
ImageNet-1K	–	–	–	8×10^{-4}

Thresholds are calibrated sequentially on the selection/calibration subset. For N_{cal} samples, exit e targets $a_e = \lfloor N_{\text{cal}} \pi_e(\gamma) \rfloor$ samples. Among samples that have not exited, its threshold is the a_e-th smallest entropy when this count lies strictly between zero and the remaining sample count. A zero target disables the exit, while a target at least as large as the remaining count accepts all remaining samples. Empty remaining sets require no further calibration. Inclusive threshold ties can produce more exits than targeted; unresolved samples use the final classifier.

Threshold quantiles are fitted using entropy scores without class labels. For each candidate, the resulting thresholds are applied unchanged to the final evaluation split. Computation is calculated from the realized routing counts, not the target path proportions. Budget-specific aggregation is documented with the additional accuracy–efficiency results.

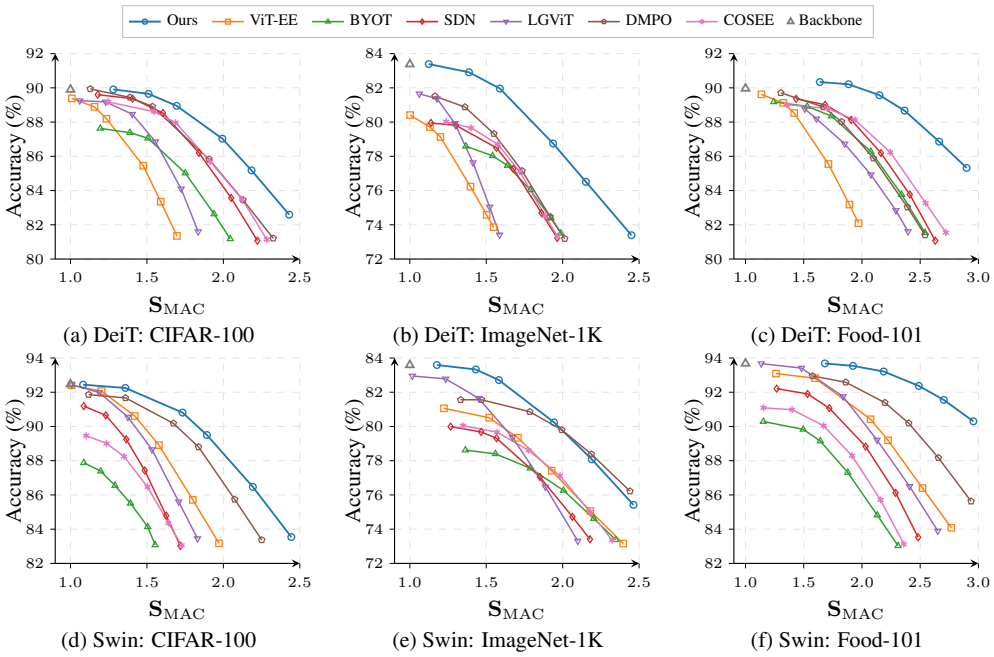
Representative operating points. For Tables 1 and 9, we construct the accuracy–S_{MAC} Pareto frontier from the final-evaluation results of the threshold sweep and manually select one representative Pareto-optimal operating point for each method. Thus, threshold fitting uses only the selection/calibration subset, whereas the representative points reported in Tables 1 and 9 are selected post hoc from the final-evaluation Pareto frontier.

B Additional Accuracy–Efficiency Results

We complement the operating-point results in Table 1 with calibration-targeted budget comparisons, multi-seed evaluation, and additional accuracy–efficiency curves.

B.1 Accuracy at Calibration-Targeted Speed-Ups

To compare methods under controlled computation targets, Table 8 reports final-evaluation performance at target $\mathbf{S}_{\text{MAC}}$ values of $1.50\times$, $2.00\times$, and $2.50\times$ across all nine dataset–backbone settings. For each method and target $\mathbf{S}_{\text{target}}$, we select on the calibration subset the candidate that maximizes calibration accuracy among those satisfying $\mathbf{S}_{\text{MAC,cal}} \geq \mathbf{S}_{\text{target}}$. The selected thresholds are then fixed and applied unchanged to the final evaluation split. Each cell reports the resulting final-evaluation accuracy together with the realized final-evaluation $\mathbf{S}_{\text{MAC}}$ in parentheses. The realized speed-up may differ from the nominal target because the calibration constraint need not be tight, the candidate set is discrete, and routing statistics can differ across splits. No final-evaluation statistic



(a) DeiT: CIFAR-100 (b) DeiT: ImageNet-1K (c) DeiT: Food-101

(d) Swin: CIFAR-100 (e) Swin: ImageNet-1K (f) Swin: Food-101

Figure 6: Accuracy–$\mathbf{S}_{\mathrm{MAC}}$ trade-offs for DeiT (top) and Swin (bottom) across three datasets.

is used for operating-point selection, and no interpolation is performed. EnViT is excluded because only externally reported operating points are available.

Across the resulting 27 dataset–backbone–budget settings, DeclareViT achieves the highest displayed final-evaluation accuracy in 25 settings and ranks second in the remaining two. It ranks first in all nine DeiT settings and all nine Swin settings, and in seven of nine ViT settings. In a few settings, dynamic early-exit accuracy slightly exceeds the corresponding full-depth accuracy, consistent with the overthinking phenomenon (Kaya et al., 2019), where additional computation can change an already correct intermediate prediction into an incorrect final prediction. Overall, these results show that DeclareViT's accuracy–efficiency advantage persists across backbones, datasets, and calibration-targeted compute budgets.

B.2 Multi-Seed Results

Table 9 reports three-seed results for DeclareViT and the evaluated baselines across three datasets and three backbones. DeclareViT maintains strong accuracy–efficiency trade-offs across different training seeds, with mean $\mathbf{S}_{\mathrm{MAC}}$ ranging from $1.58\times$ to $2.57\times$. With ViT, it achieves $90.14\pm0.11\%$ at $2.09\pm0.01\times$ on CIFAR-100, $80.56\pm0.04\%$ at $1.81\pm0.02\times$ on ImageNet-1K, and $90.28\pm0.11\%$ at $2.57\pm0.05\times$ on Food-101. The same stability is observed with DeiT and Swin, indicating that the accuracy–efficiency gains are robust to training randomness.

B.3 Additional Trade-off Curves

Figure 6 presents the complete accuracy–efficiency trade-offs for DeiT and Swin, complementing the ViT curves in Figure 4. Across the six additional dataset–backbone settings, DeclareViT generally forms the upper accuracy–efficiency frontier over the evaluated speed-up range.

Together, the calibration-targeted budget comparison, multi-seed evaluation, and complete trade-off curves show that DeclareViT's gains persist across computation budgets, training seeds, and inference operating points.

Table 8: Top-1 accuracy (%) at calibration-targeted $\mathbf{S}_{\mathrm{MAC}}$ on CIFAR-100, ImageNet-1K, and Food-101 using ViT, DeiT, and Swin. Operating points maximize calibration accuracy subject to $\mathbf{S}_{\mathrm{MAC,cal}} \geq \mathbf{S}_{\mathrm{target}}$ and are then evaluated unchanged on the final evaluation split. Each cell reports final-evaluation accuracy with the realized $\mathbf{S}_{\mathrm{MAC}}$ in parentheses. **Bold** and underlining indicate the best and second-best displayed accuracies, respectively. Actual speed-ups may differ from the targets.

Method	CIFAR-100			ImageNet-1K			Food-101		
	1.50×	**2.00×**	**2.50×**	**1.50×**	**2.00×**	**2.50×**	**1.50×**	**2.00×**	**2.50×**
ViT									
SDN	91.20% (1.52×)	86.45% (2.03×)	78.57% (2.54×)	80.70% (1.46×)	74.67% (1.95×)	66.50% (2.44×)	90.83% (1.61×)	88.16% (2.16×)	82.57% (2.72×)
BYOT	89.80% (1.51×)	85.03% (2.01×)	77.55% (2.51×)	80.03% (1.47×)	77.28% (1.94×)	71.97% (2.44×)	90.07% (1.58×)	87.30% (2.15×)	82.23% (2.73×)
ViT-EE	88.14% (1.51×)	79.75% (2.09×)	74.16% (2.52×)	77.27% (1.46×)	68.69% (2.04×)	63.16% (2.49×)	88.95% (1.62×)	82.80% (2.27×)	77.64% (2.74×)
LGViT	91.06% (1.51×)	84.23% (2.02×)	74.88% (2.50×)	76.93% (1.47×)	64.30% (1.99×)	54.70% (2.49×)	89.96% (1.58×)	85.07% (2.13×)	79.09% (2.57×)
COSEE	91.23% (1.53×)	87.91% (2.01×)	81.01% (2.54×)	80.98% (1.46×)	75.87% (1.95×)	68.64% (2.44×)	90.50% (1.58×)	89.09% (2.16×)	84.41% (2.74×)
DMPO	**91.80%** (1.52×)	90.23% (2.02×)	85.86% (2.56×)	**81.79%** (1.46×)	78.38% (1.95×)	72.45% (2.51×)	90.66% (1.63×)	89.72% (2.15×)	86.86% (2.72×)
Ours	91.60% (1.63×)	**90.43%** (2.03×)	**87.33%** (2.55×)	81.77% (1.44×)	**79.90%** (1.94×)	**76.53%** (2.45×)	**91.36%** (2.08×)	**91.17%** (2.22×)	**89.50%** (2.80×)
DeiT									
SDN	89.06% (1.51×)	84.07% (2.01×)	76.75% (2.51×)	79.32% (1.47×)	73.23% (1.96×)	65.33% (2.48×)	89.16% (1.60×)	86.19% (2.16×)	79.97% (2.72×)
BYOT	86.95% (1.52×)	81.63% (2.01×)	74.35% (2.54×)	78.39% (1.47×)	73.99% (1.96×)	66.70% (2.46×)	88.76% (1.59×)	85.44% (2.17×)	79.51% (2.72×)
ViT-EE	84.95% (1.51×)	75.31% (2.08×)	68.69% (2.50×)	75.16% (1.46×)	66.06% (2.05×)	60.33% (2.50×)	86.89% (1.61×)	78.29% (2.26×)	72.57% (2.70×)
LGViT	87.42% (1.51×)	78.18% (2.02×)	67.35% (2.50×)	76.56% (1.46×)	64.26% (1.99×)	54.72% (2.50×)	88.38% (1.58×)	84.50% (2.13×)	78.86% (2.58×)
COSEE	88.68% (1.52×)	84.77% (2.01×)	78.41% (2.50×)	79.34% (1.47×)	73.33% (1.97×)	65.22% (2.48×)	88.88% (1.65×)	86.91% (2.16×)	81.55% (2.72×)
DMPO	88.98% (1.52×)	84.56% (2.03×)	78.09% (2.59×)	80.12% (1.46×)	73.93% (1.96×)	65.95% (2.53×)	89.35% (1.58×)	85.53% (2.14×)	79.39% (2.73×)
Ours	**89.56%** (1.53×)	**86.48%** (2.05×)	**81.45%** (2.54×)	**82.74%** (1.43×)	**78.75%** (1.94×)	**73.39%** (2.45×)	**90.34%** (1.63×)	**89.28%** (2.22×)	**86.19%** (2.77×)
Swin									
SDN	87.43% (1.49×)	77.38% (1.95×)	64.03% (2.55×)	79.77% (1.44×)	76.42% (1.91×)	70.75% (2.41×)	91.76% (1.57×)	87.94% (2.13×)	80.00% (2.76×)
BYOT	84.14% (1.51×)	74.37% (1.99×)	61.41% (2.58×)	78.54% (1.47×)	76.64% (1.94×)	72.60% (2.43×)	89.56% (1.57×)	84.82% (2.13×)	77.99% (2.74×)
ViT-EE	90.09% (1.48×)	83.17% (1.97×)	73.08% (2.56×)	80.83% (1.43×)	77.42% (1.93×)	71.89% (2.52×)	92.89% (1.57×)	89.81% (2.15×)	84.08% (2.77×)
LGViT	90.02% (1.43×)	82.59% (1.88×)	73.48% (2.37×)	81.93% (1.42×)	76.47% (1.89×)	68.70% (2.40×)	93.34% (1.50×)	90.53% (1.98×)	85.22% (2.52×)
COSEE	86.47% (1.50×)	78.36% (1.97×)	66.27% (2.54×)	79.98% (1.45×)	77.67% (1.91×)	72.43% (2.41×)	90.53% (1.57×)	86.22% (2.13×)	77.93% (2.76×)
DMPO	91.32% (1.48×)	86.50% (2.01×)	80.18% (2.50×)	81.58% (1.43×)	80.10% (1.94×)	76.22% (2.44×)	92.96% (1.54×)	91.62% (2.14×)	88.17% (2.66×)
Ours	**92.03%** (1.46×)	**88.77%** (1.97×)	**83.54%** (2.44×)	**83.35%** (1.42×)	**80.57%** (1.90×)	**76.44%** (2.36×)	**93.67%** (1.53×)	**93.36%** (2.08×)	**91.96%** (2.59×)

C Wall-Clock Evaluation Protocol and Additional Results

Evaluation protocol. For the wall-clock evaluation in Section 5.4, we measure batch-one inference latency on an NVIDIA Quadro RTX 6000 GPU and an Intel Xeon Platinum 8480+ CPU using a single CPU thread. GPU inference uses FP16 autocast mixed precision, whereas CPU inference uses FP32. All models are evaluated in `eval` mode under `inference_mode` with PyTorch eager

Table 9: Multi-seed comparison on CIFAR-100, ImageNet-1K, and Food-101 using ViT, DeiT, and Swin. Top-1 accuracy (%) and $\mathbf{S}_{\mathrm{MAC}}$ are reported as the mean $\pm$ sample standard deviation over seeds 2024, 2025, and 2026. **Bold** and underlining mark the best and second-best results, respectively. Full-depth references are excluded from ranking. Full-depth reference rows reproduce the seed-2024 results from Table 1.

Method	CIFAR-100		ImageNet-1K		Food-101	
	Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑	**Acc. (%)** ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑	**Acc. (%)** ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
			ViT			
ViT	91.93	1.00×	81.77	1.00×	91.04	1.00×
SDN	88.26 ± 0.03	$1.89 \pm 0.01\times$	79.19 ± 0.01	$1.63 \pm 0.01\times$	89.19 ± 0.03	$2.03 \pm 0.01\times$
BYOT	88.31 ± 0.01	$1.64 \pm 0.06\times$	79.52 ± 0.00	$1.64 \pm 0.00\times$	88.74 ± 0.01	$1.86 \pm 0.07\times$
ViT-EE	88.65 ± 0.11	$1.50 \pm 0.05\times$	78.63 ± 0.06	$1.37 \pm 0.02\times$	88.71 ± 0.08	$1.70 \pm 0.10\times$
LGViT	88.20 ± 0.18	$1.74 \pm 0.02\times$	79.20 ± 0.02	$1.32 \pm 0.03\times$	89.83 ± 0.04	$1.62 \pm 0.04\times$
COSEE	88.48 ± 0.10	$1.96 \pm 0.02\times$	79.42 ± 0.06	$1.66 \pm 0.02\times$	89.43 ± 0.04	$2.05 \pm 0.08\times$
DMPO	89.99 ± 0.01	$1.88 \pm 0.32\times$	80.54 ± 0.04	$1.68 \pm 0.00\times$	89.60 ± 0.07	$2.30 \pm 0.08\times$
Ours	$\mathbf{90.14 \pm 0.11}$	$\mathbf{2.09 \pm 0.01}\times$	$\mathbf{80.56 \pm 0.04}$	$\mathbf{1.81 \pm 0.02}\times$	$\mathbf{90.28 \pm 0.11}$	$\mathbf{2.57 \pm 0.05}\times$
			DeiT			
DeiT	89.90	1.00×	83.38	1.00×	89.96	1.00×
SDN	88.52 ± 0.01	$1.59 \pm 0.03\times$	78.51 ± 0.03	$1.55 \pm 0.01\times$	89.02 ± 0.01	$1.71 \pm 0.03\times$
BYOT	87.07 ± 0.03	$1.52 \pm 0.01\times$	78.03 ± 0.00	$1.54 \pm 0.00\times$	88.36 ± 0.01	$1.62 \pm 0.12\times$
ViT-EE	88.19 ± 0.02	$1.23 \pm 0.01\times$	79.13 ± 0.03	$1.19 \pm 0.02\times$	88.57 ± 0.04	$1.43 \pm 0.01\times$
LGViT	88.46 ± 0.04	$1.43 \pm 0.02\times$	78.79 ± 0.11	$1.36 \pm 0.01\times$	88.71 ± 0.10	$1.40 \pm 0.09\times$
COSEE	88.62 ± 0.01	$1.49 \pm 0.07\times$	78.69 ± 0.02	$\mathbf{1.58 \pm 0.01}\times$	88.79 ± 0.02	$1.76 \pm 0.05\times$
DMPO	88.87 ± 0.01	$1.54 \pm 0.06\times$	81.00 ± 0.02	$1.33 \pm 0.02\times$	88.86 ± 0.05	$1.79 \pm 0.13\times$
Ours	$\mathbf{88.95 \pm 0.02}$	$\mathbf{1.64 \pm 0.05}\times$	$\mathbf{81.96 \pm 0.02}$	$\mathbf{1.58 \pm 0.01}\times$	$\mathbf{89.54 \pm 0.02}$	$\mathbf{2.13 \pm 0.02}\times$
			Swin			
Swin	92.47	1.00×	83.59	1.00×	93.68	1.00×
SDN	90.61 ± 0.04	$1.24 \pm 0.01\times$	79.31 ± 0.01	$1.55 \pm 0.01\times$	91.08 ± 0.02	$1.72 \pm 0.02\times$
BYOT	87.37 ± 0.09	$1.22 \pm 0.04\times$	78.40 ± 0.00	$1.50 \pm 0.05\times$	89.84 ± 0.01	$1.54 \pm 0.05\times$
ViT-EE	90.61 ± 0.01	$1.43 \pm 0.02\times$	80.49 ± 0.02	$1.52 \pm 0.01\times$	90.29 ± 0.12	$2.13 \pm 0.08\times$
LGViT	90.54 ± 0.01	$1.38 \pm 0.01\times$	81.22 ± 0.02	$1.51 \pm 0.00\times$	91.74 ± 0.07	$1.84 \pm 0.00\times$
COSEE	88.23 ± 0.06	$1.34 \pm 0.03\times$	79.65 ± 0.00	$1.54 \pm 0.06\times$	90.04 ± 0.01	$1.69 \pm 0.04\times$
DMPO	90.62 ± 0.06	$1.63 \pm 0.01\times$	81.59 ± 0.00	$1.52 \pm 0.07\times$	91.42 ± 0.10	$2.16 \pm 0.04\times$
Ours	$\mathbf{90.82 \pm 0.03}$	$\mathbf{1.72 \pm 0.02}\times$	$\mathbf{82.71 \pm 0.01}$	$\mathbf{1.59 \pm 0.01}\times$	$\mathbf{92.38 \pm 0.11}$	$\mathbf{2.47 \pm 0.04}\times$

execution. Inputs are preprocessed to 224×224 and resident on the target device before timing, excluding data loading, preprocessing, and host-to-device transfer.

For GPU measurements, we synchronize the device immediately before and after each forward pass and measure synchronized wall-clock latency with a host timer. Dynamic inference terminates immediately once an exit satisfies its entropy threshold; otherwise, computation continues to the final classifier. Each process is warmed up on 500 samples and then evaluated over the complete evaluation set. We repeat the measurement in three fresh processes and report the mean per-image latency and sample standard deviation.

All locally evaluated methods follow the same batch size, precision, warm-up, synchronization, and timing protocol while retaining their method-specific implementations. For DMPO, LoRA weights are merged into the corresponding linear layers before wall-clock evaluation.

Additional Results. As shown in Table 10, DeclareViT achieves GPU wall-clock speed-ups of $1.86\times$, $1.59\times$, and $2.26\times$, and CPU wall-clock speed-ups of $1.99\times$, $1.70\times$, and $2.44\times$ on CIFAR-100, ImageNet-1K, and Food-101, respectively. Wall-clock speed-ups are generally smaller than the corresponding MAC-based speed-ups because MAC accounting does not capture runtime overheads such as routing, synchronization, and kernel launches.

Table 10: Wall-clock evaluation of DeclareViT and the unbranched ViT-B/16 reference. Wall-clock time is reported as the mean ± standard deviation over three independent runs at batch size one, and $\mathbf{S}_{\mathrm{wall}}$ is measured relative to the corresponding unbranched ViT.

Dataset	Method	Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑	GPU		CPU	
				Wall-clock time ↓	$\mathbf{S}_{\mathrm{wall}}$ ↑	Wall-clock time ↓	$\mathbf{S}_{\mathrm{wall}}$ ↑
CIFAR-100	ViT	91.93	1.00×	14.56 ± 0.01 ms	1.00×	245.28 ± 1.27 ms	1.00×
	Ours	90.02	2.10×	7.84 ± 0.01 ms	1.86×	123.39 ± 1.11 ms	1.99×
ImageNet-1K	ViT	81.77	1.00×	14.62 ± 0.04 ms	1.00×	246.42 ± 3.85 ms	1.00×
	Ours	80.55	1.79×	9.20 ± 0.00 ms	1.59×	144.63 ± 0.70 ms	1.70×
Food-101	ViT	91.04	1.00×	14.56 ± 0.00 ms	1.00×	243.75 ± 1.92 ms	1.00×
	Ours	90.31	2.57×	6.45 ± 0.01 ms	2.26×	99.95 ± 0.65 ms	2.44×

Table 11: Further ablations on ImageNet-1K: (a) knowledge distillation, (b) independent vs. shared query states, (c) backbone K/V reuse, and (d) training protocol with ViT; (e) query accumulation with Swin. Per-exit accuracies are evaluated independently without dynamic routing, whereas Acc. denotes dynamic inference accuracy. **Bold** marks the best value for each metric within each panel.

(a) Knowledge distillation

Variant	Acc. ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
w/o KD	80.05	1.50×
w/ KD (ours)	**80.55**	**1.79×**

(b) Query states

Variant	Acc. ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
Shared	80.16	1.69×
Independent (ours)	**80.55**	**1.79×**

(c) K/V reuse

Variant	Acc. ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
Reused K/V	80.05	1.71×
Ours	**80.55**	**1.79×**

(d) Training protocol

Protocol	Per-Exit Accuracy (%)						Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
	@2	@4	@6	@8	@10	Final		
Joint-attached	55.85	67.53	74.00	77.69	80.53	81.39	80.12	1.67×
Joint-detached	**60.25**	69.94	75.59	78.67	80.74	81.50	80.23	1.72×
Two-stage (ours)	60.10	**70.08**	**75.71**	**79.03**	**81.20**	**81.77**	**80.55**	**1.79×**

(e) Query accumulation (Swin)

Readout	Per-Exit Accuracy (%)						Acc. (%) ↑	$\mathbf{S}_{\mathrm{MAC}}$ ↑
	@4	@8	@12	@16	@20	Final		
Accumulative	**59.80**	**71.39**	**77.02**	**80.13**	82.44	83.59	81.74	1.38×
Non-accumulative (ours)	55.22	68.05	75.47	79.99	**82.79**	83.59	**82.71**	**1.58×**

D Further Ablation Studies

D.1 Knowledge Distillation

Table 11(a) evaluates knowledge distillation from the frozen final classifier. Adding distillation improves dynamic accuracy from 80.05% to 80.55% and $\mathbf{S}_{\mathrm{MAC}}$ from $1.50\times$ to $1.79\times$, supporting the use of full-depth predictions to guide early-exit learning.

D.2 Independent vs. Shared Query States

The shared variant maintains a single recurrent query state used by all exits, whereas DeclareViT maintains a separate query state for each exit. As shown in Table 11(b), independent states improve dynamic accuracy from 80.16% to 80.55% and $\mathbf{S}_{\mathrm{MAC}}$ from $1.69\times$ to $1.79\times$. These results support maintaining exit-specific query states rather than sharing a single state across prediction depths.

D.3 Backbone Key/Value Reuse

Table 11(c) evaluates reusing backbone keys and values instead of computing them through separate, trainable exit-side projections. Backbone K/V reuse yields 80.05% dynamic accuracy at $1.71\times$ $\mathbf{S}_{\mathrm{MAC}}$, compared with 80.55% at $1.79\times$ for our default design. These results support learning a separate exit-side reading interface rather than directly reusing backbone K/V representations for early classification.

D.4 Further Analysis of Training Protocol

In Section 5.5, we compare Joint-attached, Joint-detached, and our two-stage protocol under matched architecture, initialization, exit objective, distillation settings, and exit-training budget. For both joint-training variants, we use separate parameter groups with a learning rate of 4.0×10^{-5} for the backbone and final classifier and 8.0×10^{-4} for the query branch. Here, we further report per-exit and full-depth accuracies. Joint-attached allows exit losses to update the backbone, Joint-detached blocks these gradients while still training the backbone jointly, and Two-stage additionally freezes the backbone and final classifier during exit optimization.

As shown in Table 11(d), Joint-attached reduces the full-depth accuracy from 81.77% to 81.39%. Applying stop-gradient improves the shallow exits, e.g., from 55.85% to 60.25% at block 2 and from 67.53% to 69.94% at block 4, while recovering the final accuracy to 81.50%. Two-stage further improves the deeper exits and fully preserves the original 81.77% full-depth accuracy, yielding 80.55% dynamic accuracy at $1.79\times$ $\mathbf{S}_{\mathrm{MAC}}$. These results support gradient isolation and are consistent with the benefits of training exits on fixed backbone representations.

D.5 Query Accumulation in Swin

Table 11(e) evaluates the Swin-specific choice of exit-local non-accumulative reading against progressive query accumulation. Although accumulation improves several standalone exit accuracies, particularly at shallow depths, it requires every active query to repeatedly attend to high-resolution intermediate feature maps. For example, the first two Swin-B blocks contain $56 \times 56 = 3136$ spatial tokens, and updating all five active queries with $D_q = 384$ incurs approximately 1.09 GMAC of reader computation per block, already exceeding the cost of a first-stage Swin-B block. Consequently, the accumulative variant achieves only 81.74% dynamic accuracy at $1.38\times$ $\mathbf{S}_{\mathrm{MAC}}$, whereas our exit-local design reaches 82.71% at $1.58\times$. These results complement the computational analysis in Appendix A.2 and show that, for hierarchical Swin representations, the improvement in shallow-exit accuracy does not compensate for the repeated high-resolution reader computation.

E Baseline Implementations and Reproduction

E.1 Baseline Methods

SDN. Shallow-Deep Networks (Kaya et al., 2019) attach internal classifiers to intermediate representations of a deep network. These classifiers allow confident inputs to terminate early and expose overthinking, where additional computation can turn a correct intermediate prediction into an incorrect final prediction.

BYOT. BYOT (Zhang et al., 2019) trains shallow classifiers through online self-distillation within a single network. The deepest classifier supplies soft prediction targets and feature supervision to earlier branches, complementing their ground-truth classification losses without requiring a separately trained teacher.

ViT-EE. ViT-EE (Bakhtiarnia et al., 2021) studies multi-exit vision transformers with different intermediate classifier designs, including MLP, convolutional, and Transformer-based branches. These branches provide alternative ways to extract class information from intermediate token representations for dynamic inference.

Table 12: Baseline implementation provenance and prediction locations. Indices are one-based; Swin indices count blocks across stages. For SDN, BYOT, ViT-EE, COSEE, and LGViT, the listed locations are early exits preceding the final classifier. DMPO lists all four predictors, including its full-depth predictor.

Method	Implementation	ViT / DeiT blocks	Swin blocks
SDN	Adapted linear-readout baseline	$2, 4, 6, 8, 10$	$4, 8, 12, 16, 20$
BYOT	Self-distillation adaptation	$2, 4, 6, 8, 10$	$4, 8, 12, 16, 20$
ViT-EE	Transformer-exit adaptation	$2, 4, 6, 8, 10$	$4, 8, 12, 16, 20$
COSEE	Visual adaptation of published objectives	$2, 4, 6, 8, 10$	$4, 8, 12, 16, 20$
LGViT	Native reimplementation	$4, \ldots, 11$	$4, 7, 10, 13, 16, 19, 22, 23$
DMPO	Adapted released implementation; DeiT/Swin extensions	$3, 6, 9, 12$	$6, 12, 18, 24$
EnViT	Externally reported	$2, 4, 6, 8, 10$	–

LGViT. LGViT (Xu et al., 2023) introduces heterogeneous exit heads tailored to network depth: local-perception heads use convolutional operators, while global-aggregation heads use spatially reduced self-attention. Its two-stage training first optimizes the multi-exit network and then freezes the backbone and final classifier while refining the exits through self-distillation.

COSEE. COSEE (He et al., 2025) addresses the mismatch between training and dynamic early-exit inference in pretrained language models. Its sample weighting mechanism simulates exiting under multiple thresholds and emphasizes classifiers near each sample's estimated exit. Online signal calibration further separates the exiting signals of correctly and incorrectly classified samples.

DMPO. DMPO (Luo et al., 2025) combines parameter-efficient backbone tuning with bypass-assisted, stage-wise predictors, using high-order predictors to strengthen early classification. Its representation-to-discrimination schedule progressively shifts supervision from later to earlier stages, while an additional loss factor regulates inter-stage optimization.

EnViT. EnViT (Dong et al., 2026) combines exit-aware structured dropout with self-distillation to improve multi-exit vision transformers. It drops entire Transformer layers on a per-sample basis, forming virtual student networks supervised by the full model. The exit-dependent dropout schedule is designed to reduce gradient conflicts between exit objectives.

E.2 Reproduction and Provenance

Our framework contains local implementations of SDN, BYOT, ViT-EE, LGViT, COSEE, and DMPO. We adapt their architectures and training procedures as described below; EnViT is included using externally reported results because its official implementation was not publicly available at the time of our experiments. Table 12 summarizes implementation origins and prediction locations.

Data and evaluation settings. Unless otherwise specified, all locally evaluated baselines use the same backbones, input resolution, data splits, preprocessing, and evaluation protocol as DeclareViT. Method-specific exit configurations and training objectives are detailed below.

Hyperparameter selection and adaptation policy. Our baseline adaptations are intentionally limited to the changes necessary for controlled comparison under a common evaluation protocol. All locally evaluated methods use the same backbone sources, dataset partitions, training/held-out split construction, preprocessing, and threshold-calibration and dynamic-evaluation procedure. Beyond these protocol-level alignments, method-specific architectures, objectives, and hyperparameters follow the original papers or released implementations whenever directly applicable and are not tuned on the final evaluation split. Modifications required to extend a method to additional backbones or to make it compatible with the unified evaluation setting are kept minimal and stated explicitly below.

Initialization. For SDN, BYOT, ViT-EE, and COSEE on CIFAR-100 and Food-101, ViT/DeiT/Swin are initialized from the dataset-specific Stage-I backbone and final-classifier checkpoints used by DeclareViT produced with the same seed.

Method-specific exits are constructed afresh; copied reader blocks are initialized from the loaded backbone. On ImageNet-1K, these methods load the corresponding pretrained backbone and original classifier. DMPO uses the same seed-matched backbone sources but discards the source classifiers and initializes its own predictors. LGViT has a separate initialization procedure described below.

SDN and ViT-EE. Our SDN adaptation uses LayerNorm followed by a linear classifier: ViT reads the class token, DeiT averages predictions from its class and distillation tokens, and Swin uses global average pooling. It does not reproduce the original SDN exit placement or progressive loss-weight schedule.

For ViT-EE, we use an independent backbone-compatible Transformer block followed by normalization and three-layer MLP readouts. ViT classifies the resulting class token, while Swin applies global average pooling before classification. Distilled DeiT uses separate MLPs for the class and distillation tokens and averages their logits. Each MLP has two hidden layers of width 768 with GELU activations. These exit readers do not write their outputs back into the backbone. Both baselines are jointly trained with hard-label cross-entropy:

$$\mathcal{L}_{\text{SDN/ViT-EE}} = 0.5\,\mathcal{L}_{\text{CE}}(\mathbf{z}_{\text{f}}, y) + 0.1 \sum_{e=1}^{5} \mathcal{L}_{\text{CE}}(\mathbf{z}_e, y). \tag{15}$$

No knowledge distillation is used for these two configurations.

BYOT. We adapt BYOT to the same lightweight readouts as SDN, adding training-only linear adapters for feature alignment. The final branch provides detached prediction and feature targets, while the backbone and classifiers are jointly optimized. We use a prediction-distillation weight of 0.3, temperature 3, and feature-loss coefficient 0.03. Our implementation uses soft-target cross-entropy without an additional T^2 factor, and feature alignment uses an unsquared L_2 norm. The implementation computes this norm over the global training batch. The final classification loss has weight 0.5; each early branch has weight 0.1, including its feature-alignment contribution. Feature adapters are excluded from inference.

COSEE. We implement COSEE's sample weighting and online signal calibration objectives on the SDN readout architecture and jointly train the backbone and classifiers. Normalized entropy is used as the exiting signal. We sample five thresholds per training step and linearly increase the sample-weighting sharpness from 0 to 1. The signal-calibration loss has coefficient 0.1 and margin 0.3. Sample-weighted classification includes all early exits and the final classifier; signal calibration applies only to early exits and averages over exits containing both correct and incorrect predictions. The sparse exit layout and vision backbones are adaptations of the original language-model setting.

LGViT. We reimplement LGViT using native `timm` backbones and its local-perception and global-aggregation head designs. We retain eight early exits, with four local-perception heads followed by four global-aggregation heads. Stage I alternates between final-classifier cross-entropy and depth-weighted joint cross-entropy. Stage II freezes the backbone and final classifier and trains the exits with prediction-distillation weight 0.3 and temperature 2. For ViT/DeiT, we additionally retain the heterogeneous and homogeneous feature-distillation losses, each weighted by 0.01. Following the released Swin Stage-II implementation, our Swin adaptation uses prediction distillation without these feature losses. The prediction-distillation KL term is averaged over both samples and classes before multiplication by T^2. For CIFAR-100 and Food-101, all three backbones execute both stages from their corresponding pretrained backbone initialization; ImageNet-1K is likewise configured to run both stages. We retain the released learning rate of 5×10^{-5} and the method-specific loss coefficients, while using 100 epochs per locally executed stage under our common training protocol.

DMPO. We adapt the released DMPO implementation and extend its token and spatial-feature interfaces to distilled DeiT and Swin. We retain rank-8 LoRA (Hu et al., 2022), rank-8 predictor-

Table 13: Analytical cumulative MACs per image for DeclareViT with ViT-B/16 on ImageNet-1K at 224×224 resolution. All numerical cost columns are in GMACs. Backbone includes patch embedding and the reached Transformer blocks. Readers include all active-query updates up to termination. Readouts include all evaluated exit-side FFNs and classifiers, and the final classifier when used. Rounded components may not sum exactly to the rounded total.

Path	Block	Backbone	Readers	Readouts	Total
Exit 1	2	3.0235	0.4796	0.0055	3.5086
Exit 2	4	5.9314	0.9562	0.0110	6.8986
Exit 3	6	8.8393	1.4299	0.0165	10.2857
Exit 4	8	11.7472	1.9006	0.0219	13.6698
Exit 5	10	14.6552	2.3684	0.0274	17.0510
Dynamic final	12	17.5631	2.3684	0.0282	19.9596
Unbranched ViT	12	17.5631	0.0000	0.0008	17.5638

specific bypasses, high-order predictors at the first two stages, and trainable prediction scales initialized to one. The four predictors include three early exits and a full-depth predictor; the original classifier is not used as an additional fallback. Pretrained core weights remain frozen, while LoRA, bypasses, predictors, and prediction scales are optimized. Following the default setting reported in the DMPO paper, the representation-to-discrimination loss weights interpolate linearly from $(0.01, 0.01, 1, 2)$ to $(1.5, 1, 0.1, 0.1)$. We use the paper-reported default consistently across all experiments. We also retain the sigmoid loss factor based on the preceding predictor's per-sample cross-entropy. We use the released learning rates of 5×10^{-4} for CIFAR-100/Food-101 and 10^{-4} for ImageNet-1K, while using a common 100-epoch training budget under our local evaluation protocol. DeiT and Swin are backbone extensions of the released ViT implementation.

Optimization. Local training uses AdamW with weight decay 0.05, a global batch size of 1024, and ten epochs of linear warmup followed by cosine decay. For SDN, BYOT, ViT-EE, and COSEE, we use the same grouped learning rate scheme as DeclareViT for the backbone and exit modules. Unless otherwise stated above, locally trained methods use a 100-epoch budget. All locally trained methods use BF16 mixed precision and are trained on four NVIDIA H100 80GB GPUs.

Dynamic evaluation and computation accounting. Local models use the sequential entropy-threshold calibration procedure in Appendix A, with the number of prediction paths set separately for each method. Thresholds are fitted on the held-out training subset and applied unchanged to the final evaluation split. The sweep uses 202 signed logarithmically spaced γ values. Expected MACs include the backbone computation and all branch operations executed before termination. MAC speed-up is measured relative to the corresponding unbranched full-depth backbone, including its classifier.

Externally reported EnViT results. We use the ViT-B/16 results from Table 2 of Dong et al. (2026) as an external reference. EnViT is not retrained or recalibrated in our framework, and its reported operating points are not aligned to the local γ grid. Its results retain the original authors' experimental settings and are therefore separate from comparisons under our local protocol.

F MAC Accounting

Counting convention. We count one multiply–accumulate pair as one MAC. Our ViT estimator includes the patch-embedding convolution, linear projections, and attention matrix products. Bias additions, LayerNorm, softmax, nonlinear activations, residual additions, pooling reductions, and routing comparisons are excluded. All costs below are per image, with $1\,\mathrm{GMAC} = 10^9$ MACs.

Cumulative inference paths. The cost C_j is the cumulative computation required to reach prediction path j. It includes all backbone blocks, reader operations, and earlier exit readouts executed before termination. For DeclareViT, active queries are updated until their designated exits and incur no further cost after being discarded. C_{base} denotes the unbranched full-depth model cost.

F.1 ViT-B/16 EXAMPLE

We consider DeclareViT with ViT-B/16 on ImageNet-1K at 224×224 resolution. The patch size is $p = 16$, giving $N = (224/16)^2 = 196$ patch tokens and $M = N + 1 = 197$ backbone and memory tokens, including [CLS]. The backbone has $L = 12$ blocks and embedding dimension $D = 768$; both the backbone and exit-side FFNs have hidden dimension $D_{\text{ffn}} = 3072$. There are $C = 1000$ classes and five exits at $\{2, 4, 6, 8, 10\}$. This example uses progressive accumulation, segment-shared readers with eight attention heads, and an exit-side FFN applied only at prediction depths.

Patch embedding and backbone blocks. Patch embedding is performed once, with cost

$$c_{\text{patch}} = N(3p^2)D = 115{,}605{,}504. \tag{16}$$

For each backbone block, the Q/K/V projections cost $3MD^2$, the attention output projection costs MD^2, the two attention matrix products cost $2M^2D$, and the two FFN linear layers cost $2MDD_{\text{ffn}}$. Thus,

$$\begin{aligned} c_{\text{block}} &= 4MD^2 + 2M^2D + 2MDD_{\text{ffn}} \\ &= 464{,}781{,}312 + 59{,}610{,}624 + 929{,}562{,}624 \\ &= 1{,}453{,}954{,}560. \end{aligned} \tag{17}$$

The final classifier costs $c_{\text{final}} = DC = 768{,}000$ MACs. The unbranched reference therefore has cost

$$\begin{aligned} C_{\text{base}} &= c_{\text{patch}} + Lc_{\text{block}} + c_{\text{final}} \\ &= 17{,}563{,}828{,}224 \approx 17.5638\,\text{GMAC}. \end{aligned} \tag{18}$$

Cross-attention readers. Let $a_\ell = |\{e : \ell \le d_e\}|$ be the number of active queries at block ℓ. For the configured exits,

$$(a_1, \ldots, a_{12}) = (5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 0, 0). \tag{19}$$

At each block, the reader processes all active exit queries together. Because the memory-side LayerNorm and the K/V projection parameters are common to these queries, the normalized memory and its K/V projections are computed once per block, rather than once per exit. The K/V projections therefore cost $2MD^2$. The query and output projections cost $2a_\ell D^2$, while the two attention products cost $2a_\ell MD$. These terms already sum over all heads; the head count introduces no additional multiplicative factor at fixed total dimension D. The reader cost is therefore

$$r_\ell = \begin{cases} 2MD^2 + 2a_\ell D^2 + 2a_\ell MD, & a_\ell > 0, \\ 0, & a_\ell = 0. \end{cases} \tag{20}$$

Numerically, $r_\ell = 232{,}390{,}656 + 1{,}482{,}240a_\ell$ when $a_\ell > 0$. For example, each of the first two blocks incurs $239{,}801{,}856$ reader MACs for five queries, not five times the cost of a separate one-query reader. Across all ten reader-equipped blocks,

$$\sum_{\ell=1}^{10} r_\ell = 10(232{,}390{,}656) + 30(1{,}482{,}240) = 2{,}368{,}373{,}760. \tag{21}$$

Sharing projection parameters across blocks does not reuse their projected features: each block has different memory and query states, so its matrix products are counted separately.

Exit readouts. Each exit applies its FFN to one query, followed by a linear classifier:

$$c_{\text{exit}} = 2DD_{\text{ffn}} + DC = 4{,}718{,}592 + 768{,}000 = 5{,}486{,}592. \tag{22}$$

The shared FFN is counted once at each evaluated exit, not once per image regardless of how many exits are evaluated.

Static path costs. For early exit $e \in \{1, \dots, E\}$,

$$C_e = c_{\text{patch}} + d_e c_{\text{block}} + \sum_{\ell=1}^{d_e} r_\ell + e c_{\text{exit}}. \tag{23}$$

For example, a sample terminating at block 2 has already updated all five queries at blocks 1 and 2 and evaluated the first readout:

$$\begin{aligned} C_1 &= 115{,}605{,}504 + 2(1{,}453{,}954{,}560) \\ &\quad + 2(239{,}801{,}856) + 5{,}486{,}592 \\ &= 3{,}508{,}604{,}928. \end{aligned} \tag{24}$$

For a sample reaching the final classifier, path $E + 1$ has cost

$$\begin{aligned} C_{E+1} &= c_{\text{patch}} + L c_{\text{block}} + \sum_{\ell=1}^{d_E} r_\ell + E c_{\text{exit}} + c_{\text{final}} \\ &= 19{,}959{,}634{,}944 \approx 19.9596\,\text{GMAC}. \end{aligned} \tag{25}$$

Table 13 lists the cumulative costs. They describe termination paths of the complete five-exit model, not separately reconfigured single-exit networks. The dynamic final path costs approximately 13.64% more than the unbranched reference because it evaluates all earlier branches.

G Further Gradient Conflict Analysis

We extend Sections 3.2 and 5.3 across backbone depths and shared exit-side modules at epochs 1, 10, and 100, complementing conflict rates with loss-weighted gradient norms.

Diagnostic protocol. At each checkpoint, we sample 57 minibatches with a global batch size of 1024. Objective-specific gradients are computed from the same augmented samples and forward pass, without intervening optimizer updates, and averaged across workers before analysis. Conflict rates measure the fraction of valid comparisons with negative cosine, as defined in Eq. 5. Valid comparisons require both gradients to satisfy $\|g\|_2^2 > \epsilon(\delta^{-1} - 1)$, with $\epsilon = 10^{-12}$ and $\delta = 10^{-5}$. We additionally report the minibatch-mean loss-weighted gradient norm $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$, where w_i is the training-loss weight and θ_g denotes the measured parameter group.

SDN. Figures 7–9 show that early–final conflicts occur across multiple backbone depths initially and persist in shallow blocks at later checkpoints. Importantly, in blocks 1–2, the earliest-exit objective has a mean loss-weighted gradient norm comparable to or larger than that of the final objective at later checkpoints. Thus, persistent directional conflicts coexist with non-negligible average exit gradients, supporting our motivation for isolating backbone feature refinement from early-exit supervision.

DeclareViT. Figures 10–12 show generally low inter-exit conflict rates in the shared cross-attention projections and exit FFN, alongside non-negligible average gradient norms. These measurements characterize residual interactions within the exit branches, while the frozen, read-only design prevents exit learning from altering the backbone. Since SDN and DeclareViT are measured on different parameter groups and training objectives, these results are complementary diagnostics rather than a controlled comparison of conflict reduction.

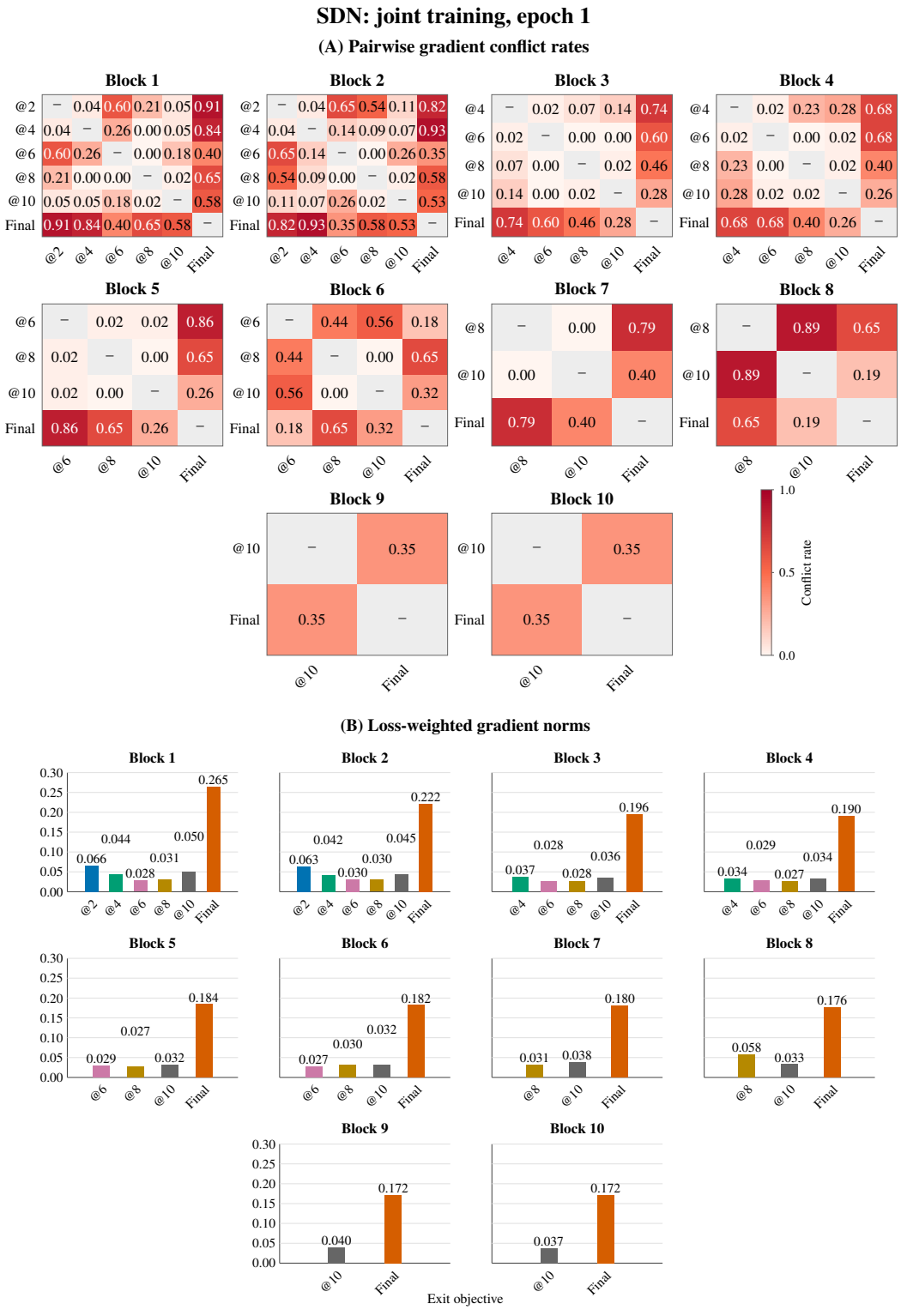


Figure 7: Block-wise gradient diagnostics for the [CLS]-based SDN adaptation at joint-training epoch 1. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$. Final denotes the full-depth objective.

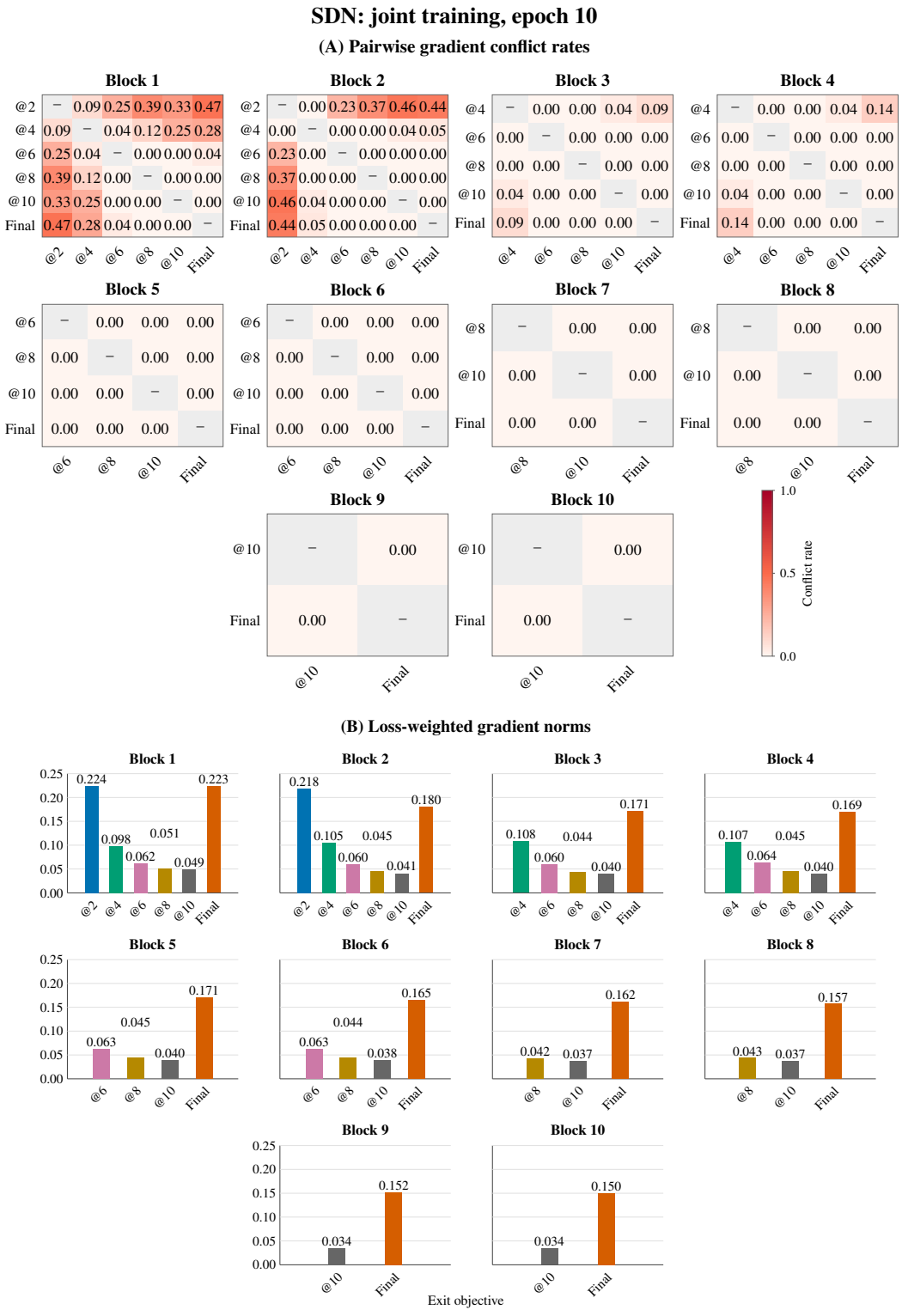


Figure 8: Block-wise gradient diagnostics for the [CLS]-based SDN adaptation at joint-training epoch 10. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$. Final denotes the full-depth objective.

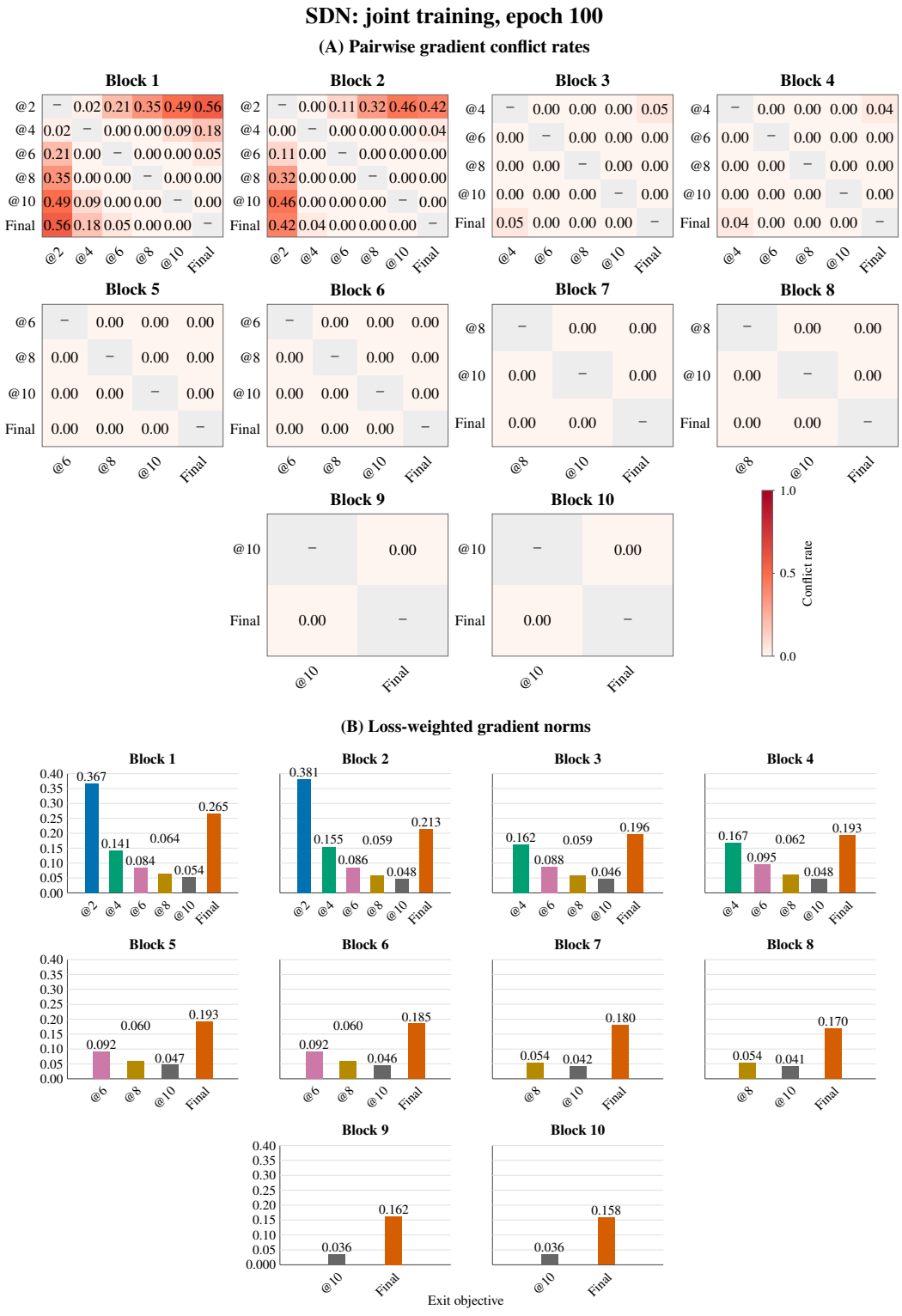


Figure 9: Block-wise gradient diagnostics for the [CLS]-based SDN adaptation at joint-training epoch 100. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$. Final denotes the full-depth objective.

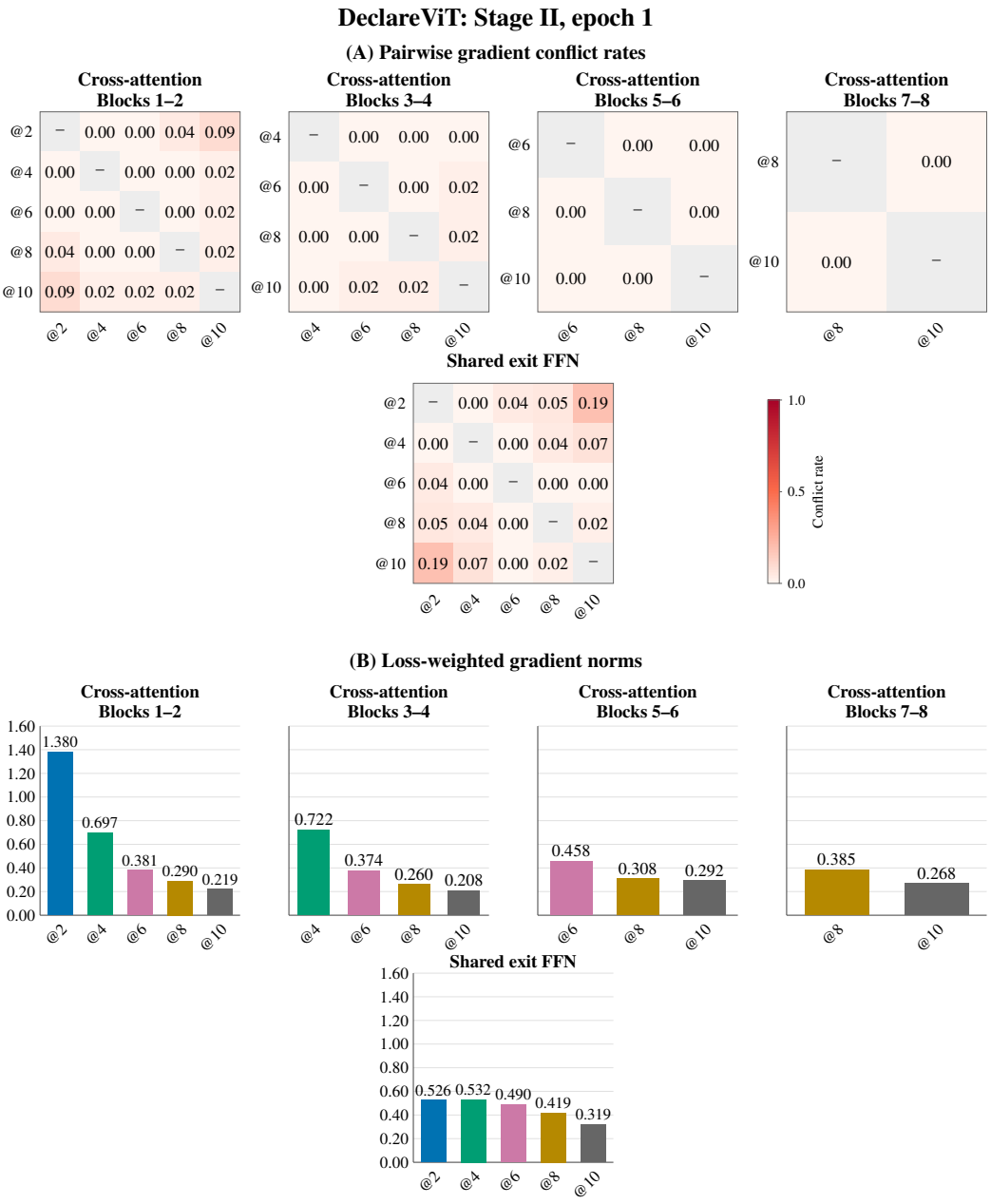


Figure 10: Gradient diagnostics for DeclareViT's segment-shared cross-attention projections and shared exit FFN at Stage II epoch 1. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$.

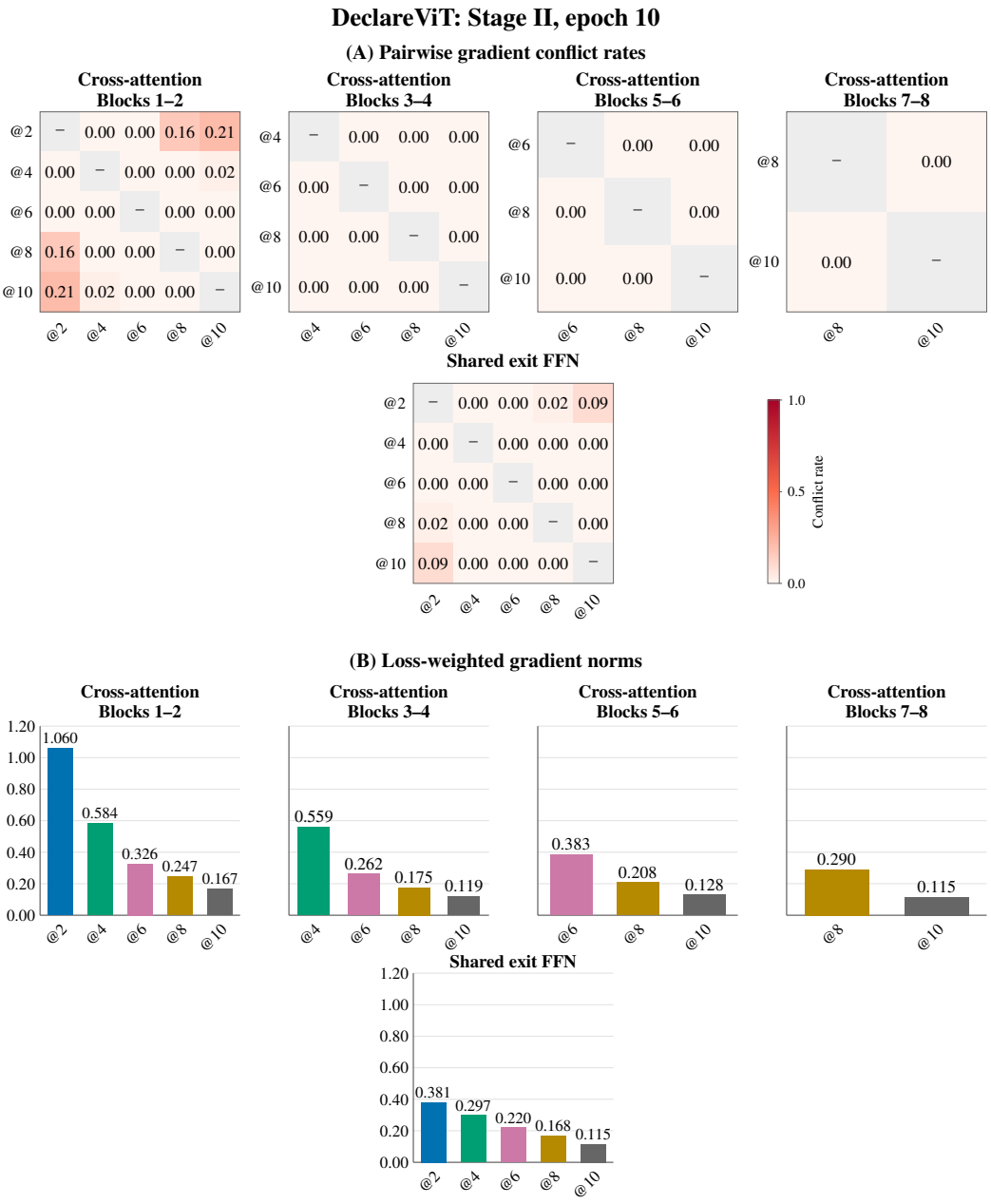


Figure 11: Gradient diagnostics for DeclareViT's segment-shared cross-attention projections and shared exit FFN at Stage II epoch 10. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$.

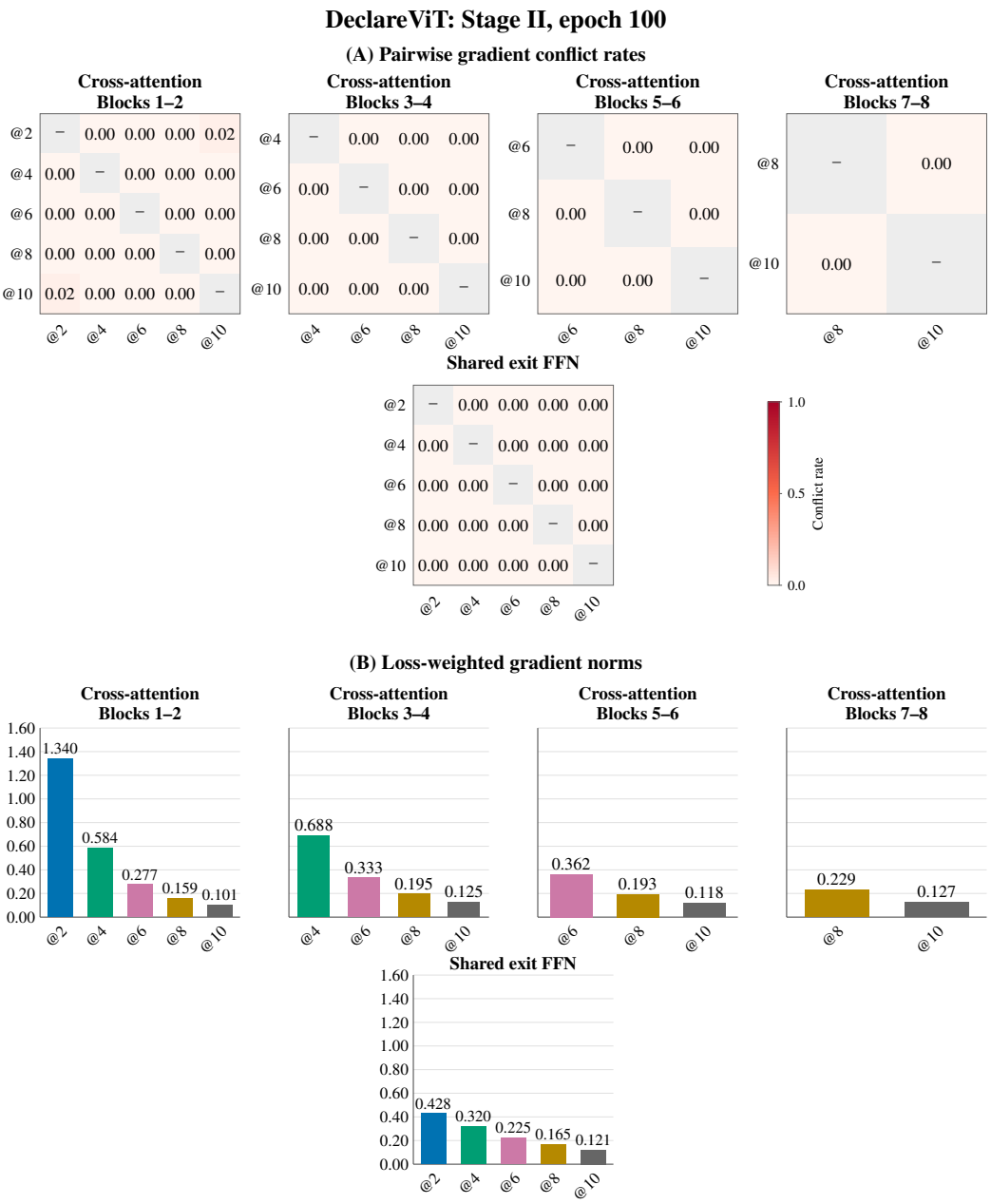


Figure 12: Gradient diagnostics for DeclareViT's segment-shared cross-attention projections and shared exit FFN at Stage II epoch 100. Top: pairwise gradient conflict rates. Bottom: mean loss-weighted gradient norms, $\|w_i \nabla_{\theta_g} \mathcal{L}_i\|_2$.

H ATTENTION VISUALIZATION OF [CLS] AND EXIT QUERIES

Figures 13, 14, and 15 visualize the patch attention of the backbone [CLS] token and the corresponding exit query at different exit depths. For each attention module, we aggregate the multi-head attention weights by averaging them across all heads and retain only the weights assigned to patch tokens. For each sample, the first two rows show the attention maps of [CLS] and the exit query, respectively, while the third row shows their pointwise difference (*Query minus CLS*). Red indicates regions receiving greater relative attention from the exit query, whereas blue indicates regions receiving greater relative attention from [CLS].

The examples show that the exit queries and backbone [CLS] token can exhibit different spatial attention patterns. In the shown cases, exit-query attention is more concentrated on object-related regions, while the [CLS] attention evolves across backbone depth. These qualitative observations are consistent with the intended separation between the dedicated early-classification state and the backbone representation in DeclareViT.

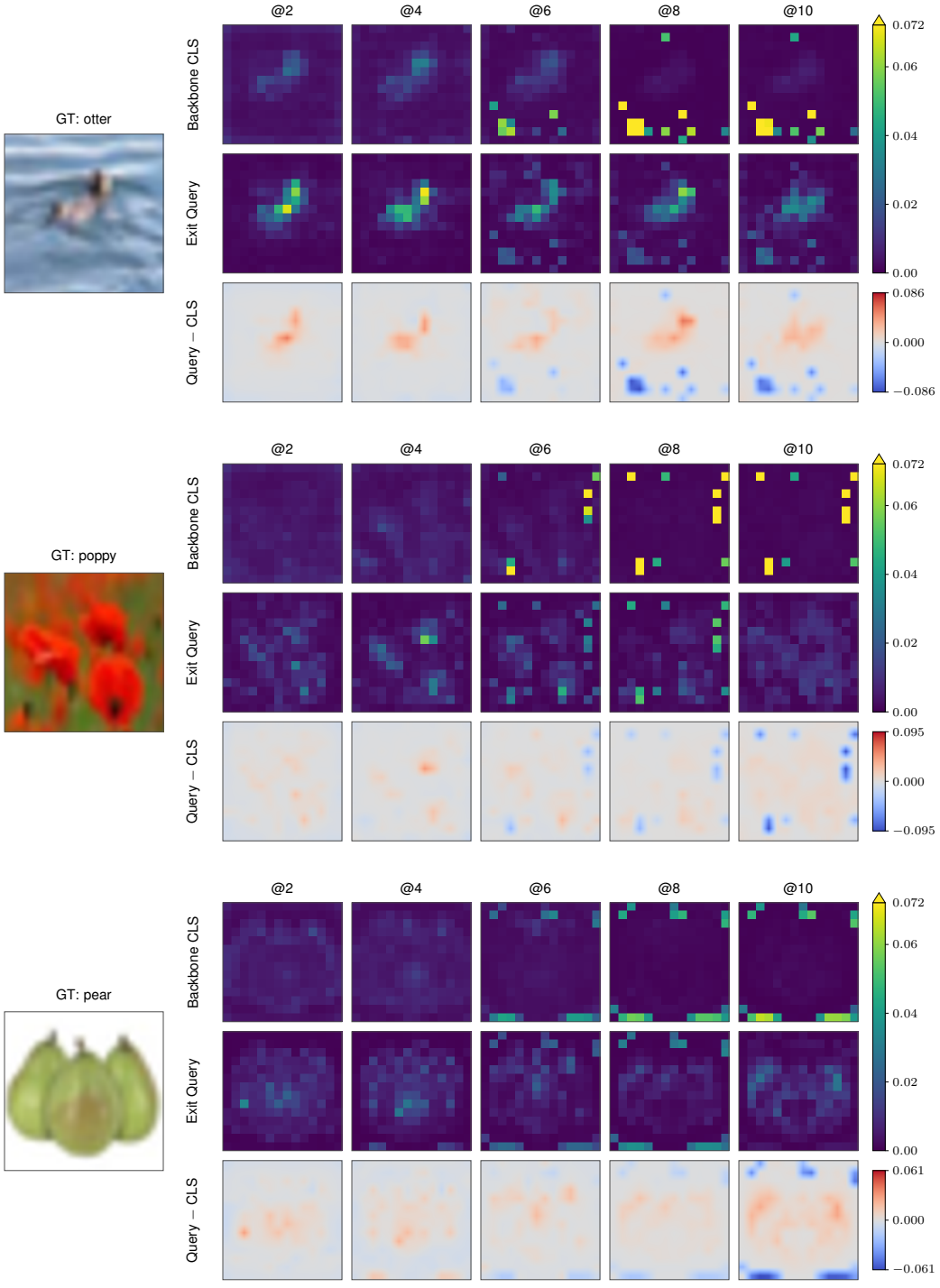


Figure 13: Attention visualization for CIFAR-100 test samples. Within each sample block, the input is on the left; the three rows on the right show attention for Backbone CLS, Exit Query, and Query minus CLS.

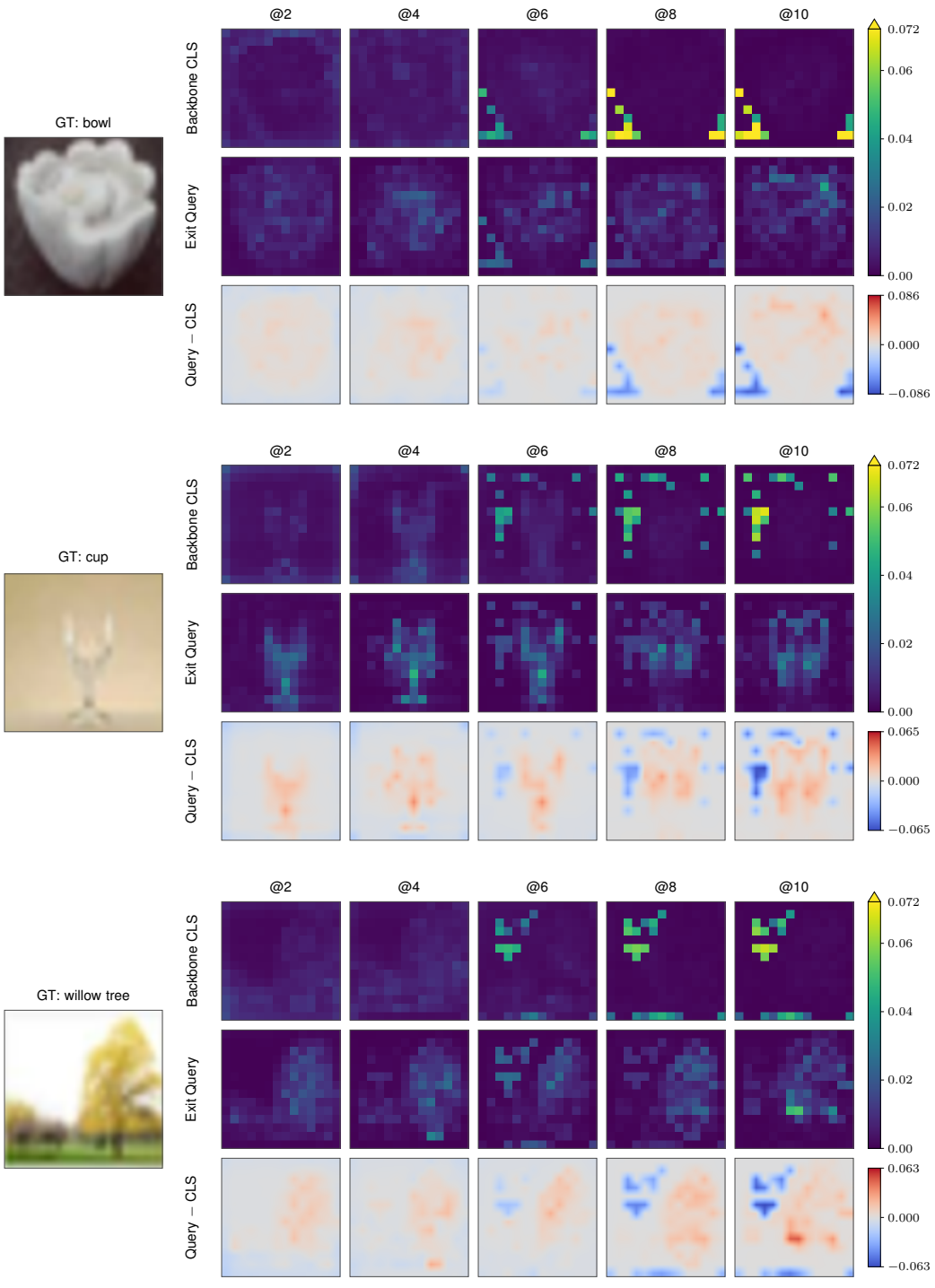


Figure 13: Attention visualization for CIFAR-100 (continued).

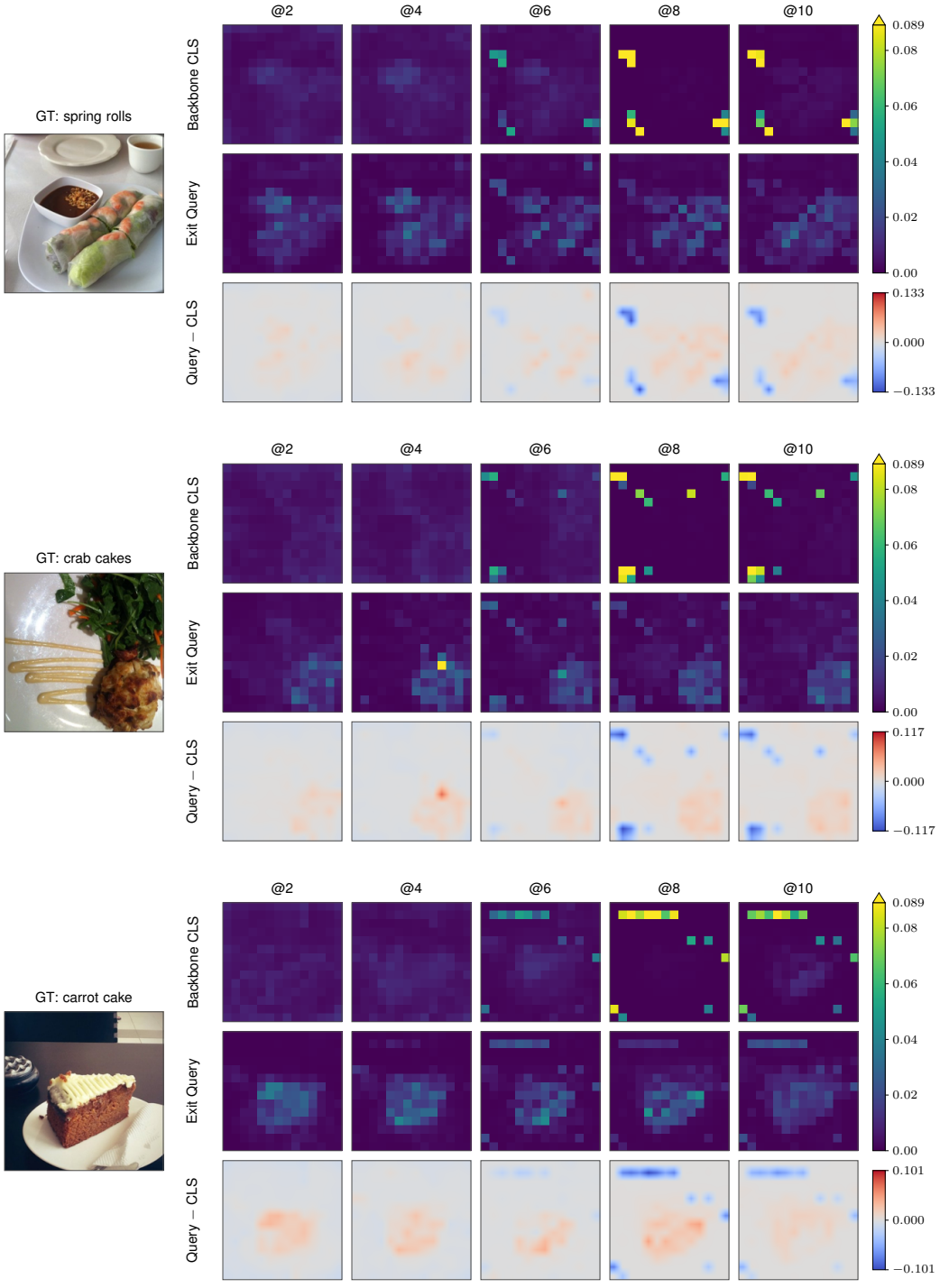


Figure 14: Attention visualization for Food-101 test samples. Within each sample block, the input is on the left; the three rows on the right show attention for Backbone CLS, Exit Query, and Query minus CLS.

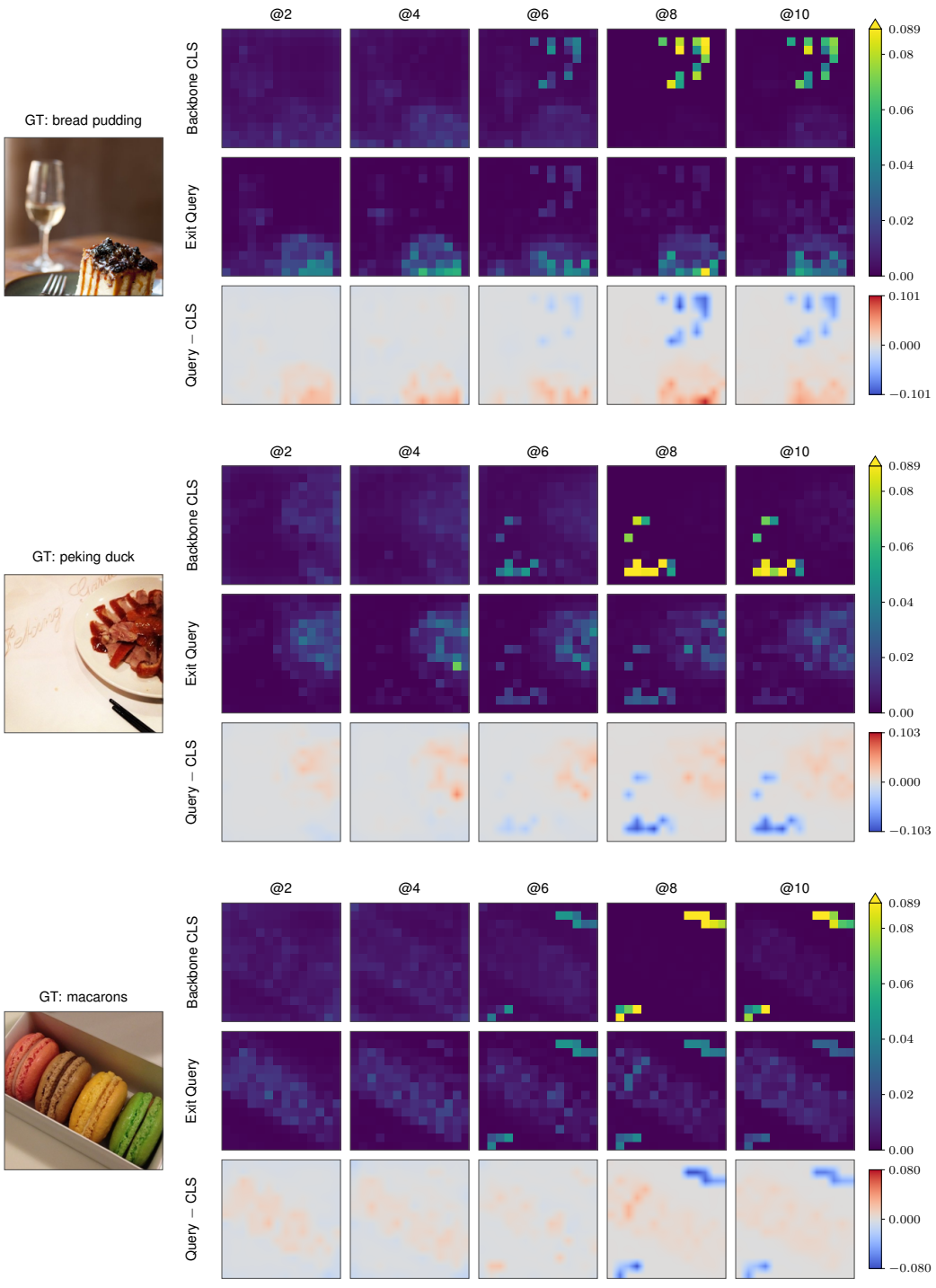


Figure 14: Attention visualization for Food-101 (continued).

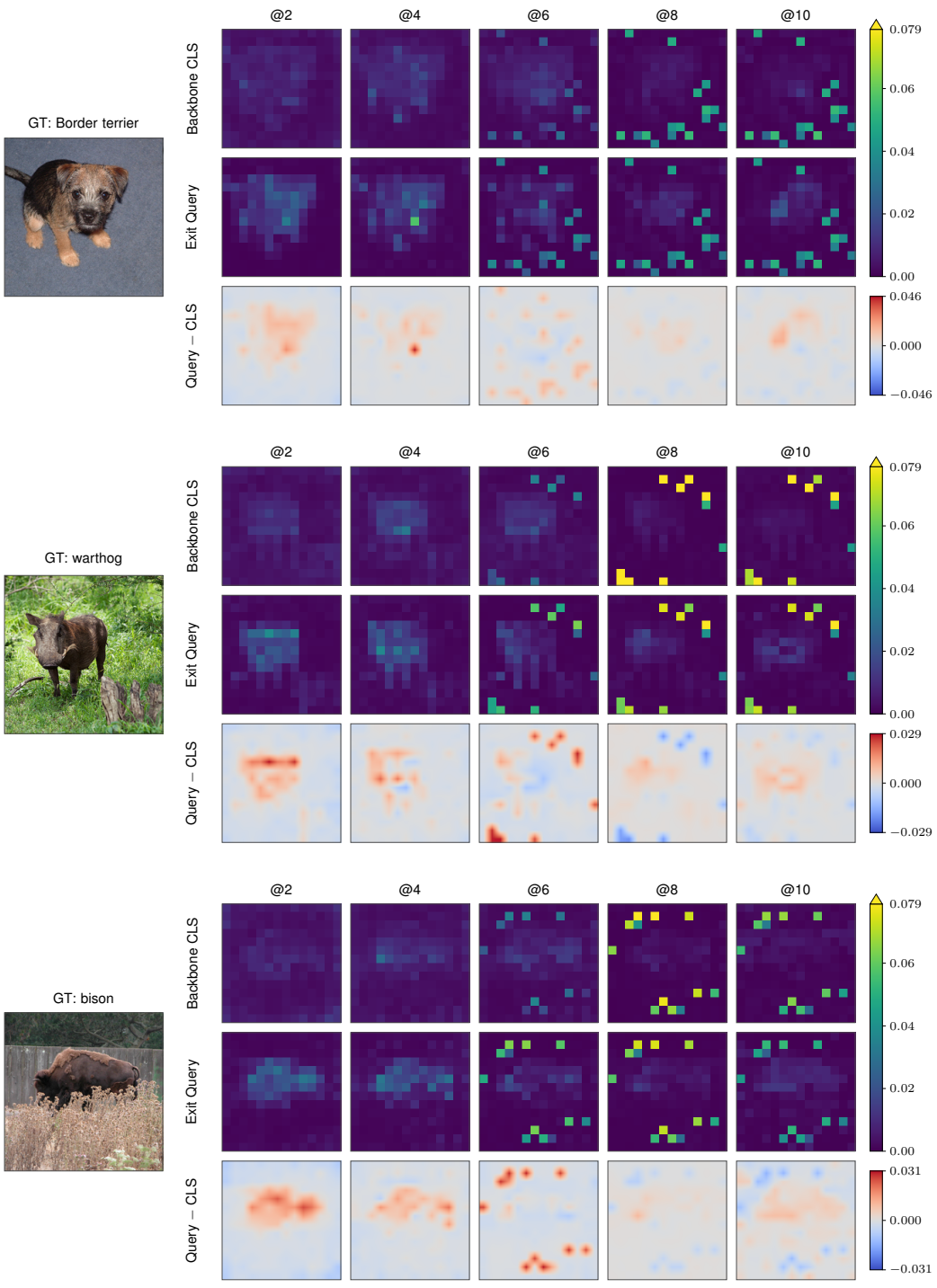


Figure 15: Attention visualization for ImageNet-1K validation samples. Within each sample block, the input is on the left; the three rows on the right show attention for Backbone CLS, Exit Query, and Query minus CLS.

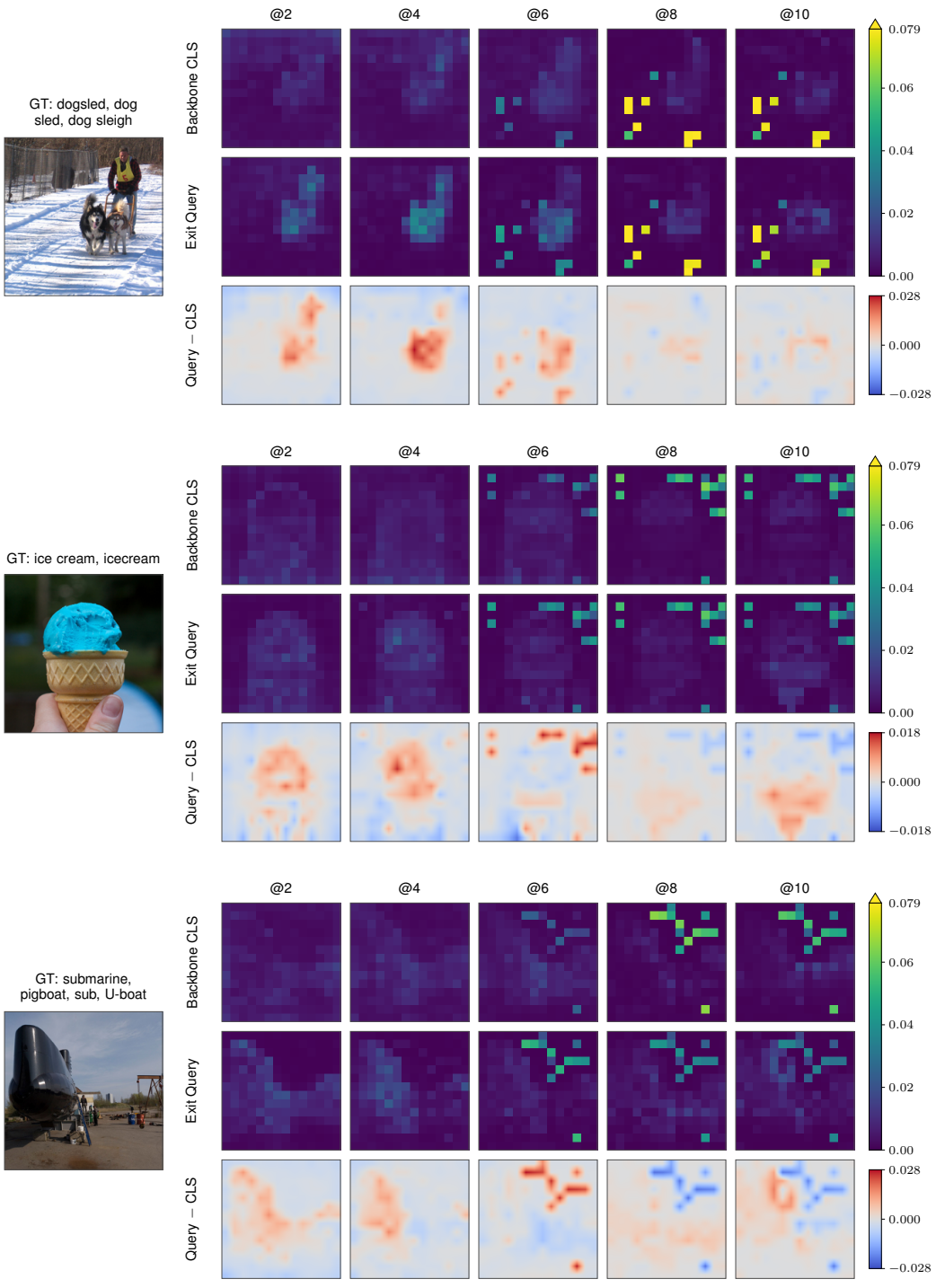


Figure 15: Attention visualization for ImageNet-1K (continued).