Breaking Weight Entanglement: Machine Unlearning with Nonlinearity

Yingdan Shi [1] Ren Wang [1]

Abstract

Machine Unlearning (MU) seeks to eliminate the influence of specific training data from a pre-trained model. One existing approach achieves unlearning via linear parameter updates by task arithmetic. However, linear parameter editing often suffers from the weight entanglement issue. In this work, we introduce an unlearning framework Mode Connectivity Unlearning (MCU), that leverages mode connectivity to discover a nonlinear unlearning pathway in parameter space. To boost both effectiveness and efficiency, we further incorporate a parameter masking strategy that enhances the forgetting process while reducing computational costs. Additionally, we present an adaptive mechanism for our unlearning penalty coefficient, which adaptively balances forgetting quality and model utility without the need for manual hyperparameter search. Distinct from existing MU techniques that produce a single unlearning model, MCU reveals multiple unlearning models along the pathway. Overall, MCU functions as a plug-and-play framework that can be integrated into all existing MU methods, consistently enhancing their unlearning performance.

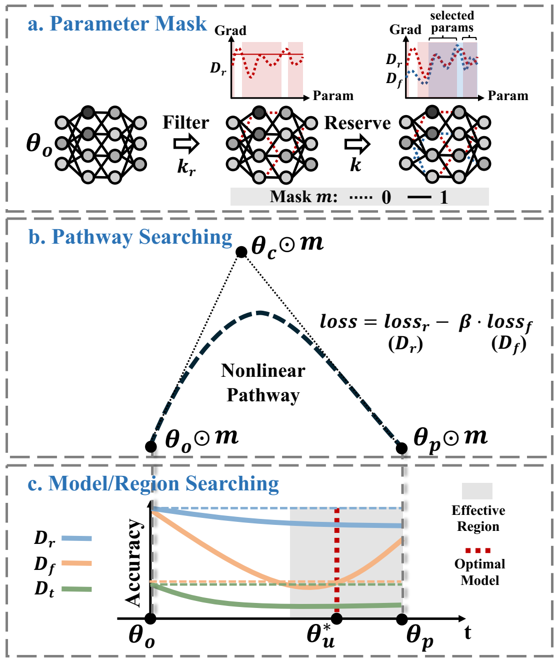


Figure 1: Overview of our MCU framework. **a.** Identify a parameter mask by first filtering out the top k_r proportion of parameters important to the retaining data, then reserving the top k proportion of parameters crucial to the forgetting data. **b.** Explore nonlinear pathways in the parameter space, where θ_c serves as the control point shaping the pathway. **c.** Locate the optimal unlearning model and an effective unlearning region along the pathway.

1. Introduction

Machine Unlearning (MU) is crucial for complying with privacy regulations and handling user-initiated data deletion. The most straightforward MU method retrains a model from scratch after removing the forgetting data, but this incurs substantial computational cost. To reduce this overhead, a variety of approximate MU methods (Fan et al., 2023; Graves et al., 2021; Ilharco et al., 2022; Kurmanji et al., 2024a; Thudi et al., 2022) have been proposed, aiming to balance unlearning effectiveness and efficiency.

A prominent direction in MU is task arithmetic, which *linearly* modifies model parameters by subtracting task vectors associated with the forgetting data (Ilharco et al., 2022; Ortiz-Jimenez et al., 2024). However, due to the nonlinear nature and high-dimensional complexity of neural networks, such linear updates can interfere with retaining data caused by **weight entanglement issue** (Ortiz-Jimenez et al., 2024). Thus, we try to break free from the constraints of linear updates and instead explore unlearning in a nonlinear manner. Moreover, most existing MU methods yield only a *single* unlearned model, limiting flexibility. A natural question is whether we can generate multiple unlearning models without repeated training. It enables us to select the solution that best aligns with our specific priority, such as prioritizing model utility preservation or forgetting quality.

To address these challenges, we propose a plug-and-play framework, **M**ode **C**onnectivity **U**nlearning (MCU), which explores MU in a *nonlinear* manner inspired by mode connectivity (Garipov et al., 2018; Wang et al., 2023; Ren et al.,

[1]Illinois Institute of Technology. Correspondence to: Ren Wang <rwang74@iit.edu>.

Published at the ICML 2025 Workshop on Machine Unlearning for Generative AI. Copyright 2025 by the author(s).

2025). As illustrated in Figure 1, MCU begins by constructing a parameter mask to isolate parameters crucial to forgetting, reducing interference with retained data (Figure 1a). A nonlinear path is then searched between the original and pre-unlearning models using our tailored loss (Figure 1b). Unlike prior work, MCU produces multiple unlearning models (Figure 1c), offering flexible trade-offs between forgetting and preserving. Overall, MCU can be seamlessly integrated with existing MU methods and consistently improves unlearning performance.

2. Mode Connectivity Unlearning

2.1. Preliminaries and Notations

In the context of MU for image classification, we consider two unlearning scenarios: **random data forgetting** and **class-wise forgetting**. We use $\mathcal{D}_f \in \mathcal{D}_{train}$ to denote the forgetting data and $\mathcal{D}_r = \mathcal{D}_{train} \backslash \mathcal{D}_f$ to denote the retaining data. The test data is denoted as $\mathcal{D}_t$. In the class-wise scenario, $\mathcal{D}_t = \mathcal{D}_{tr} \cup \mathcal{D}_{tf}$ where $\mathcal{D}_{tr}$ and $\mathcal{D}_{tf}$ are test-retaining data and test-forgetting data respectively. The objective of our work is to identify a pathway in the parameter space, where each point along the pathway corresponds to an unlearning model, denoted by $\boldsymbol{\theta}_u$.

2.2. Unlearning Pathway Searching

As shown in Figure 1b, one crucial decision is the selection of two end models on the pathway. Ideally, these two models should satisfy the following properties for unlearning: ① One end model should fully preserve model utility; ② The other end model provides essential unlearning information and trend. Then we can find an optimal pathway between two end models, ensuring a balance between model utility and unlearning effectiveness. Guided by these insights, two end models in our nonlinear pathway are as follows:

- **Original model $\boldsymbol{\theta}_o$**. The model $\boldsymbol{\theta}_o$ is trained on the training data $\mathcal{D}_{train}$ before unlearning.
- **Pre-unlearning model $\boldsymbol{\theta}_p$**. The model $\boldsymbol{\theta}_p$ is obtained by applying any existing MU method to remove the influence of forgetting data $\mathcal{D}_f$.

The goal of mode connectivity unlearning is to construct a smooth pathway from $\boldsymbol{\theta}_o$ to $\boldsymbol{\theta}_p$, ensuring an unlearning model $\boldsymbol{\theta}_u$ on the pathway can better forget $\mathcal{D}_f$ while preserving performance on $\mathcal{D}_r$.

Inspired by mode connectivity (Garipov et al., 2018), we leverage a quadratic Bézier curve to explore a nonlinear unlearning pathway between model $\boldsymbol{\theta}_o$ and model $\boldsymbol{\theta}_p$ in the parameter space. The Bézier curve is generally superior because it offers a smooth and flexible pathway. In our MU scenario, the quadratic Bézier curve $\phi_{\boldsymbol{\theta}}(t)$ between models $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_p$ in parameter space is defined as follows,

$$\phi_{\boldsymbol{\theta}_c}(t) = (1-t)^2\boldsymbol{\theta}_o + 2(1-t)t\boldsymbol{\theta}_c + t^2\boldsymbol{\theta}_p, \quad t \in [0,1]. \tag{1}$$

$\phi_{\boldsymbol{\theta}_c}(0) = \boldsymbol{\theta}_o$ and $\phi_{\boldsymbol{\theta}_c}(1) = \boldsymbol{\theta}_p$ are the original model and the pre-unlearning model respectively. For values of t between 0 and 1, it represents a spectrum of potential unlearning models $\boldsymbol{\theta}_u$ along the pathway. By optimizing this control model $\boldsymbol{\theta}_c$, we can shape the trajectory between $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_p$. It is therefore crucial to design an appropriate loss function that guides the optimization of $\boldsymbol{\theta}_c$. This loss must strike a balance between two goals, ensuring effective forgetting and preserving the model utility, which leads to our loss design,

$$\mathcal{L}_{mcu} = \mathbb{E}_{t\sim U(0,1)}[\mathcal{L}(\mathcal{D}_r; \phi_{\boldsymbol{\theta}_c}(t)) - \beta \cdot \mathcal{L}(\mathcal{D}_f; \phi_{\boldsymbol{\theta}_c}(t))], \tag{2}$$

where $\mathcal{L}(\mathcal{D}_r; \phi_{\boldsymbol{\theta}_c}(t))$ is the cross-entropy loss on retaining data $\mathcal{D}_r$, and β is an unlearning penalty coefficient controlling the trade-off between retaining predictive performance and forgetting quality. $U(0,1)$ is the uniform distribution on $[0,1]$, from which we sample a value t for each training batch. In each batch, the loss is computed at the specific point $\phi_{\boldsymbol{\theta}_c}(t)$ along the pathway, derive gradients with respect to $\boldsymbol{\theta}_c$, and update only $\boldsymbol{\theta}_c$ accordingly. Note that the pathway searching process only requires optimizing $\boldsymbol{\theta}_c$, while the entire pathway is a simple combination of $\boldsymbol{\theta}_o$, $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_p$ as defined in Eq. 1.

2.3. Parameter Mask

While the pathway searching process described above is already efficient, we aim to further improve the searching efficiency by selectively updating only the most important parameters. As illustrated in Figure 1a, our parameter mask strategy consists of two components: filtering based on $\mathcal{D}_r$ and reserving based on $\mathcal{D}_f$. The strategy effectively identifies parameters that are highly influential for $\mathcal{D}_f$ while being less critical for $\mathcal{D}_r$, ensuring a targeted update process.

Filtering based on $\mathcal{D}_r$. We first utilize the gradient of the retaining loss with respect to the original model $\boldsymbol{\theta}_o$ on the retaining dataset $\mathcal{D}_r$. A fraction k_r of the parameters is selected for exclusion, where these parameters exhibit an importance above a quantile-based threshold γ_{k_r}. The formulated equation is as follows,

$$\boldsymbol{m}_r^i = \mathbb{0}\left\{\frac{\|\nabla_{\boldsymbol{\theta}_o^i}\mathcal{L}(\mathcal{D}_r; \boldsymbol{\theta}_o)\|_2}{|\boldsymbol{\theta}_o^i|} > \gamma_{k_r}\right\}, \tag{3}$$

where $\boldsymbol{m}_r^i$ is the binary mask for the i-th parameter in whole mask $\boldsymbol{m}$, and $\|\cdot\|_2$ denotes the L_2-norm over each parameter. L_2-norm reflects the Euclidean length of gradient vectors, making it more sensitive to parameters with larger impacts. The denominator $|\boldsymbol{\theta}_o^i|$ represents the element number in the

i-th parameter of $\boldsymbol{\theta}_o$, i.e., $\boldsymbol{\theta}_o^i$, ensuring fair importance
lation across parameters with different sizes. The ele
wise indicator function $\mathbb{0}(\cdot > \gamma_{k_r})$ assigns a zero ve
$\boldsymbol{m}_r^i$ if the average importance of this parameter excee
threshold γ_{k_r}, and otherwise an all-ones vector.

Reserving based on $\mathcal{D}_f$. After removing parameters influential for $\mathcal{D}_r$, we further refine the mask by reserving parameters based on the gradient of the forgetting loss,

$$\boldsymbol{m}_f^i = \mathbb{1}\left\{ \frac{\|\nabla_{\boldsymbol{\theta}_o^i} \mathcal{L}(\mathcal{D}_f; \boldsymbol{\theta}_o)\|_2}{|\boldsymbol{\theta}_o^i|} > \gamma_k \right\}. \tag{4}$$

Similarly, the threshold γ_k is determined by selecting the top-k percentile of normalized gradient L_2 norms across parameters. The element-wise indicator function $\mathbb{1}(\cdot > \gamma_k)$ assigns an all-one vector to the entire i-th parameter $\boldsymbol{\theta}_o^i$ if its importance exceeds the threshold γ_k.

The final mask $\boldsymbol{m}$ is represented as,

$$\boldsymbol{m} = \boldsymbol{m}_r \,\&\, \boldsymbol{m}_f, \tag{5}$$

where $\&$ represents the logical operation AND. Thus, the optimization in the MCU can be formulated as follows,

$$\min_{\boldsymbol{\theta}_c \odot \boldsymbol{m}} \mathcal{L}_{mcu}, \tag{6}$$

where training efficiency is improved by reducing unnecessary gradient updates with the mask $\boldsymbol{m}$.

We evaluate the efficiency and effectiveness of our parameter mask on CIFAR-10 using PreResNet-100 with 10% random data forgetting. We set $k = k_r = 10\%$ to generate the mask in our framework with NegGrad+ (Kurmanji et al., 2024a) as $\boldsymbol{\theta}_p$. As shown in the left panel of Figure 2, the parameter mask significantly accelerates backward propagation, achieving a 75.23% reduction in average epoch runtime. In the right panel, we compare our parameter mask (solid line) to a 10% random mask (dashed line). The random mask causes noticeable performance degradation on retaining data $\mathcal{D}_r$ and testing data $\mathcal{D}_t$, with accuracy drops of 3.44% and 2.48% at $t = 0.5$, respectively. In contrast, the forget accuracy gap is only 1.32% at the same point. These results demonstrate that our parameter mask improves training efficiency while maintaining model utility.

2.4. Adaptive Unlearning Penalty Coefficient

We further propose an adaptive strategy for β, which can avoid trial-and-error cost of hyperparameter selection and potentially improve our performance. Based on two MU objectives, we establish the alignment principles behind adaptive β: ① **Preserve model utility on the retaining dataset**: align retaining accuracy ($Acc_u(\mathcal{D}_r)$) of the unlearning model with original model's training accuracy ($Acc_o(\mathcal{D}_{train})$). ② **Unlearn the forgetting data as if it were never trained**: align the unlearning pathway's forgetting accuracy ($Acc_u(\mathcal{D}_f)$) with original model's validation accuracy ($Acc_o(\mathcal{D}_v)$). Objective ② implies the performance of $\boldsymbol{\theta}_u$ along the unlearning pathway on $\mathcal{D}_f$ should match that of $\boldsymbol{\theta}_o$ on unseen data. The $Acc_o(\mathcal{D}_{train})$ and $Acc_o(\mathcal{D}_v)$ are constants and accessible, as they are recorded during the original training. Thus, the three conditions are listed as:

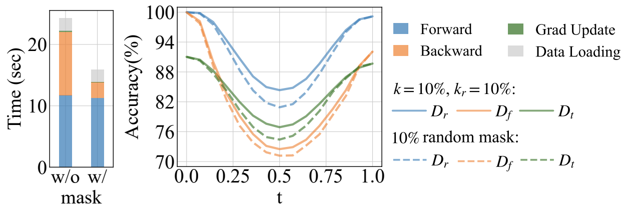


Figure 2: The efficiency and effectiveness of our parameter mask. 'w/o' and 'w/' represent the results without 10% mask and with 10% mask. The x-axis of the right panel represents the parameter t along the Bézier curve, while the y-axis corresponds to accuracy.

- **Condition ❶.** When $Acc_u(\mathcal{D}_f) \leq Acc_o(\mathcal{D}_v)$, it indicates that the model has forgotten $\mathcal{D}_f$. In this case, $\beta = 0$ to prevent further forgetting.
- **Condition ❷.** If $Acc_u(\mathcal{D}_f) > Acc_o(\mathcal{D}_v)$ and the performance degradation on the $\mathcal{D}_r$ is more severe than that on $\mathcal{D}_f$, a mild forgetting can be set to $\beta = 0.1$.
- **Condition ❸.** Otherwise, we apply a stronger forgetting adjustment with $\beta = 0.5$.

The adaptive adjustment of β is formulated as follows,

$$\beta = \begin{cases} 0, & Acc_u(\mathcal{D}_f) \leq Acc_o(\mathcal{D}_v), \ (❶) \\ 0.1, & Acc_\phi(\mathcal{D}_f) > Acc_o(\mathcal{D}_v) \ and \ (❷) \\ & \frac{Acc_\phi(\mathcal{D}_f) - Acc_o(\mathcal{D}_v)}{Acc_o(\mathcal{D}_v)} < \frac{Acc_\phi(\mathcal{D}_r) - Acc_o(\mathcal{D}_{train})}{Acc_o(\mathcal{D}_{train})}, \\ 0.5, & otherwise. \ (❸) \end{cases}$$

Unlike fixed β tuning, adaptive β can update dynamically at every batch to balance forgetting and retaining.

2.5. Optimal Model and Effective Region

As illustrated in Figure 1c, after searching the pathway, a key step is to identify the *optimal unlearning model* and the *effective unlearning region* along it. Based on the alignment principles in Section 2.4, we $Acc_o(\mathcal{D}_{train})$ and $Acc_o(\mathcal{D}_v)$ as references to compute alignment gaps. These gaps quantify the deviation of each point along the path from desired behavior, guiding both model selection and region identification. The model selection process along the Bézier curve is conducted during inference, and thus incurs negligible computational overhead. For optimal model selection, we use an efficient heuristic: we first evaluate the models at $t = 0.75$ and $t = 1$, and then apply cubic interpolation to estimate the t value that minimizes the alignment gap.

Table 1: Unlearning performance of six different pre-unlearnin models in MCU$_\beta$. The results are presented in the format a : b, with a as the mean and b as the standard deviation from independent trials. The performance gap relative to RT metho is represented in (•). The Avg. Gap is derived by averaging gap across UA, RA, TA and MIA. Smaller gaps reflect closer alignmer with the RT model's performance. Note RTE is reported in minute and UA equals 1− accuracy of $\mathcal{D}_f$. The results demonstrate that applying our MCU$_\beta$ framework to any unlearning method can significantly enhance unlearning performance.

Methods	UA	RA	TA	MIA	Avg. Gap	RTE
RT	$10.54_{\pm 0.34}$(0.00)	$99.98_{\pm 0.01}$(0.00)	$89.59_{\pm 0.22}$(0.00)	$18.41_{\pm 0.52}$(0.00)	0.00	105.70
FT	$0.42_{\pm 0.12}$(10.12)	$99.93_{\pm 0.01}$(0.05)	$90.99_{\pm 0.13}$(1.40)	$3.71_{\pm 0.25}$(14.70)	6.57	5.31
MCU$_\beta$-FT	$5.62_{\pm 0.07}$(4.92)	$99.12_{\pm 0.02}$(0.86)	$89.59_{\pm 0.10}$(0.00)	$10.68_{\pm 0.62}$(7.73)	**3.37**	9.76
RL	$4.14_{\pm 0.20}$(6.40)	$99.69_{\pm 0.02}$(0.29)	$90.16_{\pm 0.09}$(0.57)	$21.93_{\pm 0.66}$(3.52)	2.70	6.21
MCU$_\beta$-RL	$10.54_{\pm 0.02}$(0.00)	$98.60_{\pm 0.11}$(1.38)	$89.21_{\pm 0.08}$(0.38)	$22.48_{\pm 0.47}$(4.07)	**1.46**	12.24
GA	$0.06_{\pm 0.00}$(10.48)	$99.97_{\pm 0.00}$(0.01)	$90.89_{\pm 0.01}$(1.30)	$0.98_{\pm 0.11}$(17.43)	7.31	0.38
MCU$_\beta$-GA	$3.84_{\pm 0.01}$(6.70)	$98.80_{\pm 0.05}$(1.18)	$88.86_{\pm 0.33}$(0.73)	$13.22_{\pm 0.37}$(5.19)	**3.45**	5.03
NegGrad+	$7.03_{\pm 0.32}$(3.51)	$98.63_{\pm 0.19}$(1.35)	$89.26_{\pm 0.23}$(0.33)	$11.71_{\pm 0.38}$(6.70)	2.97	2.96
MCU$_\beta$-NegGrad+	$10.29_{\pm 0.24}$(0.25)	$98.69_{\pm 0.04}$(1.29)	$89.11_{\pm 0.13}$(0.48)	$16.45_{\pm 0.89}$(1.96)	**1.00**	6.82
Salun	$6.67_{\pm 0.26}$(3.87)	$97.87_{\pm 0.14}$(2.11)	$90.54_{\pm 0.19}$(0.95)	$35.45_{\pm 0.57}$(17.04)	5.99	6.38
MCU$_\beta$-Salun	$10.49_{\pm 0.07}$(0.05)	$97.55_{\pm 0.10}$(2.43)	$89.21_{\pm 0.23}$(0.38)	$30.30_{\pm 1.17}$(11.89)	**3.68**	11.35
NegTV	$2.36_{\pm 1.12}$(8.18)	$99.08_{\pm 0.60}$(0.90)	$88.53_{\pm 0.88}$(1.06)	$4.14_{\pm 0.29}$(14.27)	6.10	0.70
MCU$_\beta$-NegTV	$8.11_{\pm 0.60}$(2.43)	$98.01_{\pm 0.32}$(1.97)	$87.74_{\pm 0.33}$(1.85)	$11.48_{\pm 0.18}$(6.93)	**3.30**	5.67

This is motivated by our empirical observation that optimal models consistently lie in the interval $t \in [0.75, 1]$, allowing us to avoid exhaustive evaluation over the entire path. For effective region identification, we uniformly sample 20 points along $t \in [0, 1]$ and fit a cubic interpolation curve to the gap values. Any point on this curve with a smaller alignment gap than the pre-unlearning model (i.e., at $t = 1$) is considered part of the effective unlearning region.

3. Experiments

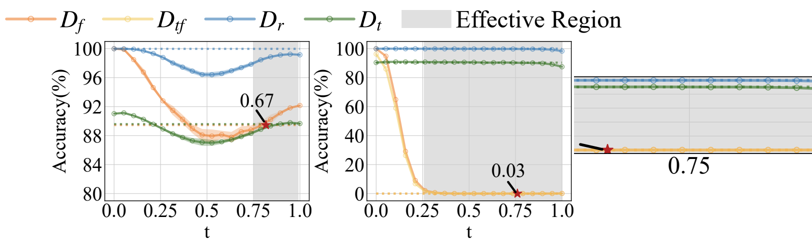
3.1. Experiment Setups

We focus on the image classification task for both random data forgetting and class-wise forgetting. Three datasets and architectures are evaluated: **CIFAR-10** on **PreResNet-110**, **ImageNet-100** on **ViT**, and **Tiny-ImageNet** on **VGG-16-BN**. We compare our method against seven MU methods: Retrain (**RT**), Finetune (**FT**) (Warnecke et al., 2021), Random Label (**RL**) (Graves et al., 2021), Gradient Ascent (**GA**) (Thudi et al., 2022), **NegGrad+** (Kurmanji et al., 2024a), **SalUn** (Fan et al., 2023), **NegTV** (Garipov et al., 2018). We denote our framework with fixed β as **MCU**[1] and with adaptive β as **MCU$_\beta$**. Unless otherwise stated, the pre-unlearning model in our framework is NegGrad+. We evaluate all methods across **UA** (Unlearning Accuracy, 1− accuracy of forgetting data $\mathcal{D}_f$), **RA** (Retaining Accuracy), **TA** (Test Accuracy), **MIA** (Membership Inference Attack), and **RTE** (Running Time Efficiency).

3.2. Experiment Results

We show the experiments on CIFAR-10 with PreResNet-110 under 10% random data forgetting. See additional results in Tables 2-3 with other datasets, architectures and forgetting scenarios in Appedndix D.

[1]The best results achieved through hyperparameter β search.



(a) 10% Random Data (b) Class-wise

Figure 3: Effective unlearning region on MCU$_\beta$. The marker ★ highlights the position with the minimum average gap from RT, with the accompanying numerical value indicating the exact average accuracy gap of $\mathcal{D}_f$, $\mathcal{D}_r$ and $\mathcal{D}_t$ (and $\mathcal{D}_{tf}$ for class-wise forgetting). The dotted line represents the RT method's accuracy, serving as a reference. The shaded gray area denotes the effective unlearning region, where models achieve better unlearning performance than the pre-unlearning model.

Overall Performance. In this experiment, we use six MU methods as pre-unlearning models in our MCU$_\beta$ framework. Table 1 compares the performance of these methods before and after incorporating the MCU$_\beta$. The results demonstrate that MCU$_\beta$ significantly enhances the performance of all MU methods. On average, the Avg. Gap is reduced by 49.71%, with particularly notable improvements in the UA metric, i.e., forgetting quality. We also present the MCU and MCU$_\beta$ results in Tables 2-3 in Appendix D, which shows the effectiveness of our adaptive β.

Effective Unlearning Region. Figure 3 shows visualization results of MCU$_\beta$ on CIFAR-10 under both 10% random data forgetting and class-wise scenarios. It validates that MCU$_\beta$ not only identifies a single effective unlearning model but also discovers a substantial region along the Bézier pathway where multiple models in this pathway exhibit effective unlearning. Within this effective unlearning region, models achieve superior unlearning performance compared to the pre-unlearning model. MCU$_\beta$ provides greater flexibility since different effective unlearning models can be selected based on task-specific requirements. For example, in Figure 3a, models to the right of marker ★ preserve better predictive performance, while those to the left demonstrate stronger forgetting efficacy.

4. Conclusion

In this work, we propose a novel framework MCU, leveraging mode connectivity to search nonlinear pathway in parameter space for unlearning. Unlike traditional methods that identify only a single unlearning model, MCU uncovers a spectrum of unlearning models along the pathway and is free from empirical hyperparameter tuning. As a plug-and-play framework, MCU seamlessly integrates with existing MU methods and improves their unlearning efficacy.

References

Cha, S., Cho, S., Hwang, D., Lee, H., Moon, T., and Lee, M. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In Proceedings of the AAAI conference on artificial intelligence, volume 38, pp. 11186–11194, 2024.

Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. When machine unlearning jeopardizes privacy. In Proceedings of the 2021 ACM SIGSAC conference on computer and communications security, pp. 896–911, 2021.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pp. 7210–7217, 2023a.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. Zero-shot machine unlearning. IEEE Transactions on Information Forensics and Security, 18: 2345–2354, 2023b.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In International conference on machine learning, pp. 1309–1318. PMLR, 2018.

Fan, C., Liu, J., Zhang, Y., Wong, E., Wei, D., and Liu, S. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. arXiv preprint arXiv:2310.12508, 2023.

Foster, J., Schoepf, S., and Brintrup, A. Fast machine unlearning without retraining through selective synaptic dampening. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 12043–12051, 2024.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems, 31, 2018.

Goel, S., Prabhu, A., Sanyal, A., Lim, S.-N., Torr, P., and Kumaraguru, P. Towards adversarial evaluations for inexact machine unlearning. arXiv preprint arXiv:2201.06640, 2022.

Golatkar, A., Achille, A., and Soatto, S. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9304–9312, 2020.

Graves, L., Nagisetty, V., and Ganesh, V. Amnesiac machine learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 11516–11524, 2021.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. Certified data removal from machine learning models. arXiv preprint arXiv:1911.03030, 2019.

Huang, Y. and Canonne, C. L. Tight bounds for machine unlearning via differential privacy. arXiv preprint arXiv:2309.00886, 2023.

Huang, Z., Cheng, X., Zheng, J., Wang, H., He, Z., Li, T., and Huang, X. Unified gradient-based machine unlearning with remain geometry enhancement. Advances in Neural Information Processing Systems, 37:26377–26414, 2025.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. arXiv preprint arXiv:2212.04089, 2022.

Iurada, L., Ciccone, M., Tommasi, T., et al. Efficient model editing with task-localized sparse fine-tuning. In LEARNING REPRESENTATIONS. INTERNATIONAL CONFERENCE. 13TH 2025.(ICLR 2025). ICLR, 2025.

Kurmanji, M., Triantafillou, E., and Triantafillou, P. Machine unlearning in learned databases: An experimental analysis. Proc. ACM Manag. Data, 2(1), March 2024a. doi: 10.1145/3639304. URL https://doi.org/10.1145/3639304.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning. Advances in neural information processing systems, 36, 2024b.

Liu, Y., Sun, C., Wu, Y., and Zhou, A. Unlearning with fisher masking. arXiv preprint arXiv:2310.05331, 2023.

Micaelli, P. and Storkey, A. J. Zero-shot knowledge transfer via adversarial belief matching. Advances in Neural Information Processing Systems, 32, 2019.

Ortiz-Jimenez, G., Favero, A., and Frossard, P. Task arithmetic in the tangent space: Improved editing of pre-trained models. Advances in Neural Information Processing Systems, 36, 2024.

Ren, J., Chen, P.-Y., and Wang, R. Revisiting mode connectivity in neural networks with bezier surface. In The Thirteenth International Conference on Learning Representations.

Ren, J., Chen, P.-Y., and Wang, R. Revisiting mode connectivity in neural networks with bezier surface. In The Thirteenth International Conference on Learning Representations, 2025.

Shi, Y. and Wang, R. Redefining machine unlearning: A conformal prediction-motivated approach. arXiv preprint arXiv:2501.19403, 2025.

Tarun, A. K., Chundawat, V. S., Mandal, M., and Kankanhalli, M. Fast yet effective machine unlearning. IEEE Transactions on Neural Networks and Learning Systems, 2023.

Thudi, A., Deza, G., Chandrasekaran, V., and Papernot, N. Unrolling sgd: Understanding factors influencing machine unlearning. In 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), pp. 303–319. IEEE, 2022.

Wang, R., Li, Y., and Liu, S. Exploring diversified adversarial robustness in neural networks via robust mode connectivity. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2346–2352, 2023.

Warnecke, A., Pirch, L., Wressnegger, C., and Rieck, K. Machine unlearning of features and labels. arXiv preprint arXiv:2108.11577, 2021.

Wei, S., Zhang, M., Zha, H., and Wu, B. Shared adversarial unlearning: Backdoor mitigation by unlearning shared adversarial examples. Advances in Neural Information Processing Systems, 36:25876–25909, 2023.

Appendix

A. Related Work

A.1. Machine Unlearning

Complete retraining for MU involves retraining from scratch after removing forgetting data, but its high cost has led to the development of efficient approximate unlearning techniques. Some works (Fan et al., 2023; Graves et al., 2021; Kurmanji et al., 2024a; Shi & Wang, 2025; Tarun et al., 2023; Thudi et al., 2022) focus on designing loss functions to achieve forgetting. Knowledge distillation-based methods (Chundawat et al., 2023a;b; Goel et al., 2022; Kurmanji et al., 2024b; Micaelli & Storkey, 2019) have emerged as promising approaches, where a student model is trained to mimic the behavior of the original model on the retaining dataset while excluding the knowledge of forgetting data. Several works (Foster et al., 2024; Golatkar et al., 2020; Liu et al., 2023) leverage the Fisher Information Matrix to identify and modify the most influential parameters associated with the forgetting data, enabling more targeted and efficient unlearning. Additionally, adversarial attacks (Cha et al., 2024; Chen et al., 2021; Wei et al., 2023) and differential privacy (Guo et al., 2019; Huang & Canonne, 2023) have also been explored as promising techniques for MU.

One pivotal advance came from task arithmetic (Ilharco et al., 2022), which enabled efficient data removal by applying negation operations. Building on this, a neural tangent kernel-based linear negation method was introduced to improve task arithmetic by constraining model updates to the tangent space (Ortiz-Jimenez et al., 2024). However, the entanglement issue still exists as they cannot guarantee that the task vector's influence localizes solely on forgetting data (see Appendix B for details). Overall, this oversimplified assumption of linear parameter updating fails to account for the complex and nonlinear characteristics of neural networks' loss landscapes and suffers from a weight entanglement issue.

In machine unlearning, existing parameter mask approaches, such as SalUn (Fan et al., 2023), primarily focus on identifying parameters that have a significant impact on the forgetting data. However, these parameters may also be crucial for the predictive performance on retaining data, which leads to unintended degradation in model utility. Subsequently, works (Foster et al., 2024; Huang et al., 2025) proposed weight saliency maps that jointly consider forgetting and retaining data based on the Fisher Information Matrix, which are computationally expensive. Moreover, all these parameter masking strategies operate at the element level within individual parameters. In the element level parameter mask, gradient computations are still required for all parameters during training, which limits practical efficiency gains.

A.2. Mode Connectivity

Mode connectivity refers to the existence of low-loss pathways between different local minima in a neural network's loss landscape. It has been observed that neural networks trained on the same dataset but initialized differently can be connected by a smooth, low-loss curve in parameter space (Garipov et al., 2018). This phenomenon has been further explored, demonstrating that such connectivity generalizes across architectures and datasets, forming high-dimensional manifolds of functionally equivalent models (Draxler et al., 2018). Recent work has extended the mode connectivity concept from Bézier curve to surface, enabling the connection of multiple networks (Ren et al.). Given its ability to identify meaningful pathways in parameter space, mode connectivity provides an efficient and effective approach for unlearning.

B. Weight Entanglement in Linear MU Method

In this section, we analyze the weight entanglement issue that arises in linear MU methods, i.e., task arithmetic (Ilharco et al., 2022; Iurada et al., 2025; Ortiz-Jimenez et al., 2024). Let $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ be a neural network that takes input $\boldsymbol{x} \in \mathcal{X}$ and is parameterized by $\boldsymbol{\theta} \in \Theta$. We assume $\mathcal{X} \subseteq \mathbb{R}^d$, $\Theta \subseteq \mathbb{R}^m$, and $\mathcal{Y} \subseteq \mathbb{R}^c$. Given the original model parameters $\boldsymbol{\theta}_o \in \mathbb{R}^m$, a fine-tuned model with parameters $\boldsymbol{\theta}_{ft}^f$ is trained on the forgetting dataset $\mathcal{D}_f$.

The unlearning task vector is defined as the difference between the fine-tuned and original model parameters, i.e., $\boldsymbol{\tau}_f = \boldsymbol{\theta}_{ft}^f - \boldsymbol{\theta}_o$ where $\boldsymbol{\theta}_{ft}^f$ is fine-tuned on forgetting data $\mathcal{D}_f$ based on $\boldsymbol{\theta}_o$. By task arithmetic, it is easy to manipulate the output behavior of the model by adding or subtracting task vectors. Thus, in our unlearning scenario, the unlearning model can be defined with the negation task vector as:

$$f(\boldsymbol{x}; \boldsymbol{\theta}_u) = f(\boldsymbol{x}; \boldsymbol{\theta}_o - \alpha\boldsymbol{\tau}_f) = f\left(\boldsymbol{x}; \boldsymbol{\theta}_o - \alpha(\boldsymbol{\theta}_{ft}^f - \boldsymbol{\theta}_o)\right), \tag{7}$$

where α is a coefficient that controls forgetting level. This formulation implicitly requires that subtracting the task vector

$\boldsymbol{\tau}_f$ does not affect the model's predictions on inputs outside the forgetting data $\mathcal{D}_f$. In other words, $\boldsymbol{\tau}_f$ should not encode any information about data outside $\mathcal{D}_f$, i.e., retaining data $\mathcal{D}_r$. Therefore, the condition for this equation to hold can be formalized as:

$$f\left(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f\right)=\begin{cases}f(\boldsymbol{x};\boldsymbol{\theta}_o), & \boldsymbol{x}\in\mathcal{D}_r\\ f(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f), & \boldsymbol{x}\in\mathcal{D}_f.\end{cases} \tag{8}$$

This condition requires that the task vector $\boldsymbol{\tau}_f$ in Eq. 7 only influences the model on the forgetting dataset, leaving the performance on retaining data $\mathcal{D}_r$ unaffected. However, task vectors obtained via simple fine-tuning on $\mathcal{D}_f$ do not guarantee this condition, which faces a weight entanglement issue.

To address this, the model must exhibit a form of weight disentanglement. Ideally, the model should behave as a composition of spatially localized components, each responsible for a specific data domain. For our unlearning case, this means the function should decompose as:

$$\begin{aligned}&f(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f)\\&=f(\boldsymbol{x};\boldsymbol{\theta}_o)\mathbb{1}(\boldsymbol{x}\in\mathcal{D}_r)+f(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f)\mathbb{1}(\boldsymbol{x}\in\mathcal{D}_f)\\&=g_o(\boldsymbol{x})+g_f(\boldsymbol{x};-\alpha\boldsymbol{\tau}_f),\end{aligned} \tag{9}$$

The term $g_o(\boldsymbol{x}):=f(\boldsymbol{x};\boldsymbol{\theta}_o)\cdot\mathbb{1}(\boldsymbol{x}\in\mathcal{D}_r)$ denotes spatially localized components for retaining data domain, and $g_o(\boldsymbol{x})=0$ for $\boldsymbol{x}\in\mathcal{D}_f$. The term $g_f(\boldsymbol{x};-\alpha\boldsymbol{\tau}_f):=f(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f)\cdot\mathbb{1}(\boldsymbol{x}\in\mathcal{D}_f)$ captures the influence of the unlearning task vector, localized within the forgetting data domain, and $g_f(\boldsymbol{x};\alpha\boldsymbol{\tau}_f)=0$ for $\boldsymbol{x}\in\mathcal{D}_r$. This decomposition encapsulates the principle that only data within $\mathcal{D}_f$ should be influenced by $\boldsymbol{\tau}_f$.

To make this decomposition tractable, linearizing the network around $\boldsymbol{\theta}_o$ via a first-order Taylor expansion is attempted to realize it by :

$$\begin{aligned}&f(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f)\\&\approx f_{\text{lin}}(\boldsymbol{x};\boldsymbol{\theta}_o-\alpha\boldsymbol{\tau}_f)=f(\boldsymbol{x};\boldsymbol{\theta}_o)-\alpha\boldsymbol{\tau}_f^\top\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}_o).\end{aligned} \tag{10}$$

This linearized model expresses the output as a combination of the original prediction and a perturbation determined by the gradient of f at $\boldsymbol{\theta}_o$.

While this form resembles the disentangled decomposition in Eq. 9, this resemblance is superficial. The disentanglement condition requires that the influence of $\boldsymbol{\tau}_f$ vanishes for all inputs not in $\mathcal{D}_f$. However, the term $\boldsymbol{\tau}_f^\top\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}_o)$ is generally non-zero for arbitrary $\boldsymbol{x}\in\mathcal{D}_r$, since neither $\boldsymbol{\tau}_f$ nor the gradient are guaranteed to be localized. That is, the linearized update will affect predictions on $\mathcal{D}_r$, unless $\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}_o)$ itself vanishes for $\boldsymbol{x}\in\mathcal{D}_r$, or unless $\boldsymbol{\tau}_f$ lies in the nullspace of these gradients.

Therefore, we conclude that both the standard task vector approach (Ilharco et al., 2022) and the linearized task vector method (Ortiz-Jimenez et al., 2024) fail to ensure weight disentanglement for ideal unlearning.

C. Implementation Details

CIFAR-10 on PreResNet-100. We train the original model and RT model for 200 epochs using the SGD optimizer with a cosine-scheduled learning rate initialized at 0.01. For the FT, RL, and SalUn methods, they are performed for 10 epochs with a learning rate of 0.01. The GA and NegGrad+ methods are trained for 5 epochs with a learning rate of 0.01. In the case of NegTV, the model undergoes a finetune model on $\mathcal{D}_f$ for 10 epochs, and the scaling coefficient α is set to 0.9 for random data forgetting and 0.2 for class-wise forgetting. For both MCU$_\beta$ and MCU, random data forgetting is performed for 10 epochs, whereas class-wise forgetting is conducted for 5 epochs, both with a learning rate of 0.01.

ImageNet-100 on ViT. We utilize a pretrained ViT and fine-tune 30 epochs with a learning rate of 0.001 to get the original model. The RT method follows the same setting as the original model. For FT, RL, and SalUn, training is performed for 5 epochs, while GA and NegGrad+ are trained for 2 epochs. Similarly, the finetuning model for the NegTV method is trained 5 epochs with a learning rate of 0.001 and a coefficient α of 0.9. For MCUs, they are trained for 2 epochs.

Tiny-ImageNet on VGG-16-BN. We train both the original model and the RT model for 100 epochs with a learning rate of 0.1. The FT, RL, and SalUn methods undergo training for 10 epochs with a learning rate of 0.01, while the GA and

Table 2: Overall performance of MU methods for **10% random data forgetting** in three datasets. The results are presented in the format $a \pm b$, with a as the mean and b as the standard deviation from 5 independent trials. The performance gap relative to RT method is represented in (•). The Avg. Gap is derived by averaging gaps across accuracy metrics, UA, RA, TA and MIA. Smaller gaps reflect closer alignment with the RT model's performance. Note RTE is reported in minutes and UA equals $1-$ accuracy of $\mathcal{D}_f$.

Methods	UA	RA	TA	MIA	Avg. Gap	RTE
CIFAR-10 with PreResNet-110						
RT	$10.54_{\pm 0.34}$(0.00)	$99.98_{\pm 0.01}$(0.00)	$89.59_{\pm 0.22}$(0.00)	$18.41_{\pm 0.52}$(0.00)	0.00	105.70
FT	$0.42_{\pm 0.12}$(10.12)	$99.93_{\pm 0.01}$(0.05)	$90.99_{\pm 0.13}$(1.40)	$3.71_{\pm 0.25}$(14.70)	6.57	5.31
RL	$4.14_{\pm 0.20}$(6.40)	$99.69_{\pm 0.02}$(0.29)	$90.16_{\pm 0.09}$(0.57)	$21.93_{\pm 0.66}$(3.52)	2.70	6.21
GA	$0.06_{\pm 0.00}$(10.48)	$99.97_{\pm 0.00}$(0.01)	$90.89_{\pm 0.01}$(1.30)	$0.98_{\pm 0.11}$(17.43)	7.31	0.38
NegGrad+	$7.03_{\pm 0.32}$(3.51)	$98.63_{\pm 0.19}$(1.35)	$89.26_{\pm 0.23}$(0.33)	$11.71_{\pm 0.38}$(6.70)	2.97	2.96
SalUn	$6.67_{\pm 0.26}$(3.87)	$97.87_{\pm 0.14}$(2.11)	$90.54_{\pm 0.19}$(0.95)	$35.45_{\pm 0.57}$(17.04)	5.99	6.38
NegTV	$2.36_{\pm 1.12}$(8.18)	$99.08_{\pm 0.60}$(0.90)	$88.53_{\pm 0.88}$(1.06)	$4.14_{\pm 0.29}$(14.27)	6.10	0.70
MCU	$9.52_{\pm 0.04}$(1.02)	$98.97_{\pm 0.01}$(1.01)	$89.00_{\pm 0.03}$(0.59)	$16.33_{\pm 0.93}$(2.08)	1.18	6.80
MCU$_\beta$	$10.29_{\pm 0.24}$(0.25)	$98.69_{\pm 0.04}$(1.29)	$89.11_{\pm 0.13}$(0.48)	$16.45_{\pm 0.89}$(1.96)	**1.00**	6.82
ImageNet-100 with ViT						
RT	$11.63_{\pm 0.23}$(0.00)	$91.93_{\pm 0.01}$(0.00)	$87.83_{\pm 0.01}$(0.00)	$13.77_{\pm 0.42}$(0.00)	0.00	525.72
FT	$8.62_{\pm 0.01}$(3.01)	$92.21_{\pm 0.07}$(0.28)	$87.74_{\pm 0.18}$(0.09)	$10.88_{\pm 0.43}$(2.89)	1.57	82.23
RL	$9.53_{\pm 0.15}$(2.10)	$92.06_{\pm 0.02}$(0.13)	$87.82_{\pm 0.10}$(0.01)	$24.32_{\pm 0.35}$(10.55)	3.20	205.73
GA	$8.96_{\pm 0.89}$(2.67)	$91.15_{\pm 0.58}$(0.78)	$87.53_{\pm 0.37}$(0.30)	$10.50_{\pm 0.07}$(3.27)	1.76	6.71
NegGrad+	$13.15_{\pm 0.10}$(1.52)	$91.71_{\pm 0.03}$(0.22)	$87.37_{\pm 0.07}$(0.46)	$16.21_{\pm 0.30}$(2.44)	1.16	63.93
SalUn	$9.38_{\pm 0.13}$(2.25)	$91.94_{\pm 0.03}$(0.01)	$87.73_{\pm 0.13}$(0.10)	$24.29_{\pm 1.00}$(10.52)	3.22	170.34
NegTV	$10.17_{\pm 0.10}$(1.46)	$91.33_{\pm 0.09}$(0.60)	$87.24_{\pm 0.04}$(0.59)	$12.25_{\pm 0.21}$(1.52)	1.04	11.02
MCU	$11.44_{\pm 0.04}$(0.19)	$92.02_{\pm 0.02}$(0.09)	$87.62_{\pm 0.08}$(0.21)	$16.33_{\pm 0.18}$(2.56)	0.76	103.47
MCU$_\beta$	$11.63_{\pm 0.08}$(0.00)	$91.92_{\pm 0.10}$(0.01)	$87.70_{\pm 0.11}$(0.13)	$16.21_{\pm 0.31}$(2.44)	**0.65**	103.52
Tiny-ImageNet with VGG-16-BN						
RT	$45.45_{\pm 0.02}$(0.00)	$99.52_{\pm 0.02}$(0.00)	$55.59_{\pm 0.17}$(0.00)	$55.79_{\pm 0.17}$(0.00)	0.00	37.46
FT	$5.76_{\pm 0.07}$(39.69)	$99.34_{\pm 0.02}$(0.18)	$56.25_{\pm 0.10}$(0.66)	$15.95_{\pm 0.41}$(39.84)	20.09	3.80
RL	$38.59_{\pm 0.25}$(6.86)	$99.03_{\pm 0.02}$(0.49)	$53.87_{\pm 0.32}$(1.72)	$86.53_{\pm 0.29}$(30.74)	9.95	13.33
GA	$5.17_{\pm 0.07}$(40.28)	$96.11_{\pm 0.04}$(3.41)	$53.66_{\pm 0.02}$(1.93)	$7.89_{\pm 0.30}$(47.90)	23.38	0.32
NegGrad+	$51.06_{\pm 12.91}$(5.61)	$83.22_{\pm 5.81}$(16.30)	$46.74_{\pm 3.00}$(8.85)	$51.97_{\pm 1.30}$(3.82)	8.65	6.58
SalUn	$36.61_{\pm 0.23}$(8.84)	$99.03_{\pm 0.03}$(0.49)	$54.04_{\pm 0.35}$(1.55)	$85.37_{\pm 0.41}$(29.58)	10.12	13.79
NegTV	$0.81_{\pm 0.01}$(44.64)	$99.35_{\pm 0.02}$(0.17)	$56.85_{\pm 0.03}$(1.26)	$4.49_{\pm 0.20}$(51.30)	24.34	0.58
MCU	$42.42_{\pm 1.23}$(3.03)	$93.32_{\pm 0.33}$(6.20)	$52.53_{\pm 0.15}$(3.06)	$44.43_{\pm 1.41}$(11.36)	5.91	10.77
MCU$_\beta$	$49.92_{\pm 0.72}$(4.47)	$92.88_{\pm 0.19}$(6.64)	$52.92_{\pm 0.21}$(2.67)	$46.90_{\pm 0.07}$(8.89)	**5.67**	10.83

NegGrad+ methods are trained for 5 epochs. The NegTV method finetunes the model on forgetting data $\mathcal{D}_f$ for 10 epochs with a learning rate of 0.01. We observe that increasing the coefficient α of NegTV causes a substantial degradation in both RA and TA. To preserve model performance, we set α to 0.1. For MCUs, training is conducted over 5 epochs with a learning rate of 0.01.

Additional Details. All our experiments are conducted on a single Tesla V100 GPU. In our setup, we split the original test set into 10% for $\mathcal{D}_v$ and 90% for $\mathcal{D}_t$. We only use 50% of the retaining data during our MCU training process. The hyperparameters k and k_r are set to 0.5 and 0.1, respectively. For searching the optimal model on the curve, we obtain single models at $t = 0.75$ and 1 first. Then we interpolate to find the optimal model according to the approach in section 2. For searching an effective region, we obtained 20 single models along the pathway.

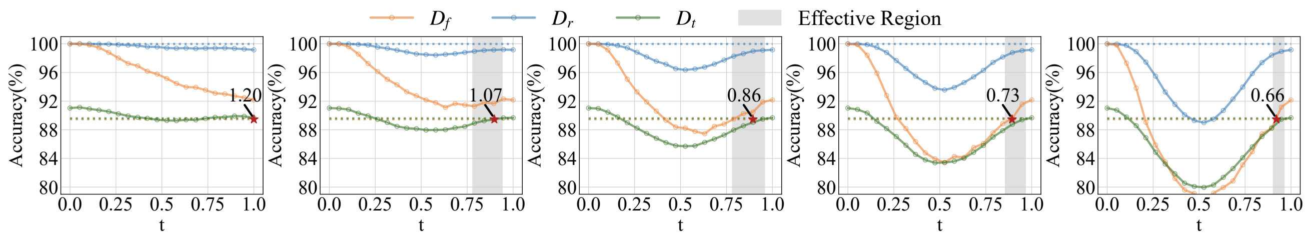
D. Additional Experimental Results

Additional Performance. We evaluate the performance of seven MU baselines and our framework MCU and MCU$_\beta$. Tables 2 and 3 present the results under 10% random data forgetting and class-wise forgetting scenario, respectively. These findings consistently align with our previous analysis, further substantiating the effectiveness of our MCU framework.

Under comprehensive metrics, both MCU and MCU$_\beta$ consistently exhibit the top two overall performances under both random data forgetting and class-wise forgetting. Notably, in the class-wise forgetting scenario, MCU$_\beta$ performs nearly on par with the RT method. Additionally, MCU$_\beta$ achieves superior overall performance compared to MCU, validating the effectiveness of our proposed adaptive β strategy. Unlike the fixed β that requires extensive tuning, the adaptive β approach dynamically adjusts during the training process, ensuring an excellent balance between forgetting and retaining performance.

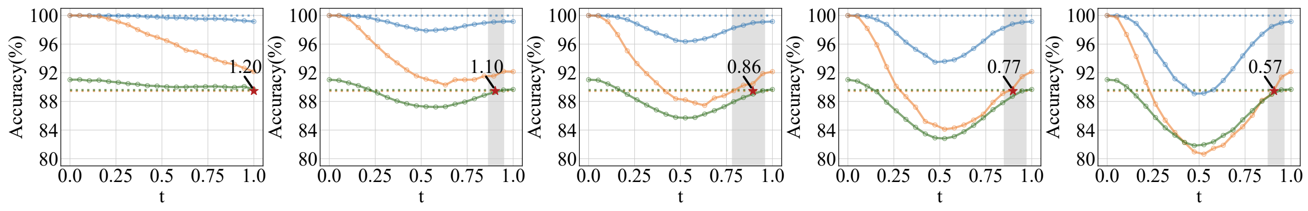
Table 3: Unlearning performance of MU methods for **class-wise forgetting** on **CIFAR-10** with **PreResNet-110**. The table adopts the same format as Table 2.

Methods	UA	UA$_{test}$	RA	TA	MIA	Avg. Gap	RTE
			CIFAR-10 with PreResNet-110				
RT	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.98$_{\pm0.00}$(0.00)	90.37$_{\pm0.08}$(0.00)	100.00$_{\pm0.00}$(0.00)	0.00	104.93
FT	18.53$_{\pm1.65}$(81.47)	24.63$_{\pm2.75}$(75.37)	99.94$_{\pm0.02}$(0.04)	91.01$_{\pm0.10}$(0.64)	43.18$_{\pm3.26}$(56.82)	42.87	5.52
RL	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	96.67$_{\pm0.45}$(3.31)	87.81$_{\pm0.61}$(2.56)	100.00$_{\pm0.00}$(0.00)	1.17	6.80
GA	85.01$_{\pm0.19}$(14.99)	88.50$_{\pm0.08}$(11.50)	90.55$_{\pm0.40}$(9.43)	80.90$_{\pm0.40}$(9.47)	86.27$_{\pm0.07}$(13.73)	11.82	0.40
NegGrad+	99.94$_{\pm0.05}$(0.06)	100.00$_{\pm0.00}$(0.00)	98.07$_{\pm0.25}$(1.91)	87.25$_{\pm0.28}$(3.12)	99.97$_{\pm0.04}$(0.03)	1.02	2.95
SalUn	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.81$_{\pm0.01}$(0.17)	90.34$_{\pm0.30}$(0.03)	100.00$_{\pm0.00}$(0.00)	0.04	6.97
NegTV	25.28$_{\pm4.92}$(74.72)	31.95$_{\pm5.35}$(68.05)	93.03$_{\pm0.04}$(6.95)	82.43$_{\pm0.22}$(7.94)	29.05$_{\pm3.75}$(70.95)	45.72	0.71
MCU	99.96$_{\pm0.01}$(0.04)	100.00$_{\pm0.00}$(0.00)	99.80$_{\pm0.01}$(0.18)	90.37$_{\pm0.03}$(0.00)	100.00$_{\pm0.00}$(0.00)	0.04	7.27
MCU$_\beta$	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.85$_{\pm0.00}$(0.13)	90.37$_{\pm0.03}$(0.00)	100.00$_{\pm0.00}$(0.00)	**0.03**	7.29
			ImageNet-100 with ViT				
RT	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	92.01$_{\pm0.08}$(0.00)	88.17$_{\pm0.11}$(0.00)	100.00$_{\pm0.00}$(0.00)	0.00	606.93
FT	80.69$_{\pm2.62}$(19.31)	83.00$_{\pm1.00}$(17.00)	92.33$_{\pm0.04}$(0.32)	87.82$_{\pm0.04}$(0.35)	83.27$_{\pm3.81}$(16.73)	10.74	100.68
RL	96.15$_{\pm0.46}$(3.85)	100.00$_{\pm0.00}$(0.00)	92.21$_{\pm0.07}$(0.20)	88.10$_{\pm0.04}$(0.07)	100.00$_{\pm0.00}$(0.00)	0.82	200.23
GA	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	81.42$_{\pm1.99}$(10.59)	78.11$_{\pm2.03}$(10.06)	100.00$_{\pm0.00}$(0.00)	4.13	0.76
NegGrad+	97.46$_{\pm1.34}$(2.54)	99.00$_{\pm1.00}$(1.00)	92.17$_{\pm0.03}$(0.16)	87.90$_{\pm0.06}$(0.27)	96.58$_{\pm0.27}$(3.42)	1.48	69.14
SalUn	95.35$_{\pm0.88}$(4.65)	100.00$_{\pm0.00}$(0.00)	92.06$_{\pm0.09}$(0.05)	88.01$_{\pm0.01}$(0.16)	100.00$_{\pm0.00}$(0.00)	0.97	174.67
NegTV	97.85$_{\pm0.15}$(2.15)	100.00$_{\pm0.00}$(0.00)	91.39$_{\pm0.02}$(0.62)	87.60$_{\pm0.02}$(0.57)	99.15$_{\pm0.00}$(0.85)	0.84	1.24
MCU	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	92.32$_{\pm0.03}$(0.21)	87.92$_{\pm0.11}$(0.25)	100.00$_{\pm0.00}$(0.00)	0.09	105.49
MCU$_\beta$	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	92.18$_{\pm0.05}$(0.17)	88.00$_{\pm0.09}$(0.17)	100.00$_{\pm0.00}$(0.00)	**0.07**	98.12
			Tiny-ImageNet with VGG-16-BN				
RT	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.34$_{\pm0.03}$(0.00)	56.94$_{\pm0.11}$(0.00)	100.00$_{\pm0.00}$(0.00)	0.00	42.06
FT	74.27$_{\pm2.45}$(25.73)	78.67$_{\pm5.25}$(21.33)	99.29$_{\pm0.02}$(0.05)	56.71$_{\pm0.12}$(0.23)	90.53$_{\pm2.07}$(9.47)	11.36	4.29
RL	98.87$_{\pm1.04}$(1.13)	100.00$_{\pm0.00}$(0.00)	98.83$_{\pm0.01}$(0.51)	56.52$_{\pm0.10}$(0.42)	100.00$_{\pm0.00}$(0.00)	0.41	7.24
GA	91.80$_{\pm0.59}$(8.20)	87.33$_{\pm0.94}$(12.67)	94.75$_{\pm0.06}$(4.59)	52.86$_{\pm0.05}$(4.08)	96.60$_{\pm0.16}$(3.40)	6.59	0.13
NegGrad+	94.76$_{\pm1.50}$(5.24)	93.60$_{\pm3.67}$(6.40)	99.33$_{\pm0.03}$(0.01)	56.73$_{\pm0.06}$(0.21)	97.33$_{\pm1.97}$(2.67)	2.91	2.25
SalUn	99.27$_{\pm0.62}$(0.73)	100.00$_{\pm0.00}$(0.00)	98.95$_{\pm0.02}$(0.39)	56.58$_{\pm0.15}$(0.36)	100.00$_{\pm0.00}$(0.00)	0.30	7.25
NegTV	0.50$_{\pm0.10}$(99.50)	50.00$_{\pm0.00}$(50.00)	99.38$_{\pm0.02}$(0.04)	56.96$_{\pm0.00}$(0.02)	6.10$_{\pm0.90}$(93.9)	48.69	0.20
MCU	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.10$_{\pm0.01}$(0.24)	56.44$_{\pm0.02}$(0.50)	100.00$_{\pm0.00}$(0.00)	**0.15**	5.78
MCU$_\beta$	100.00$_{\pm0.00}$(0.00)	100.00$_{\pm0.00}$(0.00)	99.07$_{\pm0.01}$(0.27)	56.47$_{\pm0.09}$(0.47)	100.00$_{\pm0.00}$(0.00)	**0.15**	5.77



(a) $\beta = 0.1$ (b) $\beta = 0.15$ (c) $\boxed{\beta = 0.2}$ (d) $\beta = 0.25$ (e) $\beta = 0.3$

Figure 4: Ablation study for β on MCU. Overall, increasing β effectively enhances the unlearning effect but damages retaining predictive performance, while decreasing β weakens the ability of the pathway to forget data.



(a) $k = 0.1$ (b) $k = 0.3$ (c) $\boxed{k = 0.5}$ (d) $k = 0.8$ (e) $k = 1.0$

Figure 5: Ablation study for k on MCU. As k increases, the average accuracy gap decreases, but the effective region also shrinks.

The adaptive β not only simplifies the training process but also enhances the effectiveness of our MCU framework.

The results highlight the superiority of nonlinear unlearning over the linear method NegTV, especially in the class-wise scenario in CIFAR-10 and Tiny-ImageNet. In our experiments of the class-wise scenario, we attempted to optimize NegTV by extensively tuning its scaling hyperparameter, but encountered a persistent dilemma: NegTV either resulted in under-forgetting (failing to adequately remove the influence of the forgetting class) or over-forgetting (excessively degrading model performance). This stark trade-off highlights the inherent challenge of weight entanglement in linear approaches, which struggle to achieve the balance required for effective class-wise unlearning.

As a strong baseline, SalUn and RL is generally second only to MCUs and performs especially well in the class-wise forgetting scenario. However, SalUn and RL tend to exhibit overly strong resistance to MIA, often deviating significantly from RT in terms of membership privacy. While higher MIA efficacy is typically desirable for privacy, in the context of MU, the goal is to align with the RT baseline rather than excessively suppress MIA scores. Excessive deviation from RT in MIA efficacy could indicate a shift in model behavior that may introduce unintended privacy risks, as adversaries might exploit this shift to infer whether unlearning has occurred.

In terms of RTE, our methods remain competitive with baselines. The total runtime of the MCU mainly consists of three components: pre-unlearning model training, curve training, and optimal model selection. Taking 10% random data forgetting on CIFAR-10 dataset as an example, these steps take 2.96, 2.76, and 1.1 minutes, respectively. When served as a plug-and-play enhancement to existing MU methods, MCU requires only 2.76 additional minutes on average in this case. For class-wise forgetting, MCUs demonstrate significantly higher efficiency, as it requires fewer training epochs to achieve strong performance.

Ablation Study. To better understand the role of hyperparameters, β, k, and k_r within our MCU, we conduct an ablation study on CIFAR-10 with PreResNet-110 under 10% random data forgetting scenario. Figures 4-6 maintain the same format as Figure 3, with red-framed sub-captions indicating our default settings, i.e., $\beta = 0.2$, $k = 0.5$, and $k_r = 0.1$. For each ablation experiment, we vary one parameter while keeping the others fixed at their default values.

Higher β value leads to a smaller average accuracy gap in Figure 4. Notably, when $\beta = 0.3$, the average gap is only 0.66. However, increasing β also results in a reduced effective region. This suggests that while a larger β improves forgetting, it leads to a degradation in model utility. Clearly, $\beta = 0.2$ offers the best balance between average accuracy gap and effective region. Nonetheless, choosing a larger β can still be a viable and wise option when minimizing the accuracy gap is the primary objective, and the effective region is of secondary importance.

Similarly, we analyze the impact of k and k_r, in our mask strategy. A larger k allows more parameters retained for training,

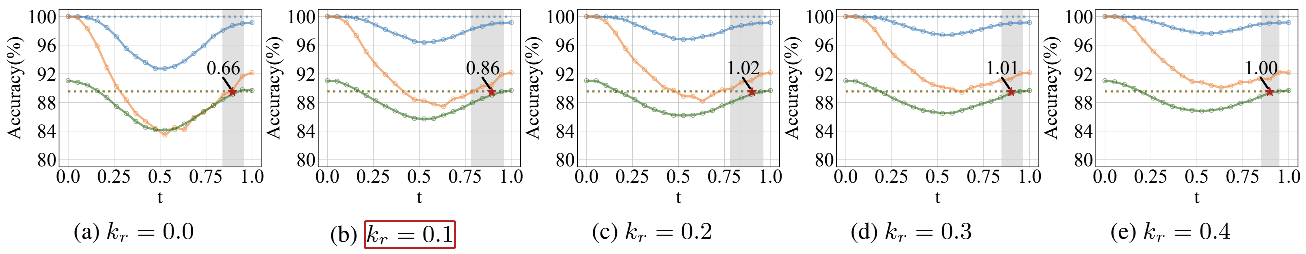


Figure 6: Ablation study for k_r on MCU. When $k_r = 0$, we preserve all parameters important to retaining data, leading to a noticeable drop in $\mathcal{D}_r$ accuracy during the unlearning process.

which significantly reduces the accuracy on $\mathcal{D}_f$, $\mathcal{D}_r$, and $\mathcal{D}_t$, especially $\mathcal{D}_f$ (orange lines in Figure 5). As for k_r, increasing k_r results in the removal of essential parameters related to $\mathcal{D}_r$, thereby effectively preserving the accuracy on $\mathcal{D}_r$ (blue lines in Figure 6). In our experiments, we set $k = 0.5$ and $k_r = 0.1$ as default values, as they provide a good balance between enhancing forgetting quality and maintaining predictive performance.

Effectiveness across Under-forgetting and Over-forgetting Pre-unlearning Models. To further demonstrate the versatility of MCU$_\beta$, we evaluate its ability to handle both under-forgetting and over-forgetting scenarios in a pre-unlearning model. While Figure 7a shows the under-forgetting case where RL is trained for 15 epochs, we intentionally over-trained RL for 20 epochs as an over-forgetting pre-unlearning model in Figure 7b. As shown in Figure 7, MCU$_\beta$-RL consistently enhances RL in both scenarios. Specifically, it reduces the average gap across D_f, D_r, D_t to 0.62 in the under-forgetting scenario and 0.43 in the over-forgetting scenario. These results highlight MCU$_\beta$'s adaptability across different pre-unlearning conditions. This is attributed to the adaptive unlearning penalty coefficient β, with the alignment condition ❶ handling over-forgetting and conditions ❷ and ❸ handling under-forgetting.

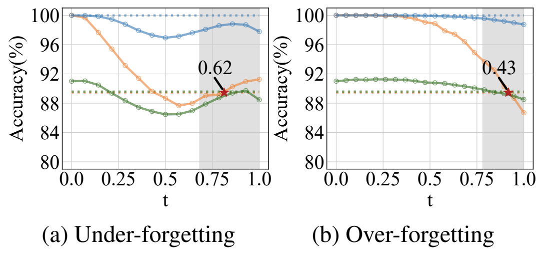


Figure 7: Effectiveness of MCU$_\beta$ across both under-forgetting and over-forgetting pre-unlearning model θ_p.

Stability to Scarce Retaining Data. In Figure 8 and 9, we validate the stability of our framework to scarce retaining data. These experiments were conducted using MCU$_\beta$ with NegGrad+ as pre-unlearning model on CIFAR-10 under the 10% random data forgetting scenario. As illustrated in Figure 8, the accuracy values of the optimal unlearning model on the MCU pathway remain stable across varying retaining data proportions. In Figure 9, we further present the results of nonlinear pathway searching across varying proportions of retaining data $\mathcal{D}_r$, ranging from 10% to 100%. MCU$_\beta$ consistently outperforms other unlearning methods across all retaining data proportion settings. As expected, the optimal performance is achieved when utilizing 100% of the retaining data for curve training. In this case, the pathway searching process fully leverages the entire dataset, leading to the highest retaining accuracy and minimizing any degradation in model utility. By comparison, the worst performance occurs when only 10% or 20% of the retaining data is available. In these cases, the retaining accuracy drops significantly, indicating that an insufficient amount of retaining data negatively impacts the learning process. However, when the proportion of $\mathcal{D}_r$ exceeds 30%, retaining accuracy remains consistently high with relatively small average accuracy gaps. This demonstrates the inherent stability of our MCU framework even under limited retaining data conditions. This stems from our framework of searching nonlinear pathways in the parameter space between the original and pre-unlearning models as end points, which effectively preserves critical retaining data information along the pathway. Consequently, an effective unlearning model can consistently be identified across the pathway, regardless of the scarce retaining data used. Overall, we suggest that maintaining at least 30% of the retaining data during pathway searching is enough to achieve a balance between training efficiency, effective unlearning, and model utility.

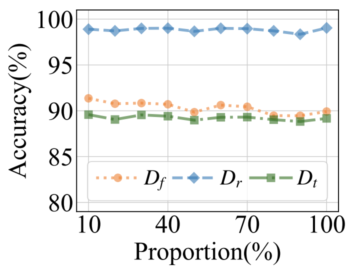


Figure 8: The accuracy on $\mathcal{D}_f$, $\mathcal{D}_r$, and $\mathcal{D}_t$ across different proportions of retaining data used in our training process. It shows that all accuracy performance remains stable even with 10% retaining data.

E. Pseudo Code of MCU Framework

The pseudo code can be found in 1. We present it with three components: parameter mask generating, nonlinear pathway searching and optimal model, and effective unlearning region searching.

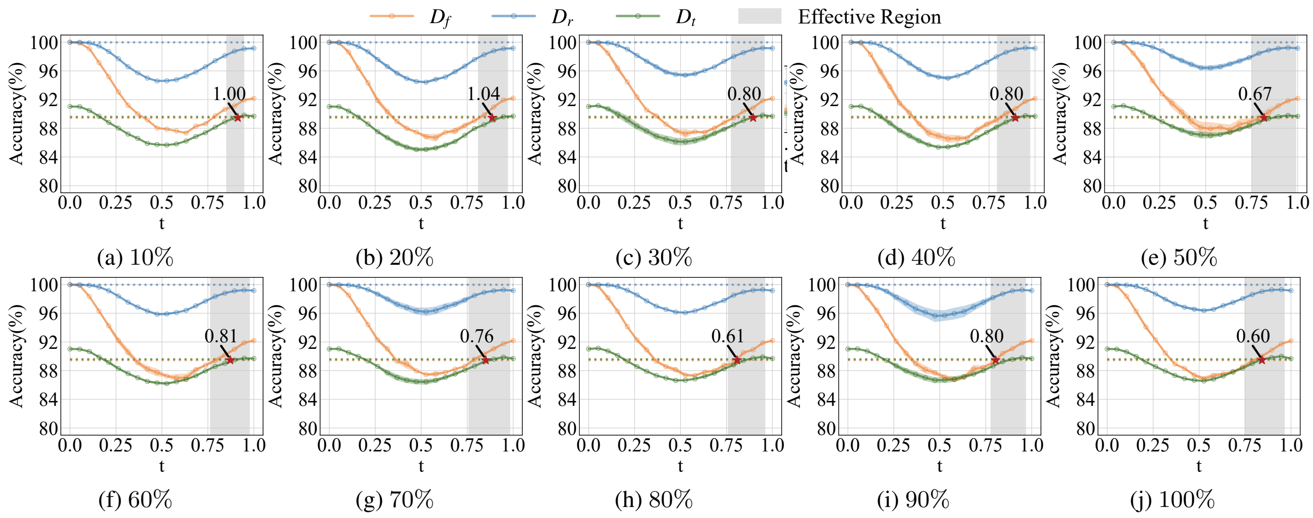


Figure 9: Performance with different proportions of retaining data in pathway searching process. The results show that MCU$_\beta$ consistently outperforms other unlearning methods across all retaining data proportion settings.

Algorithm 1 Pseudo code of MCU$_\beta$

```
1: Hyper-parameters: number of iterations $n$, learning rate $\eta$, parameter mask parameter $k$ and $k_r$
2: Require: original model $\boldsymbol{\theta}_o$, pre-unlearning model $\boldsymbol{\theta}_p$, training accuracy and test accuracy on the original model $\boldsymbol{\theta}_o$
3: # 1. Generate a parameter mask
4: Compute loss $\mathcal{L}(\mathcal{D}_r; \boldsymbol{\theta}_o)$ and $\mathcal{L}(\mathcal{D}_f; \boldsymbol{\theta}_o)$
5: Compute gradient $\nabla_{\boldsymbol{\theta}_o}\mathcal{L}(\mathcal{D}_r; \boldsymbol{\theta}_o)$ and $\nabla_{\boldsymbol{\theta}_o}\mathcal{L}(\mathcal{D}_f; \boldsymbol{\theta}_o)$
6: Calculate $\|\nabla_{\boldsymbol{\theta}_o^i}\mathcal{L}(\mathcal{D}_r; \boldsymbol{\theta}_o)\|_2/|\boldsymbol{\theta}_o^i|$ and $\|\nabla_{\boldsymbol{\theta}_o^i}\mathcal{L}(\mathcal{D}_f; \boldsymbol{\theta}_o)\|_2/|\boldsymbol{\theta}_o^i|$ for each parameter
7: Filter out top $k_r$ proportion of parameters based on $\|\nabla_{\boldsymbol{\theta}_o^i}\mathcal{L}(\mathcal{D}_r; \boldsymbol{\theta}_o)\|_2/|\boldsymbol{\theta}_o^i|$ and generate mask $\boldsymbol{m}_r$
8: Preserve top $k$ proportion of parameters based on $\|\nabla_{\boldsymbol{\theta}_o^i}\mathcal{L}(\mathcal{D}_f; \boldsymbol{\theta}_o)\|_2/|\boldsymbol{\theta}_o^i|$ and generate mask $\boldsymbol{m}_f$
9: Calculate parameter mask $\boldsymbol{m} = \mathbb{1}(\boldsymbol{m}_r \;\&\; \boldsymbol{m}_f)$
10: # 2. Search pathways in parameter space
11: $\beta \leftarrow 0.5$    (unlearing penalty coefficient is initialized as $0.5$)
12: for $i \leftarrow 1, 2, ..., n$ do
13:     Sample $t \sim U(0, 1)$
14:     Compute accuracy of retaining data and forgetting data
15:     Adaptively update $\beta$
16:     Compute cross-entropy loss $\mathcal{L}(\mathcal{D}_r; \phi_{\boldsymbol{\theta}_c}(t))$ for retaining data
17:     Compute cross-entropy loss $\mathcal{L}(\mathcal{D}_f; \phi_{\boldsymbol{\theta}_c}(t))$ for forgetting data
18:     Compute MCU loss $\mathcal{L}_{mcu} = \mathcal{L}(\mathcal{D}_r; \phi_{\boldsymbol{\theta}_c}(t)) - \beta \cdot \mathcal{L}(\mathcal{D}_f; \phi_{\boldsymbol{\theta}_c}(t))$
19:     Compute gradient $\nabla_{\boldsymbol{\theta}_c \odot \boldsymbol{m}}\mathcal{L}_{mcu}$ based on the parameter mask $\boldsymbol{m}$
20:     Update $\boldsymbol{\theta}_c$ using gradient descent:
21:         $\boldsymbol{\theta}_c \odot \boldsymbol{m} \leftarrow \boldsymbol{\theta}_c \odot \boldsymbol{m} - \eta\nabla_{\boldsymbol{\theta}_c \odot \boldsymbol{m}}\mathcal{L}_{mcu}$
22: end for
23: # 3. Search optimal model and effective unlearning region on the pathway
24: Sample $t \sim U(0, 1)$
25: for each $t$ do
26:     Compute accuracy of retaining data $\mathcal{D}_r$, forgetting data $\mathcal{D}_f$ and test data $\mathcal{D}_t$
27:     Calculate retaining gap, forgetting gap and test gap and their average gap
28:     Compare average gap with pre-unlearning model $\boldsymbol{\theta}_p$ and search the optimal model and effective unlearning models
29: end for
30: Return: The optimized pathway $\phi_{\boldsymbol{\theta}_c}(t)$ which connects $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_p$, optimal unlearning model $\boldsymbol{\theta}_u^*$ and a range of $t$
    where can generate effective unlearning models $\boldsymbol{\theta}_u$ across pathway
```