MEMORY AS THE BACKBONE: A DUAL-STORE ASSOCIATIVE MEMORY ARCHITECTURE FOR VISUAL RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

We investigate whether explicit memory can serve as the organizing principle of a general-purpose visual backbone. Inspired by the interaction between working and long-term memory in human cognition, we introduce **DuSAM** (**Du**al-**S**tore **A**ssociative **M**emory), a hierarchical memory-native visual backbone coupling image-local working memory (WM) with a fixed-capacity persistent semantic memory (PM). WM organizes local visual evidence into compact, spatially anchored states that are updated and transported across stages, then read back to refine dense features. PM consolidates semantic evidence from training images and annotated regions into prototypes, whose retrieved content enriches the same working states. DuSAM achieves competitive performance relative to representative, similarly sized backbones on ImageNet-1K classification, ADE20K semantic segmentation, and COCO object detection and instance segmentation. Ablations show contributions from both stores, with PM particularly benefiting class groups with fewer annotated training pixels. At 2048×2048, the complete DuSAM-based UPerNet segmentor achieves $2.36\times$ the throughput of the evaluated DeiT-S/16 counterpart while using 59.5% less GPU memory. Memory analyses reveal spatially selective WM access and semantic associations between retrieved prototypes and training regions. Class-targeted removal and restoration further demonstrate that category-specific experience can be replenished through state updates without changing network parameters. These results establish explicit memory as a viable organizing principle for visual backbones and provide a foundation for jointly designing contextual computation and addressable semantic memory.

1 INTRODUCTION

Visual backbones have evolved around different ways of exchanging information. Convolutional networks build context through local, weight-shared operations (He et al., 2016). Vision Transformers organize interactions through token-based self-attention (Dosovitskiy et al., 2021), while hierarchical variants extend these interactions to multi-scale representations and dense prediction (Liu et al., 2021; Wang et al., 2021). More recently, visual state-space models use selective scans and state recurrence to propagate information (Zhu et al., 2024; Liu et al., 2024). In this work, we ask: *can a general-purpose visual backbone be built around explicit memory, using memory writes and reads to organize the exchange of visual information?*

Explicit memory organizes information as addressable states that can be stored, updated, and retrieved. Existing approaches have used memory to extend language-model context (Wu et al., 2022), supply cross-image reference features for contrastive learning (Alonso et al., 2021), and preserve evidence over long video sequences (Cheng & Schwing, 2022). For a backbone built around memory, both access and state organization matter. Associative retrieval provides content-based access to stored representations, with modern Hopfield formulations making this operation differentiable (Ramsauer et al., 2021). Such a retrieval rule alone, however, does not determine what to retain or how these states should support a spatial hierarchy.

Visual recognition depends on both the spatial context of a particular scene and semantic regularities shared across scenes. The former must remain grounded in the current image and evolve with

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.

its features; the latter should remain available beyond a single input. We draw inspiration from working-memory accounts in human cognition, where a limited-capacity workspace integrates current information with content retrieved from long-term memory (Baddeley, 2000). This suggests a design principle: maintain current-image context and accumulated semantic experience in separate stores, while bringing them together during visual processing.

Following this principle, we introduce **DuSAM** (**Du**al-**S**tore **A**ssociative **M**emory), a brain-inspired hierarchical visual backbone. Image-local working memory (WM) gathers locally extracted features into compact, spatially anchored states that are transported across stages and updated with new evidence. Persistent semantic memory (PM) consolidates evidence from training images or annotated regions into a fixed-capacity prototype bank shared across the deeper stages. At these stages, image tokens retrieve prototypes through content similarity and predicted semantic addresses. The retrieved content enriches the working states, which are then read back to update dense features. WM is rebuilt for each image, whereas PM persists across training samples and is fixed at inference.

This organization has two useful consequences. Image locations exchange context through compact working states and access a bounded prototype bank, so memory interactions scale linearly with token count for fixed widths and capacities. In addition, part of the semantic experience used in prediction remains accessible as explicit content. Retrieved prototypes can be associated with training-region exemplars, and selected entries can be replenished without weight updates.

We evaluate DuSAM on ImageNet-1K classification, ADE20K semantic segmentation, and COCO object detection and instance segmentation. It achieves 77.8% top-1 accuracy, 42.90% single-scale and 44.12% multi-scale mIoU, and 41.9 box AP and 38.1 mask AP, respectively, demonstrating competitive performance relative to representative backbones of similar size. At 2048×2048, the complete DuSAM-based UPerNet segmentor achieves $2.36\times$ the throughput of the evaluated DeiT-S/16 counterpart while using 59.5% less GPU memory.

Ablations show that both stores contribute to dense prediction, with PM particularly benefiting class groups with fewer annotated training pixels. Working-state analyses reveal spatially differentiated reads, while prototype visualizations expose the visual associations of retrieved semantic content. Category-specific experience can also be replenished without re-optimizing the network: writing five annotated training regions restores prediction regions lost after target-class memory removal. Together, these findings demonstrate the viability of explicit memory as an organizing principle for visual backbones. They provide a foundation for studying visual architectures that jointly organize contextual computation and semantic experience through addressable states.

2 Related Work

Visual backbone architectures. Visual backbone research refines architectures and explores new foundations for representation learning. Within convolutional hierarchies, residual learning, channel recalibration, and modernized blocks improve feature extraction (He et al., 2016; Hu et al., 2018; Liu et al., 2022). Attention-based approaches introduce content-dependent token interactions (Dosovitskiy et al., 2021; Touvron et al., 2021), while pyramids and shifted windows support multi-scale representations (Liu et al., 2021; Wang et al., 2021). MLP and pooling-based alternatives learn competitive representations without attention (Tolstikhin et al., 2021; Yu et al., 2022), while state-space and recurrent models organize context through state updates (Zhu et al., 2024; Liu et al., 2024; Alkin et al., 2025). Despite these computational differences, cross-image semantic knowledge remains primarily in learned parameters. DuSAM represents part of this knowledge as explicit memory content, supporting prototype-level interpretation of the semantic experience used in recognition.

Associative and neural memory. Neural memory can separate stored information from the parameters that process it. Differentiable read–write systems retain and manipulate intermediate information (Graves et al., 2016), while modern Hopfield formulations link content-addressed retrieval to softmax attention (Ramsauer et al., 2021). External stores extend sequence models' access to past representations, while multi-store designs manage retention and forgetting (Wu et al., 2022; Park & Bak, 2024). Visual associative architectures integrate memory-guided attention and local/global retrieval (Sun et al., 2025; Wang et al., 2026). Access mechanisms and storage capacity alone do not determine how transient image context and accumulated semantic experience should cooperate across feature scales. DuSAM addresses this architectural problem through working states that carry

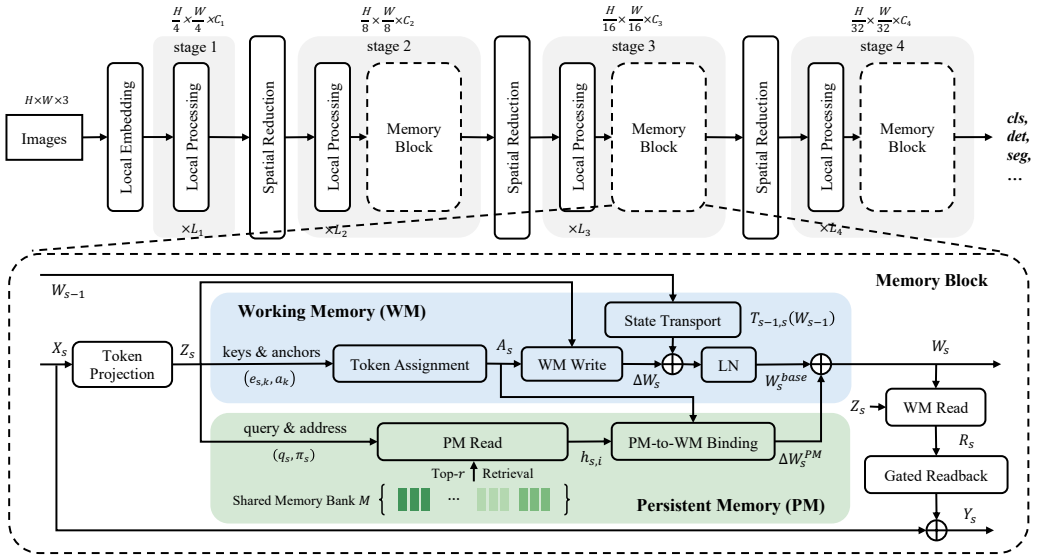

Figure 1: Overview of DuSAM. *Upper:* the four-stage backbone. *Lower:* a Memory Block. WM Write and State Transport construct the workspace, PM-to-WM Binding integrates retrieved semantics, and WM Read with Gated Readback updates dense features.

spatial context across stages and integrate persistent retrieval. This memory-native hierarchy unifies current-image communication and cross-image semantic reuse.

Memory in visual perception. Memory supplies visual evidence beyond a single input. In semantic segmentation, class-wise banks provide cross-image references for contrastive learning (Alonso et al., 2021), while dataset-level memories enrich pixel features with semantic context (Jin et al., 2021). For video, shortand long-term stores preserve evidence across frames (Cheng & Schwing, 2022); object-level reading and consolidation of past activations improve the use of temporal context (Cheng et al., 2024; Balazevic et al., 2024). These methods organize memory around particular prediction tasks or temporal dependencies. DuSAM instead makes memory an organizing principle of a general-purpose backbone for visual recognition.

3 METHODOLOGY

DuSAM organizes visual computation through two complementary stores: Working Memory (WM) gathers context within the current image, while Persistent Memory (PM) retains semantic experience across training images. As shown in Fig. 1, local embedding and successive spatial reductions form a four-stage hierarchy. Stage 1 uses local processing alone; Stages 2–4 additionally contain one Memory Block, with Stages 3–4 accessing a shared PM. See Appendix A.1 for implementation details.

3.1 IMAGE-LOCAL WORKING MEMORY

Spatially anchored state construction. WM enables distant image locations to exchange context through K working states. Fixed anchors a_k at multi-resolution grid centers provide spatial references, while learned assignment keys $e_{s,k}$ guide content collection. A pointwise projection E_s maps the local features $X_s \in \mathbb{R}^{N_s \times C_s}$ to token representations $Z_s = E_s(X_s)$, where $N_s = H_s W_s$. For a token at position p_i, its affinity with state k combines content similarity and spatial proximity:

$$u_{s,ik} = \frac{\langle \widehat{z}_{s,i}, \widehat{e}_{s,k} \rangle}{\tau_w} - \lambda_{\text{sp}} \|p_i - a_k\|_2^2, \tag{1}$$

where $\widehat{\cdot}$ denotes ℓ_2 normalization, and τ_w and λ_{sp} control content temperature and spatial weighting, respectively. To distribute token contributions more evenly across states, we normalize these affinities with a fixed number of Sinkhorn iterations (Cuturi, 2013), obtaining an approximately balanced

assignment $A_s \in \mathbb{R}^{N_s \times K}$. We then form working states by weighted token aggregation:

$$\bar{A}_{s,ik} = \frac{A_{s,ik}}{\max(\sum_j A_{s,jk}, \epsilon_w)}, \qquad \Delta w_{s,k} = \mathrm{LN}_0\left(\sum_i \bar{A}_{s,ik} z_{s,i}\right), \tag{2}$$

where LN_0 denotes channel-wise layer normalization.

Cross-stage update and dense readback. To preserve context across stages, we project previous states to the current width with a learned linear map $T_{s-1,s}$ and add new evidence:

$$W_s^{\mathrm{base}} = \mathrm{LN}(T_{s-1,s}(W_{s-1}) + \Delta W_s), \qquad s \in \{3, 4\}. \tag{3}$$

Stage 2 initializes $W_2 = \Delta W_2$ without transport or PM. At deeper stages, PM enriches the base states into W_s, as described below. Each token then retrieves context by content similarity and incorporates it through a gated residual:

$$B_s = \mathrm{softmax}_k\left(\widehat{Z}_s \widehat{W}_s^{\mathsf{T}} / \tau_w\right), \quad R_s = B_s W_s, \quad Y_s = X_s + \sigma(g_s) \odot O_s(R_s). \tag{4}$$

O_s is an output projection and $\sigma(g_s)$ a learned channel gate. The write weights A_s determine what each state collects, whereas the read weights B_s determine what each token retrieves. The block outputs both refined features Y_s and updated working states W_s.

For fixed K, matching and aggregation through working states cost $\mathcal{O}(N_s K C_s)$, compared with the $\mathcal{O}(N_s^2 C_s)$ pairwise term of global self-attention. This linear scaling supports the high-resolution efficiency gains reported in Section 4.2; full costs are detailed in Appendix A.2.

3.2 Persistent Semantic Memory

Content- and address-guided retrieval. PM retains cross-image semantic experience as P normalized d_m-dimensional prototypes for each of C_{task} categories, forming a bank $M \in \mathbb{R}^{C_{\mathrm{task}} \times P \times d_m}$. Its contents are updated through explicit writes rather than gradients. Both stages share the bank but learn separate readers. To retrieve relevant experience, each projected token $z_{s,i}$ produces a normalized content query $q_{s,i}$ and a predicted semantic-address distribution $\pi_{s,i}$. Content similarity identifies related prototypes, while the address softly favors plausible categories:

$$\ell_{s,i,c,p} = \frac{\langle q_{s,i}, m_{c,p} \rangle + \lambda_a \log \max(\pi_{s,i,c}, \epsilon_a)}{\tau_p}, \tag{5}$$

where λ_a controls the address preference and τ_p the retrieval temperature. We select the top-r valid prototypes across categories to form $\mathcal{T}_{s,i}$ and aggregate them with softmax weights, as in modern Hopfield networks (Ramsauer et al., 2021):

$$\alpha_{s,i,c,p} = \mathrm{softmax}_{(c,p) \in \mathcal{T}_{s,i}}(\ell_{s,i,c,p}), \qquad h_{s,i} = \sum_{(c,p) \in \mathcal{T}_{s,i}} \alpha_{s,i,c,p} m_{c,p}. \tag{6}$$

Persistent-to-working memory binding. To combine current and retrieved evidence, we aggregate token-level retrievals into working states using the same normalized write assignment $\bar{A}_s$:

$$H_{s,k} = \sum_i \bar{A}_{s,ik} V_s(h_{s,i}), \quad \Delta W_s^{\mathrm{PM}} = \sigma(\gamma_s) F_s(H_s), \quad W_s = W_s^{\mathrm{base}} + \Delta W_s^{\mathrm{PM}}. \tag{7}$$

V_s maps retrieved vectors to the stage width, F_s projects the pooled content, and $\sigma(\gamma_s)$ is a learned scalar gate. Tokens then read the enriched states through Eq. equation 4.

Writing and updating PM. PM accumulates training evidence from whole images for classification, connected class-specific regions for semantic segmentation, and annotated instance masks for joint detection and instance segmentation. For each semantic unit, we pool Stage 3 and Stage 4 features from a separate writer path that excludes current PM retrieval. Fixed projections map the pooled features to the d_m-dimensional prototype space. We normalize each vector, average them, and normalize the result to obtain a descriptor d.

Each descriptor is written to a slot of its annotated class c. An unused slot is filled first; otherwise, the most similar prototype is updated by a normalized exponential moving average (EMA):

$$m_{c,p^*} \leftarrow \begin{cases} d, & \text{unused slot,} \\ \mathrm{norm}((1-\eta) m_{c,p^*} + \eta d), & \text{occupied slot,} \end{cases} \tag{8}$$

Table 1: Image classification on the ImageNet-1K. ‡ denotes MLP-Mixer parameter counts excluding the classification head. RC, random crop; RRC, random resized crop; IC, Inception-style crop; HF, horizontal flip; RA, RandAugment; MU, Mixup; CM, CutMix; RE, random erasing.

Method	Architecture	Input	#param.	Data Aug.	Top-1 acc.
ResNet-18	CNN	224^2	11.7M	RRC + HF	69.8
ResNet-34	CNN	224^2	21.8M	RRC + HF	73.3
ResNet-50	CNN	224^2	25.6M	RRC + HF	76.1
ResNet-101	CNN	224^2	44.5M	RRC + HF	77.4
SE-ResNet-50	CNN	224^2	28.1M	RRC + HF	76.7
ConvNeXt-T	CNN	224^2	28.6M	RRC + HF + RA + MU + CM + RE	82.1
ViT-B/16	Transformer	384^2	86.6M	RRC + HF	77.9
ViT-L/16	Transformer	384^2	304.3M	RRC + HF	76.5
ViT-S/16	Transformer	224^2	22.1M	IC + HF + RA + MU	78.8
DeiT-S	Transformer	224^2	22.1M	RRC + HF + RA + MU + CM + RE	79.8
Swin-T	Transformer	224^2	28.3M	RRC + HF + RA + MU + CM + RE	81.2
PVT-T	Transformer	224^2	13.2M	RC + HF + MU + CM + RE	75.1
PVT-S	Transformer	224^2	24.5M	RC + HF + MU + CM + RE	79.8
MLP-Mixer-B/16	MLP	224^2	59M‡	RRC + HF + RA + MU	76.4
MLP-Mixer-S/16	MLP	224^2	18.5M‡	RRC + HF + RA + MU + CM + RE	74.3
Swin-Mixer-T/D24	MLP	256^2	20M	RRC + HF + RA + MU + CM + RE	79.4
PoolFormer-S12	MetaFormer	224^2	11.9M	RRC + HF + RA + MU + CM + RE	77.2
PoolFormer-S24	MetaFormer	224^2	21.4M	RRC + HF + RA + MU + CM + RE	80.3
Vim-T	SSM	224^2	7M	RRC + HF + RA + MU + CM + RE	76.1
Vim-S	SSM	224^2	26M	RRC + HF + RA + MU + CM + RE	80.3
DuSAM (ours)	**Memory**	224^2	20.5M	RRC + HF + RA	77.8

where η controls how much new evidence is incorporated and norm denotes ℓ_2 normalization. We also train the semantic-address heads to predict the annotated categories using auxiliary cross-entropy losses. All PM reads in a training step use the same pre-update bank. New descriptors are committed sequentially only after a successful parameter update. PM is fixed at inference.

4 Experiments

4.1 Image Classification

We evaluate DuSAM on ImageNet-1K (Deng et al., 2009), with training details in Appendix A.1.2. Table 1 compares representative backbones under their reported training and evaluation settings. Even without strong sample-mixing augmentations such as Mixup and CutMix, DuSAM achieves 77.8 top-1 accuracy with only 20.5M parameters. It exceeds ResNet-34 and ResNet-101 by 4.5 and 0.4, respectively, using less than half the parameters of the latter. Across other families, it surpasses MLP-Mixer-B/16 by 1.4 and achieves comparable accuracy to ViT-B/16 with 76.3% fewer parameters. These comparisons support competitive image-level recognition with a compact memory-native backbone.

4.2 Semantic Segmentation

We evaluate DuSAM on ADE20K (Zhou et al., 2017) with UPerNet (Xiao et al., 2018). Detailed settings are provided in Appendix A.1.2. DuSAM achieves $42.90/44.12$ SS/MS mIoU with 19.4M backbone parameters (Table 2). It exceeds ResNet-50 by 1.25 SS mIoU with 25.0% fewer segmentor parameters, and outperforms Swin-Mixer-T/D24 and Vim-S by 1.93 and 2.54, respectively. Compared with ViT-S, it improves SS/MS mIoU by $0.43/0.53$ while using 35.5% less training memory under common FP16 profiling with 16 images per GPU.

Table 2: Semantic segmentation on ADE20K with UPerNet. SS and MS denote single- and multi-scale evaluation; † indicates an additional SimpleFPN is used to produce hierarchical feature maps.

Method	Architecture	#param.		FLOPs	Train Mem.	mIoU	
		Backbone	Total			SS	MS
ResNet-18	CNN	11.2M	41.5M	220.29G	14.29GiB	38.90	39.36
ResNet-34	CNN	21.3M	51.6M	229.97G	14.77GiB	40.08	41.66
ResNet-50	CNN	23.5M	66.5M	238.19G	22.53GiB	41.65	42.60
SE-ResNet-50	CNN	26.0M	69.0M	236.96G	23.13GiB	41.63	41.66
ConvNeXt-T	CNN	27.8M	60.2M	234.94G	18.67GiB	45.00	45.92
ViT-Tiny†	Transformer	5.5M	37.7M	228.31G	24.53GiB	39.28	40.51
ViT-S†	Transformer	21.7M	53.5M	249.35G	29.42GiB	42.47	43.59
DeiT-S†	Transformer	21.7M	53.5M	249.35G	29.42GiB	43.90	44.54
Swin-T	Transformer	27.5M	59.9M	237.19G	19.90GiB	44.02	44.87
PVT-T	Transformer	12.7M	43.2M	222.63G	18.20GiB	37.40	38.57
PVT-S	Transformer	24.0M	54.5M	233.95G	21.99GiB	42.46	43.79
MLP-Mixer-S/16†	MLP	18.0M	52.6M	239.99G	36.62GiB	33.07	33.84
Swin-Mixer-T/D24	MLP	19.6M	52.0M	227.71G	21.07GiB	40.97	42.38
PoolFormer-S12	MetaFormer	11.4M	41.9M	220.36G	16.68GiB	39.68	40.28
PoolFormer-S24	MetaFormer	21.0M	51.4M	228.66G	19.68GiB	41.74	42.85
Vim-T†	SSM	7.1M	35.7M	221.99G	16.30GiB	36.82	39.47
Vim-S†	SSM	25.7M	57.6M	248.33G	24.87GiB	40.36	43.52
DuSAM (ours)	**Memory**	19.4M	49.9M	226.27G	18.98GiB	42.90	44.12

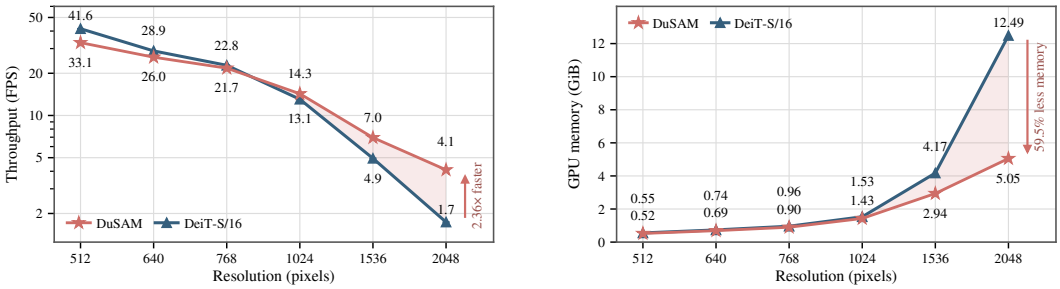

Figure 2: High-resolution inference efficiency of DuSAM and DeiT-S/16 with UPerNet.

We further assess DuSAM's high-resolution efficiency against DeiT-S/16 (Touvron et al., 2021), a representative global-attention Transformer. Both backbones are evaluated as complete UPerNet segmentors on the same NVIDIA H100 80GB GPU, using whole-image FP32 inference with batch size one. DuSAM becomes faster from 1024×1024 onward and, at 2048×2048, achieves $2.36\times$ the throughput with 59.5% less GPU memory (Fig. 2).

4.3 Object Detection and Instance Segmentation

We evaluate DuSAM on COCO 2017 (Lin et al., 2014) using Mask R-CNN (He et al., 2017) with FPN (Lin et al., 2017); detailed settings are provided in Appendix A.1.2. DuSAM achieves 41.9 box AP and 38.1 mask AP with 39.5M detector parameters (Table 3). It exceeds ResNet-101 by $1.9/2.0$ box/mask AP with 37.7% fewer parameters. These gains extend across all three object-size groups (Appendix A.3.1). Across other families, it outperforms DeiT-S, Swin-Mixer-T/D24, and Vim-S by $1.5/1.0$, $2.0/1.2$, and $2.8/2.8$, respectively, while using fewer parameters in each comparison. The gains over DeiT-S also extend to the stricter AP_{75} metric, reaching 2.4 for boxes and 1.6 for masks.

Table 3: Object detection and instance segmentation on COCO val2017 with Mask R-CNN. † indicates an additional SimpleFPN is used to produce hierarchical feature maps.

Method	Architecture	#param.	Bounding box			Instance mask		
			AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
ResNet-18	CNN	31.3M	34.0	54.0	36.8	31.2	51.0	32.9
ResNet-34	CNN	41.2M	37.2	57.5	40.5	33.7	54.3	36.1
ResNet-50	CNN	44.2M	38.0	58.6	41.4	34.4	55.1	36.7
ResNet-101	CNN	63.2M	40.0	60.5	44.0	36.1	57.5	38.6
SE-ResNet-50	CNN	46.7M	38.7	60.9	42.1	35.4	57.4	37.8
ConvNeXt-T	CNN	48.1M	43.6	65.5	48.1	40.3	62.9	43.4
ViT-T †	Transformer	27.9M	35.2	57.5	37.2	33.3	54.5	34.8
ViT-S †	Transformer	43.7M	39.6	62.5	42.5	36.7	59.1	38.7
DeiT-S †	Transformer	43.7M	40.4	63.3	43.5	37.1	59.9	39.0
Swin-T	Transformer	47.8M	42.6	65.0	46.5	39.2	62.1	42.3
PVT-T	Transformer	32.9M	36.7	59.2	39.3	35.1	56.7	37.3
PVT-S	Transformer	44.1M	40.4	62.9	43.8	37.8	60.1	40.3
MLP-Mixer-S/16 †	MLP	41.6M	28.3	49.0	28.9	27.5	46.2	28.6
Swin-Mixer-T/D24	MLP	39.9M	39.9	62.4	43.2	36.9	59.2	39.2
PoolFormer-S12	MetaFormer	31.6M	37.3	59.0	40.1	34.6	55.8	36.9
PoolFormer-S24	MetaFormer	41.0M	40.1	62.2	43.4	37.0	59.1	39.6
Vim-T †	SSM	27.2M	34.2	54.8	36.4	31.4	51.7	32.7
Vim-S †	SSM	46.3M	39.1	60.3	42.0	35.3	56.9	37.0
DuSAM (ours)	**Memory**	39.5M	41.9	64.1	45.9	38.1	60.9	40.6

Table 4: Component ablations on ADE20K. Classes are divided into four approximately equal-sized groups by annotated training-pixel count, from rare (Lowest) to frequent (Highest).

Method	#param.	PM state	Train Mem.	SS mIoU				
				Overall	Lowest	Low	High	Highest
Standard	19.43M	0.77MiB	18.98GiB	**42.90**	**37.75**	**34.81**	**42.96**	**56.43**
w/o WM+PM	17.18M	0.00MiB	18.18GiB	40.45	33.96	30.62	42.00	55.64
w/o WM transport	19.43M	0.77MiB	18.99GiB	42.48	36.50	34.29	43.23	56.29
w/o PM	18.16M	0.00MiB	18.58GiB	41.75	35.87	31.96	43.28	56.30
w/o PM, keep address	18.53M	0.00MiB	18.65GiB	41.55	35.20	31.91	43.38	56.13
Separate PM (2+2)	19.43M	0.77MiB	18.99GiB	42.39	37.30	33.49	43.00	56.14
Separate PM (4+4)	19.43M	1.53MiB	18.99GiB	42.37	37.71	32.91	42.93	56.29

4.4 Ablations and Memory Analysis

We analyze how working and persistent memory structure visual representations on ADE20K.

Memory configuration. Table 4 shows that WM improves over the local-only model, while PM provides its largest gains in the two rarer class groups. Retaining the auxiliary category-prediction heads and losses while removing PM (*w/o PM, keep address*) still lowers accuracy, so additional supervision alone cannot explain PM's benefit. Removing cross-stage transport also reduces accuracy. Separate banks with $2{+}2$ prototypes per class across stages match total capacity, whereas $4{+}4$ doubles it. Both comparisons favor a shared semantic store across stages.

To examine the capacity needed for spatial context, we compare four WM layouts (Fig. 3, *left*). Cumulatively adding 1×1, 2×2, 4×4, and 8×8 anchor grids gives $K \in \{1, 5, 21, 85\}$. The default $K = 21$ achieves the highest SS/MS mIoU with 78.75 KiB of WM storage per image, approximately one quarter of that required by $K = 85$. To assess whether more states also capture

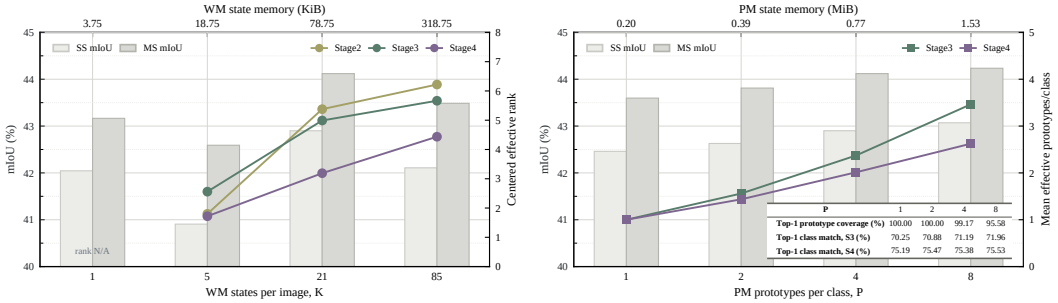

Figure 3: Memory capacity and utilization on ADE20K.

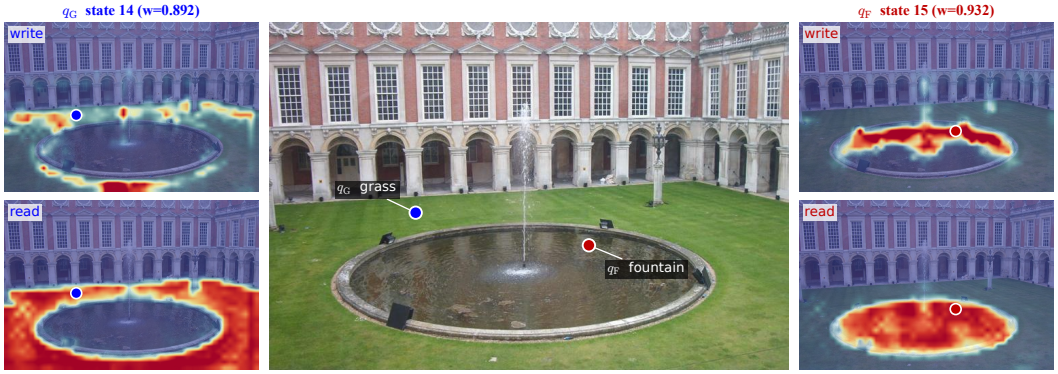

Figure 4: Visualization of WM routing for two queries.

more varied features, we measure centered effective rank. Its modest increase from $K = 21$ to 85 suggests limited additional feature diversity despite the larger workspace.

To illustrate how compact workspace is used, we visualize two queries in Fig. 4, using Stage 3 as an example. For each query, we show highest-read-weight state's current write sources and spatial read weights. The grass and fountain queries access different states (states 14 and 15 with weights 0.892 and 0.932), whose maps emphasize distinct scene regions, illustrating how compact workspace provides different locations with relevant spatial context. With parameters fixed, disabling WM or shuffling its channels causes large accuracy drops. Mean-state and rank-1 replacements reduce mIoU by 5.80 and 3.83, while removing transport costs 4.71 (Table 5). The predictor therefore exploits both distinctions among states and context propagated across stages. Appendix A.3.3 shows how write sources and read locations of the same state evolve through hierarchy.

Table 5: Inference-time memory interventions on ADE20K. Disagr.: pixel-label disagreement with Standard.

Mode	mIoU	ΔmIoU	Disagr.
Standard	**42.90**	**0.00**	**0.00**
WM_off	11.01	−31.89	57.02
WM_shuffle	20.01	−22.89	34.02
WM_mean_state	37.10	−5.80	11.93
WM_rank1_state	39.07	−3.83	9.98
WM_transport_off	38.19	−4.71	11.54
PM_off	42.01	−0.89	5.65
PM_shuffle	41.50	−1.40	6.93
PM_wrong_class	41.94	−0.96	6.62

PM capacity and semantic retrieval. To examine the balance between semantic capacity and storage, we vary $P \in \{1, 2, 4, 8\}$ (Fig. 3, *right*). Accuracy improves modestly with capacity. Although $P = 8$ performs best, we retain $P = 4$ as a compact default with a small accuracy gap and approximately half the PM storage (0.77 versus 1.53 MiB). At $P = 4$, 99.17% of prototypes are selected first at least once across the two access stages. The highest-weight prototype matches the query's class in 71.19% of queries at Stage 3 and 75.38% at Stage 4. Effective prototype counts indicate broader within-class use at Stage 3, while Stage 4 achieves higher class agreement.



Figure 5: Visualization of PM retrieval. Maps show target-class retrieval weights; exemplars are nearest training regions of the top-four prototypes at the marked query. w are not renormalized.

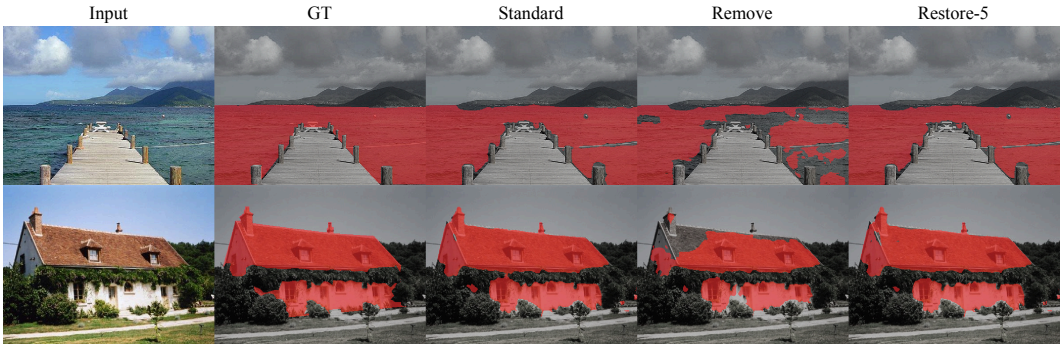

Figure 6: Class-targeted PM removal and restoration with fixed weights. Restore-5 uses five annotated training regions. Red marks the target class on backgrounds desaturated for clarity.

To visualize this semantic access, Fig. 5 shows two examples at Stage 3. Each map sums retrieval weights for the target class across image locations; the accompanying exemplars are nearest training regions of the marked query's top-four prototypes. Together, they show where class-specific memory is used and the visual patterns associated with its retrieved contents. Further examples on ADE20K and COCO appear in Appendix A.3.4, and sensitivity to η, r, and τ_p in Appendix A.3.2.

Class-targeted memory updates. To examine whether stored experience can be revised selectively, we remove each class's PM entries with network parameters fixed. For 90 of 150 output classes on ADE20K, removing their own entries ranks among the five most damaging class-bank removals. The mean absolute change in non-target class IoU is 0.020 , indicating class-directed effects with limited average collateral changes. Whole-bank removal and content perturbations also reduce segmentation accuracy (Table 5). Figure 6 shows that writing five annotated target-class training regions recovers missing predictions after removal in the illustrated cases. This enables category-specific experience to be replenished through state updates rather than network retraining. Additional examples are provided in Appendix A.3.5.

5 Conclusion

We introduced DuSAM, a brain-inspired visual backbone coupling image-local working memory with persistent semantic memory. Experiments show competitive recognition performance, high-resolution efficiency gains over the evaluated DeiT-S/16 baseline, and category-specific memory replenishment without weight updates. These findings demonstrate the viability of explicit memory as an organizing principle for visual backbones, opening a path toward jointly designing visual computation and the storage, retrieval, and updating of semantic experience.

AI USE STATEMENT

In this work, we used generative AI tools only for manuscript editing to improve clarity, readability, and language. We didn't use generative AI tools for formulating research ideas, designing experiments, implementing methods, or analyzing results. We reviewed all AI-assisted work and take responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide implementation and experimental details to support reproducibility. All datasets used in this work are publicly available benchmarks. Appendix A.1 describes the model configuration, training settings, data preprocessing, and evaluation procedures for all tasks. We will publicly release the source code and model checkpoints to facilitate reproduction and further research.

REFERENCES

Benedikt Alkin, Maximilian Beck, Korbinian Pöppel, Sepp Hochreiter, and Johannes Brandstetter. Vision-LSTM: xLSTM as generic vision backbone. In *International Conference on Learning Representations (ICLR)*, 2025.

Iñigo Alonso, Alberto Sabater, David Ferstl, Luis Montesano, and Ana C. Murillo. Semi-supervised semantic segmentation with pixel-level contrastive learning from a class-wise memory bank. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8199–8208, 2021.

Alan Baddeley. The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423, 2000.

Ivana Balazevic, Yuge Shi, Pinelopi Papalampidi, Rahma Chaabouni, Skanda Koppula, and Olivier J. Hénaff. Memory consolidation enables long-context video understanding. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 235, pp. 2527–2542, 2024.

Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 13688, pp. 640–658, 2022.

Ho Kei Cheng, Seoung Wug Oh, Brian L. Price, Joon-Young Lee, and Alexander G. Schwing. Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3151–3161, 2024.

MMEngine Contributors. MMEngine: Openmmlab foundational library for training deep learning models. 2022.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, pp. 2292–2300, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009.

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.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, 2018.

Zhenchao Jin, Tao Gong, Dongdong Yu, Qi Chu, Jian Wang, Changhu Wang, and Jie Shao. Mining contextual information beyond image for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7211–7221, 2021.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8693, pp. 740–755, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

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)*, volume 37, pp. 103031–103063, 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11966–11976, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Sangjun Park and JinYeong Bak. Memoria: Resolving fateful forgetting problem through human-inspired memory architecture. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 235, pp. 39587–39615, 2024.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David P. Kreil, Michael K. Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations (ICLR)*, 2021.

Yuwei Sun, Hideya Ochiai, Zhirong Wu, Stephen Lin, and Ryota Kanai. Associative transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4518–4527, 2025.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 24261–24272, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139, pp. 10347–10357, 2021.

Jianfeng Wang, Amine M'Charrak, Luk Koska, Xiangtao Wang, Daniel Petriceanu, Mykyta Smyrnov, Ruizhi Wang, Michael Bumbar, Luca Pinchetti, and Thomas Lukasiewicz. Vision hopfield memory networks. *arXiv preprint arXiv:2603.25157*, 2026.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–558, 2021.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 11209, pp. 432–448, 2018.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10819, 2022.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5122–5130, 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 *Proceedings of the International Conference on Machine Learning (ICML)*, volume 235, pp. 62429–62442, 2024.

A Appendix

A.1 Implementation Details

A.1.1 Model Details

The four stages use widths $(64, 128, 320, 512)$ and local depths $(3, 3, 8, 3)$. Local embedding comprises two overlapping 3×3, stride-2 projections with output widths 32 and 64, each followed by batch normalization and ReLU. Spatial reduction uses a single 3×3, stride-2 projection. Each local block combines a pre-normalized pointwise residual with a pre-normalized channel-expansion branch using expansion ratio 4, GELU, and channel-wise 3×3 mixing. WM uses $K = 21$ states anchored at the centers of 1×1, 2×2, and 4×4 grids, with $\tau_w = 0.1$, $\lambda_{\text{sp}} = 8$, and seven log-Sinkhorn iterations. PM uses $P = 4$ prototypes per class with $d_m = 320$, $r = 8$, $\tau_p = 0.1$, $\lambda_a = 0.25$, and $\eta = 0.05$. Following layer normalization, the query head projects features to d_m dimensions; the address head uses two linear layers with hidden width d_m and C_{task} outputs. Both memory gates are initialized to output 0.05. Invalid PM slots are excluded from retrieval, and an empty bank leaves the working state unchanged. The semantic-address losses at Stages 3 and 4 each have weight 0.1.

A.1.2 Training and Evaluation

Image classification. We train DuSAM from scratch on ImageNet-1K (Deng et al., 2009), using approximately 1.28 million training images and $50{,}000$ validation images across 1000 categories. Training uses 224×224 inputs, AdamW (Loshchilov & Hutter, 2019), a total batch size of 128, and an initial learning rate of 10^{-3} with cosine decay over 300 epochs. Augmentation consists of random resized cropping, horizontal flipping, and RandAugment with two operations and magnitude nine. The classification loss is cross-entropy with label smoothing 0.1. We report top-1 accuracy on the validation set.

Semantic segmentation. We evaluate on ADE20K (Zhou et al., 2017), with $20{,}210$ training and 2000 validation images from 150 categories. UPerNet (Xiao et al., 2018) uses ImageNet-1K-pretrained weights to initialize the transferable backbone parameters. Training runs for 160k iterations with 512×512 crops, global batch size 16, AdamW, learning rate 6×10^{-5}, and weight decay 0.01. A 1500-iteration linear warmup precedes polynomial decay with power one. The decoder and auxiliary cross-entropy losses have weights 1.0 and 0.4, respectively. We report single-scale (SS) and multi-scale (MS) mIoU; MS combines scales $\{0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$ with horizontal flipping. FLOPs in Table 2 are reported for a single 512×512 input using MMEngine (Contributors, 2022). GPU memory is reported as peak allocated memory.

Object detection and instance segmentation. We use Mask R-CNN (He et al., 2017) with FPN (Lin et al., 2017) on COCO 2017 (Lin et al., 2014) with 80 foreground categories. Training retains $117{,}266$ train2017 images after annotation and minimum-size filtering; evaluation uses all 5000 val2017 images. The backbone is initialized from the ImageNet-1K-pretrained DuSAM weights, with randomly initialized FPN and detection heads. We train for 12 epochs with a total batch size of 16, using AdamW with learning rate 1.5×10^{-4} and weight decay 0.05. A 500-update linear warmup starts at 10^{-3} times the base learning rate, followed by tenfold decays at epochs 8 and 11. Images retain their aspect ratio, with the shorter side resized to 800 pixels and the longer side capped at 1333 pixels; random horizontal flipping with probability 0.5 is the only additional augmentation. Evaluation uses the same resizing rule and single-scale inference without test-time augmentation. We report box and mask AP, AP_{50}, AP_{75}, and size-specific AP_s, AP_m, AP_l.

Baselines. We compare DuSAM with representative CNN, Transformer, MLP, MetaFormer, and SSM backbones across classification, semantic segmentation, and object detection tasks. The baselines include ResNet variants (He et al., 2016), SE-ResNet-50 (Hu et al., 2018), ConvNeXt-T (Liu et al., 2022), ViT-T/S (Dosovitskiy et al., 2021), DeiT-S (Touvron et al., 2021), Swin-T (Liu et al., 2021), PVT-T/S (Wang et al., 2021), MLP-Mixer-S/16 (Tolstikhin et al., 2021), Swin-Mixer-T/D24 (Liu et al., 2021), PoolFormer-S12/S24 (Yu et al., 2022), Vim-T/S (Zhu et al., 2024).

Table 6: COCO val2017 results by object size for the models in Table 3. Subscripts s, m, and l denote small, medium, and large objects. †: additional SimpleFPN.

Method	Architecture	#param.	Bounding box			Instance mask		
			AP_s	AP_m	AP_l	AP_s	AP_m	AP_l
ResNet-18 (He et al., 2016)	CNN	31.3M	19.2	36.7	43.7	14.6	33.2	44.8
ResNet-34 (He et al., 2016)	CNN	41.2M	21.4	40.4	48.5	16.0	36.3	49.2
ResNet-50 (He et al., 2016)	CNN	44.2M	21.9	40.9	49.5	18.3	37.4	47.2
ResNet-101 (He et al., 2016)	CNN	63.4M	22.6	44.0	52.6	18.8	39.7	49.5
SE-ResNet-50 (Hu et al., 2018)	CNN	46.7M	23.4	42.7	50.0	17.1	38.6	51.8
ConvNeXt-T (Liu et al., 2022)	CNN	48.1M	26.5	47.4	56.8	20.4	43.0	58.8
ViT-T † (Dosovitskiy et al., 2021)	Transformer	27.9M	19.9	37.0	48.3	15.2	34.7	51.3
ViT-S † (Dosovitskiy et al., 2021)	Transformer	43.7M	23.9	41.7	54.4	17.8	38.8	55.7
DeiT-S † (Touvron et al., 2021)	Transformer	43.7M	24.2	42.4	55.6	17.8	39.0	56.6
Swin-T (Liu et al., 2021)	Transformer	47.8M	25.8	46.2	56.1	19.2	42.2	57.6
PVT-T (Wang et al., 2021)	Transformer	32.9M	21.6	39.2	49.0	19.5	37.4	48.5
PVT-S (Wang et al., 2021)	Transformer	44.1M	22.9	43.0	55.4	20.4	40.3	53.6
MLP-Mixer-S/16 † (Tolstikhin et al., 2021)	MLP	41.6M	16.6	29.6	37.5	12.9	28.1	41.4
Swin-Mixer-T/D24 (Liu et al., 2021)	MLP	39.9M	23.1	43.0	52.9	17.5	39.1	54.5
PoolFormer-S12 (Yu et al., 2022)	MetaFormer	31.6M	21.4	40.2	48.9	16.0	37.1	50.9
PoolFormer-S24 (Yu et al., 2022)	MetaFormer	41.0M	23.6	43.5	52.4	17.7	39.8	54.0
Vim-T † (Zhu et al., 2024)	SSM	27.2M	18.0	36.1	48.5	14.6	33.3	46.3
Vim-S † (Zhu et al., 2024)	SSM	46.3M	20.9	42.0	55.1	16.9	38.0	51.4
DuSAM (ours)	**Memory**	39.5M	26.2	45.3	55.0	19.8	40.8	55.3

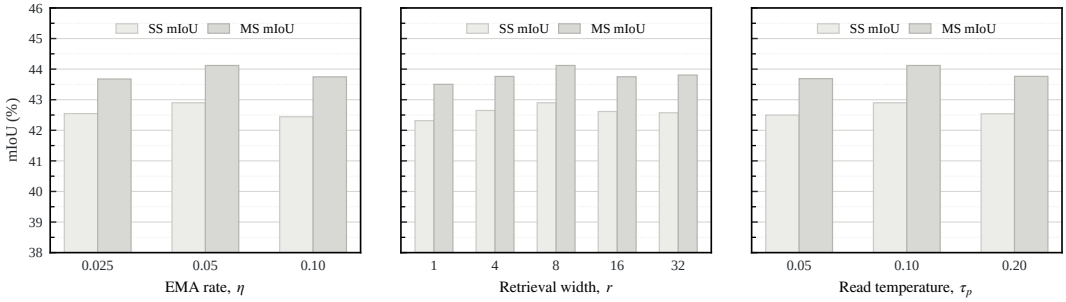

Figure 7: Sensitivity to PM update and retrieval settings. SS/MS mIoU on ADE20K under different EMA rates η, retrieval widths r, and read temperatures τ_p.

A.2 Computational Complexity

We analyze how DuSAM's memory computation scales with image resolution. For one image at stage s, let $N_s = H_s W_s$ denote the number of tokens, C_s the feature width, and K the number of working states. PM contains at most $M_0 = C_{\text{task}} P$ prototypes of dimension d_m, with P prototypes for each of C_{task} categories.

WM matches tokens to working states and aggregates their contents for writing and reading, requiring $\mathcal{O}(N_s K C_s)$ operations. Its J balancing passes add $\mathcal{O}(J N_s K)$. Token and readback projections cost $\mathcal{O}(N_s C_s^2)$, while cross-stage state transport adds $\mathcal{O}(K C_{s-1} C_s)$ when present.

PM scores candidate prototypes at cost $\mathcal{O}(N_s M_0 d_m)$ and aggregates the r selected prototypes at cost $\mathcal{O}(N_s r d_m)$. Query and address mappings, recall projection, and binding to WM together add $\mathcal{O}(N_s C_s d_m + N_s d_m C_{\text{task}} + N_s K C_s + K C_s^2)$. Top-$r$ selection is performed per token over at most M_0 candidates; it reduces value aggregation but not candidate scoring.



Figure 8: Cross-stage WM routing for the same logical state within each example. Columns correspond to Stages 2–4. *Upper:* current write-source density. *Lower:* spatial read weights.

For fixed widths, capacities, and iteration counts, these operations scale linearly with N_s, in contrast to the $\mathcal{O}(N_s^2 C_s)$ pairwise term of global self-attention. This analysis covers memory computation and its projections; local processing, the training-only writer, and task heads contribute additional costs. State storage is independent of image resolution, whereas token features and materialized matching tensors grow with token count.

A.3 Additional Experiments and Analysis

A.3.1 COCO Results by Object Size

Table 6 breaks down the results in Table 3 by object size. DuSAM improves both box and mask AP over ResNet-50 and ResNet-101 across all three size groups. Relative to ResNet-50, box AP increases by 4.3, 4.4, and 5.5 for small, medium, and large objects, respectively. The gains also extend to larger-object delineation: large-object mask AP exceeds ResNet-101 by 5.8. Compared with DeiT-S, DuSAM improves small-object box/mask AP by $2.0/2.0$ and medium-object AP by $2.9/1.8$. Both small-object metrics also exceed ViT-S and Swin-T. All six size-specific metrics improve over Swin-Mixer-T/D24 and PoolFormer-S24, while the gains over Vim-S reach 5.3 box AP and 2.9 mask AP on small objects. These comparisons show that DuSAM's benefits extend to both small-instance localization and large-object mask prediction, complementing the overall AP results.

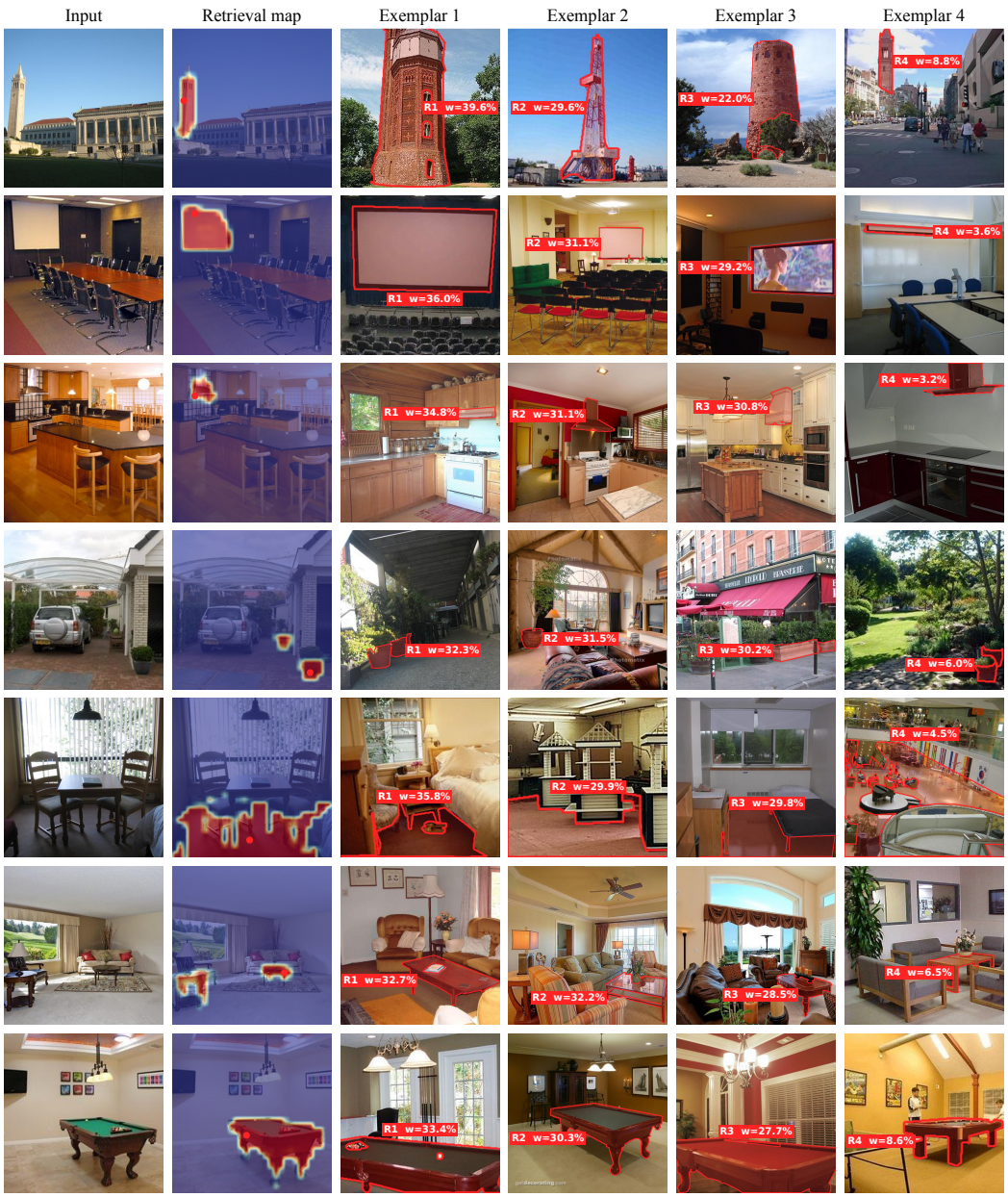

Figure 9: Additional PM retrieval examples. Red outlines mark exemplar regions, w denotes retrieval weights without renormalization.

A.3.2 Update and Retrieval Sensitivity

To assess sensitivity to how PM is updated and queried, we vary the EMA update rate η, retrieval width r, and read temperature τ_p one at a time (Fig. 7). Each configuration is trained for 160k updates with other settings fixed. The defaults, $\eta = 0.05$, $r = 8$, and $\tau_p = 0.1$, achieve the highest SS/MS mIoU among the tested values. Performance varies modestly with the update rate and temperature. Restricting retrieval to one prototype lowers accuracy, while increasing the retrieval width beyond $r = 8$ provides no further gain.

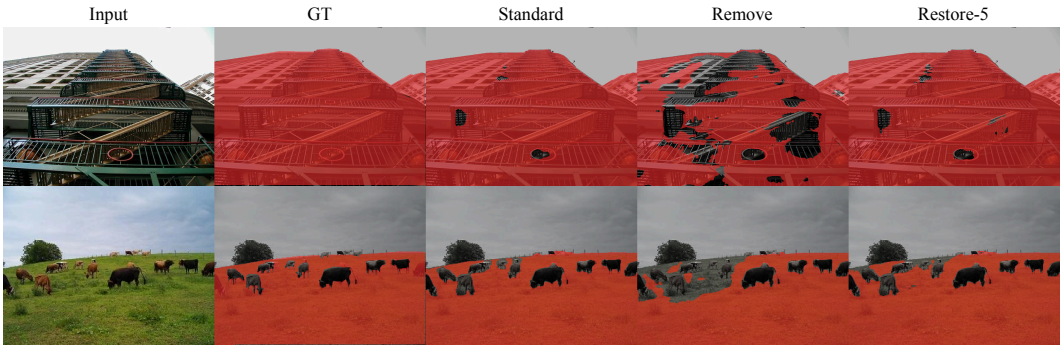

Figure 10: Additional class-targeted PM removal and restoration on ADE20K. Restore-5 uses five annotated training regions with fixed weights.

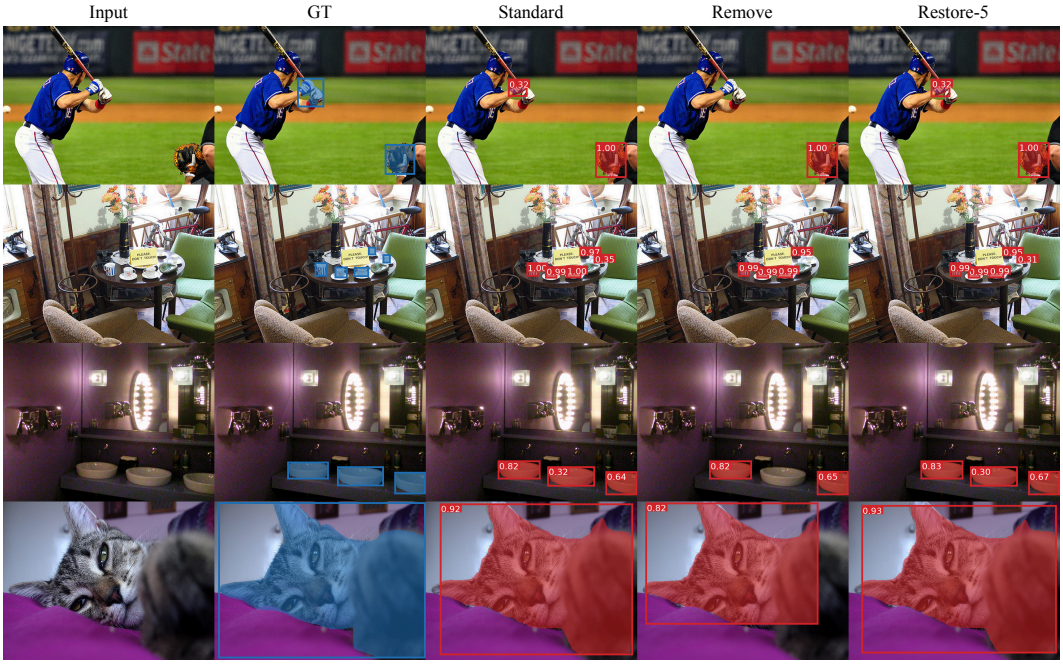

Figure 11: Additional class-targeted PM removal and restoration on COCO. Restore-5 uses five annotated training instances with fixed weights. Blue and red indicate ground truth and predictions.

A.3.3 Cross-Stage Working-Memory Routing

To examine how working memory is reused through the hierarchy, Fig. 8 tracks the same state index k and spatial anchor across Stages 2–4 within each image. Write maps show newly gathered evidence, while read maps show where the updated state is accessed. In these examples, Stage 2 weights are spatially scattered, whereas Stage 3 read maps concentrate on coherent regions such as the fountain, bed, and path. Stage 4 further redistributes access over broader scene areas. These changes illustrate how the workspace adapts both context collection and spatial readback as features evolve, while maintaining correspondence between states across stages. All maps are displayed at image resolution; the coarser feature grids at deeper stages also contribute to their smoother appearance.

A.3.4 Additional PM Retrieval Examples

To examine the visual content associated with PM retrieval across categories, Fig. 9 extends Fig. 5 with additional examples. Heatmaps sum target-class retrieval weights at each image location. Each exemplar is the nearest training region to one of the marked query's four highest-weight prototypes.

The maps concentrate around localized targets such as range hoods and planters, while floor-related retrieval extends across visible ground surfaces. The associated exemplars also vary within each category, including range hoods of different designs and billiard tables with different colors and viewpoints. Together, these examples illustrate spatially targeted access to category-related experience across varied appearances and scene contexts.

A.3.5 Additional Class-Targeted PM Removal and Restoration

Figures 10 and 11 extend the class-targeted updates to additional ADE20K and COCO examples. We remove the target class's PM entries and replenish them using five annotated training regions for ADE20K or five training instances for COCO, keeping network parameters fixed.

On ADE20K, restoration recovers much of the building and grass coverage lost after removal. On COCO, it restores missing detections in the cup and sink scenes and recovers the cat's predicted extent. These examples show that category-specific memory updates can affect both dense semantic predictions and instance-level localization and masks.

A.4 Limitations and Future Work

The current PM is organized by task-specific categories and rebuilt during downstream training, while pretrained backbone weights are transferred. Direct reuse of persistent semantic content across label vocabularies is not explored here. Aligning memory contents and semantic addresses across tasks is a promising direction for retaining this experience during transfer. Our evaluation also focuses on a compact backbone configuration. Extending the study to larger models and more extensive pretraining would help establish how network capacity, memory capacity, and data scale jointly affect performance. These directions would extend the present feasibility study toward more transferable and scalable memory-native visual architectures.