Class Unlearning via Depth-Aware Removal of Forget-Specific Directions

Supplementary Material

A. Additional Analyses and Supplementary Results

A.1. Implementation detail

All experiments were run on a single NVIDIA A100 GPU with 80 GB memory. We used PyTorch with a fixed random seed of 42 for Python, NumPy, and CUDA; CuDNN was set to deterministic mode and benchmarking was disabled. Data loading used 4 workers. For baseline training and retraining, CNN-5, ResNet-18 were optimized with SGD with momentum 0.9. On MNIST, these models were trained for 30 epochs with learning rate 0.01 and weight decay 10^{-4}, without cosine scheduling. On CIFAR-10, CIFAR-100, and Tiny ImageNet, they were trained for 50 epochs with learning rate 0.1 and weight decay 5×10^{-4}, using cosine annealing. ViT models were optimized with AdamW. On MNIST, ViT used learning rate 10^{-3}, weight decay 0.05, and 30 epochs. On CIFAR-10 and CIFAR-500, ViT used learning rate 5×10^{-4}, weight decay 0.05, and 100 epochs. On Tiny ImageNet, ViT used learning rate 3×10^{-4}, weight decay 0.05, and 200 epochs. The training batch size was 128 and the evaluation batch size was 256 for all datasets.

For unlearning baselines, GAU was run for 10 epochs using Adam with learning rate 10^{-4} and loss $\mathcal{L}_{\mathrm{GAU}} = \mathrm{CE}(\mathrm{retain}) - \lambda\,\mathrm{CE}(\mathrm{forget})$, with $\lambda = 0.1$. KDU was run for 10 epochs using Adam with learning rate 10^{-4}, temperature $T = 4.0$, and forget-loss weight $\lambda = 0.5$; it matched the teacher on retain samples via KL divergence and pushed forget samples toward the uniform distribution. DD-FT reinitialized the final classifier layer and fine-tuned the full network on retain data for 10 epochs using Adam with learning rate 5×10^{-4}. RandRelabel was run for 10 epochs using Adam with learning rate 10^{-4}, where forget-class labels were randomly reassigned to retain classes during training. LM was implemented as inference-time logit masking by setting the logits of forget classes to $-\infty$, without any weight updates. SSD was implemented as Selective Synaptic Dampening by estimating the diagonal Fisher information on the full training set and on the forget set. Parameters satisfying $F_{\mathrm{forget}}/F_{\mathrm{train}} > \alpha$ were selected with $\alpha = 25.0$, and selected parameters were dampened as $\theta \leftarrow \theta/\big(1 + \lambda\, F_{\mathrm{forget}}/F_{\mathrm{train}}\big)$ with $\lambda = 1.0$. SSD used no optimizer-based fine-tuning or weight-update epochs.

SalUn was run for 10 epochs using Adam with learning rate 10^{-4}. A saliency mask was first computed from the absolute gradients of the forget-set loss, and the top 50% most salient weights were retained (saliency threshold $\tau = 0.5$). During fine-tuning, forget-class labels were randomly reassigned to retain classes, and gradients on non-salient weights were zeroed so that only salient weights were updated. For all methods, hyperparameters were adjusted for certain datasets and architectures to achieve the best balance between retention and forgetting accuracy.

DAMP details. Class-wise edit means were computed from clean training data. For each layer, forget-class directions were obtained as residuals after projection onto the span of retain-class means, with residual threshold $\epsilon = 10^{-8}$. Multiple forget directions were orthonormalized with QR decomposition. The layer-wise edit strength was set as $\alpha_\ell = \alpha_{\mathrm{probe},\ell} \cdot \alpha_{\mathrm{depth},\ell}$, where $\alpha_{\mathrm{probe},\ell} = \mathrm{clip}(2a_\ell - 1, 0, 1)$ was derived from the accuracy a_ℓ of a logistic-regression probe trained on GAP features, and $\alpha_{\mathrm{depth},\ell} = (\ell + 1)/5$; since $L = 5$. Edits were applied in reverse layer order using the closed-form update $W' = W(I - \alpha_\ell QQ^\top)$. Logistic-regression probes used scikit-learn with `lbfgs`, `class_weight="balanced"`, $C = 1.0$, `max_iter`=1000, and an 80/20 train/test split after feature standardization. In some experiments, α was manually increased by a fixed constant to amplify the effect of weight surgery. For example, for Tiny ImageNet with ViT, all α values were increased by 3.0. The dynamic layer-wise scaling was still preserved, since earlier layers should be edited less than deeper layers.

A.2. Custom CNN-5 architecture.

For all datasets, we used the same lightweight 5-stage convolutional network, denoted *CNN-5*. The network supports variable input channels and numbers of output classes, and is instantiated with 1 input channel for MNIST and 3 input channels for CIFAR-10, CIFAR-100, and Tiny ImageNet. Each stage consists of a convolution, batch normalization, and ReLU activation, with max-pooling applied in the first three stages. The architecture is:

Stage 1: $\mathrm{Conv}(C_{\mathrm{in}}, 64, 3\times3) \to \mathrm{BN} \to \mathrm{ReLU} \to \mathrm{Pool}$
Stage 2: $\mathrm{Conv}(64, 128, 3\times3) \to \mathrm{BN} \to \mathrm{ReLU} \to \mathrm{Pool}$
Stage 3: $\mathrm{Conv}(128, 256, 3\times3) \to \mathrm{BN} \to \mathrm{ReLU} \to \mathrm{Pool}$
Stage 4: $\mathrm{Conv}(256, 256, 3\times3) \to \mathrm{BN} \to \mathrm{ReLU}$
Stage 5: $\mathrm{Conv}(256, 128, 3\times3) \to \mathrm{BN} \to \mathrm{ReLU}$

All convolutions use padding 1, and Pool denotes $\mathrm{MaxPool}(2)$. The classifier head is

$$\mathrm{AdaptiveAvgPool}(1) \to \mathrm{Flatten} \to \mathrm{Linear}(128, n_c),$$

where n_c is the number of classes.

A.3. Layer-wise Selectivity Metric

To quantify the tradeoff between forget-class removal and retain-class preservation, we define a layer-wise *selectivity* score. At layer ℓ, selectivity is computed as

$$\begin{aligned}\text{Selectivity}_\ell = &\left(\text{AUC}^{\text{baseline}}_{\text{forget},\ell} - \text{AUC}^{\text{method}}_{\text{forget},\ell}\right) \\ &- \left(\text{ACC}^{\text{baseline}}_{\text{retain},\ell} - \text{ACC}^{\text{method}}_{\text{retain},\ell}\right).\end{aligned} \tag{14}$$

The first term measures *forget removal*, namely how much the method reduces forget-vs-retain linear separability relative to the baseline model. The second term measures *retain damage*, namely how much retain-only classification accuracy degrades relative to the baseline. Thus, higher selectivity indicates a more desirable operating point: the method removes more linearly accessible forget information while incurring less damage to the retained representation structure.

A.4. Unlearning Results under FGSM and PGD Evaluation

In the main paper, we evaluate unlearning primarily through clean retain and forget accuracy, together with deeper representational analyses. To further assess whether the compared methods remain stable under adversarial perturbations, Table S1 reports results under FGSM and PGD evaluation. Specifically, we report retain accuracy (Retain, ↑) and forget accuracy (Forget, ↓) under both attacks for each method.

These results show that **DAMP** achieves retain accuracy under both attack types that is closer to the retrained network, while keeping forget accuracy at zero, whereas other methods show vulnerability, especially in retain performance under attack.

A.5. Full Representational Dissimilarity Matrices

In the main paper, Fig. 3 reports the difference between the representational dissimilarity matrix (RDM) of each method and the retrained reference model. To provide the underlying representation structure directly, Fig. S6 shows the full RDMs for the compared methods at the same deep layer (layer 5). These plots offer a more complete view of how forget-class structure and retained-class geometry are organized in feature space after unlearning.

A.6. t-SNE Visualization of Deep Representations

To complement the RDM-based analysis, in Fig. S7, we also visualize deep-layer representations using t-SNE [34]. For each method, we extract features from the same layer used in the representational analysis and project them into two dimensions using a shared t-SNE configuration. This visualization provides an intuitive view of how forget, retain, and novel classes are arranged after unlearning.

Table S1. Unlearning results under FGSM and PGD evaluation. We report retain accuracy (Retain, ↑) and forget accuracy (Forget, ↓) for each method. LM performance is close to **DAMP**, but it sets the logits of the forgotten classes to $-\infty$ only during inference, meaning there is no representational unlearning.

Method	FGSM		PGD	
	Retain	Forget	Retain	Forget
Baseline	80.9111	84.0	64.0	68.2
Retrained	80.1778	0.0	66.2111	0.0
GAU [28]	25.0667	0.0	20.0222	0.0
KDU [4]	80.2889	32.4	63.8667	14.6
DD-FT [10]	78.2333	0.0	59.1889	0.0
LM [2]	82.0222	0.0	65.2222	0.0
RandRelabel [27]	80.0556	0.0	61.8667	0.0
SSD [14]	81.5333	0.1	64.6444	0.0
SalUn [13]	80.1889	0.0	63.4889	0.0
DAMP (Ours)	81.9333	0.0	65.0778	0.0

Relative to retraining, an effective unlearning method should reduce the separability of the forget class without unnecessarily disrupting the structure of retained classes. Also, the unlearned networks should treat the novel and forget classes similarly, because the network should no longer retain high-level semantic information about the forget class. Nevertheless, the two should not be expected to behave identically: the model has already learned the low-level features of the forget class, whereas a novel class may contain low-level patterns the model has never encountered. Consequently, the network may preserve more low-level feature knowledge for the forget class than for the novel class. We emphasize that t-SNE is used only as a qualitative diagnostic, since it does not faithfully preserve global geometry. Our primary evidence remains the RDM analysis and the quantitative results in the main paper.

A.7. Effect of Depth-Aware Scaling

A central component of **DAMP** is the depth-aware scaling factor α, which controls the strength of the projection across layers. Fig. S8 presents the retain–forget tradeoff under different settings of α. The results show that the proposed scaling yields a favorable operating point, balancing strong forgetting with preservation of retained knowledge. This analysis supports the choice used in the main experiments.

A.8. Additional Final-Layer Bias Analysis

The paper shows that several baselines suppress forgetting at the output level by strongly shifting the classifier bias of the forget class, rather than genuinely removing forget-class evidence from the internal representation. Fig. S9 extends this analysis by sweeping the final-layer bias and measuring the resulting behavior. The figure further highlights that

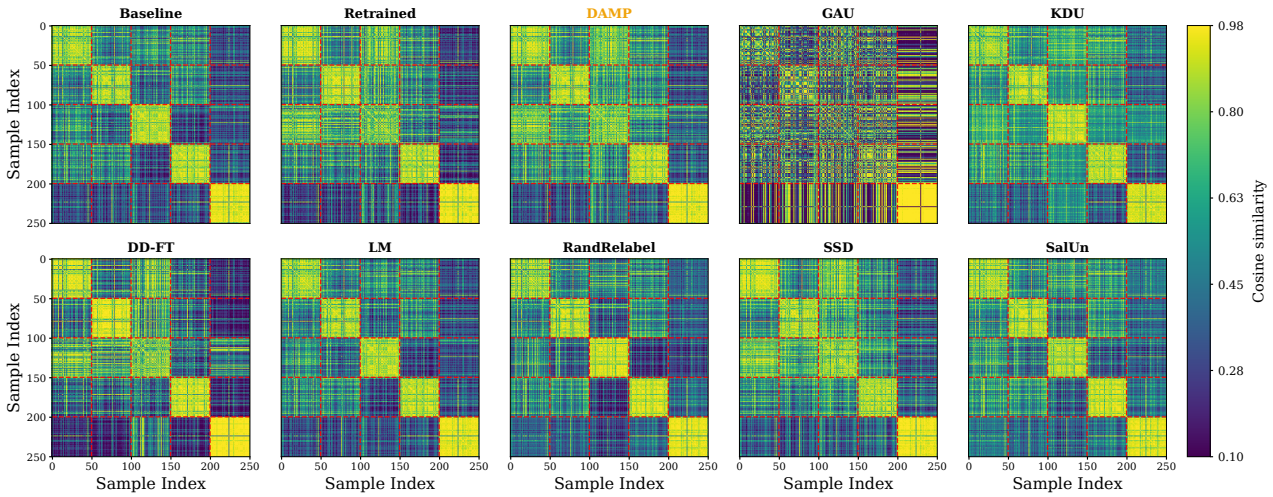

Figure S6. Full representational dissimilarity matrices (RDMs) comparing **DAMP** with the baseline, Gradient Ascent Unlearning (GAU), knowledge-distillation unlearning (KDU), Data Deletion Fine-Tuning (DD-FT), Logit Masking (LM), Random Relabeling (RandRelabel), Selective Synaptic Dampening (SSD), and Saliency Unlearning (SalUn) for a 5-layer CNN (CNN-5) on CIFAR-10; retain classes 4 (Deer), 5 (Dog), 7 (Horse), 8 (Ship) and forget class 6 (Frog).

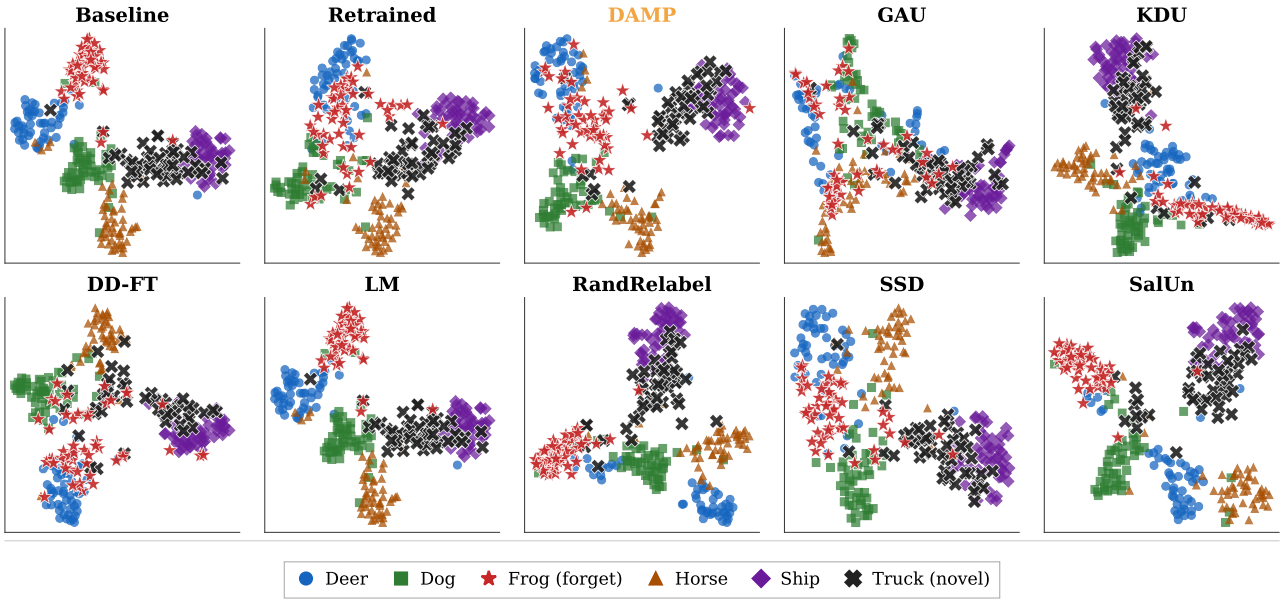

Figure S7. t-SNE visualization of deep-layer (layer 5) features after class unlearning. We project the same representations used in the RDM analysis into two dimensions for qualitative comparison across methods. The forget class is highlighted separately from retained and novel classes. **DAMP** yields a feature layout that more closely resembles retraining while reducing the visual separability of the forget class, whereas several baselines either preserve stronger forget-class clustering or introduce greater distortion among retained classes. Results are comparing **DAMP** with the baseline, Gradient Ascent Unlearning (GAU), knowledge-distillation unlearning (KDU), Data Deletion Fine-Tuning (DD-FT), Logit Masking (LM), Random Relabeling (RandRelabel), Selective Synaptic Dampening (SSD), and Saliency Unlearning (SalUn) for a 5-layer CNN (CNN-5) on CIFAR-10; retain classes 4 (Deer), 5 (Dog), 7 (Horse), 8 (Ship) and forget class 6 (Frog) and novel class 9 (Truck).

output suppression alone can artificially reduce forget accuracy while leaving deeper representations insufficiently unlearned. However, as we show in Fig. S10, **DAMP** does not use this shortcut and achieves unlearning without relying on bias shift. Even when it has the same strong bias shift as the retrained network, the results remain unchanged.

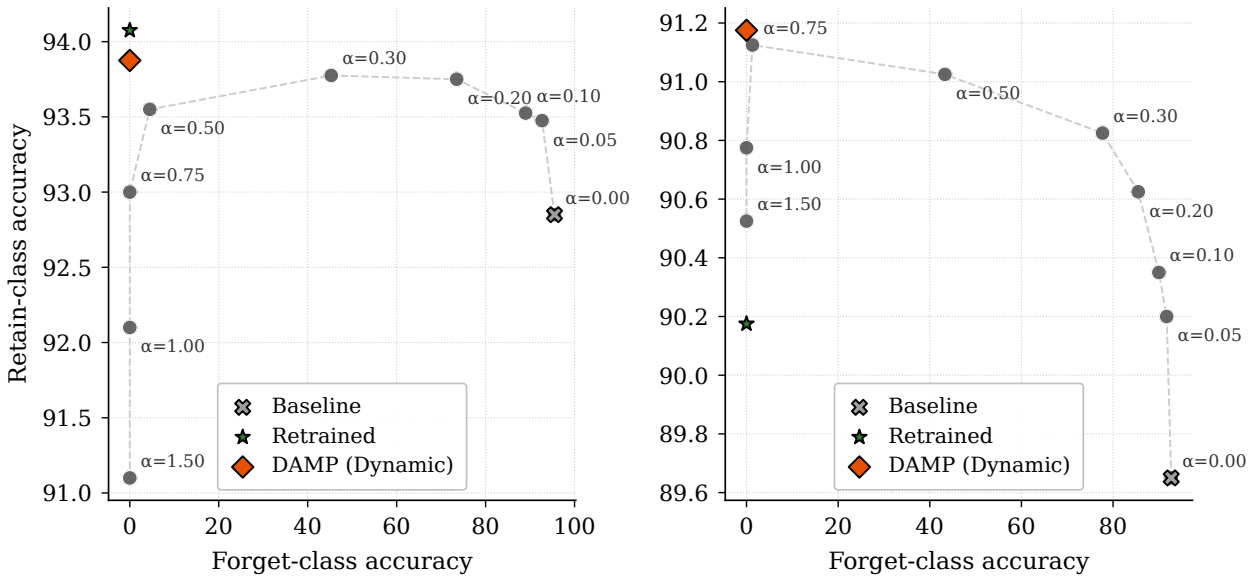

Figure S8. Left: CNN-5 Network. Right: ResNet-18 Network. Effect of the depth-aware scaling factor α on the retain-forget tradeoff. We vary the projection strength and report the resulting balance between retain accuracy and forget accuracy. The selected setting used in **DAMP** achieves strong forgetting while maintaining high retain performance, illustrating the benefit of depth-aware scaling. Results are shown on CIFAR-10; retain classes 4 (Deer), 5 (Dog), 7 (Horse), 8 (Ship) and forget class 6 (Frog).

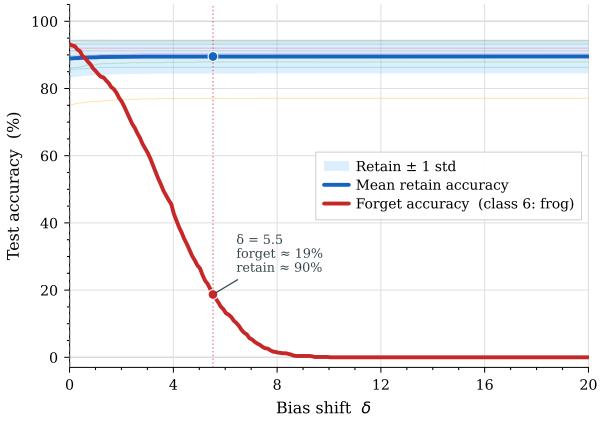

Figure S9. Additional bias-sweep analysis for the final classifier layer. We vary the bias associated with the forget class and measure the resulting change in model behavior. The results illustrate that reducing forget-class predictions can often be achieved through output-level suppression alone, reinforcing the need for representational analyses beyond classifier outputs. Results are shown for the CNN-5 architecture on CIFAR-10.

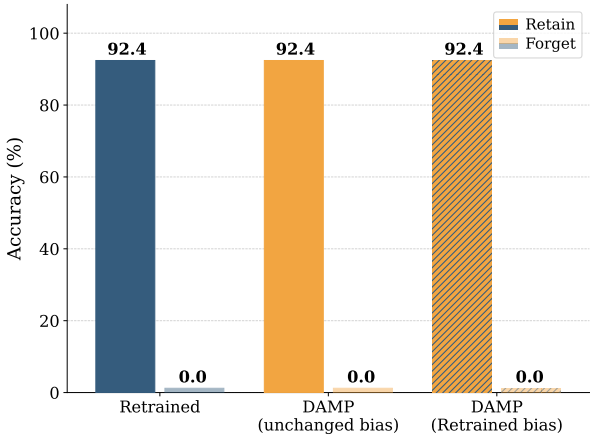

Figure S10. **DAMP** achieves forgetting through weight-space projection and representation unlearning alone. Forcing the last-layer biases to match the delta bias of the retrained network leaves retain and forget accuracy unchanged, demonstrating that bias plays no role in the forgetting mechanism for **DAMP**. Results are shown for the CNN-5 architecture on CIFAR-10.

A.9. Continual Unlearning Results

Table S2 reports continual unlearning performance on CIFAR-10 across sequential class-forgetting rounds. We report retain accuracy (R), newly forget accuracy (NF), all forget accuracy (AF), and the continuous unlearning score (CUS) after each round. These results show that **DAMP** remains competitive with retraining throughout the sequential setting, while several baselines either accumulate residual forget-class information or suffer substantial degradation on retained classes.

Table S2. Continuous unlearning results after each class-forgetting round on CIFAR-10 for the CNN-5 architecture. For each method, we report retain accuracy (R, ↑), newly forget accuracy (NF, ↓), all-forget accuracy (AF, ↓), and continuous unlearning score (CUS, ↑) after forgetting the class indicated at each round.

Method	Metric	Forget class								
		1 (air.)	2 (auto.)	3 (bird)	4 (cat)	5 (deer)	6 (dog)	7 (frog)	8 (horse)	9 (ship)
Retrained	R	89.2	88.7	90.1	94.2	95.0	97.0	97.1	97.5	100.0
	NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	AF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CUS	89.2	88.7	90.1	94.2	95.0	97.0	97.1	97.5	100.0
GAU [28]	R	43.5	19.2	30.3	30.8	37.5	40.9	33.4	49.2	1.5
	NF	0.0	0.0	0.0	0.0	0.0	19.4	32.3	0.0	49.4
	AF	0.0	0.0	0.0	0.0	0.0	3.4	6.6	13.7	17.6
	CUS	43.5	19.2	30.3	30.8	37.5	32.9	22.6	49.2	0.8
KDU [4]	R	87.6	85.8	88.8	91.3	91.3	92.2	92.8	93.9	97.0
	NF	52.3	60.1	47.0	38.3	44.1	24.6	31.0	44.8	59.8
	AF	52.3	56.3	59.6	60.7	60.8	52.5	41.8	46.4	53.3
	CUS	41.8	34.2	47.0	56.3	51.0	69.5	64.0	51.8	39.0
DD-FT [10]	R	88.6	88.5	90.3	93.9	96.1	97.6	98.2	98.0	100.0
	NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	AF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CUS	88.6	88.5	90.3	93.9	96.1	97.6	98.2	98.0	100.0
LM [2]	R	89.5	89.3	90.9	95.1	96.5	98.1	98.3	98.0	100.0
	NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	AF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	CUS	89.5	89.3	90.9	95.1	96.5	98.1	98.3	98.0	100.0
RandRelabel [27]	R	89.5	88.4	89.5	92.6	91.8	93.3	92.8	93.2	95.3
	NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	AF	0.0	44.5	62.8	67.6	69.4	73.0	74.5	76.8	78.0
	CUS	89.5	88.4	89.5	92.6	91.8	93.3	92.8	93.2	95.3
SSD [14]	R	86.4	70.8	72.9	77.4	81.0	77.8	70.9	63.7	56.0
	NF	0.0	42.3	4.2	37.8	58.3	94.1	98.2	85.4	71.4
	AF	0.0	21.2	15.4	20.0	27.9	38.9	47.4	52.2	54.3
	CUS	86.4	40.8	69.6	48.1	33.8	4.59	1.28	9.3	16.0
SalUn [13]	R	89.5	88.5	89.3	92.4	91.9	93.0	93.1	93.9	94.8
	NF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	AF	0.0	45.7	62.4	67.7	69.2	74.0	74.5	76.8	78.1
	CUS	89.5	88.5	89.3	92.4	91.9	93.0	93.1	93.9	94.8
DAMP (Ours)	R	89.5	89.1	90.6	94.9	96.0	98.0	98.2	98.2	100.0
	NF	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0
	AF	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0
	CUS	89.5	89.1	90.6	94.5	96.0	98.0	98.2	98.2	100.0

A.10. Qualitative Segmentation Unlearning Results

We further provide qualitative results for semantic segmentation unlearning in Fig. S11. The figure compares the input image, ground-truth mask, and segmentation outputs produced by the baseline model, retraining, **DAMP**, and competing baselines on representative examples. These examples highlight the visual effect of unlearning beyond aggregate accuracy metrics.

Compared with retraining, successful unlearning should remove evidence of the forgotten class while preserving the spatial structure and semantic coherence of retained re-

Algorithm S1 DAMP: Depth-Aware Model Projection for Class Unlearning

Require: Baseline model f_θ with L layers; forget set $\mathcal{F}$; retain set $\mathcal{R}$; loaders $\{\mathcal{D}_c\}_{c\in\mathcal{F}\cup\mathcal{R}}$
Ensure: Unlearned model $f_{\theta'}$

Initialize $\theta' \leftarrow \theta$
for each class $c \in \mathcal{F} \cup \mathcal{R}$, layer ℓ, minibatch $x \sim \mathcal{D}_c$ **do**
  Compute edit-space vectors $z^{(\ell)}(x)$: apply GAP if $W^{(\ell+1)}$ is linear, else unfold $a^{(\ell)}(x)$ into convolutional patches matched to $W^{(\ell+1)}$
  Accumulate $s_c^{(\ell)} \mathrel{+}= \sum z^{(\ell)}(x),\ n_c^{(\ell)} \mathrel{+}=$ (batch $\times$ locations)
end for
Compute prototypes $\mu_c^{(\ell)} \leftarrow s_c^{(\ell)}/n_c^{(\ell)}$ for all c, ℓ
for each layer $\ell = 1, \ldots, L$ **do**
  Train linear probe on GAP features $\{h_c^{(\ell)}(x)\}$ (forget vs. retain, 80/20 split); obtain accuracy a_ℓ
  $\alpha_\ell \leftarrow \mathrm{clip}(2a_\ell - 1,\ 0,\ 1) \cdot \dfrac{\ell}{L}$
end for
for $\ell = L, L{-}1, \ldots, 1$ **do**
  Form retain matrix $R^{(\ell)} = \left[\mu_r^{(\ell)}\right]_{r\in\mathcal{R}}$
  For each $f \in \mathcal{F}$, compute residual $d_f^{(\ell)} = \mu_f^{(\ell)} - R^{(\ell)} R^{(\ell)\dagger} \mu_f^{(\ell)}$
  Collect $Q^{(\ell)} = \{d_f^{(\ell)}/\|d_f^{(\ell)}\| : \|d_f^{(\ell)}\| > \varepsilon,\ f \in \mathcal{F}\}$
  if $Q^{(\ell)} \neq \emptyset$ **then**
    $\widetilde{Q}^{(\ell)} \leftarrow \mathrm{QR}([q_f^{(\ell)}]_{f\in\mathcal{F}})$
    $W^{(\ell+1)} \leftarrow W^{(\ell+1)}\left(I - \alpha_\ell\, \widetilde{Q}^{(\ell)} \widetilde{Q}^{(\ell)\top}\right)$ {flatten conv weights before, reshape after}
  end if
end for
return $f_{\theta'}$

gions. **DAMP** produces outputs that are visually closer to retraining, whereas several baselines either retain residual forgotten-class predictions or induce larger distortions in surrounding retained regions.

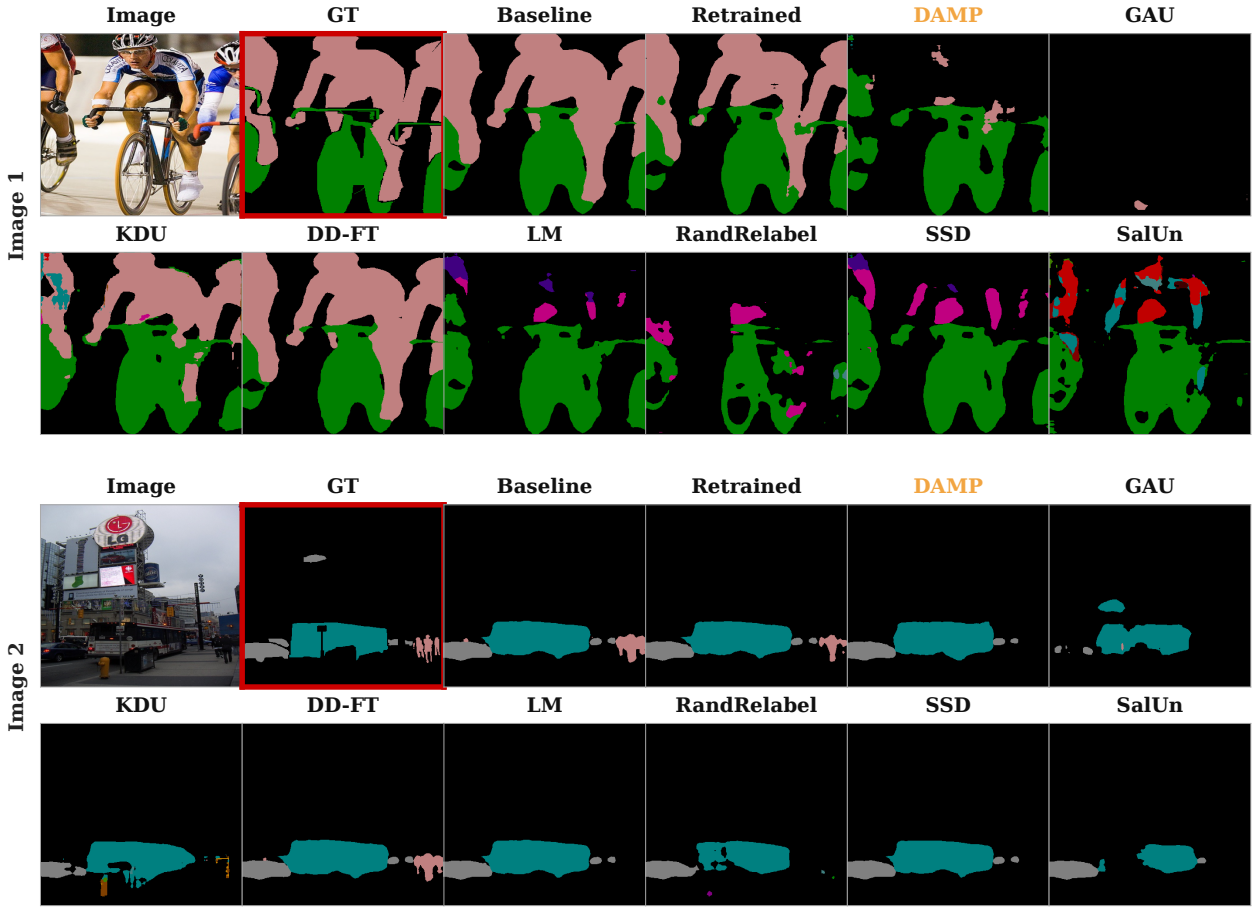

Figure S11. Qualitative results for semantic segmentation unlearning on two representative examples trying to forget Human category. Dataset is Pascal VOC 2012 semantic segmentation and the network is DeepLabV3 with a ResNet-50 backbone. We show the input image, ground-truth mask (GT), and predictions from the baseline, retraining, Gradient Ascent Unlearning (GAU), knowledge-distillation unlearning (KDU), Data Deletion Fine-Tuning (DD-FT), Logit Masking (LM), Random Relabeling (RandRelabel), Selective Synaptic Dampening (SSD), and Saliency Unlearning (SalUn). **DAMP** more closely follows the retrained model by suppressing the forgotten regions while preserving the structure of retained classes, whereas several baselines either retain residual forgotten-class predictions or introduce greater degradation in the remaining segmentation regions.