# Decoupled Distillation to Erase: A General Unlearning Method for Any Class-centric Tasks

## Supplementary Material

This appendix provides supplementary materials that could not be included in the main paper due to space limitations. In Sec. A, we present the reformulation of the loss function Eq. (5). Sec. B provides details on the implementation, baselines, and evaluation metrics. Sec. C provides additional performance comparisons of various methods across different datasets and models. Finally, Sec. D demonstrates the implementation details, quantitative results, and qualitative results of applying our method to downstream tasks, including face recognition, backdoor defense, and semantic segmentation.

## A. Reformulation

In this section, we reformulate the loss function Eq. (5) in the main paper. We prove that, with an appropriately designed mask $\mathrm{Mask}'_u(\cdot)$, the masking and softmax steps can be interchanged. After reordering, the resulting softmax vector satisfies the condition that the sum of its elements equals 1, thereby obviating the need for additional normalization.

We define the mask function as $\mathrm{Mask}'_u(\mathbf{v}) = \mathbf{v} + \mathbf{m}^u$, where the vector $\mathbf{m}^u \in \mathbb{R}^K$ is defined as:

$$\mathbf{m}_i^u = \begin{cases} 0 & \text{if } i \neq u, \\ -\infty & \text{if } i = u. \end{cases}$$

Next, we prove that interchanging the mask and softmax steps with an appropriately designed mask function results in proportional outcomes, *i.e.*,

$$\mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z})) \propto \mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z}))$$

This is beacuse, for $i \neq u$, we have:

$$\begin{aligned} \mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z}))_i &= \frac{e^{z_i}}{\sum_i e^{z_i}} \\ &= \frac{e^{z_i}}{\sum_{i\neq u} e^{z_i}} \times \frac{\sum_{i\neq u} e^{z_i}}{\sum_i e^{z_i}}. \end{aligned}$$

Meanwhile,

$$\begin{aligned} \mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z}))_i &= \frac{e^{z_i}}{e^{-\infty} + \sum_{i\neq u} e^{z_i}} \\ &= \frac{e^{z_i}}{\sum_{i\neq u} e^{z_i}}. \end{aligned}$$

For $i = u$, both element yield 0:

$$\mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z}))_i = \mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z}))_i = 0$$

Table S1. Optimization settings for image classification.

| Config | Value |
|---|---|
| Optimizer | SGD |
| Weight Decay | 5e-4 |
| Momentum | 0.9 |
| Learning Rate Scheduler | Step LR Scheduler |
| Learning Rate Step | 40 |
| Learning Rate Gamma | 0.1 |

Thus, for any $i$, combining the two cases above, we obtain the following relationship:

$$\mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z}))_i = \mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z}))_i \times \frac{\sum_{i\neq u} e^{z_i}}{\sum_i e^{z_i}}.$$

Since the elements of both vectors are proportional by a factor, normalizing these vectors results in identical outputs. Given that the softmax operation ensures the output vector is already normalized (*i.e.*, its elements sum is 1), we conclude:

$$\mathrm{Normalize}(\mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z}))) = \mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z})). \tag{S1}$$

Thus, we can interchange the mask and softmax steps via Eq. (S1), thereby simplifying Eq. (5) in the main paper by removing the need for normalization. The final loss function is expressed as:

$$\begin{aligned} \mathcal{L} &= \mathrm{KL}\left(\mathrm{Normalize}(\mathrm{Mask}_u(\mathrm{Softmax}(\mathbf{z}))) \,\|\, \mathbf{q}\right) \\ &= \mathrm{KL}\left(\mathrm{Softmax}(\mathrm{Mask}'_u(\mathbf{z})) \,\|\, \mathbf{q}\right). \end{aligned}$$

## B. Experimental Details

### B.1. Implementation Details

Our implementation is based on Python 3.8 and PyTorch 1.13. All experiments are conducted on a system equipped with NVIDIA RTX 4090 GPU and Intel Xeon Gold 6226R CPU. For the image classification unlearning task, we first train the original model from scratch. The optimizer settings are detailed in Tab. S1. The training configurations for different models and datasets during pretraining are provided in Tab. S2. The retrain model follows the same training configurations but is trained exclusively on $\mathcal{D}_\mathrm{r}$, without access to $\mathcal{D}_\mathrm{f}$.

### B.2. Baseline Details

Recalling that we compare the unlearning performance of different methods, under the two imposed constraints: **(i)**

Table S2. Pre-training setups for different datasets and models.

| Different Datasets | CIFAR-10<br>ResNet-18 | CIFAR-100<br>ResNet-18 | Tiny ImageNet<br>ResNet-18 |
|---|---|---|---|
| Pretrain Epochs | 150 | 150 | 150 |
| Pretrain LR | 0.1 | 0.1 | 0.1 |
| Batch Size | 128 | 128 | 64 |
| Different Models | VGG-16<br>CIFAR-10 | Swin-T<br>CIFAR-10 | ViT-S<br>CIFAR-10 |
| Pretrain Epochs | 80 | 100 | 150 |
| Pretrain LR | 0.01 | 0.01 | 0.1 |
| Batch Size | 128 | 64 | 128 |

No access to the remaining data; **(ii)** No intervention during the pre-training phase. Under these constraints, certain methods become infeasible. Finetune, Fisher Forget [8], SSD [7], and SISA [1] fail to work due to their reliance on remaining data or intervention in their algorithms. Bad Teacher [4], Saliency Unlearn[6], SCRUB [10], and UNSIR[16] are unable to compute regularization loss or perform additional repair phases.

Nevertheless, to better compare the performance of different methods, we select several well-known approaches and evaluate them under scenarios without the aforementioned constraints. The selected methods include Finetune, Fisher Forget, Bad Teacher, and Saliency Unlearn, with their results marked in gray in the main paper. Specifically, for Bad Teacher and Saliency Unlearn, we also remove the regularization loss that depend on remaining data and test their performance under the imposed constraints. These results are marked in black.

For training-based unlearning methods, we perform unlearning for 20 epochs and search for the optimal learning rate within the range of $[10^{-7}, 10^{-2}]$. For parameter-scrubbing unlearning methods, we search for the hyperparameter $\alpha$ in the range of $[10^{-8}, 10^{-5}]$ for Fisher Forget, and in the range of $[10^{-2}, 10^{2}]$ for Influential Unlearn. All other hyperparameter settings follow the configurations in the responding original papers.

### B.3. Mertic Details

Following prior work, we use membership inference attacks (MIA) success rate as a metric to evaluate the forgetting performance of the unlearn model [6]. The MIA implementation is based on a prediction confidence-based attack method [15]. To construct the dataset for training the MIA predictor, we sample a balanced binary classification dataset from $\mathcal{D}_\mathrm{r}$ and $\mathcal{D}_\mathrm{rt}$. The input consists of the confidence scores predicted by the model for the images, while the corresponding labels indicate whether each image originates from $\mathcal{D}_\mathrm{r}$ or $\mathcal{D}_\mathrm{rt}$.

During the evaluation phase, the confidence scores predicted by $f_\theta$ on $\mathcal{D}_\mathrm{f}$ are used as input to the MIA predictor. The classification results of the MIA predictor are then used to assess how many samples in the forgetting dataset are still memorized by the model after unlearning. The MIA metric is defined as:

$$\mathrm{MIA} = \frac{\mathrm{TP}}{|\mathcal{D}_\mathrm{f}|}$$

where it measures the proportion of samples in $\mathcal{D}_\mathrm{f}$ that the MIA predictor classifies as still being memorized by the model. In other words, it represents the proportion of samples in $\mathcal{D}_\mathrm{f}$ that were not successfully forgotten.

Note that our MIA evaluation is slightly different from Saliency Unlearn's [6]. We measure the proportion of samples in $\mathcal{D}_\mathrm{f}$ that have not been successfully forgotten, whereas Saliency Unlearn measures the proportion of successfully forgotten samples. We adopt this approach to maintain consistency with metrics such as $\mathrm{Acc}_\mathrm{f}$ and $\mathrm{Acc}_\mathrm{ft}$, where lower values indicate better forgetting performance.

Table S3. Performance comparison of single-class forgetting across different unlearning methods on Tiny ImageNet dataset. **bold** indicates the single best result among methods. The same notation applies hereafter.

| Method | $\mathrm{Acc}_\mathrm{f}\downarrow$ | $\mathrm{Acc}_\mathrm{r}\uparrow$ | $\mathrm{Acc}_\mathrm{ft}\downarrow$ | $\mathrm{Acc}_\mathrm{rt}\uparrow$ | H-Mean ↑ | MIA ↓ |
|---|---|---|---|---|---|---|
| Original Model | 100 | 99.98 | 66.0 | 64.15 | - | 99.4 |
| Retrain Model | 0 | 99.98 | 0 | 64.42 | 65.20 | 0 |
| Random Label | 3.2 | 97.58 | 2.0 | 58.84 | 61.31 | 0.2 |
| Negative Gradient | 7.8 | 95.36 | 2.0 | 56.24 | 59.87 | 3.0 |
| Boundary Shrink | 5.2 | 97.49 | 2.0 | 57.85 | 60.77 | 0.6 |
| Boundary Expand | 8.6 | 98.89 | 2.0 | 58.88 | 61.33 | 0 |
| Influence Unlearn | 12.2 | 99.36 | 2 | 60.30 | 62.09 | 1.6 |
| Learn to Unlearn | 1.0 | 85.57 | 0 | 50.73 | 57.37 | 0.2 |
| Bad Teacher | 5.8 | 98.32 | 2.0 | 59.02 | 61.41 | 0.2 |
| Saliency Unlearn | 6.6 | 98.80 | 2.0 | 59.67 | 61.76 | 0.6 |
| Ours | **0.4** | **99.94** | 0 | **62.15** | **64.02** | 0 |

## C. Experimental Results

### C.1. Performance on Different Datasets

In the main paper, we present the results of single-class forgetting on CIFAR-10 and CIFAR-100. Tab. S3 further presents the performance on Tiny ImageNet across various methods for the single-class forgetting task. While most methods achieve some degree of forgetting, several exhibit poor retention of knowledge for the remaining classes. For example, Negative Gradient, Boundary Shrink, Boundary Expand, and Learn to Unlearn show a decline of around 5% in $\mathrm{Acc}_\mathrm{rt}$ compared to the original model.

Notably, despite Learn to Unlearn performing well on CIFAR-10 and CIFAR-100, its performance drops significantly on Tiny ImageNet, falling from third place on CIFAR-10 to the lowest rank on Tiny ImageNet. This highlights its fragility and inconsistency across datasets.

In contrast, our method consistently demonstrates strong performance, achieving complete forgetting in $\mathrm{Acc}_\mathrm{ft}$ while maintaining excellent $\mathrm{Acc}_\mathrm{rt}$. Among all methods, ours is one of only two to surpass 60% in $\mathrm{Acc}_\mathrm{rt}$, outperforming

Table S4. Performance comparison of single-class forgetting on VGG-16, across different unlearning methods on CIFAR-10.

| Method | $\text{Acc}_\text{f}$ ↓ | $\text{Acc}_\text{r}$ ↑ | $\text{Acc}_\text{ft}$ ↓ | $\text{Acc}_\text{rt}$ ↑ | H-Mean ↑ | MIA ↓ |
|---|---|---|---|---|---|---|
| Original Model | 99.94 | 99.94 | 91.20 | 92.02 | - | 99.88 |
| Retrain Model | 0 | 99.71 | 0 | 92.49 | 91.84 | 0 |
| Random Label | 0 | 96.35 | 0 | 88.16 | 89.65 | 0 |
| Negative Gradient | 0.36 | 92.11 | 0.20 | 83.77 | 87.24 | 0.20 |
| Boundary Shrink | 6.68 | 91.15 | 6.80 | 82.79 | 83.59 | 2.84 |
| Boundary Expand | 2.82 | 84.95 | 2.10 | 77.27 | 82.76 | 1.38 |
| Influence Unlearn | 1.86 | 93.17 | 1.60 | 84.74 | 87.10 | 1.30 |
| Learn to Unlearn | 3.28 | 94.10 | 3.00 | 85.56 | 86.86 | 1.96 |
| Bad Teacher | 7.14 | 90.95 | 7.20 | 82.90 | 83.45 | 0 |
| Saliency Unlearn | 11.24 | 73.37 | 10.90 | 69.09 | 74.27 | 0 |
| Ours | 0 | **99.66** | 0 | **92.08** | **91.64** | 0 |

Table S5. Performance comparison of single-class forgetting on Swin-T, across different unlearning methods on CIFAR-10.

| Method | $\text{Acc}_\text{f}$ ↓ | $\text{Acc}_\text{r}$ ↑ | $\text{Acc}_\text{ft}$ ↓ | $\text{Acc}_\text{rt}$ ↑ | H-Mean ↑ | MIA ↓ |
|---|---|---|---|---|---|---|
| Original Model | 85.00 | 86.58 | 82.70 | 82.68 | - | 82.94 |
| Retrain Model | 0 | 86.41 | 0 | 82.77 | 82.73 | 0 |
| Random Label | 3.68 | 74.86 | 0.90 | 70.42 | 75.68 | 4.68 |
| Negative Gradient | 0.46 | 74.60 | 0.10 | 71.98 | 76.93 | 0.22 |
| Boundary Shrink | 5.54 | 62.94 | 2.50 | 58.18 | 67.44 | 11.36 |
| Boundary Expand | 2.58 | 70.36 | 1.00 | 65.67 | 72.81 | 3.26 |
| Influence Unlearn | 0.04 | 76.02 | 0 | 72.99 | 77.54 | 0 |
| Learn to Unlearn | 0.52 | 79.55 | 0.40 | 76.50 | 79.29 | 0.14 |
| Bad Teacher | 8.96 | 74.77 | 6.30 | 68.86 | 72.43 | 7.24 |
| Saliency Unlearn | 11.16 | 76.88 | 8.00 | 72.40 | 73.53 | 12.40 |
| Ours | **0** | **86.68** | 0 | **83.61** | **83.15** | 0 |

Table S6. Performance comparison of single-class forgetting on ViT-S, across different unlearning methods on CIFAR-10.

| Method | $\text{Acc}_\text{f}$ ↓ | $\text{Acc}_\text{r}$ ↑ | $\text{Acc}_\text{ft}$ ↓ | $\text{Acc}_\text{rt}$ ↑ | H-Mean ↑ | MIA ↓ |
|---|---|---|---|---|---|---|
| Original Model | 90.50 | 90.57 | 72.70 | 75.73 | - | 84.60 |
| Retrain Model | 0 | 92.42 | 0 | 76.44 | 74.52 | 0 |
| Random Label | 1.32 | 77.38 | 1.80 | 68.51 | 69.68 | 2.24 |
| Negative Gradient | 0.38 | 85.25 | 0.50 | 73.34 | 72.77 | 0.34 |
| Boundary Shrink | 1.84 | 73.16 | 1.70 | 65.59 | 68.19 | 1.66 |
| Boundary Expand | 8.02 | 75.86 | 7.80 | 67.42 | 66.14 | 9.24 |
| Influence Unlearn | 0.98 | 87.09 | 0.90 | 75.20 | 73.46 | 0.78 |
| Learn to Unlearn | 0.46 | 85.11 | 0.30 | 73.20 | 72.80 | 0.36 |
| Bad Teacher | 6.02 | 74.66 | 5.40 | 67.27 | 67.28 | 7.32 |
| Saliency Unlearn | 6.22 | 77.54 | 5.90 | 68.46 | 67.62 | 7.14 |
| Ours | **0** | **90.74** | **0** | **77.21** | **74.89** | **0** |

the second-best method by nearly 2%. Our method outperforms all other approaches across all performance metrics on Tiny ImageNet. This demonstrates the superior balance of our approach between effective forgetting and retention of unrelated knowledge across various datasets.

### C.2. Performance on Different Models

This part compares the performance of various methods across different models. In Tabs. S4 to S6, we present the forgetting performance of different methods on three distinct models, apart from ResNet-18 discussed in the main paper: VGG-16, Swin-T, and ViT-S.

Remarkably, our method achieves the best results across all 18 metrics on all three models. Among these, it achieves the sole best performance on 14 metrics. For the overall performance metric H-Mean, our method outperforms the second-best by margins of 1.99%, 3.86%, and 1.43% on VGG-16, Swin-T, and ViT-S, respectively. Additionally, on $\text{Acc}_\text{rt}$, which measures the preservation of knowledge for unrelated classes, our method leads by 3.92%, 7.11%, and 1.99% over the second-best on the respective models. These results highlight the strong generalization and consistent performance of our method across different architectures.

In contrast, some other methods exhibit significant performance fluctuations, reflecting their fragility when applied to different models. For example, Boundary Shrink shows a gap of 7.74% in H-Mean compared to the retrain model on ResNet-18, which expands to 15.29% on Swin-T, nearly doubling the performance gap. On the other hand, Influence Unlearn demonstrates relatively stable performance, achieving three second-place and one third-place rankings. However, its H-Mean scores are 1.22% to 5.61% lower than those of our method across all models, further emphasizing our superior performance.

## D. Application to Downstream Tasks

### D.1. Face Recognition with Unlearning

**Implementation Details.** Face recognition experiments are conducted using the VGGFace2 [2] dataset and the ResNet18 model. To prepare the dataset, we first filter individuals with more than 500 images, and randomly select 110 of them. For each individual, we allocate 400 images for the training set and 100 images for the test set.

The model is pretrained using 100 randomly selected individuals from the training set for 80 epochs to obtain a pretrained model. Subsequently, we fine-tune the model for 40 epochs using 10 other individuals. During the unlearning phase, one of these 10 individuals is randomly chosen, and the unlearning process is performed for 20 epochs. To ensure fairness, all methods are evaluated using the same individuals for the pretraining, fine-tuning, and unlearning stages. The batch size is set to 64 for all training stages, and the learning rate is fixed at 0.01 during both the pretraining and fine-tuning phases.

**Qualitative Results.** Tab. S7 presents the performance of various methods on the face recognition task. Our method achieves the best results across all metrics. In terms of overall performance, measured by H-Mean, our method demonstrates a minimal gap of just 0.61% compared to the retrain model, which is considered as the upper bound.

### D.2. Backdoor Defense with Unlearning

**Implementation Details.** Data poisoning is a common backdoor attack method, where a small fraction of the train-

Table S7. Performance comparison of face recognition task on VGG-Face dataset.

| Method | $\text{Acc}_\text{f}$ ↓ | $\text{Acc}_\text{r}$ ↑ | $\text{Acc}_\text{ft}$ ↓ | $\text{Acc}_\text{rt}$ ↑ | H-Mean ↑ | MIA ↓ |
|---|---|---|---|---|---|---|
| Original Model | 100 | 100 | 97.00 | 99.22 | - | 100 |
| Retrain Model | 0 | 100 | 0 | 98.89 | 97.94 | 0 |
| Random Label | 0 | 99.97 | 0 | 97.22 | 97.11 | 0 |
| Negative Gradient | 1.15 | 98.25 | 1.00 | 93.67 | 94.82 | 1.00 |
| Boundary Shrink | 0 | 100 | 0 | 97.33 | 97.16 | 0 |
| Boundary Expand | 4.25 | 99.94 | 3.00 | 97.33 | 95.64 | 0 |
| Influence Unlearn | 0 | 99.97 | 0 | 96.11 | 96.55 | 0 |
| Learn to Unlearn | 2.25 | 96.94 | 1.00 | 92.33 | 94.13 | 1.25 |
| Bad Teacher | 2.75 | 98.50 | 3.00 | 95.56 | 94.77 | 0 |
| Saliency Unlearn | 18.50 | 100 | 11.00 | 97.67 | 91.46 | 3.50 |
| Ours | 0 | 100 | 0 | 97.67 | **97.33** | 0 |

ing data is maliciously altered with triggers and incorrect labels. Once trained on such poisoned data, the backdoored model behaves normally on clean inputs but misclassifies those containing the embedded trigger.

To defend against such attacks, unlearning methods aim to effectively remove the influence of poisoned data from the model. Following the setup [13], we adopt BadNets [9] as the backdoor attack scenario in our experiments. Specifically, we conduct experiments on the CIFAR-10 dataset using a ResNet18 model. In this setup, 5% of the training data is poisoned by embedding triggers of specified sizes at random locations within the images. This experiment allows us to evaluate the ability of unlearning methods to eliminate the backdoor effect while preserving the model's performance on clean data.

In data poisoning scenarios, the trigger is often inaccessible. Consequently, our backdoor defense approach is divided into two phases: recovering the trigger and unlearning it. We adopt the recovery algorithm from the BAERASER [13] and replace the negative gradient method with our approach in the unlearning phase.

To evaluate the effectiveness of backdoor defense methods, we utilize two metrics: **ASR** and **Acc**. Here, ASR (Attack Success Rate) measures the success rate on the poisoned data, while Acc indicates the classification accuracy on clean samples. An effective defense algorithm should minimize ASR while maintaining a high Acc.

**Qualitative and Quantitative Results.** As shown in Tab. S8, our method achieves improved Quantitative results. Across various trigger sizes, our method shows performance gains ranging from 3.00% to 6.96% on all metrics. Notably, for small 3×3 trigger sizes, we observe improvements of 6.96% and 4.19% compared to the recovery+NG method. Meanwhile, some other methods face challenges of insufficient defense and reduced model performance. For example, NAD causes Acc to drop significantly to 69% with a 3×3 trigger, while fine pruning yields unsatisfactory defense effectiveness, with an ASR of 32.07%.

We further present the attention visualizations of the backdoored model and the model repaired using our method. As shown in the Fig. S1, our method effectively erases the influence of poisoned data, enabling the repaired model to shift its focus away from the trigger and towards classification-relevant regions.

Table S8. Attack success rate and accuracy across different trigger sizes and defense methods on CIFAR-10.

| Method | 3×3 | | 5×5 | | 7×7 | |
|---|---|---|---|---|---|---|
| | ASR ↓ | Acc ↑ | ASR ↓ | Acc ↑ | ASR ↓ | Acc ↑ |
| Original Model | 98.80 | 83.81 | 97.99 | 83.64 | 98.01 | 82.89 |
| Fine Pruning [12] | 32.07 | 77.96 | 37.41 | 78.34 | 35.24 | 76.29 |
| NAD [11] | 4.24 | 69.00 | 7.68 | 69.25 | 8.95 | 70.49 |
| Finetune [14] | 9.52 | 78.95 | 9.32 | 78.51 | 10.77 | 78.32 |
| Recovery + NG | 7.96 | 79.31 | 5.64 | 79.59 | 4.66 | 79.19 |
| Recovery + Ours | **1.00** | **83.50** | **1.24** | **83.07** | **1.66** | **82.23** |

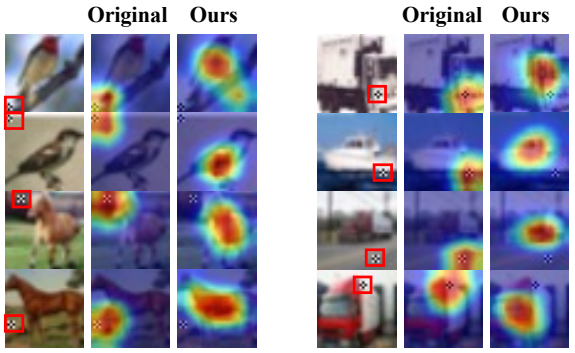


Figure S1. Visualization of attention heatmaps generated by the original backdoored model (**Original**) and the model repaired using our method (**Ours**). The red boxes ☐ highlight the backdoor triggers in the original images. The original backdoored model focuses primarily on the trigger regions, while our method successfully shifts attention to classification-relevant areas, demonstrating effective backdoor defense.

## D.3. Semantic Segmentation with Unlearning

**Implementation Details.** In this part, we explore the application of machine unlearning in the downstream task of semantic segmentation. Semantic segmentation aims to classify every pixel in an image, thereby generating a segmentation map. We use the DeepLab v3+ model [3] and the PASCAL VOC dataset [5] to conduct segmentation experiments.

Similar to unlearning in classification, we first train a segmentation model from scratch, and then unlearn an entire class from it. During the pretraining phase, we employ SGD optimization with a learning rate of 0.007, weight decay of 5e-4, and train for 50 epochs. Then we perform unlearning over 10 epochs.

To evaluate the methods' unlearning and remaining performance, we measure the Intersection over Union (IoU)

Table S9. Performance comparison of different unlearning methods on the semantic segmentation task.

| Metric | Retrain Model | Negative Gradient | Random Label | Ours |
|---|---|---|---|---|
| $\text{IoU}_{\text{ft}} \downarrow$ | 0 | 10.26 | 22.26 | **0** |
| $\text{IoU}_{\text{rt}} \uparrow$ | 71.55 | 42.26 | 45.06 | **68.93** |


Figure S2. Visualization of machine unlearning in the semantic segmentation task, including segmentation results generated by the original model (**Original**) and the unlearned model with our method (**Ours**).

on the test set. The IoU quantifies the overlap between the model's predicted segmentation map and the ground truth; a higher IoU indicates better segmentation performance. We compute the IoU for the forgotten class, denoted as $\mathbf{IoU_{ft}}$, and the average IoU for the remaining classes, denoted as $\mathbf{IoU_{rt}}$. In machine unlearning, a lower $\text{IoU}_{\text{ft}}$ indicates more effective unlearning, while a higher $\text{IoU}_{\text{rt}}$ suggests that the knowledge on remaining classes is less affected.

**Qualitative and Quantitative Results.** In terms of quantitative results, as shown in Tab. S9, our method achieves 0% IoU for the forgotten class and 68.93% IoU for the remaining classes, delivering performance closest to that of the retrain model. Other methods, while failing to effectively unlearn, exhibit a significant performance drop of over 25% in $\text{IoU}_{\text{rt}}$, indicating their inability to achieve effective unlearning or sufficient preservation of the remaining knowledge.

We further present the visualization of segmented results before and after unlearning in Fig. S2. The left three columns show segmentation results for images containing only the car class. While the original model correctly identifies and outlines the car, the unlearned model has "forgotten" and no longer detects it. The right three columns depict images with car and other objects, demonstrating that the target class is successfully forgotten, and the segmentation of people remains preserved.

## References


[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *S&P*, 2021. 2

[2] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *FG*, 2018. 3

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 4

[4] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *AAAI*, 2023. 2

[5] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 2010. 4

[6] Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. *arXiv preprint arXiv:2310.12508*, 2023. 2

[7] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *AAAI*, 2024. 2

[8] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*, 2020. 2

[9] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR*, 2017. 4

[10] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *NeurIPS*, 2024. 2

[11] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021. 4

[12] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018. 4

[13] Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. Backdoor defense with machine unlearning. In *INFOCOM*, 2022. 4

[14] Ximing Qiao, Yukun Yang, and Hai Li. Defending neural backdoors via generative distribution modeling. In *NeurIPS*, 2019. 4

[15] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX*, 2021. 2

[16] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *TNNLS*, 2023. 2