LMEraser: Large Model Unlearning via Adaptive Prompt Tuning

Jie Xu[1] **Zihan Wu**[2] **Cong Wang**[1] **Xiaohua Jia**[1]
[1]Department of Computer Science, City University of Hong Kong
[2]Department of Electrical Engineering, City University of Hong Kong

Abstract

To address the growing demand for privacy protection in machine learning, we propose an efficient and exact machine unlearning method for **L**arge **M**odels, called **LM**Eraser. LMEraser takes a divide-and-conquer strategy with an adaptive prompt tuning mechanism to isolate data influence effectively. The training dataset is partitioned into public and private datasets. Public data are used to train the backbone of the model. Private data are clustered based on their diversity, and each cluster tunes a tailored prompt independently. This approach enables targeted unlearning by updating affected prompts, significantly reduces unlearning costs and maintains high model performance. Evaluations show that LMEraser reduces unlearning costs by 100 times compared to prior work without compromising model utility.

1 INTRODUCTION

Large models such as GPT (Brown et al., 2020), Vision Transformers (ViT) (Dehghani et al., 2023), and CLIP (Radford et al., 2021), have achieved remarkable performance across various tasks. However, this success has raised privacy concerns. Large models trained on massive datasets potentially memorize and expose sensitive user information (Wu et al., 2023b). To address these concerns, regulations (Voigt and Von dem Bussche, 2017; Office of the Attorney General, 2023) grant users the "right to be forgotten." Consequently, machine unlearning, the ability to selectively remove the influence of specific training data from a trained model, has emerged as a key solution in balancing privacy protection and computational efficiency.

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

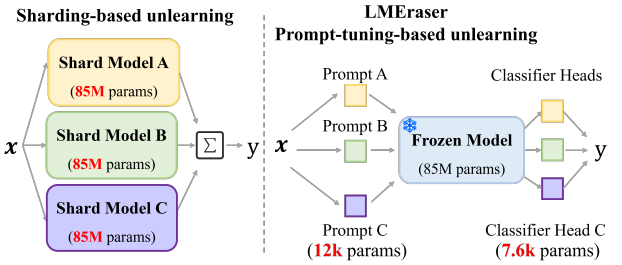


Figure 1: Comparison of sharding-based unlearning and LMEraser. Sharding-based methods retrain entire shard models (about 85M parameters), while LMEraser efficiently updates affected prompts (about 12k parameters), drastically reducing computational costs.

Existing unlearning methods, primarily designed for smaller-scale models, show inefficiency and ineffectiveness in addressing challenges posed by the scale and complexity of large models. For example, sharding-based unlearning methods (Ginart et al., 2019; Bourtoule et al., 2021) partition training data randomly, training independent models on each shard and combining their outputs for predictions (Figure 1). While unlearning only requires retraining the affected shard model, this method incurs high computational and storage costs when applied to large models, as it necessitates maintaining multiple large-scale shard models. Other methods attempt to manipulate parameters directly, but they rely on assumptions including loss convexity that do not hold for large models, failing to completely remove data influence (Guo et al., 2020; Suriyakumar and Wilson, 2022; Wu et al., 2022).

To completely remove the influence of specific training data from large models, there are several challenges. First, it is difficult to trace and quantify the impact of individual training data points, as their influence is dispersed across the model's architecture. Second, it is computationally expensive to implement unlearning operations, which involve intensive recalculations and adjustments to billions of parameters. Third, unlearning risks performance degradation and even catastrophic forgetting (Shibata et al., 2021), as it may

disrupt learned representations that are distributed across shared model components.

We propose LMEraser, an efficient and exact unlearning method that completely removes specific data influences from large vision models. The idea of LMEraser lies in two observations: 1) The training data of large models can be divided into two categories: *public data* and *private data*. Public data, such as ImageNet (Deng et al., 2009) and COCO (Lin et al., 2014), are collected, cleansed, and validated by research institutions or nonprofits, making them widely available and extensively used. Private data, such as medical records or bank transactions, are sensitive and require permissions for access and use. While users (private data owners) may request removing their data from trained models due to the "right to be forgotten", such concerns are nonexistent with public data. 2) Large models typically adopt a two-phase training process. The pre-training phase trains a model backbone using massive data to absorb broad foundational knowledge. Then, the tuning phase utilizes the backbone and task-specific data to enable the model to handle specific tasks.

Based on these foundations, LMEraser has three core designs to achieve large model unlearning. First, LMEraser takes a divide-and-conquer strategy with a prompt tuning architecture to isolate data influence, as shown in Figure 1. The training dataset is partitioned into public and private datasets. Public data are used to pre-train the model's backbone, while private data are used for prompt tuning. Second, LMEraser freezes the backbone's parameters after pre-training, reducing recalculations during unlearning and preserving performance. Finally, LMEraser designs an adaptive mechanism for prompt tuning. It adaptively clusters private data based on data diversity and generates tailored prompts and classifier heads for each cluster. Predictions are made using the nearest cluster's prompt. Unlearning only requires updating the affected cluster's prompt and classifier head, reducing computational costs and minimizing performance degradation.

In summary, the contributions of this paper include:

- We propose LMEraser, a novel unlearning method for large vision models that achieves complete and efficient removal of data influences while preserving model performance.
- We design an adaptive mechanism for prompt tuning that enhances performance and reduces unlearning costs. Our theoretical analysis highlights LMEraser's efficiency gains.
- LMEraser outperforms baselines and achieves a 100-fold reduction in unlearning costs while achieving high model utility.

2 RELATED WORK

Prompt Tuning. Prompt tuning efficiently adapts large pre-trained models to downstream tasks by appending learnable vectors ("prompts") to input data, enabling effective knowledge transfer without modifying the entire model architecture (Liu et al., 2023; Huang et al., 2023). Prompt tuning is computationally and storage efficient, as a single backbone can handle multiple tasks with different small prompts. In contrast, model parameter fine-tuning requires modifying multiple large backbones to address multiple tasks (Lester et al., 2021).

In the visual domain, various prompt tuning methods have been developed using a pre-trained ViT (Dosovitskiy et al., 2020). VPT (Jia et al., 2022) inputs learnable prompts as embeddings alongside image embeddings into ViT, while VP (Bahng et al., 2022) uses pixel-level prompts. DAM-VP (Huang et al., 2023) enhances the diversity of prompts, but its head-missing method performs poorly in complex scenarios. APT (Bowman et al., 2023) constructs user-centric models with prompts generated from user-selected sub-datasets. However, these methods often overlook data partitioning strategies that could enhance both performance and privacy. Our work addresses this limitation with an adaptive prompt tuning mechanism that enables efficient unlearning.

Machine Unlearning. Machine unlearning can be categorized into *Exact Unlearning* methods and *Approximate Unlearning* methods based on the residual influence of removed data (Xu et al., 2024). Exact unlearning methods completely remove a data point's influence on the model (Liu et al., 2022). This typically involves dividing the training data into shards, training independent models on these shards, and retraining only the affected models when data is unlearned (Bourtoule et al., 2021; Brophy and Lowd, 2020; Su and Li, 2023). While effective, these methods are resource-intensive for large models.

Approximate unlearning reduces specific data influence to an acceptable level but does not guarantee complete removal (Li et al., 2024; Kurmanji et al., 2024). Some studies use influence functions (Koh and Liang, 2017) to estimate and eliminate specific data influence (Guo et al., 2020; Tanno et al., 2022; Warnecke et al., 2023; Chen et al., 2024; Wu et al., 2023a). However, these methods rely on convex loss functions and inverting the Hessian matrix, which is infeasible for large models. Other methods, such as noise injection and selective retraining (Golatkar et al., 2020a,b, 2021), model pruning (Wang et al., 2022) have also been explored but may leave residual effects, potentially compromising data privacy.

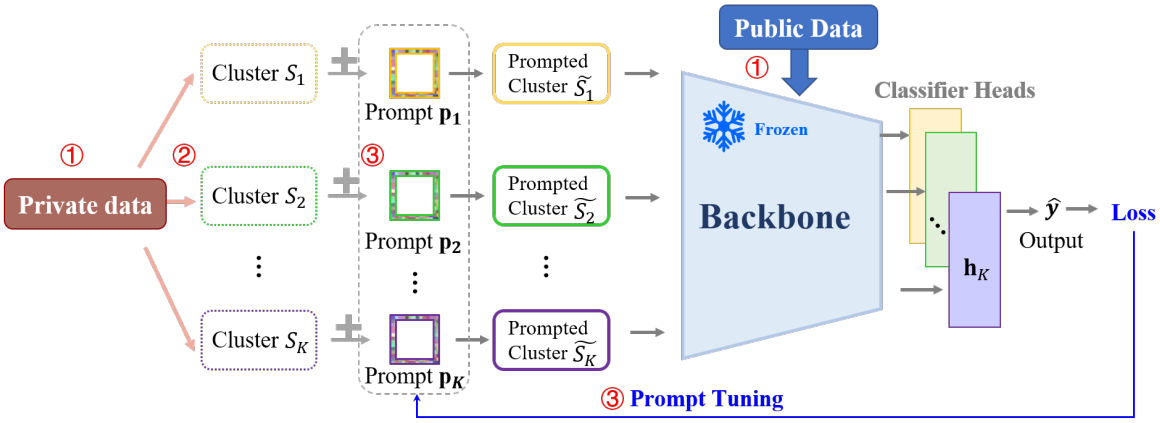


Figure 2: Overview of the LMEraser training process: 1) Partition the training dataset into public and private datasets and pre-train the backbone on the public dataset; 2) Adaptively cluster the private data based on diversity; 3) Tune prompts and classifier heads for each cluster.

3 PRELIMINARIES AND PROBLEM DEFINITION

3.1 Vision Transformer (ViT)

The ViT (Dosovitskiy et al., 2020) splits an input image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ into non-overlapping patches, where H, W, C are height, width and channels. These patches are flattened into embeddings and combined with a learnable classification token to form a sequence $\mathbf{Z}$. The sequence $\mathbf{Z}$ is processed by a Transformer encoder (Vaswani et al., 2017), and the output is fed to a classifier head for final classification results.

3.2 Visual Prompt Tuning

Definition 1 (Pixel-Level Border Prompt) *A pixel-level border prompt* $\mathbf{p} \in \mathbb{R}^{H'\times W'\times C'}$ *is a modification overlayed on the borders of an image* $\mathbf{x}$. *This modification is represented as:* $\tilde{\mathbf{x}} = g(\mathbf{x}, \mathbf{p})$, *where* $g : \mathbb{R}^{H\times W\times C} \times \mathbb{R}^{H'\times W'\times C'} \to \mathbb{R}^{H\times W\times C}$ *is a function that overlays* $\mathbf{p}$ *onto* $\mathbf{x}$. *The prompted image* $\tilde{\mathbf{x}}$ *is then input to a pre-trained model.*

Visual prompt tuning optimizes $\mathbf{p}$ to minimize a task-specific loss function, resulting in an optimized prompt $\mathbf{p}^*$ that enhances the pre-trained backbone's performance on downstream tasks.

3.3 Exact Unlearning

Definition 2 (Exact Unlearning) *Given a model* $f : X \to Y$ *trained on dataset* $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, *where* $\mathbf{x}_i \in X$ *are inputs and* $y_i \in Y$ *are labels. Exact unlearning of a subset* $\mathcal{D}_u \subseteq \mathcal{D}$ *is achieved if the updated model* $f' : X \to Y$ *satisfies the following conditions:*

- *Completeness: The updated model* f' *should have no residual influence from the unlearned data* $\mathcal{D}_u$, *behaving identically to a model trained without* $\mathcal{D}_u$.
- *Efficiency: The process of unlearning* $\mathcal{D}_u$ *is computationally efficient, requiring significantly fewer resources than retraining from scratch using* $\mathcal{D}\setminus\mathcal{D}_u$.

3.4 Problem Definition

Our goal is to design an efficient and exact unlearning method for large-scale vision models that can completely remove specific private data influence from a trained model. Formally, we aim to design an unlearning method that results in an unlearned model $f' : X \to Y$ that behaves as if it were trained on the dataset $\mathcal{D}' = \mathcal{D} \setminus \mathcal{D}_u$, where $\mathcal{D}_u \subseteq \mathcal{D}$ is the data to be removed. This method must satisfy the conditions of exact unlearning in Section 3.3 while addressing the unique challenges posed by large models.

4 METHOD

4.1 Data Partitioning and Pre-training

Existing methods train large models on mixed *public* and *private* datasets, leading to privacy risks and inefficient unlearning, as each request necessitates retraining the entire model using the entire training dataset (Golatkar et al., 2021).

To address these limitations, LMEraser strategically partitions the training dataset $\mathcal{D}$ into a public dataset $\mathcal{D}_p$ and private dataset $\mathcal{D}_s$ based on their sources after collection, as shown in Eq. (1).

$$\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_s \quad \text{and} \quad \mathcal{D}_p \cap \mathcal{D}_s = \emptyset. \tag{1}$$

Public data $\mathcal{D}_p$, such as those from ImageNet (Deng et al., 2009) and COCO (Lin et al., 2014), are collected and cleansed by research institutions or non-profit organizations, complying with legal and ethical standards. Public data are widely accessible and have been extensively verified by the scientific community, ensuring their reliability and safety.

Private data $\mathcal{D}_s$, such as medical records and bank transactions, contain sensitive information and are governed by strict privacy and security regulations.

LMEraser pre-trains a ViT backbone $\mathcal{M}$ using only public dataset $\mathcal{D}_p$, as shown in Figure 2. The distribution of public dataset $\mathcal{D}_p$ is represented by $P_p(X)$. Formally, we optimize:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim P_p(X)}[L(\mathcal{M}_\theta(x), y)], \tag{2}$$

where θ are backbone parameters, L is the loss function, and y is the ground truth label.

After pre-training, the pre-trained backbone's parameters θ are frozen, isolating private data influence from the backbone and avoiding the need to retrain the backbone for unlearning private data. Moreover, it preserves pre-training knowledge, enhancing performance stability and preventing catastrophic forgetting where removing a single data point could disrupt the learned knowledge of others (Shibata et al., 2021).

4.2 Adaptive Prompt Tuning

4.2.1 Adaptive Private Data Clustering

While pre-training on public datasets captures fundamental knowledge, it lacks domain-specific features. Private data, characterized by its unique attributes and non-standard distributions, is valuable for handling domain-specific tasks (Bommasani et al., 2021). However, the rich variability within private datasets poses significant challenges for prompt tuning (Zhou et al., 2022). Recent studies indicate that the effectiveness of prompts is greatly influenced by data diversity of features and patterns (Bahng et al., 2022). A single prompt may suffice for datasets with low diversity, but it is less effective for datasets with high diversity because a single prompt lacks the flexibility to adapt to the complex variations in diverse datasets (Huang et al., 2023). Moreover, relying on a single prompt reduces the efficiency of the unlearning process, as removing a private data point necessitates retraining the prompt with the entire private dataset.

To address these challenges, we propose an *adaptive prompt tuning mechanism* based on the diversity-aware method (Huang et al., 2023), as shown in Figure 2. Unlike traditional unlearning strategies that randomly partition data into a pre-determined number of shards without considering data characteristics, our mechanism adaptively clusters private data using Hierarchical Agglomerative Clustering (HierCluster) (Müllner, 2011) and then trains tailored prompts and classifier heads for each cluster. Detailed descriptions of the HierCluster can be found in Appendix 1.

Algorithm 1 Adaptive Private Data Clustering

Input: Private dataset $\mathcal{D}_s$, pre-trained backbone $\mathcal{M}$
Output: Clustered private dataset $\{\mathbf{S}_1, ..., \mathbf{S}_K\}$ with cluster prototypes $\{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_K\}$

$\mathcal{D}_{\text{sample}} \leftarrow \text{RandomSample}(\mathcal{D}_s)$
for each data point $\mathbf{x}_i$ in $\mathcal{D}_{\text{sample}}$ **do**
  $\mathbf{f}_i \leftarrow \mathcal{M}(\mathbf{x}_i)$ ▷ Extract feature of $\mathbf{x}_i$
end for
for each pairs (i, j) in $\mathcal{D}_{\text{sample}}$ **do**
  $\mathbf{M}_{ij} \leftarrow \|\mathbf{f}_i - \mathbf{f}_j\|_2$ ▷ Compute distance matrix
end for
$\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_K\} \leftarrow \text{HierCluster}(\mathbf{M}, \mathcal{D}_{\text{sample}})$
for each cluster $\mathbf{S}_k$ **do**
  $\mathbf{o}_k \leftarrow \frac{1}{|\mathbf{S}_k|} \sum_{\mathbf{x}_i \in \mathbf{S}_k} \mathcal{M}(\mathbf{x}_i)$ ▷ Prototype of $\mathbf{S}_k$
end for
for each data point $\mathbf{x}_i \in \mathcal{D}_s \setminus \mathcal{D}_{\text{sample}}$ **do**
  $\mathbf{f}_i \leftarrow \mathcal{M}(\mathbf{x}_i)$ ▷ Extract feature of $\mathbf{x}_i$
  $k \leftarrow \arg\min_k \|\mathbf{f}_i - \mathbf{o}_k\|_2$
  Add $\mathbf{x}_i$ to cluster $\mathbf{S}_k$
end for
return $\mathbf{S}_1, ..., \mathbf{S}_K$, $\mathbf{o}_1, ..., \mathbf{o}_K$

Algorithm 1 details the adaptive clustering process. Initially, we sample a subset $\mathcal{D}_{\text{sample}}$ from the private dataset $\mathcal{D}_s$. For each data point $\mathbf{x}_i \in \mathcal{D}_{\text{sample}}$, we extract features $\mathbf{f}_i = \mathcal{M}(\mathbf{x}_i)$ using the pre-trained backbone $\mathcal{M}$. These features are then clustered into $\mathbf{S}_1, ..., \mathbf{S}_K$ using HierCluster (Appendix 1). For each cluster $\mathbf{S}_k$, we compute a prototype $\mathbf{o}_k$ that represents the characteristic features of the cluster:

$$\mathbf{o}_k = \frac{1}{|\mathbf{S}_k|} \sum_{\mathbf{x}_i \in \mathbf{S}_k} \mathcal{M}(\mathbf{x}_i). \tag{3}$$

The remaining data points $\mathbf{x}_i \in \mathcal{D}_s \setminus \mathcal{D}_{\text{sample}}$ are then assigned to the cluster with the nearest prototype:

$$k_i^* = \arg\min_k \|\mathbf{f}_i - \mathbf{o}_k\|_2. \tag{4}$$

This sampling approach reduces clustering costs and enhances unlearning efficiency, as re-clustering is only required when unlearning sampled data.

By grouping similar data, cluster-specific prompts can detect nuanced differences, significantly enhancing the model's feature recognition and pattern analysis capabilities. Moreover, this clustering strategy reduces unlearning computational costs, as unlearning requests

only require re-optimization of the affected cluster's prompt and classifier head.

4.2.2 Prompt Tuning for Clustered Data

After clustering, each cluster $\mathbf{S}_k$ is assigned a randomly initialized pixel-level border prompt $\mathbf{p}_k$ first. These prompts are then tuned using the private training data specific to each cluster $\mathbf{S}_k$ to optimize performance.

These prompts serve as learnable transformations on the private data:

$$\tilde{\mathbf{x}_i} = g(\mathbf{x}_i, \mathbf{p}) = \mathbf{x}_i + \mathbf{p}_k, \tag{5}$$

where $\mathbf{x}_i$ is the original private data of cluster k_i^*, $\mathbf{p}_k$ is the prompt for cluster k_i^*, and $\tilde{\mathbf{x}_i}$ is the prompted data. As illustrated in Figure 2, these prompts are applied as border overlays to private data.

The prompt tuning process optimizes the prompts $\mathbf{p}_k$ and the classifier head $\mathbf{h}_k$ by minimizing cross-entropy loss:

$$\mathbf{p}_k^*, \mathbf{h}_k^* = \arg\min_{\mathbf{p}_k, \mathbf{h}_k} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{S}_k} L(H(\mathcal{M}(\mathbf{x}_i + \mathbf{p}_k; \theta); \mathbf{h}_k), y_i), \tag{6}$$

where $\mathcal{M}(\mathbf{x}_i + \mathbf{p}_k; \theta)$ represents the output of the pre-trained backbone $\mathcal{M}$ with parameters θ, and $H(\cdot; \mathbf{h}_k)$ is the classification function using the classifier head $\mathbf{h}_k$. L is the cross-entropy loss function, which measures the discrepancy between the predictions and the actual labels y_i.

Through iterative gradient descent, the prompt $\mathbf{p}_k$ adjusts the data distribution of cluster $\mathbf{S}_k$ to better align with the pre-trained backbone's learned patterns while each classifier head specializes in its cluster.

After prompts are well-optimized, we can use them for inference. During inference, the system first computes the nearest cluster k for a given input, applies the corresponding optimized prompt $\mathbf{p}_k^*$ to modify the input, and feeds this prompted input into the backbone to get the prediction.

4.3 Prompt Tuning-based Unlearning

Built on the above adaptive prompt tuning design, LMEraser enables efficient large model unlearning via targeted prompt re-tuning.

4.3.1 The Unlearning Process

When unlearning a data point $(\mathbf{x}_r, y_r)$ from cluster $\mathbf{S}_k$, only the prompt $\mathbf{p}_k$ and classifier head $\mathbf{h}_k$ of the affected cluster $\mathbf{S}_k$ are re-tuned. Formally, we first update the affected cluster to $\mathbf{S}_k' = \mathbf{S}_k \setminus (\mathbf{x}_r, y_r)$ by excluding the data $(\mathbf{x}_r, y_r)$. The prompt $\mathbf{p}_k$ and classifier head $\mathbf{h}_k$ are re-optimized for the updated cluster $\mathbf{S}_k'$, as shown in Eq. (7).

$$\mathbf{p}_k^{*\prime}, \mathbf{h}_k^{*\prime} = \arg\min_{\mathbf{p}_k, \mathbf{h}_k} \sum_{(\mathbf{x}_i, y_i) \in \mathbf{S}_k'} L(H(\mathcal{M}(\mathbf{x}_i + \mathbf{p}_k; \theta); \mathbf{h}_k), y_i) \tag{7}$$

where $\mathcal{M}(\cdot; \theta)$ is the frozen backbone with parameters θ, $H(\cdot; \mathbf{h}_k)$ is the classification operation using the classifier head $\mathbf{h}_k$, and L is the loss function.

Our targeted unlearning method offers three significant advantages. First, cluster-specific prompt optimization isolates private data within its cluster, eliminating the need to trace changes across the entire model. Second, optimizing small-size prompts and classifier heads significantly reduces computational costs and enhances unlearning efficiency. Third, combining pixel-level border prompts with a frozen pre-trained backbone maintains flexibility across architectures and preserves learned representations, effectively minimizing performance degradation and catastrophic forgetting risks.

4.3.2 Exact Unlearning Guarantees

LMEraser guarantees exact unlearning at the algorithmic level. While Membership Inference Attacks (MIA) are sometimes suggested for verifying unlearning, they face inherent limitations. First, when identical data points $\mathbf{x}_1, \mathbf{x}_2$ exist in the training dataset, successful unlearning of $\mathbf{x}_1$ would still trigger MIA detection due to $\mathbf{x}_2$'s legitimate presence in the training dataset. This false positive does not indicate unlearning failure, as the influence of $\mathbf{x}_1$ has been properly removed despite MIA's detection. On the other hand, the existence of forging methods (Thudi et al., 2022) means a model can be manipulated to become MIA-undetectable even when the target data's influence remains, creating false negatives in unlearning verification. Therefore, MIA cannot serve as a reliable metric for verifying exact unlearning. Instead, we focus on providing rigorous algorithmic-level guarantees through LMEraser's design.

LMEraser achieves exact unlearning through its algorithmic design with three key theoretical properties.

Property 1 (Data Influence Isolation) *The backbone-prompt architecture enforces strict separation:*

$$\mathcal{D} = \mathcal{D}_p \cup \mathcal{D}_s, \quad \mathcal{D}_p \cap \mathcal{D}_s = \emptyset \tag{8}$$

where $\mathcal{D}_p$ influences only backbone parameters θ, while $\mathcal{D}_s$ affects only prompts $\mathbf{p}_k$ and heads $\mathbf{h}_k$. This separation ensures that unlearning operations on private data cannot affect the backbone's learned representations.

Property 2 (Cluster Independence) *The adaptive clustering mechanism ensures that updates to one*

cluster's prompt and classifier head cannot affect other clusters' behaviors. For any two clusters i, j with their respective $(\mathbf{p}_i, \mathbf{h}_i)$ and $(\mathbf{p}_j, \mathbf{h}_j)$, modifying cluster i's prompt and classifier head has no impact on cluster j's decision boundaries or feature representations.

Property 3 (Complete Removal) *Complete retraining of affected prompt and classifier head guarantees removal of all direct and indirect influences of the unlearned data point, as the new optimized parameters $\mathbf{p}_k^{*\prime}, \mathbf{h}_k^{*\prime}$ are derived solely from $\mathbf{S}_k'$.*

These properties collectively ensure that LMEraser achieves exact unlearning at the algorithmic level, providing stronger guarantees than other verification methods.

5 EVALUATION

Our evaluation focuses on two aspects: *Model Utility* and *Unlearning Efficiency*. Model utility is measured by test accuracy on image classification tasks, verifying that the model's core functionality remains intact. Unlearning efficiency is measured by the time, computational, and storage costs required for unlearning requests, showing LMEraser's real-world applicability.

5.1 Experimental Setup

Private Datasets. We employ five diverse datasets as private datasets: HAM10000 (Tschandl et al., 2018), CIFAR10 (Krizhevsky and Hinton, 2009), CIFAR100 (Krizhevsky and Hinton, 2009), GTSRB (Stallkamp et al., 2012), and SVHN (Netzer et al., 2011). These datasets are selected due to their varying image sizes, number of images, class distributions, and degrees of diversity, providing a robust evaluation of task complexity. Details of the private datasets are summarized in Table 1.

Table 1: Statistical details of private datasets.

Dataset	Image Dimension	# Images	# Classes
HAM10000	$600 \times 450 \times 3$	10,015	7
CIFAR10	$32 \times 32 \times 3$	60,000	10
CIFAR100	$32 \times 32 \times 3$	60,000	100
SVHN	$32 \times 32 \times 3$	99,289	10
GTSRB	from $15 \times 15 \times 3$ to $250 \times 250 \times 3$	39,270	43

Public Datasets. We utilized ImageNet-1K and ImageNet-22K as public datasets. These public datasets provide a comprehensive foundation for our model backbones, ensuring a broad representation of general image features. Detailed specifications of public datasets are summarized in Table 2.

Table 2: Statistical details of public datasets.

Dataset	Image Dimension	# Images	# Classes
ImageNet-1K	$224 \times 224 \times 3$	1.28 M	1,000
ImageNet-22K	$224 \times 224 \times 3$	14 M	22,000

Pre-trained Backbones. We primarily use ViT-B-22K as the backbone, pre-trained on ImageNet-22K with 86M parameters. We also employ ViT-B-1K (Dosovitskiy et al., 2020) and Swin-B-1K (88M parameters) (Liu et al., 2021), which are pre-trained on ImageNet-1K dataset, as well as Swin-B-22K, which is pre-trained on ImageNet-22K (Deng et al., 2009).

Baselines. We compare LMEraser with retraining and state-of-the-art exact unlearning solution SISA. Due to the scarcity of exact unlearning methods for large models, we also developed Single-Prompt Unlearning and DAM-VP Unlearning for comparison. These adaptations help assess the effectiveness of modifying existing prompt tuning methods for unlearning.

- Retraining: The entire large model is retrained from scratch for each unlearning request.
- SISA (Bourtoule et al., 2021): SISA partitions training data into shards, with each shard used to train a separate model. Final predictions are made through majority voting across these models. For unlearning, only the affected shard model is retrained.
- Single-Prompt Unlearning: This method uses a single prompt and a single classifier head for private data, with a frozen backbone pre-trained on public data. The prompt and classifier head are retrained for each unlearning request. It is an adaptation of Visual Prompting (VP) (Bahng et al., 2022).
- DAM-VP Unlearning: Based on DAM-VP (Huang et al., 2023), this method clusters private data by diversity and generates prompts to each cluster with no classifier head. The affected prompt is retrained for each unlearning request.

Implementation Details. Experiments were conducted on eight Nvidia Tesla V100-FHHL GPUs, each with 16 GB memory, complemented by an Intel Xeon CPU E5-2650 v4 @ 2.20GHz with 48 cores and 251GB of RAM. The software environments included Ubuntu 22.04.3 LTS (64-bit), PyTorch v2.1.2, CUDA 12.1, and Python 3.11.5. Our code is available at: `https://github.com/lmeraser/lmeraser`.

We partitioned private datasets using hierarchical clustering (Müllner, 2011) (details in Appendix 1). Our default configuration involved sampling $1,000$ data

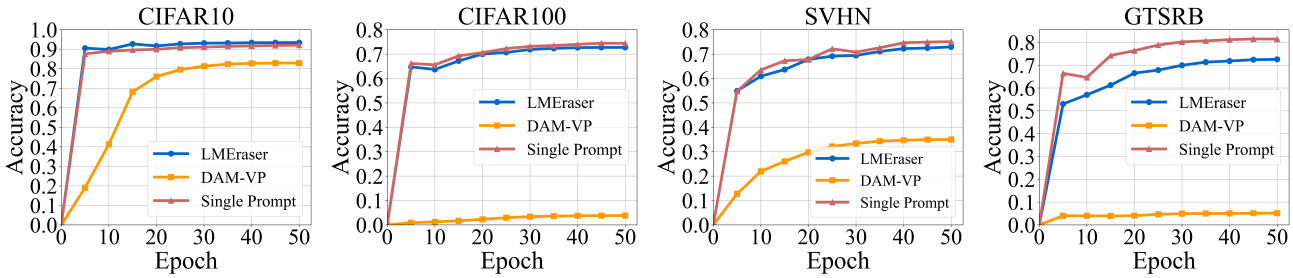


Figure 3: Classification accuracy comparison: LMEraser and baseline methods on ViT-B-22K.

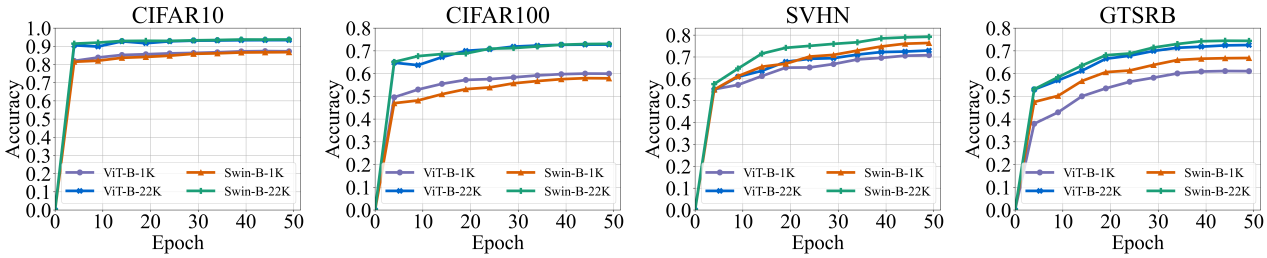


Figure 4: Test accuracy of LMEraser using different backbones.

points from each private dataset and setting the distance threshold to 10. This resulted in varying numbers of clusters across datasets: 194 for CIFAR10, 347 for CIFAR100, 30 for SVHN, and 90 for GTSRB. The number of clusters corresponds to the prompts used per dataset. Private data were standardized by resizing to 256×256 pixels and then cropping to 224×224 pixels. We applied a 30-pixel border prompt to mask each image in private datasets, consistent with methods used in DAM-VP (Huang et al., 2023).

5.2 Evaluation of Model Utility

Comparing Classification Accuracy. To validate the utility of LMEraser, we compared LMEraser's test accuracy against baseline methods using ViT-B-22K as the backbone. Figure 3 shows LMEraser achieves comparable performance to the single prompt method while outperforming DAM-VP. DAM-VP adopts a head-missing approach based on activation-feature to label mapping, falling short in tasks requiring precise classification. In contrast, LMEraser's use of cluster-specific prompts and classifier heads allows for more nuanced feature recognition, resulting in superior performance. The single prompt method, while effective, requires a long time to handle an unlearning request (Table 3). LMEraser better balances accuracy and computational efficiency.

Robustness across Different Backbones. To evaluate LMEraser's adaptability, we assessed its test accuracy using various backbone architectures while maintaining consistent conditions and prompt numbers for each dataset. Figure 4 demonstrates LMEraser's high accuracy across diverse datasets and backbones, highlighting its versatility. Models pre-trained on ImageNet-22K (e.g., Swin-B-22K and ViT-B-22K) consistently outperform their ImageNet-1K counterparts, highlighting the importance of pre-training data quality.

To further verify LMEraser's effectiveness on high-resolution domain-specific tasks, we conducted experiments on the HAM10000 medical dataset, which contains 10,015 dermatoscopic images (600×450 pixels) across seven categories of skin lesions. Figure 5 shows that LMEraser maintains strong performance on this challenging medical domain, achieving accuracy of 0.7910 (ViT-B-1K), 0.7946 (ViT-B-22K), 0.7872 (Swin-B-1K), and 0.8142 (Swin-B-22K) after convergence. These results demonstrate that our method can effectively handle basic vision tasks and high-resolution domain-specific applications.

Resilience to Data Removal. We used different unlearning ratios to evaluate the impact of data removal on model stability and robustness. The unlearning ratio is the percentage of data points requested to be removed from the private dataset for unlearning. We increased the unlearning ratio from 0% to 50% in 10% increments. At each increment, we randomly selected and removed the corresponding percentage of data points

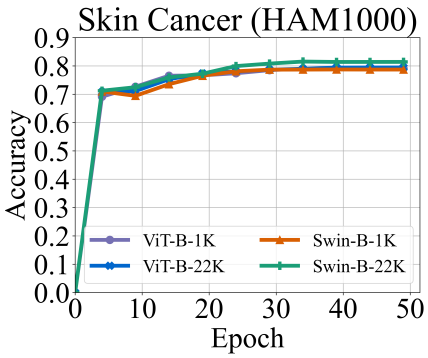


Figure 5: Test accuracy of LMEraser on HAM10000 medical dataset using different backbones.

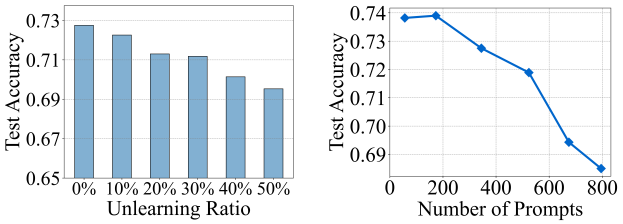


Figure 6: LMEraser's test accuracy with various unlearning privacy data ratios and number of prompts (with ViT-B-22K backbone on CIFAR100 private dataset).

from the private dataset, then retrained and tested the accuracy. Figure 6 (Left) shows a marginal decrease in test accuracy from 0.7275 to 0.6953(4.43% reduction) despite removing 50% of private data, demonstrating LMEraser's resilience to data removal.

Impact of Prompt Quantity. We analyzed the relationship between prompt quantity and model performance by adjusting the clustering distance threshold from 7 to 11, varying the number of prompts accordingly. Figure 6 (Right) shows that increasing the number of prompts leads to a moderate decrease in test accuracy because a larger number of prompts may reduce generalization and lead to overfitting.

5.3 Evaluation of Unlearning Efficiency

We evaluated LMEraser's unlearning efficiency based on three key metrics: size of affected training data, number of affected model parameters, and unlearning time. We also provide a theoretical analysis of batch unlearning.

Comparing Affected Training Data and Model Size. To assess the resource efficiency of LMEraser, we compared the number of affected training data points and affected model parameters using CIFAR10

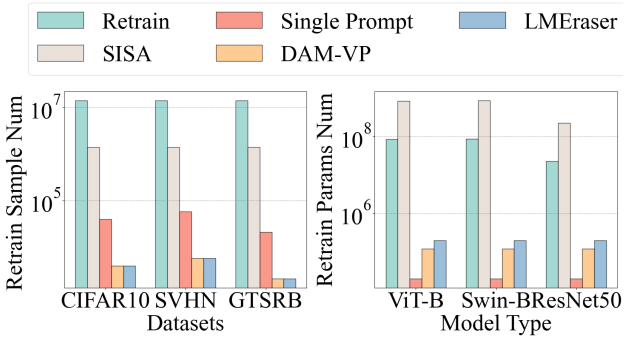


Figure 7: Comparative analysis of LMEraser and baseline methods in terms of affected training data and model size for a single unlearning request.

Table 3: Comparison of the average time required for handling an unlearning request with ViT-B-22K backbone across different datasets and methods.

Time	Retrain	SISA	SinglePrompt	DAM-VP	LMEraser
CIFAR10	Days	Days	3185s	21.26s	27.61s
CIFAR100	Days	Days	4048s	12.14s	12.45s
SVHN	Days	Days	4688s	260.07s	213.81s
GTSRB	Days	Days	1740s	26.64s	27.73s

for each unlearning request across methods, fixing the number of clusters at ten for all datasets. Figure 7 (Left) shows LMEraser and other methods that use prompt tuning architectures significantly reduce affected training data points. LMEraser affects approximately 100 times fewer training data points compared to traditional retraining and SISA. Figure 7 (Right) shows LMEraser has a more dramatic improvement, affecting approximately 1,000 times fewer model parameters compared to retraining and SISA.

Time Efficiency in Unlearning. We compare the average time required to handle an unlearning request across different methods.

As shown in Table 3, traditional methods such as retraining and SISA require days to process a single unlearning request on large backbones. In contrast, prompt-based methods like LMEraser and DAM-VP show great efficiency advantages. The design of parallel prompts significantly reduces the time from days to seconds or minutes for each request.

Theoretical Analysis of Batch Unlearning. We develop a theoretical analysis of LMEraser's performance in batch unlearning scenarios, with full derivations in Appendix 2. Our framework models unlearning requests as a Poisson process with rate λ. Consider a model with K clusters, each containing m_i data points ($i \in 1, \ldots, K$), and a total of $N = \sum_{i=1}^{K} m_i$ data points. T represents the batch collection period. We prove that the expected number of affected clusters in a time in-

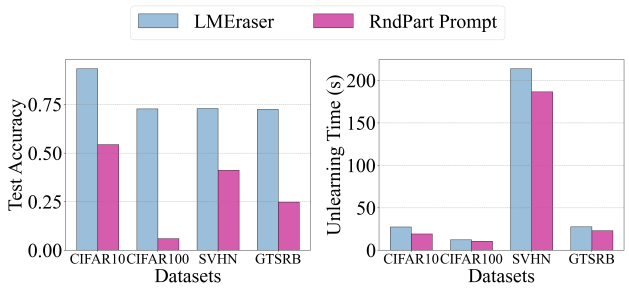


Figure 8: Performance comparison between random partitioning and LMEraser's adaptive clustering across different datasets.

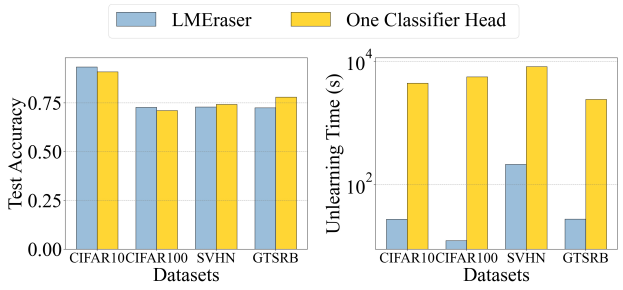


Figure 9: Performance comparison between one classifier head and LMEraser's multiple classifier heads across different datasets.

terval $[0, T]$ is:

$$\mathbb{E}[K_{\text{affected}}] = \sum_{i=1}^{K} \left(1 - e^{-\frac{m_i}{N}\lambda T}\right). \tag{9}$$

We demonstrate that for small values of $\frac{m_i}{N}\lambda T$ (i.e., when unlearning requests are infrequent), this expression simplifies to:

$$\mathbb{E}[K_{\text{affected}}] \approx \lambda T. \tag{10}$$

This approximation reveals that the expected number of affected clusters becomes independent of cluster size distribution, depending solely on the unlearning request rate λ and time interval T.

Furthermore, we derive the expected processing time for LMEraser as $\mathbb{E}[T_{\text{LMEraser}}] = T_c\left(1 - e^{-\lambda T}\right)$, where T_c represents the time required to re-tune a single cluster-specific prompt and classifier head. In contrast, for single prompt methods, the expected processing time is $\mathbb{E}[T_{\text{Single}}] = T_s\left(1 - e^{-\lambda T}\right)$, with T_s is the time to re-tune over the entire private dataset. Consequently, the efficiency gain of LMEraser is quantified by the ratio:

$$\text{Efficiency Gain} = \frac{\mathbb{E}[T_{\text{Single}}]}{\mathbb{E}[T_{\text{LMEraser}}]} = \frac{T_s}{T_c}. \tag{11}$$

Given that re-tuning only a subset of the dataset significantly reduces computational workload, T_c is much less than T_s. Consequently, LMEraser can achieve substantial efficiency gains in batch unlearning scenarios.

5.4 Ablation Studies

Adaptive Prompt Tuning. To evaluate the effectiveness of adaptive clustering, we compared LMEraser with a random partitioning approach (RndPart Prompt), wherein private data is randomly partitioned before prompt tuning while other conditions remain constant. Figure 8 shows similar unlearning times for both methods, but RndPart Prompt demonstrates significantly lower test accuracy, especially in complex tasks like CIFAR100. LMEraser's adaptive clustering captures inherent data characteristics, enabling more accurate prompt tuning.

Multiple Classifier Heads. To evaluate the effectiveness of multiple classifier heads, we compared LMEraser with a one classifier head method while other conditions remained constant. Figure 9 shows that both methods achieve similar test accuracy across datasets, but LMEraser requires much less unlearning time. The one classifier head method requires retraining the head for each unlearning request, consuming more time and resources. In contrast, LMEraser's multiple classifier heads method enables targeted retraining of only the affected heads, significantly improving unlearning efficiency.

6 CONCLUSION

LMEraser is an exact unlearning method specifically designed for large vision models. The prompt tuning architecture effectively isolates the influence of private data. Its adaptive prompt tuning method balances unlearning efficiency and model utility. Our comprehensive experiments demonstrate LMEraser's capability in achieving efficient unlearning without compromising accuracy, showing its adaptability across various datasets and large model architectures.

ACKNOWLEDGEMENTS

The authors sincerely thank the reviewers for their invaluable feedback. This work was supported in part by the Research Grants Council of Hong Kong under RIF (Research Impact Fund) R1012-21, R6021-20F, RFS2122-1S04, GRF grant (CityU 11211422), C2004-21G, C1029-22G, C6015-23G, and N_CityU139/21, and in part by the Innovation and Technology Commission of Hong Kong (ITC) under Mainland-Hong Kong Joint Funding Scheme (MHKJFS) under Grant MHP/135/23. This work was also supported by the InnoHK initiative,

The Government of the HKSAR, and the Laboratory for AI-Powered Financial Technologies (AIFT).

References

Bahng, H., Jahanian, A., Sankaranarayanan, S., and Isola, P. (2022). Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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.

Bowman, B., Achille, A., Zancato, L., Trager, M., Perera, P., Paolini, G., and Soatto, S. (2023). À-la-carte Prompt Tuning (APT): Combining Distinct Data Via Composable Prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14984–14993.

Brophy, J. and Lowd, D. (2020). Machine Unlearning for Random Forests. In *International Conference on Machine Learning*, pages 1092–1104.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Chen, R., Yang, J., Xiong, H., Bai, J., Hu, T., Hao, J., Feng, Y., Zhou, J. T., Wu, J., and Liu, Z. (2024). Fast model debias with machine unlearning. *Advances in Neural Information Processing Systems*, 36:14516–14539.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., et al. (2023). Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Ginart, A. A., Guan, M. Y., Valiant, G., and Zou, J. (2019). Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 1–14.

Golatkar, A., Achille, A., Ravichandran, A., Polito, M., and Soatto, S. (2021). Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 792–801.

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

Golatkar, A., Achille, A., and Soatto, S. (2020b). Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pages 383–398. Springer.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. (2020). Certified data removal from machine learning models. In *International Conference on Machine Learning*, pages 3832–3842. PMLR.

Huang, Q., Dong, X., Chen, D., Zhang, W., Wang, F., Hua, G., and Yu, N. (2023). Diversity-aware meta visual prompting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10878–10887.

Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. (2022). Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *Technical report, University of Toronto*.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. (2024). Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36:1957–1987.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Li, Y., Chen, C., Zhang, Y., Liu, W., Lyu, L., Zheng, X., Meng, D., and Wang, J. (2024). Ultrare: En-

hancing receraser for recommendation unlearning via error decomposition. *Advances in Neural Information Processing Systems*, 36.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European Conference on Computer Vision*, pages 740–755.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Liu, Y., Xu, L., Yuan, X., Wang, C., and Li, B. (2022). The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1749–1758. IEEE.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011.

Office of the Attorney General, S. o. C. D. o. J. (2023). California Consumer Privacy Act (CCPA). `https://oag.ca.gov/privacy/ccpa`.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Shibata, T., Irie, G., Ikami, D., and Mitsuzumi, Y. (2021). Learning with Selective Forgetting. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 989–996.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332.

Su, N. and Li, B. (2023). Asynchronous federated unlearning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE.

Suriyakumar, V. and Wilson, A. C. (2022). Algorithms that approximate data removal: New results and limitations. *Advances in Neural Information Processing Systems*, 35:18892–18903.

Tanno, R., F Pradier, M., Nori, A., and Li, Y. (2022). Repairing neural networks by leaving the right past behind. *Advances in Neural Information Processing Systems*, 35:13132–13145.

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.

Tschandl, P., Rosendahl, C., and Kittler, H. (2018). The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Voigt, P. and Von dem Bussche, A. (2017). *The EU General Data Protection Regulation (GDPR). A Practical Guide*. Springer International Publishing.

Wang, J., Guo, S., Xie, X., and Qi, H. (2022). Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM Web Conference 2022*, pages 622–632.

Warnecke, A., Pirch, L., Wressnegger, C., and Rieck, K. (2023). Machine unlearning of features and labels. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society.

Wu, G., Hashemi, M., and Srinivasa, C. (2022). Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8675–8682.

Wu, J., Yang, Y., Qian, Y., Sui, Y., Wang, X., and He, X. (2023a). Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, pages 651–661.

Wu, Y., Wen, R., Backes, M., Berrang, P., Humbert, M., Shen, Y., and Zhang, Y. (2023b). Quantifying privacy risks of prompts in visual prompt learning. *arXiv preprint arXiv:2310.11970*.

Xu, J., Wu, Z., Wang, C., and Jia, X. (2024). Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–19.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595.

Zhou, K., Liu, Z., Qiao, Y., Xiang, T., and Loy, C. C. (2022). Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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]
   Justification: Detailed descriptions of LMEraser's architecture, including adaptive prompt tuning and clustering algorithms, are provided in Sections 1 and 5.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   Justification: Time complexity analysis for clustering and unlearning operations is included in Sections 5.2 and 5.3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
   Justification: Please see our GitHub repository `https://github.com/lmeraser/lmeraser`.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   Justification: Assumptions for the theoretical analysis of batch unlearning are provided in Appendix 2.
   (b) Complete proofs of all theoretical results. [Yes]
   Justification: Derivations for the theoretical analysis of batch unlearning are included in Appendix 2.
   (c) Clear explanations of any assumptions. [Yes]
   Justification: Assumptions are explained in detail in Appendix 2.

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]
   Justification: Please see our GitHub repository `https://github.com/lmeraser/lmeraser`.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   Justification: Please see our GitHub repository.
   (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]
   Justification: Specific measures (e.g., test accuracy) are defined in Section 5.2 and our GitHub repository.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
   Justification: The hardware and software environment used for experiments is described in Section 5 and our GitHub repository.

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]
   Justification: Cited in Section 5.
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (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]
   Justification: The research does not involve crowdsourcing or human subjects.

LMEraser: Large Model Unlearning via Adaptive Prompt Tuning: Supplementary Materials

1 HIERCLUSTER AND PERFORMANCE ANALYSIS

HierCluster is an efficient and adaptive algorithm designed for large-scale data applications. It strategically groups similar data points into distinct clusters, enhancing the model's ability to detect subtle differences within each cluster. Unlike traditional clustering methods, HierCluster dynamically determines the number of clusters based on the data's inherent characteristics, eliminating the need for predefined cluster quantities. Moreover, it supports the identification of clusters with arbitrary shapes, adapting naturally to the underlying structure of the data.

Algorithm 2 HierCluster

Input: Dataset $\mathcal{D}$ and distance matrix $\mathbf{M}$
Output: Clusters $\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_K\}$

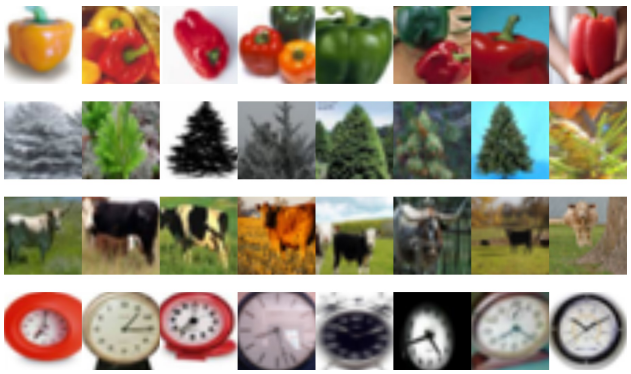
Initialize each $\mathbf{x}_i \in \mathcal{D}$ as a cluster $\mathbf{S}_i = \{\mathbf{x}_i\}$
$\mathbf{S} \leftarrow \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n\}$ ▷ Set of clusters
while True **do**
  $minDist \leftarrow \infty$
  $toMerge \leftarrow (null, null)$
  for each cluster pair $(\mathbf{S}_i, \mathbf{S}_j)$ in $\mathbf{S}$ **do**
    $dist \leftarrow$ Average linkage distance between $\mathbf{S}_i$ and $\mathbf{S}_j$ using $\mathbf{M}$
    if $dist < minDist$ **then**
      $minDist \leftarrow dist$
      $toMerge \leftarrow (\mathbf{S}_i, \mathbf{S}_j)$
    end if
  end for
  if $minDist > t$ or $|\mathbf{S}| == 1$ **then**
    ▷ Distance threshold t
    break
  end if
  $\mathbf{S}_{new} \leftarrow$ Merge clusters in $toMerge$
  Update $\mathbf{S}$ by removing merged clusters and adding $\mathbf{S}_{new}$
end while
return $\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_K\}$

The HierCluster process operates through an iterative merging mechanism, as detailed in Algorithm 2. It begins by treating each data point $\mathbf{x}_i$ in the dataset as a separate cluster $\mathbf{S}_i$. As the process advances, the algorithm utilizes a distance matrix $\mathbf{M}$ that stores all pairwise distances, merging the two closest clusters in each iteration. The merging process continues until the distance between any remaining clusters exceeds the threshold t or when all data points have been merged into a single cluster. Formally, we can define the merging criterion as:

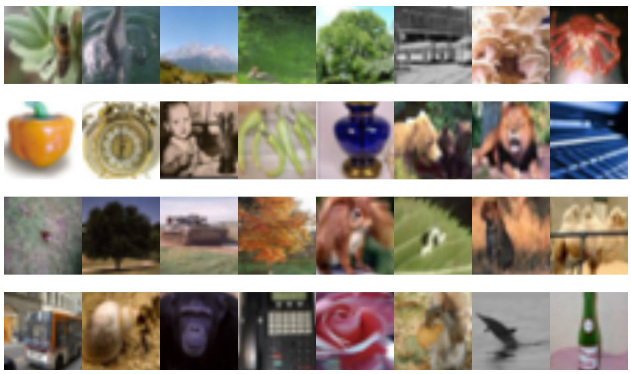
$$\text{Merge}(\mathbf{S}_i, \mathbf{S}_j) \quad \text{if} \quad d(\mathbf{S}_i, \mathbf{S}_j) < t, \tag{12}$$

where $d(\mathbf{S}_i, \mathbf{S}_j)$ is the average linkage distance between clusters $\mathbf{S}_i$ and $\mathbf{S}_j$.

The adaptability of HierCluster is particularly advantageous for handling datasets of varying sizes and complexity. It ensures minimal disruption to model performance while facilitating precise and targeted data learning and removal. The threshold t is adaptively set according to the desired granularity of the clustering, ensuring that



(a) Visualization of HierCluster clustering on CIFAR100. Each row represents a different cluster, showcasing images grouped based on their diversity.



(b) Visualization of random partitioning results on CIFAR100. Each row displays images from one group, contrasting with HierCluster results.

Figure 10: Comparison between HierCluster clustering (left) and random partitioning (right) on CIFAR100 dataset.

the algorithm effectively captures the dataset's diversity. This adaptive thresholding ensures that the clustering process can handle datasets of varying sizes and complexities, making it particularly suitable for the diverse nature of private datasets.

Our evaluation of HierCluster demonstrates its efficiency and clustering quality. Under our experiment setup, clustering a sample of 1000 data points from a private dataset was consistently completed in less than 3 seconds. This represents approximately 0.1% of the total time required for 50 epochs of model training, making the clustering time negligible in the overall process.

We visually assess the quality of clustering through a comparative analysis. Figure 10a presents a visualization of HierCluster results on CIFAR100, while Figure 10b shows comparative results from random partitioning.

As shown in Figure 10a, HierCluster effectively forms homogeneous clusters while ensuring distinct separation between them. This grouping allows LMEraser to learn more nuanced features within each cluster, potentially enhancing the accuracy of image classification tasks. In contrast, Figure 10b demonstrates that random partitioning lacks such homogeneity and clear separation.

The contrast between these clustering results underscores the advantages of HierCluster in effectively grouping similar data points. This efficient and effective clustering enables LMEraser to learn and utilize the intricate diversity of each cluster, contributing to the overall performance.

2 THEORETICAL ANALYSIS OF BATCH UNLEARNING PERFORMANCE

In practical applications, model owners may not perform unlearning operations immediately upon receiving individual requests due to efficiency and resource utilization concerns. Instead, they often collect unlearning requests over a period of time and perform batch unlearning operations. Batch unlearning optimizes resource usage but introduces a trade-off between immediacy and efficiency. In this section, we present a theoretical analysis to quantify the performance of LMEraser in such batch unlearning scenarios, comparing it to single prompt methods, while accounting for parallel processing capabilities.

2.1 Probabilistic Model for Unlearning Requests

Consider a situation with K clusters, each containing m_i data points, where $i \in 1, \ldots, K$. Let $N = \sum_{i=1}^{K} m_i$ be the total number of data points. We assume unlearning requests follow a Poisson process with a total rate λ for the entire dataset, and the probability of a request targeting a specific data point is uniform across all data points. The batch collection period is T.

Since unlearning requests are uniformly distributed among all data points as assumed, the rate at which cluster i

receives unlearning requests is proportional to its size:

$$\lambda_i = \lambda \cdot \frac{m_i}{N}. \tag{13}$$

2.2 Analysis of Affected Clusters

Expected Number of Affected Clusters The probability that a cluster i receives at least one unlearning request during the time interval $[0, T]$ is given by:

$$P_i = 1 - e^{-\lambda_i T} = 1 - e^{-\frac{m_i}{N}\lambda T}. \tag{14}$$

The expected number of affected clusters can be expressed as:

$$\begin{aligned}\mathbb{E}[K_{\text{affected}}] &= \sum_{i=1}^{K} 1 \cdot P_i = \sum_{i=1}^{K} \left(1 - e^{-\frac{m_i}{N}\lambda T}\right) \\ &\leq \sum_{i=1}^{K} \frac{m_i}{N}\lambda T = \lambda T\end{aligned} \tag{15}$$

where P_i is the probability that cluster i is affected.

Thus, the expected number of affected clusters is approximately bounded by λT, indicating that it is proportional to the total unlearning request rate λ and the time interval T. This upper bound demonstrates that, for small values of $\frac{m_i}{N}\lambda T$, the expected number of affected clusters will not exceed λT, regardless of the specific distribution of the cluster sizes.

Variance of the Number of Affected Clusters The variance in the number of affected clusters quantifies the spread or variability around the expected value. It is given by:

$$\text{Var}[K_{\text{affected}}] = \sum_{i=1}^{K} P_i \left(1 - P_i\right). \tag{16}$$

where P_i is the probability that cluster i is affected.

Using the probability expression from Eq. (14), we can rewrite the variance as:

$$\begin{aligned}\text{Var}[K_{\text{affected}}] &= \sum_{i=1}^{K} \left(1 - e^{-\frac{m_i}{N}\lambda T}\right) e^{-\frac{m_i}{N}\lambda T}. \\ &\leq \sum_{i=1}^{K} \frac{m_i}{N}\lambda T \cdot 1 = \lambda T.\end{aligned} \tag{17}$$

The variance of the number of affected clusters is thus bounded above by λT, indicating that the spread or variability in the number of affected clusters is proportional to the total unlearning request rate λ and the time interval T. This upper bound shows that, when unlearning requests are relatively infrequent (i.e., $\frac{m_i}{N}\lambda T$ is small for all clusters), the variance remains limited, reflecting a relatively tight concentration around the expected value.

2.3 Processing Time Analysis and Efficiency Gains

Expected Processing Time for LMEraser with Variable Cluster Sizes In LMEraser, re-tuning operations for different clusters can be executed in parallel due to the independence of clusters. Therefore, the total processing time depends on whether any clusters are affected and the time required to re-tune a single cluster's prompt and classifier head, denoted as T_c.

The probability that at least one cluster receives an unlearning request is:

$$\begin{aligned} &P(\text{any cluster affected}) = 1 - P(\text{no clusters affected}) \\ &= 1 - \prod_{i=1}^{K} (1 - P(\text{cluster } i \text{ affected})) \\ &= 1 - \prod_{i=1}^{K} e^{-\lambda_i T} = 1 - e^{-\sum_{i=1}^{K} \lambda_i T} \\ &= 1 - e^{-\lambda T}. \end{aligned} \tag{18}$$

Therefore, the expected processing time $\mathbb{E}[T_{\text{LMEraser}}]$ accounting for parallel processing is:

$$\begin{aligned} \mathbb{E}[T_{\text{LMEraser}}] &= T_c \cdot P(\text{any cluster affected}) \\ &= T_c \left(1 - e^{-\lambda T}\right). \end{aligned} \tag{19}$$

For small λT, we can approximate:

$$\mathbb{E}[T_{\text{LMEraser}}] \approx T_c \cdot \lambda T. \tag{20}$$

Expected Processing Time for Single Prompt Method In the single prompt method, any unlearning request necessitates re-tuning the prompt over the entire private dataset. The probability that at least one unlearning request occurs in the interval $[0, T]$ is:

$$P(\text{any unlearning}) = 1 - e^{-\lambda T}. \tag{21}$$

The expected processing time $\mathbb{E}[T_{\text{Single}}]$ is then:

$$\mathbb{E}[T_{\text{Single}}] = T_s \cdot P(\text{any unlearning}) = T_s \left(1 - e^{-\lambda T}\right), \tag{22}$$

where T_s is the time required to re-tune the single prompt over the entire private dataset.

For small λT, we can approximate:

$$\mathbb{E}[T_{\text{Single}}] \approx T_s \cdot \lambda T. \tag{23}$$

2.4 Time Efficiency Gain with Variable Cluster Sizes

The time efficiency gain of LMEraser over the single prompt method is given by the ratio of their expected processing times:

$$\text{Efficiency Gain} = \frac{\mathbb{E}[T_{\text{Single}}]}{\mathbb{E}[T_{\text{LMEraser}}]} = \frac{T_s}{T_c}. \tag{24}$$

This expression reveals that the efficiency gain is primarily determined by the ratio of the processing times per unlearning operation, T_s and T_c. Since $T_c \ll T_s$ due to the smaller size and scope of the re-tuning process in LMEraser, the efficiency gain can be substantial.

Our theoretical analysis demonstrates that LMEraser offers significant efficiency advantages in batch unlearning scenarios, even when data points are not uniformly distributed across clusters. This efficiency primarily arises from two key factors. First, LMEraser employs targeted re-tuning, where only the prompts and classifier heads of affected clusters are updated, and these clusters can be processed in parallel. Second, the expected processing time shows minimal dependence on cluster size distribution when unlearning requests are rare, ensuring consistent performance across various data distributions.

Beyond the computational advantage, LMEraser's parallel prompts architecture provides additional benefits that enhance its practical utility. First, the system's modularity and maintainability are significantly improved, as individual prompts can be updated independently. This feature simplifies maintenance processes and allows for flexible model updates without affecting the entire system. Furthermore, LMEraser has a scalable architecture that can seamlessly incorporate new prompts as data clusters evolve, ensuring the model remains adaptable to changing data landscapes. This scalability is important in real-world applications where data distributions may shift over time. Lastly, the isolation of prompts in LMEraser's architecture effectively reduces interference and mitigates the risk of catastrophic forgetting. By minimizing the impact of updates in one cluster on the performance of others, LMEraser maintains robust performance across all data domains, even as parts of the model are updated or removed.