ADAPTIVE TOKEN–BLOCK MIXING FOR EFFICIENT VISION ENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient visual representation learning requires fine-grained token selection and access to long-range context. Local attention restricts direct cross-region interaction, whereas global associative aggregation can limit contextual diversity. We introduce **Adaptive Token-Block Mixing (ATBM)**, a visual mixing operator that combines within-block softmax attention with structured communication over block-specific key-value states. A head-specific Toeplitz mixer propagates these states using learned functions of relative block-index distance, after which token queries retrieve cross-block context. Content-dependent gates fuse the local and cross-block outputs for each token and attention head. This design preserves explicit token interaction within blocks while avoiding a dense global token-to-token attention matrix. On ImageNet-1K, the tiny version ATBM-T achieves 77.2% top-1 accuracy with 6M parameters and 1.2 GFLOPs, improving over DeiT-T by 5.0 percentage points at the same reported model size. At 1280×1280 resolution, ATBM-T achieves a $5.73\times$ inference speedup over DeiT-T while reducing peak GPU memory usage by 82.3%. Evaluations in plain and hierarchical encoders, image generators, together with transfer experiments on COCO and ADE20K, further demonstrate the applicability of the proposed operator.

1 INTRODUCTION

Vision Transformers (ViTs) have established token interaction as a powerful paradigm for visual representation learning (Dosovitskiy et al., 2021; Touvron et al., 2021). Global self-attention provides query-dependent access to all visual tokens, but evaluating dense pairwise interactions incurs quadratic computation. Explicitly storing the attention matrix also requires quadratic memory, making these costs increasingly restrictive as token resolution grows.

Efficient attention mechanisms adopt more economical interaction structures. Local attention preserves explicit token selection within windows or neighborhoods but requires additional operations for cross-region communication (Liu et al., 2021; Hassani et al., 2023). Linear attention instead aggregates key-value information into reusable associative states, enabling efficient access to global context (Katharopoulos et al., 2020; Choromanski et al., 2021). However, sharing a compact global state can limit contextual diversity, motivating locality-aware formulations, multi-scale aggregation, and spatially structured or multiple-state representations (Sun et al., 2023; Han et al., 2023; Cai et al., 2023; Fan et al., 2024; Zhang et al., 2026).

These observations motivate representing visual interactions at different granularities. Focal attention combines fine-grained nearby interaction with coarser distant context (Yang et al., 2021). In language modeling, FLASH combines within-chunk quadratic attention with cross-chunk associative aggregation, including the multi-head MC-TFM++ variant (Hua et al., 2022). MHLA constructs block-specific key-value summaries and mixes them using learned query-block-specific coefficients before query retrieval (Zhang et al., 2026). Building on these developments, we investigate two coupled questions: *how should associative block states communicate, and how should their retrieved context be combined with explicit within-block interaction?*

We introduce **Adaptive Token-Block Mixing (ATBM)**, a visual mixing operator coupling distance-structured state communication with content-adaptive fusion. ATBM partitions the raster-ordered visual token sequence into non-overlapping contiguous blocks. Its local pathway performs standard

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

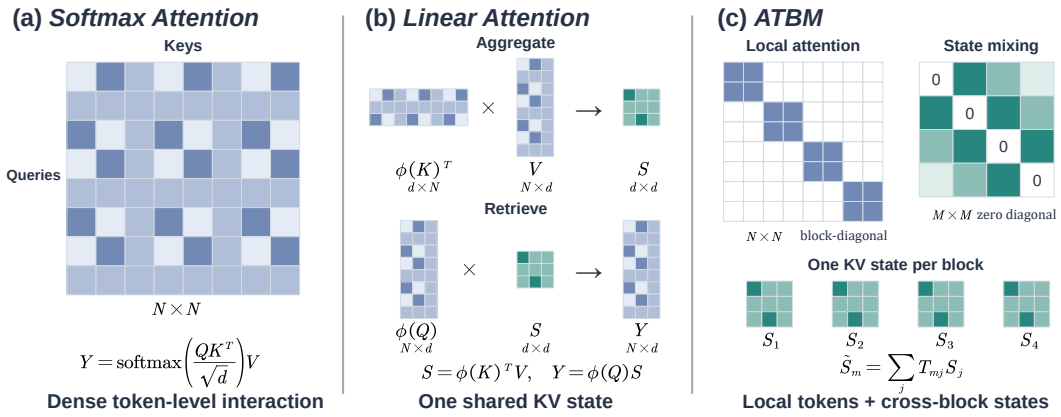


Figure 1: **From token-level interaction to token-block mixing.** (a) Softmax attention performs dense pairwise interaction over all tokens. (b) Standard linear attention aggregates global context into one shared key-value state per head. (c) ATBM combines within-block softmax attention with structured communication among block-specific associative states.

within-block softmax attention, preserving explicit query-dependent token selection. Its state pathway constructs an associative key-value state for each block and propagates these states through a **head-specific Toeplitz state mixer**. The coefficients are learned functions of relative block-index distance, parameterized using exponential components. This structure allows different heads to learn different distance profiles with a parameter count independent of the number of blocks. Self-block propagation is excluded, reserving within-block interaction for the softmax pathway.

Token queries retrieve context from the propagated states: the distance kernel structures block communication, while query features determine token-level retrieval. Because the useful balance between within-block selection and cross-block context can vary across visual features, ATBM predicts a content-dependent fusion coefficient for every token and attention head. Both pathways remain active, with the gates controlling their relative contributions. As illustrated in Fig. 1, ATBM retains explicit token interaction within blocks while transferring cross-block communication to associative states, avoiding a dense global token-to-token attention matrix.

Experiments validate ATBM across plain and hierarchical image classification, dense prediction on COCO and ADE20K, and image generation. Our contributions are summarized as follows:

- We introduce **Adaptive Token-Block Mixing (ATBM)**, a visual mixing operator combining within-block softmax attention with structured communication over associative block states.
- We design a compact head-specific Toeplitz state mixer and content-dependent token-head fusion to structure cross-block communication and adaptively combine the two pathways.
- We evaluate ATBM in plain and hierarchical encoders, demonstrate improvements over corresponding baselines on classification, generation and dense prediction, and analyze its practical efficiency and component contributions.

2 Related Work

Efficient Local-Global Interaction. Efficient vision Transformers reduce global attention costs through restricted or compressed spatial interactions. PVT (Wang et al., 2021) spatially reduces keys and values, while Swin Transformer (Liu et al., 2021) uses shifted local windows. Twins (Chu et al., 2021) combines local and spatially subsampled global attention, CrossFormer (Wang et al., 2022) separates short- and long-distance interactions, and RegionViT (Chen et al., 2022) connects local tokens through regional tokens. Focal Transformer (Yang et al., 2021) represents nearby and distant context at different granularities. These approaches motivate combining fine-grained local interaction with compact representations of broader visual context.

In language modeling, FLASH (Hua et al., 2022) combines quadratic within-chunk attention with linear cross-chunk aggregation, including a multi-head variant, MC-TFM++. ATBM shares this

local-associative organization and develops a visual mixing operator with head-specific distance-based state propagation and token-wise, head-wise fusion with local softmax attention.

Linear Attention for Visual Representation Learning. Linear attention aggregates key-value context associatively to avoid explicit quadratic token interactions (Katharopoulos et al., 2020; Choromanski et al., 2021). Visual variants improve locality and expressivity through two-dimensional vicinity modeling in VVT (Sun et al., 2023), focused feature mappings in FLatten Transformer (Han et al., 2023), multi-scale aggregation in EfficientViT (Cai et al., 2023), spatially decaying retention in RMT (Fan et al., 2024), and learnable spatially decay in SDT (Mao et al., 2026). MLLA (Han et al., 2024a) connects selective state-space models with linear attention, highlighting forgetting and block design, while MALA (Fan et al., 2025) introduces magnitude-aware attention to reduce the discrepancy with softmax attention. MHLA (Zhang et al., 2026) addresses global context collapse through multiple key-value summaries and query-conditioned mixing. ATBM instead propagates block states using head-specific relative-distance coefficients before content-dependent query retrieval, and adaptively combines the retrieved context with within-block softmax attention.

Structured Sequence Mixing and Adaptive Fusion. Toeplitz Neural Network (TNN) (Qin et al., 2023) uses relative-position-parameterized Toeplitz mixing with exponential decay for sequence modeling. ATBM applies Toeplitz mixing to associative key-value matrices, introducing head-specific cross-block state communication before query retrieval. Its symmetric distance kernel excludes self-block propagation, reserving within-block interaction for the local softmax pathway. State-space approaches such as Vision Mamba (Zhu et al., 2024) and VMamba (Liu et al., 2024) model visual context through bidirectional state-space modeling and two-dimensional selective scanning, respectively. ATBM constructs block states independently and communicates between them through a distance-structured operator. For adaptive contextualization, FocalNet (Yang et al., 2022) gates context from different receptive-field levels. ATBM uses token-wise, head-wise gates to fuse local token outputs and retrieved state context. Both pathways remain active, so gating controls their relative contribution rather than the attention computation budget.

3 METHOD

3.1 OVERVIEW

We introduce **Adaptive Token-Block Mixing (ATBM)**, a visual mixing operator that separates fine-grained token interaction from cross-block communication. ATBM consists of three components: local softmax attention within token blocks, Toeplitz mixing over compact associative block states, and content-dependent routing between the two pathways. Cross-block interaction is performed over block states, while token-level queries retain content-dependent retrieval.

Given an input sequence $\mathbf{X} \in \mathbb{R}^{B\times N\times D}$, we compute query, key, and value projections as follows:

$$[\mathbf{Q}, \mathbf{K}, \mathbf{V}] = \mathbf{X}\mathbf{W}_{qkv}, \tag{1}$$

and split them into h attention heads of dimension $d = D/h$. Visual tokens are ordered according to the raster scan of the patch grid and partitioned into $M = \lceil N/C \rceil$ non-overlapping blocks. The final block is zero-padded when necessary, and padded positions are excluded from subsequent aggregation. For block m and head h, we denote the corresponding features by $\mathbf{Q}^h_m, \mathbf{K}^h_m, \mathbf{V}^h_m \in \mathbb{R}^{C\times d}$. We omit the batch dimension from block-level notation for clarity.

3.2 LOCAL TOKEN MIXING

Within each block, we preserve explicit token-to-token interaction using standard scaled dot-product attention. For block m and head h:

$$\mathbf{A}^h_m = \mathrm{Softmax}\left(\frac{\mathbf{Q}^h_m(\mathbf{K}^h_m)^\top}{\sqrt{d}} + \mathbf{P}_m\right), \qquad \mathbf{O}^h_{L,m} = \mathbf{A}^h_m\mathbf{V}^h_m. \tag{2}$$

The additive mask $\mathbf{P}_m \in \mathbb{R}^{C\times C}$ is zero at valid key positions and $-\infty$ at padded ones. Outputs at padded query positions are discarded. Thus, $\mathbf{A}^h_m \in \mathbb{R}^{C\times C}$ and $\mathbf{O}^h_{L,m} \in \mathbb{R}^{C\times d}$, with explicit token-to-token attention confined to blocks of size C.

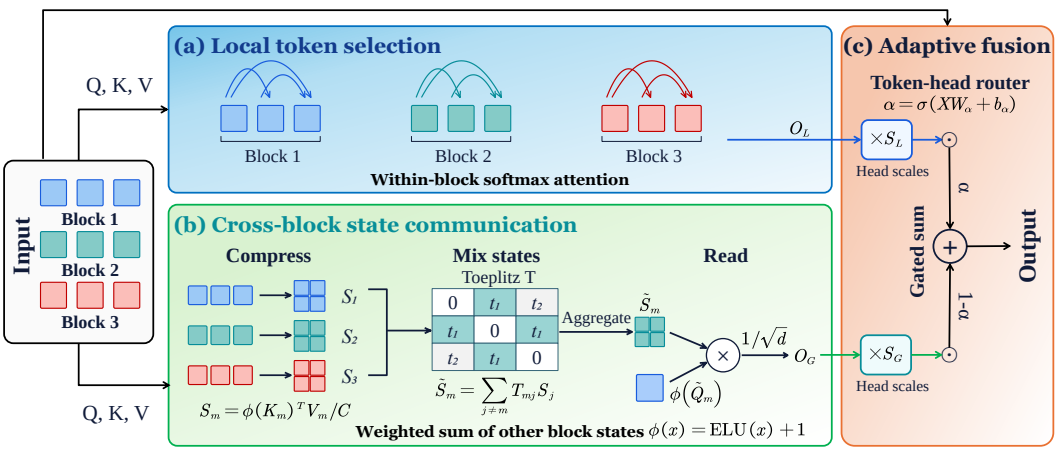


Figure 2: **Adaptive Token-Block Mixing (ATBM).** ATBM combines (a) within-block softmax attention, (b) Toeplitz communication over associative block states followed by token-level retrieval, and (c) token-wise, head-wise routing with per-head path calibration.

3.3 Cross-Block State Mixing

Associative block states. We summarize each block into an associative key-value state. Let $\mathbf{p}_m \in \{0,1\}^C$ denote the validity mask of block m, $C_m = \sum_{c=1}^{C} p_{m,c}$ its number of valid tokens, and $\mathbf{D}_m = \mathrm{Diag}(\mathbf{p}_m)$. The state for head h is:

$$\mathbf{S}_m^h = \frac{1}{C_m} \phi(\mathbf{K}_m^h)^\top \mathbf{D}_m \mathbf{V}_m^h, \tag{3}$$

where $\mathbf{S}_m^h \in \mathbb{R}^{d \times d}$ and $\phi(x) = \mathrm{ELU}(x) + 1$ is applied element-wise. The mask removes padded tokens from the aggregation, while normalization by C_m makes the state independent of the number of valid positions in the block.

Toeplitz communication. We model interactions between block states as a function of their relative block distance. For each head h, we define:

$$\kappa_h(\delta) = \sigma\left(b_h + \sum_{r=1}^{R} a_{h,r} \exp(-\lambda_{h,r}\delta)\right), \qquad T_{ij}^h = \mathbb{1}_{i \neq j}\, \kappa_h(|i-j|), \tag{4}$$

where $\sigma(\cdot)$ is the logistic sigmoid and R denotes the number of exponential components. The parameters $a_{h,r}$ and $\lambda_{h,r}$ are constrained to be positive through softplus parameterization, while b_h is an unconstrained head-specific bias.

Because T_{ij}^h depends only on $|i-j|$, $\mathbf{T}^h \in \mathbb{R}^{M \times M}$ is a symmetric Toeplitz matrix. We set $T_{ii}^h = 0$ to reserve this pathway for cross-block communication. For off-diagonal entries, the mixing strength decreases monotonically with block distance and approaches $\sigma(b_h)$ asymptotically. Using multiple exponential components allows different heads to represent mixtures of decay scales.

The propagated state for target block m and its token-level retrieval are:

$$\widetilde{\mathbf{S}}_m^h = \frac{1}{\max(M-1,\,1)} \sum_{j=1}^{M} T_{mj}^h \mathbf{S}_j^h, \qquad \mathbf{O}_{G,m}^h = \frac{1}{\sqrt{d}}\, \phi(\mathbf{Q}_m^h) \widetilde{\mathbf{S}}_m^h. \tag{5}$$

where $\mathbf{O}_{G,m}^h \in \mathbb{R}^{C \times d}$. The Toeplitz kernel determines how information is propagated across blocks, whereas the query features determine what each token retrieves from the propagated state. Cross-block interaction therefore occurs without constructing a global $N \times N$ token attention matrix.

The kernel contains only $2R + 1$ learned scalars per head, independent of the number of blocks. In our dense implementation, constructing the Toeplitz weights requires $\mathcal{O}(HRM^2)$ operations and $\mathcal{O}(HM^2)$ memory, while propagating the $d \times d$ block states costs $\mathcal{O}(HM^2d^2)$ operations. State construction and query retrieval each require $\mathcal{O}(HNd^2)$ operations. Importantly, cross-block dependencies are indexed by M block states rather than N individual tokens.

Table 1: Results of hierarchical vision backbones on ImageNet-1K. ‡ denotes models trained with MESA (Du et al., 2022).

Method	Params	FLOPs	Top-1
ConvNeXt-T	29M	4.5G	82.1
InternImage-T	30M	5.0G	83.5
Swin-T	28M	4.6G	81.3
BiFormer-S	26M	4.5G	83.8
TransNeXt-T	28M	5.7G	84.0
RMT-S	27M	4.5G	84.1
EfficientVMamba-B	33M	4.0G	81.8
LocalVMamba-T	26M	5.7G	82.7
VMamba-T	31M	4.9G	82.5
MSVMamba-T	32M	5.1G	83.0
FLatten-CSwin-T	21M	4.3G	83.1
FLatten-CSwin-S	35M	6.9G	83.8
SOFT-S++	27M	4.5G	82.6
VVT-S	26M	5.6G	82.7
ViG-H-T	29M	4.5G	82.8
MLLA-T‡	25M	4.2G	83.5
H-ATBM-T	29M	5.0G	83.6
ConvNeXt-S	50M	8.7G	83.1
InternImage-S	50M	8.0G	84.2
Swin-S	50M	8.7G	83.0
GC-ViT-S	51M	8.5G	84.3
TransNeXt-S	50M	10.3G	84.7
RMT-B	54M	9.7G	85.0
BiFormer-B	57M	9.8G	84.3
VMamba-S	50M	8.7G	83.6
LocalVMamba-S	50M	11.4G	83.7
MSVMamba-S	50M	8.8G	84.1
SOFT-M++	48M	8.7G	83.7
VVT-M	48M	9.4G	83.8
ViG-H-S	50M	8.8G	83.8
MLLA-S‡	43M	7.3G	84.4
H-ATBM-S	56M	9.0G	84.5
ConvNeXt-B	89M	15.4G	83.8
InternImage-B	97M	16.0G	84.9
iFormer-L	87M	14.0G	84.8
TransNeXt-B	90M	18.4G	84.8
GC-ViT-B	90M	14.8G	85.0
VMamba-B	89M	15.4G	83.9
MSVMamba-B	91M	16.3G	84.4
FLatten-CSwin-B	75M	15.0G	84.5
SOFT-L++	85M	15.4G	84.1
ViG-H-B	89M	15.5G	84.2
MLLA-B‡	96M	16.2G	85.3
H-ATBM-B	96M	17.0G	84.8

3.4 Adaptive Token-Block Routing

The local and cross-block pathways capture complementary interactions, and their relative importance may vary across both tokens and attention heads. We therefore predict a routing coefficient for each token-head pair:

$$\boldsymbol{\alpha} = \sigma\left(\mathbf{X}\mathbf{W}_{\alpha} + \mathbf{b}_{\alpha}\right), \tag{6}$$

where $\boldsymbol{\alpha} \in [0,1]^{B\times N\times H}$. After block partitioning, the coefficients for block m and head h are denoted by $\boldsymbol{\alpha}_m^h \in [0,1]^{C\times 1}$. Both pathways are evaluated for every valid token; the router controls their relative contribution rather than conditionally executing either pathway.

Path calibration and fusion. The two pathways use different aggregation mechanisms and therefore need not operate at the same feature scale. To separate scale alignment from content-dependent routing, we introduce positive per-head calibration factors:

$$s_L^h = \exp(\gamma_L^h), \qquad s_G^h = \exp(\gamma_G^h), \tag{7}$$

where γ_L^h and γ_G^h are initialized to zero. The final output for block m and head h is:

$$\mathbf{O}_m^h = \boldsymbol{\alpha}_m^h \odot \left(s_L^h \mathbf{O}_{L,m}^h\right) + \left(1 - \boldsymbol{\alpha}_m^h\right) \odot \left(s_G^h \mathbf{O}_{G,m}^h\right), \tag{8}$$

where $\boldsymbol{\alpha}_m^h$ is broadcast along the feature dimension. The calibration factors are shared across tokens within each head, whereas the routing coefficients vary with the input. Consequently, larger values of $\boldsymbol{\alpha}_m^h$ favor local token interaction, while smaller values favor cross-block state retrieval.

4 Experiments

4.1 Experimental Setup

Backbones. We consider two complementary architectures. **H-ATBM** follows a four-stage Swin-style hierarchy (Liu et al., 2021) with progressive spatial downsampling, whereas **ATBM** follows the plain DeiT architecture (Touvron et al., 2021) and maintains a fixed token resolution.

ATBM configuration. We use a block size of $C = 49$ and $R = 8$ exponential components for each head-specific Toeplitz kernel. Tokens are flattened in raster-scan order and partitioned into

non-overlapping blocks, with zero padding applied only when necessary. The Toeplitz kernel is parameterized independently for each head, with characteristic half-lives initialized logarithmically from 1 to 64 blocks.

Setup. We train all classification models from scratch on ImageNet-1K for 300 epochs using AdamW (Loshchilov & Hutter, 2017), a global batch size of 1024, and a peak learning rate of 1×10^{-3}. The learning rate follows cosine decay with a 20-epoch linear warm-up. We use RandAugment (Cubuk et al., 2020), Mixup (Zhang et al., 2018), CutMix (Yun et al., 2019), and Random Erasing (Zhong et al., 2020).

4.2 Image Classification

Hierarchical backbones. As shown in Table 1, H-ATBM delivers strong performance across different model scales, achieving 83.6%, 84.5%, and 84.8% Top-1 accuracy for the T, S, and B variants, respectively. Compared with the corresponding Swin models, H-ATBM improves Top-1 accuracy by 2.3 and 1.5 percentage points for the T and S variants, while operating within comparable computational budgets. H-ATBM also compares favorably with representative linear-attention architectures. In particular, H-ATBM-T outperforms FLatten-CSwin-T, SOFT-S++, VVT-S, and MLLA-T, while H-ATBM-S achieves 84.5%, slightly surpassing MLLA-S at 84.4%. At the base scale, H-ATBM-B reaches 84.8%, outperforming FLatten-CSwin-B, SOFT-L++, and ViG-H-B, and remaining competitive with MLLA-B at 85.3%.

Table 2: Results of plain vision backbones on ImageNet-1K.

Method	Params	FLOPs	Top-1
DeiT-T	6M	1.2G	72.2
LightNet	6M	–	74.5
HGRN	6M	–	75.4
Vim-T	7M	1.5G	76.1
Agent-DeiT-T	6M	1.2G	74.9
ATBM-T	6M	1.2G	77.2
DeiT-S	22M	4.6G	79.8
LightNet	23M	–	80.1
HGRN	24M	–	80.1
Vim-S	26M	5.1G	80.3
Agent-DeiT-S	23M	4.4G	80.5
ATBM-S	24M	4.8G	81.4
DeiT-B	87M	17.6G	81.8
LightNet	87M	–	81.9
ATBM-B	93M	18.4G	82.4

Plain backbones. We further isolate the effect of the token mixer using a DeiT-style architecture with fixed token resolution. As shown in Table 2, ATBM consistently improves over the corresponding DeiT baselines across all three scales. ATBM-T achieves 77.2% Top-1 accuracy with 6M parameters and 1.2G FLOPs, improving DeiT-T by 5.0 points. ATBM-S reaches 81.4%, a 1.6-point improvement over DeiT-S, while ATBM-B reaches 82.4%, compared with 81.8% for DeiT-B. ATBM-T and ATBM-S also outperform the reported results of LightNet (Qin et al., 2024a), HGRN (Qin et al., 2024b), Vim (Zhu et al., 2024), and Agent-DeiT (Han et al., 2024b) at comparable model scales. The consistent gains in this non-hierarchical setting indicate that the effectiveness of ATBM does not depend on progressive spatial downsampling.

Overall, the hierarchical and plain evaluations provide complementary evidence that ATBM can serve as an effective token-mixing operator under substantially different backbone organizations.

4.3 Object Detection and Instance Segmentation

Setup. We further evaluate the transferability of ATBM to dense prediction tasks on COCO 2017 (Lin et al., 2014). Following ViTDet (Li et al., 2022), the backbones are initialized with ImageNet-1K pretrained weights and fine-tuned on COCO 2017. We adopt the large-scale jittering (LSJ) augmentation at 1024×1024 resolution. All models are trained for 50 epochs using the AdamW optimizer with a base learning rate of 1×10^{-4}, and a batch size of 32, combined with layer-wise learning-rate decay of 0.7. A SimpleFPN produces a five-level pyramid feeding a Cascade Mask R-CNN (Cai & Vasconcelos, 2019) head.

Table 3: Results of object detection and instance segmentation on the COCO val set.

Metric	DeiT-T	ATBM-T
FLOPs	457G	384G
AP^b	42.3	44.2
AP^b50	62.3	63.6
AP^m	36.7	38.3
AP^m50	58.3	60.2

Results. As reported in Table 3, replacing the DeiT-T backbone with ATBM-T consistently improves performance while reducing computational cost. Specifically, ATBM-T improves box AP

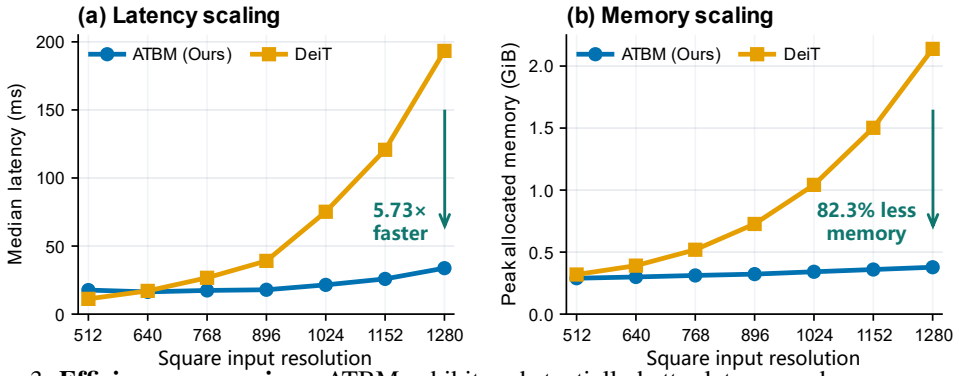


Figure 3: **Efficiency comparison.** ATBM exhibits substantially better latency and memory scaling.

from 42.3 to 44.2 (+1.9 AP) and box AP50 from 62.3 to 63.6 (+1.3 AP). For instance segmentation, mask AP increases from 36.7 to 38.3 (+1.6 AP), while mask AP50 improves from 58.3 to 60.2 (+1.9 AP). Meanwhile, the total computational cost is reduced from 457G to 384G FLOPs. These results demonstrate that the advantages of ATBM extend to object detection and instance segmentation, yielding better accuracy with substantially lower computational cost.

4.4 Semantic Segmentation

Setup. We evaluate the transferabiliy of ATBM on the ADE20K dataset (Zhou et al., 2017) using UperNet (Xiao et al., 2018) as the segmentation framework. The results are summarized in Table 4.

Table 4: Results of semantic segmentation results on ADE20K.

Backbone	Params	FLOPs	mIoU↑
ResNet-50	67M	–	41.2
ResNet-101	86M	–	44.9
DeiT-T	11M	49G	39.2
DeiT-S	43M	186G	44.0
ATBM-T	12M	44G	40.7
ATBM-S	47M	176G	**45.2**

Results. ATBM consistently improves over DeiT under comparable model sizes. ATBM-T achieves 40.7 mIoU with 12M parameters, outperforming DeiT-T by 1.5 points while requiring lower backbone FLOPs. Similarly, ATBM-S reaches 45.2 mIoU with 47M parameters, surpassing DeiT-S by 1.2 points, with FLOPs reduced from 186G to 176G. These results demonstrate that ATBM provides a better accuracy-efficiency trade-off for dense prediction and improves semantic representation without substantially increasing model size. This advantage becomes more pronounced at higher resolutions (Fig. 3). We benchmark the Tiny-scale ATBM and DeiT backbones under the same UperNet configuration on an RTX 3090 with batch size 1, an fused attention implementations. Latency is the median of 50 synchronized runs after 15 warm-up iterations. Memory is the peak PyTorch-allocated CUDA memory. Under this setting, ATBM is 5.73× faster and uses 82.3% less memory.

4.5 Image Generation

Setup. We evaluate ATBM under a generative objective by incorporating it into DiT (Peebles & Xie, 2023). We retain the original diffusion framework and plain backbone organization and replace the Transformer token mixer with ATBM. Following DiT, we consider S, B, and L models with patch size 2 for class-conditional ImageNet generation at 256×256. The models are trained for 400K optimization steps with a batch size of 256.

Table 5: Results of Class-conditional image generation on ImageNet-1K.

Model	FID↓	Params
LDM-8	15.51	395M
LDM-4	10.56	400M
DiT-S	68.40	32M
DiT-B	43.47	130M
DiT-L	23.33	459M
ATBM-DiT-S	61.24	34M
ATBM-DiT-B	37.86	137M
ATBM-DiT-L	21.16	484M

Results. Table 5 shows that replacing the original DiT token mixer with ATBM improves FID consistently across model scales under the 400K-step setting. ATBM-DiT-S reduces FID from 68.40 to 61.24, ATBM-DiT-B from 43.47 to 37.86, and ATBM-DiT-L from 23.33 to 21.16. The gains therefore persist as model capacity increases rather than being confined to a particular scale. These experiments show that ATBM trans-

Table 6: **ATBM component ablation.** *State*: cross-block state pathway; *Adapt.*: adaptive token-head routing; *Calib.*: per-head path calibration. Fixed fusion uses $\alpha = 0.5$. Top-1 accuracy (%) on ImageNet-1K is reported.

Variant	Local	State	Adapt.	Calib.	Top-1
Local only	✓				75.5
State only		✓			76.1
Fixed fusion	✓	✓			76.5
Adaptive routing	✓	✓	✓		76.9
Full ATBM	✓	✓	✓	✓	**77.2**

Table 7: **Cross-block state mixer ablation.** *Dist.*: distance-dependent; *Head-spec.*: head-specific; *Len.-indep.*: parameter count independent of M. *Exp.*: exponential Toeplitz mixing with R components. Top-1 accuracy (%) is reported.

Mixer	Dist.	Head-spec.	Len.-indep.	Top-1
Uniform			✓	76.5
Shared Toep.	✓		✓	76.9
Free Toep.	✓	✓		76.6
Exp., $R = 4$	✓	✓	✓	77.0
Exp., $R = 8$	✓	✓	✓	**77.2**
Exp., $R = 16$	✓	✓	✓	77.1

fers beyond discriminative recognition and remains effective when used as the token mixer of a diffusion Transformer.

4.6 Ablation Studies

All ablation experiments are conducted on ImageNet-1K with the same backbone, training recipe, and input resolution as the main experiments. For each comparison, we vary only the component under study and keep all other architectural and optimization settings unchanged.

Contribution of Each Component. We evaluate the contributions of the local pathway, the cross-block state pathway, adaptive token-head routing, and per-head path calibration by progressively combining these components. As shown in Table 6, the local-only and state-only variants achieve 75.5% and 76.1% Top-1 accuracy, respectively. Combining the two pathways with fixed fusion ($\alpha = 0.5$) improves accuracy to 76.5%, outperforming either pathway alone and supporting their complementarity. Replacing fixed fusion with adaptive token-head routing further increases accuracy to 76.9%. Finally, per-head path calibration raises accuracy to 77.2%, providing an additional gain of 0.3 percentage points.

Design of the Cross-Block State Mixer. We compare the proposed exponential Toeplitz mixer with uniform cross-block mixing, a Toeplitz profile shared across heads, and a free head-specific Toeplitz parameterization. As shown in Table 7, the shared Toeplitz mixer achieves 76.9% Top-1 accuracy, compared with 76.5% for uniform mixing, supporting the benefit of incorporating relative-distance structure. The free head-specific Toeplitz parameterization achieves 76.6%, indicating that greater parameter flexibility alone does not yield better accuracy in this comparison. The exponential Toeplitz mixer achieves 77.0%, 77.2%, and 77.1% with $R = 4$, 8, and 16, respectively, outperforming all three alternatives while keeping its parameter count independent of the number of blocks M. Among the tested settings, $R = 8$ achieves the highest accuracy, increasing the number of components to 16 provides no further gain.

Granularity of Adaptive Routing. We compare three fusion schemes, with learnable per-head path calibration enabled in all variants. Fixed fusion uses $\alpha = 0.5$, the learned global scalar uses one input-independent coefficient shared across the network, and adaptive token-head fusion predicts a coefficient for each token and head. As shown in Table 8(a), these variants achieve 76.6%, 76.9%, and 77.2% Top-1 accuracy, respectively. Since the global scalar can be absorbed into the path scales, it changes the optimization parameterization rather than the representable fusion functions. Adaptive token-head fusion introduces input-dependent mixing and achieves the highest accuracy.

Cross-Block Normalization. We examine three normalization strategies for cross-block aggregation: unnormalized mixing, row-wise ℓ_1 normalization of the mixing weights, and division of the aggregated state by $M - 1$, where M is the number of blocks. Diagonal entries are excluded in all variants, so each block receives contributions only from the other $M - 1$ blocks. As shown in Table 8 (b), row-wise ℓ_1 normalization improves Top-1 accuracy from 76.8% to 77.1%. Dividing by $M - 1$ achieves the highest accuracy of 77.2%. This result supports scaling the cross-block aggregation by the number of contributing blocks.

Path Calibration. The local and cross-block pathways use different aggregation mechanisms, which can produce outputs with different scales. We evaluate whether explicitly calibrating these

Table 8: **Ablation of routing granularity, cross-block normalization, and path calibration.**

(a) Routing granularity

Routing	Top-1 (%)
Fixed $\alpha = 0.5$	76.6
Learned global scalar	76.9
Adaptive token-head	77.2

(b) Cross-block normalization

Normalization	Top-1 (%)
None	76.8
Row-wise ℓ_1	77.1
Divide by $M - 1$	77.2

(c) Path calibration

Calibration	Top-1 (%)
None	76.9
Shared path scales	77.0
Per-head path scales	77.2

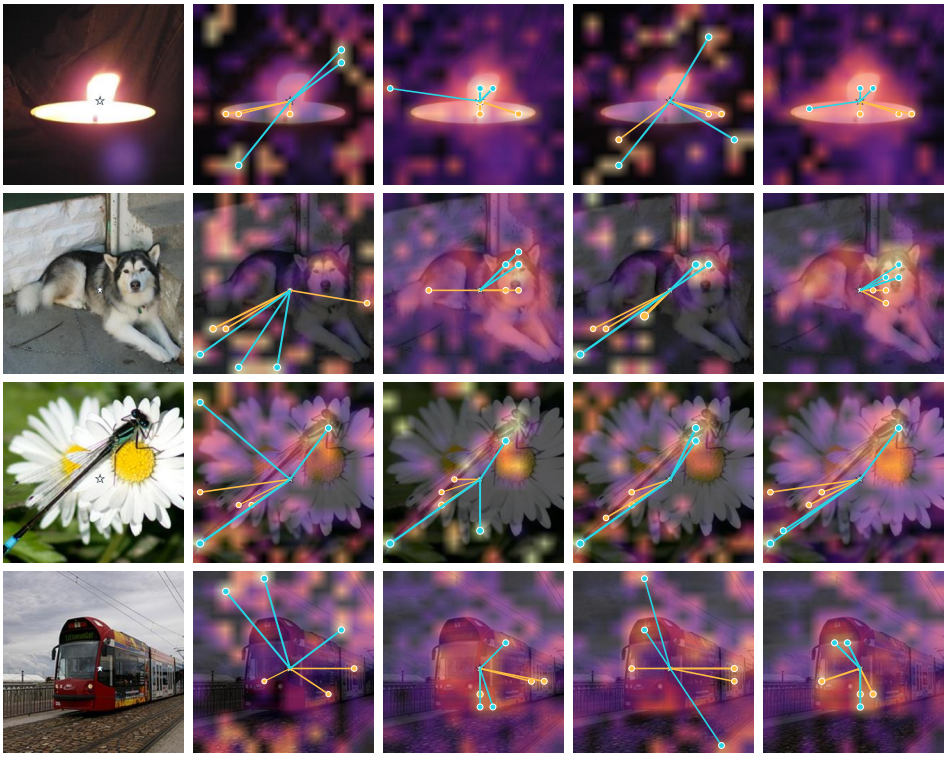


Figure 4: **Qualitative ablation of cross-block normalization and path calibration.** Heatmaps visualize normalized token-mixing weights. Yellow and cyan lines indicate the top-three within-block and cross-block connections, respectively.

outputs improves their fusion. Keeping the adaptive token-head router unchanged, we vary only the pathway scales in Eq. 8. *None* fixes both scales to one; *Shared path scales* learns one pair of scales per layer; and *Per-head path scales* learns a separate pair for each attention head. As shown in Table 8 (c), shared path scales improve Top-1 accuracy from 76.9% to 77.0%, while per-head path scales further increase accuracy to 77.2%. These results support head-specific calibration of the local and cross-block outputs in addition to adaptive routing.

Fig. 4 further provides a qualitative comparison of cross-block normalization and per-head path calibration, applied individually and jointly. In the illustrated examples, combining both components produces connections more concentrated on the depicted objects, with fewer connections extending to distant background regions than in the base model.

5 Conclusion

We introduced Adaptive Token-Block Mixing (ATBM), which combines within-block softmax attention with structured communication over associative block states. Head-specific Toeplitz mixing enables cross-block context propagation, while token-head gates adaptively fuse local and cross-block features. ATBM offers a unified approach to preserving fine-grained token interaction and incorporating broader context without constructing dense global token-to-token attention.

AI USE STATEMENT

In this work, generative AI tools were used to assist with automating experiment execution and debugging software code. The research ideas, proposed method, main model implementation, and ablation-study implementation were developed by the authors without the use of generative AI. Generative AI tools were not used to formulate hypotheses, design the proposed method, design experiments, interpret experimental results, or develop mathematical claims or proofs; other required-disclosure tasks are not applicable to this work.

All AI-assisted code modifications and experiment automation scripts were reviewed by the authors, and their correctness was verified through code inspection and experimental execution. The authors independently verified the experimental results and conclusions. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the implementation details necessary to reproduce our method in the main paper. The supplementary material provides a detailed description of the model architecture and training configurations, including the experimental settings and hyperparameters used in our experiments. These materials are intended to provide the information necessary for reproducing the reported experimental results.

REFERENCES

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. EfficientViT: Lightweight multi-scale attention for high-resolution dense prediction. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 17302–17313, 2023.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(5): 1483–1498, 2019.

Chun-Fu Chen, Rameswar Panda, and Quanfu Fan. RegionViT: Regional-to-local attention for vision transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9355–9366, 2021.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop*, 2020.

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

Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Qihang Fan, Huaibo Huang, Mingrui Chen, Hongmin Liu, and Ran He. RMT: Retentive networks meet vision transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5641–5651, 2024.

Qihang Fan, Huaibo Huang, Yuang Ai, and Ran He. Rectifying magnitude neglect in linear attention. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 21505–21514, 2025.

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. FLatten Transformer: Vision transformer using focused linear attention. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 5961–5971, 2023.

Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *European Conference on Computer Vision (ECCV)*, 2024b.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6185–6194, 2023.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. In *International Conference on Machine Learning (ICML)*, volume 162, pp. 9099–9117, 2022.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020.

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision (ECCV)*, pp. 280–296. Springer, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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 *IEEE International Conference on Computer Vision (ICCV)*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yuxin Mao, Zhen Qin, Jinxing Zhou, Bin Fan, Jing Zhang, Yiran Zhong, and Yuchao Dai. Learning spatial decay for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 7945–7953, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Zhen Qin, Xiaodong Han, Weixuan Sun, Bowen He, Dong Li, Dongxu Li, Yuchao Dai, Lingpeng Kong, and Yiran Zhong. Toeplitz neural network for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhen Qin, Yuxin Mao, Xuyang Shen, Dong Li, Jing Zhang, Yuchao Dai, and Yiran Zhong. You only scan once: Efficient multi-dimension sequential modeling with lightnet. *arXiv preprint arXiv:2405.21022*, 2024a.

Zhen Qin, Songlin Yang, and Yiran Zhong. Hierarchically gated recurrent neural network for sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024b.

Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. Vicinity vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 45(10):12635–12649, 2023.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 568–578, 2021.

Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. CrossFormer: A versatile vision transformer hinging on cross-scale attention. In *International Conference on Learning Representations (ICLR)*, 2022.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision (ECCV)*, pp. 432–448, 2018.

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 30008–30022, 2021.

Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Kewei Zhang, Ye Huang, Yufan Deng, Jincheng Yu, Junsong Chen, Huan Ling, Enze Xie, and Daquan Zhou. MHLA: Restoring expressivity of linear attention via token-level multi-head. In *International Conference on Learning Representations (ICLR)*, 2026.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 13001–13008, 2020.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5122–5130. IEEE, 2017.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. In *International Conference on Machine Learning (ICML)*, pp. 62429–62442, 2024.

A Additional Method Details

A.1 Backbone Configurations

Table 9 summarizes the backbone configurations. The plain ATBM models use a patch size of 16×16 and maintain a constant token resolution across 12 blocks. Each block is followed by a pre-normalized ATBM module and a feed-forward network. Classification uses global average pooling after the final LayerNorm, without a class token.

The hierarchical H-ATBM models use a convolutional stem with an overall stride of 4, followed by four stages with output strides of 4, 8, 16, and 32. Convolutional downsampling is applied between stages.

Table 9: Backbone configurations. Lists specify values for successive stages. The MLP ratio is the hidden-width expansion factor. The actual hidden dimension is rounded down to an integer.

Model	Output strides	Depths	Embedding dimensions	Heads	MLP ratio
ATBM-T	16	12	192	6	4.0
ATBM-S	16	12	384	6	4.0
ATBM-B	16	12	768	12	3.6
H-ATBM-T	[4,8,16,32]	[1,3,9,4]	[64,128,320,512]	[2,4,10,16]	4.0
H-ATBM-S	[4,8,16,32]	[2,6,18,8]	[64,128,320,512]	[2,4,10,16]	4.0
H-ATBM-B	[4,8,16,32]	[2,6,18,8]	[96,192,448,640]	[3,6,14,20]	3.8

A.2 Mixing Module and Initialization

The initial half-lives of the exponential components are logarithmically spaced between 1 and 64 blocks. For a component with half-life τ_r, the initial decay rate is $\lambda_r = \log(2)/\tau_r$. We initialize the unconstrained decay parameters using inverse softplus. Raw amplitude parameters and kernel biases are initialized to zero, giving initial amplitudes of $\log 2$. Heads share the same initialization but learn independent kernel parameters. Per-head calibration log-scales are initialized to zero, corresponding to unit initial scales. The backbone implementations use truncated-normal initialization with standard deviation 0.02 for linear weights and zero initialization for linear biases, with LayerNorm scales and biases initialized to one and zero, respectively, under the default initialization setting.

B Additional Experimental Settings

B.1 Image Classification

We train the ATBM and H-ATBM classification models from scratch on ImageNet-1K for 300 epochs at an input resolution of 224×224. We use AdamW with a global batch size of 1,024 and a peak learning rate of 10^{-3}. The learning rate follows a cosine schedule with a 20-epoch linear warm-up. Training augmentations include RandAugment, Mixup, CutMix, and Random Erasing. Top-1 accuracy is evaluated on the ImageNet-1K validation set. The main classification settings are summarized in Table 10.

Table 10: ImageNet-1K classification settings for ATBM and H-ATBM.

Setting	Value
Input resolution	224×224
Training epochs	300
Global batch size	1,024
Optimizer	AdamW
Peak learning rate	10^{-3}
Learning-rate schedule	Cosine decay
Warm-up	20 epochs, linear
Augmentations	RandAugment, Mixup, CutMix, Random Erasing
Evaluation metric	Top-1 accuracy

B.2 Detection and Segmentation Settings

Object detection and instance segmentation. We use the ImageNet-pretrained plain backbone with a SimpleFPN and Cascade Mask R-CNN on COCO 2017. SimpleFPN produces five feature levels with 256 channels. Training uses large-scale jittering with a scale range of $[0.1, 2.0]$, random horizontal flipping, and cropping/padding to 1024×1024. We use AdamW with a learning rate of 10^{-4}, weight decay of 0.1, and layer-wise learning-rate decay of 0.7. The training schedule comprises 184,375 iterations, with 250 iterations of linear warm-up and learning-rate reductions by a factor of 0.1 at iterations 163,889 and 177,546. Evaluation resizes images while preserving the aspect ratio and pads them to 1024×1024.

Semantic segmentation. We use UPerNet on ADE20K with features extracted from blocks 3, 6, 9, and 12 of the ImageNet-pretrained plain backbone. A MultiLevelNeck rescales these features by factors of $[4, 2, 1, 0.5]$. The UPerNet decoder uses 256 channels and pooling scales of $[1, 2, 3, 6]$, with an auxiliary 256-channel FCN head weighted by 0.4. Training uses 512×512 random crops, random resizing with a ratio range of $[0.5, 2.0]$, horizontal flipping, and photometric distortion. We train for 160,000 iterations using AdamW, a learning rate of 6×10^{-5}, and weight decay of 0.01. A 1,500-iteration linear warm-up is followed by polynomial learning-rate decay with power 1.0. Normalization parameters are exempt from weight decay. The configured evaluation uses single-scale, whole-image inference with aspect-ratio-preserving resizing to fit within 2048×512.

Table 11: Downstream training configurations. Batch sizes are global across eight GPUs. Drop-path rates are listed for Tiny/Small backbones.

Setting	COCO	ADE20K
Framework	Cascade Mask R-CNN	UPerNet
Training crop	1024×1024	512×512
Training iterations	184,375	160,000
Batch size	32	16
Optimizer	AdamW	AdamW
Adam betas	$(0.9, 0.999)$	$(0.9, 0.999)$
Learning rate	10^{-4}	6×10^{-5}
Weight decay	0.1	0.01
Warm-up iterations	250	1,500
LR schedule	Multi-step	Polynomial, power 1.0
Layer-wise LR decay	0.7	Not used
Drop-path rate (T/S)	0.0 / 0.1	0.0 / 0.2
Mixed precision	Enabled	Enabled

Backbone adaptation. For both tasks, the classification head and global pooling operation are removed. Detection uses the final stride-16 feature map, while segmentation uses four intermediate feature maps before neck rescaling. The downstream configurations enable token-head routing, per-head path calibration, and the local positional branch, with Q/K normalization disabled. The Small backbone uses a maximum drop-path rate of 0.1 for detection and 0.2 for segmentation; the Tiny backbone uses 0.0 for both.

B.3 Class-Conditional Image Generation

We evaluate ATBM on class-conditional ImageNet-1K generation at 256×256 resolution using the official DiT implementation. We consider Small, Base, and Large models with a latent patch size of 2. ATBM-DiT replaces the self-attention token mixer with ATBM while retaining the DiT conditioning mechanism, diffusion objective, and training pipeline. All models are trained for 400,000 optimization steps with a global batch size of 256.

Training configuration. Following DiT, we use AdamW with a constant learning rate of 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and zero weight decay. Training images are center-cropped to the target resolution and augmented with random horizontal flipping. A frozen pretrained VAE maps images to latent representations with a spatial downsampling factor of 8. We retain the original diffusion

configuration, with 1,000 training timesteps and a linear noise schedule. An exponential moving average (EMA) of the model parameters is maintained with a decay rate of 0.9999.

Sampling and evaluation. We evaluate the EMA checkpoints at 400,000 training steps. For each model, we generate 50,000 images with class labels sampled uniformly from the 1,000 ImageNet classes, using the ancestral DDPM sampler in the official DiT sampling code with 250 sampling steps without classifier-free guidance. We compute Fréchet Inception Distance (FID) using corresponding ImageNet 256×256 reference statistics. The DiT and ATBM-DiT comparisons use identical sampling, guidance, VAE, and FID evaluation settings.

B.4 Software and Hardware

All experiments use eight NVIDIA GeForce RTX 3090 GPUs, PyTorch 2.1.0 with CUDA runtime 12.1, and MMEngine 0.10.7. Detection and segmentation are implemented in MMDetection and MMSegmentation, respectively, with automatic mixed precision.