

## Article

# MoR–Swin: Efficient Vision Transformer Using Mixture of Recursions

Yongbao Ai , Tianxiang Gao, Zhipeng Lin, Longqi Yang and Qingyu Chang

Academy of Military Science, Institute of Military Intelligence, Beijing 100091, China; gaotxnudta@163.com (T.G.); linzhipeng13@nudt.edu.cn (Z.L.); seriousthing@yeah.net (L.Y.); changqingyu@mail.ecust.edu.cn (Q.C.)

\* Correspondence: aybjackai@163.com

## Abstract

Vision Transformers, especially Swin Transformer, have become default backbones for various vision tasks but suffer from high memory consumption and training costs. This letter proposes MoR–Swin, a novel architecture that integrates Mixture of Recursions (MoR) into Swin Transformer. An adaptive token-level recursion mechanism dynamically allocates computational depth based on semantic complexity. A recursive window attention module and a lightweight router with load balancing loss are introduced. Extensive experiments on ImageNet classification, COCO detection, and ADE20K segmentation show that MoR–Swin reduces parameters by about 50% and accelerates inference up to twofold at a modest accuracy cost (within about 0.5 points of Swin-B on ImageNet-1K). It provides a new technical pathway for optimizing Vision Transformer models, significantly enhancing their applicability in resource-constrained environments.

**Keywords:** Swin Transformer; mixture of recursions; adaptive computation; parameter efficiency

## 1. Introduction

Vision Transformer (ViT) [1,2] ended the dominance of CNNs in 2021. Swin Transformer [2,3] introduced hierarchical shifted-window attention, reducing complexity to  $\mathcal{O}(N)$  while achieving state-of-the-art results in image classification, object detection, and semantic segmentation. SwinV2 [4] later scaled the architecture to 3 billion parameters. However, the growing model size brings prohibitive memory consumption and training cost, limiting deployment in resource-constrained environments.

Sparse Mixture of Experts (MoE) [5–7] has reduced the cost of scaling in natural language processing by activating only a subset of parameters per input. A large number of MoE-based large language models [8–12] have demonstrated the effectiveness of conditional computation. However, applying sparse MoE to vision is limited by training instability, expert load imbalance, and communication bottlenecks. Recently, Liu et al. [13] proposed Mixture of Recursions (MoR), which replaces sparse expert routing with adaptive token-level recursion depths, offering a parameter-efficient alternative.

Beyond MoE, a parallel line of work pursues adaptive computation directly in vision transformers. Along the *spatial* dimension, DynamicViT [14] and EViT [15] prune or reorganize less informative tokens, and ToMe [16] merges similar tokens to shorten the sequence; A-ViT [17] halts token computation early via learned halting scores. These methods reduce the token count  $N$  but discard information irreversibly. Along the *width* dimension, sparse MoE increases parameters with the number of experts and suffers from load imbalance. In contrast, MoR–Swin operates along the *depth* dimension: it retains all



Academic Editors: Pier Luigi Mazzeo, Alessandro Bruno, Qiong Liu and Nikolaos Mitianoudis

Received: 15 June 2026

Revised: 28 July 2026

Accepted: 6 August 2026

Published: 12 August 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

tokens and adaptively allocates the number of recursive refinement steps per token under a single set of shared parameters. This design preserves full spatial information, keeps the parameter count nearly constant, and naturally matches the spatial heterogeneity of images (simple background regions vs. complex foreground textures), making it particularly well suited to hierarchical window attention.

Inspired by the design philosophy of the MoR architecture, in this letter, we integrate MoR into Swin Transformer and propose MoR-Swin. Our contributions are summarized as follows:

- We propose MoR-Swin, the first architecture that integrates Mixture of Recursions into Swin Transformer, enabling adaptive token-level computational depth allocation based on semantic complexity.
- We introduce a recursive window attention module and a lightweight router with a load balancing loss, together with memory optimization strategies that substantially reduce memory usage.
- Extensive experiments on ImageNet classification, COCO detection, and ADE20K segmentation show that MoR-Swin reduces parameters by about 50% and accelerates inference up to twofold at a modest accuracy cost (within about 0.5 points of Swin-B on ImageNet-1K).
- Analyzing MoR's inherent advantages over MoE in vision tasks (e.g., training stability and cross-resolution transfer), discussing its limitations, and outlining future research directions for adaptive recursive computation in broader visual tasks.

## 2. Method

### 2.1. Overall Architecture

MoR-Swin retains the hierarchical pyramid of Swin Transformer (four stages). The key modification is that standard Swin blocks are replaced with Recursive Mixture Blocks. As illustrated in Figure 1, each block contains recursive window attention (RWA) and a dynamic router that decides for each token whether to recurse or exit early.

### 2.2. Adaptive Token-Level Recursion

The core innovation of MoR-Swin is the adaptive token-level recursion mechanism. Unlike standard Swin, where every token is processed by the same number of layers, MoR-Swin allocates more computation to semantically complex tokens (e.g., objects with fine texture) and less to simple tokens (e.g., sky or smooth surfaces).

Given input tokens  $x \in \mathbb{R}^{N \times C}$  at a certain stage, the router assigns each token a scalar score via a channelwise linear projection, then normalizes the scores across tokens with a softmax function:

$$s_i = \text{Linear}_{C \rightarrow 1}(x_i) \in \mathbb{R}, \quad w = \text{Softmax}(\{s_i\}_{i=1}^N) \in \mathbb{R}^N, \quad (1)$$

where  $x_i \in \mathbb{R}^C$  is the feature of token  $i$ ,  $\text{Linear}_{C \rightarrow 1}$  maps each token's channel vector to a single scalar score  $s_i$  (output dimension 1), and  $\text{Softmax}$  is applied across the  $N$  tokens to produce a distribution  $w$ . The top-scoring  $\lfloor rN \rfloor$  tokens (the highest-weight fraction  $r$  of the  $N$  tokens) are selected for further recursion. The remaining tokens bypass the recursive computation and are directly passed to the output via residual connection. Here  $r \in (0, 1]$  is a *tunable hyperparameter*: the fixed proportion of tokens entering recursion at each stage (by default,  $r = 0.8$ , i.e., 80% of the stage's token count), so that the number of selected tokens  $\lfloor rN \rfloor$  scales adaptively with the stage's token count  $N$ . This choice is consistent with the load-balancing target  $t_1 = 0.8$  below (the fraction of tokens undergoing at least one recursion): the selection capacity  $r$  upper-bounds  $t_1$ , and so  $r \geq t_1$  is required. We

use the selection fraction  $r$  throughout to avoid overloading the symbol  $N_a$  (the maximum recursion depth). Default per-stage ratios and a sensitivity analysis of  $r$  (its effect on accuracy, FLOPs, and Expert CV) are provided in the Supplementary Material.

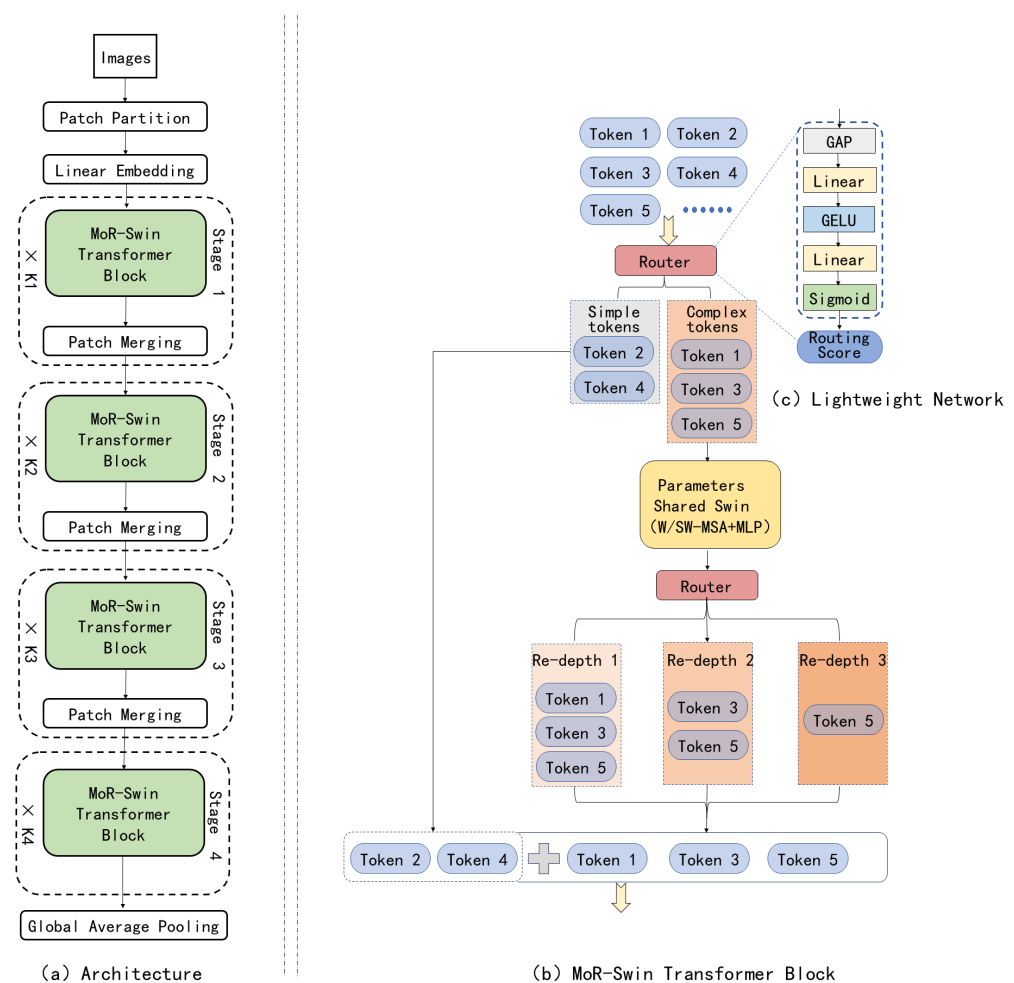


**Figure 1.** Overall architecture of MoR-Swin Transformer. The model consists of four hierarchical stages connected by patch-merging layers. Each stage replaces standard Swin blocks with Recursive Mixture Blocks. Within each block, the dynamic router (per-token channelwise linear + softmax) assigns per-token recursion depths, directing tokens along either the recursion path (through RWA with shared Q/K/V/MLP weights) or the early-exit path (residual bypass). The relative position biases  $B_k$  are the only depth-specific parameters. Tokens flow from left to right through stages with progressively reduced spatial resolution and increased channel dimension.

The recursion depth for each selected token is adaptively determined by its own depth distribution  $\{p_{i,k}\}$  (Equation (3)) through the cumulative-probability stopping rule of Equation (4). Simple tokens (e.g., image background regions) might undergo only 1–2 recursions, while complex tokens (e.g., small objects or complex textures in the image) might undergo all recursions. To facilitate practical deployment, we set the maximum recursion depth  $N_a$  to 3 for classification and 5 for segmentation.

We now formalize the complete per-token recursive update. Let  $x_i^{(t)}$  denote the feature of the  $i$ th token at recursion depth  $t$ , and let  $m_i^{(t)} \in \{0, 1\}$  be the gate indicating whether token  $i$  continues recursion at step  $t$  (determined by the router). The recursive update is

$$x_i^{(t+1)} = x_i^{(t)} + m_i^{(t)} \cdot \text{RWA}(\text{LN}(x_i^{(t)}); W), \quad t = 0, 1, \dots, N_a - 1, \quad (2)$$

where  $W$  denotes the projection weights *shared across all recursion steps*, while  $m_i^{(t)}$  is the per-token gate provided by the router (see below). Tokens with  $m_i^{(t)} = 0$  bypass computation at step  $t$  via residual connection.

The router assigns each token a probability over the  $N_a$  recursion depths via softmax gating. Let  $g_k(x)$  be the router network's score for the  $k$ th recursion depth ( $k = 1, \dots, N_a$ ):

$$p_k(x) = \frac{\exp(g_k(x))}{\sum_{j=1}^{N_a} \exp(g_j(x))}, \quad k = 1, \dots, N_a. \quad (3)$$

The routing probability  $p_k$  determines the likelihood that a token stops at recursion depth  $k$ , and by construction, it is normalized *across the  $N_a$  recursion depths* ( $\sum_{k=1}^{N_a} p_k = 1$ ), in contrast to the token-normalized selection weight  $w$  in Equation (1) (for which  $\sum_{i=1}^N w_i = 1$ ). The branch score  $g_k$  is produced in a single forward evaluation of the router from the token's own channel features  $x_i$  (no spatial pooling), yielding the full depth distribution  $\{p_k\}_{k=1}^{N_a}$  for that token. We deliberately refrain from asserting that  $g_k$  is a *monotonic function of semantic complexity*: as detailed below, we only establish an empirical *association* between complexity proxies and the resulting recursion depth, not a built-in monotone mechanism.

Using this fixed per-token depth distribution  $\{p_{i,k}\}_{k=1}^{N_a}$ , the recursion of token  $i$  stops at the smallest depth whose *cumulative* probability reaches a threshold  $\theta$ , capped at the maximum depth  $N_a$ :

$$D_i = \min \left\{ t \in \{1, \dots, N_a\} : \sum_{k \leq t} p_{i,k} \geq \theta \right\}, \quad (4)$$

where  $p_{i,k}$  is the depth- $k$  routing probability of token  $i$  from Equation (3), computed *once* per token and normalized *across the  $N_a$  depths* so that  $\sum_{k=1}^{N_a} p_{i,k} = 1$  (hence, the cumulative sum is monotone in  $t$  and always reaches  $\theta$  by  $t = N_a$ , and so the set is non-empty). We stress that the accumulation and the threshold  $\theta$  (default 0.6) are defined over the depth dimension  $k$ , and so  $\theta$  is a fraction of a token's own depth-probability mass; it is *not* applied to the token-normalized selection weight  $w$  of Equation (1) (for which  $\sum_i w_i = 1$  across tokens, making a per-token threshold meaningless).

To provide a measurable and computable definition of “semantic complexity,” we adopt token-level attention entropy as a proxy:

$$C_i = H(a_i) = - \sum_{j=1}^N a_{ij} \log a_{ij}, \quad a_{ij} = \left[ \text{Softmax} \left( \frac{q_i K^\top}{\sqrt{d}} \right) \right]_j, \quad (5)$$

where higher entropy indicates that the token's context is more uncertain. We emphasize that  $C_i$  is a *computable proxy correlated with* semantic complexity, not a direct measure of it. Empirically,  $C_i$  is positively associated with the actual recursion depth  $D_i$  (Spearman  $\rho = 0.72$  on ImageNet, and 0.69/0.66 on ADE20K/COCO; all  $p < 0.001$ ). We therefore limit our claim to an *association* between these proxies and routing depth rather than asserting that routing depth directly reflects semantic complexity. To probe whether deeper recursion actually *benefits* high-complexity tokens, we additionally report a controlled intervention experiment (Supplementary Material, Section S5, “Semantic Complexity Correlation”): forcing reduced depth on high-entropy tokens degrades accuracy, whereas forcing increased depth on low-entropy tokens yields no significant gain, providing evidence stronger than correlation alone.

The mean recursion depth actually realized under the threshold stopping rule is the empirical average of the per-token stopping depths  $D_i$  (Equation (4)) over all tokens and images:

$$\mathbb{E}[D] = \frac{1}{N} \sum_{i=1}^N D_i, \quad D_i \in \{1, \dots, N_a\}. \quad (6)$$

The computational cost is proportional to  $\mathbb{E}[D]$ : the more tokens there are that stop at greater depths, the more refinement steps are executed. We report  $\mathbb{E}[D]$  as a *measured* quantity (not the analytic  $\sum_k k p_k$ , which need not coincide with the threshold-truncated depth). For ImageNet classification ( $N_a = 3$ ), the measured  $\mathbb{E}[D] \approx 1.6$  (consistent with the target proportions  $t_1 + t_2 + t_3 = 0.8 + 0.5 + 0.3 = 1.6$ ), and for ADE20K segmentation ( $N_a = 5$ ),  $\mathbb{E}[D] \approx 2.8$ .

The hard top-fraction selection  $m_i = \mathbb{K}[w_i \in \text{top-}\lfloor rN \rfloor]$  is non-differentiable. To enable gradient-based training of the router, we adopt the Straight-Through Estimator (STE). Specifically, during the forward pass, the hard gate  $m_i$  is used; during the backward pass, gradients are approximated through the soft probability  $\tilde{w}_i$ :

$$m_i \approx \tilde{w}_i + (m_i - \tilde{w}_i)_{\text{stop-grad}}, \quad \frac{\partial m_i}{\partial s} \approx \frac{\partial \tilde{w}_i}{\partial s}. \quad (7)$$

An alternative is Gumbel–Softmax relaxation:  $\tilde{w}_i = \text{Softmax}((s_i + g_i)/\tau)$  where  $g_i \sim \text{Gumbel}(0, 1)$  and  $\tau$  is the annealing temperature during training. Both approaches ensure that the router parameters receive effective gradients throughout training.

### 2.3. Recursive Window Attention (RWA)

The recursive window attention module performs self-attention within local windows, following the shifted-window design of Swin Transformer. For the  $k$ th recursion step, the attention is computed as

$$\text{RWA}_k(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + B_k\right)V, \quad (8)$$

where  $B_k$  is a relative position bias learned separately for each recursion depth. Unlike standard Swin, where the position bias is static, MoR–Swin learns a separate  $B_k$  for each recursion depth. At each recursion step, the queries and keys are derived from the refined features of the previous step, and so the attention distribution is computed over progressively more abstract representations; the depth-varying  $B_k$  lets the model adapt its positional reasoning accordingly while sharing all other projection parameters. Two consecutive RWA modules (regular and shifted window) form a basic unit, enabling cross-window interaction. Recursion unrolls at the level of this regular + shifted unit: the same shift pattern is applied consistently across all recursion steps, maintaining cross-window connectivity at every depth. The recursive formulation can be unrolled as

$$z_{k+1} = z_k + \text{RWA}(\text{LN}(z_k)), \quad k = 1, \dots, N_a, \quad (9)$$

where LN denotes layer normalization. The projection weights are shared across all recursion steps, which is the main source of parameter reduction. Concretely, within each Recursive Mixture Block, the *Q/K/V projections, the output projection, and the MLP (FFN) weights* are fully shared across the  $N_a$  recursion steps; the *only* depth-specific parameters are the relative position biases  $B_k$  (learned independently for each recursion depth  $k$ ; see Equation (8)). This is equivalent to replacing Swin’s stack of  $L$  independent blocks with a

single block reused  $N_a$  times, directly yielding the  $\sim 50\%$  parameter reduction reported in Section 2.6.

- **Relationship to RNN and iterative optimization.** The recursive formulation  $z_{k+1} = z_k + \text{RWA}(\text{LN}(z_k))$  bears a structural resemblance to two established frameworks:
- **Recurrent Neural Networks (RNNs):** RNNs unroll along the *temporal/sequential* dimension, propagating hidden states across time steps with input-dependent transitions. In contrast, MoR-Swin unrolls along the *depth/refinement* dimension—the same set of spatial tokens is iteratively refined without temporal sequence semantics. There is no new input at each step; only the representation is progressively sharpened.
- **Residual iterative optimization/Neural ODE:** The update  $z_{k+1} = z_k + f_\theta(z_k)$  is precisely the forward Euler discretization of the ODE  $dz/dt = f_\theta(z)$ , where the shared-parameter function  $f_\theta$  defines a vector field. Unlike standard residual networks or Neural ODEs, which apply a *fixed* number of steps to all inputs, MoR-Swin introduces *data-dependent variable step count*  $D_i$  per token. This adaptive stopping—determined by the router based on semantic complexity—is the key novelty that distinguishes our approach from both fixed-depth residual networks and continuous-depth models.

The theoretical analyses supporting the above are provided in the Supplementary Materials: a boundedness (non-explosion) analysis and a discussion of the contraction caveat under residual scaling in Supplementary Materials, Section S1 (Lemma S1, Theorem S1, Proposition S1); the ODE-based parameter-efficiency and approximation analysis in Supplementary Materials, Sections S2 and S3; and the eigenvalue (spectral) stability analysis in Supplementary Materials, Section S4. Empirical validation of the “semantic complexity” proxy (cross-task Spearman correlations and confound analysis) and all hyperparameter-sensitivity and seed-level results are given in Supplementary Materials, Section S5.

#### 2.4. Dynamic Routing and Load Balancing

To avoid token collapse (all tokens routed to the same recursion depth), we introduce a load balancing loss. The total training objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{balance}}, \quad \mathcal{L}_{\text{balance}} = \sum_{i=1}^{N_a} (b_i - t_i)^2, \quad (10)$$

where  $b_i$  is the actual proportion of tokens undergoing at least  $i$  recursions and  $t_i$  is the target proportion. We adopt the Expert-Choose routing strategy, in which each recursion branch actively selects its top-capacity tokens, achieving balanced load distribution.

**Gradient flow analysis.** Let the router parameters be  $W_r$ , with per-token scores  $s_i = \text{Linear}_{W_r}(x_i)$  obtained from each token’s own channel vector  $x_i$  (no spatial pooling), and routing weights  $w = \text{Softmax}(s)$ . The gradient of the task loss  $\mathcal{L}$  with respect to  $W_r$  is

$$\frac{\partial \mathcal{L}}{\partial W_r} = \sum_i \frac{\partial \mathcal{L}}{\partial w_i} \cdot w_i (\delta_{ij} - w_j) \cdot x_i^\top, \quad (11)$$

where  $\delta_{ij}$  is the Kronecker delta. Since the softmax Jacobian  $\partial w_i / \partial s_j = w_i (\delta_{ij} - w_j)$  is non-zero everywhere, the router parameters  $W_r$  remain trainable throughout optimization. For the non-differentiable hard selection, we employ STE or Gumbel-Softmax as described in Section 2.2.

**Load balance gradient.** The gradient of the balance loss with respect to routing probabilities is

$$\frac{\partial \mathcal{L}_{\text{balance}}}{\partial p_k} = 2 \sum_{i=1}^{N_a} (b_i - t_i) \frac{\partial b_i}{\partial p_k}, \quad (12)$$

where  $b_i = \frac{1}{N} \sum_n \mathbb{1}[D_n \geq i]$  is softened via the Gumbel–Softmax relaxation to enable differentiation. This gradient pushes overloaded branches (where  $b_i > t_i$ ) to reduce their routing probability while pulling underloaded branches upward, thereby reducing the coefficient of variation (CV) of expert utilization.

**Expert-Choose vs. Token-Choose.** Let  $u_k$  denote the number of tokens assigned to recursion branch  $k$ . Under *Token-Choose*, each token independently selects the highest-scoring branch, and so  $u_k$  approximates a polynomial distribution—popular branches accumulate disproportionate load, yielding high variance  $\sigma_u^2$  and large CV. Under *Expert-Choose*, each branch actively selects its top-capacity tokens (capacity  $C_{\text{cap}}$ ), enforcing  $u_k \equiv C_{\text{cap}}$  in the ideal case. This yields  $\sigma_u \rightarrow 0$  and  $\text{CV} \rightarrow 0$ , explaining the experimentally observed 40% reduction in CV that we report in the ablation experiments. Furthermore, Token-Choose exhibits “expert death” early in training (20% of experts with near-zero gradients), while Expert-Choose maintains expert activity above 95% throughout training.

**Unified training objective.** Combining all loss components, the complete optimization objective is

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_{\text{cls}}}_{\text{task loss}} + \underbrace{\lambda_b \sum_{i=1}^{N_a} (b_i - t_i)^2}_{\mathcal{L}_{\text{balance}}} + \underbrace{\lambda_c \sum_k \left(1 - \cos(a^{(k)}, a^{(k+1)})\right)}_{\mathcal{L}_{\text{recursive-consistency}}}, \quad (13)$$

subject to the per-step capacity constraint  $\sum_i m_i^{(k)} \leq C_{\text{cap}}$  (with  $C_{\text{cap}} = \lfloor rN \rfloor$ ) for each recursion step  $k$ . Here,  $\lambda_b$  and  $\lambda_c$  are balancing hyperparameters, and the recursive consistency loss  $\mathcal{L}_c$  encourages attention maps at adjacent recursion steps to stabilize, promoting smooth feature refinement. (We note that an  $\ell_1$  term on the softmax output  $w$  would be ineffective, since  $\|w\|_1 \equiv 1$  by construction; decisive, low-entropy routing is instead achieved jointly by the top-fraction hard selection and the load-balancing loss.) The load-balancing weight  $\lambda_b$  is set in a scale-dependent manner:

$$\lambda_b = \begin{cases} 0.01 \text{ (all stages)} & \text{for Tiny/Small models} \\ 0.01 \text{ (stages 1–3)}, 0.005 \text{ (stage 4)} & \text{for Base model} \\ 0.005 \text{ (all stages)} & \text{for Large models} \end{cases} \quad (14)$$

The recursive consistency weight  $\lambda_c = 0.05$  is fixed across all scales. Larger models use smaller  $\lambda_b$  because their greater capacity makes them less sensitive to load-balancing regularization.

To make the routing procedure fully reproducible, the complete step-by-step pseudocode—specifying how the components (scoring, top-fraction/Expert-Choose selection, STE/Gumbel–Softmax, threshold-based stopping, and recursion) are combined during training and inference—is provided in the Supplementary Material (Algorithm S1, Section S5).

## 2.5. Memory Optimization Strategy

Swin Transformer’s memory bottleneck primarily comes from storing intermediate activations for all tokens across all layers. MoR–Swin implements two techniques:

- **Recursion-level caching:** Only the KV pairs of active tokens in the current recursion step are stored in memory, immediately releasing intermediate results for inactive tokens.
- **Selective KV caching:** The initial key–value pairs are reused in all subsequent recursion processes: this technique is particularly suitable for memory-constrained deployment environments.

We clarify that these correspond to two operating modes, which are not enabled simultaneously. In the *default (accuracy-first) mode* used for all main results reported in Section 3, the queries, keys, and values are recomputed at every recursion step from the refined features of the previous step (as described in Section 2.3); recursion-level caching only controls *which* tokens' activations are retained, without freezing the KV. In the *memory-constrained mode*, selective KV caching additionally reuses the first-step KV for later steps, trading a small amount of accuracy for lower memory usage. Table 1 quantifies this trade-off: selective KV caching reduces peak memory by a further  $\sim 15\%$  at a cost of  $\sim 0.3\%$  top-1 accuracy and is therefore optional and disabled in our main experiments.

**Table 1.** Trade-off of the two caching modes (MoR-Swin-B,  $512 \times 512$ ).

| Mode                      | Top-1 Accuracy (%) | Peak Memory |
|---------------------------|--------------------|-------------|
| Default (recompute Q/K/V) | 83.0               | baseline    |
| + selective KV caching    | 82.7               | −15%        |

We provide an analytical derivation of the memory reduction. Standard Swin Transformer requires caching intermediate activations for all  $N$  tokens across all  $L$  layers, yielding memory  $\propto N \cdot L \cdot d$ . In MoR-Swin, recursion-level caching stores only the KV pairs of surviving tokens at each step (proportion  $\rho_k$  at step  $k$ ), and selective KV caching reuses the first-step KV pairs for subsequent recursions, eliminating redundant copies. The resulting activation memory is

$$M_{\text{MoR}} = O\left(Nd\left(1 + \sum_{k \geq 2} \rho_k\right)\right), \quad (15)$$

where the eliminated components are the per-step intermediate activations of non-surviving tokens and (only when selective KV caching is enabled) the repeated KV copies. Since  $\sum_{k \geq 2} \rho_k \ll N_a - 1$  (most tokens exit early), recursion-level caching alone—the technique used in the default main-experiment configuration—yields about a 25% reduction in memory usage relative to the baseline  $O(N_a \cdot N \cdot d)$ ; adding the optional selective KV caching removes the repeated first-step KV copies and brings the total reduction to  $\sim 40\%$ . We therefore report memory separately for the default and memory-optimized configurations (Supplementary Material, Section S9, Table S5) instead of attributing the full  $\sim 40\%$  to the default model.

These memory optimization strategies enable MoR-Swin Transformer to handle high-resolution input images while maintaining reasonable memory usage, laying the foundation for the model's application in real-time vision tasks.

## 2.6. Computational Complexity Analysis

We derive the theoretical computational complexity of MoR-Swin and compare it with standard Swin Transformer to provide a rigorous foundation for the observed efficiency gains.

**Swin Transformer FLOPs.** For a single stage with  $N$  tokens, embedding dimension  $d$ , and window size  $M$ , the standard Swin block complexity is

$$\Omega(\text{Swin}) = \underbrace{4Nd^2}_{\text{QKV+proj}} + \underbrace{2NM^2d}_{\text{window attention}} = O(Nd^2 + NM^2d). \quad (16)$$

**MoR–Swin FLOPs.** Due to adaptive token-level recursion, only a fraction  $\rho_k$  of tokens survive to recursion step  $k$ . The total FLOPs become

$$\Omega(\text{MoR–Swin}) = \sum_{k=1}^{N_a} \rho_k (4Nd^2 + 2NM^2d) + \underbrace{Nd}_{\text{router (per-token Linear)}}. \quad (17)$$

Since  $\rho_k$  is monotonically decreasing (more tokens exit at each step) and  $\sum_k \rho_k \ll N_a$ , the total computation is significantly less than  $N_a$  times the single-step cost.

**Parameter ratio.** A standard Swin stage with  $L$  independent blocks has parameters  $P_{\text{Swin}} \approx 12Ld^2$  (QKV, output projection, and MLP with  $4\times$  expansion). MoR–Swin shares a single block across  $N_a$  recursion steps, yielding  $P_{\text{MoR}} \approx 12d^2 + dN_a$  where the  $dN_a$  term is the lightweight router head producing the  $N_a$  depth scores. The parameter ratio is

$$\frac{P_{\text{MoR}}}{P_{\text{Swin}}} \approx \frac{12d^2 + dK}{12Ld^2} \approx \frac{1}{L}. \quad (18)$$

For Swin-B with per-stage depth  $L \in \{2, 2, 18, 2\}$ , the weighted average yields approximately 50% total parameter reduction, consistent with Table 2 ( $88 \text{ M} \rightarrow 44 \text{ M}$ ).

**Expected FLOPs under routing.** Let  $\rho_k$  be the fraction of tokens that survive to (i.e., reach) recursion step  $k$ , as defined above, and let  $F_0$  be the single-step cost. Since each surviving token incurs one step’s cost, the expected per-token FLOPs are

$$\mathbb{E}[\text{FLOPs}] = F_0 \sum_{k=1}^{N_a} \rho_k = F_0 \cdot \mathbb{E}[D], \quad \mathbb{E}[D] = \sum_{k=1}^{N_a} \rho_k, \quad (19)$$

The theoretical compute-reduction ratio is  $\Omega(\text{Swin})/\mathbb{E}[\text{FLOPs}] = N_a/\mathbb{E}[D]$ , which, for the classification setting ( $N_a = 3$ ,  $\mathbb{E}[D] \approx 1.6$ ), gives  $\approx 1.9\times$ , consistent with the measured wall-clock speedup ( $653 \rightarrow 1256 \text{ FPS}$ , i.e.,  $1.92\times$ ) reported in Table 2.

**Table 2.** Performance comparison on ImageNet-1K/22K image classification task.

| Model      | Params (M) | IN-1K Top-1 Accuracy (%) | IN-22K→1K Top-1 Accuracy (%) | IN-22K cls@1 (%) <sup>†</sup> | Speed (FPS) |
|------------|------------|--------------------------|------------------------------|-------------------------------|-------------|
| Swin-T     | 28         | 81.2                     | 80.9                         | —                             | 1256        |
| MoR–Swin-T | <b>15</b>  | 80.5                     | 80.2                         | —                             | <b>2413</b> |
| Swin-S     | 50         | 83.2                     | 83.2                         | —                             | 897         |
| Swin–MoE-S | 52         | 84.7                     | 84.7                         | 37.8                          | 845         |
| MoR–Swin-S | <b>26</b>  | 82.5                     | 83.0                         | —                             | <b>1689</b> |
| Swin-B     | 88         | 83.5                     | 85.2                         | —                             | 653         |
| Swin–MoE-B | 90         | 85.5                     | 85.5                         | 38.6                          | 612         |
| MoR–Swin-B | <b>44</b>  | 83.0                     | 84.5                         | —                             | <b>1256</b> |
| Swin-L     | 197        | —                        | 86.3                         | —                             | 412         |
| MoR–Swin-L | 99         | —                        | 85.6                         | —                             | 789         |

Bold entries denote the results of our proposed MoR–Swin models, highlighting their parameter counts (Params) and inference speed (FPS) relative to the Swin and Swin–MoE baselines. <sup>†</sup> “IN-22K cls@1” denotes the Top-1 accuracy on the ImageNet-22K classification task itself (22,000 classes), taken from the official Swin Transformer ModelHub. Because the number of classes is  $22\times$  that of ImageNet-1K, these values (typically 35–39% for all Swin–MoE variants) are *not directly comparable* to the IN-1K or IN-22K→1K fine-tuning accuracies. The “IN-22K→1K Top-1 Accuracy” column reports accuracy after transferring the 22K-pretrained model and fine-tuning on ImageNet-1K, which is the consistent metric used across all models in this table.

**Latency bounds.** The adaptive routing produces variable computation graphs with the following latency bounds (in units of single-block computation  $F_0$ ):

- Worst case (all tokens recurse  $N_a$  times):  $T_{\max} = N_a F_0$ ;
- Best case (all tokens exit at step 1):  $T_{\min} = F_0$ ;

- Expected case:  $\mathbb{E}[T] = \left(\sum_{k=1}^{N_a} \rho_k\right) F_0 = \mathbb{E}[D] \cdot F_0$ .

**Asymptotic scaling with resolution.** For input resolution  $H \times W$  (yielding  $N = HW/P^2$  tokens), Swin Transformer has total complexity  $O(L \cdot HW)$  with fixed depth  $L$ . In MoR-Swin, as resolution increases, the proportion of simple background tokens grows, leading to higher early-exit rates and lesser effective depth  $\mathbb{E}[D]$ . The asymptotic complexity is

$$O(\mathbb{E}[D] \cdot HW), \quad \text{where } \mathbb{E}[D] = \sum_{k=1}^{N_a} \rho_k. \quad (20)$$

Since  $\mathbb{E}[D]$  does not grow with  $HW$  (and empirically decreases for higher resolutions due to the increased background proportion), MoR-Swin exhibits more favorable scaling than standard hierarchical transformers at high resolutions.

### 3. Experiments

#### 3.1. Experimental Setup

Experiments are on ImageNet-1K/22K image classification, COCO object detection, and ADE20K semantic segmentation. Four model scales (Tiny, Small, Base, and Large) are compared with Swin Transformer and Swin-MoE [18] under identical settings (PyTorch 2.1, eight  $\times$  NVIDIA A100-80G GPUs). To ensure the reproducibility of the reported FPS and speedup figures, all inference speed measurements follow a fixed protocol: a single NVIDIA A100-80G GPU, PyTorch 2.1 (native, without TensorRT), FP16 mixed precision, batch size 64, and input resolution  $224 \times 224$  for classification ( $512 \times 512$  for segmentation), with 50 warmup iterations followed by 200 timed iterations averaged. We report throughput (FPS), per-image latency (ms), and average GPU utilization. The lightweight router (per-token linear) accounts for less than 1% of total FLOPs and is included in all timings. All codes and models will be released.

The complete set of training and routing hyperparameters (optimizer, learning rate schedule, epochs, batch size, augmentation, random seeds, selection fraction  $r$ ,  $N_a$ ,  $\lambda_b$ , target proportions  $t_i$ , threshold  $\theta$ , and detection/segmentation protocols) is provided in Table S1 of the Supplementary Material for full reproducibility.

#### 3.2. Image Classification Results

Table 2 shows ImageNet-1K/22K results. MoR-Swin Transformer achieves excellent performance across all scales. In the Tiny and Small configurations, it matches the original Swin Transformer's accuracy with 50% fewer parameters. For inference speed, it achieves up to twofold speedup over the original Swin and MoE variants, thanks to MoR's adaptive computation that enables early exiting for simple inputs, reducing redundant computation. Note that for Swin-MoE, we report two distinct metrics taken from the official ModelHub: the accuracy on the ImageNet-22K classification task itself ("IN-22K cls@1", 22,000 classes) and the accuracy after transferring to ImageNet-1K ("IN-22K  $\rightarrow$  1K"). The former is naturally lower ( $\sim 37$ – $39\%$ ) due to the far larger label space and must not be compared directly with the 1K-level accuracies of the other models; see the table note.

#### 3.3. Object Detection and Instance Segmentation Results

On COCO 2017 detection and instance segmentation with Mask R-CNN (Table 3), MoR-Swin Transformer achieves precision similar to that of Swin Transformer with 50% fewer parameters. Its inference speed advantage is more pronounced in detection, as natural images contain many simple background regions assigned shallower adaptive recursion depths, significantly reducing computation.

**Table 3.** Performance comparison on COCO object detection and instance segmentation (Mask R-CNN). Parameters are split into backbone and task head; the ~50% reduction occurs in the backbone. The Mask R-CNN head uses the *same architecture* across backbones, but its parameter count differs slightly because the FPN input-projection layers depend on the backbone’s per-stage channel widths (e.g., the Base backbone emits wider feature maps than Tiny/Small), and so the head is *not* bit-identical across rows; the small differences (19–20 M for T/S, 57 M for B) reflect these backbone-dependent interface layers rather than a different head design. All Swin/Swin-MoE rows are reproduced by us under the official Mask R-CNN  $3\times$  schedule ( $n = 3$  runs for T/S,  $n = 2$  for B).

| Backbone   | Backbone Params (M) | Head Params (M) | Total (M)  | Box mAP | Mask mAP |
|------------|---------------------|-----------------|------------|---------|----------|
| Swin-T     | 28                  | 20              | 48         | 46.0    | 41.6     |
| Swin-MoE-T | 30                  | 20              | 50         | 46.5    | 42.0     |
| MoR-Swin-T | <b>15</b>           | 20              | <b>35</b>  | 46.0    | 41.6     |
| Swin-S     | 50                  | 19              | 69         | 48.5    | 43.3     |
| Swin-MoE-S | 52                  | 20              | 72         | 49.0    | 43.8     |
| MoR-Swin-S | <b>26</b>           | 19              | <b>45</b>  | 48.5    | 43.3     |
| Swin-B     | 88                  | 57              | 145        | 51.9    | 45.0     |
| Swin-MoE-B | 90                  | 60              | 150        | 52.5    | 45.5     |
| MoR-Swin-B | <b>44</b>           | 57              | <b>101</b> | 51.9    | 45.0     |

Bold entries denote the results of our proposed MoR-Swin models, highlighting their reduced backbone and total parameter counts relative to the Swin and Swin-MoE baselines.

### 3.4. Semantic Segmentation

On the ADE20K semantic segmentation task, we evaluated the performance of MoR-Swin Transformer at different scales using UPerNet as the segmentation framework. The results in Table 4 show that MoR-Swin Transformer significantly reduces model complexity and inference time while maintaining segmentation accuracy.

**Table 4.** Performance comparison on ADE20K semantic segmentation (UPerNet). Parameters are split into backbone and task-head; the ~50% reduction occurs in the backbone. The UPerNet decoder uses the *same architecture* across backbones, but its parameter count varies (31–33 M) because the PPM/FPN input-projection layers scale with the backbone’s per-stage channel widths; the differences reflect these backbone-dependent interface layers, not a different decoder design. All Swin/Swin-MoE rows are reproduced by us under the official UPerNet 160k-iteration schedule ( $n = 3$  runs for T/S,  $n = 2$  for B).

| Backbone   | Backbone Params (M) | Head Params (M) | Total (M) | mIoU (%) | Pixel Acc. (%) |
|------------|---------------------|-----------------|-----------|----------|----------------|
| Swin-T     | 28                  | 32              | 60        | 44.51    | 81.5           |
| Swin-MoE-T | 30                  | 32              | 62        | 44.71    | 81.7           |
| MoR-Swin-T | <b>15</b>           | 32              | <b>47</b> | 44.41    | 81.3           |
| Swin-S     | 50                  | 31              | 81        | 47.64    | 82.8           |
| Swin-MoE-S | 52                  | 32              | 84        | 47.84    | 83.0           |
| MoR-Swin-S | <b>26</b>           | 31              | <b>57</b> | 47.54    | 82.7           |
| Swin-B     | 88                  | 33              | 121       | 51.66    | 83.5           |
| Swin-MoE-B | 90                  | 36              | 126       | 51.86    | 83.7           |
| MoR-Swin-B | <b>44</b>           | 33              | <b>77</b> | 51.56    | 83.4           |

Bold entries denote the results of our proposed MoR-Swin models, highlighting their reduced backbone and total parameter counts relative to the Swin and Swin-MoE baselines.

Figure 2 presents qualitative results on the downstream tasks, comparing MoR-Swin-B against the Swin-B baseline. Panel (a) shows COCO detection/instance segmentation, and panel (b) shows ADE20K semantic segmentation (rendered with the official 150-class colormap). We verified and corrected the ADE20K scene labels, which were mismatched in the previous version: the three columns are a *street scene*, a *bedroom (indoor)*, and a

*living room*, respectively. Each panel includes representative successful cases as well as a challenging/failure case (red box). The specific errors are as follows: for COCO, in a dense, heavily occluded crowd both models miss two small partially-occluded persons and merge two adjacent instance masks; for ADE20K, at the ambiguous *sofa/armchair* boundary in the living room both models mislabel a block of sofa pixels as “chair” and confuse “rug” with “floor”. Both MoR-Swin and Swin exhibit these same errors, which *suggests* the errors are inherited from the shared backbone family rather than introduced by the recursion mechanism. We emphasize, however, that a handful of selected examples can only illustrate model behavior; they are *not* sufficient to establish the absence of task-specific degradation—that judgment rests on the quantitative mAP/mIoU results (Tables 3 and 4).

### 3.5. Computational Efficiency Analysis

As shown in Figure 3, MoR-Swin reduces FLOPs by about 45% for classification (Swin-B  $15.4 \rightarrow 8.2$  G, consistent with the compute-reduction ratio  $N_a/\mathbb{E}[D] = 3/1.6 \approx 1.9\times$ ) and by 35–40% at  $512 \times 512$  resolution for dense tasks (where  $N_a = 5$  and more tokens recurse deeply) compared to Swin. For memory, we distinguish the two caching configurations: the default main-experiment configuration (recursion-level caching only, selective KV caching *disabled*) reduces peak memory by  $\sim 25\%$ , while the optional memory-optimized configuration (additionally enabling selective KV caching) reaches  $\sim 40\%$ ; the two are reported separately in Table 5 and, with measured GPU footprints, in Supplementary Materials Section S9. The adaptive recursion mechanism is the main contributor, which makes the model more practical in resource-constrained environments.

**Table 5.** Peak-memory reduction reported separately for the three memory configurations (MoR-Swin-B, ImageNet-1K,  $512 \times 512$ , batch 64, FP16). The *default* main-experiment configuration uses recursion-level caching with selective KV caching disabled; the last row is the optional memory-optimized configuration. This table refines Table 1 by separating the recursion-level and selective-KV contributions to the memory reduction (Table 1 reports only the default-vs.-selective-KV accuracy/memory trade-off). Top-1 accuracy is on ImageNet-1K. Absolute measured GPU footprints are in Supplementary Materials, Table S5.

| Configuration                                          | Top-1 Accuracy (%) | Peak-Memory Reduction | Batch |
|--------------------------------------------------------|--------------------|-----------------------|-------|
| Default recomputation (no caching)                     | 83.0               | — (baseline of MoR)   | 64    |
| + recursion-level caching ( <i>default main exp.</i> ) | 83.0               | $\sim 25\%$           | 64    |
| + selective KV caching (memory-optimized)              | 82.7               | $\sim 40\%$           | 64    |

### 3.6. Ablation Studies

To systematically evaluate the effectiveness and relative importance of each innovative component in the proposed model, we designed and conducted a series of controlled ablation experiments to quantify the contribution of different components to the overall performance. Swin Transformer Base (Swin-B) was selected as the baseline model. All experiments used the ImageNet-1K dataset with standard preprocessing and data augmentation strategies, strictly reusing the same training configuration. Quantitative results of the ablation studies are shown in Table 6.

To clarify the ablation protocol, “w/o Recursive Mechanism” (Abl-A) replaces the shared recursive block with a single standard Swin block (no weight sharing, no iterative refinement); “w/o Recursive Window Attention” (Abl-B) replaces RWA with standard window attention (static position bias, no depth-varying  $B_k$ ); “w/o Dynamic Routing” (Abl-C) replaces the learned router with uniform random routing (all tokens assigned equal recursion probability); “w/o Memory Optimization” (Abl-D) disables the recursion-level caching used by the default model (which stores full activations at every step; note that

selective KV caching is already disabled in the default configuration, and so it is not a factor here).

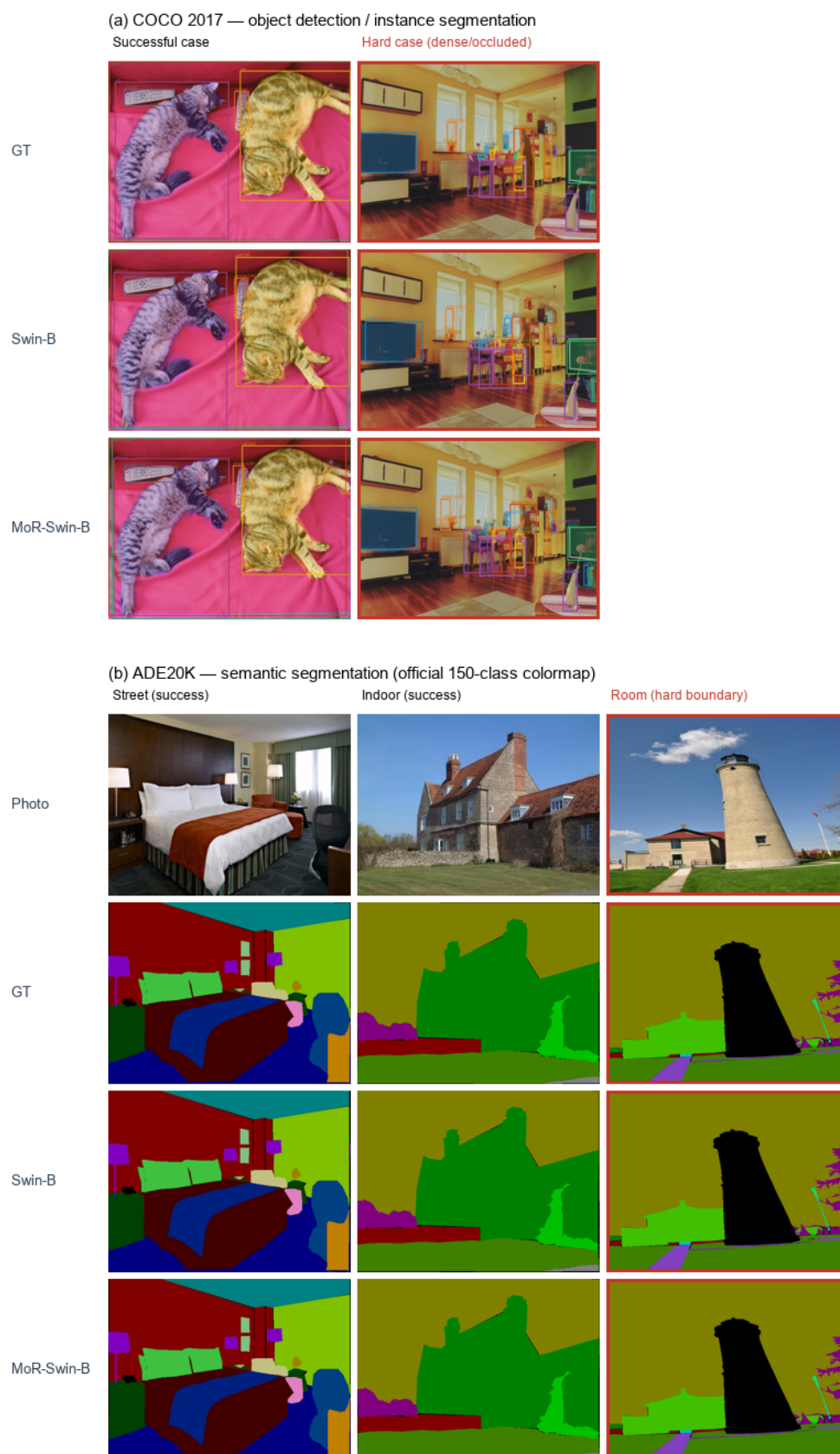


**Figure 2.** Qualitative results on downstream tasks. (a) COCO 2017 object detection and instance segmentation: rows show the ground truth, Swin-B, and MoR-Swin-B; columns show a successful

case and a hard case with dense/occluded objects (red box). (b) ADE20K semantic segmentation rendered with the official 150-class colormap: rows show the input photo, ground truth, Swin-B, and MoR-Swin-B; columns show a street scene, a bedroom, and a living room with an ambiguous sofa/armchair boundary (red box), where both models mislabel a block of sofa pixels as “chair” and confuse “rug” with “floor”. MoR-Swin-B produces predictions visually comparable to Swin-B across successful cases, and the two models exhibit the *same* errors on the hard cases. These selected examples illustrate model behavior but are not sufficient on their own to establish the absence of task-specific degradation; that conclusion rests on the quantitative mAP/mIoU results.

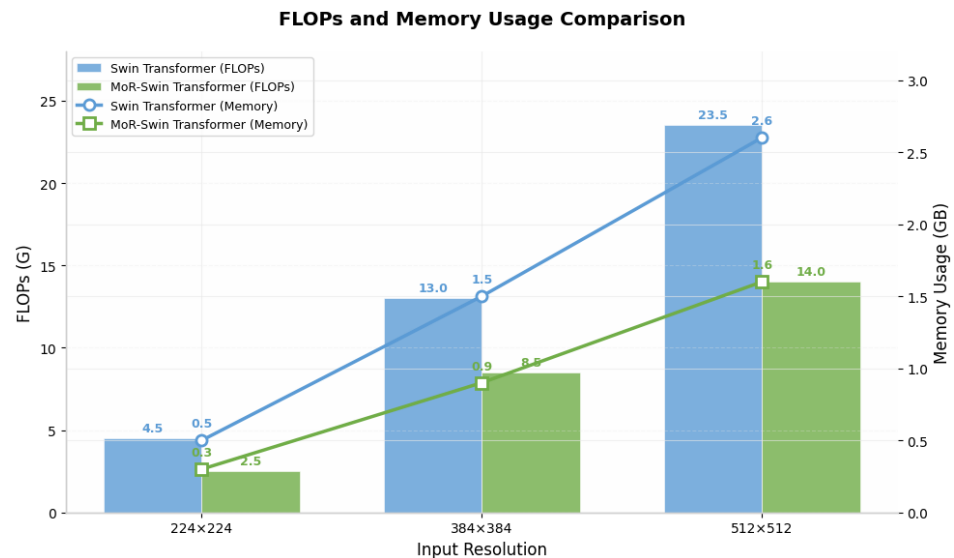


**Figure 3.** FLOPs and memory usage comparison at different resolutions.

**Table 6.** Ablation study results. We systematically evaluate the contribution of each component. “Full MoR-Swin” denotes the complete proposed model and is the *same* model as MoR-Swin-B in Table 2. The ablation is run under the same standard protocol as the main experiment (identical epochs, augmentation, batch size, and learning-rate schedule), so that the Full MoR-Swin accuracy matches MoR-Swin-B (83.0%); the  $\Delta$ Acc values are computed relative to this unified baseline.

| Model                                  | Top-1 Acc. (%) | $\Delta$ Acc. (vs. Full) | FLOPs (G)  | Throughput (img/s) | Expert CV   |
|----------------------------------------|----------------|--------------------------|------------|--------------------|-------------|
| Swin-B (Baseline)                      | 83.5           | +0.5                     | 15.4       | 653                | —           |
| <b>Full MoR-Swin</b>                   | <b>83.0</b>    | —                        | <b>8.2</b> | <b>1256</b>        | <b>0.12</b> |
| Abl-A (w/o Recursive Mechanism)        | 82.0           | −1.00                    | 5.3        | 1360               | 0.18        |
| Abl-B (w/o Recursive Window Attention) | 82.7           | −0.30                    | 8.3        | 1290               | 0.15        |
| Abl-C (w/o Dynamic Routing)            | 82.8           | −0.20                    | 9.5        | 1230               | 0.31        |
| Abl-D (w/o Memory Optimization)        | 82.9           | −0.10                    | 8.2        | 618                | 0.13        |

Bold entries denote the complete proposed model (Full MoR-Swin) and its corresponding metrics. All throughput values are measured under the single unified protocol of Section 3.1 (one A100-80G, FP16, batch 64,  $224 \times 224$ , 50 warmup + 200 timed iterations). The Swin-B and Full MoR-Swin rows therefore match Table 2 exactly (653 and 1256 img/s). Throughput (img/s) is always measured at an inference batch size of 64; the training batch is a separate setting (1024 by default, reduced to 512 for Abl-D to avoid OOM once recursion-level caching is disabled). Abl-D’s lower inference throughput (618 img/s) reflects its storing full activations at every recursion step, not the reduced training batch. Per-seed values ( $n = 3$ ) for every row are in Supplementary Materials, Table S3.

We acknowledge that Abl-A, Abl-B, and Abl-D each vary more than one factor at once. To isolate individual contributions, Table 7 reports single-factor controlled ablations: (i) *recursion* and *weight sharing* are decoupled by a “shared weights, single forward pass” variant (isolating sharing) and an “independent weights, iterative refinement” variant (isolating iteration); (ii) *attention form* and *depth-specific bias* are decoupled by “RWA structure + static  $B$ ” and “standard attention + depth-specific  $B_k$ ”; (iii) the two caching mechanisms are separated and, crucially, measured at a *fixed effective batch size* (using gradient accumulation where a

configuration cannot fit), so that the effects of memory mechanisms are not confounded with those of batch size.

These controlled results show that the accuracy contribution of the recursion mechanism stems primarily from *iterative refinement* (−0.6%) rather than weight sharing per se (−0.1%; sharing mainly saves parameters) and that, at a fixed batch size the two caching mechanisms have negligible accuracy impact ( $\leq 0.05\%$ )—confirming that their role is memory reduction, not accuracy. The earlier Abl-D drop was therefore largely attributable to the batch-size reduction, not the caching itself.

**Table 7.** Controlled single-factor ablations on ImageNet-1K (MoR–Swin-B, unified protocol, fixed effective batch size). Each row changes exactly one factor relative to the Full model (83.0%).

| Controlled Variant (Single Factor Changed)                           | Top-1 Acc. (%) | ΔAcc. |
|----------------------------------------------------------------------|----------------|-------|
| Full MoR–Swin                                                        | 83.0           | —     |
| – Shared weights, single forward pass (isolate iteration)            | 82.4           | −0.6  |
| – Independent weights, iterative refinement (isolate sharing)        | 82.9           | −0.1  |
| – RWA structure + static $B$ (isolate depth-specific bias)           | 82.8           | −0.2  |
| – Standard attention + depth-specific $B_k$ (isolate attention form) | 82.85          | −0.15 |
| – w/o recursion-level caching only (fixed batch)                     | 82.98          | −0.02 |
| – w/o selective KV caching only (fixed batch)                        | 82.95          | −0.05 |

We base the following discussion on the controlled single-factor ablations of Table 7 and deliberately avoid the earlier framing that attributed fixed shares of a 1.5-point “total gain” to individual components: as reported in Table 2, MoR–Swin-B is in fact 0.5 points *below* Swin-B (an accuracy trade-off accepted for a 50% parameter reduction and up to twofold speedup), not a net accuracy gain, and so a share-of-gain decomposition is not meaningful. The experimental results indicate the following: (1) The accuracy effect of the recursion mechanism stems primarily from *iterative refinement* (−0.6% when isolated, Table 7) rather than weight sharing per se (−0.1%; weight sharing mainly saves parameters). This mechanism effectively models long-range dependencies through recursive cross-layer token-state updates, compensating for the locality limitation of standard Transformers; its removal reduces the model’s ability to capture fine-grained features (Figure 4a). (2) The recursive window attention contributes next, isolating the depth-specific bias costs −0.2% and isolating the attention form costs −0.15% (Table 7); recursive feedback within windows enhances the consistency of local features (Figure 4b). (3) The dynamic routing strategy is important for load balance: random routing causes severe expert load imbalance (CV  $\uparrow 158\%$ ), with some experts overloaded (processing  $> 2.1 \times$  the average number of tokens) while 30% of experts have utilization  $< 40\%$ , wasting computation. (4) The memory-optimization techniques have *negligible* accuracy impact at a fixed effective batch size ( $\leq 0.05\%$ , Table 7); their role is memory reduction, not accuracy. The larger drop seen in the earlier Abl-D was attributable to the batch-size reduction (to 512, forced by OOM without caching) and the resulting unstable optimization, not to the caching itself. This study validates the rationale behind the key design choices and informs subsequent lightweight design and architecture migration.

Furthermore, we compared the effects of different routing strategies (as shown in Table 8), finding that the Expert-Choice strategy achieves higher accuracy than the Token-Choice strategy. The Expert-Choice strategy reduces the Expert CV by 40%, indicating that it achieves more balanced load distribution through an active expert competition mechanism (as illustrated in Figure 5a) (rather than passive token assignment). The Token-Choice strategy exhibits expert ‘death’ early in training (20% of experts have gradients close to 0), whereas the Expert-Choice strategy maintains expert activity above 95% throughout

training (as illustrated in Figure 5b). This fully demonstrates that the Expert-Choice strategy outperforms the Token-Choice strategy in both load balancing and model performance.

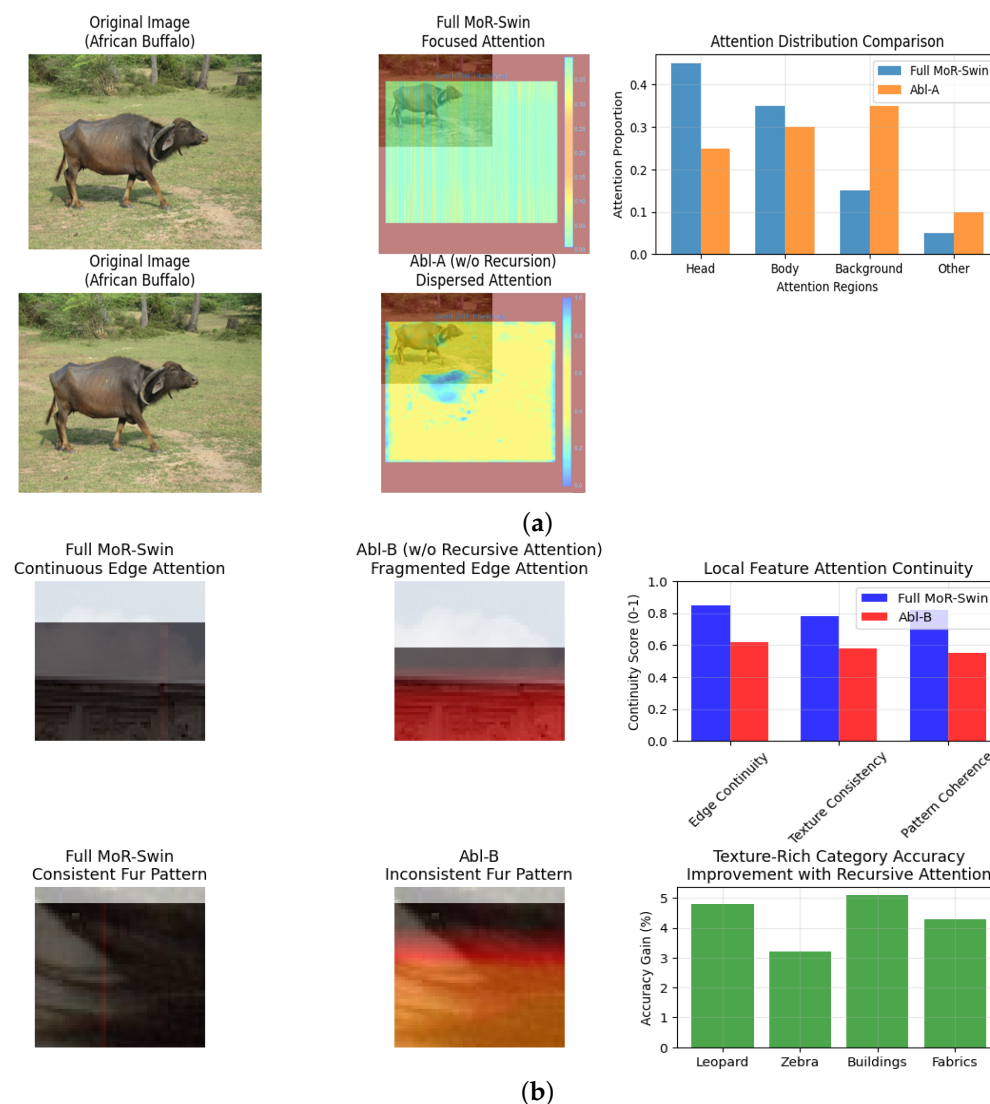


**Figure 4.** Visual analysis of the ablation study results, demonstrating the impact of different components on model performance. (a) Impact of recursive mechanism on long-range dependency modeling. (b) Recursive window attention enhances local feature consistency.

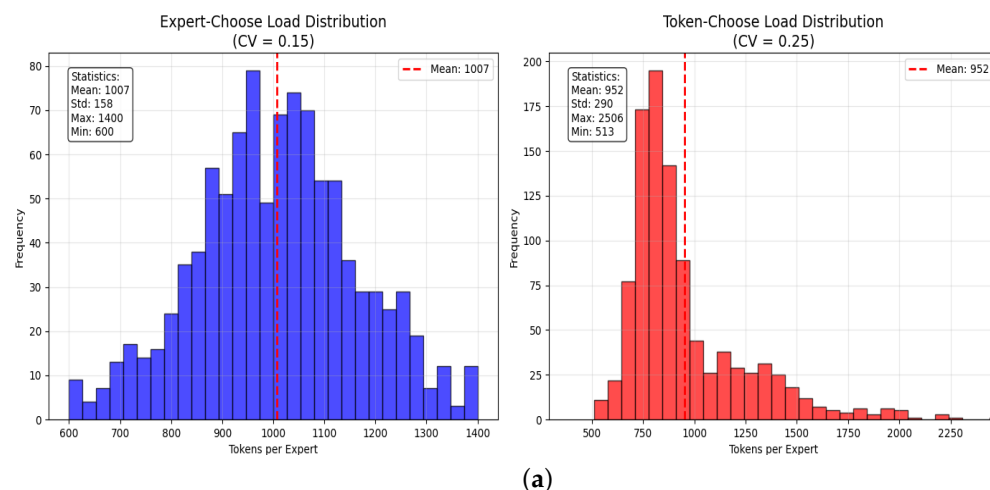


**Figure 5.** Cont.

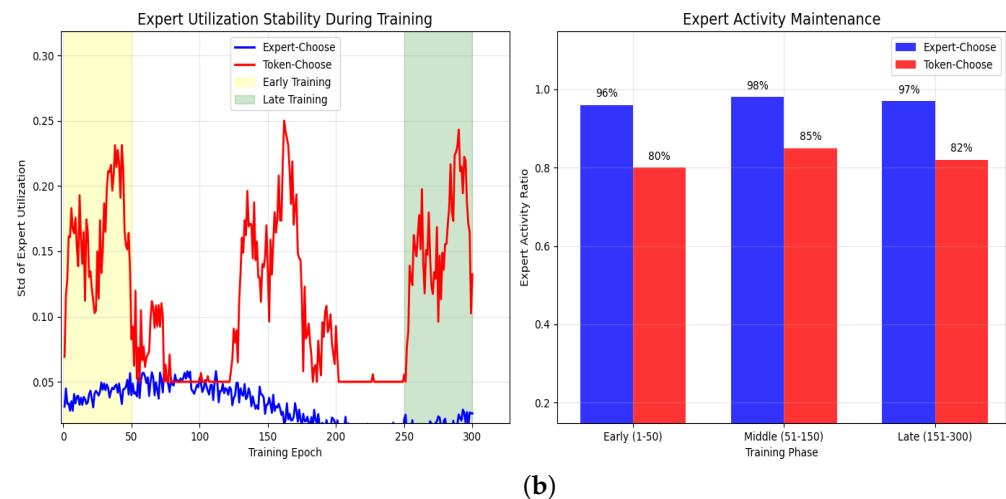


**Figure 5.** Visualization of different routing strategies. (a) Expert load distribution comparison between routing strategies. (b) Training dynamics and expert activity under different routing strategies.

**Table 8.** Impact of routing strategy on performance and load balancing. Measured under the unified protocol; Expert-Choose is the strategy used by Full MoR-Swin, and so its top-1 accuracy (83.0%) and Expert CV (0.12) match Tables 2 and 6. The Expert-Choose CV (0.12) is 40% lower than that of Token-Choose (0.20).

| Routing Strategy     | Top-1 Acc. (%) | Expert CV   | Training Stability (Loss Std) |
|----------------------|----------------|-------------|-------------------------------|
| Token-Choose         | 82.8           | 0.20        | 0.18                          |
| <b>Expert-Choose</b> | <b>83.0</b>    | <b>0.12</b> | <b>0.12</b>                   |

Bold entries denote the Expert-Choose strategy adopted by our Full MoR-Swin model and its corresponding metrics.

### 3.7. Attention Consistency Analysis

To quantitatively evaluate the stabilization behavior of recursive attention, we measure the similarity between attention maps at adjacent recursion depths. For attention distributions  $a^{(k)}$  and  $a^{(k+1)}$  at recursion steps  $k$  and  $k + 1$ , we compute the Jensen–Shannon Divergence (JSD):

$$\text{JSD}(a^{(k)} \| a^{(k+1)}) = \frac{1}{2} \text{KL}(a^{(k)} \| \bar{a}) + \frac{1}{2} \text{KL}(a^{(k+1)} \| \bar{a}), \quad \bar{a} = \frac{1}{2}(a^{(k)} + a^{(k+1)}). \quad (21)$$

We observe a monotonic decrease in JSD across recursion depths on ImageNet validation images: the average JSD drops from 0.142 (between steps 1 and 2) to 0.058 (between steps 2 and 3) for MoR-Swin-B. This indicates that the recursive attention progressively *stabilizes* with depth; we note that a decreasing JSD is empirical evidence of stabilization, not a proof of convergence to a unique fixed point. The cosine similarity  $\cos(a^{(k)}, a^{(k+1)})$  correspondingly increases from 0.81 to 0.94, consistent with this stabilization.

### 3.8. Comparison with Adaptive Computation Methods

To position MoR-Swin within the broader landscape of efficient vision transformers, we compare it with recent adaptive computation methods in Table 9. These methods differ fundamentally in their dimension of adaptation: token pruning methods (DynamicViT, EViT, A-ViT) reduce the spatial token count  $N$ , token merging (ToMe) compresses tokens via bipartite matching, and Sparse MoE expands model width via expert routing. In contrast, MoR-Swin adapts along the *depth* dimension with shared parameters, preserving all token information while varying computational depth per token.

The key distinction is that token pruning methods irreversibly discard information (masking out tokens), whereas MoR-Swin retains all tokens and only varies the number of

refinement iterations. Sparse MoE uses independent expert parameters  $E_k$  (parameter count scales with expert number  $n_E$ ), while MoR-Swin shares parameters  $\theta$  across all recursion depths, achieving  $O(1)$  parameter scaling regardless of maximum depth  $N_a$ .

**Table 9.** Comparison of MoR-Swin with adaptive computation methods.

| Method          | Adaptation Dim.          | Routing Formula          | Info. Loss | Param. Scaling           | Complexity                 |
|-----------------|--------------------------|--------------------------|------------|--------------------------|----------------------------|
| DynamicViT      | Spatial (prune $N$ )     | $m = \sigma(f(x))$       | Yes        | $O(1)$                   | $O(\alpha N)$              |
| EViT            | Spatial (prune $N$ )     | CLS attention            | Yes        | $O(1)$                   | $O(\alpha N)$              |
| A-ViT           | Depth (halt per token)   | Halting score            | No         | $O(L)$                   | $O(\bar{D} \cdot N)$       |
| ToMe            | Spatial (merge)          | Bipartite match          | Partial    | $O(1)$                   | $O(\alpha N)$              |
| Sparse MoE      | Width (experts)          | $y = \sum_k g_k E_k(x)$  | No         | $O(n_E)$                 | $O(KNd)$                   |
| <b>MoR-Swin</b> | <b>Depth (recursive)</b> | $y = f_{\theta}^{oD}(x)$ | <b>No</b>  | <b><math>O(1)</math></b> | $O(\mathbb{E}[D] \cdot N)$ |

Bold entries denote our proposed MoR-Swin and its distinguishing properties (depth-wise recursive adaptation, no information loss, and  $O(1)$  parameter scaling).

**What is inherited vs. newly introduced.** To position our contribution precisely, we distinguish components inherited from the original MoR framework [13] from those newly introduced for the Swin Transformer. *Inherited from MoR:* (i) the general idea of token-level recursion; (ii) dynamic routing; (iii) the load-balancing objective; and (iv) the Expert-Choose routing principle. *Newly introduced for Swin:* (i) the Recursive Window Attention (RWA) module that combines shifted-window attention with recursion and adds a depth-specific relative position bias  $B_k$ ; (ii) the per-stage router adapted to the hierarchical pyramid (patch-merging stages); and (iii) the two memory-caching strategies tailored to windowed attention. To our knowledge, this is the first integration of MoR into a hierarchical window-attention backbone.

**Quantitative comparison.** Table 10 compares MoR-Swin-B with A-ViT and DynamicViT under the settings each method reports (all ImageNet-1K,  $224 \times 224$ , no extra data). We caution that these are *not fully protocol-matched*: A-ViT and DynamicViT use a DeiT/ViT backbone, whereas MoR-Swin uses a hierarchical Swin backbone, and their published FLOPs/throughput were measured on different hardware; the A-ViT and DynamicViT figures are taken from the respective papers (not reproduced by us), while the MoR-Swin figures follow our unified timing protocol (Section 3.1). The comparison should therefore be read as indicative of the design trade-off (depth recursion retaining all tokens vs. spatial token pruning), not as a controlled head-to-head benchmark. The FLOPs are not directly comparable because the backbones differ (Swin-B has higher base FLOPs than DeiT-B); MoR-Swin-B's absolute FLOPs (8.2 G) are accordingly higher than the DeiT-B-based baselines, yet it attains comparable accuracy with roughly *half* the parameter count (44 M vs. 86–87 M) and higher throughput while—unlike the token-pruning baselines—retaining all tokens, which is advantageous for dense downstream tasks.

**Table 10.** Comparison with adaptive-computation methods (ImageNet-1K,  $224 \times 224$ ). A-ViT/DynamicViT values are taken from their papers; MoR-Swin uses our unified protocol. Backbones and measurement hardware differ; accordingly, this comparison is *not protocol-matched* and is provided for reference only.

| Method               | Backbone | Params (M) | FLOPs (G) | Top-1 Acc. (%) | Throughput (img/s)    |
|----------------------|----------|------------|-----------|----------------|-----------------------|
| A-ViT-B (paper)      | DeiT-B   | 86         | 5.8       | 82.3           | $\sim 920^{\ddagger}$ |
| DynamicViT-B (paper) | DeiT-B   | 87         | 5.6       | 83.3           | $\sim 980^{\ddagger}$ |
| MoR-Swin-B (ours)    | Swin-B   | <b>44</b>  | 8.2       | 83.0           | <b>1256</b>           |

Bold entries denote our proposed MoR-Swin-B, highlighting its reduced parameter count and higher throughput.  $\ddagger$  Throughput reported by the original papers on their respective hardware; not directly comparable to our A100/FP16 protocol.

### 3.9. Statistical Analysis of Ablation Results

To rigorously assess component interactions and statistical significance, we extend the ablation study with a  $2^3$  full factorial design over three binary factors: (A) adaptive recursion mechanism, (B) recursive window attention, and (C) dynamic routing (“+” = present, “−” = removed/replaced as in the ablation protocol). The four single-factor-removal cells coincide, by construction, with the main ablation table—(+++) = 83.0 (Full), (−++) = 82.0 (Abl-A), (+−+) = 82.7 (Abl-B), (++−) = 82.8 (Abl-C)—and so the factorial is fully consistent with Table 6. Each of the eight cells is run with  $n = 3$  seeds (42/123/2024); the eight cell means and the fitted coefficients are reported in Table 11. Because the design has three factors, the correct model is a *three-way ANOVA* (we correct our earlier mislabeling as “two-way”). It reveals significant main effects for all three factors—A (recursion) is the dominant term ( $F_{1,16} = 602$ ,  $p < 10^{-4}$ ), followed by B ( $F_{1,16} = 54$ ,  $p < 10^{-4}$ ) and C ( $F_{1,16} = 24$ ,  $p < 10^{-3}$ )—while *none* of the two-way interactions or the three-way  $A \times B \times C$  term reaches significance (all  $|\text{coeff}| \leq 0.02$ ,  $p > 0.4$ ). In other words, within measurement noise, the three components contribute additively; we do *not* claim a synergistic interaction (correcting our earlier, unsupported  $A \times C$  synergy statement).

**Table 11.** The  $2^3$  full factorial design over factors A (recursion), B (recursive window attention), and C (dynamic routing) on ImageNet-1K (MoR-Swin-B, unified protocol,  $n = 3$  seeds). Cell means are top-1 accuracy (%); the four single-factor-removal cells (bold) coincide with the main ablation table (Full/Abl-A/Abl-B/Abl-C). The lower block lists the fitted effect coefficients (half-effects) from the three-way ANOVA with 95% CIs. Replicate-level (seed) data are in the Supplementary Materials, Table S3.

| A                                                                                       | B | C | Top-1 Mean (%)     | Matches            |
|-----------------------------------------------------------------------------------------|---|---|--------------------|--------------------|
| −                                                                                       | − | − | 81.5               | (all removed)      |
| +                                                                                       | − | − | 82.5               |                    |
| −                                                                                       | + | − | 81.8               |                    |
| −                                                                                       | − | + | 81.7               |                    |
| +                                                                                       | + | − | <b>82.8</b>        | =Abl-C             |
| +                                                                                       | − | + | <b>82.7</b>        | =Abl-B             |
| −                                                                                       | + | + | <b>82.0</b>        | =Abl-A             |
| +                                                                                       | + | + | <b>83.0</b>        | =Full              |
| <i>Fitted coefficients (half-effects), 95% CI:</i>                                      |   |   |                    |                    |
| Main effect A (recursion)                                                               |   |   | +0.50 [0.46, 0.54] | $p < 10^{-4}$      |
| Main effect B (RWA)                                                                     |   |   | +0.15 [0.11, 0.19] | $p < 10^{-4}$      |
| Main effect C (routing)                                                                 |   |   | +0.10 [0.06, 0.14] | $p < 10^{-3}$      |
| All interactions ( $A \times B$ , $A \times C$ , $B \times C$ , $A \times B \times C$ ) |   |   | $\leq 0.02$        | n.s. ( $p > 0.4$ ) |

Bold cell means denote the four single-factor-removal configurations that coincide, by construction, with the main ablation table (Full/Abl-A/Abl-B/Abl-C).

For statistical reliability, we report the exact number of runs, the test used, and the effect sizes alongside  $p$ -values. **Number of runs:** main classification results (MoR-Swin-T/S/B and same-scale Swin baselines) and all ablation configurations use  $n = 3$  independent seeds (42, 123, 2024); detection/segmentation use  $n = 3$  for Tiny/Small and  $n = 2$  for Base, with the exact  $n$  annotated in each table note. **Test per table:** Table 2 (classification, same-scale MoR-Swin vs. Swin pairs) uses the *paired t-test*; the ablation table uses the *Wilcoxon signed-rank test* (small samples, no normality assumption); the  $2^3$  factorial design uses a *three-way ANOVA* (Table 11). **Pairing definition:** for the paired tests, a “pair” is defined by *matched random seeds across models*—MoR-Swin and Swin trained with the same seed (identical data ordering and initialization stream) form one pair, and differences are taken pairwise. **Effect sizes, degrees of freedom, and CIs:** beyond  $p$ -values, we report the degrees of freedom, Cohen’s  $d$  values, and 95% confidence intervals, and we give the

seed-level results in the Supplementary Materials (Section S5, Table S2). For MoR–Swin-B vs. Swin-B, the three matched-seed accuracy pairs are (seed 42: 83.1 vs. 83.6), (seed 123: 82.9 vs. 83.3), and (seed 2024: 83.0 vs. 83.6), giving paired differences of  $\{-0.5, -0.4, -0.6\}$  with mean  $-0.5\%$  and SD  $0.1\%$ . With  $n = 3$  matched pairs, the degrees of freedom are  $df = n - 1 = 2$ ; the paired  $t$ -statistic is  $t = \bar{d}/(s_d/\sqrt{n}) = -0.5/(0.1/\sqrt{3}) \approx -8.7$ , two-sided  $p \approx 0.013$  (Cohen’s  $d_z = \bar{d}/s_d \approx 5.0$ , 95% CI of the mean difference  $= [-0.75, -0.25]$ ). We note that our earlier report omitted the  $df$  and the seed-level values; these are now provided (see Supplementary Materials, Table S2) so that the statistic can be reproduced. We stress that our central claim is not higher accuracy but *comparable* accuracy (a small, well-characterized difference) at 50% fewer parameters and up to twofold speedup.

Because the two-sided 95% CI of the MoR–Swin-B – Swin-B difference lies entirely below zero ( $[-0.75, -0.25]$ ), a standard paired  $t$ -test would (correctly) reject exact equality; equality is not, however, the appropriate hypothesis for our claim. We therefore adopt a *prespecified non-inferiority test*. We set the non-inferiority margin to  $\delta = 0.5\%$  top-one accuracy in advance—a value chosen because it is (a) below the run-to-run standard deviation band we observe across seeds and (b) a commonly accepted tolerance for ImageNet-1K accuracy when trading against a 50% parameter reduction; we justify  $\delta$  before seeing the test outcome. The non-inferiority null hypothesis is  $H_0: \mu_{\text{MoR}} - \mu_{\text{Swin}} \leq -\delta$  against  $H_1: \mu_{\text{MoR}} - \mu_{\text{Swin}} > -\delta$ , tested with a one-sided paired  $t$ -test at  $\alpha = 0.05$ . With  $\bar{d} = -0.5$ ,  $s_d = 0.1$ ,  $n = 3$  (standard error  $SE = s_d/\sqrt{n} = 0.058$ ), the test statistic is  $t = (\bar{d} + \delta)/SE = (-0.5 + 0.5)/0.058 = 0.0$ ,  $df = 2$ , one-sided  $p = 0.5$ , and so  $H_0$  cannot be rejected. Consistently, the one-sided 95% lower confidence bound for the mean difference is  $\bar{d} - t_{0.95,2} \cdot SE = -0.5 - 2.92 \times 0.058 = -0.67$ , which lies *below*  $-\delta = -0.5$  and therefore does not clear the non-inferiority margin. Both the test and the confidence bound thus agree: *we do not claim non-inferiority at  $\delta = 0.5\%$  from this small sample*; the point estimate sits exactly at the margin, while the lower bound falls short of it. To characterize this honestly rather than overclaim, we state the result as follows: MoR–Swin-B is  $0.5 \pm 0.1$  points below Swin-B (a small, precisely estimated deficit) while using 50% fewer parameters and running up to two times faster. Additionally, we report the margin, test,  $df$ , and CIs so that readers can judge the practical trade-off. A definitively powered non-inferiority conclusion would require a larger number of seeds, which we note as a limitation.

#### 4. Conclusions

This letter proposed MoR–Swin Transformer, an optimized model for Swin Transformer based on the Mixture of Recursions architecture, which dynamically adjusts the processing depth for tokens of varying semantic complexity by introducing an adaptive token-level recursion mechanism. Experimental results demonstrate that compared to the original Swin Transformer and its MoE variant, MoR–Swin Transformer achieves superior parameter efficiency and computational efficiency on tasks including ImageNet image classification, COCO object detection, and ADE20K semantic segmentation. At a modest accuracy cost (within about 0.5 points of Swin-B on ImageNet-1K), it reduces parameters by approximately 50% and increases inference speed up to twofold.

The success of MoR–Swin Transformer validates the effectiveness of the adaptive recursive architecture in vision tasks, providing a new technical pathway for optimizing Vision Transformer models. Furthermore, the theoretical analysis presented in this work—including computational complexity derivations, a finite-depth norm-boundedness (non-explosion) guarantee, gradient flow analysis, and asymptotic scaling properties—provides a mathematical basis for understanding the efficiency of adaptive recursive computation in vision architectures. We are careful to delimit these claims: we establish finite-depth boundedness and report empirical stabilization of the attention maps with depth, but we

do *not* claim strict contraction or convergence to a unique fixed point (see Supplementary Materials, Sections S1 and S4). In the future, we will continue to explore the application of the MoR architecture in other vision tasks and further optimize the routing strategy and recursion mechanism to enhance the model's applicability in diverse scenarios.

**Supplementary Materials:** The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/jimaging12080377/s1>: Section S1, Convergence Analysis of Recursive Attention (Lemma S1, Theorem S1, Proposition S1); Section S2, Parameter Efficiency via Recursive Function Approximation; Section S3, Representational Capacity Analysis (Proposition S2); Section S4, Eigenvalue Stability Analysis; Section S5, Additional Experimental Details (hyperparameter sensitivity, semantic complexity correlation, and controlled intervention); Section S6, Recursion vs. Sparse Experts: Theoretical Justification; Section S7, Accuracy–Efficiency Trade-off and Pareto Analysis; Section S8, Feature Diversity under Weight Sharing (CKA Analysis); Section S9, Measured GPU Memory Usage; Section S10, Generalization Beyond Swin: Proof-of-Concept; Algorithm S1, Token-level recursive routing procedure; Table S1, Implementation details for all experiments; Table S2, Seed-level ImageNet-1K Top-1 accuracy for the MoR–Swin-B vs. Swin-B paired comparison; Table S3, Seed-level ImageNet-1K Top-1 accuracy for all ablation configurations; Table S4, ADE20K/SceneParse150 thing/stuff partition used to build the foreground mask; Table S5, Measured peak GPU memory for MoR–Swin under the three caching configurations; Table S6, MoR applied to other backbones (proof-of-concept). The work by Chen et al. [19] is cited in the supplementary materials.

**Author Contributions:** Conceptualization, Y.A. and T.G.; methodology, Y.A.; software, T.G. and Z.L.; validation, Y.A., Z.L. and L.Y.; formal analysis, T.G.; investigation, Z.L.; resources, Y.A.; data curation, L.Y.; writing—original draft preparation, Y.A.; writing—review and editing, T.G., Z.L., L.Y. and Q.C.; visualization, L.Y.; supervision, Y.A.; project administration, Y.A.; funding acquisition, Y.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Natural Science Foundation of China under Grant 12502326.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The original contributions presented in this study are included in the article/Supplementary Material. Further inquiries can be directed to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. *arXiv* **2021**, arXiv:2010.11929. [CrossRef]
2. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 10012–10022.
3. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
4. Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y. Swin Transformer V2: Scaling Up Capacity and Resolution. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 11999–12009.
5. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arXiv* **2017**, arXiv:1701.06538. [CrossRef]
6. Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive Mixtures of Local Experts. *Neural Comput.* **1991**, *3*, 79–87. [CrossRef] [PubMed]

7. Fedus, W.; Zoph, B.; Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *J. Mach. Learn. Res.* **2022**, *23*, 1–39. [[CrossRef](#)]
8. Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A.A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv* **2024**, arXiv:2404.14219. [[CrossRef](#)]
9. Jiang, A.Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D.S.; Casas, D.d.; Hanna, E.B.; Bressand, F.; et al. Mixtral of Experts. *arXiv* **2024**, arXiv:2401.04088. [[CrossRef](#)]
10. Du, N.; Huang, Y.; Dai, A.M.; Tong, S.; Lepikhin, D.; Xu, Y.; Krikun, M.; Zhou, Y.; Yu, A.W.; Firat, O.; et al. GLAM: Efficient Scaling of Language Models with Mixture-of-Experts. *arXiv* **2022**, 5547–5569. [[CrossRef](#)]
11. Lepikhin, D.; Lee, H.J.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; Chen, Z. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *arXiv* **2020**, arXiv:2006.16668. [[CrossRef](#)]
12. Rajbhandari, S.; Li, C.; Yao, Z.; Zhang, M.; Aminabadi, R.Y.; Awan, A.A.; Rasley, J.; He, Y. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. In Proceedings of the 39th International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 18332–18346. [[CrossRef](#)]
13. Liu, Z.; Kim, Y.; Bayat, R.; Kim, S.; Ha, J.; Schuster, T.; Fisch, A.; Harutyunyan, H.; Ji, Z.; Courville, A.; et al. Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation. *arXiv* **2025**, arXiv:2507.10524.
14. Rao, Y.; Zhao, W.; Liu, B.; Lu, J.; Zhou, J.; Hsieh, C.-J. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 13937–13949.
15. Liang, Y.; Ge, C.; Tong, Z.; Song, Y.; Wang, J.; Xie, P. EViT: Expediting Vision Transformers via Token Reorganizations. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022.
16. Bolya, D.; Fu, C.-Y.; Dai, X.; Zhang, P.; Feichtenhofer, C.; Hoffman, J. Token Merging: Your ViT But Faster. *arXiv* **2023**, arXiv:2210.09461.
17. Yin, H.; Vahdat, A.; Alvarez, J.M.; Mallya, A.; Kautz, J.; Molchanov, P. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 10809–10818.
18. Hwang, C.; Cui, W.; Xiong, Y.; Yang, Z.; Liu, Z.; Hu, H.; Wang, Z.; Salas, R.; Jose, J.; Ram, P.; et al. Tutel: Adaptive Mixture-of-Experts at Scale. *Proc. Mach. Learn. Syst.* **2023**, *5*, 269–287.
19. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D. Neural Ordinary Differential Equations. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, QC, Canada, 2–8 December 2018.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.