PRUNEFORGET: JOINT UNLEARNING AND PRUNING OF VISION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning and model pruning are increasingly *coupled* in the real world. Models must support unlearning requests, *e.g.*, for safety concerns, *while also* meeting requirements in latency and memory budget. Until recently, existing works have studied each aspect as an independent problem, *i.e.*, running unlearning and pruning sequentially. In this work, we show that unlearning and pruning are naturally aligned and should be solved jointly to be aware of each other. Intuitively, parameters that encode information of the unlearned samples are natural pruning targets, as unlearning and pruning both call for the "deletion" of such parameters. We propose PruneForget, a method that uses the unlearn set as a guide for pruning, so that unlearning and pruning mutually benefit each other. Extensive experiments on image classifiers and generative models show that PruneForget removes the influence of the unlearned samples while producing a more compact model with reduced inference cost.

1 INTRODUCTION

Deploying vision models in real-world applications increasingly requires satisfying two constraints *simultaneously*. This includes **(a)** compliance with safety or privacy requirements, *e.g.*, GDPR (European Union, 2016) and CCPA (Bonta, 2022), and **(b)** computation requirements for efficient inference. Until recently (Zhang et al., 2025), each of these aspects is typically studied by two disjoint areas of machine unlearning or model pruning. Machine unlearning focuses on approximating the model that would have been obtained if trained *without* the data in a given unlearn set (Thudi et al., 2022; Neel et al., 2021; Izzo et al., 2021; Koh & Liang, 2017; Fan et al., 2024; Kurmanji et al., 2023; Gandikota et al., 2023; Zhang et al., 2024a; Heng & Soh, 2023; Huang et al., 2024; He et al., 2019; Chundawat et al., 2023; Wu et al., 2025; Foster et al., 2024; Zheng et al., 2026). Model pruning aims to remove parameters from a trained model while maintaining performance (Ding et al., 2019a; Liu et al., 2021a; Fang et al., 2023b;a; 2024; Dong et al., 2017; Lee et al., 2019; Park et al., 2020; Guo et al., 2020; Ding et al., 2019b; Ye et al., 2018; Yu et al., 2018; Gao et al., 2021; Lin et al., 2020; Orseau et al., 2020; Sanh et al., 2020; Han et al., 2015; Yin et al., 2026). Both fields have made significant progress, and recent works (Jia et al., 2023; Xiao et al., 2025; Shirkavand et al., 2025) have begun to show that the two areas are not as separated as they may seem.

Jia et al. (2023) show that imposing sparsity regularization, common in pruning techniques, can be beneficial to unlearning. However, they fall short of actually removing parameters to produce a smaller model. Other works (Xiao et al., 2025; Shirkavand et al., 2025) have studied unlearning and pruning together, but their pruning strategies do not explicitly consider the effect of the unlearn set. LLM-Eraser by Zhang et al. (2025) leverages the unlearn set for pruning, but focuses its study on large language models (LLMs). While it may seem that this technique could transfer to vision models, empirically, we find that a direct application of LLM-Eraser to vision models does not perform well. This motivates us to study machine unlearning and model pruning *jointly* for vision models.

In this work, we focus on structured pruning methods (Fang et al., 2023a; 2024; 2023b), which have been successful in producing smaller vision models that can be efficiently deployed. At a high level, we observe that the set of data to be unlearned should help identify which of the model's parameters to prune, since pruning inherently removes encoded information from a model. We propose **PruneForget**, which uses the unlearn set to guide structured pruning across all three pruning stages (sparse training, pruning, and fine-tuning). Specifically, we introduce a sensitivity score to

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

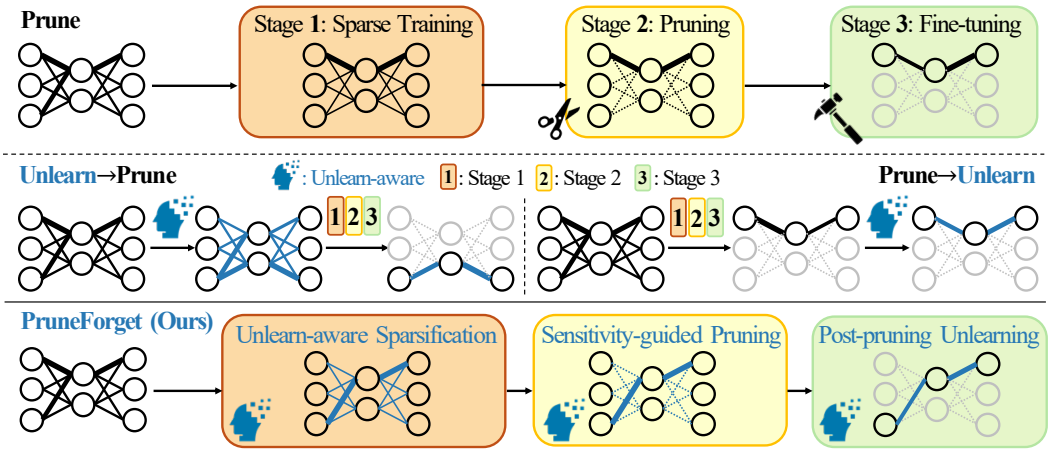


Figure 1: **Sequential methods *vs*. PruneForget.** Model pruning typically consists of three stages: sparse training, pruning, and fine-tuning. Sequential baselines (Unlearn→Prune or Prune→Unlearn) treat unlearning and pruning as independent tasks. Our proposed **PruneForget** integrates unlearning awareness into the three stages by introducing unlearn-aware sparsification, sensitivity-guided pruning, and post-pruning unlearning; achieving a joint unlearning and pruning framework.

quantify the information each parameter group retains with respect to the unlearn set and relative to the remaining data. A comparison between sequential methods vs. ours is provided in Fig. 1.

To validate our method, we performed extensive unlearning experiments across image classification and generation, subject to target pruning ratios. For classification, we evaluate on CIFAR-10 (Krizhevsky et al., 2009) and TinyImageNet (Le et al., 2015) with ResNet (He et al., 2016) and Swin Transformer (Liu et al., 2021b). For generative model unlearning, we conduct experiments on CIFAR-10 and ImageNet (Deng et al., 2009) using DDPM (Ho et al., 2020) and DiT (Peebles & Xie, 2023) architectures. The experiments also cover three different pruning techniques (Fang et al., 2023a; 2024; 2023b). In both domains, PruneForget achieves a negligible performance gap relative to the oracle (retrain-then-prune) model while requiring less unlearning and pruning time.

Our main contributions are as follows:

- We demonstrate that the unlearn set is useful for guiding which model parameters to structurally prune, enabling effective joint pruning and unlearning of vision models.
- Our approach introduces a novel *sensitivity score* that measures how much a pruning parameter group encodes information that should be unlearned. We use this score in **PruneForget**, the proposed three-stage framework for joint unlearning and pruning.
- Experiments across image classification and generation, multiple architectures, and pruning methods demonstrate the general applicability of PruneForget. We outperform existing sequential baselines as well as recent baselines, *e.g.*, LLM-Eraser.

2 Related Work

Machine unlearning aims to produce models that behave as if specific training data were never seen, without the cost of full retraining from scratch (Cao & Yang, 2015). Methods include gradient-based forgetting (Lu et al., 2026; Thudi et al., 2022; Neel et al., 2021), influence function approximations (Izzo et al., 2021; Koh & Liang, 2017), and Fisher information-guided parameter updates (Golatkar et al., 2020; Becker & Liebig, 2022; Heng & Soh, 2023; Foster et al., 2024). Fine-tuning-based approaches have since become the dominant paradigm, spanning image classifiers (Fan et al., 2024; Kurmanji et al., 2023; Huang et al., 2024; Jia et al., 2023; Chundawat et al., 2023; Heng & Soh, 2023), generative models (Gandikota et al., 2023; Zhang et al., 2024a; Heng & Soh, 2023; Wu et al., 2025; Huang et al., 2024), and LLMs (Zhang et al., 2024b; Yuan et al., 2025; Zhang et al., 2025). Broadly speaking, these methods all follow the general framework of jointly optimizing an unlearning loss on the unlearn set and a retain loss on the remainder of the dataset (Fan et al., 2024; Yuan et al., 2025;

Huang et al., 2024; Kurmanji et al., 2023; Chundawat et al., 2023; Zheng et al., 2026), with different designs of the loss function and optimization techniques. Regardless of formulation, these unlearning algorithms never modify the deep-net architecture, *i.e.*, parameter count remains the same before and after unlearning.

Model pruning. As modern deep-nets are typically over-parameterized, model pruning has been introduced to reduce computation and memory overhead. Model pruning generally involves evaluating an importance weight (mainly heuristic-based) to identify and discard less important parameters. The design of an importance weight depends on the specific pruning method. Fang et al. (2023a) propose using the L_2 norm of grouped parameters, Liu et al. (2021a) utilize Fisher information to rank channel importance, and Diff-Pruning (Fang et al., 2023b) utilizes first-order gradients across timesteps in diffusion models to determine which parameters to prune. These approaches are broadly categorized into unstructured (Dong et al., 2017; Lee et al., 2019; Park et al., 2020) and structured pruning (Ding et al., 2019a; Liu et al., 2021a; Fang et al., 2023b;a; 2024; Guo et al., 2020; Zhuang et al., 2020; Zhang et al., 2021; Schindler et al., 2020). While unstructured pruning zeros out individual weights, structured pruning removes entire channels (Fang et al., 2023a) or blocks (Guo et al., 2020), enabling hardware speedups without requiring specialized accelerators. Recently, model-specific pruning methods have been tailored for CNNs (Guo et al., 2020; Ding et al., 2019b; Ye et al., 2018), Transformers (Zhu et al., 2021; Yu et al., 2022; Mao et al., 2021; Song et al., 2022; Yin et al., 2026), diffusion models (Fang et al., 2023b; Ganjdanesh et al., 2024; Castells et al., 2024), and VLMs (Wu et al., 2026; Kang et al., 2026).

Unlearning with pruning. While both machine unlearning and model pruning remove information from a trained model, existing works treat the two as independent tasks. In ℓ_1-MU, Jia et al. (2023) show that model sparsity can facilitate unlearning. However, they *do not* actually prune the model. Despite the success of existing unlearning and model pruning methods, naively applying the two sequentially often results in ineffective unlearning or reduced performance. To bridge this gap, Un-pruning (Xiao et al., 2025) interleaves pruning and unlearning steps, while Shirkavand et al. (2025) focus specifically on the post-pruning phase and propose a bilevel optimization strategy for joint fine-tuning and unlearning. LLM-Eraser (Zhang et al., 2025) further incorporates unlearning directly into the pruning phase to eliminate weights tied to unlearned knowledge in LLMs. We summarize in Appendix Tab. A1 that these approaches only partially integrate the two domains into the three stages of pruning (sparse training, pruning, post-pruning fine-tuning). In contrast, our proposed PruneForget makes all three stages unlearn-aware to further improve the unlearning performance of pruned models.

3 PRELIMINARIES

Machine unlearning. The oracle unlearned model is defined as

$$\theta' \triangleq \arg\min_{\theta} \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta), \tag{1}$$

where a model is retrained using loss $\mathcal{L}$ from scratch on $\mathcal{D} \setminus \mathcal{U}$ by removing the unlearn set $\mathcal{U}$ from the training set $\mathcal{D}$. However, such retraining following Eq. (1) is prohibitively expensive for modern deep networks and large datasets. Unlearning algorithms therefore aim to efficiently produce an approximation $\tilde{\theta}$ of the oracle θ'. This approximation $\tilde{\theta}$ is expected to be close to the oracle θ', *i.e.*,

$$F_{\tilde{\theta}}(\boldsymbol{x}) \approx F_{\theta'}(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{X}, \tag{2}$$

where $F_\theta(\cdot)$ denotes the model's output.

Existing methods (Fan et al., 2024; Graves et al., 2021; Zhang et al., 2025; Huang et al., 2024; Kurmanji et al., 2023; Thudi et al., 2022) approximate θ' by updating the pre-trained model $\theta^\star$ (trained on the full dataset $\mathcal{D}$) using gradient-based optimization. The difference lies in the design choices of the unlearning loss $\mathcal{L}_\mathtt{U}$ and the retain loss $\mathcal{L}_\mathtt{R}$, and the gradient update algorithm

$$\tilde{\theta} \triangleq \mathtt{GD}(\mathcal{L}_\mathtt{U}, \mathcal{L}_\mathtt{R}, \mathcal{U}, \mathcal{D}, \theta^\star), \tag{3}$$

where $\mathtt{GD}$ denotes gradient descent starting from $\theta^\star$ that optimizes a combination of the unlearning loss $\mathcal{L}_\mathtt{U}$ and the retain loss $\mathcal{L}_\mathtt{R}$. The unlearning loss $\mathcal{L}_\mathtt{U}$ encourages the model to "forget" the unlearn set $\mathcal{U}$, with common choices including maximizing the cross-entropy loss (Graves et al., 2021) or

pushing predictions of the unlearned samples toward a uniform distribution (Zhang et al., 2025) (we denote this as $\mathcal{L}_{\texttt{Uni}}$). Next, the retain loss $\mathcal{L}_{\texttt{R}}$ is the standard supervised loss on the retain set $\mathcal{D} \setminus \mathcal{U}$ to preserve model performance.

Structured model pruning. Model pruning aims to reduce the computational and memory costs of deep-nets by removing redundant or less important parameters while preserving model performance. Existing approaches (Fang et al., 2023a; Liu et al., 2017; You et al., 2019; Zhang et al., 2021; Fang et al., 2024; 2023b) on structured pruning generally follow a three-step pipeline: sparsity training to prepare weights for removal, pruning to identify and remove unimportant parameters, and fine-tuning to recover performance on the pruned network.

(i) Sparse training (optional). Directly pruning a trained model $\theta^\star$ tends to remove weights that still carry useful information. To avoid this, weights θ are fine-tuned with a sparsity-inducing regularizer to encourage unimportant weights to be close to zero, *i.e.*,

$$\theta^{\texttt{s}} \triangleq \texttt{GD}(\mathcal{L}_{\texttt{prune}}, \mathcal{D}, \theta^\star), \text{ where } \mathcal{L}_{\texttt{prune}}(\mathcal{D}, \theta) \triangleq \mathcal{L}(\mathcal{D}, \theta) + \lambda \mathcal{R}(\theta). \tag{4}$$

Here $\mathcal{L}$ is the standard supervised loss, $\mathcal{R}$ is a sparsity-inducing regularizer, and λ is a hyperparameter that controls the strength of the regularization. A popular choice for the regularizer is proposed by Fang et al. (2023a), which considers structural dependencies between the weights. In more detail,

$$\mathcal{R}(\theta) \triangleq \sum_{g \in \mathcal{G}} \sum_{l \in g} \sum_{k \in l} \gamma_{l[k]} \left\| \theta_{l[k]} \right\|_2^2, \tag{5}$$

where they construct a set $\mathcal{G} = \{g\}$ of parameter groups g, where layers l are placed into g based on their dependencies in the computational graph. The k indexes the prunable dimension in a layer, *i.e.*, $\theta_{l[k]}$ denotes the k^{th} prunable weight of layer l. Finally, γ is a heuristic that determines the penalty amount; please refer to Fang et al. (2023a) for specifics.

(ii) Pruning. After the sparse training, pruning is performed by removing unimportant weights, *i.e.*, masking out the parameters for the pruned model

$$\theta^{\texttt{pr}} \triangleq \theta^{\texttt{s}} \odot \boldsymbol{m}, \tag{6}$$

where $\odot$ is element-wise multiplication and $\boldsymbol{m} \in \{0,1\}^{\text{DIM}(\theta)}$ is the binary pruning mask. An importance weight $I(\theta_{l[k]})$ is used to determine whether to keep the parameter, which depends on the specific pruning method. For example, in Group-level Pruning (Fang et al., 2023a), the importance weight is determined per group, *i.e.*, $I(\theta_{l[k]}) \triangleq \sum_{l' \in g(l)} \left\| \theta_{l'[k]} \right\|_2^2$, where $g(l)$ is the parameter group.

(iii) Fine-tuning. Performance usually drops after model pruning. Hence, to recover the performance, the pruned model is fine-tuned again on the training data. That is, another round of gradient-based updates is applied to obtain the final fine-tuned model $\theta^{\texttt{f}}$.

4 METHOD

The two objectives of machine unlearning and model pruning are naturally aligned. The unlearn set $\mathcal{U}$ identifies the very information that must be erased from the model, and the weights most responsible for encoding that information are essentially the ones that should be removed by pruning. Hence, in this work, our goal is to produce a pruned **and** unlearned model $\tilde{\theta}^{\texttt{f}}$ from a model $\theta^\star$ trained on the full dataset $\mathcal{D}$. Rather than treating unlearning and pruning as sequential post-hoc procedures, our proposed `PruneForget` leverages the unlearn set $\mathcal{U}$ to guide the pruning process, *i.e.*, the unlearning and pruning benefit each other within a single unified framework. The overall algorithm is stated in Alg. 1, where unlearn-aware terms are highlighted in blue. We now describe the details.

4.1 PRUNEFORGET

At a high level, we make the three-stage structured pruning pipeline (reviewed in Sec. 3) aware of unlearning. For this, we introduce a per-parameter-group *sensitivity score* $S_{l[k]}$, which measures how important the unlearn set $\mathcal{U}$ is relative to the retain set $\mathcal{D} \setminus \mathcal{U}$. In more detail,

$$S_{l[k]} \triangleq \sqrt{\left(\sum_{l' \in g(l)} \left\| \nabla_{\theta_{l'[k]}} \mathcal{L}_{\texttt{Uni}}(\mathcal{U}, \theta) \right\|_2^2 \right) \Big/ \left(\sum_{l' \in g(l)} \left\| \nabla_{\theta_{l'[k]}} \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta) \right\|_2^2 + \epsilon \right)}, \tag{7}$$

Algorithm 1 PruneForget

Inputs: Datasets $\mathcal{D}, \mathcal{U}$, parameter group set $\mathcal{G}$, importance weight I, loss weights α, η, λ, penalty heuristic γ, # iterations T_s, T_f.

$\theta \leftarrow \theta^\star$ # The pre-trained model

$$S_{l[k]} = \sqrt{\frac{\sum_{l' \in g(l)} \left\| \nabla_{\theta_{l'[k]}} \mathcal{L}_{\texttt{Uni}}(\mathcal{U}, \theta) \right\|_2^2}{\sum_{l' \in g(l)} \left\| \nabla_{\theta_{l'[k]}} \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta) \right\|_2^2 + \epsilon}}$$

$S'_{l[k]} = \text{Standardize}(S_{l[k]})$

 # Stage 1: Unlearn-aware sparsification

for $t = 1$ **to** T_s **do**

$$\mathcal{R}'(\theta) = \lambda \sum_{g \in \mathcal{G}} \sum_{l \in g} \sum_{k \in l} S'_{l[k]}\, \gamma_{l[k]}\, \left\| \theta_{l[k]} \right\|_2^2$$

$$\mathfrak{g} = \nabla_\theta \Big(\underbrace{-\alpha \mathcal{L}(\mathcal{U}, \theta)}_{\mathcal{L}_{\texttt{U}}} + \underbrace{\eta\, \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta) + \mathcal{R}'(\theta)}_{\mathcal{L}_{\texttt{prune}}} \Big)$$

$\theta \leftarrow \theta - \mathfrak{g}$

 # Stage 2: Sensitivity-guided pruning

τ is selected to satisfy a target prune ratio.

$I'(\theta_{l[k]}) \leftarrow I(\theta_{l[k]}) / S'_{l[k]}$

$$\boldsymbol{m}_i = \begin{cases} 1 & \text{if } I'(\theta_{l[k]}) > \tau \\ 0 & \text{otherwise} \end{cases}$$

$\theta \leftarrow \theta \odot \boldsymbol{m}$

 # Stage 3: Post-pruning unlearning

for $t = 1$ **to** T_f **do**

$$\mathbf{g} = \nabla_\theta \Big(\underbrace{-\alpha \mathcal{L}(\mathcal{U}, \theta)}_{\mathcal{L}_{\texttt{U}}} + \underbrace{\eta\, \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta)}_{\mathcal{L}_{\texttt{R}}} \Big)$$

$\theta \leftarrow \theta - \mathbf{g}$

Return: $\tilde{\theta}^{\texttt{f}} \leftarrow \theta$ # Pruned unlearned model

where $g(l)$ denotes the parameter group to which layer l belongs, and $\mathcal{L}_{\texttt{Uni}}$ is the KL divergence of the prediction class probability with a uniform distribution. We use the gradient norm as a proxy for importance, due to its effectiveness in the model pruning literature (Fang et al., 2024; 2023b; Liu et al., 2021a; Zhang et al., 2025). A large $S_{l[k]}$ indicates that the group encodes unlearn-set information more than retain-set utility, making it a suitable target for removal. The scores are then standardized, $S'_{l[k]} = \text{Standardize}(S_{l[k]})$, and passed to the first two stages.

Stage 1: Unlearn-aware sparsification. Before any structured pruning, Stage 1 encourages the forget-sensitive weights toward zero via a sensitivity-weighted regularizer applied alongside an unlearning procedure. Over T_s gradient steps, the model minimizes

$$\mathcal{L}_{\texttt{U}} + \mathcal{L}_{\texttt{prune}} \triangleq \overbrace{-\alpha\, \mathcal{L}(\mathcal{U}, \theta) + \underbrace{\eta\, \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta)}_{\text{Pruning}} + \mathcal{R}'(\theta)}^{\text{Unlearning}}, \tag{8}$$

where the sensitivity-weighted regularizer is

$$\mathcal{R}'(\theta) = \lambda \sum_{g \in \mathcal{G}} \sum_{l \in g} \sum_{k \in l} S'_{l[k]}\, \gamma_{l[k]}\, \left\| \theta_{l[k]} \right\|_2^2 . \tag{9}$$

This regularizer differs from the group regularizer reviewed in Eq. (5) in that the penalty on each group is scaled by $S'_{l[k]}$. Intuitively, a high sensitivity score S' pushes the weights that influence the unlearn set to be smaller. At the same time, the gradient ascent term $-\alpha\, \mathcal{L}(\mathcal{U}, \theta)$ simultaneously performs unlearning in this stage.

Stage 2: Sensitivity-guided pruning. Structured pruning uses an importance weight $I(\theta_{l[k]})$ to determine which parameters to prune. That is, if $I(\theta_{l[k]})$ is less than a threshold τ, then the parameter is pruned. To make the importance weight aware of the unlearn set, we scale it by the inverse of the proposed sensitivity (Eq. (7)), *i.e.*,

$$I'(\theta_{l[k]}) \triangleq I(\theta_{l[k]}) \;/\; S'_{l[k]}. \tag{10}$$

Dividing by $S'_{l[k]}$ lowers this sensitivity-guided importance weight of the parameters. Parameters with higher sensitivity are more related to the unlearn set; hence, dividing by sensitivity makes the parameter more likely to be pruned. Note that this approach is importance-weight-agnostic, *i.e.*, any existing pruning method's I can be made unlearn-aware by applying Eq. (10).

Stage 3: Post-pruning unlearning. As in model pruning, further fine-tuning is needed to recover the model's performance. As our goal is to have an unlearned model, we perform standard unlearning in this Stage 3, *i.e.*,

$$\theta \leftarrow \theta - \nabla_\theta \Big(\underbrace{-\alpha\, \mathcal{L}(\mathcal{U}, \theta)}_{\mathcal{L}_{\texttt{U}}} + \underbrace{\eta\, \mathcal{L}(\mathcal{D} \setminus \mathcal{U}, \theta)}_{\mathcal{L}_{\texttt{R}}} \Big). \tag{11}$$

The retain-set term $\mathcal{L}_{\texttt{R}}$ aims to recover the model's performance, whereas $\mathcal{L}_{\texttt{U}}$ removes the influence of data in the unlearn set.

Table 1: **CIFAR-10 (ResNet-18) random unlearning on classification averaged over 10 trials.** PruneForget achieves performance comparable to the oracle (Retrain→3SP). Baselines include SFRon (Huang et al., 2024), ℓ_1-MU (Jia et al., 2023), Un-pruning (Xiao et al., 2025), Bilevel (Shirkavand et al., 2025), and LLM-Eraser (Zhang et al., 2025). Performance gaps (Δ) relative to the oracle are reported first, with the raw UA, RA, and TA values provided in parentheses. Avg Δ reflects overall performance. Best results are in **blue**, while second best are ***italicized***.

Method	Random Unlearn 10%						Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 (6.03$_{\pm 0.56}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (93.74$_{\pm 0.32}$)	0.00	0.00	2215	0.00 (7.60$_{\pm 0.22}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (92.20$_{\pm 0.30}$)	0.00	0.00	1518
Pretrain→3SP	0.89 (5.15$_{\pm 0.38}$)	**0.00** (100.00$_{\pm 0.00}$)	0.41 (94.15$_{\pm 0.20}$)	0.43	**0.20**	984	1.54 (6.06$_{\pm 0.17}$)	**0.00** (100.00$_{\pm 0.00}$)	0.81 (93.01$_{\pm 0.27}$)	0.78	**0.23**	556
3SP→SFRon	14.84 (20.88$_{\pm 0.77}$)	1.88 (98.12$_{\pm 0.36}$)	3.38 (90.37$_{\pm 0.37}$)	6.70	0.24	1141	5.39 (12.99$_{\pm 0.54}$)	0.74 (99.25$_{\pm 0.11}$)	3.42 (88.79$_{\pm 0.56}$)	3.19	0.27	593
SFRon→3SP	***0.16*** (5.87$_{\pm 0.57}$)	**0.00** (100.00$_{\pm 0.01}$)	***0.04*** (93.70$_{\pm 0.31}$)	***0.07***	***0.22***	1152	**0.15** (7.45$_{\pm 0.21}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.34*** (91.87$_{\pm 0.28}$)	***0.16***	0.27	627
ℓ_1-MU→2SP	5.00 (11.03$_{\pm 0.46}$)	5.19 (94.81$_{\pm 0.26}$)	4.90 (88.85$_{\pm 0.21}$)	5.03	0.27	813	4.95 (12.55$_{\pm 0.20}$)	3.43 (96.57$_{\pm 0.27}$)	5.05 (87.16$_{\pm 0.16}$)	4.48	0.34	478
Un-pruning	1.70 (7.74$_{\pm 0.25}$)	0.05 (99.95$_{\pm 0.01}$)	1.66 (92.08$_{\pm 0.19}$)	1.14	0.41	937	2.72 (10.31$_{\pm 0.16}$)	0.07 (99.93$_{\pm 0.02}$)	2.94 (89.26$_{\pm 0.31}$)	1.91	0.48	550
Bilevel	7.71 (13.74$_{\pm 0.76}$)	7.01 (92.99$_{\pm 0.76}$)	7.04 (86.71$_{\pm 0.69}$)	7.25	0.55	942	6.61 (14.20$_{\pm 0.88}$)	3.72 (96.28$_{\pm 0.76}$)	6.73 (85.48$_{\pm 0.91}$)	5.69	0.63	642
LLM-Eraser	12.60 (18.63$_{\pm 1.37}$)	16.33 (83.67$_{\pm 1.24}$)	12.51 (81.23$_{\pm 1.35}$)	13.81	0.53	1482	10.46 (18.05$_{\pm 1.62}$)	14.00 (85.99$_{\pm 1.60}$)	10.59 (81.61$_{\pm 1.47}$)	11.68	0.45	810
PruneForget	**0.03** (6.01$_{\pm 0.28}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.01** (93.73$_{\pm 0.22}$)	**0.01**	***0.22***	998	***0.16*** (7.44$_{\pm 0.32}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.02** (92.22$_{\pm 0.23}$)	**0.06**	***0.25***	594

Table 2: **CIFAR-10 (ResNet-18) class-wise unlearning on classification averaged over 10 classes.** PruneForget achieves close performance to the oracle and the lowest Avg Δ in both scenarios.

Method	Unlearn 1 Class (10%)						Unlearn 5 Classes (50%)					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 (100.00$_{\pm 0.00}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (94.27$_{\pm 0.82}$)	0.00	0.00	2241	0.00 (100.00$_{\pm 0.00}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (96.38$_{\pm 1.39}$)	0.00	0.00	1554
Pretrain→3SP	**0.00** (100.00$_{\pm 0.00}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.44*** (94.72$_{\pm 0.57}$)	***0.15***	**0.25**	989	**0.00** (100.00$_{\pm 0.00}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.22*** (96.60$_{\pm 1.23}$)	**0.07**	**0.36**	566
3SP→SFRon	**0.00** (100.00$_{\pm 0.00}$)	2.77 (97.22$_{\pm 1.03}$)	2.34 (91.94$_{\pm 1.31}$)	1.70	0.46	1078	**0.00** (100.00$_{\pm 0.00}$)	3.46 (96.54$_{\pm 2.84}$)	3.53 (92.85$_{\pm 3.20}$)	2.33	1.54	619
SFRon→3SP	**0.00** (100.00$_{\pm 0.00}$)	***0.01*** (99.99$_{\pm 0.01}$)	0.67 (93.60$_{\pm 0.68}$)	0.23	0.33	1074	**0.00** (100.00$_{\pm 0.00}$)	***0.01*** (99.99$_{\pm 0.01}$)	0.89 (95.48$_{\pm 2.00}$)	0.30	0.47	615
ℓ_1-MU→2SP	**0.00** (100.00$_{\pm 0.00}$)	4.93 (95.07$_{\pm 0.70}$)	4.44 (89.84$_{\pm 0.78}$)	3.12	0.31	805	**0.00** (100.00$_{\pm 0.00}$)	1.74 (98.26$_{\pm 0.86}$)	3.18 (93.20$_{\pm 1.94}$)	1.64	0.45	453
Un-pruning	**0.00** (100.00$_{\pm 0.00}$)	0.05 (99.95$_{\pm 0.02}$)	1.53 (92.75$_{\pm 0.56}$)	0.52	0.48	937	**0.00** (100.00$_{\pm 0.00}$)	0.04 (99.96$_{\pm 0.03}$)	1.60 (94.77$_{\pm 1.47}$)	0.55	0.66	533
Bilevel	**0.00** (100.00$_{\pm 0.00}$)	58.23 (41.77$_{\pm 14.96}$)	52.53 (41.74$_{\pm 14.83}$)	36.92	2.19	920	**0.00** (100.00$_{\pm 0.00}$)	17.57 (82.43$_{\pm 11.71}$)	15.82 (80.56$_{\pm 10.23}$)	11.13	1.34	643
LLM-Eraser	**0.00** (100.00$_{\pm 0.00}$)	21.50 (78.49$_{\pm 11.71}$)	17.58 (76.69$_{\pm 11.08}$)	13.03	0.78	1512	**0.00** (100.00$_{\pm 0.00}$)	11.06 (88.94$_{\pm 3.39}$)	9.36 (87.02$_{\pm 3.55}$)	6.81	0.71	795
PruneForget	**0.00** (100.00$_{\pm 0.00}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.41** (93.86$_{\pm 0.71}$)	**0.14**	***0.28***	984	**0.00** (100.00$_{\pm 0.00}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.20** (96.17$_{\pm 1.21}$)	**0.07**	***0.38***	596

4.2 Practical Details of PruneForget

Unlearning loss $\mathcal{L}_{\texttt{Uni}}$. For classification, $\mathcal{L}_{\texttt{Uni}}$ is the KL divergence between the model's predicted class probabilities and a uniform distribution, as defined in Sec. 3. For generative models, we use the MSE between the model's predicted noise and randomly sampled Gaussian noise, which serves to drive the model's output on $\mathcal{U}$ toward an uninformative target (Aravindan et al., 2025).

Sensitivity-weighted penalty coefficient $\gamma_{l[k]}$. As in Fang et al. (2023a), our method also has this $\gamma_{l[k]}$ coefficient defined as $\gamma_{l[k]} \triangleq 2^{\kappa(I^{\max}_{g(l)} - I_{l[k]})/(I^{\max}_{g(l)} - I^{\min}_{g(l)})}$, where $I^{\max}_{g(l)}$ and $I^{\min}_{g(l)}$ are the maximum and minimum importance weights within group $g(l)$. In Fang et al. (2023a), they choose a fixed $\kappa = 4$. In this work, we find it beneficial to use a scheduler for this term to gradually increase the importance. Specifically, we choose $\kappa = 4 \cdot t/T_s$ across the T_s steps of Stage 1.

Unlearning & retain balance. As the unlearn set $\mathcal{U}$ is typically much smaller than the retain set $\mathcal{D}\backslash\mathcal{U}$, running the same number of gradient steps for each of them leads to unbalanced updates. Following SFRon (Huang et al., 2024), we adopt a forget-frequency schedule, *i.e.*, running f gradient steps on $\mathcal{L}_{\texttt{R}}$ for every one step on $\mathcal{L}_{\texttt{U}}$. The values of f are found by grid search; we provide more details in an appendix. Moreover, we employ a dynamic unlearning loss weight $\alpha_t = \alpha\,\zeta_t$ for classification tasks. Specifically, $\zeta_t = \text{clip}\big((\ell_{\max} - \mathcal{L}(\mathcal{U},\theta))/\ell_{\max},\, 0,\, 1\big)$, where $\ell_{\max}$ is the cross-entropy of a uniform predictor. Once $\mathcal{L}(\mathcal{U},\theta) \geq \ell_{\max}$, $\mathcal{L}_{\texttt{U}}$ is permanently switched off. For generative models, $\mathcal{L}_{\texttt{U}}$ is turned off once the generation of the unlearned concepts deviates too far from a random Gaussian; we threshold based on MSE (equivalent to KL divergence up to constant scaling).

5 Experiments

To evaluate the effectiveness of PruneForget, we conduct experiments on both image classification and class-conditioned generation tasks. We assess the unlearning efficacy of models pruned to a **fixed target ratio, *i.e.*, all the pruned models are equally efficient**. See the Appendix for implementation details, extended results, and global pruning experiments.

5.1 Unlearning for Classification

Setup. Experiments incorporate aspects from both the unlearning and pruning literature. For the unlearning setup, we follow the setting of Huang et al. (2024) and evaluate ResNet-18 (He

Table 3: **TinyImageNet (Swin-T) random unlearning on classification averaged over 10 trials.** PruneForget achieves near-oracle performance across all metrics with the lowest Avg Δ.

Method	Random Unlearn 10%						Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→2SP	0.00 (19.88$_{\pm 0.00}$)	0.00 (98.66$_{\pm 0.00}$)	0.00 (80.18$_{\pm 0.00}$)	0.00	0.00	8814	0.00 (22.61$_{\pm 0.00}$)	0.00 (99.24$_{\pm 0.00}$)	0.00 (77.56$_{\pm 0.00}$)	0.00	0.00	5704
Pretrain→2SP	1.64 (18.24$_{\pm 0.36}$)	0.05 (98.71$_{\pm 0.03}$)	***0.09*** (80.27$_{\pm 0.21}$)	***0.59***	**0.25**	6278	***2.32*** (20.30$_{\pm 0.13}$)	**0.07** (99.32$_{\pm 0.04}$)	***0.26*** (77.82$_{\pm 0.18}$)	***0.88***	**0.30**	3771
2SP→SFRon	7.26 (27.14$_{\pm 0.87}$)	0.62 (98.04$_{\pm 0.18}$)	4.83 (75.35$_{\pm 0.78}$)	4.24	0.45	6968	4.61 (27.22$_{\pm 0.72}$)	0.28 (98.96$_{\pm 0.08}$)	5.64 (71.92$_{\pm 0.59}$)	3.51	0.53	4779
SFRon→2SP	***1.63*** (18.25$_{\pm 0.31}$)	**0.03** (98.69$_{\pm 0.04}$)	0.20 (80.38$_{\pm 0.22}$)	0.62	***0.30***	7039	2.51 (20.11$_{\pm 0.22}$)	***0.09*** (99.34$_{\pm 0.05}$)	0.44 (78.00$_{\pm 0.27}$)	1.01	***0.34***	4865
ℓ_1-MU→2SP	29.54 (49.42$_{\pm 2.31}$)	39.17 (59.49$_{\pm 2.60}$)	29.36 (50.82$_{\pm 2.10}$)	32.69	1.36	7412	33.71 (56.33$_{\pm 0.48}$)	45.53 (53.71$_{\pm 0.78}$)	34.04 (43.51$_{\pm 0.50}$)	37.76	1.55	4486
Un-pruning	5.66 (25.54$_{\pm 0.61}$)	2.85 (95.81$_{\pm 0.32}$)	6.30 (73.88$_{\pm 0.52}$)	4.94	0.51	10040	5.92 (28.53$_{\pm 0.42}$)	1.87 (97.37$_{\pm 0.32}$)	6.88 (70.67$_{\pm 0.62}$)	4.89	0.58	5694
Bilevel	13.00 (6.87$_{\pm 0.24}$)	2.52 (96.14$_{\pm 0.04}$)	2.99 (83.17$_{\pm 0.20}$)	6.17	1.21	18672	16.29 (6.33$_{\pm 0.07}$)	2.11 (97.13$_{\pm 0.09}$)	5.36 (82.92$_{\pm 0.11}$)	7.92	1.47	18785
LLM-Eraser	6.40 (13.47$_{\pm 0.28}$)	3.05 (95.61$_{\pm 0.18}$)	2.13 (82.31$_{\pm 0.25}$)	3.86	0.35	7804	8.35 (14.27$_{\pm 0.19}$)	2.76 (96.49$_{\pm 0.10}$)	3.40 (80.96$_{\pm 0.17}$)	4.83	0.41	4241
PruneForget	**1.50** (18.38$_{\pm 0.24}$)	**0.03** (98.69$_{\pm 0.03}$)	**0.07** (80.25$_{\pm 0.17}$)	**0.54**	***0.30***	7026	**0.98** (21.64$_{\pm 0.12}$)	0.10 (99.35$_{\pm 0.03}$)	**0.21** (77.35$_{\pm 0.27}$)	**0.43**	0.36	4912

Table 4: **Ablation of pipeline stages.** Evaluating the contribution of each optimization stage in PruneForget across 50% random unlearning on CIFAR-10. Integrating all three stages yields the optimal balance between unlearning efficacy and model utility.

Stage 1	Stage 2	Stage 3	Random Unlearn 50% ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
			1.54 (6.06$_{\pm 0.17}$)	**0.00** (100.00$_{\pm 0.00}$)	0.81 (93.01$_{\pm 0.27}$)	0.78	**0.23**	556
✓			0.93 (6.67$_{\pm 0.17}$)	**0.00** (100.00$_{\pm 0.00}$)	0.29 (92.50$_{\pm 0.24}$)	0.41	0.25	588
✓	✓		***0.80*** (6.80$_{\pm 0.16}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.22*** (92.43$_{\pm 0.20}$)	***0.34***	***0.24***	590
✓	✓	✓	**0.16** (7.44$_{\pm 0.32}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.02** (92.22$_{\pm 0.23}$)	**0.06**	0.25	594

et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) and pre-trained Swin-T (Liu et al., 2021b) on TinyImageNet (Le et al., 2015). We report on both random unlearning (*i.e.*, randomly removing a subset of the training data) and class-wise unlearning (*i.e.*, removing all images of a specific class). For the pruning setup, we follow the setup of Group-level Pruning (Fang et al., 2023a) for ResNet-18 and Isomorphic Pruning (Fang et al., 2024) for Swin-T as the default structural model pruning mechanisms. We enforce a target pruning ratio p of 0.5 for ResNet-18 and 0.2 for Swin-T, *i.e.*, every baseline evaluated has exactly $100 \cdot p\%$ channels per layer removed. Because the pruning methods may differ in the number of stages (either the last two or three described in Sec. 3), we introduce the notation n**SP** (n-Stage Pruning) to denote the number of stages applied. For a fair comparison, PruneForget adopts the *same* pruning mechanism (2SP or 3SP) as the baselines it is compared against.

Evaluation metrics. To measure unlearning performance, prior works (Fan et al., 2024; Jia et al., 2023) report the *gap* relative to the oracle model. In our setting, the oracle model (Retrain→nSP) is created by first obtaining an unlearned model by training from scratch, followed by n-stage pruning. To quantify this gap, we report $\Delta\text{Acc}(\mathcal{S}) \triangleq |\text{Acc}(\tilde{\theta}, \mathcal{S}) - \text{Acc}(\theta', \mathcal{S})|$, which measures the absolute accuracy difference between the unlearned model $\tilde{\theta}$ and the pruned oracle θ' over a dataset split $\mathcal{S}$. We evaluate this gap on the **U**nlearn set $\mathcal{U}$ (ΔUA), the **R**etain set $\mathcal{D} \setminus \mathcal{U}$ (ΔRA), and the **T**est set $\mathcal{T}$ (ΔTA). Note we report the error rate ($100 - \text{Acc}$) as the UA to show unlearning efficacy. To assess overall performance, we report the mean gap **Avg Δ** as our primary evaluation metric. Additionally, we report the KL divergence (D_{KL}) to quantify the distributional difference between the model outputs and the oracle's, alongside the total execution **Time** (in seconds) for the unlearn and prune procedure.

Baselines. We compare our method against several baselines: **Pretrain**→n**SP**, which prunes the pre-trained model without unlearning; n**SP**→**SFRon** (Huang et al., 2024), which applies n pruning stages followed by the SFRon unlearning algorithm; **SFRon** (Huang et al., 2024)→n**SP**, which unlearns using SFRon followed by the pruning stages; ℓ_1**-MU** (Jia et al., 2023)→**2SP**, utilizing sparsity-inducing weights for unlearning followed by the last two pruning stages; **Un-pruning** (Xiao et al., 2025), which iteratively interleaves unlearning and partial weight reinitialization before a final pruning step; **Bilevel** (Shirkavand et al., 2025), integrating unlearning into the post-pruning fine-tuning stage via bilevel optimization; and **LLM-Eraser** (Zhang et al., 2025), which incorporates unlearning into the pruning and fine-tuning stages and applies them iteratively.

Results on *random* unlearning on CIFAR-10 (ResNet-18). We report the 10% and 50% random unlearning results in Tab. 1. Without an explicit unlearning objective, the Pretrain→3SP baseline fails to erase the influence of the data, resulting in a UA that is 1.54% lower than the oracle model (6.06% *vs.* 7.60%). Conversely, while other prior works achieve some unlearning effectiveness, they struggle to maintain low ΔTA in both cases. In contrast, PruneForget consistently achieves the lowest Avg Δ under both unlearning ratios. Overall, our method shows robust unlearning with a zero ΔRA alongside low ΔUA, ΔTA, and D_{KL}, while consuming less than 50% of the oracle's runtime.

Table 5: **Sensitivity score design comparison.** We evaluate the impact of various designs of $\mathcal{S}$ on 50% random unlearning on CIFAR-10. Ours achieves the lowest Avg Δ among all methods.

Design of $\mathcal{S}$	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
1 (Uniform)	1.49 ($6.11_{\pm 0.33}$)	**0.00** ($100.00_{\pm 0.00}$)	0.71 ($92.91_{\pm 0.21}$)	0.73	**0.23**	575
Random	2.29 ($5.31_{\pm 0.45}$)	**0.00** ($100.00_{\pm 0.00}$)	0.98 ($93.19_{\pm 0.19}$)	1.09	**0.23**	574
Unlearn Only	**0.15** ($7.44_{\pm 0.15}$)	**0.00** ($100.00_{\pm 0.00}$)	***0.05*** ($92.25_{\pm 0.22}$)	***0.07***	0.26	587
Retain Only	1.05 ($8.64_{\pm 0.43}$)	**0.00** ($100.00_{\pm 0.00}$)	1.09 ($91.11_{\pm 0.50}$)	0.71	0.27	589
Fisher Score	**0.15** ($7.74_{\pm 0.58}$)	**0.00** ($100.00_{\pm 0.00}$)	0.15 ($92.05_{\pm 0.36}$)	0.10	***0.25***	620
Ours	***0.16*** ($7.44_{\pm 0.32}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.02** ($92.22_{\pm 0.23}$)	**0.06**	***0.25***	594

Table 6: **Comparison of updating vs. static sensitivity score.** We compare an updating sensitivity score versus a static one. The static approach achieves a smaller gap to the oracle across all metrics.

Method	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time (s)
Retrain→3SP	0.00 ($7.60_{\pm 0.22}$)	0.00 ($100.00_{\pm 0.00}$)	0.00 ($92.20_{\pm 0.30}$)	0.00	0.00	1518
Updating Sensitivity	0.41 ($8.00_{\pm 0.49}$)	**0.00** ($100.00_{\pm 0.00}$)	0.51 ($91.70_{\pm 0.35}$)	0.30	0.26	604
Static (PruneForget)	**0.16** ($7.44_{\pm 0.32}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.02** ($92.22_{\pm 0.23}$)	**0.06**	**0.25**	594

Results on *class-wise* unlearning on CIFAR-10 (ResNet-18). In Tab. 2, we evaluate class-wise unlearning on 1-class and 5-class scenarios. PruneForget achieves the lowest Avg Δ and zero gap in both UA and RA, with a near-zero gap in TA and a low D_{KL}, outperforming the compared baselines. This result is expected, as class-wise unlearning is typically the easier task vs. random unlearning.

Results on *random* unlearning on TinyImageNet (Swin-T). We report the 10% and 50% random unlearning results in Tab. 3. The Pretrain→2SP baseline again suffers from a significant unlearning deficit (ΔUA = 2.32 on 50% unlearning). While other baselines also struggle to balance UA and TA, PruneForget consistently achieves the lowest Avg Δ. Notably, our method yields the smallest ΔUA and ΔTA in both cases while maintaining a robust performance across all metrics. This result demonstrates PruneForget's applicability to different datasets and model architectures.

Ablation on the three stages of PruneForget. We conduct an ablation study on the CIFAR-10 50% random unlearning task to demonstrate the contribution of each stage in PruneForget. When a specific stage is ablated, we revert to the default pruning procedure. In Tab. 4, we observe that all stages meaningfully contribute to the overall unlearning efficacy. Specifically, progressively incorporating Stage 1 and Stage 2 reduces ΔUA from 1.54% to 0.93% and 0.80% and ΔTA from 0.81% to 0.29% and 0.22%. Integrating all three stages further achieves the best trade-off, lowering ΔUA to 0.16% and ΔTA to a near-oracle 0.02%, yielding the lowest overall Avg Δ of 0.06%. Overall, these results show the value of integrating unlearning awareness into every stage of the pruning pipeline.

Sensitivity score design comparison. In Tab. 5, we evaluate the impact of various formulations for the sensitivity score $\mathcal{S}$ on the CIFAR-10 50% random unlearning task. Naive baselines, such as uniform or random assignments, lead to substantial performance gaps (Avg $\Delta \geq 0.73\%$). Relying solely on the retain set impairs forgetting efficacy ($\Delta\text{UA} = 1.05\%$). While the unlearn-only subset and the Fisher score improve forgetting, they still result in suboptimal test retention ($\Delta\text{TA} \geq 0.05\%$). In contrast, our proposed formulation achieves the lowest overall Avg Δ alongside a near-oracle ΔTA of 0.02%, which is the best trade-off across all other designs.

Ablation on iteratively updating the sensitivity score. In Alg. 1, the sensitivity score $S_{l[k]}$ is computed once on the pre-trained model. We now conduct an ablation to determine whether updating this score iteratively during Stage 1 would be beneficial. In Tab. 6, we use the setting of CIFAR-10 50% random unlearning for this ablation. We use a running mean across Stage 1 to iteratively update $S_{l[k]}$. Overall, the static approach (PruneForget) achieves better Avg Δ, indicating that analyzing the entire dataset beforehand is sufficient to determine the contribution of weights to both the unlearned and the retained information. This design choice is consistent with SFRon (Huang et al., 2024), which computes its gradient mask once on the pre-trained model and reuses it throughout training.

Ablation on dynamic weighted unlearning. In Tab. 7, we evaluate dynamic weighted unlearning against a fixed baseline on CIFAR-10 with 50% random unlearning. Compared to the fixed variant, our dynamic scheme effectively reduces both ΔUA and ΔTA, lowering Avg Δ from 0.12% to 0.06%. That is, the proposed dynamic weighting improves the trade-off between unlearn and retain sets.

Table 7: **Ablation of dynamic weighted unlearning.** We evaluate dynamic weighted unlearning on 50% random unlearning of CIFAR-10, demonstrating that it effectively improves Avg Δ.

Method	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 ($7.60_{\pm 0.22}$)	0.00 ($100.00_{\pm 0.00}$)	0.00 ($92.20_{\pm 0.30}$)	0.00	0.00	1518
$\zeta_t = 1$	0.20 ($7.80_{\pm 0.33}$)	**0.00** ($100.00_{\pm 0.00}$)	0.17 ($92.03_{\pm 0.22}$)	0.12	0.26	612
Dynamic ζ_t	**0.16** ($7.44_{\pm 0.32}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.02** ($92.22_{\pm 0.23}$)	**0.06**	**0.25**	594

Table 8: **Ablation on iterative pruning rounds.** We evaluate the impact of the number of iterative pruning rounds (R). A single round ($R = 1$) achieves the closest oracle alignment.

Method	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time (s)
Retrain→3SP	0.00 ($7.60_{\pm 0.22}$)	0.00 ($100.00_{\pm 0.00}$)	0.00 ($92.20_{\pm 0.30}$)	0.00	0.00	1518
$R = 1$ (Ours)	**0.16** ($7.44_{\pm 0.32}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.02** ($92.22_{\pm 0.23}$)	**0.06**	**0.25**	594
$R = 2$	0.57 ($7.02_{\pm 0.14}$)	**0.00** ($100.00_{\pm 0.00}$)	0.49 ($92.69_{\pm 0.22}$)	0.35	**0.25**	599
$R = 5$	0.67 ($6.93_{\pm 0.14}$)	**0.00** ($100.00_{\pm 0.00}$)	0.57 ($92.77_{\pm 0.16}$)	0.41	**0.25**	646
$R = 10$	0.45 ($7.15_{\pm 0.18}$)	**0.00** ($100.00_{\pm 0.00}$)	0.48 ($92.69_{\pm 0.14}$)	0.31	**0.25**	702

Ablation on iterative pruning rounds. In Tab. 8, we evaluate iterative pruning rounds (R) by alternating between Stage 1 and Stage 2. To match the compute, we divided the total iterations by a factor of R for each iteration of Stage 1 and 2, and iterated for R times. Increasing R yields no performance gains; thus, we maintain single-round pruning as our default.

5.2 Unlearning for class-conditioned generation

Setup & evaluation metrics. For unlearning, we follow Huang et al. (2024) using DDPMs (Ho et al., 2020) on CIFAR-10 and DiT (Peebles & Xie, 2023) on ImageNet-1K (Deng et al., 2009). As for pruning, we enforce $p = 0.3$ via Diff-Pruning (Fang et al., 2023b) and compare against the same baselines detailed in Sec. 5.1. Note that DiT results are deferred to Appendix Sec. A4. We evaluate performance by reporting the gap relative to the oracle model (Retrain→nSP) using three metrics: **UA**, the unlearned accuracy of a pre-trained classifier on images generated for the unlearn classes; **FID**, the Fréchet Inception Distance (Heusel et al., 2017) assessing image generation quality on the retain classes; and **Time**, the total execution time in seconds.

Results on class-wise unlearning on CIFAR-10. We show the average class-wise unlearning results of DDPM over 5 classes (*Automobile*, *Cat*, *Dog*, *Horse*, and *Truck*) in Tab. 9. We observe PruneForget effectively matches the oracle model across all cases. It yields the lowest ΔUA alongside highly competitive ΔFID scores, demonstrating its ability to unlearn targeted concepts while maintaining overall image quality effectively. Importantly, PruneForget achieves these near-zero performance gaps while accelerating runtime by over 10× relative to the oracle model. Qualitative results are provided in Appendix Fig. A1, and we observe them to be consistent with the quantitative metrics.

Table 9: **CIFAR-10 (DDPM) class-wise unlearning results.** Averaged across 5 classes, PruneForget obtains close performance to the oracle with over 10× acceleration.

Method	Average (5 Classes)		Time
	ΔUA ↓	ΔFID ↓	
Retrain→2SP	0.00 ($98.39_{\pm 1.65}$)	0.00 ($16.73_{\pm 0.63}$)	398622
Pretrain→2SP	0.45 ($97.94_{\pm 1.70}$)	0.30 ($16.95_{\pm 0.47}$)	21304
2SP→SFRon	0.39 ($98.00_{\pm 1.52}$)	**0.22** ($16.92_{\pm 0.70}$)	25652
SFRon→2SP	0.55 ($97.83_{\pm 1.77}$)	***0.23*** ($16.96_{\pm 0.71}$)	26674
ℓ_1-MU→2SP	***0.20*** ($98.19_{\pm 1.76}$)	0.76 ($17.49_{\pm 0.73}$)	22675
Un-pruning	0.75 ($98.89_{\pm 2.23}$)	326.24 ($342.96_{\pm 31.06}$)	36586
Bilevel	4.30 ($94.09_{\pm 2.68}$)	0.98 ($17.71_{\pm 0.40}$)	33863
LLM-Eraser	3.08 ($95.31_{\pm 4.50}$)	7.41 ($24.13_{\pm 2.52}$)	27570
PruneForget	**0.13** ($98.26_{\pm 1.64}$)	0.26 ($16.92_{\pm 0.76}$)	26254

6 Conclusion

We presented PruneForget, a method that jointly performs machine unlearning and model pruning by using the unlearn set to guide which model parameters to remove. Intuitively, if a weight encodes information that should be unlearned, then it is a good pruning candidate. To realize this intuition, we proposed a sensitivity score that measures the ratio between the unlearned information and the information in the retain set. We then incorporated this score into the three-stage pruning pipeline, which consists of sparse training, pruning, and fine-tuning. By doing so, PruneForget produces an unlearned model that is pruned to satisfy a compute budget. Experiments across image classification/generation tasks, different architectures, and pruning mechanisms demonstrate that PruneForget is effective and outperforms existing baselines that jointly unlearn and prune models.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing, searching on related works, and adapting prior works' code for experiments. We have not used generative AI tools for drafting sections of the paper, generating synthetic datasets, or proving mathematical claims. We have reviewed all AI-assisted work. For example, any related work suggested by the AI was checked manually. Furthermore, AI-assisted code adaptations from prior works used in our experiments were manually verified and tested for correctness within our pipeline. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Ashwath Vaithinathan Aravindan, Abha Jha, Matthew Salaway, Atharva Sandeep Bhide, and Duygu Nur Yaldiz. Sealing the backdoor: Unlearning adversarial text triggers in diffusion models using knowledge distillation. *arXiv preprint arXiv:2508.18235*, 2025. 6

Alexander Becker and Thomas Liebig. Evaluating machine unlearning via epistemic uncertainty. *arXiv preprint arXiv:2208.10836*, 2022. 2

Rob Bonta. California consumer privacy act (CCPA). *Retrieved from State of California Department of Justice:* https://oag.ca.gov/privacy/ccpa, 2022. 1

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *Proc. IEEE S&P*, 2015. 2

Thibault Castells, Hyoung-Kyu Song, Bo-Kyeong Kim, and Shinkook Choi. LD-Pruner: Efficient pruning of latent diffusion models using task-agnostic insights. In *Proc. CVPR*, 2024. 3

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 *Proc. AAAI*, 2023. 1, 2, 3, 17

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 2, 9

Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal SGD for pruning very deep convolutional networks with complicated structure. In *Proc. CVPR*, 2019a. 1, 3

Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive CNN width optimization. In *Proc. ICML*, 2019b. 1, 3

Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Proc. NeurIPS*, 2017. 1, 3

European Union. Regulation (EU) 2016/679 of the european parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (general data protection regulation), 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj. OJ L 119, 4.5.2016, p. 1–88. 1

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. In *Proc. ICLR*, 2024. 1, 2, 3, 7, 17

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. DepGraph: Towards any structural pruning. In *Proc. CVPR*, 2023a. 1, 2, 3, 4, 6, 7, 14, 15, 19

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Proc. NeurIPS*, 2023b. 1, 2, 3, 4, 5, 9, 15

Gongfan Fang, Xinyin Ma, Michael Bi Mi, and Xinchao Wang. Isomorphic pruning for vision models. In *Proc. ECCV*, 2024. 1, 2, 3, 4, 5, 7, 15, 19

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proc. AAAI*, 2024. 1, 2

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. ICLR*, 2019. 17

Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proc. ICCV*, 2023. 1, 2, 19

Alireza Ganjdanesh, Reza Shirkavand, Shangqian Gao, and Heng Huang. Not all prompts are made equal: Prompt-based pruning of text-to-image diffusion models. *arXiv preprint arXiv:2406.12042*, 2024. 3

Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proc. CVPR*, 2021. 1

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

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proc. AAAI*, 2021. 3

Jinyang Guo, Wanli Ouyang, and Dong Xu. Multi-dimensional pruning: A unified framework for model compression. In *Proc. CVPR*, 2020. 1, 3

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 2, 6

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proc. CVPR*, 2019. 1

Alvin Heng and Harold Soh. Selective amnesia: A continual learning approach to forgetting in deep generative models. In *Proc. NeurIPS*, 2023. 1, 2

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proc. NeurIPS*, 2017. 9

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. NeurIPS*, 2020. 2, 9

Zhehao Huang, Xinwen Cheng, JingHao Zheng, Haoran Wang, Zhengbao He, Tao Li, and Xiaolin Huang. Unified gradient-based machine unlearning with remain geometry enhancement. In *Proc. NeurIPS*, 2024. 1, 2, 3, 6, 7, 8, 9, 14, 15, 17, 18

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *Proc. AISTATS*, 2021. 1, 2

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. In *Proc. NeurIPS*, 2023. 1, 2, 3, 6, 7, 16

Minseok Kang, Hyunwoo Kim, Chanyoung Kim, Minwoo Kim, Jaekoo Lee, and Dahuin Jung. Prune once: Retraining-free task-agnostic pruning for vision-language models. In *Proc. ECCV*, 2026. 3

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. ICML*, 2017. 1, 2

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 7

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Proc. NeurIPS*, 2023. 1, 2, 3, 17

Yann Le, Xuan Yang, et al. Tiny ImageNet visual recognition challenge. *CS 231N*, 2015. 2, 7

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. *arXiv preprint arXiv:1906.06307*, 2019. 1, 3

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. HRank: Filter pruning using high-rank feature map. In *Proc. CVPR*, 2020. 1

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *Proc. ICML*, 2021a. 1, 3, 5

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 *Proc. ICCV*, 2021b. 2, 7

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proc. ICCV*, 2017. 4

Juxin Lu, Haoyu Shi, Mengyao Wang, and Huaiwen Zhang. Machine unlearning via adaptive gradient reweighting and multi-stage objective optimization. In *Proc. CVPR*, 2026. 2

Jiachen Mao, Huanrui Yang, Ang Li, Hai Li, and Yiran Chen. TPrune: Efficient transformer pruning for mobile devices. *ACM Transactions on Cyber-Physical Systems*, 2021. 3

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proc. ALT*, 2021. 1, 2

Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. In *Proc. NeurIPS*, 2020. 1

Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning. *arXiv preprint arXiv:2002.04809*, 2020. 1, 3

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proc. CVPR*, 2023. 2, 9

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Proc. NeurIPS*, 2020. 1

Günther Schindler, Wolfgang Roth, Franz Pernkopf, and Holger Fröning. Parameterized structured pruning for deep neural networks. In *Proc. LOD*, 2020. 3

Seonguk Seo, Dongwan Kim, and Bohyung Han. Revisiting machine unlearning with dimensional alignment. In *Proc. WACV*. IEEE, 2025. 14

Reza Shirkavand, Peiran Yu, Shangqian Gao, Gowthami Somepalli, Tom Goldstein, and Heng Huang. Efficient fine-tuning and concept suppression for pruned diffusion models. In *Proc. CVPR*, 2025. 1, 3, 6, 7, 16

Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX security symposium*, 2021. 14

Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. CP-ViT: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570*, 2022. 3

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding factors influencing machine unlearning. In *Proc. EuroS&P*, 2022. 1, 2, 3, 17

Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. In *Network and Distributed System Security Symposium (NDSS)*, 2023. 17

Jing Wu, Trung Le, Munawar Hayat, and Mehrtash Harandi. Erasing undesirable influence in diffusion models. In *Proc. CVPR*, 2025. 1, 2

Zimeng Wu, Yunhong Wang, Donghao Wang, and Jiaxin Chen. Collaborative multi-mode pruning for vision-language models. In *Proc. CVPR*, 2026. 3

Yang Xiao, Gen Li, Jie Ji, Ruimeng Ye, Xiaolong Ma, and Bo Hui. The right to be forgotten in pruning: Unveil machine unlearning on sparse models. *arXiv preprint arXiv:2507.18725*, 2025. 1, 3, 6, 7, 14, 16

Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018. 1, 3

Jianjian Yin, Liulei Li, Tao Chen, Yi Chen, Yazhou Yao, and Wenguan Wang. Cut-vit: Task-specific model pruning via gram anchoring subspace consistency. In *Proc. ECCV*, 2026. 1, 3

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Proc. NeurIPS*, 2019. 4

Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *Proc. AAAI*, 2022. 3

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. NISP: Pruning networks using neuron importance score propagation. In *Proc. CVPR*, 2018. 1

Xiaojian Yuan, Tianyu Pang, Chao Du, Kejiang Chen, Weiming Zhang, and Min Lin. A closer look at machine unlearning for large language models. In *Proc. ICLR*, 2025. 2

Gong Zhang, Kai Wang, Xingqian Xu, Zhangyang Wang, and Humphrey Shi. Forget-me-not: Learning to forget in text-to-image diffusion models. In *Proc. CVPR*, 2024a. 1, 2

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *Proc. COLM*, 2024b. 2

Shengming Zhang, Le Zhang, Jingbo Zhou, Zhi Zheng, and Hui Xiong. LLM-Eraser: Optimizing large language model unlearning through selective pruning. In *Proc. KDD*, 2025. 1, 2, 3, 4, 5, 6, 7, 16

Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Aligned structured sparsity learning for efficient image super-resolution. In *Proc. NeurIPS*, 2021. 3, 4

Amber Yijia Zheng, Yu-Shan Tai, and Raymond A Yeh. Designing to forget: Deep semi-parametric models for unlearning. In *Proc. CVPR*, 2026. 1, 3, 14

Mingjian Zhu, Yehui Tang, and Kai Han. Vision transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021. 3

Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *Proc. NeurIPS*, 2020. 3

Appendix

The appendix is organized as follows:

- The background comparison is in Sec. A1.
- The implementation and experimental details are presented in Sec. A2.
- Additional experiment results for **classification** are reported in Sec. A3.
- Additional experiment results for **class-conditioned generation** are reported in Sec. A4.
- Additional experiment results for **global pruning** are reported in Sec. A5.

A1 Background Comparison

Table A1: **Comparison with related works.** Typical pruning methods consist of three stages of sparsity training, pruning, and post-pruning fine-tuning. Prior works have mainly focused on making the fine-tuning stage unlearn-aware.

Related Work	Unlearn-aware Stages		
	Sparsity	Pruning	Fine-tuning
ℓ_1-MU			✓
Un-pruning			✓
Bilevel			✓
LLM-Eraser		✓	✓
PruneForget (Ours)	✓	✓	✓

As summarized in Tab. A1, prior approaches only partially integrate unlearning into the three stages of pruning (sparse training, pruning, post-pruning fine-tuning). In contrast, our proposed PruneForget makes all three stages unlearn-aware to further improve the unlearning performance of pruned models.

A2 Implementation and Experiment Details

We now provide the implementation details for the pruning methods, PruneForget, and baselines. For any unspecified training hyperparameters, we apply the default settings of the respective pruning methods. We adopt the pruning ratios specified in the original methods and apply comparable values for models not evaluated in the original studies.

As we apply uniform pruning, where an equal proportion of channels is removed across all layers, all baselines share identical pruned architectures and achieve the same operational speedup. Parameter counts and MACs for each pruned configuration are detailed in Tab. A2. We follow the official implementation of SFRon (Huang et al., 2024) to obtain the pre-trained classifiers utilized to evaluate Unlearned Accuracy (UA) in generative tasks. Consistent with their evaluation protocol, we synthesize 1,000 images per class to compute both the FID and UA metrics for DDPM on CIFAR-10. For DiT on ImageNet-1K, we sample 500 images for the unlearn class and randomly sample 10,000 images from the retain classes. We omit Membership Inference Attack (MIA) (Song & Mittal, 2021) evaluation, as recent studies (Seo et al., 2025; Xiao et al., 2025; Zheng et al., 2026) demonstrate it is an unreliable metric for assessing data removal. The runtime is measured on a single NVIDIA L40S GPU.

Details for pruning methods. Our method works with various pruning methods. For completeness, we document the details for these pruning methods.

- **Group-level Pruning (Fang et al., 2023a):** This baseline constructs parameter groups $\mathcal{G} = \{g\}$, assigning layers l into groups g based on their dependencies in the computational graph. The method employs a standard three-stage pipeline consisting of sparse training, structural pruning, and post-pruning fine-tuning. Group regularization is applied using the weighted L_2-norm of the parameter groups, which simultaneously serves as the criterion for the importance weights. Following their implementation, both T_s and T_f are set to 100 epochs. The model is optimized using stochastic gradient descent (SGD) with a learning rate of 0.01, a momentum of 0.9, and no weight decay, decayed via a cosine annealing scheduler.

Table A2: **Model compute reduction after pruning.** We report MACs, parameter counts, and theoretical speedup across pruned models (with different pruning ratios p). Here, we follow Fang et al. (2023a) and report the theoretical speedup $\triangleq \frac{\text{MACs(base)}}{\text{MACs(pruned)}}$.

Model	Dataset	Input	p	MACs	Params	Speedup
ResNet-18	CIFAR-10 / CIFAR-100	32×32	0	556.6M	11.17M	1.00×
			0.3	269.4M	5.46M	2.07×
			0.5	139.9M	2.80M	3.98×
			0.7	49.8M	1.00M	11.17×
Swin-T	TinyImageNet	224×224	0	4.36G	27.67M	1.00×
			0.2	3.56G	20.19M	1.22×
DDPM	CIFAR-10	32×32	0	6.06G	36.08M	1.00×
			0.3	2.07G	14.05M	2.93×
DiT-XL/2	ImageNet	256×256	0	118.74G	675.13M	1.00×
			0.3	51.81G	309.46M	2.29×

Table A3: **Hyperparameter search of f_{base} on CIFAR-10 (ResNet-18) random unlearning.** We use grid search to determine the most suitable unlearning frequencies and identify $f_{base} = 20$ and $f_{base} = 80$ as the optimal settings for 10% and 50% unlearning, respectively.

f_{base}	Random Unlearn 10%						Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
20	**0.03** ($6.01_{\pm 0.28}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.01** ($93.73_{\pm 0.22}$)	**0.01**	**0.22**	998	1.43 ($9.03_{\pm 1.45}$)	0.01 ($99.99_{\pm 0.01}$)	0.67 ($91.54_{\pm 0.75}$)	0.70	0.30	609
50	***0.06*** ($6.09_{\pm 0.42}$)	**0.00** ($100.00_{\pm 0.00}$)	0.25 ($93.49_{\pm 0.26}$)	0.10	**0.22**	996	0.57 ($8.17_{\pm 0.23}$)	**0.00** ($100.00_{\pm 0.00}$)	0.21 ($92.00_{\pm 0.28}$)	0.26	0.27	603
80	0.12 ($5.92_{\pm 0.49}$)	**0.00** ($100.00_{\pm 0.00}$)	***0.07*** ($93.67_{\pm 0.24}$)	***0.06***	**0.22**	986	**0.16** ($7.44_{\pm 0.32}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.02** ($92.22_{\pm 0.23}$)	**0.06**	**0.25**	594
100	0.20 ($5.84_{\pm 0.40}$)	**0.00** ($100.00_{\pm 0.00}$)	0.16 ($93.59_{\pm 0.19}$)	0.12	**0.22**	973	***0.19*** ($7.41_{\pm 0.36}$)	**0.00** ($100.00_{\pm 0.00}$)	***0.04*** ($92.24_{\pm 0.22}$)	***0.07***	**0.25**	579

- **Isomorphic Pruning (Fang et al., 2024):** This baseline groups sub-structures into various isomorphic groups and performs importance ranking within these groups. The method employs a two-stage pipeline consisting of structural pruning and post-pruning fine-tuning, utilizing a Taylor-based pruning framework. We set T_f to 50 epochs. Following their implementation, the model is optimized using AdamW with a learning rate of 5×10^{-4} and a weight decay of 0.05, decayed via a cosine annealing scheduler.
- **Diff-Pruning (Fang et al., 2023b):** This is a structural pruning method specialized for diffusion models. It employs a two-stage pipeline consisting of structural pruning followed by post-pruning fine-tuning. To calculate importance weights, the method accumulates first-order gradients over critical denoising steps, specifically focusing on larger timesteps. T_f and the learning rate are set to 100k iterations and 2×10^{-4} for DDPM, and 30k iterations and 1×10^{-4} for DiT. Both models are optimized using Adam with zero weight decay.

Details for PruneForget. The standardization of the sensitivity score is computed as

$$S'_{l[k]} = \text{Standardize}(S_{l[k]}) = 1 + \frac{S_{l[k]}}{S^{\text{median}}_{l[k]} + \epsilon}, \tag{A12}$$

where ϵ is set to 10^{-8}. For the unlearning loss $\mathcal{L}_{\mathtt{U}}$ applied in Stages 1 and 3, we follow Huang et al. (2024) and apply a gradient-based masked parameter update. We set the mask sparsity to 0.5, meaning 50% of the weights are updated by $\mathcal{L}_{\mathtt{U}}$. Considering that directly applying $\mathcal{L}_{\mathtt{U}}$ from the very beginning of a pruned model can destabilize the recovery of retain-set performance, we apply $\mathcal{L}_{\mathtt{U}}$ during the interval $0.5T_f < t < 0.9T_f$ in Stage 3 for generative models, incorporating an early stopping mechanism when the pixel-wise mean squared error for $\mathcal{U}$ exceeds 0.2. We choose $f \triangleq f^{\text{base}} \cdot |\mathcal{D} \setminus \mathcal{U}|/|\mathcal{U}|$ and the unlearning loss weight $\alpha \triangleq |\mathcal{U}|/|\mathcal{D} \setminus \mathcal{U}|$, to counteract the imbalances in the dataset sizes. We perform a grid search over $f^{\text{base}} \in \{20, 50, 80, 100\}$ for ResNet-18, with the detailed search results reported in Tab. A3 and Tab. A4. Following the same strategy for the other architectures, we set $f^{\text{base}} = 3$ for Swin-T and $f^{\text{base}} = 5$ for both DDPM and DiT. The regularization weight λ is set to 1×10^{-3}, and the supervised loss weight η follows the learning rate of the pruning mechanism. For the updating sensitivity variant evaluated in Tab. 6, sensitivity scores are calculated via a running mean over 20 batches of data and updated every 10 epochs throughout Stage 1.

Details for baselines. Our experimental framework is based on the officially released code of SFRon (Huang et al., 2024). We follow their implementation to obtain the pre-trained models,

Table A4: **Hyperparameter search of f_{base} on CIFAR-10 (ResNet-18) class unlearning.** Grid search results indicate that $f_{base} = 100$ achieves the best performance for both settings.

f_{base}	Unlearn 1 Class (10%)						Unlearn 5 Classes (50%)					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
20	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	0.51 ($93.76_{\pm 0.68}$)	0.17	***0.28***	992	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	0.42 ($95.96_{\pm 1.23}$)	0.14	0.40	602
50	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	***0.45*** ($93.82_{\pm 0.83}$)	***0.15***	**0.28**	984	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	0.27 ($96.10_{\pm 1.29}$)	0.09	***0.39***	600
80	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.41** ($93.86_{\pm 0.71}$)	**0.14**	**0.28**	984	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	***0.25*** ($96.13_{\pm 1.28}$)	***0.08***	0.40	600
100	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.41** ($93.86_{\pm 0.71}$)	**0.14**	**0.28**	984	**0.00** ($100.00_{\pm 0.00}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.20** ($96.17_{\pm 1.21}$)	**0.07**	**0.38**	596

retrained models, and the core SFRon unlearning algorithm. For the remaining baselines, we refer to their respective papers and properly adapt them to our experimental settings.

- **ℓ_1-MU (Jia et al., 2023):** For ResNet-18 on CIFAR-10, we employ an ℓ_1 penalty as the regularizer $\mathcal{R}(\theta)$ during Stage 1, followed by the two standard pruning stages. For Swin-T and DDPM, since the applied pruning methods consist of only the latter two stages, we introduce an additional 10-epoch ℓ_1 regularization phase before Stages 2 and 3. We apply a linear decay to the regularization coefficient λ, as detailed in their original paper. We compute the importance weights using an ℓ_1-based group magnitude metric.
- **Un-pruning (Xiao et al., 2025):** The un-pruning process operates through an N-step iterative cycle of weight re-initialization, k epochs of unlearning via fine-tuning (FT), and soft pruning, culminating in a final physical pruning step. We utilize FT as the unlearning algorithm due to its superior empirical results. To ensure the total computational runtime remains comparable to our default pruning baselines, we configure the iterative process to 20 steps of 10 epochs for ResNet-18, and 3 steps of 25 epochs for both Swin-T and DDPM. Finally, following the magnitude-based strategy of the original paper, we compute structural importance using an ℓ_2-norm group magnitude metric.
- **Bilevel (Shirkavand et al., 2025):** This baseline combines an unlearning loss with knowledge distillation from the pre-trained network in the post-pruning training stage. Because structured pruning inherently alters the network's intermediate feature dimensions, we omit feature-level distillation and rely exclusively on logit-based distillation. For the unlearning objective, we employ gradient ascent for classification tasks and random labeling for generative tasks. Following the original authors' configuration, we train the model for 20,000 iterations, executing 20 lower steps between two upper steps.
- **LLM-Eraser (Zhang et al., 2025):** This framework alternates between iterative pruning and unlearning steps. We utilize the authors' proposed Taylor score-based method to determine weight importance. For the unlearning phase, we strictly follow their algorithmic design, optimizing a joint objective that comprises generic selective training, KL divergence minimization, and contrastive disentanglement. To ensure the total computational runtime remains comparable, we configure the iterative process to 10 steps of 10 epochs for ResNet-18, 1 step of 25 epochs for Swin-T, 8 steps of 10 epochs for DDPM, and 4 steps of 2 epochs for DiT.

A3 Additional Results for Classification

Comparison with pure unlearning task (without pruning). In this section, we contrast our joint task of unlearning and pruning with pure unlearning paradigms, which update model weights without modifying the original architecture. Tab. A5 presents the results for 10% random unlearning on CIFAR-10. Note that these two tasks require different oracle models for evaluation. Pure unlearning targets a standard retrained model. Differently, our oracle for the joint task is a model trained from scratch and then pruned. We observe that sequentially applying pruning to the best-performing pure unlearning baseline (3SP→SFRon and SFRon→3SP) yields suboptimal results. In contrast, our proposed method achieves significantly better performance in this combined setting.

Comparison with prune-then-retrain baseline. While we treat the retrain-then-prune baseline as our oracle, we evaluate an additional baseline, *prune-then-retrain*, for random 50% unlearning on CIFAR-10 in Tab. A6. Specifically, we retrain a pruned model from scratch. Thus, the primary difference between `Pretrain→3SP` and `Prune→retrain` is that the latter reinitializes the weight parameters before the fine-tuning stage. In Tab. A6, we observe that retraining the pruned models results in lower utility, as the TA drops by 3.86% (89.15% *vs.* 93.01%) compared to the model fine-

Table A5: **Comparison with pure unlearning.** We compare pure unlearning (baselines include FT (Warnecke et al., 2023), GA (Thudi et al., 2022), SalUn (Fan et al., 2024), BT (Chundawat et al., 2023), SCRUB (Kurmanji et al., 2023), and SFRon (Huang et al., 2024)) against unlearning with pruning on a 10% random subset of CIFAR-10. Note that these are inherently distinct tasks evaluated against different respective oracles.

Method	Random Unlearn 10%				
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓
Task: Pure Unlearning (No Pruning)					
Retrain	0.00 (4.38$_{\pm 0.25}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (95.34$_{\pm 0.08}$)	0.00	0.00
FT	4.28 (0.10$_{\pm 0.05}$)	**0.01** (99.99$_{\pm 0.00}$)	**0.39** (94.94$_{\pm 0.15}$)	1.56	0.26
GA	***1.71*** (6.09$_{\pm 1.67}$)	6.24 (93.76$_{\pm 1.89}$)	8.34 (87.00$_{\pm 1.64}$)	5.43	0.36
SalUn	4.38 (0.00$_{\pm 0.01}$)	**0.01** (99.99$_{\pm 0.01}$)	***0.45*** (94.89$_{\pm 0.09}$)	1.61	0.27
BT	3.26 (1.12$_{\pm 0.00}$)	**0.01** (99.99$_{\pm 0.00}$)	0.71 (94.63$_{\pm 0.06}$)	***1.33***	***0.24***
SCRUB	3.82 (0.56$_{\pm 0.31}$)	***0.12*** (99.88$_{\pm 0.08}$)	1.20 (94.13$_{\pm 0.35}$)	1.71	0.25
SFRon	**0.96** (3.42$_{\pm 0.77}$)	***0.12*** (99.88$_{\pm 0.16}$)	1.15 (94.19$_{\pm 0.33}$)	**0.74**	**0.15**
Task: Unlearning + Pruning					
Retrain→3SP	0.00 (6.03$_{\pm 0.56}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (93.74$_{\pm 0.32}$)	0.00	0.00
Pretrain→3SP	0.89 (5.15$_{\pm 0.38}$)	**0.00** (100.00$_{\pm 0.00}$)	0.41 (94.15$_{\pm 0.20}$)	0.43	**0.20**
3SP→SFRon	14.84 (20.88$_{\pm 0.77}$)	1.88 (98.12$_{\pm 0.36}$)	3.38 (90.37$_{\pm 0.37}$)	6.70	0.24
SFRon→3SP	***0.16*** (5.87$_{\pm 0.57}$)	**0.00** (100.00$_{\pm 0.01}$)	***0.04*** (93.70$_{\pm 0.31}$)	***0.07***	***0.22***
PruneForget	**0.03** (6.01$_{\pm 0.28}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.01** (93.73$_{\pm 0.22}$)	**0.01**	***0.22***

Table A6: **Comparison with pruning then retraining from scratch.** We evaluate the Prune→retrain baseline on CIFAR-10 50% random unlearning. The results demonstrate that pruned models are inherently difficult to optimize from a random initialization.

Method	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time (s)
Retrain→3SP	0.00 (7.60$_{\pm 0.22}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (92.20$_{\pm 0.30}$)	0.00	0.00	1518
Pretrain→3SP	1.54 (6.06$_{\pm 0.17}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.81*** (93.01$_{\pm 0.27}$)	***0.78***	**0.23**	556
Prune→retrain	2.05 (9.65$_{\pm 0.19}$)	0.01 (99.99$_{\pm 0.01}$)	3.05 (89.15$_{\pm 0.24}$)	1.70	0.35	598
PruneForget	**0.16** (7.44$_{\pm 0.32}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.02** (92.22$_{\pm 0.23}$)	**0.06**	***0.25***	594

tuned without reinitialization. This observation aligns with the Lottery Ticket Hypothesis (Frankle & Carbin, 2019), which describes that pruned models are difficult to train from scratch.

Comparison under different pruning ratios. To evaluate the robustness of our method across varying degrees of sparsity, we compare PruneForget against baselines at both lower ($p = 0.3$) and higher ($p = 0.7$) pruning ratios for random unlearning on CIFAR-10 with ResNet-18. As shown in Tab. A7, PruneForget consistently achieves the lowest Avg Δ under a low pruning ratio, recording an Avg Δ of 0.11 and 0.07 for the 10% and 50% unlearning scenarios, respectively. Furthermore, Tab. A8 demonstrates that PruneForget maintains a small gap to the oracle even under a higher pruning ratio. It achieves an Avg Δ of 0.01 in the 10% unlearning setting, while keeping the gap highly competitive (Avg Δ = 0.44) when unlearning 50% of the data. These results highlight the stability and generalizability of PruneForget, proving its effectiveness across different compression regimes.

Experiment results on CIFAR-100 (ResNet-18). While we have demonstrated the effectiveness of PruneForget for ResNet-18 on CIFAR-10, we further explore its generalizability across more complex datasets. Tab. A9 presents the results for 50% random unlearning on the CIFAR-100 dataset. PruneForget outperforms all baselines, achieving the lowest Avg Δ of just 0.72. It optimally balances unlearning and utility, recording the lowest gaps in unlearn (ΔUA = 1.46) and test accuracy (ΔTA = 0.69) while perfectly preserving retain accuracy (ΔRA = 0.00). In contrast, the closest baseline (SFRon→3SP) trails at Avg Δ = 1.20, and other baselines suffer severe degradation. Furthermore, PruneForget achieves these near-oracle results highly efficiently, operating over 4.3× faster than the retrain oracle.

Table A7: **CIFAR-10 (ResNet-18) random unlearning on classification with $p = 0.3$.** Among all baselines, PruneForget achieves the lowest Avg Δ under a low pruning ratio.

Method	Random Unlearn 10%						Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 (5.15$_{\pm 0.31}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (94.75$_{\pm 0.18}$)	0.00	0.00	2212	0.00 (6.98$_{\pm 0.20}$)	0.00 (100.00$_{\pm 0.00}$)	0.00 (92.78$_{\pm 0.20}$)	0.00	0.00	1494
Pretrain→3SP	1.14 (4.01$_{\pm 0.10}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.05** (94.80$_{\pm 0.14}$)	***0.40***	**0.14**	1057	2.40 (4.58$_{\pm 0.12}$)	**0.00** (100.00$_{\pm 0.00}$)	1.04 (93.82$_{\pm 0.16}$)	1.15	**0.20**	638
3SP→SFRon	15.17 (20.32$_{\pm 0.61}$)	1.78 (98.22$_{\pm 0.13}$)	4.10 (90.65$_{\pm 0.23}$)	7.02	0.21	1171	5.30 (12.29$_{\pm 0.29}$)	0.41 (99.59$_{\pm 0.05}$)	2.90 (89.88$_{\pm 0.15}$)	2.87	0.25	698
SFRon→3SP	***1.10*** (6.25$_{\pm 0.41}$)	**0.00** (100.00$_{\pm 0.00}$)	1.35 (93.40$_{\pm 0.23}$)	0.82	0.20	1167	***0.14*** (6.84$_{\pm 0.16}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.22*** (92.55$_{\pm 0.29}$)	***0.12***	0.25	707
ℓ_1-MU→2SP	5.94 (11.09$_{\pm 0.26}$)	5.20 (94.80$_{\pm 0.29}$)	5.95 (88.80$_{\pm 0.31}$)	5.69	0.27	931	5.46 (12.44$_{\pm 0.18}$)	3.25 (96.75$_{\pm 0.26}$)	5.41 (87.37$_{\pm 0.21}$)	4.71	0.33	545
Un-pruning	1.92 (7.07$_{\pm 0.33}$)	***0.02*** (99.98$_{\pm 0.01}$)	2.12 (92.63$_{\pm 0.11}$)	1.35	0.35	914	2.54 (9.52$_{\pm 0.12}$)	***0.03*** (99.97$_{\pm 0.01}$)	2.71 (90.06$_{\pm 0.27}$)	1.76	0.43	553
Bilevel	8.49 (13.64$_{\pm 0.75}$)	6.98 (93.02$_{\pm 0.96}$)	7.97 (86.78$_{\pm 0.86}$)	7.81	0.54	1023	6.73 (13.72$_{\pm 1.12}$)	2.94 (97.06$_{\pm 0.91}$)	6.70 (86.08$_{\pm 1.04}$)	5.46	0.61	722
LLM-Eraser	13.51 (18.66$_{\pm 2.64}$)	16.53 (83.47$_{\pm 2.39}$)	13.76 (80.99$_{\pm 2.28}$)	14.60	0.53	1589	10.30 (17.29$_{\pm 1.51}$)	13.02 (86.98$_{\pm 1.68}$)	10.29 (82.49$_{\pm 1.62}$)	11.20	0.45	853
PruneForget	**0.01** (5.14$_{\pm 0.29}$)	**0.00** (100.00$_{\pm 0.00}$)	***0.32*** (94.43$_{\pm 0.15}$)	**0.11**	***0.16***	1103	**0.07** (6.92$_{\pm 0.14}$)	**0.00** (100.00$_{\pm 0.00}$)	**0.16** (92.94$_{\pm 0.15}$)	**0.07**	***0.22***	651

Table A8: **CIFAR-10 (ResNet-18) random unlearning on classification with $p = 0.7$.** PruneForget achieves a small gap to the oracle under a high pruning ratio.

Method	Random Unlearn 10%						Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 (8.39$_{\pm 0.31}$)	0.00 (99.89$_{\pm 0.00}$)	0.00 (91.34$_{\pm 0.18}$)	0.00	0.00	2131	0.00 (9.71$_{\pm 0.20}$)	0.00 (99.98$_{\pm 0.00}$)	0.00 (90.09$_{\pm 0.20}$)	0.00	0.00	1454
Pretrain→3SP	0.49 (7.90$_{\pm 0.38}$)	**0.03** (99.92$_{\pm 0.02}$)	0.40 (91.74$_{\pm 0.23}$)	***0.31***	**0.14**	991	**0.05** (9.65$_{\pm 0.32}$)	**0.01** (99.97$_{\pm 0.01}$)	**0.19** (89.90$_{\pm 0.31}$)	**0.08**	**0.20**	597
3SP→SFRon	13.88 (22.26$_{\pm 0.94}$)	7.48 (92.41$_{\pm 1.08}$)	4.91 (86.43$_{\pm 0.93}$)	8.76	0.21	1097	6.47 (16.18$_{\pm 0.58}$)	5.39 (94.59$_{\pm 0.97}$)	5.09 (85.00$_{\pm 0.46}$)	5.65	0.25	618
SFRon→3SP	1.28 (9.66$_{\pm 0.74}$)	0.50 (99.39$_{\pm 0.37}$)	1.11 (90.23$_{\pm 0.70}$)	0.96	0.20	1120	2.43 (12.14$_{\pm 0.42}$)	0.33 (99.65$_{\pm 0.09}$)	2.62 (87.47$_{\pm 0.46}$)	1.79	0.25	668
ℓ_1-MU→2SP	3.17 (11.56$_{\pm 0.46}$)	5.76 (94.13$_{\pm 0.33}$)	3.05 (88.29$_{\pm 0.32}$)	3.99	0.27	859	3.38 (13.09$_{\pm 0.25}$)	4.17 (95.81$_{\pm 0.34}$)	3.42 (86.68$_{\pm 0.30}$)	3.66	0.33	505
Un-pruning	1.73 (10.12$_{\pm 0.41}$)	1.26 (98.63$_{\pm 0.46}$)	1.73 (89.62$_{\pm 0.41}$)	1.57	0.35	918	2.94 (12.65$_{\pm 0.40}$)	1.67 (98.30$_{\pm 0.40}$)	3.19 (86.90$_{\pm 0.52}$)	2.60	0.43	536
Bilevel	7.33 (15.72$_{\pm 1.43}$)	10.21 (89.68$_{\pm 1.74}$)	6.66 (84.69$_{\pm 1.58}$)	8.06	0.54	965	5.78 (15.48$_{\pm 0.82}$)	6.17 (93.81$_{\pm 0.96}$)	5.76 (84.33$_{\pm 0.96}$)	5.90	0.61	671
LLM-Eraser	11.92 (20.30$_{\pm 3.24}$)	18.20 (81.69$_{\pm 3.08}$)	11.92 (79.42$_{\pm 2.75}$)	14.01	0.53	1415	9.63 (19.33$_{\pm 1.33}$)	16.02 (83.96$_{\pm 1.23}$)	9.85 (80.24$_{\pm 1.25}$)	11.83	0.45	750
PruneForget	**0.01** (8.38$_{\pm 0.38}$)	**0.02** (99.87$_{\pm 0.03}$)	**0.01** (91.35$_{\pm 0.35}$)	**0.01**	***0.16***	1008	***0.51*** (10.21$_{\pm 0.32}$)	***0.04*** (99.94$_{\pm 0.03}$)	***0.78*** (89.31$_{\pm 0.35}$)	***0.44***	***0.22***	606

A4 Additional Results for Class-conditioned Generation

Results on class-wise unlearning on CIFAR-10 (DDPM). In Tab. A10, we show the class-wise unlearning results over 5 classes (*Automobile*, *Cat*, *Dog*, *Horse*, and *Truck*). We observe PruneForget effectively matches the oracle model across all tasks. It yields the lowest ΔUA in most cases alongside highly competitive ΔFID scores, demonstrating its ability to effectively unlearn targeted concepts while maintaining overall image quality. Importantly, PruneForget achieves these near-zero performance gaps while accelerating runtime by over 10× relative to the oracle model.

Results on class-wise unlearning on ImageNet (DiT-XL/2). In Tab. A11, we present the class-wise unlearning results. Following Huang et al. (2024), we omit the oracle (Retrain→2SP) due to computational constraints and use the raw values of UA and FID for comparison. We observe that PruneForget effectively unlearns the target class (achieving 100 UA in all cases) while maintaining high generation fidelity on the retain classes (average FID is 13.47). In contrast, baseline methods either fail to forget the specified concept (UA $<$ 10) or suffer catastrophic degradation in their generative capabilities (FID $>$ 100).

Ablation of early stopping. Tab. A12 shows the ablation of the early stopping mechanism mentioned in Sec. 4.2 for generative tasks, evaluating DDPM across five individual CIFAR-10 classes. The results highlight that without early stopping, the model is highly vulnerable to the destructive effects of an unconstrained unlearning loss. This aggressive unlearning catastrophically degrades overall image synthesis quality, evidenced by sharp increases in ΔFID (*e.g.*, from 0.13 to 6.64 for *Cat* and from 0.04 to 8.41 for *Horse*). This demonstrates that early stopping is important for diffusion unlearning, allowing the network to preserve high-fidelity generation capabilities while successfully eradicating targeted concepts.

Qualitative results on CIFAR-10 (DDPM). In Fig. A1, we show the images generated from the unlearned models on the unlearned classes: *Automobile*, *Cat*, *Dog*, *Horse*, and *Truck*. We observe that the pre-trained model generates recognizable images for all categories, while the oracle (Retrain→2SP) and PruneForget successfully erase the forgotten concepts. Specifically, for the unlearn classes (columns I1–I5), the outputs from PruneForget lose their original class semantics. Observe that the images become ambiguous or belong to another class in the retain classes, which closely mirrors the behavior of the oracle. Next, for the retain classes (columns C1–C9), PruneForget preserves high visual fidelity, producing images that are very similar to those generated by the oracle model. These visual results show that PruneForget effectively unlearns the class information while maintaining alignment with the oracle.

Qualitative results on ImageNet (DiT-XL/2). In Fig. A2, we present images generated by the unlearned models for the five evaluated unlearn classes: *Cacatua galerita*, *Golden retriever*, *White*

Table A9: **CIFAR-100 (ResNet-18) random unlearning on classification.** PruneForget achieves performance comparable to the oracle (Retrain→3SP), demonstrating its generalizability across more complex datasets.

Method	Random Unlearn 50%					
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time
Retrain→3SP	0.00 (32.48$_{\pm 0.31}$)	0.00 (99.99$_{\pm 0.01}$)	0.00 (67.40$_{\pm 0.18}$)	0.00	0.00	2551
Pretrain→3SP	3.22 (29.27$_{\pm 0.71}$)	**0.00** (99.99$_{\pm 0.01}$)	1.50 (68.90$_{\pm 0.69}$)	1.57	**0.20**	567
3SP→SFRon	5.39 (37.87$_{\pm 0.31}$)	0.44 (99.55$_{\pm 0.05}$)	1.67 (65.72$_{\pm 0.37}$)	2.50	0.25	631
SFRon→3SP	***2.12*** (30.37$_{\pm 0.28}$)	**0.00** (99.99$_{\pm 0.01}$)	***1.48*** (68.88$_{\pm 0.30}$)	***1.20***	0.25	635
ℓ_1-MU→2SP	6.28 (38.76$_{\pm 0.36}$)	7.59 (92.40$_{\pm 0.74}$)	6.13 (61.27$_{\pm 0.53}$)	6.67	0.33	467
Un-pruning	5.19 (37.67$_{\pm 0.35}$)	0.08 (99.90$_{\pm 0.03}$)	5.08 (62.32$_{\pm 0.55}$)	3.45	0.43	543
Bilevel	6.00 (38.48$_{\pm 1.03}$)	4.07 (95.92$_{\pm 0.71}$)	5.05 (62.35$_{\pm 0.87}$)	5.04	0.61	625
LLM-Eraser	14.43 (46.91$_{\pm 1.45}$)	33.41 (66.58$_{\pm 2.34}$)	14.45 (52.95$_{\pm 1.68}$)	20.76	0.45	793
PruneForget	**1.46** (33.94$_{\pm 1.24}$)	**0.00** (99.99$_{\pm 0.01}$)	**0.69** (66.70$_{\pm 0.76}$)	**0.72**	***0.22***	581

Table A10: **CIFAR-10 (DDPM) class-wise unlearning on generation.** PruneForget consistently achieves close performance to the oracle.

Method	Automobile		Cat		Dog		Horse		Truck		Time
	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	
Retrain→2SP	0.00 (98.87)	0.00 (17.10)	0.00 (95.50)	0.00 (15.96)	0.00 (98.79)	0.00 (16.26)	0.00 (99.12)	0.00 (17.54)	0.00 (99.66)	0.00 (16.78)	398622
Pretrain→2SP	0.65 (98.22)	**0.10** (17.20)	0.55 (94.95)	0.61 (16.57)	0.38 (98.41)	***0.07*** (16.33)	0.13 (98.99)	0.21 (17.33)	0.55 (99.11)	***0.52*** (17.30)	21304
2SP→SFRon	0.66 (98.21)	0.34 (17.44)	***0.15*** (95.35)	***0.13*** (16.09)	0.36 (98.43)	***0.07*** (16.19)	0.19 (98.93)	**0.02** (17.56)	0.59 (99.07)	0.53 (17.31)	25652
SFRon→2SP	0.81 (98.06)	***0.30*** (17.40)	0.78 (94.72)	**0.11** (16.07)	0.25 (98.54)	**0.05** (16.31)	0.16 (98.96)	0.21 (17.75)	0.77 (98.89)	**0.47** (17.25)	26674
ℓ_1-MU→2SP	***0.21*** (98.66)	1.03 (18.13)	0.39 (95.11)	0.55 (16.51)	***0.16*** (98.63)	0.63 (16.89)	***0.07*** (99.05)	0.49 (18.03)	**0.15** (99.51)	1.11 (17.89)	22675
Un-pruning	1.02 (99.89)	371.14 (388.24)	0.60 (94.90)	286.38 (302.34)	1.08 (99.87)	315.46 (331.72)	0.88 (100.00)	331.98 (349.52)	**0.15** (99.81)	326.22 (343.00)	36586
Bilevel	6.34 (92.53)	1.01 (18.11)	5.47 (90.03)	1.38 (17.34)	2.56 (96.23)	1.04 (17.30)	3.18 (95.94)	0.22 (17.76)	3.95 (95.71)	1.26 (18.04)	33863
LLM-Eraser	1.05 (97.82)	7.42 (24.52)	3.49 (92.01)	6.46 (22.42)	9.74 (89.05)	9.91 (26.17)	0.19 (98.93)	9.24 (26.78)	0.93 (98.73)	4.00 (20.78)	27570
PruneForget	**0.17** (98.70)	0.38 (17.48)	**0.13** (95.37)	***0.13*** (15.83)	**0.04** (98.75)	0.10 (16.36)	**0.04** (99.08)	***0.04*** (17.50)	***0.26*** (99.40)	0.63 (17.41)	26254

wolf, *Arctic fox*, and *Otter*. While the pre-trained model generates recognizable images for all categories, PruneForget successfully erases the forgotten concepts. Specifically, for the unlearn classes (columns I1–I2), the outputs from PruneForget degrade into pure noise. Conversely, for the retain classes (columns C1–C5), PruneForget preserves high visual fidelity, producing images that closely match those generated by the pre-trained model.

A5 Additional Results using Global Pruning

In previous sections, we evaluated our method under uniform pruning, which removes an equal proportion of channels per layer to create hardware-friendly architectures. To achieve better accuracy retention, an alternative strategy is global pruning, which pools parameters across all layers to adaptively preserve the network's most critical components, which is supported by Group-level Pruning (Fang et al., 2023a) and Isomorphic Pruning (Fang et al., 2024). Note that this approach results in varying MACs even at identical pruning ratios. We evaluate our method under this setting to demonstrate its effectiveness in a global pruning scenario, which also shows the structural impact of unlearn-aware pruning.

We compare PruneForget against the unlearn-unaware baselines (Pretrain→nSP) on ResNet-18 and Swin-T under 50% random unlearning. As shown in Tab. A13 and Tab. A14, PruneForget consistently attains a lower Avg Δ despite operating with fewer or comparable MACs.

Next, we analyze the channel sparsity distributions from the different methods. In Fig. A3, we visualize the channel distribution for ResNet-18. We observe that integrating the unlearning objective directly into pruning yields different sparse architectures. PruneForget retains noticeably more channels in the deeper stages, whereas the baseline aggressively sparsifies the final layer. For Swin-T shown in Fig. A4, PruneForget prunes the attention (QKV) projections more heavily in both the initial and final blocks. While we did not observe an interpretable reason as to why PruneForget prunes these layers, this finding seems to be consistent with existing works on unlearning, where they also focus on optimizing the attention layers (Gandikota et al., 2023).

Table A11: **ImageNet-1K (DiT-XL/2) class-wise unlearning on generation.** PruneForget uniquely achieves perfect forgetting while maintaining generative quality. **Averages** (shaded) across the 5 classes are reported in the summary section. We omit the oracle (Retrain→2SP) due to computational constraints. Best results are in **blue**, while second best are ***italicized***.

Method	Cacatua galerita		Golden retriever		White wolf		Arctic fox		Otter		Average		Time
	UA ↑	FID ↓	UA ↑	FID ↓	UA ↑	FID ↓	UA ↑	FID ↓	UA ↑	FID ↓	UA ↑	FID ↓	
Pretrain→2SP	4.2	***12.51***	2.0	**12.65**	2.6	**12.74**	***9.8***	**12.60**	3.2	***12.82***	$4.36_{\pm 3.15}$	$\mathbf{12.66}_{\pm 0.12}$	35296.4
2SP→SFRon	**100.0**	115.01	**100.0**	115.68	**100.0**	21.62	**100.0**	316.17	**100.0**	22.25	$\mathbf{100.0}_{\pm 0.00}$	$118.15_{\pm 120.15}$	35930.8
SFRon→2SP	***5.4***	**12.36**	***2.8***	***12.73***	***8.8***	***13.18***	5.0	***12.87***	***10.8***	**12.80**	$\mathbf{6.56}_{\pm 3.20}$	$\mathbf{12.79}_{\pm 0.29}$	37440.7
LLM-Eraser	**100.0**	564.48	**100.0**	359.76	**100.0**	564.48	**100.0**	357.60	**100.0**	315.15	$\mathbf{100.0}_{\pm 0.00}$	$432.29_{\pm 121.97}$	824133.9
PruneForget	**100.0**	12.95	**100.0**	13.68	**100.0**	13.40	**100.0**	13.48	**100.0**	13.84	$\mathbf{100.0}_{\pm 0.00}$	$13.47_{\pm 0.34}$	35765.0

Table A12: **Ablation of early stopping on generation.** We evaluate the impact of the early stopping mechanism on DDPM for CIFAR-10 class-wise unlearning. PruneForget consistently maintains tighter ΔUA and ΔFID gaps relative to the oracle.

Method	Automobile		Cat		Dog		Horse		Truck		Time
	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	ΔUA ↓	ΔFID ↓	
Retrain→2SP	0.00 (98.87)	0.00 (17.10)	0.00 (95.50)	0.00 (15.96)	0.00 (98.79)	0.00 (16.26)	0.00 (99.12)	0.00 (17.54)	0.00 (99.66)	0.00 (16.78)	398622
w/o Early Stop	5.60 (93.27)	3.21 (20.31)	1.00 (96.50)	6.64 (22.60)	0.15 (98.64)	4.82 (21.08)	0.20 (99.32)	8.41 (25.95)	**0.04** (99.62)	1.19 (17.97)	26396
PruneForget	**0.17** (98.70)	**0.38** (17.48)	**0.13** (95.37)	**0.13** (15.83)	**0.04** (98.75)	**0.10** (16.36)	**0.04** (99.08)	**0.04** (17.50)	0.26 (99.40)	**0.63** (17.41)	26254

Figure A1: **Qualitative results of CIFAR-10 (DDPM) class-wise unlearning.** We show generated samples for the unlearn classes (*Automobile*, *Cat*, *Dog*, *Horse*, and *Truck*) alongside several retain classes. We observe that PruneForget successfully unlearns the class concept, *e.g.*, when the pre-trained model successfully generated a car, PruneForget and the oracle instead generate an ambiguous object. Overall, we observe that PruneForget consistently generates images that closely match the oracle (Retrain→2SP). These observations are consistent with the quantitative metrics.

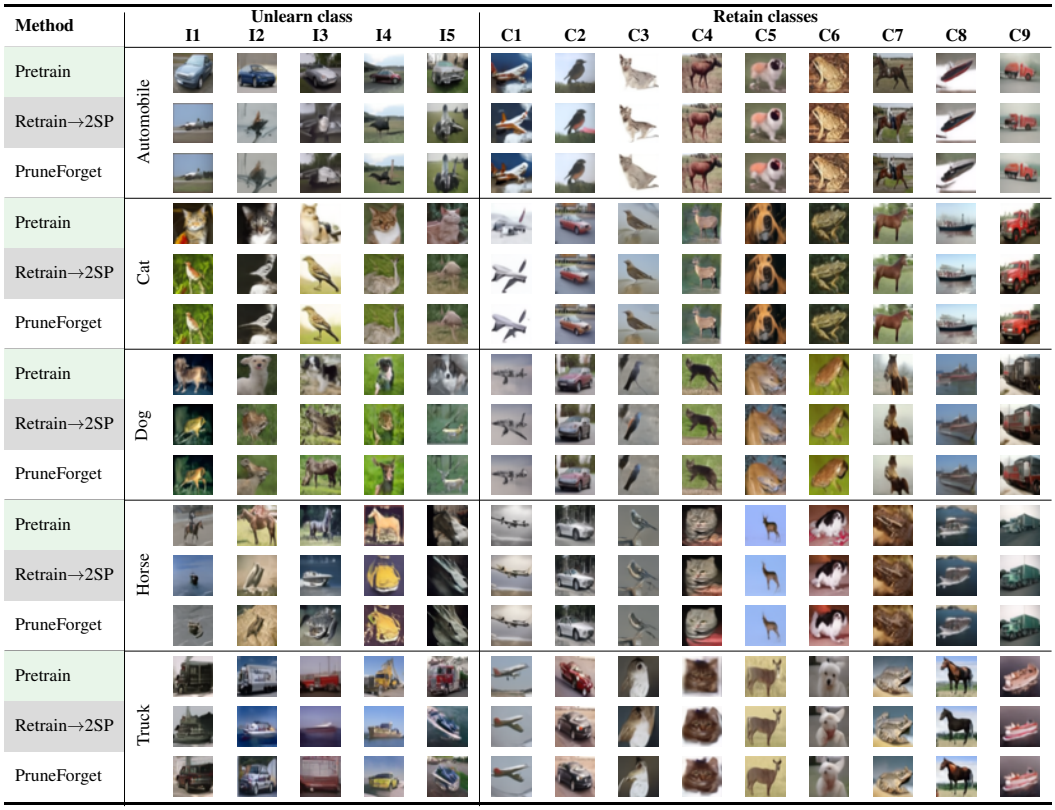


Figure A2: **Qualitative results of ImageNet (DiT-XL/2) class-wise unlearning.** We show generated samples for the 5 evaluated unlearn classes (*Cacatua galerita*, *Golden retriever*, *White wolf*, *Arctic fox*, and *Otter*) alongside several retain classes. PruneForget achieves perfect forgetting (by generating noise for the unlearn class) while maintaining generative quality in the retain classes (matching the pre-trained model). We omit the oracle due to computational constraints.

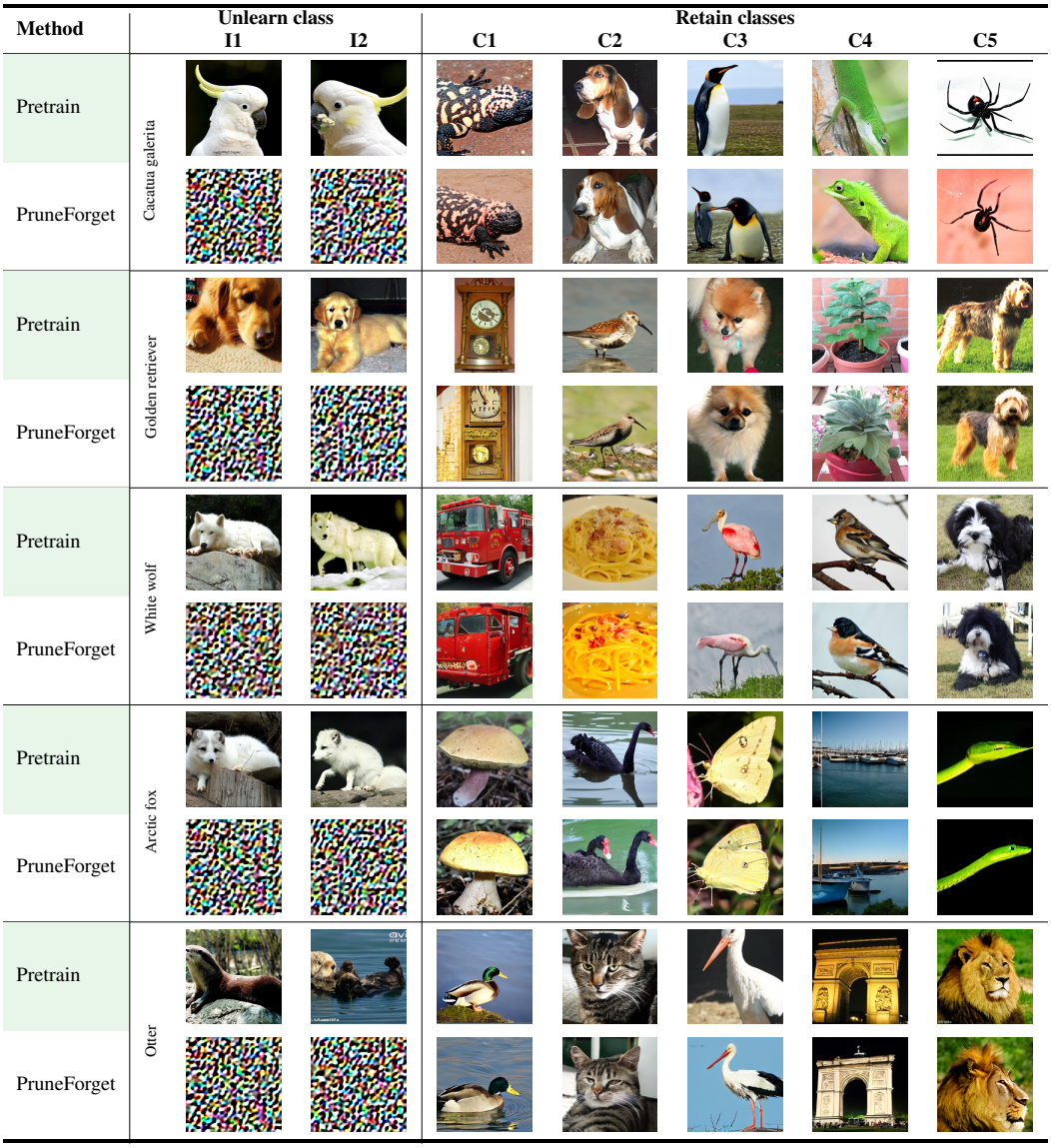
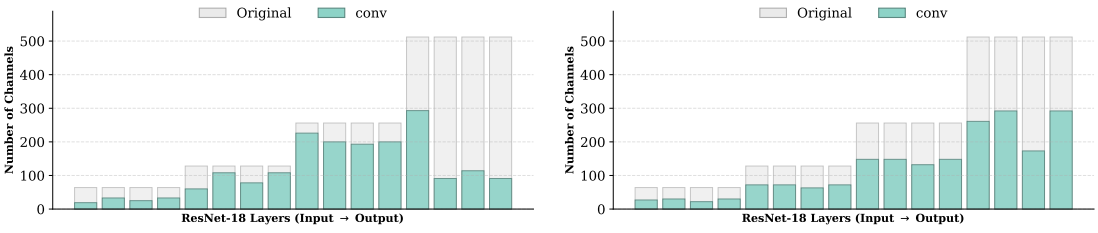


Table A13: **CIFAR-10 (ResNet-18) under global pruning.** Evaluating PruneForget under global pruning on 50% random unlearning demonstrates superior alignment to the oracle (**Avg Δ**) and higher speedup compared to the unlearning-unaware baseline.

Method	Random Unlearn 50%							
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	MACs	Speedup
Retrain→3SP	0.00 ($7.21_{\pm 0.25}$)	0.00 ($100.00_{\pm 0.00}$)	0.00 ($92.54_{\pm 0.29}$)	0.00	0.00	1471	175.5M	3.17×
Pretrain→3SP	1.79 ($5.42_{\pm 0.14}$)	**0.00** ($100.00_{\pm 0.00}$)	0.99 ($93.53_{\pm 0.16}$)	0.93	**0.23**	596	186.9M	2.98×
PruneForget	**0.05** ($7.26_{\pm 0.29}$)	**0.00** ($100.00_{\pm 0.00}$)	**0.12** ($92.42_{\pm 0.29}$)	**0.06**	0.24	648	137.4M	**4.05**×

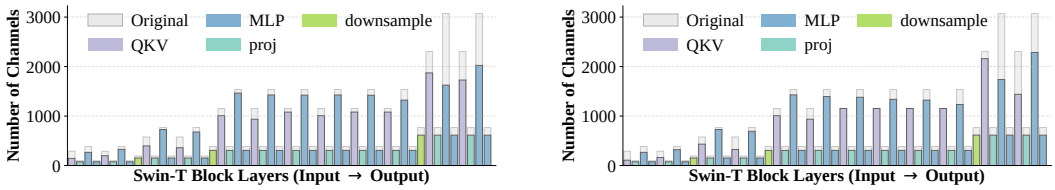
Table A14: **TinyImageNet (Swin-T) under global pruning.** Evaluating PruneForget under global pruning on 50% random unlearning shows lower **Avg Δ** than the baseline under the same speedup.

Method	Random Unlearn 50%							
	ΔUA ↓	ΔRA ↓	ΔTA ↓	Avg Δ ↓	D_{KL} ↓	Time	MACs	Speedup
Retrain→2SP	0.00 ($22.95_{\pm 0.18}$)	0.00 ($98.68_{\pm 0.07}$)	0.00 ($77.29_{\pm 0.21}$)	0.00	0.00	5429	2962.6M	1.50×
Pretrain→2SP	2.37 ($20.58_{\pm 0.14}$)	**0.22** ($98.90_{\pm 0.05}$)	0.27 ($77.56_{\pm 0.26}$)	0.95	**0.31**	3663	2960.4M	1.51×
PruneForget	**1.16** ($21.79_{\pm 0.17}$)	0.35 ($99.02_{\pm 0.05}$)	**0.12** ($77.17_{\pm 0.20}$)	**0.54**	0.37	4506	2942.9M	1.51×



(a) Pretrain→3SP (b) PruneForget

Figure A3: **Channel sparsity distribution of pruned ResNet-18.** In (a), the pruning process operates independently of the unlearning phase; conversely, in (b), the proposed PruneForget integrates unlearning awareness into the pruning stage. This distinction leads to noticeably different parameter distributions across the network. Specifically, PruneForget tends to preserve significantly more channels in the final stage.



(a) Pretrain→2SP (b) PruneForget

Figure A4: **Channel sparsity distribution of pruned Swin-T.** In (a), the pruning process operates independently of the unlearning phase; conversely, in (b), the proposed PruneForget integrates unlearning awareness into the pruning stage. This results in the difference in pruned parameter distributions across the model. Specifically, we observe that PruneForget prunes the initial QKV layer more aggressively than in (a). While the majority of pruning for both methods occurs toward the end of the model, (b) more significantly prunes the QKV layer at the end.