Layer Attack Unlearning: Fast and Accurate Machine Unlearning via Layer Level Attack and Knowledge Distillation

Hyunjune Kim¹, Sangyong Lee², Simon S. Woo¹, 2, 3

¹ Department of Applied Data Science, Sungkyunkwan University, South Korea
² Department of Artificial Intelligence, Sungkyunkwan University, South Korea
³ Department of Computer Science and Engineering, Sungkyunkwan University, South Korea

hyunjune.kim@g.skku.edu, sang8961@g.skku.edu, swoo@g.skku.edu

Abstract

Recently, serious concerns have been raised about the privacy issues related to training datasets in machine learning algorithms when including personal data. Various regulations in different countries, including the GDPR, grant individuals the right to have personal data erased, known as ‘the right to be forgotten’ or ‘the right to erasure’. However, there has been less research on effectively and practically deleting the requested personal data from the training set while not jeopardizing the overall machine learning performance. In this work, we propose a fast and novel machine unlearning paradigm at the layer level called layer attack unlearning, which is highly accurate and fast compared to existing machine unlearning algorithms. We introduce the Partial-PGD algorithm to locate the samples to forget efficiently. In addition, we only use the last layer of the model inspired by the Forward-Forward algorithm for unlearning process. Lastly, we use Knowledge Distillation (KD) to reliably learn the decision boundaries from the teacher using soft label information to improve accuracy performance. We conducted extensive experiments with SOTA machine unlearning models and demonstrated the effectiveness of our approach for accuracy and end-to-end unlearning performance.

Introduction

Deep neural networks (DNNs) have achieved significant progress and dramatic performance gains in challenging machine learning tasks in recent years. Among others, large amounts of available training datasets have been the foundation for enabling the revolution of large-scale models. However, recently, due to the privacy concerns raised by ChatGPT (Bourtoule et al. 2021; Burgess 2023), the training dataset would contain personal information or possible information that can leak one’s privacy. For example, many vision-based applications would involve training one’s face images, which are personally identifiable information (PII). Several nations have implemented some types of regulations, such as the General Data Protection Regulation (GDPR) (Mantelero 2013) and the EU/US Copyright Law (Kaye 2022; Kublik 2019), in order to address the potential misuse of personal information and further grant individuals the right to have personal data erased, known as ‘the right to be forgotten’ or ‘the right to erasure.’ The goal of such regulations is to provide data owners the right to request and erase the personal or copyrighted data they want if they have not agreed and consented in the first place.

Therefore, companies using personal data should delete the requested data from the training set. One potential approach for corporations to mitigate the aforementioned concerns involves the exclusion of the required dataset from the training dataset, followed by a complete retraining process from scratch. Nevertheless, as models like ChatGPT get bigger and datasets grow, retraining them from scratch requires excessive computational resources and time.

Machine unlearning has emerged to tackle this challenge, allowing ML models to discard specific data selectively. (Bourtoule et al. 2021) Machine unlearning can be divided into two primary strategies: instance-wise and class-wise unlearning. The former involves forgetting knowledge related to specific instances from ML models, while the latter, which we focus on, completely removes particular classes from ML models. For example, face recognition and social media classification systems may need to erase data related to specific religion, nationality, age, disease, gender, etc., for security and privacy reasons. A few approaches (Chen et al. 2023; Cha et al. 2023) have explored the adversarial attacks for unlearning by harnessing the forgetting data’s noise to navigate the adjacent latent space. However, they used the original PGD (Madry et al. 2017) for unlearning, which can be slow.

In this work, we propose Layer Attack Unlearning, a fast and novel machine unlearning algorithm to tackle the class-wise unlearning problem. Our approach first introduces Partial-PGD, which is a new adversarial attack generation strategy to efficiently search the close vicinity of the data points to delete (See Fig. 1). Our proposed Partial-PGD is designed only to attack fully connected (classification) layer for probing the neighboring latent space to shift the forgetting data. Surprisingly, we do not utilize any feature layer information while achieving efficiency and accuracy. As shown in Fig. 1, Partial-PGD is much more efficient than the original PGD, as it can create adversarial examples only via the classification layer.

In particular, Hinton (2022)’s Forward-Forward algorithm has inspired us, and we provide the foundation of the concept of layer-based attack for machine unlearning based on the Forward-Forward. According to Hinton (2022), each
layer undergoes individualized training in the Forward-Forward algorithm to achieve its specific objectives. Similarly, in line with the Forward-Forward research, we aim to accomplish machine unlearning objectives at layers with characteristics directly relevant to data and features we want to forget. Hence, we focus on performing **machine unlearning at the layer level** rather than using the entire model. Our layer-wise unlearning approach clearly avoids unnecessary loss calculations during the unlearning process. Furthermore, updating only the layers’ weights related to forgetting data will ensure a reduction in computational costs.

Finally, we employ Knowledge Distillation (KD) (Hinton, Vinyals, and Dean 2015) to modify the decision boundary for the forgetting data and preserve the decision boundary for the retain data. The primary objective in unlearning is to utilize hard labels and acquire soft label information from the teacher model for unlearning tasks to maintain performance. We show that it achieves a stable placement of forgettable data to nearby spaces. However, a significant difference between our approach and these methods lies in the target of the attack. Our approach directs the unlearning process towards layers with specific classification objectives instead of using entire layers. Furthermore, we aim to introduce effective ways of utilizing PGD in unlearning.

### Our Approach

The main objective of our approach is to accurately and efficiently perform **class-wise** unlearning, which is to completely remove specific classes from the classification model. In this section, we describe our Partial-PGD, KD architecture, and our connection to the Forward-Forward algorithm.

**Data-Driven Unlearning Methods**

A “data-driven” approach utilizes data-centric strategies such as partitioning and augmentation (Nguyen et al. 2022) to address unlearning. SISA (Bourtoule et al. 2021) and Selective Forgetting (Shibata et al. 2021) are two representative data-driven unlearning methods. In SISA, data is divided into shard units, sequentially trained in slices, and multiple model checkpoints are created. Once an unlearning query is requested, it reverts the query to the checkpoint before learning and retrained this reverted query with the ensemble technique. However, it is challenging to calculate the probability of encountering unlearning queries on data points.

On the other hand, Selective Forgetting (Shibata et al. 2021) involves lifelong learning to perform unlearning. A “mnemonic code” signal is embedded in the data during training. During the unlearning process, the mnemonic code information is selectively incorporated into the loss function to remove forgetting data. This strategy requires storing mnemonic codes for all data points, considering unlearning queries before building the original model. This could be more practical in a real-world scenario.

**Model-Agnostic Unlearning Methods**

A “model-agnostic” approach is a methodology for handling the unlearning process by adjusting the model’s learning parameters to achieve data unlearning (Nguyen et al. 2022). Such approaches include various methods such as Summation form (Cao and Yang 2015), Negative Gradient (Golatkar, Achille, and Soatto 2020), Fisher Forgetting (Golatkar, Achille, and Soatto 2020), Boundary unlearning (Chen et al. 2023), Instance-wise Unlearning (Cha et al. 2023), etc. Some methods utilize adversarial attacks to the original model to avoid naively excluding and deleting forgetting data. Among the mentioned algorithms, approaches like ours include Boundary unlearning and Instance-wise Unlearning. These two algorithms perform unlearning by utilizing adversarial attacks to transition forgettable data to nearby spaces. However, a significant difference between our approach and these methods lies in the target of the attack. Our approach directs the unlearning process towards layers with specific classification objectives instead of using entire layers. Furthermore, we aim to introduce effective ways of utilizing PGD in unlearning.

![Figure 1: Illustration of the original PGD vs. Partial-PGD. While the original PGD involves backpropagation to compute $x_{adv}$ with respect to input $x$ for all the layers, (b) Partial-PGD computes $x_{adv}$ in $F_c$ after passing $x$ through $F_f$ to calculate $\ell$. Step in both (a) and (b) indicates the iteration.](image-url)

---

**Related Work**

There are two main approaches to the current machine unlearning problem in DNNs. The first involves considering unlearning during the learning process, while the second focuses on fine-tuning. This paper will refer to the approach that considers the learning process as “data-driven” and the approach that involves fine-tuning as “model-agnostic.”

---

21242
Figure 2: The overall procedure of our approach. Our method involves the unlearning task on the classification layer instead of the entire model, where each classification layer represents the student and the teacher model. For the unlearning task, we perform Knowledge Distillation by combining the teacher logit and student logit via the unlearned mask. The teacher logit is derived from the adversarial examples obtained after applying Partial-PGD.

**Preliminaries and Notations**

First, we formulate a machine unlearning problem as follows: We define a training dataset $D_{\text{train}} = \{x^i, y^i\}_{i=1}^N$, consisting of inputs $x^i \in X$ and their corresponding class labels $y^i \in Y$. The forgetting dataset $D_f$ is a subset of $D_{\text{train}}$ that we intend to forget from the pre-trained model. Conversely, the retain dataset $D_r = D_{\text{train}} \setminus D_f$ is the dataset we want to preserve the overall performance.

Next, we define the original model $M_{\theta}: \mathbb{R}^n \rightarrow \mathbb{R}^n$, which comprises a set of feature layers denoted by $F^i_{\theta}: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and a fully connected layer denoted as $F^f_{\theta}: \mathbb{R}^n \rightarrow \mathbb{R}^n$, where $\theta$ represents the optimal parameters for the model trained on $D_{\text{train}}$. The following provides a compositional representation of the model $M_{\theta}$ as $F^f_{\theta} \circ F^i_{\theta}$. Also, we denote $x^{\text{adv}}$ to represent the adversarial examples (Goodfellow, Shlens, and Szegedy 2014) for the input data $x$. In particular, we define $\ell^{\text{adv}}$ as the adversarial example from Partial-PGD, generated from the intermediate latent feature $\ell$ obtained from the outputs of $F^i_{\theta}$, as shown in Fig. 1.

**Partial-Projected Gradient Descent (PGD)**

The main reason for employing adversarial examples is to search and identify neighboring candidate spaces more effectively that will assign the forgetting data samples. Assigning forgetting classes to random or irrelevant classes can dramatically reduce downstream task performance.

Therefore, carefully exploring the neighboring space allows us not only to forget $D_f$ but also to preserve the decision boundary of other classes. Hence, adversarial attacks (Madry et al. 2017; Chen et al. 2023) can be explored below:

$$x^{t+1} = \Pi(x^t + (\epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, y, \theta))))$$  \hspace{1cm} (1)

where the parameter $\theta$ represents the weights of the target model under attack, and generated noise for crafting adversarial examples is produced by computing the gradient $\nabla_x \mathcal{L}$ of the loss function $\mathcal{L}$ with respect to the input $x$. This noise is added to $x^t$ and then projected using the projection method $\Pi$ to calculate $x^{t+1}$, which is repeated $t$ times. Once $x^{t+1}$ represents $x^{\text{adv}}$, it is an adversarial example.

However, we clarify the purpose of adversarial examples used in our work, which differs from prior approaches. The original PGD approach may generate excessive noise and slow the unlearning process considerably. Therefore, there is no need to calculate gradients throughout the entire model to create adversarial examples.

Hence, our proposed Partial-PGD utilizes $F^i_{\theta}$ to generate adversarial examples for the unlearning process, as shown in Fig. 1. This technique effectively identifies the neighboring space to allocate $D_f$, the forgetting data, similar to conventional PGD. However, it significantly reduces unlearning time by omitting feature layer information, as depicted in Fig. 1. We define our Partial-PGD as follows:

$$\ell^{t+1} = \Pi(\ell^t + (\epsilon \cdot \text{sign}(\nabla_\ell \mathcal{L}(\ell, y, \theta))))$$ \hspace{1cm} (2)

where Partial-PGD applies an adversarial attack to the intermediate latent $\ell$ obtained from $F^i_{\theta}$, where $\ell$ undergoes gradient computation based solely on passing through $F^i_{\theta}$. Then, the result is mapped to the nearby space of a different label of $\ell$ and becomes $\ell^{\text{adv}}$, which we use for unlearning $D_f$ as knowledge to be forgotten.

**Layer Unlearning**

While other approaches use entire layers for unlearning, we focus on unlearning only the relevant layers. Inspired by the Forward-Forward technique, we focus on the classification layer $F^f_{\theta}$ to forget specific classes in the model for class-wise unlearning. Therefore, our layer unlearning focuses on only modifying the parameters of $F^f_{\theta}$ tied to classification instead of the entire layers and model $M_{\theta}$ to forget $D_f$ effectively.

We define the following equation to describe our unlearning process, where we focus on the $F^f_{\theta}$ during the unlearning process to remove $D_f$ from the model:

$$M_{\theta^*} = F^f_{\theta^*} \circ F^i_{\theta^*}$$ \hspace{1cm} (3)

where $\theta^*$ is the ideal parameters after forgetting $D_f$. 


We show that layer unlearning accelerates the unlearning process by selectively updating relevant layer weights and optimizing efficiency. Interestingly, it outperforms models with whole layers in accuracy.

End-to-End Unlearning Process

We describe our end-to-end unlearning process, where we apply the KD to improve the overall performance further. As illustrated in Fig. 2, the classification layer $F^\theta_\theta$ serves as our student model $S_\theta$. Additionally, at the beginning of each epoch, we duplicate the $S_\theta$ as our teacher $T_\theta$. The model uses forgetting data $D_f$ as input to create an intermediate latent feature $\ell_f$ through the feature layer $F^\theta_f$. Then, $\ell_f$ becomes an adversarial example $\ell^adv_f$ after applying a Partial-PGD on the $T_\theta$.

Next, $\ell_f$ and $\ell^adv_f$ are passed through $S_\theta$ and $T_\theta$, respectively, becoming logits for each student and teacher, as shown in Fig. 2. Then, the logit obtained from $S_\theta$ is compared with the ground truth $y_f$. If a discrepancy is observed, it is considered unlearned. Then, the unlearned logit replaces the adversarial logit from $T_\theta$. This student’s logit is used to compute the cross-entropy loss as follows:

$$L_{CE} = \begin{cases} CE(S_\theta(\ell_f), y^adv_f) & \text{if } y_{S_\theta} = y_f \\ CE(S_\theta(\ell_f), y_{S_\theta}) & \text{otherwise} \end{cases}$$

where $y_{S_\theta}$ represents the predicted label from $S_\theta(\ell_f)$, and CE is the cross-entropy function. This loss leaves the unlearned data in a state, where it makes wrong (unlearned) predictions. If not, it is trained to be a predicted label $y^adv_f$ of adversarial logit, leading to its unlearning process. Next, let $Z$ be the double Softmax representation, which is defined as:

$$Z = \begin{cases} \sigma(T_\theta(\ell^adv_f)) & \text{if } y_{S_\theta} = y_f \\ \sigma(S_\theta(\ell_f)) & \text{otherwise} \end{cases}$$

where $\sigma$ represents Softmax function. In Eq. 5, we performed double Softmax to distill knowledge by adjusting the probability distribution of the output from $T_\theta$. This approach is intended to convey soft label information to $S_\theta$. Exclusively unlearning $F^\theta_f$ maintains the decision boundaries of retain data, and slightly improves the overall accuracy. But, layer unlearning without double Softmax showed variable accuracy, as shown in the Fashion-MNIST dataset (Xiao, Rasul, and Vollgraf 2017). We show this effect in Section .

Next, we define our distillation loss as follows:

$$L_{DI} = KL \left( \sigma \left( \frac{S_\theta(\ell_f)}{T} \right), \sigma \left( \frac{Z}{T} \right) \right)$$

where knowledge is distilled from $Z$ of $T_\theta$ and KL is the KL divergence. During distillation, the computation of loss $L_{DI}$ between the outputs of $S_\theta$ and $T_\theta$ focuses on creating a similar boundary to the teacher model, ensuring performance while removing information of $D_f$. The temperature $T$ is a hyper-parameter. Generally, increasing $T$ will generate smoother soft labels that assists $S_\theta$ in mimicking $T_\theta$. The effects of changes in $T$ are described in Suppl. Mat.

Algorithm 1: End-to-End Unlearning Process

Input: $F^\theta_\theta, F^\theta_f, D_f$
Parameter: Learning rate $\eta$, Hyper-parameters $\alpha$, Temperature $T$, Number of Epochs $E$
Output: $M_{\theta^*}$

1. $S_\theta \leftarrow F^\theta_\theta$
2. $\theta^* \leftarrow \theta$
3. for $e$ in range $E$ do
4. $T_\theta \leftarrow S_\theta$.
5. $L \leftarrow (1 - \alpha) \cdot L_{CE} + \alpha \cdot T^2 \cdot L_{DI}$
6. $\theta^* \leftarrow \theta^* - \eta \cdot L$
7. if $F^\theta_\theta \circ F^\theta_f(X_f)! = Y_f$ then
8. break
9. end if
10. end for
11. $M_{\theta^*} \leftarrow F^\theta_\theta \circ F^\theta_f$
12. return $M_{\theta^*}$

Using $L_{CE}$ and $L_{DI}$, our final loss function is constructed as follows:

$$L = (1 - \alpha) \cdot L_{CE} + \alpha \cdot T^2 \cdot L_{DI},$$

where the value of $\alpha$ represents the weight assigned to the loss between $L_{CE}$ and $L_{DI}$. As a hyper-parameter, $\alpha$ ranges from 0 to 1. Assigning additional weight to $L_{CE}$ may boost unlearning time but decrease performance. Conversely, if we provide more weight to $L_{DI}$, the unlearning speed may slow down but can increase accuracy. We conducted the ablation study for $\alpha$ values to capture the trade-off. The effects of changes in the exponent of $T^2$ are described in Suppl. Mat.

In addition, we provide the end-to-end unlearning process in Alg. 1. We distill knowledge from $T_\theta$, while gradually reducing boundaries. Algorithm 1 finishes either when all epochs are completed or when $D_f$ becomes unlearned within a batch during an epoch. Finally, we obtain our unlearning model $M_{\theta^*}$ by combining $F^\theta_\theta$ with the classification layer, $F^\theta_f$, as shown in Eq. 3.

Summary. In Fig. 3, we pictorially describe our end-to-end unlearning process by displaying the boundary change for the retain and forgetting data.

![Figure 3: Boundary evolution in the unlearning process. As shown in (a), the original model receives the initial knowledge about the boundary. As the epoch progresses, the boundary information updates as depicted in (b) and (c) from the distilled knowledge.](image-url)
### Experimental Results

We experiment and evaluate popular unlearning benchmarks used in other unlearning research (Golatkar, Achille, and Soatto 2020; Chen et al. 2023; Cha et al. 2023) on image classification tasks.

**Datasets and Models.** We conducted experiments on CIFAR-10 (Krizhevsky, Hinton et al. 2009), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), and VGGFace2 (Cao et al. 2018) datasets. For the VGGFace2 dataset, we randomly select ten individuals from a training dataset containing over 600 images, ensuring a balanced gender distribution. Furthermore, we perform training from scratch for three different architectures: VGG16 (Simonyan and Zisserman 2014), ResNets (He et al. 2016), and ViT (Dosovitskiy et al. 2020).

**Baseline Approaches.** The subsequent unlearning baseline methods are used:

1. **Original:** We train the model on the $D_{\text{train}}$ dataset before undergoing the unlearning process.
2. **Retrain:** We train the model from scratch utilizing $D_r$ as the retrained model, an optimal unlearning strategy.
3. **Negative Gradient (NG)** (Golatkar, Achille, and Soatto 2020): We fine-tune the **Original** with $D_f$ by following the direction of gradient ascent.
4. **Fine-tune (FT)** (Golatkar, Achille, and Soatto 2020): We fine-tune the **Original** using $D_f$ with a large learning rate.
5. **Random Label (RL)** (Golatkar, Achille, and Soatto 2020): We fine-tune the **Original** by assigning arbitrary labels randomly to $D_f$.
6. **Fisher Forgetting (FF)** (Golatkar, Achille, and Soatto 2020): The Fisher Forgetting model identifies influential parameters significantly affecting $D_f$ and then introduces noise to neutralize their impact.

7. **Boundary Shrink (BS)** (Chen et al. 2023): We create adversarial examples from $D_f$ and assign new adversarial labels to shrink towards different classes.

8. **IWU** ( Cha et al. 2023): Generating adversarial instances for distinct labels via $D_f$ and incorporating a regularization term. While initially designed for instance-wise learning, we adapt this method for class-wise unlearning problems.

**Implementation Details and Evaluation Metrics.** Our method and other baselines are implemented in Python 3.7 and use the PyTorch library (Paszke et al. 2019), employing a single NVIDIA GeForce RTX 3090 GPU. The initial model was trained using an LR scheduler and an SGD optimizer with specific settings (momentum: 0.9, weight decay: $5 \times 10^{-4}$, initial learning rate: 0.01). For the unlearning phase, we employ the SGD optimizer and conduct experiments with varying learning rates (ranging from 0.001 to 0.01), KD $\alpha$ values (ranging from 0.3 to 0.7), KD temperature $T$ value (fixed at 4), and Partial-PGD values (ranging from 0.4 to 1.0). As defined, $D_f$ and $D_r$ represent the forgetting and retain data, respectively. Additionally, $D_f^{0.5}$ corresponds to the test forgetting data, and $D_r^{0.5}$ represents the test retain data. We assess the accuracy of all four different datasets.

**Accuracy Performance.** To achieve the best unlearning performance, it should completely forget information related to $D_f$. Therefore, guaranteeing accuracy on a pair with those achieved by the **Retrain** for both $D_f$ and $D_r$ will be the best. Table 1 presents
test results from different classification models, datasets, and metrics. The tested models include VGG16, ResNet18, ResNet50, and ViT. The datasets used for testing were CIFAR-10, Fashion-MNIST, and VGGFace2. In addition to the accuracy metric, we evaluate the performance using the unlearning score (US) as follows:

$$\text{US}(\text{acc}_r, \text{acc}_f) = \frac{\exp(\text{acc}_r/100) + \exp(1 - \text{acc}_f/100) - 2}{2 \cdot (\exp(1) - 1)},$$

(8)

where $\text{acc}_r$ and $\text{acc}_f$ denote the accuracy of the retain and forgetting dataset, respectively. If the $D_{tr}$ approaches 100% and $D_{tf}$ approaches 0%, the US metric approaches 1, indicating a stable result on the unlearning process. We provide a more detailed explanation of why this metric is useful for unlearning in Suppl. Mat.

Finally, Table 1 presents the performance of each unlearning method for a specific single class in the aforementioned datasets. We measure the accuracy for datasets $D_r$, $D_f$, $D_{tr}$, and $D_{tf}$, along with the metric US. For the Negative Gradient, the unstable variability in the loss of negative gradient contributes to less favorable overall performance results. Fine-tuning shows strong performance in forgetting and retaining information. Nevertheless, this methodology requires utilizing the complete dataset $D_r$ during training. Such extensive data is time-consuming, and we analyze and compare their worse time performance in Table 2. In the case of Random Label, except for VGGFace2’s ResNet18, most cases have poor accuracy and US. Due to the random nature of forgetting, converging towards arbitrary labels in the classification space is challenging, resulting in low performance.

Fisher Forgetting exhibits poor performance, with low accuracy and US on the overall test. Also, the Fisher matrix information required a significant amount of time. For Boundary Shrink, they also utilized adversarial attack examples, but they used the hard label information of the attack examples on $D_f$, which resulted in an unstable unlearning process. IWU approach involves utilizing adversarial attack examples while incorporating regularization to achieve a stable unlearning process. However, this gains an average US of 0.859 in the overall test.

Finally, Ours completely removes the forgetting dataset (0% accuracy) on all the test cases and retains the highest unlearning performance. The accuracy for both $D_f$ and $D_{tf}$ reaches 0, while the accuracy for $D_r$ and $D_{tr}$ is comparable to or sometimes even higher than the Retrain. Also, ours demonstrates superior performance compared to almost all baseline models across various scenarios, with a high US average of 0.944. Our approach that utilizes Partial-PGD and KD-based unlearning processes on layers with explicit objectives clearly achieves the best unlearning performance.

### Data Usage & Time Performance

Table 2 presents each method’s elapsed time and data usage on CIFAR-10. The Retrain, Fisher Forgetting, and Fine-tuning leverage the entire $D_r$ dataset, resulting in significant time costs for unlearning. Including our method, the rest of the unlearning methods utilize only $D_f$. In the case of Fisher Forgetting, it takes a longer time than the Retrain, and its unlearning performance is significantly poor. While the Fine-tune exhibits favorable unlearning performance, it comes with the drawback of consuming a considerable time. However, our method showcases optimal unlearning performance, while consuming only 3.76 seconds in the quickest scenario. To summarize, our approach exhibits higher efficiency, compared to competing methods.

### Ablation Study

We performed several different ablation experiments to analyze and show the benefits of our approach.

**Original PGD vs. Partial-PGD.** Table 3 compares unlearning performance when applying the original PGD vs. Partial-PGD within our method on the CIFAR-10 dataset. While the original PGD yields high unlearning performance, Partial-PGD indicates even superior outcomes. Notably, Partial-PGD accelerates the unlearning process by up to nearly tenfold compared to the original PGD.

**Double Softmax.** In our technique, the teacher logits undergo a softmax function before being integrated into the

![Image](image-url)

**Figure 4:** Impact of hyper-parameter $\alpha$ in Knowledge Distillation vs. Accuracy on CIFAR-10 with ResNet18.
distillation loss. We have coined this method “Double Softmax”, where Double Softmax enhances the robustness of our method across diverse datasets and models. And, Table 4 presents unlearning performance with and without double Softmax in our methods on the Fashion-MNIST dataset.

Data Usage Ratio. The class-specific $D_f$ dataset for one class in CIFAR-10 contains 5,000 samples. As shown in Table 5, we reduced the dataset size to 50% (2,500) and 10% (500) for each model to perform the unlearning task. We measure the accuracy, US, and execution time of $D_{tr}$, $D_{tf}$. In the following scenario, all models completed the unlearning for 2,500 samples, but ViT still had 0.1% retaining for 500 samples. The execution speed increases as the size of $D_f$ decreases. Our experiment shows the potential for achieving superior unlearning performance by focusing on critical subsets of $D_f$ rather than employing the complete dataset, saving time nearly seven times.

Hyper-parameter $\alpha$ in KD. As shown in Fig. 4, we examine the accuracy variation of $D_{tr}$ and $D_{tf}$ with respect to changes in the hyper-parameter $\alpha$ in Eq. 6. As the $\alpha$ approaches zero, it exclusively prioritizes the removal of $D_f$ without taking into account any information from $D_r$. Consequently, the information about $D_{tf}$ is completely removed, resulting in a decrease in the accuracy of $D_{tr}$. As $\alpha$ approaches one, heavily relying on the teacher model for retaining information increases $D_{tf}$ accuracy, indicating ineffective unlearning. Therefore, selecting the appropriate $\alpha$ value can maximize unlearning performance. Consequently, we used $\alpha$ ranging from 0.4 to 0.6 in this work. In more detail, the effects of changes in $\alpha$ are described in Suppl. Mat.*

<table>
<thead>
<tr>
<th>Model</th>
<th>VGG16</th>
<th>ResNet18</th>
<th>ResNet50</th>
<th>ViT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Usage</td>
<td>2,500</td>
<td>500</td>
<td>2,500</td>
<td>500</td>
</tr>
<tr>
<td>$D_{tr}$</td>
<td>92.42</td>
<td>92.38</td>
<td>93.51</td>
<td>93.38</td>
</tr>
<tr>
<td>$D_{tf}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>US</td>
<td>0.94</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>Time</td>
<td>1.91</td>
<td>1.21</td>
<td>2.28</td>
<td>1.45</td>
</tr>
</tbody>
</table>

Table 5: The changes in time and accuracy performance with the reduction in $D_f$ data on the CIFAR-10.

Visualization on Decision Boundary
Figure 5 presents the Original, Retrain, and Ours using t-SNE on the CIFAR-10 dataset. The red dots represent samples of ship images, indicated as $D_f$. As shown in Fig. 5(b), $D_f$ was totally misclassified in the Retrain. On the other hand, Our unlearning method produces results correctly, as shown in Fig. 5(c), where the decision boundary of $D_f$ has been successfully absorbed into the surrounding space.

Conclusion
In this paper, we introduce a novel and fast machine unlearning algorithm, layer attack unlearning, which is the new layer-based unlearning paradigm. Our work proposes Partial-PGD, layer unlearning method, and KD end-to-end framework to improve the overall accuracy performance while completely removing the forgetting dataset. Through extensive experimental evaluations, we demonstrated that modifying only specific layers’ learning objectives can lead to successful unlearning. Our approach effectively decreases both the number of parameters and their updates (computational cost), consequently reducing the overall time required for unlearning. We believe our layer attack unlearning paves a new way for future research in effectively addressing various unlearning challenges.
Acknowledgments

The authors would thank anonymous reviewers. Simon S. Woo is the corresponding author. This work was partly supported by Institute for Information & communication Technology Planning & evaluation (IITP) grants funded by the Korean government MSIT: (No. 2022-0-01199, Graduate School of Convergence Security at Sungkyunkwan University), (No. 2022-0-01045, Self-directed Multi-Modal Intelligence for solving unknown, open domain problems), (No. 2022-0-00688, AI Platform to Fully Adapt and Reflect Privacy-Policy Changes), (No. 2021-0-00421, AI Graduate School Support Program at Sungkyunkwan University), and (No. RS-2023-0023037, Advanced and Proactive AI Platform Research and Development Against Malicious Deepfakes).

References


