OCSHELL-MAMBA: OBJECT-CENTRIC SHELL PROPAGATION FOR VISION STATE SPACE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision state space models (SSMs) achieve efficient visual modeling by serializing 2D feature maps into 1D sequences. However, serialization is not representation-invariant: the scan order determines how information accumulates in the recurrent state. Conventional scan paths, such as raster and bidirectional scans, often disrupt local neighborhoods and fragment object structures, while existing adaptive scans remain token-centric and do not explicitly preserve object-level coherence. We propose OCShell-Mamba, an object-centric vision SSM that organizes token-level state propagation into contiguous segments corresponding to object-like regions. Without requiring additional object-level annotations for routing, OCShell-Mamba uses an internal objectness field to construct shell-wise sequences that expand from each candidate center toward its boundary. Its core Shell-wise Object-Centric rOuting (SOCO) module partitions each candidate into concentric shells and orders its tokens shell by shell from the center outward for SSM propagation. This design better aligns sequence adjacency with image geometry while preserving local and object-level coherence. We further introduce Persistent topology Memory (PoMe), a fixed-size memory bank that maintains object-level representations across resolution changes. On ImageNet-1K, ADE20K, and COCO, OCShell-Mamba consistently outperforms a matched DAMamba baseline, achieving 83.9% top-1 accuracy, 50.8% mIoU, 49.1% box AP, and 43.7% mask AP. These results show that vision SSMs benefit not only from adaptive scan paths but also from explicitly modeling object-centric state propagation.

1 INTRODUCTION

Recurrent State Space Models (SSMs) have emerged as competitive alternatives to attention-based architectures, achieving strong visual performance with linear-time sequence modeling (Gu et al., 2021; Gu & Dao, 2024). However, applying SSMs to images requires a 2D feature map to be flattened into a 1D sequence. Since images have no natural 1D order, the scan path implicitly defines a sequential neighborhood by determining which spatial tokens become adjacent during state propagation. An inappropriate ordering may separate related object tokens, mix them with background regions, and require the recurrent state to repeatedly reconstruct object-level evidence.

Scan order is therefore more than an implementation detail. Row-wise, column-wise, and serpentine traversals produce different token adjacencies and directly affect how visual evidence is accumulated. VMamba (Liu et al., 2024) introduces a 2D selective scan to better align state propagation with image structure, while recent studies show that patch ordering strongly affects Mamba-based vision models (Hardan et al., 2025). Hence, scan design is a core component of visual state propagation rather than merely a data-layout choice.

Existing scan strategies address this issue from different perspectives, but typically define traversal at the image or feature-map level. Fixed scans, such as raster and bidirectional traversal, are efficient but tied to the image grid, introducing abrupt transitions and potentially mixing object tokens with background regions. Structured geometric scans, including fractal (Xiao et al., 2025), Hilbert-style (Jeong & Rhee, 2025), or ring-shaped traversal (Hsieh et al., 2026), improve locality or robustness, but still apply largely content-independent ordering patterns over the feature map.

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.

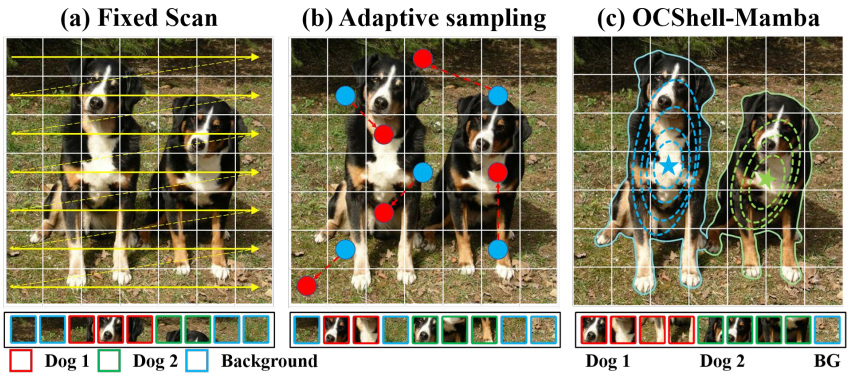


Figure 1: **Why object-centric shell-like scanning ?** Vision SSMs propagate information along a serialized sequence, making scan order a structural prior for visual state propagation. **(a)** Fixed scans may interleave object and background tokens, disrupting object continuity. **(b)** Adaptive scans improve flexibility, but still do not ensure object-contiguous propagation. **(c)** OCShell-Mamba assigns each object candidate a local center and propagates tokens outward in a shell-like manner, forming compact object-wise sequence segments while better preserving spatial structure.

Recent methods improve flexibility through adaptive or deformable scanning. DAMamba (Li et al., 2025b) adapts scan paths to image content, DefMamba (Liu et al., 2025b) learns deformable traversal locations, and InfoScan (Wu et al., 2026) prioritizes informative regions. However, these methods mainly adapt token-level traversal without explicitly constructing a separate propagation sequence for each object. Samba (He et al., 2025) uses saliency-guided neighboring scans to preserve foreground continuity, but focuses on salient regions without explicitly treating individual object candidates as separate propagation units. These scan paths may therefore move between objects and background regions, often disrupting local adjacency and fragment object structures, thus limiting the efficacy of short-range recurrence.

Object-centric organization identifies object-like regions without determining how tokens should be scanned within each object. Conventional fixed scans often introduce abrupt row transitions, separate related tokens, and interrupt object-level propagation. To address this issue, we propose **OCShell-Mamba** with a **Shell-wise Object-Centric rOuting (SOCO)** module. An internal objectness field is first thresholded and grouped by connected components to form object candidates, and an objectness-weighted center is computed for each candidate. We then partition the tokens into concentric shells according to their normalized distance from the center and traverse them from the innermost to the outermost shell, analogous to peeling an onion from the inside out (see Fig. 1). By creating an object-specific center-to-boundary sequence, the SOCO module reduces raster-induced discontinuities and forms compact object-wise segments. Since object candidates are reconstructed at each deep stage and may become unstable as downsampling changes object geometry and resolution, we introduce **Persistent Topology Memory (PoMe)** to maintain and update fixed-size object representations across stages, allowing later stages to reuse previously accumulated object evidence.

We evaluate OCShell-Mamba on three benchmarks spanning image classification, semantic segmentation, and object detection and instance segmentation. Under matched model-scale and training settings, it outperforms DAMamba on every task, achieving 83.9% top-1 accuracy on ImageNet-1K, 50.8% mIoU on ADE20K, and 49.1% box AP and 43.7% mask AP on COCO. These gains over a baseline that already uses adaptive scanning suggest that the benefits extend across task types. Routing visualizations provide qualitative evidence that the learned objectness field responds to visually subtle instances, despite receiving only task-level supervision. We summarize our contributions as follows:

- We formulate an object-centric perspective on propagation in vision state space models, motivating recurrent processing over candidate object regions rather than isolated tokens.
- We propose **Shell-wise Object-Centric Routing (SOCO)**, a center-to-boundary scanning strategy that groups tokens into concentric shells to form compact, spatially coherent sequences for object-level state propagation.

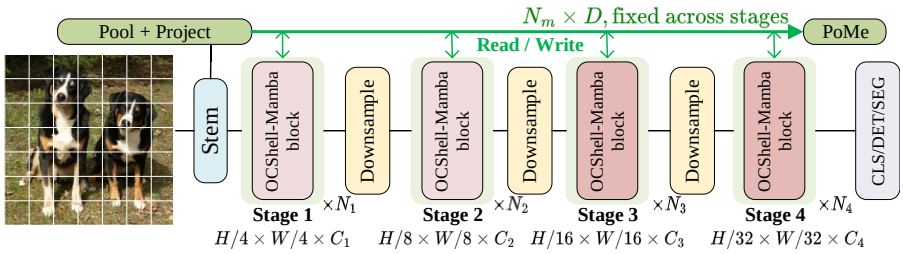


Figure 2: **OCShell-Mamba architecture:** The spatial stream adopts a standard 4-stage hierarchical backbone with a task-specific head for classification, detection, or segmentation. Persistent topology memory (PoMe) is initialized from stem features via a pool-and-project module (G^0), maintains a fixed $N_m \times D$ representation and updating once per stage (green dashed arrows). Early stages (1–2) use confidence-based routing in the shallow regime, while later stages (3–4) employ shell-wise object-centric routing where semantic responses become more reliable. Block numbers N_i and channel dimensions C_i follow standard hierarchical designs.

- We develop **OCShell-Mamba**, which combines SOCO with **PoMe** to aggregate object evidence within each stage and carry it across changes in feature resolution.
- We reach the state-of-the-art performance across image classification, semantic segmentation, object detection, and instance segmentation. OCShell-Mamba outperforms strong adaptive-scan baselines on ImageNet-1K, ADE20K, and COCO under matched training settings.

2 Related Work

Vision State Space Models. State space models (SSMs), especially Mamba (Gu & Dao, 2023), have gained attention as linear-complexity sequence models. Vision Mamba typically serializes 2D features into 1D tokens and propagates states along this order. Vim (Zhu et al., 2024) adds bidirectional modeling, while VMamba (Liu et al., 2024) introduces 2D Selective Scan (SS2D) to link 1D propagation with 2D traversal. Later Mamba backbones extend to recognition, detection, segmentation, restoration, medical analysis, and other dense tasks (Gu et al., 2026; Jiang et al., 2025; Liu et al., 2025a; Ma et al., 2024; Xing et al., 2026; Zhou et al., 2026). Despite task differences, most hierarchical vision SSMs serialize tokens per stage and pass cross-stage information mainly via downsampled features, making token order and cross-stage evidence preservation key to object-level topology modeling.

Scan Strategies in Vision SSMs. Scan design defines the sequence neighborhood for recurrent state accumulation. Fixed scans give spatial priors: VMamba (Liu et al., 2024) uses multi-directional cross scans, PlainMamba (Yang et al., 2024) continuous scans, LocalMamba (Huang et al., 2024) windowed selective scan, and FractalMamba (Xiao et al., 2025) fractal curves for spatial dependencies. Patch order also affects Mamba performance (Hardan et al., 2025), confirming scan order matters. However, fixed/geometry-driven scans impose preset traversal priors: they improve locality, coverage, or dependencies but do not adapt to object candidates, so same-object tokens may remain separated, especially for off-center, thin, elongated, or cluttered objects. Recent methods increase adaptivity: DAMamba (Li et al., 2025b) assigns scan order/regions by content, DefMamba (Liu et al., 2025b) uses deformable paths, and InfoScan (Wu et al., 2026) allocates compute to informative regions. These improve sampling, adaptivity, or allocation. Our work is complementary: instead of where/how much to sample, we organize foreground object candidates as contiguous units for state propagation.

Object-Aware and Foreground-Guided Routing. Object-level cues are studied in self-supervised learning and unsupervised object discovery. DINO (Caron et al., 2021) shows emergent object localization in ViTs; LOST (Siméoni et al., 2021), TokenCut (Wang et al., 2023), and FOUND (Siméoni et al., 2023) extract seeds, graph partitions, or background complements from features, showing object/foreground cues can emerge without labels. However, they target localization/segmentation, not organizing recurrent state propagation in vision SSMs. Foreground-aware scanning is closer to

our setting: Samba (He et al., 2025) uses saliency-guided Mamba with spatial-neighbor scans to preserve salient continuity, showing the value of foreground-aware traversal. But Samba is task-specific and relies on saliency. In contrast, OCShell-Mamba is a backbone-level routing method: it derives an internal objectness field and uses shell-wise object-centric routing to group foreground candidates as contiguous propagation units, without external masks or saliency labels.

Cross-Stage Information Flow and Persistent Tokens. Hierarchical backbones reduce spatial resolution stage by stage to build efficient semantic representations, so evidence must pass through compressed spatial streams to later stages. Skip connections (U-Net (Ronneberger et al., 2015)), FPN (Lin et al., 2017), and residual/multi-scale designs (Liu et al., 2022) reuse spatial features across scales but do not preserve topology accumulated by SSM state propagation as persistent cross-stage representations. Compact token carriers exist in attention models: ViTs use class tokens (Dosovitskiy et al., 2020), Set Transformer uses inducing points (Lee et al., 2019), and Perceiver uses latent arrays (Jaegle et al., 2021), showing compact tokens can carry global information. However, they do not preserve object-contiguous evidence from SSM scan propagation across stages. OCShell-Mamba instead introduces the Persistent Topology Memory, a fixed-size token bank carrying object-level evidence across stages and refining it with the spatial stream.

3 METHODOLOGY

We present **OCShell-Mamba**, an object-centric vision SSM that organizes state propagation around object candidates rather than isolated tokens. Unlike standard vision SSMs, where tokens from the same object may be separated by the scan order, OCShell-Mamba explicitly forms object candidates and enforces object-centered propagation within each candidate. Given a stage feature map, OCShell-Mamba predicts an internal objectness field to group high-objectness regions into object candidates with local centers. As shown in Fig. 2, the model consists of a hierarchical spatial stream and a parallel memory highway that exchanges information once per stage.

We first introduce **Shell-wise Object-Centric rOuting (SOCO)**, which traverses each object candidate from its center to its boundary to preserve object-level continuity. We then present **Routed State Propagation**, a hierarchy-aware strategy that applies confidence-based routing in shallow stages and shell routing in deeper stages. Finally, we introduce **Persistent Topology Memory (PoMe)**, a fixed-size memory bank that preserves object-level representations across stages despite changes in spatial resolution. Overall, OCShell-Mamba reorganizes state propagation and augments it with persistent memory while preserving the original token-to-location correspondence after each SSM block.

3.1 SHELL-WISE OBJECT-CENTRIC ROUTING (SOCO)

Fig. 3 illustrates the SOCO pipeline. Given a 2D feature map, SOCO predicts an objectness field, groups connected responses into object candidates, and estimates a local center for each candidate. It then orders tokens within each candidate from the center toward the boundary and concatenates the resulting sequences for SSM processing. SOCO changes the order of state propagation without changing the tokens' spatial identities: after the SSM block, the outputs are mapped back to their original positions in the 2D feature map.

Objectness field. Let $F^s \in \mathbb{R}^{H_s \times W_s \times C_s}$ be the feature map at stage s, flattened into $L_s = H_s W_s$ tokens: $\mathcal{T}^s = \{\mathcal{T}_i^s\}_{i=1}^{L_s}, \qquad \mathcal{T}_i^s \in \mathbb{R}^{C_s}$, where $\mathcal{T}_i^s$ is a token with the spatial coordinates $p_i^s \in \mathbb{R}^2$. A lightweight head Φ_s predicts an internal objectness field:

$$R_i^s = \sigma\big(\Phi_s(\mathcal{T}_i^s)\big), \qquad R^s \in [0,1]^{L_s}, \tag{1}$$

where σ is the sigmoid function, and higher values mean higher likelihood of belonging to a foreground object candidate. The detailed architecture of Φ_s is provided in the Supplement. R_i^s is learned end-to-end from the task objective. Since the scan route is constructed from R_i^s via discrete operations (thresholding, connected-component grouping, binning, sorting), we construct routing using a stop-gradient (SG) copy:

$$\bar{R}_i^s = \mathcal{SG}(R_i^s), \tag{2}$$

where $\mathcal{SG}(\cdot)$ blocks gradient flow in the backward pass. Therefore, Φ_s receives no direct gradient via the routing path; it is updated only via the task-driven gradients flowing into its input $\mathcal{T}^s$, so routing patterns that improve task accuracy are implicitly reinforced from task supervision alone.

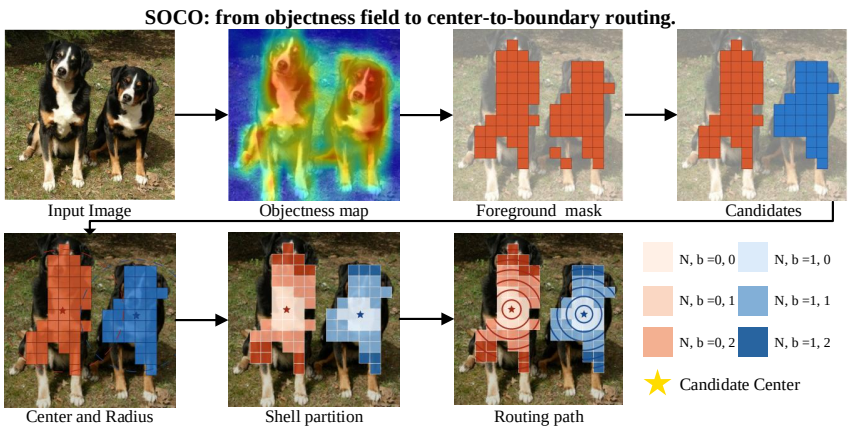


Figure 3: **SOCO pipeline.** At stage s, the objectness predictor Φ_s generates an objectness map $R^s \in [0,1]$. Thresholding with τ_s and connected components extract object candidates $\{\Omega_n^s\}$ without mask supervision. Each candidate is represented by an objectness-weighted center c_n^s (star) and radius r_n^s (dashed circle), and its tokens are grouped into B_s^n concentric shells according to normalized distance from the center. Routing sequence π_{shell}^s traverses tokens from the inner to the outer shells, with candidates processed in descending confidence order. The objectness map R^s is learned end-to-end from the task loss.

Object candidate formation. Foreground-like tokens are grouped into connected components:

$$\{\Omega_1^s, \ldots, \Omega_{N_s}^s\} = \mathrm{CC}_{m_s}\big(\{\mathcal{T}_i^s \mid \bar{R}_i^s > \tau_s\}\big), \tag{3}$$

where τ_s denotes the stage-dependent threshold computed from $\bar{R}^s$ using Otsu's thresholding method Otsu (1979), $\mathrm{CC}\,m_s(\cdot)$ removes connected components smaller than m_s, and N_s is the number of retained candidates. Since each component should contain sufficient tokens, m_s is set to two for all stages. The retained components are treated as foreground candidates, whereas smaller components are reassigned to the background to suppress noisy responses. For each retained candidate Ω_n^s, we compute its center location c_n^s and confidence score a_n^s as

$$c_n^s = \frac{\sum_{\mathcal{T}_i \in \Omega_n^s} \bar{R}_i^s \, p_i^s}{\sum_{\mathcal{T}_i \in \Omega_n^s} \bar{R}_i^s + \epsilon_c}, \qquad a_n^s = \frac{1}{|\Omega_n^s|} \sum_{\mathcal{T}_i \in \Omega_n^s} \bar{R}_i^s, \tag{4}$$

where ϵ_c is a small numerical stability constant and a_n^s is used to rank candidates in the routing order.

Center-to-boundary shell ordering. For each Ω_n^s, we define the normalized distance $d_{n,i}^s$ from the token $\mathcal{T}_i \in \Omega_n^s$ to its center as:

$$d_{n,i}^s = \frac{\|p_i^s - c_n^s\|_2}{r_n^s + \epsilon_r}, \qquad r_n^s = \max_{\mathcal{T}_i \in \Omega_n^s} \|p_i^s - c_n^s\|_2, \tag{5}$$

where ϵ_r is a small stability constant. Let B_s^n denote the number of shells at the stage s for the nth retained candidate, where $B_s^n = \lfloor r_n^s / \tau_t \rfloor$ and τ_t is the radius of a vision token. Based on $d_{n,i}^s$, each token $\mathcal{T}_i$ is assigned to the bth shell with the equation:

$$b = \lfloor B_s^n \, d_{n,i}^s \rfloor, \tag{6}$$

where $\lfloor v \rfloor$ rounds v down to an integer, partitioning the normalized distances into B_s^n equal-width shells indexed from the center outward: index 0 corresponds to the innermost region around the center, and index $B_s^n - 1$ to the outermost band along the candidate boundary. Let $\mathcal{Q}_{n,b}^s$ denote the set of tokens in the b-th shell within an object Ω_n^s. Let ρ^s denote a specific spatial ordering on the feature grid, and $\mathrm{Ord}_{\rho^s}(\mathcal{A})$ the ordered indices of a set $\mathcal{A}$ under ρ^s. The within-candidate sequence is constructed by traversing shells from center to boundary:

$$\mathrm{Shell}(\Omega_n^s) = \Big[\mathrm{Ord}_{\rho^s}(\mathcal{Q}_{n,b}^s)\Big]_{b=0}^{B_s^n - 1}. \tag{7}$$

Here, $[\cdot]_{b=0}^{B_s^n-1}$ denotes ordered concatenation in ascending shell index b. The final routing order concatenates candidates sorted by a descending order with the confidence a_n^s, followed by background tokens:

$$\pi_{\text{shell}}^s = \left[\text{Shell}(\Omega_{(1)}^s), \ldots, \text{Shell}(\Omega_{(N_s)}^s), \text{Ord}_{\rho^s}(\Omega_{\text{bg}}^s)\right], \tag{8}$$

where $\Omega_{\text{bg}}^s = \{1, \ldots, L_s\} \setminus \bigcup_n \Omega_n^s$ denotes all remaining background tokens. If no candidates are selected, the original spatial ordering $\text{Ord}_{\rho^s}(\{1, \ldots, L_s\})$ is used. This ensures that each candidate forms a compact sequence segment, improving the modeling of off-center, elongated, and boundary-heavy objects.

3.2 Routed State Propagation

Fig. 2 illustrates why routing should depend on network depth. In shallow stages (light coral), features primarily capture local edges and textures, so objectness estimates may not support stable grouping into object candidates. In deeper stages (deep coral), features are more semantic and spatially compressed, making candidate grouping more reliable. We therefore use confidence-based routing in shallow stages and shell-wise object-centric routing in deep stages.

Hierarchy-aware routing. In shallow stages, we avoid forming explicit object candidates. Instead, we group tokens by objectness confidence. Let $\mathcal{B}_b^s$ denote the set of tokens in the b-th shell extracted from $\mathcal{T}^s$. For the bth shell, $\mathcal{B}_b^s$ is obtained as follows:

$$\mathcal{B}_b^s = \left\{\mathcal{T}_i \;\middle|\; \min\left(\left\lfloor B_s \bar{R}_i^s \right\rfloor, B_s - 1\right) = b\right\}, \quad b = 0, \ldots, B_s - 1, \tag{9}$$

where $\bar{R}_i^s$ is defined in Eq. (2). We visit bins in descending confidence order while retaining the order specified by ρ^s within each bin:

$$\pi_{\text{conf}}^s = \left[\text{Ord}_{\rho^s}(\mathcal{B}_{B_s-1}^s), \ldots, \text{Ord}_{\rho^s}(\mathcal{B}_0^s)\right]. \tag{10}$$

The stage-dependent routing policy is

$$\pi^s = \begin{cases} \pi_{\text{conf}}^s, & s \in \mathcal{S}_{\text{shallow}}, \\ \pi_{\text{shell}}^s, & s \in \mathcal{S}_{\text{deep}}. \end{cases} \tag{11}$$

The stage partition is specified in Appendix.

Permutation and SSM propagation. Given π^s, we reorder tokens before SSM propagation and restore their original spatial positions afterward:

$$Y^s = \mathcal{P}_{\pi^s}^{-1}\left(\text{SSM}_s\left(\mathcal{P}_{\pi^s}(\widetilde{X}^s)\right)\right), \tag{12}$$

where $\mathcal{P}_{\pi^s}$ is the permutation operator and $\widetilde{X}^s$ is the token representation after the PoMe memory read; without PoMe, $\widetilde{X}^s = X^s$. For a fixed π^s, permutation and its inverse preserve gradient flow through the token features and SSM. Gradients do not pass through the discrete construction of π^s from $\bar{R}^s$. Thus, routing changes the SSM propagation order while preserving the spatial alignment of its outputs.

3.3 Persistent Topology Memory

SOCO forms object-contiguous evidence within each stage, but hierarchical backbones repeatedly change spatial resolution, so this evidence will be weakened after downsampling. **Persistent Topology Memory** (PoMe) preserves it through a fixed-size token bank maintained across stages. Let $G^s \in \mathbb{R}^{N_m \times D}$ denote the memory after stage s, where N_m and D are the number and dimension of memory tokens, respectively. Unlike spatial features, the memory retains the same shape throughout the backbone and is not itself spatially downsampled. We initialize it from the stem features:

$$G^0 = \mathcal{I}(X^0), \tag{13}$$

where $\mathcal{I}$ consists of pooling and projection. Before routing at stage s, spatial tokens read from the preceding memory:

$$\widetilde{X}^s = X^s + \mathcal{R}_s(X^s, G^{s-1}). \tag{14}$$

After routed SSM propagation produces Y^s, the memory is updated by

$$G^s = \mathrm{LN}\big(G^{s-1} + \mathcal{W}_s(G^{s-1}, Y^s)\big)\,. \tag{15}$$

Here, $\mathcal{R}_s$ and $\mathcal{W}_s$ are lightweight cross-attention modules with outputs in $\mathbb{R}^{L_s \times C_s}$ and $\mathbb{R}^{N_m \times D}$, respectively. During reading, spatial tokens query memory tokens; during writing, memory tokens query the routed spatial outputs. Stage-specific projections accommodate the changing spatial channel dimension C_s while keeping the memory dimension D fixed.

SOCO organizes evidence within each stage, and PoMe carries information from routed outputs to subsequent stages. OCShell-Mamba is trained end-to-end without external mask or instance annotations. The complete training objective, including the auxiliary loss applied to memory, is provided in the Supplementary Material.

4 EXPERIMENTS

We evaluate **OCShell-Mamba** on three standard benchmarks: ImageNet-1K (Russakovsky et al., 2015) for image classification, COCO (Lin et al., 2014) for object detection and instance segmentation, and ADE20K (Zhou et al., 2017) for semantic segmentation. Following standard backbone evaluation (Liu et al., 2024; Li et al., 2025b), ImageNet-1K measures image-level representation learning, while COCO and ADE20K assess transferability to dense prediction tasks. Unless noted otherwise, we follow each baseline's training and evaluation protocol, replacing only the backbone with OCShell-Mamba for fair comparison. We report model parameters, FLOPs, and task-specific metrics to assess both accuracy and efficiency. The analysis of time complexity of **OCShell-Mamba** is detailed in the Supplement.

Image Classification on ImageNet-1K: We evaluate OCShell-Mamba on ImageNet-1K (Russakovsky et al., 2015), which contains 1.28M training images and 50K validation images across 1,000 categories. To isolate the effects of the proposed object-centric routing and persistent topology memory, we follow the training recipe of DAMamba (Li et al., 2025b) unless otherwise specified, ensuring that performance differences primarily arise from architectural changes rather than training protocols. OCShell-Mambais trained for 300 epochs using AdamW (Loshchilov & Hutter, 2017) with a cosine learning rate schedule and 20-epoch linear warm-up. All experiments use an input resolution of 224×224 and adopt the same augmentation and regularization settings as the recipe, including RandAugment, repeated augmentation, Mixup, CutMix, random erasing, label smoothing, weight decay, and stochastic depth. During evaluation, images are center-cropped to 224×224, and top-1 accuracy is reported. All experiments are conducted on 6 NVIDIA RTX 3090 GPUs. We additionally evaluate exponential moving average (EMA), but it does not provide consistent improvements; therefore, all main results are reported without EMA.

Object-centric propagation under a saturated scan benchmark. Table 1 compares OCShell-Mamba with recent vision Mamba models that explore diverse scanning strategies, including local, geometric, sparse, deformable, and adaptive scanning. OCShell-Mamba achieves 83.9% top-1 accuracy at 224×224 resolution, outperforming all compared the same scale and similarly lightweight vision Mamba models.

The most relevant comparison is with DAMamba, as both methods are evaluated under the same scale training recipe. While DAMamba improves scan flexibility by adapting the traversal path, OCShell-Mamba instead redefines the propagation unit itself: object-like regions are organized into compact shell-wise sequences and preserved across stages through persistent topology memory. The gain over DAMamba, from 83.8% to 83.9%, is achieved in an already strong and highly saturated setting where improvements from scan design alone are increasingly limited. This result suggests that object-centric propagation is not merely another scanning variant, but a complementary mechanism for structuring recurrent state aggregation in vision SSMs.

OCShell-Mamba is not designed for efficiency, and its 5.7G FLOPs are higher than efficiency-oriented methods such as InfoScan and EfficientVMamba. The goal of this comparison is therefore not to optimize the efficiency–accuracy trade-off, but to evaluate whether object-contiguous propagation improves representation quality under controlled settings. Overall, the results support our main finding: even for image classification, vision SSMs benefit when state propagation preserves object-level structure rather than relying only on more flexible grid-based traversal.

Table 1: Comparison on ImageNet-1K classification. The symbol * marks the result reported in the original DAMamba paper; our reproduction obtained 83.6% Top-1 accuracy.

Model	Params.	FLOPs	Top-1
Vim (2024)	26M	5.3G	80.5%
VMamba (2024)	30M	5.6G	82.6%
SiMBA (2024)	15M	5.0G	81.7%
Zigma (2024)	31M	5.1G	82.4%
QuadMamba (2024)	31M	5.5G	81.4%
LocalMamba (2024)	26M	5.7G	82.7%
FractalMamba (2025)	31M	4.8G	83.0%
Adventurer (2025)	12M	4.2G	78.2%
SparX-Mamba (2025)	27M	5.2G	83.5%
EfficientVMamba (2025)	33M	4.0G	81.8%
PlainMamba (2024)	25M	8.1G	81.6%
GroupMamba (2025)	23M	7.0G	83.3%
VSSD (2025)	14M	4.5G	82.5%
DefMamba (2025b)	26M	4.8G	83.5%
MaIR (2025a)	26M	5.4G	83.1%
DAMamba (2025b)	26M	4.8G	83.8%*
INFOSCAN (2026)	10M	2.5G	83.4%
OCShell-Mamba (ours)	27M	5.7G	**83.9%**

Table 2: Comparison of OCShell-Mamba on ADE20K semantic segmentation. Parameters and FLOPs are reported in millions and billions, respectively.

Method	Params.	FLOPs	mIoU
ResNet-50 (2016)	67M	–	41.2
Vim-S (2024)	46M	–	44.9
DeiT-S (2021)	43M	–	44.0
UniRepLKNet (2024)	61M	946G	48.6
ConvNeXt (2022)	60M	939G	46.0
Swin (2021)	60M	945G	44.4
Agent-Swin (2024)	61M	954G	46.7
NAT (2023)	58M	934G	47.1
QFormer$_h$ (2023)	61M	–	46.9
PartialFormer-B3 (2024)	65M	923G	47.0
FractalMamba (2025)	55M	937G	48.0
BiFormer (2023)	–	–	49.8
MambaOut (2025)	54M	938G	47.4
VMamba (2024)	62M	949G	48.0
LocalVMamba (2024)	57M	970G	47.9
DAMamba (2025b)	55M	937G	50.3
OCShell-Mamba (ours)	59M	944G	**50.8**

Semantic Segmentation on ADE20K: We evaluate semantic segmentation on ADE20K (Zhou et al., 2017) to assess transfer from image classification to dense prediction. Following DAMamba (Li et al., 2025b), we use UPerNet (Xiao et al., 2018) as the segmentation head. The backbone is initialized with ImageNet-1K pretrained weights, while the decoder and task-specific layers are randomly initialized. OCShell-Mamba is trained for 160K iterations using AdamW and a batch size of 16. All experiments use MMSegmentation (Contributors, 2020). This controlled setup keeps the segmentor, initialization, and training schedule identical to the matched DAMamba configuration, ensuring that performance differences primarily reflect the backbone design.

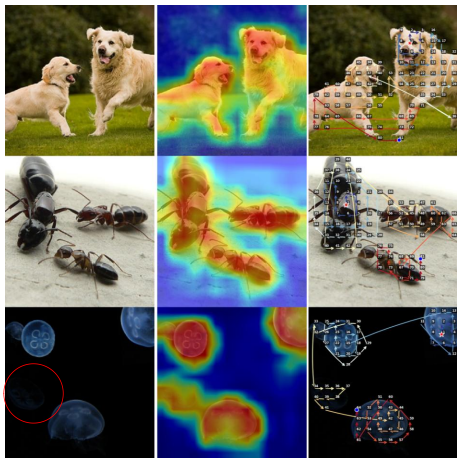
Object-centric propagation under pixel-level prediction. Table 2 reports single-scale ADE20K results with the UPerNet decoder. OCShell-Mamba reaches 50.8% mIoU, outperforming the matched DAMamba baseline by +0.5 points under the same training setup. While DAMamba uses adaptive scanning, OCShell-Mamba instead propagates features via compact shell-wise object regions, indicating that object-centric propagation complements adaptive scanning, especially for fine-grained spatial reasoning. This trend is consistent with ImageNet-1K but appears in a more demanding dense prediction task. Since the segmentor, initialization, and training schedule are fixed, the gains mainly reflect the differences in backbone representation, supporting the benefit of preserving object-contiguous structure for dense prediction.

Object Detection and Instance Segmentation on COCO 2017: We evaluate object detection and instance segmentation on COCO 2017 (Lin et al., 2014), following the DAMamba (Li et al., 2025b) protocol with Mask R-CNN (He et al., 2017) as the detector. The backbone is initialized from ImageNet-1K pretrained weights, while task-specific heads are randomly initialized. OCShell-Mamba is trained for 12 epochs using the standard $1\times$ schedule in MMDetection (Chen et al., 2019). This controlled setup keeps the detector architecture, initialization, and training schedule identical to DAMamba, ensuring that performance differences primarily arise from the backbone design.

Object-centric propagation under instance-level recognition. Table 3 reports the COCO 2017 results under the Mask R-CNN framework. OCShell-Mamba achieves 49.1 box AP and 43.7 mask AP, improving over the matched DAMamba baseline by $+0.6$ AP^b and $+0.3$ AP^m. The gains are consistent across stricter localization and mask-quality metrics, including $+0.6$ $AP^b{}_{75}$, $+0.9$ $AP^m{}_{50}$, and $+0.6$ $AP^m{}_{75}$. This comparison is targeted: DAMamba improves scan flexibility via adaptive traversal, whereas OCShell-Mamba changes the propagation unit by organizing object-like regions into compact shell-wise sequences. The consistent improvements suggest that object-centric propagation complements adaptive scanning in instance-level recognition. COCO is more challenging than image classification, needing precise localization and mask prediction instead of image-level labels.

Table 3: Object detection and instance segmentation results on COCO 2017 using Mask R-CNN. Parameters and FLOPs are reported in millions and billions, respectively. FLOPs are measured at an input resolution of 1280×800. our method has achieved the best performance in both Box AP and Mask AP while keeping parameters and FLOPs comparable.

Backbone	Box AP			Mask AP			Par. (M)	FLOPs (G)
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}		
Swin	42.7	65.2	46.8	39.3	62.2	42.2	48	267
DAT	44.7	67.6	48.7	42.4	66.1	45.5	42	278
CSWin	46.7	68.6	51.3	42.2	65.6	45.4	42	279
ConvNeXt	44.2	66.6	48.8	40.1	63.3	42.8	48	262
PVTv2-B2	45.3	66.1	49.6	41.1	64.2	44.5	39	309
$QFormer_h$	45.9	68.5	50.3	41.5	65.2	44.6	49	–
PartialFormer-B3	45.0	–	–	40.9	–	–	54	248
BiFormer	47.8	69.8	52.2	43.0	66.8	46.5	–	–
MambaOut	45.1	67.3	49.3	41.0	64.1	44.1	43	262
VMamba	47.3	69.3	52.0	42.7	66.4	45.9	50	271
LocalVMamba	46.7	68.7	51.2	42.2	65.9	45.5	41	296
FractalMamba	46.8	68.7	50.8	42.4	65.9	45.8	45	266
DAMamba	48.5	70.3	53.3	43.4	67.2	46.7	45	284
OCShell-Mamba	**49.1**	**70.8**	**53.9**	**43.7**	**68.1**	**47.3**	49	294



(a) Input image (b) Objectness field (c) Routing path

Figure 4: **SOCO routing on three scenes.** **(a)** input image, **(b)** learned objectness field (warm colors: stronger foreground responses), **(c)** resulting routing paths.

With the detector architecture and training schedule fixed, the gains mainly reflect the backbone differences rather than detector-specific tuning. OCShell-Mamba uses slightly more computation than DAMamba (294G *vs.* 284G FLOPs), so the results do not indicate higher efficiency. Instead, they reinforce the trend across ImageNet and ADE20K: preserving object-contiguous structure during state propagation improves downstream visual representations across tasks.

Shell-wise Object-Centric Routing on real images: Fig. 4 visualizes SOCO routing on three ImageNet-1K validation images with different object configurations. The bottom row highlights a faint jellyfish in the lower-left region. Despite its weak contrast, the objectness field identifies it as an independent candidate, allowing SOCO to route it separately from the larger regions above. Similar behavior is observed in the crowded scenes in the top two rows, where closely spaced or overlapping regions remain separate when the response between them is sufficiently weak, and each candidate occupies a contiguous sequence segment ordered from smaller to larger radial shells. These examples suggest that meaningful object structure can emerge without segmentation, instance, or saliency supervision, though adjacent instances may merge when connected by strong responses.

5 Conclusion

OCShell-Mamba replaces global scan-based token propagation with object-centric routing over object candidates. Each candidate is derived from a learned objectness field, assigned a local center, and traversed shell-by-shell from center to boundary, promoting compact and object-aligned state propagation. A persistent topology memory further preserves object structure across stages despite resolution changes. On ImageNet-1K, ADE20K, and COCO 2017, OCShell-Mamba consistently outperforms the matched DAMamba baseline for image classification and dense prediction tasks under the same scale and training setup. These gains over a strong adaptive-scan baseline suggest that object-centric recurrent state propagation is an effective, complementary design for vision state space models. We explain the limitations of OCShell-Mamba and future work in the Supplement.

AI use statement

Generative AI tools were used to assist with manuscript editing, literature and bibliographic checks, and consistency checks between the method description, reported results, and released code. All research decisions, experimental design, implementation, results, and conclusions were independently verified and finalized by the authors.

REFERENCES

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

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

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. UniRepLKNet: A universal perception large-kernel ConvNet for audio, video, point cloud, time-series and image recognition. In *CVPR*, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Hongyang Gu, Qisong Yang, Lei Pu, Siming Han, and Yao Ding. Reidmamba: Learning discriminative features with visual state space model for person re-identification. *IEEE Transactions on Multimedia*, 2026.

Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Siyuan Pan, Pengfei Wan, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *ECCV*, 2024.

Osama Hardan, Omar Elshenhabi, Tamer Khattab, and Mohamed Mabrok. Flatten wisely: How patch order shapes mamba-powered vision for mri segmentation. *arXiv preprint arXiv:2507.13384*, 2025.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6185–6194, 2023.

Jiahao He, Keren Fu, Xiaohong Liu, and Qijun Zhao. Samba: A unified mamba-based framework for general salient object detection. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25314–25324, 2025.

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

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Yi-Kuan Hsieh, Jun-Wei Hsieh, Ming-Ching Chang, Yu-Chee Tseng, et al. Partial ring scan: Revisiting scan order in vision state space models. *arXiv preprint arXiv:2602.04170*, 2026.

Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes Fischer, and Björn Ommer. Zigma: A dit-style zigzag mamba diffusion model. In *European conference on computer vision*, pp. 148–166. Springer, 2024.

Tao Huang, Xiaohuan Pei, Shan You, Fei Wang, Chen Qian, and Chang Xu. Localmamba: Visual state space model with windowed selective scan. In *European conference on computer vision*, pp. 12–22. Springer, 2024.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Min Wu Jeong and Chae Eun Rhee. Lc-mamba: Local and continuous mamba with shifted windows for frame interpolation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17671–17681. IEEE, 2025.

Jianfei Jiang, Qiankun Liu, Hongyuan Liu, Haochen Yu, Liyong Wang, Jiansheng Chen, and Huimin Ma. Mvsmamba: Multi-view stereo with state space model. *arXiv preprint arXiv:2511.01315*, 2025.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Boyun Li, Haiyu Zhao, Wenxin Wang, Peng Hu, Yuanbiao Gou, and Xi Peng. MaIR: A locality-and continuity-preserving mamba for image restoration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7491–7501, 2025a.

Tanzhe Li, Caoshuo Li, Jiayi Lyu, Hongjuan Pei, Baochang Zhang, Taisong Jin, and Rongrong Ji. Damamba: Vision state space model with dynamic adaptive scan. *arXiv preprint arXiv:2502.12627*, 2025b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Hanzhou Liu, Chengkai Liu, Jiacong Xu, Peng Jiang, and Mi Lu. Xyscannet: A state space model for single image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 779–789, 2025a.

Leiye Liu, Miao Zhang, Jihao Yin, Tingwei Liu, Wei Ji, Yongri Piao, and Huchuan Lu. Defmamba: Deformable visual state space model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 8838–8847, 2025b.

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 A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 103031–103063. Curran Associates, Inc., 2024. doi: 10.52202/079017-3273. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/baa2da9ae4bfed26520bb61d259a3653-Paper-Conference.pdf.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

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

Meng Lou, Yunxiang Fu, and Yizhou Yu. Sparx: A sparse cross-layer connection mechanism for hierarchical vision mamba and transformer networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19104–19114, 2025.

Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.

Badri N Patro and Vijay S Agneeswaran. Simba: Simplified mamba-based architecture for vision and multivariate time series. *arXiv preprint arXiv:2403.15360*, 2024.

Xiaohuan Pei, Tao Huang, and Chang Xu. Efficientvmamba: Atrous selective scan for light weight visual mamba. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 6443–6451, 2025.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Abdelrahman Shaker, Syed Talal Wasim, Salman Khan, Juergen Gall, and Fahad Shahbaz Khan. Groupmamba: Efficient group-based visual state space model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14912–14922, 2025.

Yuheng Shi, Mingjia Li, Minjing Dong, and Chang Xu. Vssd: Vision mamba with non-causal state space duality. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10819–10829, 2025.

Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. *arXiv preprint arXiv:2109.14279*, 2021.

Oriane Siméoni, Chloé Sekkat, Gilles Puy, Antonín Vobeckỳ, Éloi Zablocki, and Patrick Pérez. Unsupervised object localization: Observing the background to discover objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3176–3186, 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*, pp. 10347–10357. PMLR, 2021.

Xuan-Thuy Vo, Duy-Linh Nguyen, Adri Priadana, and Kang-Hyun Jo. Efficient vision transformers with partial attention. In *European Conference on Computer Vision*, pp. 298–317. Springer, 2024.

Feng Wang, Timing Yang, Yaodong Yu, Sucheng Ren, Guoyizhe Wei, Angtian Wang, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Adventurer: Optimizing vision mamba architecture designs for efficiency. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30157–30166, 2025.

Yangtao Wang, Xi Shen, Yuan Yuan, Yuming Du, Maomao Li, Shell Xu Hu, James L Crowley, and Dominique Vaufreydaz. Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. *IEEE transactions on pattern analysis and machine intelligence*, 45(12):15790–15801, 2023.

Yifeng Wu, Huimin Huang, Shangjie Zhou, Yawen Huang, Hao Zheng, Yun Chen, Xian Wu, Ruize Han, and Guanhua Chen. Infoscan: Information-efficient visual scanning via resource-adaptive walks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=aiGqJwOE2x`.

Haoke Xiao, Lv Tang, Peng-tao Jiang, Hao Zhang, Jinwei Chen, and Bo Li. Boosting vision state space model with fractal scanning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 8646–8654, 2025.

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

Fei Xie, Weijia Zhang, Zhongdao Wang, and Chao Ma. Quadmamba: Learning quadtree-based selective scan for visual state space model. *Advances in Neural Information Processing Systems*, 37:117682–117707, 2024.

Zhaohu Xing, Tian Ye, Yijun Yang, Du Cai, Baowen Gai, Xiao-Jian Wu, Feng Gao, and Lei Zhu. Segmamba-v2: Long-range sequential modeling mamba for general 3-d medical image segmentation. *IEEE Transactions on Medical Imaging*, 45(1):4–15, 2026. doi: 10.1109/TMI.2025.3589797.

Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *arXiv preprint arXiv:2403.17695*, 2024.

Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4484–4496, 2025.

Qiming Zhang, Jing Zhang, Yufei Xu, and Dacheng Tao. Vision transformer with quadrangle attention. *TPAMI*, 2023.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Jiaheng Zhou, Wei Fang, Luyuan Xie, Yanfeng Zhou, Lianyan Xu, Minfeng Xu, Ge Yang, and Yuxing Tang. Dual-res tandem mamba-3d: Bilateral breast lesion detection and classification on non-contrast chest ct. *Advances in Neural Information Processing Systems*, 38:25494–25527, 2026.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10323–10333, 2023.

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 *ICML*, 2024.

APPENDIX

A ARCHITECTURE AND IMPLEMENTATION DETAILS

This appendix provides architecture, implementation, and experimental details for the method defined in the main paper. Equations, figures, tables, and algorithms are numbered within each appendix section. All cross-references are generated from labels.

We use M_s to denote the number of blocks in stage s; this is the block count denoted by N_i in Fig. 2, with $i = s$. It is distinct from N_s, the number of object candidates in Eq. (3).

A.1 OBJECTNESS HEAD: ARCHITECTURE AND COST

Architecture. The objectness head Φ_s operates on the spatial feature map $F^s = \mathrm{Reshape}_{H_s,W_s}(\widetilde{X}^s)$, where $\widetilde{X}^s \in \mathbb{R}^{L_s \times C_s}$ is defined in Eq. (14) and $L_s = H_s W_s$. When memory is disabled, $\widetilde{X}^s = X^s$. The head consists of a 5×5 depthwise convolution, a 1×1 convolution reducing the channels from C_s to $C_s/4$ followed by GELU, a second 5×5 depthwise convolution, and a final 1×1 projection to a single-channel logit map:

$$\begin{aligned} \Phi_s &: \mathbb{R}^{H_s \times W_s \times C_s} \to \mathbb{R}^{H_s \times W_s \times 1}, \\ R^s &= \mathrm{Flatten}(\sigma(\Phi_s(F^s))) \in [0,1]^{L_s}. \end{aligned} \tag{A.1}$$

In Eq. (1), the per-token notation identifies the response at the corresponding grid position. All convolutions preserve spatial resolution. Each score R_i^s corresponds to one grid position and aggregates a local neighborhood through the head's 9×9 receptive field. The four heads of the Tiny model add 0.13M parameters in total, below one percent of the backbone.

Ordering instantiation. We instantiate the base spatial order ρ^s as row-major (raster) order: rows are visited from top to bottom, and positions within each row are visited from left to right. This same order is used within each shell, each confidence bin, and the background segment. Shells are concatenated from the innermost outward, and candidates are concatenated in descending confidence order, followed by the background, as in Eqs. (7)–(8).

The main paper uses Ω_n^s and $\mathcal{Q}_{n,b}^s$ for sets of tokens. The pseudocode instead operates on their integer indices. For a stage with $L_s = H_s W_s$ tokens, we define

$$\begin{aligned} \mathcal{I}_s &= \{1, \dots, L_s\}, \\ \mathcal{J}_n^s &= \{i \in \mathcal{I}_s : \mathcal{T}_i^s \in \Omega_n^s\}, \\ \mathcal{H}_{n,b}^s &= \{i \in \mathcal{J}_n^s : l_i = b\}, \\ \mathcal{J}_{\mathrm{bg}}^s &= \mathcal{I}_s \setminus \bigcup_{n=1}^{N_s} \mathcal{J}_n^s. \end{aligned} \tag{A.2}$$

Here, $\mathcal{I}_s$ contains all token indices at stage s. For $n \in \{1, \dots, N_s\}$, $\mathcal{J}_n^s$ contains the indices of tokens in the n-th retained candidate. For $b \in \{0, \dots, B_s^n - 1\}$, $\mathcal{H}_{n,b}^s$ contains the indices assigned to shell b of candidate n. For each $i \in \mathcal{J}_n^s$, l_i stores the shell index computed with that candidate's shell count B_s^n in Eq. (6). Finally, $\mathcal{J}_{\mathrm{bg}}^s$ contains all indices outside the retained candidates, including those from components discarded by the minimum-size criterion. If $N_s = 0$, the union is empty and $\mathcal{J}_{\mathrm{bg}}^s = \mathcal{I}_s$.

In the pseudocode, for an index set $\mathcal{A} \subseteq \mathcal{I}_s$, $\mathrm{Ord}_{\rho^s}(\mathcal{A})$ returns the indices in $\mathcal{A}$ as a list in raster order; an empty set returns an empty list. This is the index-set version of the ordering operator in the main paper. Each connected component retained by the minimum-size criterion is stored as its index set $\mathcal{J}_n^s$. The background segment is $\mathrm{Ord}_{\rho^s}(\mathcal{J}_{\mathrm{bg}}^s)$.

Implementation constants and stage partition. At deep stages, the shell count B_s^n is computed separately from each retained candidate's radius, following the radius-based rule in the main paper. The token radius τ_t and the coordinates p_i^s use the same spatial units. At shallow stages, we use $B_s = 5$ confidence bins in Eqs. (9) and (10). Here, B_s denotes the number of objectness-confidence bins; it is distinct from the candidate-dependent radial shell count B_s^n used at deep stages. Confidence bins group tokens by objectness score and do not require candidate centers or radii.

We set $m_s = 2$ and $\epsilon_c = \epsilon_r = 10^{-6}$. The stage partition in Eq. (11) is $\mathcal{S}_{\mathrm{shallow}} = \{1, 2\}$ and $\mathcal{S}_{\mathrm{deep}} = \{3, 4\}$, consistent with Fig. 2. Shallow stages use confidence-based routing; only deep stages construct connected components and shells.

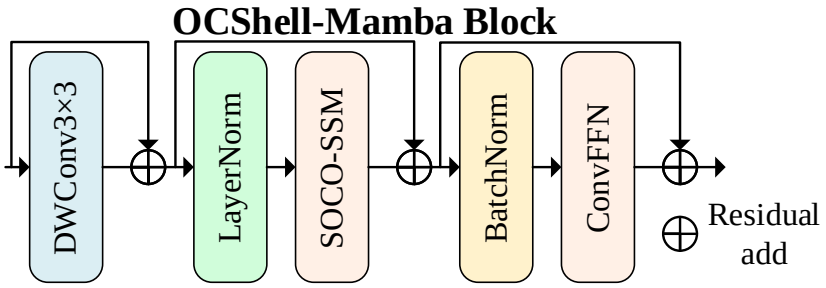


Figure A.1: **OCShell-Mamba block architecture.** The depthwise convolution performs local spatial mixing, SOCO-SSM reorders tokens for object-centric state propagation and restores their original spatial locations afterward, and ConvFFN performs channel mixing. Every submodule is residual, with LayerNorm before SOCO-SSM and BatchNorm before ConvFFN.

At deep stages, every component retained by CC_{m_s} participates in the routing order. Candidates are sorted by descending mean objectness, with ties broken by the smallest raster index in each component. The background consists of all remaining tokens.

Complexity. The objectness head in Eq. (A.1) is evaluated once per stage. With fixed convolution kernel sizes, its cost is $O(L_s C_s^2)$, including channel projections and depthwise convolutions.

At shallow stages, confidence quantization is linear in L_s; a sorting-based construction of the confidence order costs $O(L_s \log L_s)$. At deep stages, let $\mathcal{C}_{\mathrm{CC}}(L_s)$ denote the cost of connected-component labeling. With component-index lists, candidate statistics and shell assignments can be computed by visiting each foreground token a constant number of times. Tokens can then be grouped by their assigned shell indices and ordered by candidate rank, shell index, and base spatial order. Empty shells contribute no indices and can be skipped. This sorting-based construction costs $\mathcal{C}_{\mathrm{CC}}(L_s) + O(L_s \log L_s)$. The pseudocode below groups tokens once by shell and iterates only over occupied shell indices; it does not rescan a candidate for every possible shell. All retained candidates participate in the route.

The route is constructed once per stage and shared by its M_s blocks. Within each block, applying the permutation and its inverse costs $O(L_s C_s)$. Reordering preserves the sequence length and does not change the SSM recurrence. These expressions describe SOCO; PoMe has additional cross-attention and projection costs. Wall-clock overhead depends on the implementations of component labeling, sorting, and token reordering and requires empirical profiling.

A.2 OCShell-Mamba block

Figure A.1 illustrates the block-level architecture of OCShell-Mamba. Operating on features of the $H_s \times W_s$ grid with C_s channels, each block sequentially applies a 3×3 depthwise convolution, a SOCO-SSM module, and a ConvFFN, performing local spatial mixing, object-centric state propagation, and channel mixing, respectively. Every submodule is residual: the depthwise convolution uses an identity-initialized re-parameterized residual and no normalization, LayerNorm precedes SOCO-SSM, and BatchNorm precedes ConvFFN, with stochastic depth applied to the latter two branches.

The overall structure follows the hierarchical Mamba block of the baseline, with its conventional SSM module replaced by SOCO-SSM. The routing order π^s is constructed once per stage from the objectness field following Eqs. (1)–(11) and is shared by all M_s blocks of that stage. Inside SOCO-SSM, the input tokens are gathered according to π^s and processed as a routed 1D sequence by the SSM, and the inverse permutation restores every output token to its original spatial location before the residual connection, as in Eq. (12). Apart from the routing, the remaining components of the module follow the baseline configuration unchanged. Thus SOCO changes only the order in which recurrent states are updated; it does not alter the spatial correspondence or the output resolution of the feature map.

A.3 Per-stage data flow

Figure A.2 illustrates how the spatial and memory streams interact within a single stage. The spatial stream carries the stage features $X^s \in \mathbb{R}^{L_s \times C_s}$, with $L_s = H_s W_s$, whereas the memory stream maintains the fixed-size bank $G^{s-1} \in \mathbb{R}^{N_m \times D}$ inherited from the previous stage. The two streams interact once

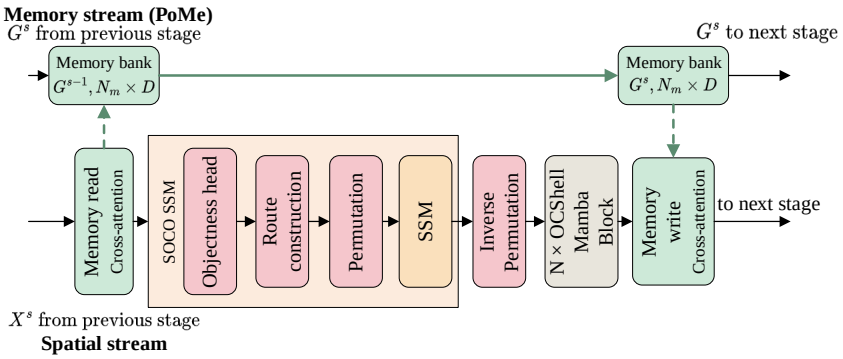


Figure A.2: **Per-stage interaction between the spatial and memory streams.** At the beginning of stage s, the read module $\mathcal{R}_s$ retrieves information from G^{s-1} and injects it into the spatial features, as in Eq. (14). The objectness head then constructs the routing order, which is shared by the M_s OCShell-Mamba blocks of the stage; within each block, SOCO reorders tokens for state propagation and restores their original spatial locations afterward. At the end of the stage, the write module $\mathcal{W}_s$ updates the memory using the stage output Y^s, as in Eq. (15). The memory maintains a fixed shape of $N_m \times D$ across all stages, while stage-specific projections connect it to spatial features of dimension C_s.

before and once after the spatial blocks, through the stage-specific read and write modules of Eqs. (14) and (15).

At the beginning of stage s, the read module $\mathcal{R}_s$ uses the spatial tokens as queries and the memory tokens as keys and values, so that the cross-attention of Eq. (14) retrieves information accumulated in earlier stages and injects it into the current spatial representation. Stage-specific projections map the spatial channel dimension C_s to the fixed memory dimension D when computing the interaction. The read module projects its attention output back to C_s channels before adding it to X^s. The write module returns an update of shape $N_m \times D$ for addition to G^{s-1}.

The augmented features $\widetilde{X}^s$ are then processed by the M_s OCShell-Mamba blocks of the stage. At the start of the stage, the objectness head predicts R^s on $\widetilde{X}^s$ and constructs π^s according to Eq. (11): confidence-based routing at shallow stages and object-candidate shell routing at deep stages. The resulting permutation is shared by all M_s blocks. Within each SOCO-SSM module, tokens are gathered according to π^s and processed by the SSM as a 1D sequence, and the inverse permutation restores every output token to its original spatial location before the residual connection and the subsequent block are applied. Routing therefore changes only the order of recurrent state propagation, while the spatial resolution and the token-to-location correspondence remain unchanged.

After the final complete OCShell-Mamba block, including its residual connections and ConvFFN, the stage output Y^s is written back to the memory by $\mathcal{W}_s$ following Eq. (15). The memory tokens serve as queries and the stage output as keys and values. The resulting update is added to G^{s-1} and layer-normalized, and G^s is passed to the next stage.

Because the memory bank always contains N_m tokens of dimension D, its shape remains fixed across all four stages. Unlike the spatial stream, it is neither downsampled nor resized when the feature resolution changes. It can therefore retain and transfer object-level information across the network hierarchy while interacting with each stage through lightweight cross-attention.

B TRAINING OBJECTIVE

OCShell-Mamba is trained end-to-end with the task objective, an auxiliary classification loss on the memory tokens, and two annotation-free regularizers on the objectness field. The route itself is built from the stop-gradient copy $\bar{R}^s$ of Eq. (2) and carries no gradient, so the latter two act on R^s directly:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{mem}}\mathcal{L}_{\text{mem}} + \lambda_{\text{obj}}\sum_s \mathcal{L}^s_{\text{obj}} + \lambda_{\text{coh}}\sum_s \mathcal{L}^s_{\text{coh}}, \tag{B.1}$$

where $\mathcal{L}_{\text{task}}$ is the standard objective of the target task: cross-entropy for classification, the standard segmentation loss for ADE20K, and the standard detection and instance segmentation losses for

COCO. The two objectness terms are summed over the stages that perform routing. The objectness regularizers use no additional segmentation, saliency, or instance annotations beyond the standard supervision of the target task.

Memory objective. $\mathcal{L}_{\mathrm{mem}}$ is the cross-entropy between the ground-truth label y and the logits produced by an auxiliary head Ψ applied to the memory G^S after the last of the S stages, obtained by iterating Eq. (15):

$$\mathcal{L}_{\mathrm{mem}} = \mathrm{CE}\big(\Psi(G^S),\, y\big), \tag{B.2}$$

where Ψ pools G^S along the memory-token axis and projects to the class logits. The memory classifier uses the same training targets as the task classifier, including the target mixing and smoothing applied by the classification training recipe. The memory bank thus receives a direct training signal in addition to the gradients flowing back through the read and write modules $\mathcal{R}_s$ and $\mathcal{W}_s$. At $\lambda_{\mathrm{mem}} = 0$ the memory is shaped only indirectly through those modules, while a larger value ties it more tightly to the classification target. The head Ψ and $\mathcal{L}_{\mathrm{mem}}$ are used only during ImageNet classification pretraining. For dense prediction, Ψ is discarded and $\mathcal{L}_{\mathrm{mem}}$ is omitted.

Objectness prior. Nothing in Eqs. (3)–(11) constrains what Φ_s should respond to, since the route is built from $\bar{R}^s$. We therefore add a weak prior derived from the features themselves. Let $\hat{\mathcal{T}}_i^s = \mathcal{SG}(\mathcal{T}_i^s / \max(\|\mathcal{T}_i^s\|_2, \epsilon_{\mathrm{n}}))$, where $\epsilon_{\mathrm{n}} > 0$, and let $\partial^s \subseteq \mathcal{I}_s$ be the indices of tokens on the border of the feature grid. A background prototype u^s and a per-token dissimilarity q_i^s are given by

$$u^s = \frac{\sum_{i\in\partial^s} \hat{\mathcal{T}}_i^s}{\max\Big(\Big\|\sum_{i\in\partial^s} \hat{\mathcal{T}}_i^s\Big\|_2, \epsilon_{\mathrm{n}}\Big)}, \qquad q_i^s = 1 - \big\langle u^s, \hat{\mathcal{T}}_i^s \big\rangle, \tag{B.3}$$

and the pseudo-target is computed by stabilized per-image min–max rescaling,

$$t_i^s = \mathcal{SG}\left(\frac{q_i^s - \min_j q_j^s}{\max(\max_j q_j^s - \min_j q_j^s, \epsilon_{\mathrm{t}})}\right), \qquad \epsilon_{\mathrm{t}} > 0. \tag{B.4}$$

The objectness term combines a binary cross-entropy against this pseudo-target with a separation term between the objectness-weighted foreground and background prototypes,

$$\mathcal{L}_{\mathrm{obj}}^s = \mathrm{BCE}\big(R^s,\, t^s\big) + \beta \big\langle v_{\mathrm{fg}}^s, v_{\mathrm{bg}}^s \big\rangle, \tag{B.5}$$

where v_{fg}^s and v_{bg}^s are the ℓ_2-normalized averages of $\hat{\mathcal{T}}_i^s$ weighted by R_i^s and by $1 - R_i^s$, and $\beta = 0.1$. Because the background prototype, the pseudo-target, and the feature directions $\hat{\mathcal{T}}_i^s$ in Eqs. (B.3)–(B.5) are all treated as constants, the gradient of this term reaches the network only through R^s. It uses only the image content: no segmentation, saliency, or instance annotation is involved.

Routing coherence. With the routing permutation held fixed, the second term penalizes differences in objectness scores between adjacent positions of the routed sequence. It includes transitions between candidates and between foreground and background. Writing $\pi^s(j)$ for the original token index at sequence position j,

$$\mathcal{L}_{\mathrm{coh}}^s = \begin{cases} \dfrac{1}{L_s - 1} \displaystyle\sum_{j=1}^{L_s-1} \big|R_{\pi^s(j)}^s - R_{\pi^s(j+1)}^s\big|, & L_s > 1, \\ 0, & L_s = 1. \end{cases} \tag{B.6}$$

The permutation is held fixed when this term is evaluated, so the gradient reaches Φ_s through R^s only.

Weights. During ImageNet classification pretraining, we set $\lambda_{\mathrm{mem}} = 0.4$ and $\lambda_{\mathrm{obj}} = \lambda_{\mathrm{coh}} = 0.1$. The sensitivity of λ_{mem} is reported in Appendix C.3.

C Ablation studies

We use mini-ImageNet to study the contribution of each component in OCShell-Mamba and to set the hyperparameters used in the main experiments.

Table C.1: **Component ablation on mini-ImageNet.** We compare SOCO-only, PoMe-only, and full OCShell-Mamba configurations. The SOCO-only variant removes the memory bank, whereas the PoMe-only variant replaces SOCO with an outside-in spiral scan. DAMamba-T is retrained on the same mini-ImageNet split (Sec. C.1) and included only as an external reference; Δ denotes the absolute Top-1 gain over this reference. PoMe hyperparameters are tuned on the PoMe-only variant (Tab. C.2) and reused unchanged in the full model.

Configuration	SOCO	PoMe	Top-1 (%)	Δ
DAMamba-T (ref.)	–	–	87.48	–
OCShell-Mamba variants				
SOCO only	✓	–	88.50	+1.02
PoMe only	–	✓	88.60	+1.12
Full model	✓	✓	**88.91**	**+1.43**

C.1 Dataset and protocol

mini-ImageNet is a 100-class subset of ImageNet-1K that keeps the original images at 224×224 resolution. The split contains 50,000 training images and 10,000 validation images drawn from the ImageNet-1K training set, plus 5,000 test images (50 per class) drawn from the ImageNet-1K validation set. We use it for ablation rather than the full ImageNet-1K because each run is much shorter under our compute, which lets us sweep hyperparameters at the cost of a single ImageNet-1K run. All numbers are top-1 accuracy on the test split, averaged over three independent runs per configuration. Hyperparameters and checkpoints are selected using the validation split. The test split is used only for reporting and does not influence model selection. Optimizer, schedule, augmentations, and the 300-epoch budget all match the main classification setup.

For a fair comparison, we retrain DAMamba on the same mini-ImageNet split using the published code and our protocol; the 87.48% in Tab. C.1 is this retrained number, not the figure reported on full ImageNet-1K.

C.2 Component ablation

OCShell-Mamba has two design choices that distinguish it from a standard hierarchical Mamba backbone: shell-wise object-centric routing (SOCO) and persistent topology memory (PoMe). We compare three variants of our architecture and use DAMamba-T as an external reference (Tab. C.1).

The SOCO-only variant uses shell-wise object-centric routing without the memory bank; the PoMe-only variant keeps the memory bank but replaces SOCO with an outside-in spiral scan that traverses the feature grid from the boundary toward the center independently of any object structure.

SOCO alone reaches 88.50%, $+1.02$ over the DAMamba-T reference, and PoMe alone reaches 88.60%, $+1.12$. The full OCShell-Mamba combining both reaches 88.91%, $+1.43$ over the reference and $+0.31$ over the stronger single component. The full model has the highest mean accuracy among the three OCShell-Mamba configurations. It improves on SOCO-only by 0.41 percentage points and on PoMe-only by 0.31 percentage points under the reported protocol.

C.3 Hyperparameter ablation

We next examine three PoMe hyperparameters: the auxiliary loss weight λ_{mem}, the number of memory tokens N_m, and the memory dimension D. All sweeps use the PoMe-only configuration with the outside-in spiral scan, keeping SOCO disabled. We use $\lambda_{\text{mem}}{=}0.4$, $N_m{=}64$, and $D{=}256$ as the reference configuration, which corresponds to the 88.60% PoMe-only result in Tab. C.1. For each sweep, one hyperparameter is varied while the other two remain fixed. The results are reported in Tab. C.2.

Auxiliary loss weight λ_{mem}. The auxiliary loss controls how strongly the memory representations are regularized during training. Among the tested weights, $\lambda_{\text{mem}}{=}0.2$ yields 88.44%. Increasing the weight to 0.4 or 0.6 improves the accuracy to 88.60%, whereas a larger weight of 1.0 reduces it to

Table C.2: **Hyperparameter ablation on mini-ImageNet.** Each block varies one PoMe hyperparameter while keeping the other two at the reference configuration (λ_{mem}=0.4, N_m=64, and D=256). Bold indicates the reference setting. Δ denotes the absolute Top-1 gain over DAMamba-T (87.48%; Tab. C.1).

Hyperparameter	Setting	Top-1 (%)	Δ
Aux. weight λ_{mem}	0.2	88.44	+0.96
	0.4	**88.60**	**+1.12**
	0.6	88.60	+1.12
	1.0	88.49	+1.01
Memory size N_m	32	88.19	+0.71
	64	**88.60**	**+1.12**
	128	88.19	+0.71
Memory dim. D	128	88.37	+0.89
	256	**88.60**	**+1.12**

88.49%. The reference weight λ_{mem}=0.4 attains the highest reported mean accuracy in this sweep, tied with λ_{mem}=0.6, while using a smaller auxiliary loss weight.

Memory bank size N_m. The memory bank size determines how many memory tokens are carried across stages. With N_m=32, the model reaches 88.19%, which is 0.41 percentage points below the 64-token configuration. Increasing the bank to 64 tokens improves the accuracy to 88.60%. Expanding it further to 128 tokens does not provide additional gains and instead returns to 88.19%. We therefore use N_m=64, which achieves the best performance without introducing unnecessary memory overhead.

Memory dimension D. The memory dimension determines the representation width of each memory token. Increasing D from 128 to 256 improves the accuracy from 88.37% to 88.60%, showing that the wider representation provides greater capacity for preserving object-level information. We adopt D=256 as the default setting. We do not further increase the dimension to 512 because the computational and storage costs of PoMe grow with D, while our objective is to maintain a compact fixed-size memory.

Overall, the reference configuration achieves the best or tied-best performance in all three sweeps. Following the validation-based selection protocol in Sec. C.1, we use λ_{mem}=0.4, N_m=64, and D=256 for the full OCShell-Mamba model, where adding SOCO routing increases the accuracy from 88.60% to 88.91% (Tab. C.1). The same memory size $N_m = 64$ and dimension $D = 256$ are used for ADE20K and COCO. We do not perform task-specific re-tuning, allowing the downstream experiments to evaluate whether the fixed-size memory design transfers across tasks and spatial resolutions.

D PSEUDOCODE

Algorithm D.1 gives the construction of the SOCO routing order at a deep stage for a single image; the implementation vectorizes every step over the batch. The shallow-stage confidence order is constructed by quantization and stable sorting. For token i, let $b_i = \min\{B_s - 1, \lfloor B_s \bar{R}_i^s \rfloor\}$ denote its confidence-bin index. We visit bins in descending b_i, with raster order within each bin. This agrees with Eqs. (9) and (10) for $0 \le \bar{R}_i^s \le 1$.

Objectness field and thresholding. The head Φ_s produces one logit per grid position. Applying the sigmoid and flattening the resulting map gives $R^s \in [0,1]^{L_s}$, whose entry R_i^s corresponds to token i. The final routing permutation is determined by discrete thresholding, connected-component labeling, shell assignments, and sorting. Although centers and distances are intermediate numerical quantities, the discrete route does not provide a differentiable path to the objectness head. We use the stop-gradient copy $\bar{R}^s$ in Eq. (2) throughout route construction, while the non-detached field R^s remains available to the objectness prior and routing-coherence losses. The threshold τ_s is

Algorithm D.1 Shell-wise Object-Centric Routing (SOCO)

Require: stage features $F^s \in \mathbb{R}^{H_s \times W_s \times C_s}$; head Φ_s; token coordinates $\{p_i^s\}_{i=1}^{L_s}$ with $L_s = H_s W_s$; constants m_s, $\tau_t > 0$, $\epsilon_c > 0$, $\epsilon_r > 0$; base ordering ρ^s
Ensure: routing order π_{shell}^s and its inverse

$R^s \leftarrow \mathrm{Flatten}\big(\sigma(\Phi_s(F^s))\big)$; $\bar{R}^s \leftarrow \mathcal{SG}(R^s)$ {Eqs. (1)–(2)}
$\tau_s \leftarrow \mathrm{Otsu}(\bar{R}^s)$; $\mathcal{M}^s \leftarrow \mathbf{1}\big[\bar{R}^s > \tau_s\big]$
$\mathcal{C}^s = \{\mathcal{J}_1^s, \ldots, \mathcal{J}_{N_s}^s\} \leftarrow \mathrm{CC}_{m_s}(\mathcal{M}^s)$ {Eq. (3)}
if $\mathcal{C}^s = \emptyset$ **then**
  return $\mathrm{Ord}_{\rho^s}(\{1, \ldots, L_s\})$ and its inverse
end if
for all $\mathcal{J}_n^s \in \mathcal{C}^s$ **do**
  $c_n^s \leftarrow \sum_{i \in \mathcal{J}_n^s} \bar{R}_i^s \, p_i^s \big/ \big(\sum_{i \in \mathcal{J}_n^s} \bar{R}_i^s + \epsilon_c\big)$ {Eq. (4)}
  $a_n^s \leftarrow |\mathcal{J}_n^s|^{-1} \sum_{i \in \mathcal{J}_n^s} \bar{R}_i^s$
end for
$\eta \leftarrow [1, \ldots, N_s]$, sorted by decreasing a_n^s; break ties by increasing $\min \mathcal{J}_n^s$
$\pi_{\text{fg}} \leftarrow [\,]$; $\mathcal{J}_{\text{fg}} \leftarrow \emptyset$
for all n in the ordered list η **do**
  $r_n^s \leftarrow \max_{i \in \mathcal{J}_n^s} \|p_i^s - c_n^s\|_2$
  $B_s^n \leftarrow \max\big\{1, \lfloor r_n^s / \tau_t \rfloor\big\}$
  for all $i \in \mathcal{J}_n^s$ **do**
    $d_{n,i}^s \leftarrow \|p_i^s - c_n^s\|_2 / (r_n^s + \epsilon_r)$ {Eq. (5)}
    $l_i \leftarrow \min\big(\lfloor B_s^n d_{n,i}^s \rfloor, \, B_s^n - 1\big)$ {Floating-point guard for Eq. (6)}
  end for
  Group indices in $\mathcal{J}_n^s$ by l_i to form the nonempty sets $\mathcal{H}_{n,b}^s$
  for all occupied shell indices b in increasing order **do**
    $\pi_{\text{fg}} \leftarrow \mathrm{Concat}\big(\pi_{\text{fg}}, \, \mathrm{Ord}_{\rho^s}(\mathcal{H}_{n,b}^s)\big)$
  end for
  $\mathcal{J}_{\text{fg}} \leftarrow \mathcal{J}_{\text{fg}} \cup \mathcal{J}_n^s$
end for
$\pi_{\text{bg}} \leftarrow \mathrm{Ord}_{\rho^s}\big(\{1, \ldots, L_s\} \setminus \mathcal{J}_{\text{fg}}\big)$
return $\pi_{\text{shell}}^s \leftarrow \mathrm{Concat}(\pi_{\text{fg}}, \, \pi_{\text{bg}})$ and its inverse

recomputed for every image by Otsu's method rather than fixed in advance, because the scale of the objectness response varies across images and shifts over the course of training; a fixed cut would select nearly all tokens on some inputs and none on others.

Object candidate formation. This step realizes CC_{m_s} of Eq. (3), turning the unstructured mask into a set of candidates that the routing can treat as units. Connected-component labeling groups spatially contiguous foreground positions, so two responses separated by background become separate candidates while one contiguous response stays whole. We use 8-connectivity, matching the implementation: at the resolutions where shell routing operates, an object may be only a few tokens wide, and diagonal adjacency is often the only link holding such a region together. Components smaller than m_s tokens are dropped, since a single-token component cannot provide a meaningful multi-token shell traversal. Dropped tokens are not removed from the sequence; they rejoin the final background segment.

Center-to-boundary shell ordering. The remaining lines follow Eqs. (4)–(8) and impose two levels of ordering, one between candidates and one inside each. Between candidates, the mean objectness a_n^s ranks them, so the most confident object is traversed first and its tokens enter the recurrence before less confident candidates. All N_s retained candidates participate in the routing order. Inside a candidate, the center c_n^s is computed from the objectness-weighted token coordinates with the stabilizing denominator in Eq. (4). Higher-response tokens contribute more strongly to its location. Distances to the center are divided by $r_n^s + \epsilon_r$, as in Eq. (5). The shell count B_s^n is computed from each candidate's radius, so different candidates may have different numbers of shells. The pseudocode retains at least one shell by using $B_s^n = \max\{1, \lfloor r_n^s / \tau_t \rfloor\}$. This agrees with the floor rule in the main paper when $r_n^s \geq \tau_t$ and supplies a single-shell fallback otherwise. Within a candidate, shell indices increase with radial distance; the same shell index need not represent the same normalized distance across candidates with different shell counts. Shells are radial intervals

restricted to the candidate support; they need not follow an irregular candidate boundary. The minimum in the pseudocode guards against floating-point rounding at the outer endpoint. For $B_s^n \geq 1$ and $\epsilon_r > 0$, it is inactive in exact arithmetic, so the shell index agrees with Eq. (6). Within each shell, tokens are visited in the base spatial order ρ^s. Empty shells contribute empty lists and are skipped in the pseudocode without changing $\mathrm{Shell}(\Omega_n^s)$ in Eq. (7). All remaining tokens follow in the base order, and an empty candidate set reduces the whole route to ρ^s, as specified in Eq. (8). After the SSM has processed the routed sequence, the inverse permutation returns every output token to its original grid position, so the reordering is confined to the recurrence.

E LIMITATIONS AND FUTURE WORK

Limitations: OCShell-Mamba has several limitations. First, thresholded objectness may merge nearby objects in crowded scenes. Moreover, concentric shells defined by Euclidean distance may not align well with irregular candidate boundaries. Second, our evaluation is limited to one scale and training recipe, leaving scalability to larger models or higher-resolution pretraining unexplored. Third, the model needs more computation than efficiency-oriented SSMs, and the persistent topology memory adds inter-stage communication overhead.

Future work includes improving routing robustness in crowded scenes via more precise object separation and boundary-aware grouping, and studying scalability to larger models, higher-resolution inputs, and larger pretraining datasets (e.g., ImageNet-21K). Finally, reducing the computational and memory overhead of shell routing and persistent topology memory will improve efficiency and applicability.