An Illusion of Unlearning? Assessing Machine Unlearning Through Internal Representations

Yichen Gao[1,*] **Altay Unal**[2,*] **Akshay Rangamani**[2,†] **Zhihui Zhu**[1,†]
[1]Department of Computer Science & Engineering, The Ohio State University
[2]Department of Data Science, New Jersey Institute of Technology
[*]Equal contribution [†]Equal advising

Abstract

While numerous machine unlearning (MU) methods have recently been developed with promising results in erasing the influence of forgotten data, classes, or concepts, they are also highly vulnerable—for example, simple fine-tuning can inadvertently reintroduce erased concepts. In this paper, we address this contradiction by examining the *internal representations* of unlearned models, in contrast to prior work that focuses primarily on output-level behavior. Our analysis shows that many state-of-the-art MU methods appear successful mainly due to a misalignment between last-layer features and the classifier—a phenomenon we call *feature–classifier misalignment*. In fact, hidden features remain highly discriminative, and simple linear probing can recover near-original accuracy. Assuming neural collapse in the original model, we further demonstrate that adjusting only the classifier can achieve negligible forget accuracy while preserving retain accuracy, and we corroborate this with experiments using classifier-only fine-tuning. Motivated by these findings, we propose MU methods based on a class-mean features (CMF) classifier, which explicitly enforces alignment between features and classifiers. Experiments on standard benchmarks show that CMF-based unlearning reduces forgotten information in representations while maintaining high retain accuracy, highlighting the need for faithful representation-level evaluation of MU.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

1 INTRODUCTION

Machine Unlearning (MU) (Bourtoule et al., 2021) aims to remove the influence of specific training samples from a model without retraining it from scratch. This capability is increasingly critical in practice due to requirements such as compliance with privacy regulations, protection of intellectual property and copyrighted content, and safety concerns arising from the retention of harmful or biased data (Voigt and Von dem Bussche, 2017). Beyond performance, unlearning directly impacts the trustworthiness and ethical deployment of machine learning systems (Jin et al., 2023) since some of the data might be tainted (Jagielski et al., 2018) or the data might contain harmful biases (Fabbrizzi et al., 2022).

Due to these practical and ethical demands, MU has been extensively studied and empirically demonstrated across a range of tasks, including in classifiers (Choi and Na, 2023; Golatkar et al., 2020; Tarun et al., 2023) and generative models (Fan et al., 2023; Li et al., 2024). MU in classification focuses on forgetting individual examples or entire classes used in training, with the goal of erasing their influence while preserving performance on the remaining data. MU in generative models targets the removal of specific concepts, ensuring that the model cannot produce outputs based on them.

Despite these encouraging results, recent studies have also revealed significant vulnerabilities of MU. For example, MU can be unstable in generative models, where forgotten concepts may re-emerge. In particular, simply fine-tuning on seemingly unrelated images can inadvertently reintroduce erased content (Suriyakumar et al., 2024). In addition, unintended concepts can also be affected by the erase process of a concept (Lu et al., 2024; Yu et al., 2025). This raises a fundamental question: *do unlearned models truly forget, and how should we faithfully assess their performance?*

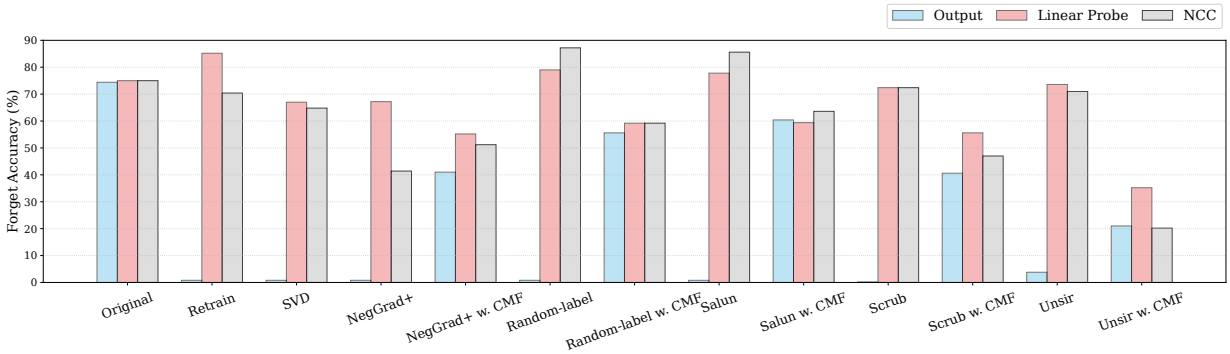


Figure 1: Comparison of forget accuracies evaluated in the output level (blue bars) VS feature-level via linear probe (red bars) and nearest class center accuracy (NCC, grey bars) for original, retain-only retrain, and various MU methods and those with CMF classifier on cifar100 with forget 1 class scenario.

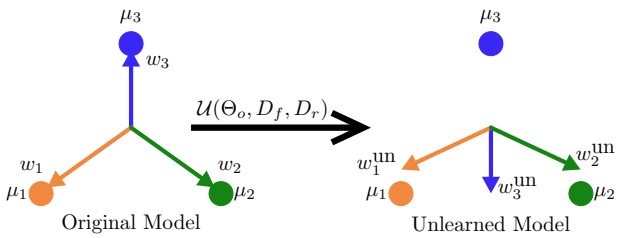
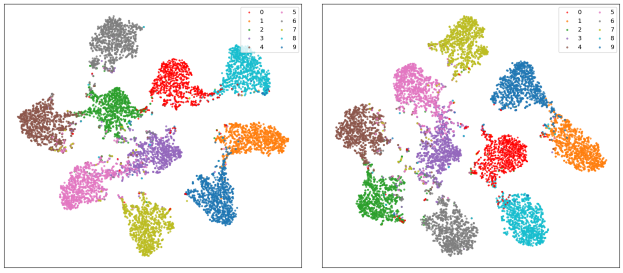


Figure 2: Visualization of Proposition 1. We observe a misalignment caused by the unlearning methods between the class mean and the corresponding classifier weights for the forget class (blue) while the alignment is mainly preserved for the retain classes (green and orange). The spheres represent the class means while the arrows represent the last layer classifier weights.



(a) Original (b) Random-label unlearned

Figure 3: t-SNE on CIFAR-10 with (a) original model and (b) Random-label unlearned model. The forgotten class (dark blue points) remains linearly separable in unlearned models.

To investigate these questions, we focus on forgetting entire classes in the context of image classification, since the presence of label information makes it easier to assess the effectiveness of unlearning through accuracy metrics. Classification has served as a testbed for developing and validating MU methods, and heuristic approaches have been proposed in recent years (Choi and Na, 2023; Kodge et al., 2024; Kurmanji et al., 2023; Tarun et al., 2023). Yet, measuring the effectiveness of MU remains a challenging problem. Current evaluations primarily rely on output-level metrics, which measure model predictions on the forget set (*forget accuracy*) and the retain set (*retain accuracy*). However, it remains unclear whether forgetting truly occurs at the level of internal feature representations or whether MU methods merely suppress classifier outputs while forgotten concepts persist in the representation space. In this work, we propose to assess unlearning effectiveness by studying the internal representations rather than relying solely on output-level metrics.

Contribution Our main contributions are summarized as follows:

- We find that while many state-of-the-art MU methods, including Random Label, SalUn, NegGrad+, SCRUB, and UNSIR, achieve negligible forget accuracy, their hidden-layer features remain highly discriminative; see Figure 3 for t-SNE plot. As shown in Figure 1, a simple linear probe on the last-layer representation can recover near-original accuracy. This reveals that even when unlearning appears successful according to standard metrics (forget and retain accuracy), current methods often fail to remove information from the hidden representation space, leaving latent traces of forgotten data that can be recovered with retraining.

- Inspired by the *neural collapse (NC)* phenomenon in deep classifiers (Papyan et al., 2020) (see Section 2.1 for details), we study the alignment between class-mean features and the classifier. We show that after unlearning, self-duality between the classifier and last-layer class-mean features persists for retain classes—classifiers remain almost perfectly matched with their class means—but for forget classes there exists significant *feature–classifier misalignment* as depicted in Figure 2. Assuming NC in the original model, we further demonstrate that a simple MU method can be constructed by adjusting the classifier, yielding negligible accuracy on the forget set

while preserving accuracy on the retain set. We corroborate this analysis by applying SOTA MU methods with classifier-only fine-tuning, which achieve comparable forgetting and retaining accuracy at the output level, underscoring the limitations of current evaluation metrics.

- Motivated by our analysis, we propose a representation-level unlearning framework that enforces alignment between features and the classifier, ensuring that forgetting occurs within the hidden representations as well. Specifically, we employ a class-mean features (CMF) classifier, which explicitly sets each classifier weight to the mean feature vector of its corresponding class and can be seamlessly integrated into existing MU methods. As shown in Figure 1, experiments on standard benchmarks demonstrate that CMF-based unlearning substantially reduces the retention of forgotten information at the representation level (i.e., achieving much lower forget accuracy under linear probing), while maintaining high accuracy on retained data.

We make our code publicly available at `https://github.com/ycgao1/CMF_Unlearning`.

2 PRELIMINARIES

2.1 Neural Networks and Neural Collapse

A standard deep neural network (DNN) classifier $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^K$ consists of a multi-layer nonlinear compositional feature mapping $\phi_{\boldsymbol{\theta}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^d$ with $\boldsymbol{\theta}$ denoting the network parameters in the feature mapping and a linear classifier $(\boldsymbol{W}, \boldsymbol{b})$ with $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]^\top \in \mathbb{R}^{K\times d}$ and $\boldsymbol{b}$, which can expressed as

$$f_{\boldsymbol{\Theta}}(\boldsymbol{x}) = \boldsymbol{W}\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{b} \in \mathbb{R}^K. \tag{1}$$

Here $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \boldsymbol{W}, \boldsymbol{b}\}$ denotes all the network parameters. The feature extractor $\phi_{\boldsymbol{\theta}}(\cdot)$ generates the data-dependent feature vectors in $\mathbb{R}^d$, while the linear classifier $(\boldsymbol{W}, \boldsymbol{b})$ determines the linear decision boundary in the feature space.

With an appropriate loss function, the parameters $\boldsymbol{\Theta}$ of the network are optimized to learn the underlying relation between an input sample $\boldsymbol{x}$ and its corresponding target $\boldsymbol{y}$, such that the network output $f_{\boldsymbol{\Theta}}(\boldsymbol{x})$ approximates $\boldsymbol{y}$. Specifically, let $\mathcal{D} = \{(\boldsymbol{x}_{i,k}, \boldsymbol{y}_{i,k})\}_{i=1}^N$ be a dataset of N training samples, where $\boldsymbol{x}_{i,k}$ is the i-th sample from k-th class and $\boldsymbol{y}_{i,k} \in \mathbb{R}^K$ is the corresponding one-hot label vector. The parameters $\boldsymbol{\Theta}$ are learned by minimizing the empirical risk over all the training samples:

$$\boldsymbol{\Theta}_o = \arg\min_{\boldsymbol{\Theta}} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathcal{L}(f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{i,k}), \boldsymbol{y}_{i,k}), \tag{2}$$

where $\mathcal{L}(f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{i,k}), \boldsymbol{y}_{i,k})$ is a predefined loss function, such as the cross-entropy loss, that appropriately measures the discrepancy between the output $f_{\boldsymbol{\Theta}}(\boldsymbol{x}_{i,k})$ and the target $\boldsymbol{y}_{i,k}$.

Neural Collapse Neural Collapse ($\mathcal{NC}$) (Papyan et al., 2020) is an intriguing phenomenon observed in the last-layer classifier and feature representations during the terminal phase of training (TPT), when the training error approaches zero. In this regime, features from the final layer align with their corresponding class mean vectors, which collectively form a simplex equiangular tight frame (ETF) structure.

More precisely, $\mathcal{NC}$comprises the following properties: **(i) Variability collapse ($\mathcal{NC}_1$):** features within each class collapse to their class mean; **(ii) Simplex ETF structure ($\mathcal{NC}_2$):** the class means, centered at their global mean, are not only linearly separable but are maximally separated and form a simplex ETF; **(iii) Feature–classifier alignment ($\mathcal{NC}_3$):** each class mean is perfectly aligned with the corresponding last-layer linear classifier; **(iv) Nearest class center decision rule ($\mathcal{NC}_4$):** the last-layer classifier becomes equivalent to a nearest class center (NCC) classifier.

To quantify $\mathcal{NC}$, let $\boldsymbol{h}_{i,k} = \phi_{\boldsymbol{\theta}}(\boldsymbol{x}_{i,k})$ denote the learned feature representation of sample $\boldsymbol{x}_{i,k}$ from class k. We define the class-wise mean features and the global mean feature as

$$\boldsymbol{\mu}_k := \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{h}_{i,k}, \qquad \boldsymbol{\mu}_G := \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{\mu}_k. \tag{3}$$

Neural collapse characterizes the convergence of features $\boldsymbol{h}_{i,k}$ toward their corresponding class means $\boldsymbol{\mu}_k$, along with the alignment of the classifier weights $\boldsymbol{w}_k$ with these means.

In the context of unlearning, two NC measures are particularly informative. The first is **feature–classifier alignment**, measured by

$$\mathcal{NC}_3 := \left\| \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|} - \frac{\boldsymbol{\mu}_k - \boldsymbol{\mu}_G}{\|\boldsymbol{\mu}_k - \boldsymbol{\mu}_G\|} \right\|, \tag{4}$$

which quantifies the alignment between the normalized classifier weight $\boldsymbol{w}_k$ and the centered class mean $\boldsymbol{\mu}_k - \boldsymbol{\mu}_G$.

The second measure is the **nearest class center (NCC) classification accuracy**, defined as

$$\text{NCC} := \mathbb{P}\left[y = \arg\min_k \|\phi_{\boldsymbol{\theta}}(\boldsymbol{x}) - \boldsymbol{\mu}_k\|_2 \right], \tag{5}$$

where $\phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ denotes the representation of input $\boldsymbol{x}$, and the probability is taken over data samples $(\boldsymbol{x}, y)$. In words, under the NCC rule, a sample $\boldsymbol{x}$ is assigned to the closest class mean.

In this paper, we adopt $\mathcal{NC}$ analysis as a diagnostic tool for unlearning by tracking the $\mathcal{NC}_3$ and NCC met-

rics throughout the unlearning process.

2.2 Machine Unlearning

Machine unlearning (MU) is a paradigm that aims to make a machine learning (ML) model forget about certain data. Originally motivated by privacy concerns and the "right to be forgotten", the goal of machine unlearning is to allow people to opt out of their data being used in the training of ML models. Machine unlearning is also useful in contexts outside of privacy such as correcting models trained on erroneous data (Ali et al., 2025), removing classes from classifier, etc. Since training ML models from scratch may be quite expensive, machine unlearning aims to provide a sustainable solution for such cases.

Given a dataset $\mathcal{D}$, let $\mathcal{D}_f \subset \mathcal{D}$ denote the subset of data targeted for unlearning, referred to as the forget set. Its complement, $\mathcal{D}_\mathrm{r} = \mathcal{D} \setminus \mathcal{D}_\mathrm{f}$, is the portion of the dataset to be retained, referred to as the retain set. The content of forget and retain sets varies according to the application. For the class unlearning scenario, $\mathcal{D}_\mathrm{f}$ and $\mathcal{D}_\mathrm{r}$ denote data corresponding to the forgotten classes C_f and retained classes C_r, respectively. $\mathcal{D}_\mathrm{f}$ contains all the examples belonging to C_f while $\mathcal{D}_\mathrm{r}$ contains the rest of the training data.

In the literature, retraining a fresh model $\mathbf{\Theta}_\mathrm{r}$ solely on $\mathcal{D}_\mathrm{r}$ is widely regarded as the *gold standard* for MU (Bourtoule et al., 2021; Thudi et al., 2022). Nevertheless, full retraining is both computationally expensive and time-consuming, which is impractical for large-scale models or frequent removal requests. Recent research therefore focuses on designing approximate methods that modify the original model $\mathbf{\Theta}_\mathrm{o}$ to achieve the effect of unlearning. Formally, given training data $\mathcal{D}$ and an original trained model $\mathbf{\Theta}_\mathrm{o}$, an unlearning algorithm defines a transformation

$$\mathbf{\Theta}_\mathrm{u} \;=\; \mathcal{U}(\mathbf{\Theta}_\mathrm{o}, \mathcal{D}_\mathrm{f}, \mathcal{D}_\mathrm{r}), \tag{6}$$

where $\mathbf{\Theta}_\mathrm{u}$ is the unlearned model and $\mathcal{U}$ denotes the unlearning operator.

NegGrad (Choi and Na, 2023; Golatkar et al., 2020) performs gradient ascent on the forget set, sometimes combined with a retain loss to mitigate over-forgetting. Random-label (Golatkar et al., 2020) assigns random labels to the forget samples, forcing the model to fit noise and degrade its predictive ability on $\mathcal{D}_\mathrm{f}$. Saliency Unlearning (SalUn) (Fan et al., 2023) improves upon this by updating only parameters most salient to the forget set, enhancing efficiency and stability. SCRUB (Kurmanji et al., 2023) formulates unlearning as a selective knowledge distillation problem, encouraging the model to diverge from the teacher on the forget set while preserving behavior on the retain set. UNSIR (Tarun et al., 2023) generates error-maximizing noise to impair model weights associated with the forget classes, followed by a repair step using retain data to restore overall model performance. Beyond gradient-based strategies, SVD-based unlearning (Kodge et al., 2024) offers a gradient-free alternative by projecting feature representations onto the orthogonal complement of the forget subspace to suppress discriminative information.

3 EVALUATION OF UNLEARNING

Evaluating the effectiveness of machine unlearning is challenging. Currently, machine unlearning methods are mainly evaluated using output-level metrics, which focus solely on model predictions. These metrics are unsuitable for assessing unlearning in the learned representation space (Xu et al., 2024) since the inner representation space has larger dimensionality. In addition to output-level metrics, some models consider relearn time as a metric for evaluating machine unlearning (Xue et al., 2025), which refers to the number of epochs for an unlearned model to relearn and restore its performance on the forgotten data. However, this is also unsuitable, since we will explain further that performance on forgotten data can be easily retrieved.

In this section, we first describe the current output level evaluation metrics, then propose feature-level evaluation metrics for machine unlearning. In this section, we first review existing output-level evaluation metrics, and then introduce feature-level evaluation metrics for machine unlearning. Most prior work evaluates the effectiveness of unlearning by measuring the performance of the entire network at the output layer. From this perspective, we refer to such evaluations as shallow unlearning. In contrast, we also assess the effectiveness of unlearning at the feature level, which we term deep unlearning.

3.1 Evaluation Metrics for Shallow Unlearning

Broadly, evaluation is based on two aspects: unlearning effectiveness, measured on the forget set, and post-unlearning model utility, measured on the retain set. Accuracy-based metrics are the most widely used. Specifically, given a testing datset $\mathcal{D}$, *forget accuracy*, denoted by Acc_f, quantifies prediction performance on the forget set $\mathcal{D}_\mathrm{f}$, while retain accuracy, denoted by Acc_r, measures performance on the retain set $\mathcal{D}_\mathrm{r}$. An effective unlearning method should substantially reduce Acc_f while maintaining high Acc_r on test data.

Another common evaluation metric is the success of membership inference attacks (MIA) (Shokri et al., 2017), which tests whether an adversary can determine if a sample was included in training. MIA is a useful metric to measure privacy guarantees where individuals would like their data to be excluded. In this paper

Table 1: Evaluation of various MU methods on three datasets for unlearning certain number of classes. For all the (unlearned) models, we report both mean forget accuracy and mean retain accuracy evaluated for the entire model (labeled as Output) and the feature mapping by linear probe and NCC classification accuracy.

Method	Accuracy	CIFAR-10				CIFAR-100				Tiny-ImageNet			
		1		3		1		10		1		20	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Output	93.98	93.98	94.00	93.94	74.61	74.40	74.47	75.88	65.27	58.80	65.15	66.02
	Linear Probe	94.02	94.02	94.03	94.00	74.53	75.00	74.38	75.90	65.10	60.80	64.97	66.08
	NCC	94.00	93.99	94.03	93.92	74.40	75.00	74.28	75.68	64.65	60.00	64.56	65.00
Retain-only Retrain	Output	94.74	0.00	95.37	0.00	76.01	0.00	76.50	0.00	66.52	0.00	66.38	0.00
	Linear Probe	90.49	77.35	85.64	67.33	74.09	85.20	69.34	60.94	65.90	46.40	65.21	30.36
	NCC	93.31	47.06	91.37	37.07	73.90	70.40	70.98	43.18	63.57	71.20	59.37	44.30
Retain-only FT	Output	94.26	47.67	95.24	52.48	74.08	53.20	74.53	64.96	65.26	37.60	65.60	50.92
	Linear Probe	93.94	89.71	94.14	90.44	73.89	73.80	73.97	74.50	64.44	56.00	64.05	63.82
	NCC	93.70	89.63	93.80	88.75	74.04	73.20	74.08	73.56	64.00	57.60	63.90	63.58
NegGrad+	Output	92.85	0.00	93.29	0.01	69.90	0.00	70.80	0.28	57.96	0.00	59.06	0.00
	Linear Probe	92.14	67.09	88.18	73.91	72.55	67.20	72.20	62.32	60.43	58.00	60.75	54.68
	NCC	91.33	52.00	87.28	59.17	71.58	41.40	70.08	41.24	59.01	37.60	56.45	36.78
SVD	Output	92.02	0.00	94.05	57.43	71.09	0.00	73.10	55.56	64.43	2.00	65.45	59.02
	Linear Probe	90.44	61.80	92.43	83.58	73.14	67.00	73.38	74.08	63.10	60.40	63.03	63.64
	NCC	90.11	34.54	93.23	72.32	71.81	64.80	73.33	72.22	64.65	60.00	64.56	65.00
Random-label	Output	92.93	0.00	94.14	0.00	72.33	0.00	72.19	0.00	65.48	0.40	64.85	0.98
	Linear Probe	92.65	92.49	92.45	90.25	73.08	79.00	72.08	72.08	64.07	58.00	62.02	57.82
	NCC	92.25	80.25	91.92	73.22	72.38	87.20	70.67	62.22	63.20	69.20	59.42	42.72
SalUn	Output	93.19	0.00	94.43	0.00	72.96	0.00	72.92	0.06	65.49	0.40	64.63	3.40
	Linear Probe	93.05	92.57	92.89	89.63	73.26	77.80	72.10	72.66	64.07	55.60	61.86	56.52
	NCC	91.31	93.70	91.99	68.45	72.65	85.60	70.62	63.34	63.33	68.80	59.07	39.70
SCRUB	Output	91.37	0.00	93.61	0.00	73.67	0.20	74.56	0.32	65.43	1.20	65.30	5.48
	Linear Probe	91.71	74.79	91.88	78.23	73.69	72.40	73.05	66.92	64.44	56.00	62.87	56.90
	NCC	89.96	55.30	89.73	47.52	73.51	72.40	72.35	53.58	64.37	60.40	61.41	53.50
UNSIR	Output	91.84	0.48	92.87	0.01	73.77	3.80	73.58	14.58	64.66	0.00	65.46	9.72
	Linear Probe	89.71	85.29	88.21	70.59	73.40	73.60	72.15	67.64	63.88	61.20	62.87	61.10
	NCC	89.73	62.72	87.73	49.42	73.05	71.00	71.26	59.24	63.16	59.20	61.80	56.14

we focus on class forgetting in the context of classification, and evaluating whether "forgotten" knowledge can be retrieved. In this context MIA is not a relevant metric (Kurmanji et al., 2023).

3.2 Evaluation Metrics for Deep Unlearning

According to (1), a DNN classifier consists of two components: the feature mapping $\phi_{\boldsymbol{\theta}}$ and the linear classifier. If the model performs poorly, the source of error may lie in either the feature mapping or the classifier. Similarly, in the context of unlearning, even if the classifier is updated to suppress performance on the forget set, the feature extractor may still preserve discriminative information about the forgotten data, which means that the applied unlearning method may not have succeeded at all. Output-level metrics, therefore, risk overlooking hidden representations that continue to encode forgotten concepts as they simplify the evaluation of the unlearning methods. A robust MU method is expected to remove the influence of the forget set across the entire model, particularly within the feature mapping.

Motivated by this, we propose to evaluate MU not only at the output level but also at the representation level. Specifically, we assess the effectiveness of the feature mapping in terms of its discriminative and predictive ability. A common tool for this purpose is the linear probe: a new linear classifier is trained on top of the frozen features using the full dataset $\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_f$, after which performance on the forget set and retain set is evaluated. We also adopt $\mathcal{NC}$ analysis as we mentioned in Section 2.1. $\mathcal{NC}$ describes how features $\boldsymbol{h}_{i,k}$ of each class converge to their class mean $\boldsymbol{\mu}_k$, and classifier weights $\boldsymbol{w}_k$ align with these means. We will be utilizing $\mathcal{NC}_3$ and $\mathcal{NC}_3$ for the unlearning to evaluate the unlearned features.

Overall, feature-level metrics (linear probe, $\mathcal{NC}_3$, NCC) provide a more faithful evaluation of unlearning by directly testing whether forgotten concepts survive in the internal representation, complementing output-level metrics. With these metrics, we can actually observe the true performance of the unlearning methods as the evaluation is not restricted to the accuracy metrics anymore. Unlike output-level evaluation metrics, there has not been a universally established

feature-level evaluation metric. Although there have been some recent developments in terms of defining a new feature-level metric regarding the representations based on information theory (Jeon et al., 2024) and Centered Kernel Alignment (CKA) (Kim et al., 2025), both metrics compare against a model trained on only the retain set $\mathcal{D}_r$ which is a significant computational expense.

4 THE ILLUSION OF UNLEARNING

Data, models, and unlearning setups. Following prior work (Fan et al., 2023; Kodge et al., 2024), we conduct class-unlearning experiments on both convolutional and transformer-based architectures. For convolutional models, we use ResNet architectures (He et al., 2016), evaluating ResNet-18 on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), and ResNet-50 on Tiny-ImageNet (Le and Yang, 2015). For each dataset, we consider both single-class and multi-class unlearning scenarios. Specifically, the numbers of forgotten classes $|\mathcal{D}_f|$ are $\{1, 3\}$ for CIFAR-10, $\{1, 10\}$ (corresponding to two super-classes) for CIFAR-100, and $\{1, 20\}$ for Tiny-ImageNet.

For ResNet-based experiments, we evaluate several types of models: (i) the *Original* model trained on the full dataset, (ii) the *Retain-only Retrain* model trained from scratch using only the retain dataset, (iii) the *Retain-only Fine-tuning (FT)* model obtained by fine-tuning the original model on the retain dataset, and (iv) various unlearned models produced by applying representative machine unlearning methods on the original model. In particular, we consider six unlearning algorithms: Retain-only FT, NegGrad+, Random Label, SalUn, SVD, SCRUB, and UNSIR, as described in Section 2.2 while experimental setup is described in Appendix A.3. We also report additional results with variance across ResNet and Vision Transformers (ViT-S/16) in Appendix D.

4.1 Shallow Unlearning with Persistent Discriminative Features

Table 1 displays the effectiveness of different MU methods in terms of mean output-level accuracies and mean feature-level accuracies (evaluated by linear probe and NCC) for both forget set (forget accuracy).

Observation 1: Representations remain linearly separable after unlearning. As shown in Table 1, many unlearning algorithms appear successful when evaluated at the output level: the accuracy on the forget set drops to nearly zero, suggesting effective forgetting. However when the learned representations are frozen and a linear probe is trained on top of them, the forget accuracy recovers to a high level. A similar

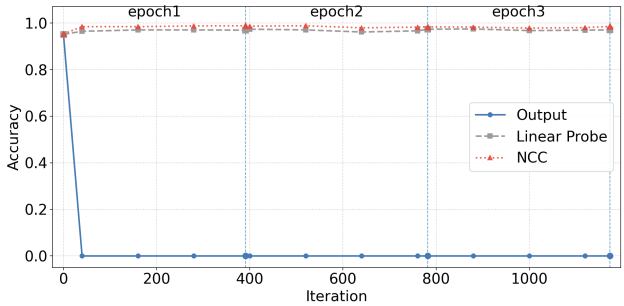


Figure 4: Learning curve of Random-label unlearning on CIFAR-10 when forgetting class 0 (airplane). While the output-level forget accuracy drops to zero quickly, the linear-probe and NCC accuracies remain consistently high throughout the unlearning process.

trend is observed under the NCC accuracy indicating that the "forgotten" representations still cluster around their class means. This implies that while current MU methods appear successful according to standard output-level metrics (forget and retain accuracy) they often fail to remove information from the hidden representation space, leaving latent traces of forgotten data that can be recovered by a simple classifier.

This phenomenon is further illustrated by the learning curves in Figure 4, using the random-label unlearning method an an example. The output-level forget accuracy drops to nearly zero within the first few iterations, whereas the linear-probe and NCC accuracies remain almost unchanged throughout the entire unlearning process. This suggests that unlearning primarily suppresses the output classifier early in training, while the underlying feature representations of the forgotten class remain largely preserved and linearly separable.

Remarkably, the linear separability of forget class representations also persists in the model trained only on the retain set (denoted as Retain-only Retrain in Table 1), a baseline commonly regarded as the "gold standard" for MU. Such a model achieves zero forget accuracy simply because it has not seen the forget data, yet linear probing still yields substantial forget accuracy. This is largely due to the transferability of DNN representations—typically viewed as one of their main advantages. In the context of unlearning, transferability makes it challenging to truly remove information at the representation level.

Hence in this paper we primarily evaluate unlearning algorithms on their feature-level forget and retain accuracies. Since the "gold standard" model already contains the transferable features, this suggests a potential trade-off between output-level and feature-level retain and forget accuracies. This opens the door to methods that can outperform Retain-only Retrain by performing worse on output level metrics while being

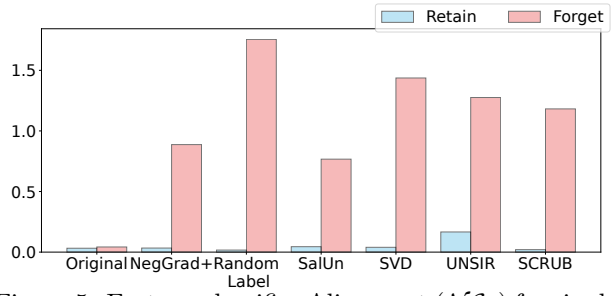


Figure 5: Feature-classifier Alignment ($\mathcal{NC}_3$) for single class forgetting on CIFAR-10: distance between class means and classifier weights for forget class is increased while the distance is preserved for retain class.

less transferable. For example, NegGrad+ achieves lower forget accuracy on CIFAR-10 when forgetting one class, albeit with a slight drop in retain accuracy. Nevertheless, as shown in Table 1, such cases are rare, and Retain-only Retrain typically achieves lower feature-level forget accuracies overall. In Section 4.3, we will introduce new MU methods that can remove information from hidden representations and consistently achieve lower feature-level forget accuracies.

4.2 Feature-classifier Misalignment by $\mathcal{NC}$ Analysis

In the previous section, we have shown that current unlearning methods allow us to obtain zero forget accuracy while maintaining comparable accuracy on the retain set. However, performance on the forget set can be recovered with simple linear probing. What is the mechanism of unlearning that explains this observation? In this subsection we explain this illusion of unlearning through the lens of Neural Collapse ($\mathcal{NC}$).

In collapsed models, the last layer features of samples within the same class are concentrated around their class means ($\mathcal{NC}_1$), the class means form a simplex ETF ($\mathcal{NC}_2$), the classifier weights align with the class means ($\mathcal{NC}_3$), and the NCC rule at the last layer agrees with the decision of the deep network ($\mathcal{NC}_4$).

Observation 2: The illusion of unlearning is primarily caused by feature-classifier misalignment. The NCC accuracy ($\mathcal{NC}_4$) reported in Table 1 shows that the NCC classifier also achieves high forget accuracy across many unlearning methods. Since NCC classification depends only on distances between features and class means, this indicates that the feature representations of the forgotten classes remain clustered around their class means even after unlearning. In other words, the discriminative structure of the feature space is largely preserved. We next explore how models that have their last layer features clustered around their class means still have near-zero forget accuracy. To measure this, we calculated the feature-weight alignment between the last layer features and the classifier weights ($\mathcal{NC}_3$). As shown in Figure 5, the alignment for the retain classes is largely preserved after unlearning, whereas the classifier weight corresponding to the forget class becomes significantly misaligned with its class mean, which we term as *feature-classifier misalignment*. This shows that the model achieves zero forget accuracy by only shifting the last layer weights in an appropriate manner.

In fact, under the assumptions of $\mathcal{NC}$ and fixed class means, we can show that the optimal configuration of last layer weights after unlearning using NegGrad flips the forget classifier vector to be maximally misaligned with the forget class mean while minimally shifting the weights of the retain classifier vectors (a similar argument holds for Random label unlearning as well). We prove the following proposition in Appendix C.

Proposition 1. *Let $f_{\boldsymbol{W},\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}\phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ be a classification model trained to collapse with last layer class mean features $\{\boldsymbol{\mu}_k\}_{k=1}^K$ that form a simplex equiangular tight frame, and assume that the mean features do not change during unlearning. If class $k \in [K]$ is unlearned using the NegGrad objective, then the resulting weights satisfy $\boldsymbol{w}_k^{un} = -(1-\gamma)\boldsymbol{\mu}_k$ for the forget class k, and $\boldsymbol{w}_i^{un} \propto \alpha\boldsymbol{\mu}_i + \beta\boldsymbol{\mu}_k$ for the retain classes $i \neq k$ where $0 < \alpha, \beta, \gamma < 1$.*

We depict this optimal configuration with the misaligned classifier vector in Figure 2. This configuration of last layer weights also achieves zero output-level forget accuracy.

Corollary 1. *Consider the same setting as proposition 1. Let $\hat{y}(\boldsymbol{x}) = \arg\max_{k\in[K]} \cos(\angle \boldsymbol{w}_k^{un}, \phi_{\boldsymbol{\theta}}(\boldsymbol{x}))$ be the prediction of the model where class k has been unlearned. This model achieves zero forget accuracy, i.e., $\hat{y}(\boldsymbol{x}_{i,k}) \neq k$ for training samples $\boldsymbol{x}_{i,k}$ that belong to class k.*

Proof. Recall that in the $\mathcal{NC}$ setting the last layer features are mapped to the fixed class means $\{\boldsymbol{\mu}_c\}_{c=1}^K$, and the class means form a simplex ETF, i.e. $\boldsymbol{M}^\top\boldsymbol{M} = \left(\boldsymbol{I}_K - \frac{1}{K}\boldsymbol{1}_K\boldsymbol{1}_K^\top\right)$ where $\boldsymbol{M}$ is a matrix with the columns set to the class means. This means that training samples in class c are mapped to $\boldsymbol{\mu}_c, \forall c \in [K]$. For samples in the forget class k we have

$$\boldsymbol{w}_i^{\mathrm{un}\top}\boldsymbol{\mu}_k = (\alpha\boldsymbol{\mu}_i + \beta\boldsymbol{\mu}_k)^\top\boldsymbol{\mu}_k = \frac{-\alpha}{K-1} + \beta$$

$$\implies \cos(\angle \boldsymbol{w}_c^{\mathrm{un}}, \boldsymbol{\mu}_k) = \frac{\frac{-\alpha}{K-1} + \beta}{\sqrt{\alpha^2 + \beta^2 - \frac{2\alpha\beta}{K-1}}} > -1, \forall c \neq k$$

$$\boldsymbol{w}_k^{\mathrm{un}\top}\boldsymbol{\mu}_k = -(1-\gamma)\boldsymbol{\mu}_k^\top\boldsymbol{\mu}_k = \gamma - 1$$

$$\implies \cos(\angle \boldsymbol{w}_k^{\mathrm{un}}, \boldsymbol{\mu}_k) = -1$$

This means that $\arg\max_{c\in[K]} \cos(\angle \boldsymbol{w}_c^{\mathrm{un}}, \boldsymbol{\mu}_k) \neq k$, and hence the model achieves zero forget accuracy. □

Table 2: Full model VS classifier-only unlearning evaluated via mean output-level forget and retain accuracies.

Method	Layers Finetuned	CIFAR-10				CIFAR-100			
		1		3		1		10	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Full Model	93.98	93.98	94.00	93.94	74.61	74.40	74.47	75.88
Retain-only Retrain	Full Model	94.74	0.00	95.37	0.00	76.01	0.00	76.50	0.00
Retain-only FT	Full Model	94.26	47.67	95.24	52.48	74.08	53.20	74.53	64.96
	Classifier only	93.20	0.00	93.66	0.00	73.29	0.00	73.51	0.00
NegGrad+	Full Model	92.85	0.00	93.29	0.01	69.90	0.00	70.80	0.28
	Classifier only	93.08	0.00	93.72	0.00	73.28	0.00	66.85	0.00
Random-label	Full Model	92.93	0.00	94.14	0.00	72.33	0.00	72.19	0.00
	Classifier only	93.39	0.00	94.56	0.00	73.87	0.20	74.09	2.32
Salun	Full Model	93.19	0.00	94.43	0.00	72.96	0.00	72.92	0.06
	Classifier only	93.38	0.00	94.44	0.00	73.93	1.00	73.98	4.18
SVD	Full Model	92.02	0.00	94.05	57.43	71.09	0.00	73.10	55.56
	Classifier only	93.53	0.01	93.73	68.45	73.47	2.60	74.13	50.56
SCRUB	Full Model	91.37	0.00	93.61	0.00	73.67	0.20	74.56	0.32
	Classifier only	93.12	0.00	93.68	0.00	74.01	5.00	74.01	0.00
UNSIR	Full Model	91.84	0.48	92.87	0.01	73.77	3.80	73.58	14.58
	Classifier only	94.48	0.00	94.46	1.39	74.64	0.00	74.27	0.52

In order to confirm our hypothesis that last layer unlearning is sufficient to get zero forget set accuracy, we run experiments where we update only the last layer weights during unlearning and show the results in Table 2. This leads us to our next observation.

Observation 3: Classifier-only unlearning achieves comparable performance at the output level. As shown in Table 2, updating only the classifier during unlearning achieves performance that is comparable to, or even slightly better than, full-model unlearning in terms of both forgetting and retaining accuracy across different MU methods for most scenarios. However, although models appear to forget at the output level, the feature mappings remain unchanged from the original model and continue to encode information about the forgotten classes. This finding underscores that current MU methods primarily achieve output suppression rather than representation erasure, challenging the validity of evaluating unlearning effectiveness solely through output-level metrics.

4.3 Class-Mean-Features Unlearning

To enable unlearning in the feature space, we propose representation-level unlearning methods that employ class-mean features (CMF) classifiers to address the classifier–feature misalignment described above. Specifically, inspired by the self-duality between features and classifiers, the CMF classifier was originally proposed in (Jiang et al., 2024) to reduce trainable parameters by setting classifier weights to the exponential moving average of the mini-batch class-mean features during training. In our work, we adapt CMF classifiers to the unlearning setting, leveraging them to enforce alignment between classifiers and features throughout the unlearning process.

Formally, we construct the CMF classifier via

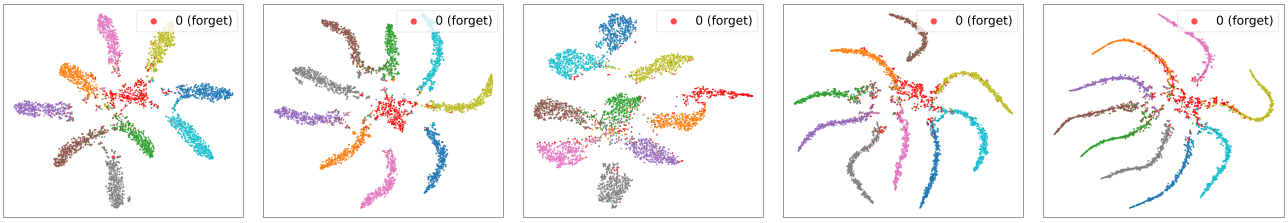
$$\boldsymbol{W}_{\text{CMF}} = \begin{bmatrix}\boldsymbol{\mu}_1 & \cdots & \boldsymbol{\mu}_K\end{bmatrix}^\top \in \mathbb{R}^{K\times d}, \tag{7}$$

where $\boldsymbol{\mu}_c$ denotes the mean feature vector for class c as defined in (3) and can be updated at each epoch during training. The CMF classifier can be seamlessly integrated into existing MU methods by replacing $\boldsymbol{W}$ with $\boldsymbol{W}_{\text{CMF}}$ in (1) and plugging the resulting model into the general MU objective in (6).

We apply the CMF classifier to multiple representative MU methods, including Random Label, SalUn, NegGrad+, SCRUB, and UNSIR, with mean results summarized in Table 3. By explicitly enforcing alignment between classifiers and features, the unlearning process becomes more challenging: CMF-based methods no longer trivially achieve zero forget accuracy at the output level. Nevertheless, model features now encode substantially less information about the forgotten classes, as indicated by the much lower feature-level forget accuracy under linear probing. Remarkably, the proposed CMF-enhanced unlearning methods consistently achieve significantly lower feature-level forget accuracy than the Retain-only Retrain baseline, while incurring only a very mild decrease in retain accuracy. This result highlights the strength of CMF in mitigating feature–classifier misalignment, ensuring that for-

Table 3: Evaluation of MU methods with CMF classifiers for unlearning certain number of classes.

Method	Accuracy	CIFAR-10				CIFAR-100				Tiny-ImageNet			
		1		3		1		10		1		20	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Output	93.98	93.98	94.00	93.94	74.61	74.40	74.47	75.88	65.27	58.80	65.15	66.02
	Linear Probe	94.02	94.02	94.03	94.00	74.53	75.00	74.38	75.90	65.10	60.80	64.97	66.08
	NCC	94.00	93.99	94.03	93.92	74.40	75.00	74.28	75.68	64.65	60.00	64.56	65.00
Retain-only Retrain	Output	94.74	0.00	95.37	0.00	76.01	0.00	76.50	0.00	66.52	0.00	66.38	0.00
	Linear Probe	90.49	77.35	85.64	67.33	74.09	85.20	69.34	60.94	65.90	46.40	65.21	30.36
	NCC	93.31	47.06	91.37	37.07	73.90	70.40	70.98	43.18	63.57	71.20	59.37	44.30
Random-label with CMF	Output	94.27	80.70	94.81	75.69	74.38	55.60	74.85	54.40	62.01	22.40	62.25	22.20
	Linear Probe	94.19	85.71	94.49	81.47	74.63	59.20	74.98	66.44	62.56	32.80	62.38	36.58
	NCC	94.25	82.04	94.73	76.74	74.49	59.20	74.82	60.24	61.96	27.20	61.93	28.24
Salun with CMF	Output	94.33	78.07	95.01	75.96	74.62	60.40	74.98	62.10	62.62	30.00	63.14	32.74
	Linear Probe	94.26	84.68	94.60	83.39	74.79	59.40	75.18	67.20	63.18	41.20	63.12	46.14
	NCC	94.32	79.50	94.91	77.31	74.78	63.60	75.03	65.20	62.67	37.20	62.77	38.10
NegGrad+ with CMF	Output	91.87	54.50	94.97	60.00	71.82	41.00	71.45	30.38	61.18	22.00	60.82	41.42
	Linear Probe	92.35	68.02	94.60	75.29	72.86	55.20	72.35	50.62	62.90	41.20	62.66	53.82
	NCC	91.99	57.83	94.93	62.89	72.36	51.20	71.75	41.54	62.31	38.80	62.11	51.30
Scrub with CMF	Output	92.51	33.78	95.37	35.11	73.86	40.60	74.27	35.02	61.64	27.20	63.31	47.76
	Linear Probe	92.48	60.68	95.26	62.77	74.03	55.60	74.18	58.34	62.13	36.80	63.76	54.28
	NCC	92.48	35.53	95.34	40.04	73.87	47.00	74.12	42.82	61.66	34.40	63.28	50.42
UNSIR with CMF	Output	91.79	12.91	93.56	11.51	72.72	21.00	72.61	9.16	60.81	14.00	61.34	14.44
	Linear Probe	91.63	31.16	93.01	28.65	72.91	35.20	72.51	20.98	60.96	26.80	61.27	23.02
	NCC	91.87	9.29	93.79	8.16	72.67	20.20	72.48	9.94	60.63	26.00	60.80	16.72



(a) Random-label (b) SalUn (c) NegGrad+ (d) SCRUB (e) UNSIR

Figure 6: t-SNE of features learned with CMF-based unlearning methods. The forgotten class (red points) exhibits a distribution that is markedly different from the retain classes and shows noticeable overlap with them. Here the overall feature distributions are reshaped due to an additional normalization step.

getting occurs not just in predictions but also within the hidden representations. We can qualitatively observe this in the t-SNE plots of Figure 6. Overall, our findings underscore that CMF provides a principled and effective framework for representation-level unlearning, offering a more faithful approach to removing information from deep models compared to existing baselines.

5 CONCLUSION AND FUTURE WORK

In this paper we describe an *illusion of unlearning* where models appear to forget classes when evaluated at the output level, while still retaining information about the forgotten data in their hidden representations. We demonstrate that training linear probes on features from unlearned models can recover performance on the forget set. Through an $\mathcal{NC}$ analysis, we observe that the unlearning methods mainly alter the final classifier weights to be misaligned to the forget classes, while maintaining the representations of the forget classes in layers below the last layer. To mitigate this issue, we propose *class-mean-features unlearning*, which ties classifier weights to class-mean features and encourages the removal of forgotten information from the representation space.

There are several promising directions for future work. First is the extension to generative diffusion and language models where our shallow unlearning observations are aligned with recent findings such as the fact that simple fine-tuning can inadvertently reintroduce erased concepts (Suriyakumar et al., 2024). Next, the transferability of features in deep learning poses challenges for unlearning. We would like to characterize the trade-off between removing feature-level information about the forget data while maintaining performance on the retained data. Finally, neural collapse phenomena have also been observed in intermediate layers (Rangamani et al., 2023), and future work may investigate whether extending CMF-style constraints to deeper layers can further improve representation-level unlearning.

Acknowledgement

YG and ZZ acknowledge support from NSF grants IIS-2312840 and IIS-2402952. We gratefully acknowledge Jinxin Zhou and Huminhao Zhu for valuable discussions.

References

Ali, Z., Muhammad, A., Adnan, R., Alkhalifah, T., and Aslam, S. (2025). Evaluating machine unlearning: Applications, approaches, and accuracy. *Engineering Reports*, 7(1):e13081.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2021). Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE.

Choi, D. and Na, D. (2023). Towards machine unlearning benchmarks: Forgetting the personal identities in facial recognition systems. *arXiv preprint arXiv:2311.02240*.

Fabbrizzi, S., Papadopoulos, S., Ntoutsi, E., and Kompatsiaris, I. (2022). A survey on bias in visual datasets. *Computer Vision and Image Understanding*, 223:103552.

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

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

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 19–35. IEEE.

Jeon, D., Jeung, W., Kim, T., No, A., and Choi, J. (2024). An information theoretic evaluation metric for strong unlearning. *arXiv preprint arXiv:2405.17878*.

Jiang, J., Zhou, J., Wang, P., Qu, Q., Mixon, D. G., You, C., and Zhu, Z. (2024). Generalized neural collapse for a large number of classes. In *Proceedings of the 41st International Conference on Machine Learning*, pages 22010–22041.

Jin, R., Chen, M., Zhang, Q., and Li, X. (2023). Forgettable federated linear learning with certified data unlearning. *arXiv preprint arXiv:2306.02216*.

Kim, Y., Cha, S., and Kim, D. (2025). Are we truly forgetting? a critical re-examination of machine unlearning evaluation protocols. *arXiv preprint arXiv:2503.06991*.

Kodge, S., Saha, G., and Roy, K. (2024). Deep unlearning: Fast and efficient gradient-free class forgetting. *Transactions on Machine Learning Research*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. (2023). Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987.

Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

Li, G., Hsu, H., Chen, C.-F., and Marculescu, R. (2024). Machine unlearning for image-to-image generative models. *arXiv preprint arXiv:2402.00351*.

Lu, S., Wang, Z., Li, L., Liu, Y., and Kong, A. W.-K. (2024). Mace: Mass concept erasure in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6430–6440.

Papyan, V., Han, X., and Donoho, D. L. (2020). Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663.

Rangamani, A., Lindegaard, M., Galanti, T., and Poggio, T. A. (2023). Feature learning in deep classifiers through intermediate neural collapse. In *International conference on machine learning*, pages 28729–28745. PMLR.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Suriyakumar, V. M., Alur, R., Sekhari, A., Raghavan, M., and Wilson, A. C. (2024). Unstable unlearning: The hidden risk of concept resurgence in diffusion models. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

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

Thudi, A., Jia, H., Shumailov, I., and Papernot, N. (2022). On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4007–4022, Boston, MA. USENIX Association.

Voigt, P. and Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555.

Xu, H., Zhu, T., Zhou, W., and Zhao, W. (2024). Don't forget too much: Towards machine unlearning on feature level. *IEEE Transactions on Dependable and Secure Computing*.

Xue, L., Hu, S., Lu, W., Shen, Y., Li, D., Guo, P., Zhou, Z., Li, M., Zhang, Y., and Zhang, L. Y. (2025). Towards reliable forgetting: A survey on machine unlearning verification, challenges, and future directions. *arXiv preprint arXiv:2506.15115*.

Yu, Z., Idris, M. Y. I., and Wang, P. (2025). Forgetme: Evaluating selective forgetting in generative models. *arXiv preprint arXiv:2504.12574*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   See Sections 2.1 and 4.3 for the main description. Algorithm 1, Algorithm 2 in Appendix B provide detailed pseudocode for the CMF unlearning strategies.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   See Appendix B.2 for a discussion of the time, space, and sample complexity of CMF-based unlearning.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
   An anonymized version of the source code with all dependencies (e.g., PyTorch, PyTorch Lightning, Torchmetrics, NumPy) will be released upon acceptance.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   See Section 2.1 for assumptions on Neural Collapse and class mean features, and Section 4.3 for assumptions underlying the unlearning analysis.
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes] The datasets that we use are public.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

An Illusion of Unlearning? Assessing Machine Unlearning Through Internal Representations: Appendix

A Experimental Settings

A.1 Experiment Environment

All experiments are conducted on a Linux server running Ubuntu 20.04 (kernel 5.4), equipped with 8 NVIDIA RTX A5000 GPUs (24 GB memory each), 64 CPU cores, and 252 GB RAM. The software environment uses Python 3.11, PyTorch 2.5.1, and CUDA 12.1.

A.2 Datasets

We evaluate class-unlearning on three standard image classification benchmarks: CIFAR-10, CIFAR-100, and Tiny-ImageNet. CIFAR-10 contains 10 classes with 50,000 training images and 10,000 test images. CIFAR-100 has the same number of images but with 100 classes. Tiny-ImageNet contains 200 classes with 500 training images and 50 validation images per class.

For original model training, we use train/validation/test splits for model selection and evaluation. For unlearning experiments, we use the standard train/test split and report performance on the test set.

A.3 Unlearning Scenarios

We evaluate unlearning algorithms under both*single-class* and *multi-class* forgetting scenarios on each dataset. For every dataset and scenario, we construct multiple retain–forget dataset combinations by selecting different class indices as the forget set. Each unlearning algorithm is evaluated on 5–10 such combinations, depending on the dataset and setting. For each experiment group, we report the mean and standard deviation of the evaluation metrics across all combinations.

CIFAR-10. For CIFAR-10, we evaluate both single-class and three-class forgetting. In the single-class setting, we sweep across all classes ($\{0\}, \{1\}, \ldots, \{9\}$). In the three-class setting, we evaluate several representative class combinations such as $\{0, 1, 2\}$, $\{3, 4, 5\}$, $\{6, 7, 8\}$, $\{0, 5, 9\}$, and $\{2, 4, 8\}$.

CIFAR-100. CIFAR-100 consists of 100 fine-grained classes grouped into 20 coarse classes, each containing 5 fine classes. For single-class unlearning, we evaluate several representative classes $\{0\}, \{1\}, \{2\}, \{3\}, \{5\}$. These classes are selected to cover different coarse classes. Notably, classes 1 and 4 belong to the same coarse class, and therefore we evaluate only one of them to avoid redundant experiments within the same superclass.

For multi-class unlearning, we remove 10 classes at a time. Each such setting corresponds to the union of two coarse classes (i.e., 2×5 fine classes). We evaluate multiple such combinations to cover different regions of the label space. The specific class sets used in our experiments include: $\{3, 15, 19, 21, 31, 38, 42, 43, 88, 97\}$, $\{47, 52, 54, 56, 59, 62, 70, 82, 92, 96\}$, $\{5, 20, 22, 25, 39, 40, 84, 86, 87, 94\}$, $\{8, 13, 41, 48, 59, 69, 81, 85, 89, 90\}$, and $\{1, 4, 30, 32, 55, 67, 72, 73, 91, 95\}$.

Tiny-ImageNet. Tiny-ImageNet contains 200 classes. We evaluate both single-class forgetting and larger group unlearning settings.

For single-class forgetting, we evaluate several representative classes, including $\{2\}, \{3\}, \{5\}, \{7\}, \{9\}$.

For multi-class unlearning, we remove groups of 20 classes at a time. The class groups are constructed as contiguous ranges of class indices, including $\{0, \ldots, 19\}$, $\{20, \ldots, 39\}$, $\{40, \ldots, 59\}$, $\{60, \ldots, 79\}$, and $\{80, \ldots, 99\}$. These groups cover different regions of the label space.

A.4 Original Model Training

We use ResNet and ViT models in our experiments. For the ResNet experiments, we train all models from scratch. Specifically, we use ResNet-18 on CIFAR-10 and CIFAR-100, and ResNet-50 on Tiny-ImageNet. For

the ViT experiments, we use ViT-S/16 models initialized from ImageNet-pretrained weights and then fine-tune them on each target dataset, including CIFAR-10, CIFAR-100, and Tiny-ImageNet.

During training, we apply standard data augmentation, including random cropping and random horizontal flipping.

For ViT models, input images are resized to 224×224 resolution.

For ResNet training, we use batch size 128 and train for up to 300 epochs with early stopping (patience 50). For optimization, we use SGD with momentum 0.9 and weight decay 5×10^{-4}. The initial learning rate is set to 5×10^{-2}. We apply a learning-rate warmup for the first 5 epochs followed by cosine learning-rate decay with a minimum learning rate of 1×10^{-5}.

For ViT experiments, we use ViT-S/16 models initialized from ImageNet-pretrained weights and fine-tune them on each target dataset. The models are fine-tuned for 10 epochs with batch size 128 and learning rate 3×10^{-4}. We use the same optimizer configuration as the ResNet training.

The resulting full-data model serves as the *original model*, which is used as the starting point for all subsequent unlearning algorithms.

A.5 Retrain-on-Retain Baseline (Gold Standard)

To establish a reference for unlearning methods, we retrain models from scratch using only the retain subset. This baseline is considered the "gold standard" for machine unlearning.

The retrain models follow the same training settings described in Appendix A.4. For each unlearning scenario, we construct the retain dataset according to the corresponding retain–forget split defined in Appendix A.3, and train separate retrain models as references for comparison with unlearning methods. These models are reported as *Retain-only Retrain* in the experimental results.

A.6 Unlearning Methods

We evaluate several representative unlearning algorithms:

- **Random Label (RL) (Golatkar et al., 2020)** Forget-class samples are reassigned random labels from the retain set.
- **SalUn (Fan et al., 2023)** SalUn perturbs important model parameters associated with the forget classes based on saliency scores.
- **NegGrad+ (Grad-Ascent-Descent) (Choi and Na, 2023)** NegGrad+ adjusting the model's output on forget data by performing gradient ascent on forget samples and gradient descent on retain samples.
- **SCRUB (Kurmanji et al., 2023)** SCRUB formulates unlearning as a teacher–student distillation problem. The original model acts as a teacher and a student model is trained to match the teacher on retain data while diverging from the teacher on forget data.
- **UNSIR (Tarun et al., 2023)** UNSIR performs unlearning through an impair–repair process. First, an error-maximizing noise matrix is generated to maximize the loss for the target forget classes. The model is then updated using this noise together with a subset of retain data, followed by additional training on the retain data only to recover the model's performance.
- **SVD (Training-Free) (Kodge et al., 2024)** A training-free method that removesforget-class information by performing singular value decomposition (SVD) on class-specific feature activations to estimate retain and forget subspaces, and suppressing the forget-discriminative components in the model parameters.

The detailed training hyperparameters for all methods are summarized in Table 4 and Table 5.

A.7 Summary of Hyperparameters

Tables 4 and 5 summarize the hyperparameters across unlearning methods on ResNet and ViT models.

A.8 Additional Results with Standard Deviations

Tables 6–10 report the mean accuracy together with the standard deviation over multiple runs for all evaluated methods.

B CMF Unlearning Algorithms

B.1 CMF-based Unlearning Framework

In this appendix, we provide pseudocode for the core components of our CMF-based unlearning framework. Our goal is to make the CMF head reconstruction compatible with a wide range of existing machine unlearning methods.

The key idea is to decouple the classifier head construction from the underlying unlearning objective. Instead of updating the classifier weights through standard gradient training, we reconstruct the classifier head directly from class mean features at the beginning of each epoch. The head is then kept frozen while the encoder is updated according to the objective of the chosen unlearning method.

Algorithm 1 describes the CMF head reconstruction procedure. Given an encoder and a dataset, we compute the feature mean for each class, center the class means, and normalize them to obtain the classifier weights. The resulting CMF head captures the geometric structure of the feature space and is fixed during the subsequent optimization steps.

Algorithm 2 illustrates how the reconstructed CMF head can be integrated into a generic gradient based unlearning pipeline. At the beginning of each epoch, the CMF classifier is rebuilt from the current encoder features.

Algorithm 1 CMF Head Reconstruction

Require: Encoder $z_\theta(\cdot)$, dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with class set K
for each class $k \in K$ **do**
  Compute class-mean feature $\boldsymbol{\mu}_k = \frac{1}{|D_k|} \sum_{(\boldsymbol{x}_i, y_i) \in D_k} z_\theta(\boldsymbol{x}_i)$
end for
Compute global mean $\bar{\boldsymbol{\mu}} = \frac{1}{K} \sum_{k \in K} \boldsymbol{\mu}_k$
for each class $k \in K$ **do**
  Center and normalize $\boldsymbol{w}_k = \frac{\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}}}{\|\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}}\|}$
end for
Form classifier head $\boldsymbol{W}_{\text{CMF}} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$
Freeze $\boldsymbol{W}_{\text{CMF}}$
return $\boldsymbol{W}_{\text{CMF}}$

Algorithm 2 Gradient-Based Unlearning with CMF

Require: Encoder $z_\theta(\cdot)$, dataset $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, epochs E, unlearning objective $\mathcal{L}_U$
Reconstruct CMF head $\boldsymbol{W}_{\text{CMF}} \leftarrow$ CMF Head Reconstruction(z_θ, D)
for $e = 1$ **to** E **do**
  for mini-batch $\mathcal{B} = \{(\boldsymbol{x}_i, y_i)\}$ **do**
    Compute unlearning loss $\mathcal{L} = \mathcal{L}_U(\boldsymbol{W}_{\text{CMF}} z_\theta(\boldsymbol{x}_i), y_i)$
    Update encoder parameters θ
  end for
  Reconstruct CMF head $\boldsymbol{W}_{\text{CMF}} \leftarrow$ CMF Head Reconstruction(z_θ, D)
end for
return encoder z_θ and $\boldsymbol{W}_{\text{CMF}}$

The loss $\mathcal{L}_U$ in Algorithm 2 corresponds to the objective used by the underlying unlearning method. For example, $\mathcal{L}_U$ may correspond to the random-label cross-entropy loss in Random Label, or the ascent–descent objective used in gradient-based unlearning methods such as NegGrad+. Therefore, CMF head reconstruction can serve as a modular component that augments a wide range of existing unlearning approaches.

B.2 Complexity Analysis

The CMF head reconstruction described in Algorithm 1 introduces only a small computational overhead compared to standard stochastic gradient descent (SGD) training.

At the beginning of each epoch, the class-mean features are computed by a forward pass of the encoder over all samples in the dataset D. Let T_f denote the cost of one forward pass of the encoder. Computing the class

means therefore requires $O(|D|T_f)$ time.

After obtaining the class means, reconstructing the CMF classifier head requires centering and normalizing the mean vectors, which costs $O(Cd)$ where C is the number of classes and d is the feature dimension.

The additional memory overhead is $O(Cd)$ for storing the class-mean vectors, which is negligible compared to the parameters of the encoder.

Therefore, the overall computational cost per epoch consists of the standard training cost plus one small additive cost from CMF head reconstruction. Since the primary cost during training derives from the forward/backward propagation of the encoder, the overhead introduced by CMF reconstruction is small in practice.

C Analysis of Last layer Unlearning

In section 4.2 we measure the Neural Collapse (NC) metrics for networks that have been unlearned and observe that while the classes in the retain and forget sets remain separable, the distance between the classifier and the class mean features increases. This, combined with our Linear Probing results suggests that class unlearning in deep networks is primarily achieved by changing the alignment of the classifier without changing the features. To analyze how the classifier changes during unlearning, we derive the minima of the Neg-Grad loss under the assumption that the original model was trained to collapse, and that the class mean features do not move.

Proposition 2. *Let $f_{\boldsymbol{W},\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}\phi_{\boldsymbol{\theta}}(\boldsymbol{x})$ be a classification model trained to collapse with last layer class mean features $\{\boldsymbol{\mu}_k\}_{k=1}^K$ that form a simplex equiangular tight frame, and assume that the mean features do not change during unlearning. If class $k \in [K]$ is unlearned using the Neg-grad objective, then the resulting weights satisfy $\boldsymbol{w}_k^{un} = -(1-\gamma)\boldsymbol{\mu}_k$ for the forget class k, and $\boldsymbol{w}_i^{un} \propto \alpha\boldsymbol{\mu}_i + \beta\boldsymbol{\mu}_k$ for the retain classes $i \neq k$ where $0 < \alpha, \beta, \gamma < 1$.*

Proof. The regularized Neg-grad objective is given by:

$$\begin{aligned}\mathcal{L} = \frac{1}{K-1}\sum_{i\neq k}\log\left(\sum_{j=1}^{K}\exp(\boldsymbol{w}_j^\top\boldsymbol{\mu}_i)\right) - \boldsymbol{w}_i^\top\boldsymbol{\mu}_i \\ + \boldsymbol{w}_k^\top\boldsymbol{\mu}_k - \log\left(\sum_{j=1}^{K}\exp(\boldsymbol{w}_j^\top\boldsymbol{\mu}_k)\right) + \frac{\lambda_W}{2}\|\boldsymbol{W}\|_F^2\end{aligned} \tag{8}$$

Consider the gradients of the objective $\mathcal{L}$ wrt the weights of the forget and retain classes:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_j} = \frac{1}{K-1}\sum_{i\neq k}\frac{\exp(\boldsymbol{w}_j^T\boldsymbol{\mu}_i)}{\Lambda_i}\boldsymbol{\mu}_i - \frac{\boldsymbol{\mu}_j}{K-1} - \frac{\exp(\boldsymbol{w}_j^T\boldsymbol{\mu}_k)}{\Lambda_k}\boldsymbol{\mu}_k + \lambda_W\boldsymbol{w}_j$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_k} = \frac{1}{K-1}\sum_{i\neq k}\frac{\exp(\boldsymbol{w}_k^T\boldsymbol{\mu}_i)}{\Lambda_i}\boldsymbol{\mu}_i + \boldsymbol{\mu}_k - \frac{\exp(\boldsymbol{w}_k^T\boldsymbol{\mu}_k)}{\Lambda_k}\boldsymbol{\mu}_k + \lambda_W\boldsymbol{w}_k$$

Here Λ_i denotes the logsumexp of the classifier scores for mean feature $\boldsymbol{\mu}_i$. From Lemma 1 we can observe that at stationary points of the objective $\mathcal{L}$ for all $j \neq k$ and $l \neq j, k$ we have that $\boldsymbol{w}_j^\top\boldsymbol{\mu}_l := b$ are all equal, $\boldsymbol{w}_j^\top\boldsymbol{\mu}_j := a$ are equal, and $\boldsymbol{w}_k^\top\boldsymbol{\mu}_j := c$ are equal. Moreover, we also have that $\boldsymbol{w}_j^\top\boldsymbol{\mu}_k := b'$ are equal for $j \neq k$ and $\boldsymbol{w}_k^\top\boldsymbol{\mu}_k = c$. This means that $\Lambda_k = \exp(c) + (K-1)\exp(b')$ and $\Lambda_i = \exp(a) + (K-2)\exp(b) + \exp(c) = \Lambda$ are equal for all $i \neq k$. Plugging this into the above gradient expressions and computing the stationary points, we obtain for $j \neq k$:

$$\begin{aligned}\lambda_W\boldsymbol{w}_j^{\text{un}} &= \frac{\boldsymbol{\mu}_j}{K-1} - \frac{1}{K-1}\left[\frac{\exp(b)}{\Lambda}\sum_{i\neq j,k}\boldsymbol{\mu}_i + \frac{\exp(a)}{\Lambda}\boldsymbol{\mu}_j\right] + \frac{\exp(b')}{\Lambda_k}\boldsymbol{\mu}_k \\ &= \frac{1}{K-1}\left(1 - \frac{\exp(b)-\exp(a)}{\Lambda}\right)\boldsymbol{\mu}_j + \left(\frac{\exp(b')}{\Lambda_k} + \frac{\exp(b)}{(K-1)\Lambda}\right)\boldsymbol{\mu}_k\end{aligned} \tag{9}$$

Where we have used the simplex ETF condition to obtain $\sum_{i\neq j,k}\boldsymbol{\mu}_i = -\boldsymbol{\mu}_j - \boldsymbol{\mu}_k$.

For the forget class k we have:

$$\begin{aligned}\lambda_W \boldsymbol{w}_k^{\text{un}} &= -\boldsymbol{\mu}_k + \frac{\exp(\boldsymbol{w}_k^T \boldsymbol{\mu}_k)}{\Lambda_k}\boldsymbol{\mu}_k - \frac{1}{K-1}\sum_{i\neq k}\frac{\exp(\boldsymbol{w}_k^T \boldsymbol{\mu}_i)}{\Lambda_i}\boldsymbol{\mu}_i \\ &= -\boldsymbol{\mu}_k + \frac{\exp(c)}{\Lambda_k}\boldsymbol{\mu}_k - \frac{1}{K-1}\frac{\exp(c)}{\Lambda}\sum_{i\neq k}\boldsymbol{\mu}_i \\ &= -\left(1 - \frac{\exp(c)}{\Lambda_k} - \frac{1}{K-1}\frac{\exp(c)}{\Lambda}\right)\boldsymbol{\mu}_k\end{aligned} \tag{10}$$

Since the two factors are < 1, we have that $\boldsymbol{w}_k^{\text{un}} = -(1-\gamma)\boldsymbol{\mu}_k$ for $\gamma < 1$

□

Lemma 1. *Let $\boldsymbol{z}, \boldsymbol{z'} \in \mathbb{R}^K$ be any two real vectors, and $\boldsymbol{y}_i, \boldsymbol{y}_k$ be two one-hot vectors corresponding to different classes. Consider the following constrained optimization problem:*

$$\min_{\boldsymbol{z},\boldsymbol{z'}} \mathcal{L}_{CE}(\boldsymbol{z}, \boldsymbol{y}_i) - \mathcal{L}_{CE}(\boldsymbol{z'}, \boldsymbol{y}_k) \quad s.t. \|\boldsymbol{z}\|_2^2 \leq 1, \|\boldsymbol{z'}\|_2^2 \leq 1, z_k = z'_k$$

The KKT points of this objective satisfy the following conditions $z_j = z_l, \forall j, l \neq k, i$ and $z'_j = z'_l, \forall j, l \neq k$.

Proof. The lagrangian for our problem is:

$$\begin{aligned}\mathcal{L} = \log\left(\sum_{j=1}^K \exp(z_j)\right) - z_i + z'_k - \log\left(\sum_{j=1}^K \exp(z'_j)\right) \\ + \lambda_1(\|\boldsymbol{z}\|_2^2 - 1) + \lambda_2(\|\boldsymbol{z'}\|_2^2 - 1) + \lambda_3(z_k - z'_k)\end{aligned}$$

at stationary points of the Lagrangian, we have for entries of $\boldsymbol{z}$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_j} &= \frac{\exp z_j}{\sum_{l=1}^K \exp(z_l)} + \lambda_1 z_j = 0 \quad j \neq i, k \\ \frac{\partial \mathcal{L}}{\partial z_i} &= \frac{\exp z_i}{\sum_{l=1}^K \exp(z_l)} - 1 + \lambda_1 z_i = 0 \\ \frac{\partial \mathcal{L}}{\partial z_k} &= \frac{\exp z_k}{\sum_{l=1}^K \exp(z_l)} + (\lambda_1 - \lambda_3) z_k = 0\end{aligned}$$

From the conditions on $z_j, j \neq k, i$ we obtain the following equation: $\frac{\exp z_j}{-\lambda_1 \sum_{l=1}^K \exp(z_l)} = z_j$. Since the equation $c\exp(x) = x$ has only one solution in $x \in \mathbb{R}$, we get the condition that z_j are all equal for $j \neq k, i$. The stationarity conditions for z_i, z_k are different, and hence those values will be different.

Using a similar argument for the stationarity conditions on $z'_j, j \neq k$ we show that z'_j are all equal for $j \neq k$ □

D Tables on Experiments and Results

D.1 Hyperparameter Settings

Table 4: Hyperparameters for ResNet-based experiments across CIFAR-10, CIFAR-100, and Tiny-ImageNet. SVD is training-free: no encoder updates; the batch size correspond only to data samples drawn separately from the retain and forget datasets.

Dataset	Model	Method	Epochs	Batch	LR	Mom.	Other Key Flags
CIFAR-10	ResNet18	Original	300	128	0.01	0.9	cosine LR; WD=5×10^{-4}
CIFAR-100	ResNet18	Original	300	128	0.01	0.9	cosine LR; WD=5×10^{-4}
Tiny-ImageNet	ResNet50	Original	300	128	0.05	0.9	cosine LR; WD=5×10^{-4}
CIFAR-10	ResNet18	Retain-only Retrain	200	128	0.01	0.9	WD=5×10^{-4}; val-ratio=0.1
CIFAR-100	ResNet18	Retain-only Retrain	200	128	0.01	0.9	WD=5×10^{-4}
Tiny-ImageNet	ResNet50	Retain-only Retrain	150	256	0.05	0.9	WD=5×10^{-4}
CIFAR-10	ResNet18	Retain-only FT	3	128	1×10^{-3}	0.9	
CIFAR-100	ResNet18	Retain-only FT	3	128	1×10^{-3}	0.9	
Tiny-ImageNet	ResNet50	Retain-only FT	3	128	1×10^{-3}	0.9	
CIFAR-10	ResNet18	Random Label	3	128	1×10^{-4}	0.9	
CIFAR-100	ResNet18	Random Label	3	128	3×10^{-3}	–	
Tiny-ImageNet	ResNet50	Random Label	3	128	5×10^{-4}	–	
CIFAR-10	ResNet18	SalUN	3	128	1×10^{-4}	–	threshold=0.5
CIFAR-100	ResNet18	SalUN	3	128	1×10^{-3}	–	threshold=0.5
Tiny-ImageNet	ResNet50	SalUN	3	128	5×10^{-4}	–	threshold=0.5
CIFAR-10	ResNet18	NegGrad+	3	128	1×10^{-4}	–	grad-clip=1.0
CIFAR-100	ResNet18	NegGrad+	3	128	5×10^{-3}	–	grad-clip=1.0
Tiny-ImageNet	ResNet50	NegGrad+	3	128	5×10^{-4}	–	grad-clip=1.0
CIFAR-10	ResNet18	SCRUB	3	64	1×10^{-4}	–	sgda-bsz=64; msteps=2
CIFAR-100	ResNet18	SCRUB	3	64	1×10^{-3}	–	sgda-bsz=64; msteps=2
Tiny-ImageNet	ResNet50	SCRUB	3	64	5×10^{-3}	–	sgda-bsz=64; msteps=2
CIFAR-10	ResNet18	UNSIR	3	128	5×10^{-5}	–	3 epochs impair/repair training
CIFAR-100	ResNet18	UNSIR	3	128	3×10^{-5}	–	3 epochs impair/repair training
Tiny-ImageNet	ResNet50	UNSIR	3	128	2×10^{-5}	–	3 epochs impair/repair training
CIFAR-10	ResNet18	SVD (TF)	–	900	–	–	$\alpha_r = 1000, \alpha_f = 30$
CIFAR-100	ResNet18	SVD (TF)	–	990	–	–	$\alpha_r = 1000, \alpha_f = 30$
Tiny-ImageNet	ResNet50	SVD (TF)	–	999	–	–	$\alpha_r = 30, \alpha_f = 10$
CIFAR-10	ResNet18	Random Label + CMF	4	128	2×10^{-3}	–	
CIFAR-100	ResNet18	Random Label + CMF	4	128	2×10^{-3}	–	
Tiny-ImageNet	ResNet50	Random Label + CMF	4	128	1×10^{-2}	–	
CIFAR-10	ResNet18	SalUN + CMF	4	128	2×10^{-3}	–	threshold=0.5
CIFAR-100	ResNet18	SalUN + CMF	4	128	2×10^{-3}	–	threshold=0.5
Tiny-ImageNet	ResNet50	SalUN + CMF	4	128	1×10^{-2}	–	threshold=0.5
CIFAR-10	ResNet18	NegGrad+ + CMF	3	128	1×10^{-4}	–	grad-clip=1.0
CIFAR-100	ResNet18	NegGrad+ + CMF	3	128	1×10^{-4}	–	grad-clip=1.0
Tiny-ImageNet	ResNet50	NegGrad+ + CMF	3	128	3×10^{-5}	–	grad-clip=1.0
CIFAR-10	ResNet18	SCRUB + CMF	3	64	5×10^{-3}	–	sgda-bsz=64; msteps=2
CIFAR-100	ResNet18	SCRUB + CMF	3	64	5×10^{-3}	–	sgda-bsz=64; msteps=2
Tiny-ImageNet	ResNet50	SCRUB + CMF	3	64	1×10^{-3}	–	sgda-bsz=64; msteps=2
CIFAR-10	ResNet18	UNSIR + CMF	3	128	5×10^{-5}	–	3 epochs impair/repair training
CIFAR-100	ResNet18	UNSIR + CMF	3	128	5×10^{-5}	–	3 epochs impair/repair training
Tiny-ImageNet	ResNet50	UNSIR + CMF	3	128	2×10^{-5}	–	3 epochs impair/repair training

Table 5: Hyperparameters for ViT-based unlearning experiments across CIFAR-10, CIFAR-100, and Tiny-ImageNet.

Dataset	Model	Method	Epochs	Batch	LR	Mom.	Other Key Flags
CIFAR-10	ViT-S/16	Original	10	128	3×10^{-4}	–	pretrained backbone
CIFAR-100	ViT-S/16	Original	10	128	3×10^{-4}	–	pretrained backbone
Tiny-ImageNet	ViT-S/16	Original	10	128	1×10^{-4}	–	pretrained backbone
CIFAR-10	ViT-S/16	Retrain	10	128	3×10^{-4}	–	pretrained backbone
CIFAR-100	ViT-S/16	Retrain	10	128	3×10^{-4}	–	pretrained backbone
Tiny-ImageNet	ViT-S/16	Retrain	10	128	1×10^{-4}	–	pretrained backbone
CIFAR-10	ViT-S/16	Random Label	3	128	3×10^{-4}	–	
CIFAR-100	ViT-S/16	Random Label	3	128	3×10^{-4}	–	
Tiny-ImageNet	ViT-S/16	Random Label	10	128	1×10^{-4}	–	
CIFAR-10	ViT-S/16	SalUN	3	128	3×10^{-4}	–	threshold=0.5
CIFAR-100	ViT-S/16	SalUN	3	128	3×10^{-4}	–	threshold=0.5
Tiny-ImageNet	ViT-S/16	SalUN	10	128	1×10^{-4}	–	threshold=0.5
CIFAR-10	ViT-S/16	NegGrad+	3	128	3×10^{-4}	–	grad-clip=1.0
CIFAR-100	ViT-S/16	NegGrad+	3	128	3×10^{-4}	–	grad-clip=1.0
Tiny-ImageNet	ViT-S/16	NegGrad+	3	128	1×10^{-4}	–	grad-clip=1.0
CIFAR-10	ViT-S/16	Random Label + CMF	4	128	1×10^{-3}	–	
CIFAR-100	ViT-S/16	Random Label + CMF	4	128	2×10^{-2}	–	
Tiny-ImageNet	ViT-S/16	Random Label + CMF	4	128	2×10^{-3}	–	
CIFAR-10	ViT-S/16	SalUN + CMF	4	128	$3 \times 10^{-3}/2 \times 10^{-3}$	–	threshold=0.5
CIFAR-100	ViT-S/16	SalUN + CMF	4	128	$2 \times 10^{-2}/1 \times 10^{-2}$	–	threshold=0.5
Tiny-ImageNet	ViT-S/16	SalUN + CMF	4	128	1×10^{-2}	–	threshold=0.5
CIFAR-10	ViT-S/16	NegGrad+ + CMF	3	128	$5 \times 10^{-5}/5 \times 10^{-4}$	–	grad-clip=1.0
CIFAR-100	ViT-S/16	NegGrad+ + CMF	3	128	$5 \times 10^{-5}/5 \times 10^{-4}$	–	grad-clip=1.0
Tiny-ImageNet	ViT-S/16	NegGrad+ + CMF	3	128	$1 \times 10^{-5}/1 \times 10^{-2}$	–	grad-clip=1.0

D.2 Experimental Results

Table 6: Evaluation of various MU methods on three datasets for unlearning certain number of classes. For all the (unlearned) models, we report both forget accuracy and retain accuracy evaluated for the entire model (labeled as Output) and the feature mapping by linear probe and nearest class center (NCC) classification accuracy. We report the variance of the results to complement the mean performance values shown in Table 1.

Method	**Accuracy**	**CIFAR-10**				**CIFAR-100**				**Tiny-ImageNet**			
		1		3		1		10		1		20	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Output	93.98±0.39	93.98±3.48	94.00±0.89	93.94±2.09	74.61±0.13	74.40±12.86	74.47±0.72	75.88±6.48	65.27±0.06	58.80±12.54	65.15±0.47	66.02±4.25
	Linear Probe	94.02±0.39	94.02±3.54	94.03±0.90	94.00±2.10	74.53±0.12	75.00±11.64	74.38±0.73	75.90±6.60	65.10±0.05	60.80±9.01	64.97±0.44	66.08±3.96
	NCC	94.00±0.39	93.99±3.51	94.03±0.93	93.92±2.17	74.40±0.11	75.00±11.60	74.28±0.72	75.68±6.44	64.65±0.04	60.00±8.25	64.56±0.41	65.00±3.70
Retain-only Retrain	Output	94.74±0.53	0.00±0.00	95.37±1.02	0.00±0.00	76.01±0.14	0.00±0.00	76.50±0.63	0.00±0.00	66.52±0.18	0.00±0.00	66.38±0.61	0.00±0.00
	Linear Probe	90.49±1.00	77.35±4.55	85.64±2.69	67.33±7.48	74.09±1.06	85.20±5.89	69.34±1.42	60.94±5.76	65.90±0.16	46.40±16.88	65.21±0.48	30.36±1.72
	NCC	93.31±0.45	47.06±3.68	91.37±1.86	37.07±8.71	73.90±1.28	70.40±8.73	70.98±0.68	43.18±6.33	63.57±1.10	71.20±2.68	59.37±0.56	44.30±2.52
Retain-only FT	Output	94.26±0.57	47.67±29.14	95.24±1.10	52.48±20.15	74.08±0.19	53.20±18.75	74.53±0.99	64.96±7.84	65.26±0.09	37.60±9.84	65.60±0.63	50.92±6.92
	Linear Probe	93.94±0.32	89.71±5.08	94.14±0.81	90.44±2.40	73.89±0.16	73.80±10.76	73.97±0.74	74.50±6.31	64.44±0.11	56.00±9.80	64.05±0.44	63.82±2.80
	NCC	93.70±0.25	89.63±4.27	93.80±0.87	88.75±2.69	74.04±0.16	73.20±10.38	74.08±0.79	73.56±6.18	64.00±0.14	57.60±8.17	63.90±0.29	63.58±2.98
NegGrad+	Output	92.85±1.20	0.00±0.00	93.29±0.84	0.01±0.01	69.90±1.53	0.00±0.00	70.80±1.03	0.28±0.22	57.96±0.89	0.00±0.00	59.06±2.62	0.00±0.00
	Linear Probe	92.14±0.46	67.09±4.88	88.18±1.50	73.91±5.55	72.55±0.44	67.20±13.55	72.20±0.51	62.32±7.49	60.43±0.40	58.00±9.17	60.75±0.44	54.68±2.33
	NCC	91.33±0.62	52.00±3.45	87.28±2.10	59.17±7.68	71.58±0.81	41.40±6.66	70.08±0.80	41.24±4.58	59.01±0.43	37.60±8.17	56.45±1.21	36.78±3.98
SVD	Output	92.02±1.47	0.00±0.00	94.05±1.02	57.43±1.35	71.09±0.60	0.00±0.00	73.10±1.23	55.56±4.46	64.43±0.19	2.00±1.41	65.45±0.45	59.02±4.38
	Linear Probe	90.44±1.24	61.80±4.21	92.43±1.65	83.58±2.23	73.14±0.37	67.00±10.26	73.38±0.84	74.08±5.37	63.10±0.06	60.40±11.78	63.03±0.52	63.64±4.66
	NCC	90.11±1.64	34.54±3.88	93.23±1.47	72.32±1.62	71.81±0.58	64.80±9.09	73.33±0.97	72.22±5.38	64.65±0.04	60.00±8.25	64.56±0.41	65.00±3.70
Random-label	Output	92.93±0.94	0.00±0.00	94.14±1.06	0.00±0.00	72.33±0.15	0.00±0.00	72.19±0.19	0.00±0.00	65.48±0.05	0.40±0.89	64.85±0.47	0.98±0.33
	Linear Probe	92.65±0.58	92.49±3.70	92.45±1.00	90.25±3.07	73.08±0.27	79.00±11.60	72.08±0.59	72.08±7.13	64.07±0.19	58.00±8.49	62.02±0.48	57.82±2.86
	NCC	92.25±1.09	80.25±11.82	91.92±1.20	73.22±11.14	72.38±0.51	87.20±8.67	70.67±0.65	62.22±7.95	63.20±0.28	69.20±7.69	59.42±0.55	42.72±2.57
SalUn	Output	93.19±0.79	0.00±0.00	94.43±0.96	0.00±0.00	72.96±0.20	0.00±0.00	72.92±0.84	0.06±0.09	65.49±0.04	0.40±0.89	64.63±0.31	3.40±0.83
	Linear Probe	93.05±0.42	92.57±3.23	92.89±0.81	89.63±3.78	73.26±0.12	77.80±16.12	72.10±0.70	72.66±6.93	64.07±0.07	55.60±9.53	61.86±0.28	56.52±2.54
	NCC	91.31±0.42	93.70±2.14	91.99±0.98	68.45±7.29	72.65±0.30	85.60±10.64	70.62±0.83	63.34±6.14	63.33±0.20	68.80±6.57	59.07±0.57	39.70±3.48
SCRUB	Output	91.37±2.25	0.00±0.00	93.61±1.25	0.00±0.00	73.67±0.17	0.20±0.45	74.56±0.83	0.32±0.32	65.43±0.11	1.20±1.79	65.30±0.46	5.48±3.46
	Linear Probe	91.71±1.76	74.79±8.03	91.88±1.11	78.23±2.99	73.69±0.13	72.40±13.20	73.05±0.72	66.92±6.67	64.44±0.34	56.00±9.49	62.87±0.50	56.90±4.63
	NCC	89.96±1.60	55.30±10.92	89.73±1.85	47.52±6.11	73.51±0.19	72.40±10.76	72.35±0.54	53.58±6.04	64.37±0.08	60.40±6.07	61.41±0.41	53.50±4.93
UNSIR	Output	91.84±0.75	0.48±0.58	92.87±1.33	0.01±0.01	73.77±0.28	3.80±3.56	73.58±0.85	14.58±4.89	64.66±0.27	0.00±0.00	65.46±0.61	9.72±5.11
	Linear Probe	89.71±0.50	85.29±7.14	88.21±1.97	70.59±6.63	73.40±0.27	73.60±14.88	72.15±0.78	67.64±6.85	63.88±0.23	61.20±9.96	62.87±0.33	61.10±3.24
	NCC	89.73±0.49	62.72±5.04	87.73±2.37	49.42±8.20	73.05±0.40	71.00±13.32	71.26±0.55	59.24±6.82	63.16±0.16	59.20±5.02	61.80±0.32	56.14±2.40

Table 7: Unlearning from full model VS only classifier evaluated through output-level forget and retain accuracies. We report the variance of the results to complement the mean performance values shown in Table 2.

Method	**Layers Finetuned**	**CIFAR-10**				**CIFAR-100**			
		1		3		1		10	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Full Model	93.98±0.39	93.98±3.48	94.00±0.89	93.94±2.09	74.61±0.13	74.40±12.86	74.47±0.72	75.88±6.48
Retain-only Retrain	Full Model	94.74±0.53	0.00±0.00	95.37±1.02	0.00±0.00	76.01±0.14	0.00±0.00	76.50±0.63	0.00±0.00
Retain-only FT	Full Model	94.26±0.57	47.67±29.14	95.24±1.10	52.48±20.15	74.08±0.19	53.20±18.75	74.53±0.99	64.96±7.84
	Classifier only	93.20±0.72	0.00±0.00	93.66±0.85	0.00±0.00	73.29±0.37	0.00±0.00	73.51±0.84	0.00±0.00
NegGrad+	Full Model	92.85±1.20	0.00±0.00	93.29±0.84	0.01±0.01	69.90±1.53	0.00±0.00	70.80±1.03	0.28±0.22
	Classifier only	93.08±0.85	0.00±0.00	93.72±0.97	0.00±0.00	73.28±1.11	0.00±0.00	66.85±6.25	0.00±0.00
Random-label	Full Model	92.93±0.94	0.00±0.00	94.14±1.06	0.00±0.00	72.33±0.15	0.00±0.00	72.19±0.19	0.00±0.00
	Classifier only	93.39±0.75	0.00±0.00	94.56±1.04	0.00±0.00	73.87±0.12	0.20±0.40	74.09±0.73	2.32±0.56
Salun	Full Model	93.19±0.79	0.00±0.00	94.43±0.96	0.00±0.00	72.96±0.20	0.00±0.00	72.92±0.84	0.06±0.09
	Classifier only	93.38±0.72	0.00±0.00	94.44±1.07	0.00±0.00	73.93±0.20	1.00±0.63	73.98±0.72	4.18±1.20
SVD	Full Model	92.02±1.47	0.00±0.00	94.05±1.02	57.43±1.35	71.09±0.60	0.00±0.00	73.10±1.23	55.56±4.46
	Classifier only	93.53±0.66	0.01±0.03	93.73±0.87	68.45±9.33	73.47±0.18	2.60±1.86	74.13±0.66	50.56±5.57
SCRUB	Full Model	91.37±2.25	0.00±0.00	93.61±1.25	0.00±0.00	73.67±0.17	0.20±0.45	74.56±0.83	0.32±0.32
	Classifier only	93.12±0.76	0.00±0.00	93.68±0.95	0.00±0.00	74.01±0.24	5.00±9.01	74.01±0.75	0.00±0.00
UNSIR	Full Model	91.84±0.75	0.48±0.58	92.87±1.33	0.01±0.01	73.77±0.28	3.80±3.56	73.58±0.85	14.58±4.89
	Classifier only	94.48±0.66	0.00±0.00	94.46±0.82	1.39±0.96	74.64±0.13	0.00±0.00	74.27±0.79	0.52±1.11

Table 8: Evaluation of MU methods on **ResNet** with CMF classifiers on three datasets for unlearning certain number of classes. We report the variance of the results to complement the mean performance values shown in Table 3.

Method	**Accuracy**	**CIFAR-10**				**CIFAR-100**				**Tiny-ImageNet**			
		1		**3**		**1**		**10**		**1**		**20**	
		Retain	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**
Original	Output	93.98±0.39	93.98±3.48	94.00±0.89	93.94±2.09	74.61±0.13	74.40±12.86	74.47±0.72	75.88±6.48	65.27±0.06	58.80±12.54	65.15±0.47	66.02±4.25
	Linear Probe	94.02±0.39	94.02±3.54	94.03±0.90	94.00±2.10	74.53±0.12	75.00±11.64	74.38±0.73	75.90±6.60	65.10±0.05	60.80±9.01	64.97±0.44	66.08±3.96
	NCC	94.00±0.39	93.99±3.51	94.03±0.93	93.92±2.17	74.40±0.11	75.00±11.60	74.28±0.72	75.68±6.44	64.65±0.04	60.00±8.25	64.56±0.41	65.00±3.70
Retain-only Retrain	Output	94.74±0.53	0.00±0.00	95.37±1.02	0.00±0.00	76.01±0.14	0.00±0.00	76.50±0.63	0.00±0.00	66.52±0.18	0.00±0.00	66.38±0.61	0.00±0.00
	Linear Probe	90.49±1.00	77.35±4.55	85.64±2.69	67.33±7.48	74.09±1.06	85.20±5.89	69.34±1.42	60.94±5.76	65.90±0.16	46.40±16.88	65.21±0.48	30.36±1.72
	NCC	93.31±0.45	47.06±3.68	91.37±1.86	37.07±8.71	73.90±1.28	70.40±8.73	70.98±0.68	43.18±6.33	63.57±1.10	71.20±2.68	59.37±0.56	44.30±2.52
Random-label with CMF	Output	94.27±0.57	80.70±8.73	94.81±0.99	75.69±4.88	74.38±0.12	55.60±16.35	74.85±0.77	54.40±7.52	62.01±0.25	22.40±14.66	62.25±0.61	22.20±1.76
	Linear Probe	94.19±0.52	85.71±5.82	94.49±0.95	81.47±3.78	74.63±0.17	59.20±12.68	74.98±0.76	66.44±6.87	62.56±0.11	32.80±9.96	62.38±0.30	36.58±2.56
	NCC	94.25±0.55	82.04±8.34	94.73±0.95	76.74±4.65	74.49±0.11	59.20±15.42	74.82±0.79	60.24±7.04	61.96±0.24	27.20±11.80	61.93±0.48	28.24±2.40
Salun with CMF	Output	94.33±0.60	78.07±10.38	95.01±0.99	75.96±4.06	74.62±0.11	60.40±14.31	74.98±0.69	62.10±6.97	62.62±0.08	30.00±14.21	63.14±0.42	32.74±2.53
	Linear Probe	94.26±0.58	84.68±7.30	94.60±1.01	83.39±4.22	74.79±0.14	59.40±15.99	75.18±0.72	67.20±6.73	63.18±0.15	41.20±8.07	63.12±0.20	46.14±2.37
	NCC	94.32±0.60	79.50±9.80	94.91±0.99	77.31±4.34	74.78±0.05	63.60±13.92	75.03±0.65	65.20±6.75	62.67±0.10	37.20±9.96	62.77±0.31	38.10±2.10
NegGrad+ with CMF	Output	91.87±1.15	54.50±6.24	94.97±0.87	60.00±6.42	71.82±1.26	41.00±15.46	71.45±0.70	30.38±4.43	61.18±0.67	22.00±12.88	60.82±1.04	41.42±3.41
	Linear Probe	92.35±0.94	68.02±5.00	94.60±0.93	75.29±4.62	72.86±0.72	55.20±13.88	72.35±0.70	50.62±5.74	62.90±0.45	41.20±11.01	62.66±0.92	53.82±2.59
	NCC	91.99±1.10	57.83±5.83	94.93±0.87	62.89±6.01	72.36±1.11	51.20±14.04	71.75±0.72	41.54±5.18	62.31±0.40	38.80±11.54	62.11±0.98	51.30±2.66
Scrub with CMF	Output	92.51±0.91	33.78±6.39	95.37±1.00	35.11±7.27	73.86±0.34	40.60±18.53	74.27±0.74	35.02±7.13	61.64±0.18	27.20±13.54	63.31±0.35	47.76±4.62
	Linear Probe	92.48±0.83	60.68±5.61	95.26±0.98	62.77±6.06	74.03±0.41	55.60±15.95	74.18±0.67	58.34±7.13	62.13±0.21	36.80±4.82	63.76±0.34	54.28±3.55
	NCC	92.48±0.90	35.53±6.51	95.34±0.98	40.04±7.86	73.87±0.39	47.00±15.41	74.12±0.68	42.82±6.21	61.66±0.25	34.40±6.07	63.28±0.35	50.42±3.23
UNSIR with CMF	Output	91.79±0.92	12.91±7.43	93.56±1.02	11.51±3.00	72.72±0.15	21.00±11.98	72.61±1.10	9.16±3.23	60.81±0.22	14.00±8.25	61.34±0.54	14.44±0.90
	Linear Probe	91.63±0.90	31.16±6.35	93.01±1.17	28.65±3.13	72.91±0.16	35.20±17.25	72.51±0.87	20.98±5.73	60.96±0.21	26.80±3.35	61.27±0.36	23.02±0.59
	NCC	91.87±0.91	9.29±4.22	93.79±1.03	8.16±2.32	72.67±0.16	20.20±10.57	72.48±1.17	9.94±3.48	60.63±0.24	26.00±3.16	60.80±0.46	16.72±1.44

Table 9: Evaluation of machine unlearning (MU) methods on **ViT-S/16** across CIFAR-10, CIFAR-100, and Tiny-ImageNet. We report mean accuracy (± standard deviation) on **retain** and **forget** subsets under different unlearning bucket sizes (single vs. multi-class). Results are shown for three evaluation heads: the full model classifier, a linear probe on frozen features (LP), and the Nearest Class-Center (NC) classifier. All experiments are based on models pre-trained on **ImageNet**. For ViT-S/16, the pre-trained model is obtained by fine-tuning the ImageNet-pre-trained backbone on the full target dataset. *Retrain* refers to a model obtained by fine-tuning the same pre-trained backbone using only the retain subset, serving as an oracle baseline that fully removes the target data. All unlearning methods are likewise initialized from the same pre-trained model. All values are averaged over multiple runs..

Method	**Accuracy**	**CIFAR-10**				**CIFAR-100**				**Tiny-ImageNet**			
		1		3		1		10		1		20	
		Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget	Retain	Forget
Original	Output	98.20±0.15	98.20±1.39	98.16±0.44	98.29±1.02	90.09±0.06	92.00±5.87	90.09±0.46	90.32±4.18	84.90±0.05	79.20±10.83	84.81±0.31	85.44±2.82
	Linear Probe	98.17±0.14	98.17±1.26	98.15±0.36	98.22±0.85	89.93±0.05	92.20±5.07	89.94±0.43	90.08±3.83	84.97±0.04	81.20±7.82	84.87±0.26	85.68±2.31
	NCC	98.06±0.20	98.06±1.76	98.02±0.50	98.15±1.16	89.62±0.05	93.00±5.10	89.67±0.47	89.58±4.01	83.62±0.03	80.40±6.84	83.49±0.31	84.70±2.83
Retain-only Retrain	Output	98.46±0.29	0.00±0.00	98.61±0.51	0.00±0.00	89.95±0.05	0.00±0.00	90.10±0.46	0.00±0.00	83.85±0.14	0.00±0.00	84.14±0.67	0.00±0.00
	Linear Probe	98.28±0.08	97.71±1.24	98.17±0.36	97.35±1.32	90.32±0.11	90.80±7.09	90.13±0.42	87.82±4.67	84.47±0.13	77.60±5.55	84.45±0.38	81.44±2.79
	NCC	98.12±0.17	94.24±3.73	97.98±0.43	94.51±1.95	89.24±0.09	87.80±5.97	89.31±0.36	81.88±5.19	82.93±0.05	74.80±5.76	82.74±0.30	78.64±3.19
Random-label	Output	98.50±0.25	0.00±0.00	98.69±0.43	0.00±0.00	90.59±0.10	0.00±0.00	90.79±0.42	0.10±0.17	85.87±0.07	7.20±3.35	86.23±0.55	16.24±4.51
	Linear Probe	98.35±0.10	98.64±1.08	98.15±0.47	98.26±1.56	90.47±0.05	94.00±5.15	90.26±0.37	88.32±5.18	85.69±0.07	82.80±5.40	85.57±0.23	85.10±2.47
	NCC	97.50±0.24	99.57±0.46	96.46±0.70	97.84±3.37	90.01±0.09	96.00±4.64	89.56±0.62	82.70±6.70	84.22±0.09	82.40±6.84	84.42±0.35	83.92±3.07
Salun	Output	98.46±0.25	0.00±0.00	98.68±0.46	0.00±0.00	90.58±0.13	0.20±0.45	90.64±0.46	0.18±0.16	85.64±0.06	7.60±5.18	85.99±0.71	15.72±4.48
	Linear Probe	98.33±0.18	98.59±1.16	98.13±0.35	98.26±1.59	90.36±0.19	93.60±5.18	90.22±0.51	88.24±4.93	85.50±0.09	81.60±7.67	85.33±0.32	84.74±2.58
	NCC	97.51±0.19	99.51±0.49	96.61±0.46	97.85±3.50	89.96±0.11	95.60±4.88	89.33±0.62	81.68±4.38	84.16±0.11	82.00±8.72	84.11±0.32	83.46±3.06
NegGrad+	Output	98.00±0.33	0.00±0.00	98.21±0.56	6.63±14.82	89.88±0.26	0.00±0.00	89.85±0.55	6.14±5.61	85.12±0.26	0.00±0.00	85.48±0.73	0.52±0.69
	Linear Probe	98.03±0.16	96.15±2.47	97.57±0.67	95.76±1.13	90.24±0.19	87.20±7.43	89.96±0.55	82.98±4.66	84.95±0.18	75.20±5.93	84.74±0.29	67.50±4.21
	NCC	94.88±1.60	87.05±5.49	93.86±2.70	73.38±4.37	89.00±0.32	86.80±6.26	89.20±0.73	43.40±5.94	83.29±0.29	84.40±5.55	82.94±0.30	57.16±7.44

Table 10: Evaluation of **CMF-based machine unlearning** methods on the **ViT-S/16** model across CIFAR-10, CIFAR-100, and Tiny-ImageNet. Results report mean accuracy on retain and forget subsets under different unlearning bucket sizes. These results demonstrate that CMF-based unlearning effectively removes information not only at the output level but also in the **feature representations** of ImageNet-pretrained vision transformers.

Method	**Accuracy**	**CIFAR-10**				**CIFAR-100**				**Tiny-ImageNet**			
		1		**3**		**1**		**10**		**1**		**20**	
		Retain	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**	**Retain**	**Forget**
Original	Output	98.20±0.15	98.20±1.39	98.16±0.44	98.29±1.02	90.09±0.06	92.00±5.87	90.09±0.46	90.32±4.18	84.90±0.05	79.20±10.83	84.81±0.31	85.44±2.82
	Linear Probe	98.17±0.14	98.17±1.26	98.15±0.36	98.22±0.85	89.93±0.05	92.20±5.07	89.94±0.43	90.08±3.83	84.97±0.04	81.20±7.82	84.87±0.26	85.68±2.31
	NCC	98.06±0.20	98.06±1.76	98.02±0.50	98.15±1.16	89.62±0.05	93.00±5.10	89.67±0.47	89.58±4.01	83.62±0.03	80.40±6.84	83.49±0.31	84.70±2.83
Retain-only Retrain	Output	98.46±0.29	0.00±0.00	98.61±0.51	0.00±0.00	89.95±0.05	0.00±0.00	90.10±0.46	0.00±0.00	83.85±0.14	0.00±0.00	84.14±0.67	0.00±0.00
	Linear Probe	98.28±0.08	97.71±1.24	98.17±0.36	97.35±1.32	90.32±0.11	90.80±7.09	90.13±0.42	87.82±4.67	84.47±0.13	77.60±5.55	84.45±0.38	81.44±2.79
	NCC	98.12±0.17	94.24±3.73	97.98±0.43	94.51±1.95	89.24±0.09	87.80±5.97	89.31±0.36	81.88±5.19	82.93±0.05	74.80±5.76	82.74±0.30	78.64±3.19
Random-label with CMF	Output	97.38±0.30	52.77±15.71	96.78±1.23	46.83±8.54	86.99±0.18	55.20±14.82	87.81±0.78	53.06±14.14	77.60±0.10	61.60±8.41	78.12±0.61	62.56±2.43
	Linear Probe	97.20±0.33	72.30±9.32	96.19±1.14	66.69±8.22	84.79±0.18	24.60±14.77	86.01±0.97	41.10±16.54	80.20±0.09	68.40±7.92	80.57±0.52	71.04±3.19
	NCC	97.35±0.29	55.82±14.25	96.66±1.20	49.97±9.02	86.97±0.20	55.20±11.90	87.79±0.79	52.54±14.90	77.09±0.11	61.20±8.44	77.55±0.59	60.48±3.34
Salun with CMF	Output	97.30±0.38	53.69±15.79	96.94±1.20	49.31±10.28	86.20±0.43	53.20±24.73	87.42±0.69	46.40±12.73	77.91±0.07	68.00±8.00	78.34±0.76	68.66±3.22
	Linear Probe	97.11±0.37	71.98±9.51	96.23±1.28	68.07±8.89	83.43±0.59	23.20±18.21	85.37±0.89	34.70±14.59	81.07±0.11	74.40±8.88	81.36±0.32	75.24±2.45
	NCC	97.26±0.35	56.14±14.52	96.81±1.23	52.30±10.73	86.20±0.46	51.20±22.88	87.39±0.70	46.34±12.84	77.61±0.08	64.00±8.49	77.84±0.77	67.62±3.17
NegGrad+ with CMF	Output	92.94±2.02	41.42±14.64	93.41±1.23	48.50±6.68	85.48±1.45	55.80±13.18	83.91±2.25	47.08±5.58	77.81±1.57	17.20±8.44	65.38±8.49	31.24±4.90
	Linear Probe	93.76±1.60	55.61±10.32	93.68±1.02	56.75±8.04	83.47±1.72	24.40±15.79	81.19±2.46	22.30±4.31	81.06±1.11	34.40±12.36	69.65±8.88	33.98±9.05
	NCC	93.04±1.92	43.89±12.64	93.45±1.16	48.08±5.34	85.53±1.47	58.40±13.32	83.97±2.30	45.28±5.97	77.10±1.82	12.00±3.74	63.65±8.81	27.58±4.94