Journal of Machine Learning Research 26 (2025) 1-33 Submitted 3/24; Revised 7/25; Published 10/25

Efficient Knowledge Deletion from Trained Models Through Layer-wise Partial Machine Unlearning

Vinay Chakravarthi Gogineni VIGO@MMMI.SDU.DK
Applied AI and Data Science Unit
The Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Camusvej 55, Odense M, 5230

Esmaeil S. Nadimi ESI@MMMI.SDU.DK
Applied AI and Data Science Unit
The Mærsk Mc-Kinney Møller Institute
University of Southern Denmark
Camusvej 55, Odense M, 5230

Editor: Kilian Weinberger

Abstract

Machine unlearning has garnered significant attention due to its ability to selectively erase knowledge obtained from specific training data samples in an already trained machine learning model. This capability enables data holders to adhere strictly to data protection regulations. However, existing unlearning techniques face practical constraints, often causing performance degradation, demanding brief fine-tuning post unlearning, and requiring significant storage. In response, this paper introduces a novel class of layer-wise partial machine unlearning algorithms that enable selective and controlled erasure of targeted knowledge. Of these, *partial amnesiac unlearning* integrates layer-wise selective pruning with the state-of-the-art amnesiac unlearning. This method selectively prunes and stores updates made to the model during training, enabling the targeted removal of specific data from the trained model. Other methods assimilates *layer-wise partial-updates* into label-flipping and optimization-based unlearning, thereby mitigating the adverse effects of specific knowledge deletion on model efficacy. Through a detailed experimental evaluation, we showcase the effectiveness of proposed unlearning methods. Experimental results highlight that the partial amnesiac unlearning not only preserves model efficacy but also eliminates the necessity for brief fine-tuning post unlearning, unlike conventional amnesiac unlearning. Further, employing layer-wise partial updates in label-flipping and optimization-based unlearning techniques demonstrates superiority in preserving model efficacy compared to their naive counterparts.

Keywords: Machine unlearning, approximate unlearning, amnesiac unlearning, layer-wise pruning, layer-wise partial-updates

1. Introduction

In the current data-centric era, data holders actively gather valuable information from various sources, encompassing personal information of individuals such as images, speech, text and medical records. Machine learning (ML) (Jordan and Mitchell, 2015; Simonyan and Zisserman, 2015b; LeCun et al., 2015) has orchestrated a profound transformation in navigating and analyzing the immense reservoir of data available to data holders. ML facilitates the extraction of valuable insights from these extensive and information-rich repositories, thereby enhancing user experiences and providing an overall improved service in numerous applications. Furthermore, in practical, real-world scenarios, data pipelines often demonstrate dynamic behavior, consistently acquiring fresh data. Life-long ML leverages this continuous inflow of information to dynamically update knowledge and fine-tune ML model parameters (Chen et al., 2018).

On the contrary, there are situations where the imperative for data deletion becomes obvious. Envision a scenario where a patient, who initially disclosed a specific medical condition with the healthcare system, later decides to retract this particular information due to privacy reasons. Similarly, users, attentive to privacy, may wish to erase their purchase history from an e-commerce platform, especially if they made unusual or atypical purchases during a specific period. Provisions like the *right to be forgotten* in general data protection regulation (GDPR) (European Parliament and Council of the European Union, 2016), empowers individuals to exert control over their personal data and imposes a legal obligation for the removal of such data when requested.

In situations where data has already been used to train an ML model, the mere act of deleting the data from data holder's storage is inadequate. This inadequacy stems from the inherent ability of ML models, particularly complex deep neural networks (DNNs), to potentially memorize intricate patterns present within the training data (Arpit et al., 2017; Zhang et al., 2017). These models can retain information in their parameters, inadvertently revealing information about the data they were trained on. This vulnerability is exploited through various attacks like the membership inference attack (Shokri et al., 2016; Choquette-Choo et al., 2021) and the model inversion attacks (Fredrikson et al., 2015; Struppek et al., 2022). Therefore, it is essential for a trained ML model to go through an unlearning process, purging the knowledge acquired from the data targeted for unlearning, often referred to as *targeted data*. This necessity prompts an exploration into the realm of *machine unlearning*; a process that selectively removes the influence of specific training data samples from an already trained ML model (Xu et al., 2023; Bourtoule et al., 2020). The objective of this process is to ensure that, after unlearning, the model demonstrates behavior akin to a model that has never been trained on the target data.

Machine unlearning proves invaluable not only in safeguarding privacy but also instrumental in enhancing model security and resilience against adversarial attacks, especially data-poisoning attacks (Steinhardt et al., 2017). In such attacks, adversaries inject carefully crafted malicious data into the training set to influence the model's behavior and undermine its trustworthiness. Machine unlearning serves as a defense mechanism against these attacks by selectively forgetting manipulated data, thus restoring the model's integrity. Additionally, machine unlearning enhances the adaptability of models in dynamic environments over time. It becomes indispensable in selectively discarding outdated knowledge acquired from obsolete data, enabling models to stay agile and relevant in evolving scenarios. This strategic forgetting of obsolete information is paramount, ensuring that the model remains robust and efficient.

Numerous studies in the literature have investigated the nascent field of machine unlearning. These efforts can be broadly categorized into two groups: *exact unlearning* and *approximate unlearning* methods (Eisenhofer et al., 2023). Exact unlearning methods completely erase the influence of a target data on existing ML model. This is achieved by excluding the target data samples from the training data set and then retraining the model. As unlearning requests persist, the need to retrain the model from scratch becomes resource-intensive, particularly due to the training of large-scale DNNs on data reservoirs. To tackle this challenge, several efficient procedures for swift retraining (Cao and Yang, 2015; Brophy and Lowd, 2021; Wu et al., 2020; Bourtoule et al., 2021; Neel et al., 2021) have been proposed. On the other hand, approximate unlearning methods entail modifying the existing model parameters in a way that the model behaves as if it has never seen the target data. Selective forgetting through scrubbing (Golatkar et al., 2019), linear filtration (Baumhauer et al., 2022), certified removal (Guo et al., 2020), and amnesiac unlearning (Graves et al., 2020) are a few examples of this kind. A subgroup of approximate unlearning methods perceives unlearning as an optimization problem (Jang et al., 2023; Warnecke et al., 2021; Tarun et al., 2021; Chundawat et al., 2022), modifying the existing model parameters by maximizing the loss related to the target data samples. Nevertheless, the majority of these unlearning works are compromised by one or more of the ensuing issues:

1. The majority of existing approximate unlearning techniques e.g., (Graves et al., 2020; Golatkar et al., 2019; Tarun et al., 2021), exhibit a degradation in model performance after handling an unlearning request. This degradation becomes more pronounced with an increase in the number of targeted data samples.

2. A few other existing unlearning techniques demand substantial storage capacity to facilitate the deletion of target data samples. This issue arises from the retention of storage-intensive models trained on data subsets (Bourtoule et al., 2021; Neel et al., 2021) or the storage of updates made to model parameters during training (Graves et al., 2020).

3. Most prevailing unlearning techniques make use of *retained data*, i.e., the original training data excluding the targeted data, during the unlearning process. Certain unlearning techniques, exemplified in (Graves et al., 2020; Tarun et al., 2021), involve a brief fine-tuning phase on the retained data for a limited number of iterations to restore model performance as it was before the unlearning. It is important to note that the fast yet effective machine unlearning (FMUL) method (Tarun et al., 2021) utilizes a subset of the retained data set during the unlearning procedure (in a noisy form), and also employs the original retained data during the repair step, which is a fine-tuning phase to restore model efficacy.

Our Contributions

This paper aims to address the aforementioned concerns while effectively erasing the impact of targeted data from an already trained model. Our contributions are as follows:

- We advocate for adopting layer-wise pruning in tandem with conventional amnesiac unlearning to efficiently erase the targeted data from an already trained model. The proposed method, termed *partial amnesiac unlearning*, selectively discards a portion of the updates made to the model layer-by-layer basis during the training phase and stores them for subsequent use in the unlearning phase. In the unlearning phase, subtracting the layer-wise pruned updates from the trained model efficiently erases the impact of target data while preserving the model's efficacy on retained data. A critical aspect of the proposed partial amnesiac unlearning is that it does not require any brief fine-tuning following the unlearning phase. Furthermore, storing the pruned updates significantly reduces storage space requirements in contrast to conventional amnesic unlearning (Graves et al., 2020).

- We propose to integrate *layer-wise partial-updates* (Godavarti and Hero, 2005) into label-flipping- and optimization-based unlearning. Employing layer-wise partial updates during the unlearning phase strategically mitigates the potential negative impact on model efficacy. This approach aims to maintain the model's effectiveness on retained data while efficiently erasing the impact of target data from the trained model.
- We conduct a comprehensive empirical assessment to demonstrate the effectiveness of the proposed unlearning methods, wherein the membership inference metric serves as a key performance indicator. Our analysis encompasses diverse data sets such as MNIST (LeCun et al., 2010), and street view house numbers (SVHN) (Netzer et al., 2011), CIFAR10 (Krizhevsky and Hinton, 2009) and medical MNIST (MEDMNIST) (Yang et al., 2022). Further, we explore various neural network architectures, including multilayer perceptron (MLP) with 2 fully connected layers, the well-known Lenet (LeCun et al., 1998), AlexNet (Krizhevsky et al., 2012), 11-layer and 19-layer visual geometry group networks (VGGs) (Simonyan and Zisserman, 2015a), 9-layer, 18-layer and 50-layer residual networks (ResNets) (He et al., 2016), as well as a simple Vision transformer with depth 4 (Simple ViT) and the large ViT model of depth 24 (Dosovitskiy et al., 2021). Our experimental results consistently highlight the superiority of the proposed methods over their conventional counterparts.

2. Related Work

A naive method to ensure information erasure involves discarding target data samples from the training data set and then retraining an ML model from scratch. However, initiating retraining from scratch for each unlearning request would demand a substantial amount of resources and time. To devise a more efficient alternative, various efforts have been made in the literature. The concept of unlearning was initially introduced in (Cao and Yang, 2015) to facilitate the forgetting of training data samples from trained ML model. This approach represents the learning algorithm in a summation form, where each summation is the sum of transformed data samples. To fulfill an unlearning request, this approach simply updates the summations, making it faster than retraining from scratch.

Further advancements include the data removal-enabled (DaRE) forests, a variant of random forests proposed in (Brophy and Lowd, 2021), which facilitates the removal of training data with minimal retraining. The DeltaGrad algorithm, presented in (Wu et al., 2020), facilitates the swift retraining of stochastic gradient-based machine learning algorithms when forgetting a limited set of data samples. DeltaGrad is applicable for strong convex and smooth objective functions. In (Bourtoule et al., 2021), a deterministic deletion procedure named sharded, isolated, sliced, and aggregated training (SISA) has been introduced. In the SISA framework, the entire data set undergoes random partitioning into K non-overlapping parts, commonly referred to as shards, which are further divided into slices. Each shard independently undergoes model training, and the resulting separate models are subsequently averaged to produce the final model. Notably, SISA mitigates the necessity for complete retraining, as the model can be specifically retrained on the shard where the deletion request has been initiated.

Despite its simple procedure, SISA's performance is greatly influenced by the number of shards and deletion requests, and the distribution of data within those shards. Further, SISA is susceptible to membership inference attacks, particularly when an adversary possesses the ability to observe both the model before and after the removal of a specific user's data point (Chen et al., 2021). This concern was addressed in (Neel et al., 2021) by implementing a differential privacy mechanism over the shard models before aggregating them to generate the final model. All these retraining techniques are termed as exact unlearning methods, as the resultant model has not been exposed to the target data.

Another category of unlearning algorithms, known as approximate unlearning methods, modifies the already trained model parameters to ensure that the model behaves as if it has never encountered the targeted data. Approximate unlearning methods circumvent the need for retraining, requiring fewer resources compared to exact unlearning methods. In (Golatkar et al., 2019), a quadratic scrubbing procedure based on Newton's update has been proposed. This method uses the identities of gradient and Hessian to selectively erase information from an already trained model. The linear filtration method proposed in (Baumhauer et al., 2022) applies a linear transformation to the logits of a classifier, designed to handle class-wide deletion requests in a computationally efficient manner. In (Guo et al., 2020), a certified data removal mechanism has been proposed based on Newton's update. This method ensures the removal of data up to the level of differential privacy guarantees.

The amnesiac unlearning method proposed in (Graves et al., 2020) stores the details of data samples presented in each batch, along with their corresponding contributions to the model during training. In the unlearning phase, the contributions made by the target data samples are subtracted from the trained model. In (Jang et al., 2023; Warnecke et al., 2021), the unlearning process involves retraining the already trained model for a limited number of steps on the target data. These strategic approaches aim to maximize the loss instead of minimizing it. Motivated by the error-minimizing noise framework (Huang et al., 2021), a method for generating error-maximization noise in the context of unlearning has been introduced in (Tarun et al., 2021). This method learns the noise that maximizes the loss on target data samples. In (Chundawat et al., 2022), error-minimization and maximization concepts are integrated for efficient unlearning. A novel adaptive machine unlearning approach leveraging attention mechanisms has been introduced in (Shaik et al., 2023). This approach computes attention scores for each data sample in the data set and incorporates these scores into the unlearning process. The exploration of unlearning concepts has been extended into the realm of graph data in (Chen et al., 2022). A preliminary extension of machine unlearning within federated network settings has been explored in (Liu et al., 2021; Wu et al., 2022; Sheng et al., 2024).

3. Preliminaries

In this section, we provide a concise overview of machine learning, delve into the intriguing concept of machine unlearning, and present a very brief overview of selective prominent machine unlearning algorithms relevant to our work. We systematically introduce the essential notation, paving the foundation for a clear presentation of the proposed algorithms. It is worth to emphasize that our focus throughout this paper is specifically on supervised learning.

3.1 Machine Learning

Consider a training data set $\mathcal{D}$ consists of N number of data pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$; where $\mathbf{x}_i$ is ith input data sample (e.g., an image) and y_i is its corresponding reference (e.g., class label). The objective in supervised learning is to identify the nonlinear function f, parametrized by a DNN model $\mathbf{w}$, which characterizes the relationship between input data samples $\mathbf{x}_i$ and their corresponding references y_i. This relationship is expressed mathematically as $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ represent the spaces encompassing all input data samples and references, respectively.

Learning the DNN model parameters $\mathbf{w}$ entails minimizing the empirical risk $\mathcal{L}(\mathcal{D}, \mathbf{w})$, measure how effectively the model $\mathbf{w}$ maps data samples $\mathbf{x}_i$ to their corresponding references $\mathbf{y}_i$, for $i = 1, 2, \ldots, N$. For example, in the context of classification task, cross-entropy is a widely recognized loss function, defined as:

$$\mathcal{L}(\mathcal{D}, \mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot \log(f(\mathbf{x}_i)_c), \tag{1}$$

where C denotes the number of classes and $y_{i,c} = 1$ if the true label for data sample $\mathbf{x}_i$, i.e., y_i is class c and 0 otherwise. The predicted probability of the DNN model for class c given the input $\mathbf{x}_i$ is denoted as $f(\mathbf{x}_i)_c$. The model parameters $\mathbf{w}$ can be learned recursively by following a steepest descent search on $\mathcal{L}(\mathcal{D}, \mathbf{w})$. The batch gradient descent recursion for updating the model parameters is:

$$\mathbf{w}_e = \mathbf{w}_{e-1} - \alpha\, \nabla_{\mathbf{w}}\mathcal{L}(\mathcal{D}, \mathbf{w})\Big|_{\mathbf{w}_{e-1}}, \tag{2}$$

where $\mathbf{w}_e$ and $\mathbf{w}_{e-1}$ are the model parameters from the current and previous epochs, respectively. $\nabla_{\mathbf{w}}\mathcal{L}(\mathcal{D}, \mathbf{w})$ is the gradient of the loss function with respect to model parameters $\mathbf{w}$ and α is the learning rate. To improve computational efficiency, it is often preferable to use minibatches of data $\mathcal{D}^b$, for $b = 1, 2, \ldots, B$, rather than the entire data set $\mathcal{D}$ during training. The minibatch gradient descent recursion for updating the model parameters during epoch e is:

$$\mathbf{w}_{e,b} = \mathbf{w}_{e,b-1} - \alpha\, \nabla_{\mathbf{w}}\mathcal{L}(\mathcal{D}^b, \mathbf{w})\Big|_{\mathbf{w}_{e,b-1}}, \tag{3}$$

with $\mathbf{w}_{e,0} = \mathbf{w}_{e-1}$. After processing the final minibatch with index B, the updated model parameters $\mathbf{w}_{e,B}$ initiate the subsequent epoch $e + 1$. This process continues until convergence is achieved.

3.2 Machine Unlearning

Consider a scenario where a DNN model, parameterized by $\mathbf{w}$, is trained on a data set $\mathcal{D}$. Let $\mathcal{D}_t$ be the targeted data set, which includes only the data samples designated for deletion. Additionally, let $\mathcal{D}_r$ is the retained data set, which is the original training data set with excluding targeted data, i.e., $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_t$, where $\setminus$ is the set minus operator. The goal of machine unlearning is to erase the impact of the target data $\mathcal{D}_t$ from the trained model $\mathbf{w}$. Subsequently, the resulting model post-unlearning, denoted as $\mathbf{w}'$, is expected to perform on the retained data set $\mathcal{D}_r$ similar to the model before unlearning.

3.2.1 Sharded, Isolated, Sliced, Aggregation (SISA)

The SISA training exhibits some resemblance to that of distributed training strategies. In SISA training (Bourtoule et al., 2021), the original data set $\mathcal{D}$ is partitioned into S disjoint shards denoted as $\mathcal{D}_s$, for $s = 1, 2, \ldots, S$. Each shard $\mathcal{D}_s$, for $s = 1, 2, \ldots, S$, is subsequently divided into R slices denoted as $\mathcal{D}_{s,r}$, for $r = 1, 2, \ldots, R$. The training process for each shard s begins with the initial training of a model $\mathbf{w}_{s,1}$ on the first slice $\mathcal{D}_{s,1}$, with the resulting parameters are stored. Subsequently, the model, initially trained on the first slice $\mathbf{w}_{s,1}$, undergoes additional training on the second slice $\mathcal{D}_{s,2}$, and the parameters of the resulting model $\mathbf{w}_{s,2}$ are stored. This iterative process continues until training is completed on the Rth slice and the parameters of $\mathbf{w}_{s,R}$ are stored. The final model, denoted as $\mathbf{w}_{s,R}$, represents the shard model $\mathbf{w}_s$. Finally, the shard models $\mathbf{w}_s$, for $s = 1, 2, \ldots, S$, are aggregated to form a comprehensive final model.

When an unlearning request arises to delete specific data samples, retraining becomes necessary only for the shard model whose shard contains the target data samples. The retraining process can be swiftly performed by utilizing the model parameters, stored before the training on the slice containing the data sample marked for unlearning.

SISA exhibits several limitations. Firstly, if the data set is partitioned in a non-i.i.d manner, the resulting final model accuracy tends to be notably low. Secondly, when unlearning requests arise simultaneously on multiple shards, the retraining process demands a substantial amount of computational resources and time. Lastly, the performance of the final model is significantly dependent on the number of shards and slices employed.

3.2.2 Amnesiac Unlearning

Amnesiac unlearning (Graves et al., 2020) selectively erases acquired knowledge associated with the targeted data. During the training process, the data holder, responsible for training an ML model, actively keeps track of the details of data samples within the data set, precisely noting which data samples appear in each batch. Additionally, the data holder also stores the corresponding updates made to the model parameters. This systematic storage of information serves as a crucial repository, ensuring a comprehensive record of the model's learning journey.

Upon receiving an unlearning request, the data holder takes action by removing the updates incorporated into the model from the specific batches containing the targeted data samples. Let $\mathbf{w}$ be an already trained model, and $\mathcal{B}_t$ denotes the set of batches containing the targeted data samples. Then, the amnesic unlearning generate the new model $\mathbf{w}'$ by effectively removing the impact of the targeted data from the already trained model $\mathbf{w}$ as

(Graves et al., 2020):

$$\mathbf{w}' = \mathbf{w} - \sum_{e=1}^{E} \sum_{b \in \mathcal{B}_t} \Delta \mathbf{w}_{e,b}, \tag{4}$$

where E denotes the number of epochs that the training phase has been carried out and $\Delta \mathbf{w}_{e,b}$ represents the contribution made by minibatch b to the model parameters $\mathbf{w}$ during epoch e of the training phase.

Amnesiac unlearning has proven to be effective in erasing the influence of a limited number of targeted data samples on trained model. As the number of affected batches increases, the resulting model efficacy after amnesiac unlearning diminishes proportionally. Furthermore, amnesiac unlearning exhibits diminished model efficacy on retained data when higher batch sizes are employed. It is important to emphasize that when a larger number of batches is affected, the model requires a brief retraining phase following the amnesic unlearning step to restore its performance. Notably, the implementation of amnesic learning demands substantial storage space to accommodate the tracking of updates made to the model parameters during training.

3.2.3 Label-Flipping-based Unlearning

The objective of unlearning through label-flipping is to intentionally obscure the model's comprehension of the targeted data, ensuring it possesses no valuable knowledge about that specific data . In this method, incorrect labels are randomly assigned to the targeted data samples. Subsequently, the network undergoes retraining through multiple iterations using this modified targeted data to fortify the intentional obfuscation (Graves et al., 2020). To unlearn an entire class, every example within that class needs to be assigned with a randomly chosen incorrect label. On the other hand, to unlearning a particular set of examples, assigning a few examples with erroneous labels is sufficient. The study in (Graves et al., 2020) recommends to retrain the model on the modified data only for a very few iterations for efficient unlearning.

3.2.4 Optimization-based Unlearning

In the optimization-based unlearning, the network undergoes a few iterations of retraining on the targeted data set $\mathcal{D}_t$ with objective to maximize the empirical loss instead of minimizing it. The optimization-based unlearning obtains the modified model, denoted as $\mathbf{w}'$, from the already trained model $\mathbf{w}$ using the batch gradient descent rule as follows:

$$\mathbf{w}'_e = \mathbf{w}'_{e-1} + \alpha' \, \nabla_{\mathbf{w}'} \mathcal{L}(\mathcal{D}_t, \mathbf{w}') \Big|_{\mathbf{w}'_{e-1}}, \tag{5}$$

where $\mathbf{w}'_0 = \mathbf{w}$ and α' is the learning rate of the retraining.

Although label-flipping and optimization-based unlearning methods effectively erase the knowledge of targeted data from trained models, the resulting models often perform worse on the retained data compared to their performance prior to unlearning.

4. Proposed Machine Unlearning Algorithms

In this section, we introduce a new class of unlearning algorithms referred to as layer-wise partial machine unlearning algorithms. Unlike conventional unlearning methods that erase targeted knowledge without regard for what has been learned from retained data, our approach introduces a selective, layer-wise erasure mechanism. By focusing on the internal representations within each layer, particularly those most critical for task-specific decisions, the proposed layer-wise erasure enables controlled and localized knowledge deletion. This allows us to preserve generalizable features while effectively removing targeted knowledge. This contrasts with existing methods that apply uniform erasure across the network, often leading to over-forgetting and degraded performance on retained data.

We integrate this layer-wise erasure mechanism into three state-of-the-art unlearning paradigms: amnesiac unlearning, label-flipping-based unlearning, and optimization-based unlearning. These adaptations give rise to a novel class of layer-wise partial machine unlearning algorithms, described in the following sections, that aim to efficiently erase the targeted knowledge while maintaining performance comparable to that before unlearning.

4.1 Partial Amnesiac Unlearning

As in conventional amnesia unlearning, subtracting the entire update associated with specific batches containing the targeted data adversely affects the model's performance. This happens because these updates include contributions not only from the targeted data but also from the retained data presented in those batches. Hence, the resultant model from conventional amnesiac unlearning exhibits poor performance on retained data, necessitating resource-intensive fine-tuning to restore model performance, especially for large-scale DNNs. Furthermore, to keep track of the updates made to the model during training phase, conventional amnesiac unlearning requires substantial storage space.

The proposed partial amnesiac unlearning effectively tackles these challenges by incorporating layer-wise selective pruning into the conventional amnesiac unlearning. During training phase, partial amnesiac unlearning prunes the updates made to the model layer-by-layer basis and store the pruned updates instead of retaining the entire update. Let $\Delta\mathbf{w}_{e,b}$ represent the contribution made by batch b during epoch e, and $\Delta\mathbf{w}_{l,e,b}$ denote the update corresponding to the lth layer in the DNN model. Then, partial amnesiac unlearning prunes $\Delta\mathbf{w}_{l,e,b}$ and obtains its pruned version $\widetilde{\Delta\mathbf{w}}_{l,e,b}$ as

$$\widetilde{\Delta\mathbf{w}}_{l,e,b} = \mathbf{P}_l \odot \Delta\mathbf{w}_{l,e,b}, \tag{6}$$

where $\mathbf{P}_l$ is the pruning matrix whose size is the same size as that of the lth layer of the DNN model. The elements of $\mathbf{P}_l$ are either 1 or 0. The symbol $\odot$ denotes the elementwise product or Hadamard product operator. The position of ones in $\mathbf{P}_l$ dictates which weights in the updates are going to be subtracted from the trained model. Furthermore, the number of zeros in $\mathbf{P}_l$ is determined by the pruning percentage.

In a nutshell, partial amnesiac unlearning sets certain weights in the update for each layer to zero before storing it. This process repeats for every layer in the DNN model, with percentage of pruning decreases as we move deeper in the network, until the model $\mathbf{w}$ is fully trained. It is important to highlight that storing the pruned update requires less storage space compared to storing the entire update. As a result, partial amnesiac unlearning demands less storage compared to its conventional counterpart.

Algorithm 1 Partial Amnesiac Unlearning

```
Require: Data set $\mathcal{D}$ in batches, number of epochs $E$, layer-wise pruning percentage $\{\rho_l\}$,
    pruning mode $\in$ {Random, Magnitude, Global}
 1: Initialize model weights $\mathbf{w}$
 2: Create empty update storage $\mathcal{S}$
 3: for $e = 1$ to $E$ do                                                   ▷ Training Phase
 4:     for batch $b$ in $\mathcal{D}$ do
 5:         Compute update $\Delta\mathbf{w}_{e,b}$
 6:         for each layer $l$ do
 7:             Construct pruning matrix $\mathbf{P}_l$ using $\rho_l$ and pruning mode
 8:             Prune layer update: $\widetilde{\Delta\mathbf{w}}_{l,e,b} = \mathbf{P}_l \odot \Delta\mathbf{w}_{l,e,b}$
 9:         end for
10:         Concatenate across layers: $\widetilde{\Delta\mathbf{w}}_{e,b} = \bigoplus_l \widetilde{\Delta\mathbf{w}}_{l,e,b}$
11:         Store $\widetilde{\Delta\mathbf{w}}_{e,b}$ in $\mathcal{S}$
12:         Update model: $\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}_{e,b}$
13:     end for
14: end for

Require: Targeted data batch index set $\mathcal{B}_t$
15: Set $\mathbf{w}' \leftarrow \mathbf{w}$
16: for all $(e, b, \widetilde{\Delta\mathbf{w}}_{e,b}) \in \mathcal{S}$ where $b \in \mathcal{B}_t$ do          ▷ Unlearning Phase
17:     $\mathbf{w}' \leftarrow \mathbf{w}' - \widetilde{\Delta\mathbf{w}}_{e,b}$
18: end for
19: return Unlearned model $\mathbf{w}'$
```

During the unlearning phase, partial amnesic unlearning subtracts the stored pruned updates $\widetilde{\Delta\mathbf{w}}_{e,b}$ corresponding to the specific batches containing the targeted data. Let $\mathcal{B}_t$ denotes the set of batches containing the data samples of targeted data. Then, the partial amnesic unlearning generate the new model $\mathbf{w}'$ from the trained model $\mathbf{w}$ as:

$$\mathbf{w}' = \mathbf{w} - \sum_{e=1}^{E} \sum_{b \in \mathcal{B}_t} \widetilde{\Delta\mathbf{w}}_{e,b}. \tag{7}$$

By selectively removing only a portion of the contribution from specific batches made to the model during the training phase, the proposed partial amnesiac unlearning minimizes the loss of knowledge acquired from the retained data in those batches. This approach guarantees the preservation of valuable knowledge from the retained data while effectively erasing the influence of targeted data from the model.

Within the framework of partial amnesiac unlearning, we explore three distinct pruning mechanisms, such as layer-wise random pruning, layer-wise magnitude-based pruning, and global pruning. Layer-wise random pruning involves randomly removing the parameters from the model. Layer-wise magnitude-based pruning, on the other hand, identifies and removes parameters based on the magnitude, typically removing those with lower magnitudes. Global pruning, in contrast, involves the removal of a fixed percentage of the least important parameters across the entire DNN model. It is important to highlight that when the pruning percentage for each layer of the DNN is set to zero, the proposed partial amnesiac unlearning transforms into conventional amnesiac unlearning.

Given that initial layers of a DNNs play a crucial role in learning fundamental representations, a prudent approach is to employ more aggressive pruning in these initial layers of the update. As we traverse deeper into the network, the pruning percentage can be gradually decreased. This strategy aims to preserve the low-level representations of retained data within the model, preventing the inadvertent removal of essential learned features during the unlearning process. The proposed partial amnesiac unlearning is summarized in Algorithm 1.

4.2 Layer-wise Partial Updates induced Label-Flipping-based Unlearning

Assigning incorrect labels to the targeted data set $\mathcal{D}_t$ and updating the trained model $\mathbf{w}$ for a limited number of iterations can effectively erase the targeted knowledge. However, when such updates are applied indiscriminately across all layers of the model, they may distort useful representations learned from the retained data. In DNNs, the final layers are typically responsible for task-specific decisions such as class predictions in classification tasks. Fully updating these layers using incorrectly labeled data can degrade the model's ability to distinguish between classes, resulting in performance drops on the retained data. By introducing partial updates that prioritize early layers while limiting updates to deeper layers, our method aims to unlearn targeted data with minimal disruption to retained knowledge.

To address this issue, we propose to utilize layer-wise partial update mechanism that selectively adjusts weights across different layers during unlearning. The proposed approach leverages the observation that initial layers in DNNs learn fundamental, task-agnostic representations that are shared across classes. These features are more resilient to label noise and input perturbations, and contribute significantly to the model's ability to generalize. Hence, performing full updates in these layers can help the model realign shared features without harming generalization. In contrast, deeper layers encode class-specific features and are more sensitive to label correctness. Updating them using flipped labels may distort the decision boundaries and degrade the model's performance on retained classes.

Let $\mathcal{D}'_t$ be the modified targeted data with incorrect labels. Layer-wise partial-updates induced label-flipping-based unlearning obtains the new model $\mathbf{w}'$ from the trained model $\mathbf{w}$ as:

$$\mathbf{w}'_e = \mathbf{w}'_{e-1} - \alpha' \, \widetilde{\nabla_{\mathbf{w}'}} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}')\Big|_{\mathbf{w}'_{e-1}}, \tag{8}$$

where $\widetilde{\nabla_{\mathbf{w}'}} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}')$ represents the layer-wise partial gradient of the loss, i.e., a fraction of the actual gradient $\nabla_{\mathbf{w}'} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}')$ with $\mathbf{w}'_0 = \mathbf{w}$ (the original model parameters), and α' is the learning rate of the unlearning. In $\widetilde{\nabla_{\mathbf{w}'}} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}')$, the lth layer partial gradient, denoted by $\widetilde{\nabla \mathbf{w}'_l} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}')$ is:

$$\widetilde{\nabla \mathbf{w}'_l} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}') = \mathbf{S}_{l,e} \odot \nabla_{\mathbf{w}'_l} \mathcal{L}(\mathcal{D}'_t, \mathbf{w}'), \tag{9}$$

Algorithm 2 Layer-wise Partial Updates Induced Label-Flipping-based Unlearning

```
Require: Trained model $\mathbf{w}$, targeted data $\mathcal{D}_t$, learning rate $\alpha'$, number of unlearning
    epochs $E$, layer-wise pruning percentage $\{\rho_l\}$
 1: Initialize unlearning model: $\mathbf{w}'_0 \leftarrow \mathbf{w}$
 2: Initial selection matrices $\{\mathbf{S}_{l,0}\}$ for all layers $l$
 3: Randomly flip the original labels and obtain the modified targeted data $\mathcal{D}'_t$
 4: for $e = 1$ to $E$ do                                          ▷ Unlearning Phase
 5:     for each layer $l$ do
 6:         Compute actual gradient: $\nabla_{\mathbf{w}'_l}\mathcal{L}(\mathcal{D}'_t, \mathbf{w}'_{e-1})$
 7:         Apply layer-wise pruning: $\widetilde{\nabla_{\mathbf{w}'_l}}\mathcal{L} = \mathbf{S}_{l,e-1} \odot \nabla_{\mathbf{w}'_l}\mathcal{L}(\mathcal{D}'_t, \mathbf{w}'_{e-1})$
 8:     end for
 9:     Aggregate layer-wise partial gradients: $\widetilde{\nabla_{\mathbf{w}'}}\mathcal{L} = \bigoplus_l \widetilde{\nabla_{\mathbf{w}'_l}}\mathcal{L}$
10:     Partially update model: $\mathbf{w}'_e \leftarrow \mathbf{w}'_{e-1} - \alpha' \cdot \widetilde{\nabla_{\mathbf{w}'}}\mathcal{L}$
11:     for each layer $l$ do
12:         Update selection matrix: $\mathbf{S}_{l,e} \leftarrow \texttt{CircularShift}(\mathbf{S}_{l,e-1})$
13:     end for
14: end for
15: return Unlearned model $\mathbf{w}'$
```

where $\nabla_{\mathbf{w}'_l}\mathcal{L}(\mathcal{D}'_t, \mathbf{w}')$ is the actual gradient corresponds to lth layer of DNN model. $\mathbf{S}_{l,e}$ is the selection matrix, containing elements of either zero or one. The positions of ones in $\mathbf{S}_{l,e}$ determine which weights of the lth layer receive updates during epoch e. The selection matrix for the next epoch $\mathbf{S}_{l,e+1}$, can be obtained by performing a simple circular shift operation on the selection matrix of the current epoch $\mathbf{S}_{l,e}$. If all elements of the selection matrix $\mathbf{S}_{l,e}$ are set to one, the proposed layer-wise partial updates induced label-flipping-based unlearning reduces to its conventional counterpart. The proposed layer-wise partial updates induced label-flipping-based unlearning is summarized in Algorithm 2.

To maximize unlearning efficacy while preserving generalization, we recommend configuring the selection matrices to allow denser updates in the initial layers and progressively sparser updates in the deeper layers.this strategy helps to preserve essential generic features while mitigating the risk of distorting class-specific representations.

Importantly, this approach is not equivalent to using smaller layer-specific learning rates. While a small learning rate scales down the magnitude of all parameter updates, it does not restrict which parameters are getting updated. In contrast, our method explicitly enforces sparsity in the updates, i.e., only a subset of parameters is modified, leading to quite different behavior. For instance, a small learning rate would still gradually update all weights, potentially overwriting useful knowledge over time. Our method avoids this by selectively preventing updates to most weights, thereby improving control over which information is retained and which is removed. This targeted update strategy can lead to better retention of general, features while still enabling effective unlearning.

Algorithm 3 Layer-wise Partial Updates Induced Optimization-based Unlearning

```
Require: Trained model $\mathbf{w}$, targeted data $\mathcal{D}_t$, number of epochs $E$, learning rate $\alpha'$, layer-wise pruning percentage $\{\rho_l\}$
 1: Initialize unlearning model: $\mathbf{w}'_0 \leftarrow \mathbf{w}$
 2: Initialize selection matrices $\{\mathbf{S}_{l,0}\}$ for all layers $l$
 3: for $e = 1$ to $E$ do                                          ▷ Unlearning Phase
 4:     for each layer $l$ do
 5:         Compute gradients $\nabla_{\mathbf{w}'}\mathcal{L}(\mathcal{D}_t, \mathbf{w}'_{e-1})$
 6:         Apply layer-wise pruning: $\widetilde{\nabla_{\mathbf{w}'_l}}\mathcal{L} = \mathbf{S}_{l,e-1} \odot \nabla_{\mathbf{w}'_l}\mathcal{L}(\mathcal{D}_t, \mathbf{w}'_{e-1})$
 7:     end for
 8:     Aggregate layer-wise partial gradients: $\widetilde{\nabla_{\mathbf{w}'}}\mathcal{L} = \bigoplus_l \widetilde{\nabla_{\mathbf{w}'_l}}\mathcal{L}$
 9:     Partially update model: $\mathbf{w}'_e \leftarrow \mathbf{w}'_{e-1} + \alpha'\widetilde{\nabla_{\mathbf{w}'}}\mathcal{L}$
10:     for each layer $l$ do
11:         Update selection matrix: $\mathbf{S}_{l,e} \leftarrow \texttt{CircularShift}(\mathbf{S}_{l,e-1})$
12:     end for
13: end for
14: return Unlearned model $\mathbf{w}'$
```

4.3 Layer-wise Partial Updates induced Optimization-based Unlearning

Updating the trained model $\mathbf{w}$ on the targeted data set $\mathcal{D}_t$ by maximizing its empirical loss is a well-established approach to erasing target data knowledge. However, when such updates are applied uniformly across all layers, they can disrupt essential, task-agnostic representations that contribute to generalization. As a result, performance on the retained data may drop significantly.

To address this limitation, we propose to utilize layer-wise partial update mechanism that selectively modifies the model during optimization-based unlearning. This approach leverages the observation that early layers in DNNs tend to encode low-level, general-purpose features shared across classes. Updating these layers with gradients designed to increase the loss on $\mathcal{D}_t$ may corrupt general knowledge, leading to excessive forgetting. By restricting updates to deeper layers and limiting the amount of updates in earlier layers, the proposed strategy allows targeted erasure with minimal impact on retained knowledge.

Layer-wise partial updates induced optimization-based unlearning obtains the new model $\mathbf{w}'$ from the trained model $\mathbf{w}$ as:

$$\mathbf{w}'_e = \mathbf{w}'_{e-1} + \alpha' \, \widetilde{\nabla_{\mathbf{w}'}}\mathcal{L}(\mathcal{D}_t, \mathbf{w}')\Big|_{\mathbf{w}'_{e-1}}, \tag{10}$$

where $\widetilde{\nabla_{\mathbf{w}'}}\mathcal{L}(\mathcal{D}_t, \mathbf{w}')$ represents the layer-wise partial gradient of the loss, i.e., a fraction of the actual gradient $\nabla_{\mathbf{w}'}\mathcal{L}(\mathcal{D}_t, \mathbf{w}')$ with $\mathbf{w}'_0 = \mathbf{w}$ (the original trained model parameters), and α' is the learning rate of the unlearning. In $\widetilde{\nabla_{\mathbf{w}'}}\mathcal{L}(\mathcal{D}_t, \mathbf{w}')$, the lth layer partial gradient, denoted by $\widetilde{\nabla \mathbf{w}'_l}\mathcal{L}(\mathcal{D}_t, \mathbf{w}')$ is same as given in (9). The proposed layer-wise partial updates induced optimization-based unlearning is summarized in Algorithm 3.

When employing layer-wise partial updates into optimization-based unlearning method, it is recommended to configure the selection matrices to update a fewer parameters in the initial layers and gradually increase the number of updated parameters as the network depth increases. In this way it is possible to preserve essential generic features while selectively modifying class-specific representations, allowing the erasure of targeted class knowledge.

5. Experimental Results

Extensive experiments were conducted to evaluate the efficacy of the proposed class of unlearning methods and provided comparison to their naive counterparts. We considered the model accuracy on the targeted data and retained data as performance indicator.

5.1 Experimental Hardware Setup

All experiments were performed in Python 3.11 and use the PyTorch deep learning library Paszke et al. (2019). The system was equipped with an NVIDIA Tesla V100 GPU with 48GB of memory.

5.2 Data Sets

Four diverse data sets were considered in our experiments, three from computer vision domain and the other from medical domain. This deliberate choice allows for a thorough assessment of the effectiveness and generalizability of our proposed methods across varied domains, ensuring the reliability and applicability of our experimental results.

1. MNIST Handwritten Digits data set (LeCun et al., 2010): is a widely recognized benchmark datset, comprises a total of $70,000$ grayscale images, with $60,000$ images designated for training and $10,000$ images for testing. Each image is of size 28×28 pixels depicting handwritten digits from 0 to 9. Each image is accompanied by corresponding labels indicating the digit represented.

2. Street View House Numbers (SVHN) data set (Netzer et al., 2011): is derived from real-world street images, designed for multi-class classification of the 10 digits (0 through 9). The data set comprises $99,289$ color images in the training set, $73,257$ color images in the test set. Each image, depicting a digit from house number plates, has a resolution of $3 \times 32 \times 32$ pixels. Due to its diverse and realistic collection of digit images, capturing variations in lighting, occlusion, and background noise, the SVHN data set provides a challenging benchmark for image classification algorithms, making it a valuable resource for advancing research in computer vision and machine learning.

3. CIFAR10 Image data set (Krizhevsky and Hinton, 2009): is another prominent benchmark data set, consisting of a total of $60,000$ color images, with $50,000$ images allocated for training and $10,000$ images for testing. The data set is organized into 10 classes, with 6000 images per class.. Each image, depicting an object from one of the class, with a resolution of $3 \times 32 \times 32$ pixels. The CIFAR-10 data set brings richness to the evaluation of image classification algorithms through its diverse and comprehensive collection of real-world images.

4. MEDMNIST-OrganAMNIST Medical Image data set (Yang et al., 2022): is a publicly available benchmark data set for medical image analysis. Derived from 3D-computed tomography (CT) images sourced from the Liver Tumor Segmentation Benchmark (LiTS) (Bilic et al., 2023), the data set is tailored for multi-class classification of 11 body organs. With a total of $58,850$ grayscale images, the data set is partitioned into $34,581$ training images, $6,491$ validation images, and $17,778$ testing images. Each image, depicting a body organ, possesses a resolution of $1 \times 224 \times 224$ pixels. The OrganAMNIST data set enhances the experimental evaluation of image classification algorithms, bringing richness through its diverse and comprehensive collection of medical domain images.

5.3 Network Architectures

- Multi-layer Perceptron (MLP): comprises two fully connected layers. The hidden layer consists of 500 neurons.
- LeNet: one of the foundational models in the development of modern CNNs (LeCun et al., 1998). LeNet typically consists of seven layers, including three convolutional layers followed by two pooling layers and two fully connected layers.
- AlexNet: winner of the ImageNet large scale visual recognition challenge (ILSVRC) in 2012, which significantly advanced the field of computer vision. Layer wise details of AlexNet can be found in (Krizhevsky et al., 2012).
- VGG11 and VGG19: are variants of visual geometry group (VGG) models, employs a uniform structure of stacked 3×3 convolutional layers followed by max-pooling layers for spatial dimension reduction (Simonyan and Zisserman, 2015a).
- ResNet9, ResNet18 and ResNet50: are variants of the residual network (ResNet) architecture, typically consist of several residual blocks, each containing convolutional layers and shortcut connections followed by a single fully connected layer (He et al., 2016).
- Simple ViT and ViT-Large: are variants of vision transformers (ViTs) (Dosovitskiy et al., 2021). Of these, Simple ViT consists of 4 Transformer encoder layers, each equipped with six-headed self-attention mechanisms and MLP blocks. The ViT-Large model comprises 24 encoder layers, each with 16-headed self-attention mechanisms and MLP blocks.

5.4 Assessment of Partial Amnesiac Unlearning

5.4.1 Performance Comparison: Partial Amnesiac Unlearning versus State-of-the-Art Methods

We evaluated the performance of the proposed partial amnesiac unlearning method across several benchmark data sets, including MNIST, CIFAR10, SVHN and OrganAMNIST. This evaluation encompassed a variety of architectures, ranging from simple MLP to more complex CNNs and ViTs. During the evaluation, we ensured zero accuracy on the targeted data set, while assessing the performance on the retained data set. In the MNIST data set, class 3 was chosen as the targeted data, while the other classes were considered to be retained data. For CIFAR10 and SVHN data sets, class 3 was targeted; in OrganAMNIST, class 2 was designated as the targeted class.

We compared the proposed partial amnesiac unlearning with random pruning against several state-of-the-art unlearning techniques: naive retraining, which serves as the baseline; SISA, implemented with 8 shards, each having 4 slices; FMUL; and conventional amnesiac unlearning. For a fair comparison, both FMUL and conventional amnesiac unlearning were evaluated without repair and fine-tuning steps, respectively. The results are presented in Table 1.

Table 1: Performance comparison of the proposed partial amnesiac unlearning with random pruning against the state-of-the-art unlearning techniques, such as naive retraining, SISA (Bourtoule et al., 2021), FMUL (Tarun et al., 2021) and amnesiac unlearning (AUL) (Graves et al., 2020). Model accuracy (in %) on retained data set is presented while the model accuracy on targeted data set being zero.

		Retraining	SISA	FMUL	AUL	Proposed
MNIST	MLP	97.85	**96.27**	92.58	59.58	95.39
	LeNet	97.80	92.21	**98.00**	65.89	97.11
	ResNet9	99.24	96.62	**99.46**	74.81	98.62
CIFAR10	VGG11	77.18	61.15	73.80	16.77	**75.46**
	ResNet18	73.61	67.02	67.42	13.64	**73.72**
	Simple ViT	65.77	57.78	62.64	21.66	**63.27**
SVHN	VGG19	93.72	88.71	80.51	22.29	**90.12**
	ResNet18	92.23	88.73	86.06	17.24	**89.57**
	Simple ViT	90.50	80.78	86.31	24.25	**88.39**
OrganAMNIST	AlexNet	87.71	76.86	**86.51**	40.73	81.30
	ResNet50	92.79	80.21	88.36	18.03	**89.89**
	ViT-Large	86.77	76.45	83.76	15.15	**85.00**

From Table 1, it is evident that SISA performs effectively on simpler data sets like MNIST and with shallow network architectures. However, it struggled with deeper architectures and more complex data sets, such as CIFAR-10, SVHN and OrganAMNIST. Furthermore, our experiments revealed that SISA's performance significantly drops when data is distributed across shards in a non-i.i.d manner.

On the other hand, FMUL, without the repair step, achieves performance comparable to naive retraining for the MNIST data set. However, it exhibited performance drop for CIFAR10, SVHN and OrganAMNIST data sets. Further, FMUL relies on a subset of the retained data set during the unlearning process, which sets it apart from methods that do not utilize any retained data for unlearning, such as conventional amnesiac unlearning, and our proposed partial amnesiac unlearning.

When it comes to conventional amnesiac unlearning without fine-tuning, there is a significant performance drop for all data sets. This decline in performance can be attributed to the intrinsic nature of conventional amnesiac unlearning, which not only removes the representations of targeted data from trained models but also erases representations acquired from the retained data.

Whereas, the proposed partial amnesiac unlearning adeptly mitigates this detrimental effect, exhibiting better performance consistently across all data sets and network architectures. This is accomplished by selectively subtracting the layer-wise pruned model updates, rather than discarding the entire updates made by the affected batches to the model during the training process. Subtracting pruned updates, with the pruning amount decreasing as going into deeper into the network, mitigates the disturbance of representations acquired from retained data in the affected batches. As a result, the proposed partial amnesiac unlearning method showcases model efficacy comparable to the performance of naive retraining. It is worth noting that this performance is achieved without requiring any fine-tuning post-unlearning and with less storage space compared to conventional amnesiac unlearning.

5.4.2 Model Behavior During Training and Unlearning Phases

To investigate the effectiveness of the proposed partial amnesiac unlearning, we plotted model behavior during training and unlearning phases and made direct comparisons with conventional amnesiac unlearning. Similar to the previous experiment, specific classes were designated as the targeted data. In the MNIST, CIFAR, and SVHN data sets, class 3 was chosen as the targeted data, while the other classes were considered to be retained data. For OrganAMNIST, class 2 was designated as the targeted class.

Initially, various models were trained for 8 epochs with a batch size of 128 across different data sets: a 2-layer MLP, LeNet, and ResNet9 for MNIST; VGG11, ResNet18, and SimpleViT for CIFAR-10; VGG19, ResNet18, and SimpleViT for SVHN; and AlexNet, ResNet50, and ViT-Large for OrganAMNIST. The learning rate was set to 0.001 for shallow networks, while deeper networks were trained with a lower learning rate of either 0.0005 or 0.0001, depending on their depth and complexity. After 8 epochs, an unlearning request was initiated to erase the knowledge of the targeted class from the trained models. Both the proposed partial amnesiac unlearning with random pruning and conventional amnesiac unlearning (without any fine-tuning) were simulated to process the unlearning request.

The validation accuracy on targeted and retained data sets before and after unlearning is illustrated in Figure 1. From MNIST results (first row), it is evident that conventional amnesiac unlearning successfully erases the targeted data within 2 to 3 epochs but exhibits a performance drop on the retained data. Whereas, the proposed partial amnesiac unlearning takes slightly longer time to remove the targeted data but maintains accuracy on the retained data, similar to pre-unlearning performance.

A similar trend was observed for CIFAR-10 (second row) and SVHN (third row). Conventional amnesiac unlearning effectively erases the targeted data knowledge from trained models but shows significant performance drop on the retained data. Whereas, the proposed partial amnesiac unlearning with random pruning, preserves the accuracy on the retained data while effectively removing the impact of targeted data from trained models.

For OrganAMNIST (fourth row), which involves complex DNNs, the performance drop of conventional amnesiac unlearning on the retained data is even more pronounced. The partial amnesiac unlearning method again demonstrated superior performance, maintaining

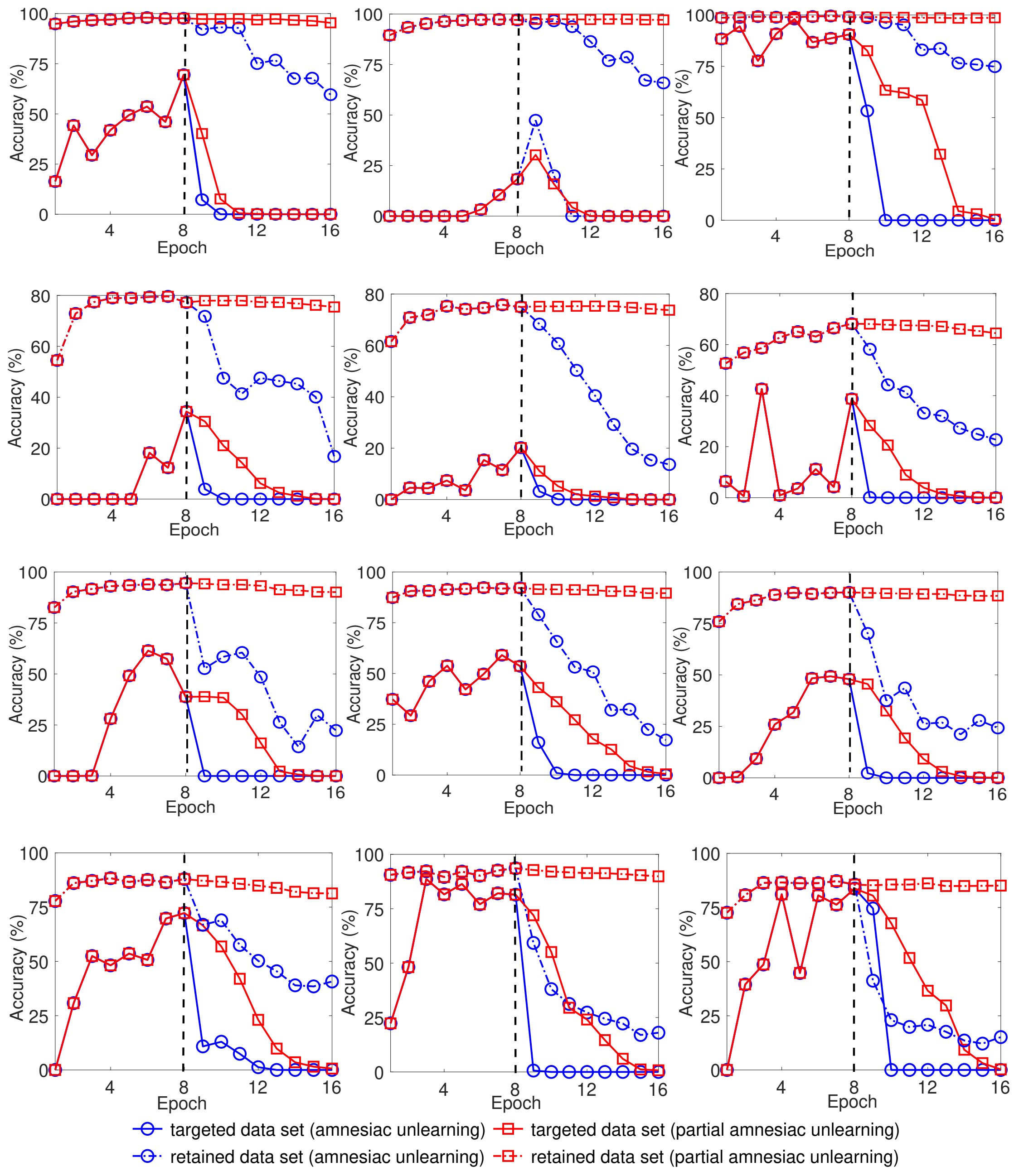

Figure 1: Comparison of model behavior between partial amnesiac unlearning and conventional amnesiac unlearning during training and unlearning phases. 1st row: MNIST data set with MLP (left), LeNet (middle), ResNet9 (right) architectures. 2nd row: CIFAR10 data set with VGG11 (left), ResNet18 (middle), SimpleViT (right). 3rd row: SVHN data set with VGG19 (left), ResNet18 (middle), SimpleViT (right). 4th row: OrganAMNIST data set with AlexNet (left), ResNet50 (middle), ViT-Large (right)

architectures. The vertical dashed line marks the unlearning request initiation.

comparable accuracy on the retained data at levels similar to pre-unlearning while effectively erasing the targeted data without requiring any fine-tuning.

In brief, the proposed partial amnesiac unlearning method consistently outperforms conventional amnesiac unlearning by preserving model performance on retained data while effectively removing targeted data across all data sets and neural network architectures. It achieves this without requiring any additional fine-tuning or using retained data during the unlearning process, making it a robust and efficient unlearning technique.

5.4.3 Performance on Retained Data Set vs. Amount of Pruning Applied to the Model Updates

Next, we investigated the effectiveness of partial amnesiac unlearning for different levels of layer-wise selective pruning, also referred to as sparsity levels. For this, we considered the data sets such as MNIST, CIFAR10, SVHN, and OrganAMNIST, and the network architectures tested are MLP, CNNs such as LeNet and AlexNet, ResNet9, ResNet18, ResNet50, VGG11, VGG19, SimpleViT and ViT-Large. The corresponding results depicting the accuracy on the retained data set versus the sparsity of model updates, are displayed in Figure 2.

In our results, zero sparsity (i.e., no pruning) corresponds conventional amnesiac unlearning. On the other extent, model updates made during training were pruned as much as possible while ensuring the unlearning model exhibited zero accuracy on the targeted data set. If we prune beyond this point, i.e., the sparsity levels corresponds to red colour bar in all plots, the targeted data cannot be completely erased from the model. After subtracting these pruned updates from the trained model, the accuracy of the resulting unlearned model on the retained data set is plotted. Additionally, the plots include performance values at intermediate sparsity levels of 25% and 50% to illustrate the impact of various pruning ratios. it is important to mention that the layer-wise pruning was applied only to the weight parameters of model updates, leaving bias parameters untouched.

From Figure 2, it is evident that subtracting the entire model updates made during training from trained model results in poor performance on retained data set. Furthermore, storing whole model updates demands a significant amount of storage space. In contrast, subtracting layer-wise pruned model updates—where the amount of pruning decreases as we progress deeper into the network (e.g., pruning 90% in the initial layers and restricting pruning to 50% to 40% in the final fully connected layers), preserves performance while significantly reducing the storage space required to keep track of model updates made during training. These results indicate that the proposed partial amnesiac unlearning achieves improved performance performance on retained data sets, while requiring an average of 75% less storage space compared to conventional amnesiac unlearning. More importantly, the proposed partial amnesiac unlearning eliminates the need for fine-tuning.

5.4.4 Model Accuracy on Targeted and Retained Classes

To elucidate the reduction in model effectiveness on the retained data after applying conventional amnesiac unlearning, we present the model accuracy for specific randomly chosen retained classes across various data sets. For instance, we examined classes 2, 7, and 9 in MNIST, CIFAR10, classes 2, 5, and 8 in SVHN and classes 0, 6, and 10 in OrganAMNIST.

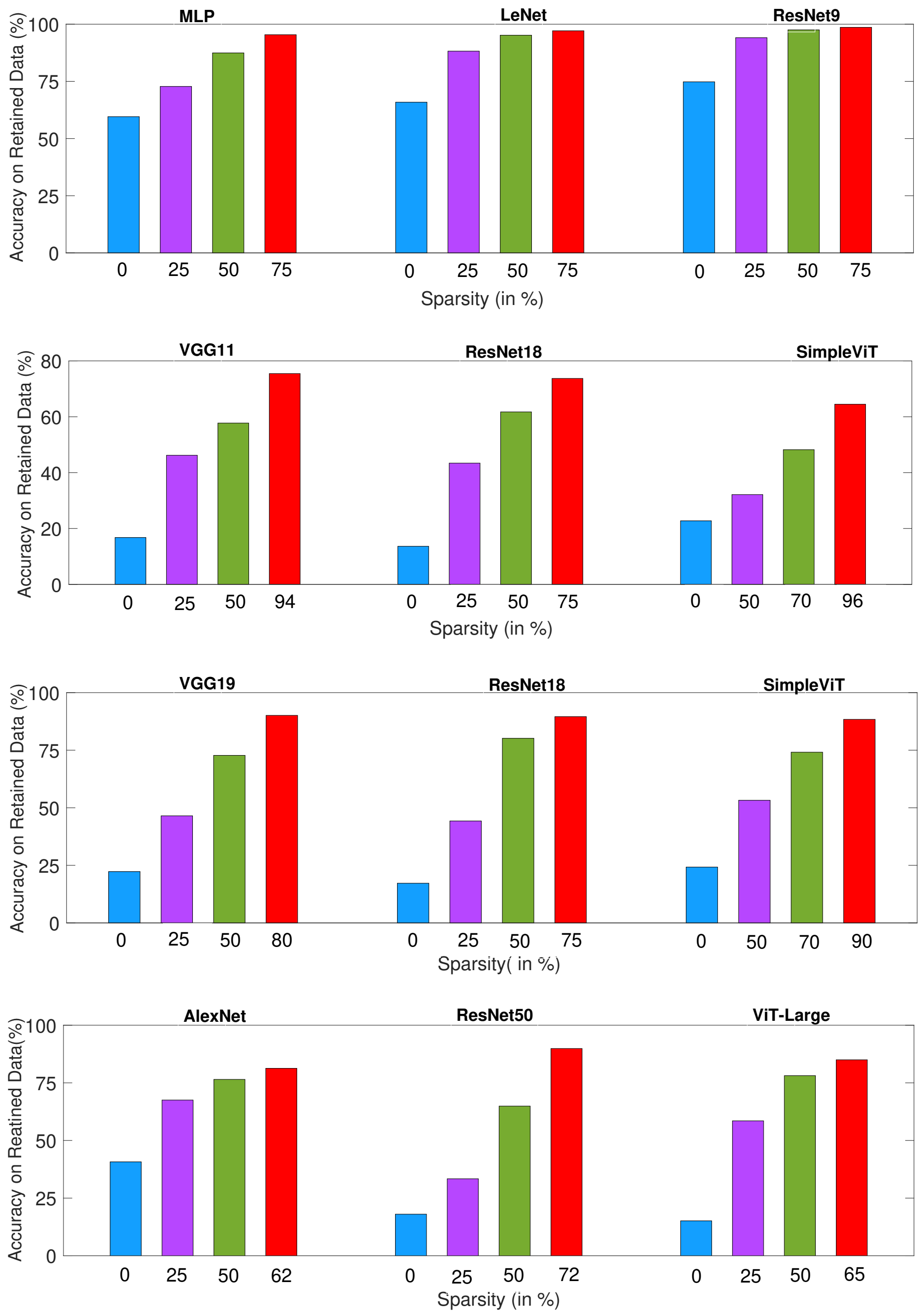

Figure 2: Performance of partial amnesiac unlearning with random pruning on retained data set versus amount of pruning applied to model updates. Zero sparsity corresponds to conventional amnesiac unlearning. 1st row: MNIST data set, 2rd row: CIFAR10, 3rdrow: SVHN data set, 4throw: OrganAMNIST data set. The sparsity associated with the red colour bar represents the maximum amount of pruning that results in zero accuracy on targeted data.

	Network Architecture	Method	$\mathcal{D}_t \downarrow$	$\mathcal{D}_r \uparrow$	Retained Classes		
					Class 2 $\uparrow$	Class 7 $\uparrow$	Class 9 $\uparrow$
MNIST	MLP	(Graves et al., 2020)	0	59.58	89.72	0.38	99.60
		Proposed	0	95.39	98.15	90.75	99.50
	LeNet	(Graves et al., 2020)	0	65.89	54.36	96.01	98.81
		Proposed	0	97.11	96.99	95.81	96.92
	ResNet9	(Graves et al., 2020)	0	74.81	3.58	98.44	0.09
		Proposed	0	98.62	96.22	99.70	96.13
					Class 2 $\uparrow$	Class 7 $\uparrow$	Class 9 $\uparrow$
CIFAR10	VGG11	(Graves et al., 2020)	0	16.77	41.60	6.50	2.90
		Proposed	0	75.46	82.80	87.20	75.20
	ResNet18	(Graves et al., 2020)	0	13.64	1.20	0	0
		Proposed	0	73.72	44.70	78.30	73.60
	SimpleViT	(Graves et al., 2020)	0	21.66	22.60	0	0
		Proposed	0	63.27	43.50	85.30	67.60
					Class 2 $\uparrow$	Class 5 $\uparrow$	Class 8 $\uparrow$
SVHN	VGG19	(Graves et al., 2020)	0	22.29	0	0	0
		Proposed	0	90.12	91.80	91.56	56.86
	ResNet18	(Graves et al., 2020)	0	17.24	0	48.40	92.46
		Proposed	0	89.57	92.19	93.54	86.50
	SimpleViT	(Graves et al., 2020)	0	24.25	20.36	0	0
		Proposed	0	88.39	95.63	87.83	79.27
					Class 0 $\uparrow$	Class 6 $\uparrow$	Class 10 $\uparrow$
OrganAMNIST	AlexNet	(Graves et al., 2020)	0	40.73	1.25	99.93	1.27
		Proposed	0	81.30	36.67	98.78	74.52
	ResNet50	(Graves et al., 2020)	0	18.03	37.45	0	0
		Proposed	0	89.89	83.88	95.98	92.03
	ViT-Large	(Graves et al., 2020)	0	15.15	3.37	0	0
		Proposed	0	85.00	83.78	97.19	65.23

Table 2: Comparison of model efficacy between partial amnesiac unlearning and conventional amnesiac unlearning on targeted and retained classes of MNIST, CIFAR10, SVHN, and OrganAMNIST data sets.

These results are shown in Table 2. These results, shown in Table 2, illustrate how conventional amnesiac unlearning affects the classification accuracy of retained classes, providing a clear comparison of its detrimental effects.

The results clearly indicates that the conventional amnesiac unlearning is successful in erasing the impact of targeted data from trained models, but adversely impacts the classification accuracy of retained classes. For example, in the case of the MNIST data set and MLP model, when erasing the knowledge of the targeted class (i.e., class 3) from the trained model, conventional amnesiac unlearning completely eliminated the model's ability to identify class 7. Similarly, in the case of LeNet and ResNet9, the model's capability to correctly identify class 2 is significantly compromised after the unlearning process. On the other hand, employing the proposed partial amnesiac unlearning for erasing targeted class from trained models shows a very minimal impact on the accuracy of the retained classes.

The detrimental effects of conventional amnesiac unlearning are more noticeable in the CIFAR10, SVHN, and OrganAMNIST data sets. Table 2 displays the model accuracy for specific retained classes. For instance, in the case of the CIFAR10 data set and the ResNet18 model, erasing the targeted class 3 from the trained model resulted in the complete erasure of retained classes 2, 7 and 9. Similarly, in the case of the OrganAMNIST data set and the ViT-Large model, erasing the targeted class 3 from the trained models resulted in the complete erasure of retained classes 1, 6, and 10. The results in Table 2 underscore the significant adverse impact of conventional amnesiac unlearning on retained classes, demonstrating notably poor model efficacy for most retained classes. In contrast, the proposed partial amnesiac unlearning method successfully erases the impact of the targeted class from the trained models with minimal negative influence on the accuracy of the retained classes.

The rationale behind this lies in subtracting the pruned updates, as opposed to the entire update made during training, which minimizes the adverse effects on the model's behavior in partial amnesiac unlearning. Additionally, applying more aggressive pruning to the initial layers, with the pruning percentage decreasing as the network depth increases, shows no detrimental impact on the model's ability to learn fundamental features. Consequently, the model after partial amnesiac unlearning exhibits better performance on the retained data set and requires no brief fine-tuning, unlike in the case of conventional amnesiac unlearning.

5.4.5 Assessment of Partial Amnesiac Unlearning using Class Activation Maps

To further examine the performance decline associated with conventional amnesiac unlearning, we leveraged class activation maps (CAMs) (Zhou et al., 2016). CAMs are generated by utilizing the feature maps obtained from the final convolutional layer of a CNN. Combining these feature maps using the the model's weights for a specific class generates heatmap, which visually represents the image regions that most contribute to the prediction of a that class. CAMs are very useful for interpreting the behavior of CNNs.

CAMs of LeNet model for MNIST images belonging to class-8 and class-7 before unlearning and corresponding CAMs of the same images after conventional amnesiac unlearning and the proposed partial amnesiac unlearning are presented in Figure. 3. In CAMS presented in Figure. 3, blue colour signifies low activation regions, indicating minimal influence on the predicted class, while red denotes high activation regions, representing significant contributions to the classification decision.

From Figure. 3, it is evident that the proposed partial amnesiac unlearning preserves the regions identified to be important for predicting the specific class. As a result the model performs well on retained classes without necessitating for brief-fine tuning post-unlearning. On the other hand, conventional amnesiac unlearning alters a few important regions to insignificant regions, resulting in poor performance on the retained classes. CAMs clearly illustrate the spatial changes in model focus, offering insight into why partial amnesiac unlearning can maintain performance on the retained data set, while its conventional counterpart cannot.

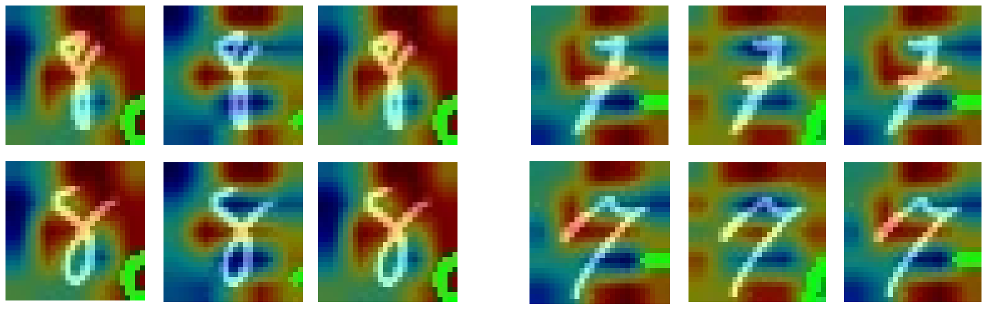

Figure 3: Class activation maps of the MNIST data set using LeNet model. Images from retained class 8 are on the left side, and retained class 7 are on the right side. In each case, the first, second, and third column images represent the class activation maps before unlearning, after applying conventional amnesiac unlearning, and proposed partial amnesiac unlearning, respectively. Blue colour indicates low activation regions, while red colour indicates high activation regions.

5.4.6 Model Accuracy against Number of Affected Batches for Different Pruning Strategies

Next, we investigated how the number of affected batches influences model efficacy in partial amnesiac learning. In this context, affected batches indicate the batches that consists of targeted data samples. To do so, we plotted the percentage of affected batches vs model accuracy on the retained data set for MNIST, OrganAMNIST, CIFAR10, and SVHN data sets as shown in Figure 4. The figure also includes the partial amnesiac unlearning curves for three pruning strategies: layer-wise random pruning, layer-wise magnitude-based pruning, and global pruning.

Figure 4 clearly indicates that the deterioration of model efficacy on the retained data, following conventional amnesiac unlearning, becomes increasingly apparent as the number of affected batches rises. This decline in performance stems from the inherent nature of conventional amnesiac unlearning, which not only removes knowledge of targeted data from the trained models but also eliminates knowledge acquired from the retained data. On the contrary, partial amnesiac unlearning adeptly mitigates this detrimental effect. This is accomplished by selectively removing the pruned updates, rather than discarding the entire updates made by the affected batches to the model during the training process. This helps to preserve the knowledge gained from retained data in the affected batches. As a result, the partial amnesiac unlearning method showcases model efficacy comparable to the performance observed pre-unlearning.

Among pruning strategies, global pruning aims to prune a specified amount of parameters from model updates without considering how this pruning is distributed across different layers. This indiscriminate pruning can lead to situations where some layers remain unpruned, while others may lose a disproportionate amount of updates. As a consequence, fundamental representations learned from the retained data in the initial layers, as well as class-specific representations in the final layers, may become corrupted. Ultimately, global pruning often leads to a decline in performance on the retained data set, as the model

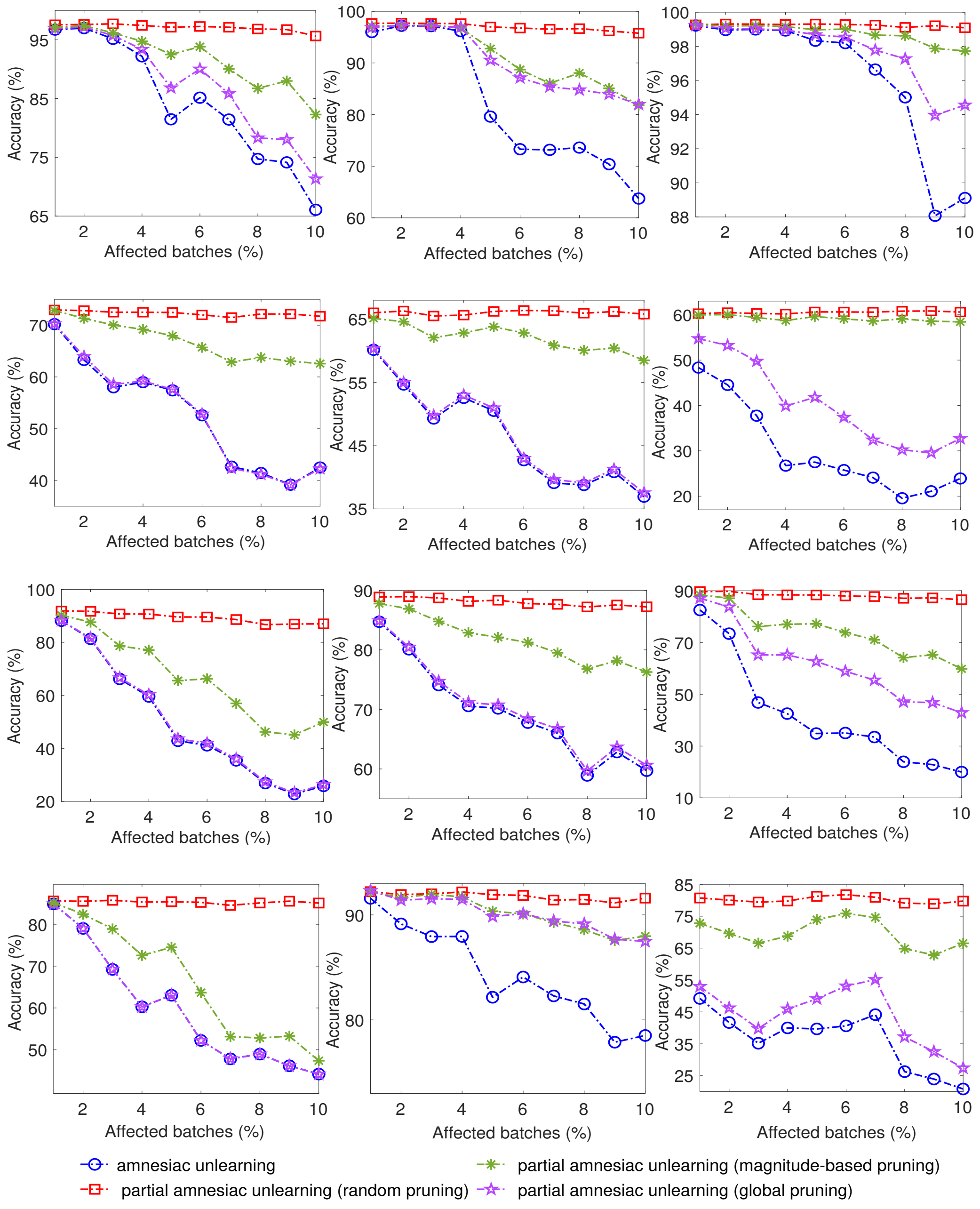

Figure 4: Comparison of model efficacy between partial amnesiac unlearning between conventional amnesiac unlearning against percentage of affected batches. 1st row: MNIST data set with MLP (left), LeNet (middle), ResNet9 (right) architectures. 2nd row: CIFAR10 data set with VGG11 (left), ResNet18 (middle), SimpleViT (right). 3rd row: SVHN data set with VGG19 (left), ResNet18 (middle), SimpleViT (right). 4th row: OrganAMNIST data set with AlexNet (left), ResNet50 (middle), ViT-Large (right)

architectures. The vertical dashed line marks the unlearning request initiation.

struggles to preserve the integrity of the knowledge that existed prior to unlearning. We can clearly see this in Figure 4.

On the other hand, magnitude-based layer-wise selective pruning targets parameters with small magnitudes in each layer, which typically represent less significant features, while retaining the important ones. However, when these pruned updates are subtracted from the trained model, important features are inadvertently erased from the model. This removal can degrade the model's ability to generalize and retain knowledge about the retained data set. For instance, when 90% pruning applied to the initial layers of model updates, magnitude-based layer-wise pruning still preserves important fundamental features in the pruned updates. As a results, subtracting these pruned updates from the trained model risks erasing important fundamental representations of the retained classes, ultimately leading to a decline in model efficacy. The performance decline associated with magnitude-based layer-wise selective pruning is evident in all plots of Figure 4.

Whereas random layer-wise selective pruning indiscriminately removes coefficients from model updates, regardless of their magnitude or importance. This means that both small and large parameters values in model updates could be pruned. As a result, the random pruning strategy can preserve the fundamental representations learned by the model, particularly in the initial layers. By selectively pruning updates, this approach minimizes disruption to the model's representation of retained data, which is crucial for maintaining overall accuracy. Therefore, as shown in Figure 4 it exhibits superior performance compared to both magnitude-based and global pruning strategies.

5.5 Evaluation of Layer-wise Partial-Updates induced Label-Flipping-based Unlearning

To evaluate the effectiveness of layer-wise partial-updates induced label-flipping-based unlearning, we presented the model accuracy on the targeted class (i.e., class 3) and specific retained classes, such as 2, 7, and 9 in MNIST, CIFAR10, classes 2, 5, and 8 in SVHN and in OrganAMNIST, classes 0, 6, and 10. The corresponding results are presented in Table 3. For comparison, we also presented the accuracies of naive label-flipping-based unlearning. To ensure a fair comparison, we tuned the hyperparameters of the naive label-flipping-based unlearning approach to achieve a targeted accuracy of zero or to match the targeted accuracy of the proposed method, particularly when the targeted accuracy slightly exceeds 1%.

From Table 3, it is evident that naive label-flipping-based unlearning effectively erases the knowledge of the targeted class from the trained models across all data sets. However, it also has a slightly negative impact on a few retained classes. For instance, there is an adverse impact on class 9 in the case of the MNIST data set with the LeNet model. Similarly, this adverse effect is more pronounced on class 5 in the case of the SVHN data set with the VGG19 model. However, leveraging layer-wise partial-updates in label-flipping-based unlearning mitigates this adverse effect and demonstrates improved performance compared to its naive counterpart, in most of the cases across all data sets and network architectures.

The reason for this is that the naive label-flipping-based unlearning approach randomly assigns labels to the target data in an attempt to modify the model. While this effectively allows the model to learn incorrect features for the targeted class thereby erase targeted

	Network Architecture	Method	$\mathcal{D}_t \downarrow$	$\mathcal{D}_r \uparrow$	Retained Classes		
					Class 2 $\uparrow$	Class 7 $\uparrow$	Class 9 $\uparrow$
MNIST	MLP	(Graves et al., 2020)	0	92.19	93.60	94.35	74.43
		Proposed	0	93.39	94.57	95.62	78.49
	LeNet	(Graves et al., 2020)	2.67	87.66	96.70	84.14	55.40
		Proposed	2.67	90.62	97.38	82.39	72.74
	ResNet9	(Graves et al., 2020)	0	97.77	97.28	99.80	91.57
		Proposed	0	98.20	97.86	99.61	93.85
					Class 2 $\uparrow$	Class 7 $\uparrow$	Class 9 $\uparrow$
CIFAR10	VGG11	(Graves et al., 2020)	0.8	57.07	44.60	68.60	52.80
		Proposed	0.8	66.31	56.00	78.9	67.60
	ResNet18	(Graves et al., 2020)	0.4	69.30	60.80	75.90	45.20
		Proposed	0.7	70.85	63.60	79.20	49.50
	SimpleViT	(Graves et al., 2020)	0.5	66.2	53.1	69.69	66.5
		Proposed	0.5	67.11	56.00	71.7	73.1
			0		Class 2 $\uparrow$	Class 5 $\uparrow$	Class 8 $\uparrow$
SVHN	VGG19	(Graves et al., 2020)	0	80.56	83.12	29.02	73.55
		Proposed	0	89.31	93.42	71.09	70.96
	ResNet18	(Graves et al., 2020)	2.7	87.23	90.50	73.65	65.66
		Proposed	2.7	88.29	91.34	73.36	70.18
	SimpleViT	(Graves et al., 2020)	0	82.83	87.58	63.46	65.30
		Proposed	0.10	84.82	91.41	69.08	63.49
					Class 0 $\uparrow$	Class 6 $\uparrow$	Class 10 $\uparrow$
OrganAMNIST	AlexNet	(Graves et al., 2020)	0.12	75.53	81.17	43.61	78.98
		Proposed	0.25	80.37	84.55	58.21	84.28
	ResNet50	(Graves et al., 2020)	0	91.85	89.38	95.83	89.73
		Proposed	0	94.15	95.75	97.26	94.10
	ViT-Large	(Graves et al., 2020)	0	83.47	88.80	93.06	63.26
		Proposed	0	84.62	89.53	92.84	69.38

Table 3: Comparison of model efficacy between the proposed layer-wise partial-updates induced label-flipping-based unlearning and its counterpart, naive label-flipping-based unlearning on targeted and retained classes of MNIST, CIFAR10, SVHN and OrganAMNIST data sets.

data knowledge, it also adversely affects the representations of the retained classes. For example, assigning class 9 to the targeted data (i.e., class 3) in MNIST case may corrupt the representation of the actual class 9. In contrast, using layer-wise partial updates mitigates this adverse effect by slowing down the modification of the retained class representations.

	Network Architecture	Method	$\mathcal{D}_t\downarrow$	$\mathcal{D}_r\uparrow$	Retained Classes		
					Class 2↑	Class 7↑	Class 9↑
MNIST	MLP	(Warnecke et al., 2021)	0	93.31	97.18	97.56	79.68
		Proposed	0	95.82	97.48	97.85	88.70
	LeNet	(Warnecke et al., 2021)	1.68	94.33	95.44	88.71	89.29
		Proposed	1.68	94.72	95.83	90.17	89.49
	ResNet9	(Warnecke et al., 2021)	0	90.84	87.98	69.94	79.68
		Proposed	0	96.75	95.54	91.34	92.07
					Class 2↑	Class 7↑	Class 9↑
CIFAR10	VGG11	(Warnecke et al., 2021)	0.80	65.62	92.80	76.70	55.40
		Proposed	0.89	72.7	88.70	81.60	66.70
	ResNet18	(Warnecke et al., 2021)	0.30	66.47	55.60	81.60	83.30
		Proposed	0.20	71.02	60.69	85.39	78.5
	SimpleViT	(Warnecke et al., 2021)	0	67.33	60.30	72.30	71.20
		Proposed	0	67.66	61.00	71.70	71.10
					Class 2↑	Class 5↑	Class 8↑
SVHN	VGG19	(Warnecke et al., 2021)	0	84.92	82.18	93.70	38.31
		Proposed	0	89.78	89.46	96.77	51.86
	ResNet18	(Warnecke et al., 2021)	0.45	80.56	95.56	89.59	43.3
		Proposed	0.41	85.69	96.04	89.68	47.04
	SimpleViT	(Warnecke et al., 2021)	0	87.77	93.03	90.18	76.62
		Proposed	0	88.00	93.13	90.52	77.40
					Class 0↑	Class 6↑	Class 10↑
OrganAMNIST	AlexNet	(Warnecke et al., 2021)	0	78.75	82.81	77.50	81.05
		Proposed	0.25	81.78	84.16	78.01	87.42
	ResNet50	(Warnecke et al., 2021)	0.5	60.93	20.65	64.37	63.90
		Proposed	0.85	71.71	40.73	69.92	77.76
	ViT-Large	(Warnecke et al., 2021)	0	81.67	94.01	89.25	78.55
		Proposed	0	83.43	93.87	91.37	85.71

Table 4: Comparison of model efficacy between the proposed layer-wise partial-updates induced optimization-based unlearning and its counterpart conventional label-flipping-based unlearning on targeted and retained classes of MNIST, CIFAR10, SVHN and OrganAMNIST, data sets.

5.6 Evaluation of Layer-wise Partial-Updates in Optimization-based Unlearning

To evaluate the impact of layer-wise partial-updates on optimization-based unlearning, we compared its model accuracy against its naive counterpart on targeted and specific retained classes of MNIST, OrganAMNIST, CIFAR10 and SVHN data sets. The corresponding results are presented in Table 4. The findings suggest that, although naive optimization-based unlearning effectively erases knowledge of the targeted data, it adversely affects the classification accuracy of certain retained classes. However, leveraging layer-wise partial-updates in optimization-based unlearning mitigates this adverse effect and demonstrates slightly better performance compared to its naive counterpart, across all data sets and network architectures.

The rationale behind the observed phenomenon is rooted in the approach of naive optimization-based unlearning, where a brief retraining is conducted on the targeted data with the objective of maximizing the loss rather than minimizing it. Consequently, the resulting model not only loses all knowledge of the targeted class but also has a detrimental impact on the fundamental representations, shared among different classes. This is particularly problematic, as it introduces unintended distortions in the learned features of the retained classes. In contrast, the utilization of layer-wise partial-updates acts as a mitigating factor against this adverse effect. This is achieved by deliberately slowing down the process of modifying the representations of retained classes through partial updates. Partial updates in optimization-based unlearning ensure that erasing knowledge of the targeted class is achieved with minimal impact on the representations of retained classes.

6. Conclusions

This paper presented novel class of machine unlearning algorithms, namely layer-wise partial machine unlearning, to address critical challenges associated with current machine unlearning methods. The proposed algorithms selectively erases knowledge from neural networks with minimal disruption to the representations learned from retained data. Unlike conventional approaches that indiscriminately remove targeted information, our approach apply controlled, layer-wise updates, enabling a finer-grained and more effective forgetting process. One of our proposed methods, partial amnesiac unlearning, integrates layer-wise pruning with amnesiac unlearning, offering a compelling approach to forget specific data from trained models. By subtracting the pruned updates, as opposed to the entire update made during the training, partial amnesiac unlearning minimizes the adverse effects on the model's behavior. Consequently, the model after partial amnesiac unlearning exhibits better performance on the retained data set, and requires very little storage space and eliminates the need for brief fine-tuning, unlike in the case of conventional amnesiac unlearning. Additionally, the integration of layer-wise partial-updates into label-flipping and optimization-based unlearning methods minimize the adverse effects on the representation of retained classes. The utilization of layer-wise partial-updates mitigates this adverse effect by slowing down the process of modifying the representations of retained classes through partial updates. The comprehensive experimental assessments across diverse data sets and neural network architectures have showcased the proficiency of the proposed class of unlearning methods in effectively erasing targeted data from trained models. Importantly, these methods have demonstrated their effectiveness to preserve model performance on retained data compared to their naive counterparts. In near future, we will explore the layer-wise structured and adaptive partial machine unlearning to erase targeted data from trained models.

References

Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua

Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 233–242. JMLR.org, 2017.

Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *Mach. Learn.*, 111(9):3203–3226, sep 2022.

Patrick Bilic, Patrick Christ, Hongwei Bran Li, Eugene Vorontsov, Avi Ben-Cohen, Georgios Kaissis, Adi Szeskin, Colin Jacobs, Gabriel Efrain Humpire Mamani, Gabriel Chartrand, Fabian Lohöfer, Julian Walter Holch, Wieland Sommer, Felix Hofmann, Alexandre Hostettler, Naama Lev-Cohain, Michal Drozdzal, Michal Marianne Amitai, Refael Vivanti, Jacob Sosna, Ivan Ezhov, Anjany Sekuboyina, Fernando Navarro, Florian Kofler, Johannes C. Paetzold, Suprosanna Shit, Xiaobin Hu, Jana Lipková, Markus Rempfler, Marie Piraud, Jan Kirschke, Benedikt Wiestler, Zhiheng Zhang, Christian Hülsemeyer, Marcel Beetz, Florian Ettlinger, Michela Antonelli, Woong Bae, Míriam Bellver, Lei Bi, Hao Chen, Grzegorz Chlebus, Erik B. Dam, Qi Dou, Chi-Wing Fu, Bogdan Georgescu, Xavier Giró i Nieto, Felix Gruen, Xu Han, Pheng-Ann Heng, Jürgen Hesser, Jan Hendrik Moltz, Christian Igel, Fabian Isensee, Paul Jäger, Fucang Jia, Krishna Chaitanya Kaluva, Mahendra Khened, Ildoo Kim, Jae-Hun Kim, Sungwoong Kim, Simon Kohl, Tomasz Konopczynski, Avinash Kori, Ganapathy Krishnamurthi, Fan Li, Hongchao Li, Junbo Li, Xiaomeng Li, John Lowengrub, Jun Ma, Klaus Maier-Hein, Kevis-Kokitsi Maninis, Hans Meine, Dorit Merhof, Akshay Pai, Mathias Perslev, Jens Petersen, Jordi Pont-Tuset, Jin Qi, Xiaojuan Qi, Oliver Rippel, Karsten Roth, Ignacio Sarasua, Andrea Schenk, Zengming Shen, Jordi Torres, Christian Wachinger, Chunliang Wang, Leon Weninger, Jianrong Wu, Daguang Xu, Xiaoping Yang, Simon Chun-Ho Yu, Yading Yuan, Miao Yue, Liping Zhang, Jorge Cardoso, Spyridon Bakas, Rickmer Braren, Volker Heinemann, Christopher Pal, An Tang, Samuel Kadoury, Luc Soler, Bram van Ginneken, Hayit Greenspan, Leo Joskowicz, and Bjoern Menze. The liver tumor segmentation benchmark (lits). *Medical Image Analysis*, 84:102680, 2023. ISSN 1361-8415.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159, 2021.

Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 1092–1104. PMLR, 18–24 Jul 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy*, pages 463–480, 2015.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM*

SIGSAC Conference on Computer and Communications Security, page 896–911. Association for Computing Machinery, 2021. ISBN 9781450384544.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 499–513, New York, NY, USA, 2022. Association for Computing Machinery.

Zhiyuan Chen, Bing Liu, Ronald Brachman, Peter Stone, and Francesca Rossi. *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2nd edition, 2018. ISBN 1681733021.

Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 1964–1974. PMLR, 18–24 Jul 2021.

Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan S. Kankanhalli. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 18: 2345–2354, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Thorsten Eisenhofer, Doreen Riepel, Varun Chandrasekaran, Esha Ghosh, Olga Ohrimenko, and Nicolas Papernot. Verifiable and provably secure machine unlearning, 2023.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, page 1322–1333. Association for Computing Machinery, 2015.

M. Godavarti and A.O. Hero. Partial update lms algorithms. *IEEE Transactions on Signal Processing*, 53(7):2382–2399, 2005.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9301–9309, 2019.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *AAAI Conference on Artificial Intelligence*, 2020.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*. JMLR.org, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, June 2016.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *ICLR*, 2021.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408. Association for Computational Linguistics, July 2023.

M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, Toronto, Canada, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10, 2021.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132, pages 931–962. PMLR, 16–19 Mar 2021.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Thanveer Shaik, Xiaohui Tao, Lin Li, Haoran Xie, Taotao Cai, Xiaofeng Zhu, and Qing Li. Framu: Attention-based machine unlearning using federated reinforcement learning, 2023.

Xinyi Sheng, Wei Bao, and Liming Ge. Robust federated unlearning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 2034–2044, 2024.

R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016.

K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. pages 1–14. Computational and Biological Learning Society, 2015a.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015b.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 3520–3532, Red Hook, NY, USA, 2017.

Lukas Struppek, Dominik Hintersdorf, Antonio De Almeida Correira, Antonia Adler, and Kristian Kersting. Plug & play attacks: Towards robust and flexible model inversion attacks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 20522–20545. PMLR, 17–23 Jul 2022.

Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan S. Kankanhalli. Fast yet effective machine unlearning. *IEEE transactions on neural networks and learning systems*, PP, 2021.

Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *ArXiv*, abs/2108.11577, 2021.

Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *ArXiv*, abs/2201.09441, 2022.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. DeltaGrad: Rapid retraining of machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 10355–10366. PMLR, 13–18 Jul 2020.

Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. Machine unlearning: A survey. *ACM Comput. Surv.*, 56(1), aug 2023.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Nature Scientific Data*, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.