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

Hyunjune Kim[1], **Sangyong Lee**[2], **Simon S. Woo**[1, 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 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 forgetting data in the space subjected to carefully created adversarial examples. We incorporated KD into our final loss function to improve performance.

Our main contributions are summarized as follows:

- We introduce Layer Attack Unlearning (LAU) algorithm, which is a novel and fast unlearning method by proposing Partial-PGD and performing unlearning at the layer level.
- In addition, we propose KD method to further improve the overall accuracy and data erasure performance by effectively distilling the decision boundary knowledge from the teacher model for unlearning task.
- Our extensive experimental results with seven baselines with four different backbones, including ViT over three other datasets, show that our approach outperforms previous SOTA methods in accuracy and time performance while completely forgetting the requested class.

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."

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 retrains 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.

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.

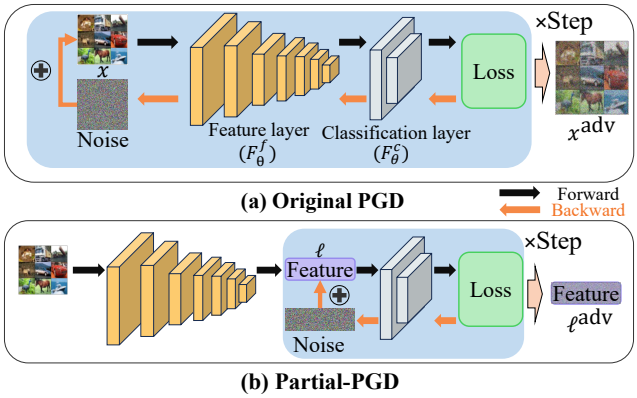


Figure 1: Illustration of the original PGD vs. Partial-PGD. While the original PGD involves backpropagation to compute x^{adv} with respect to input x for all the layers, (b) Partial-PGD computes ℓ^{adv} in $\mathcal{F}_\theta^c$ after passing x through $\mathcal{F}_\theta^f$ to calculate ℓ. Step in both (a) and (b) indicates the iteration.

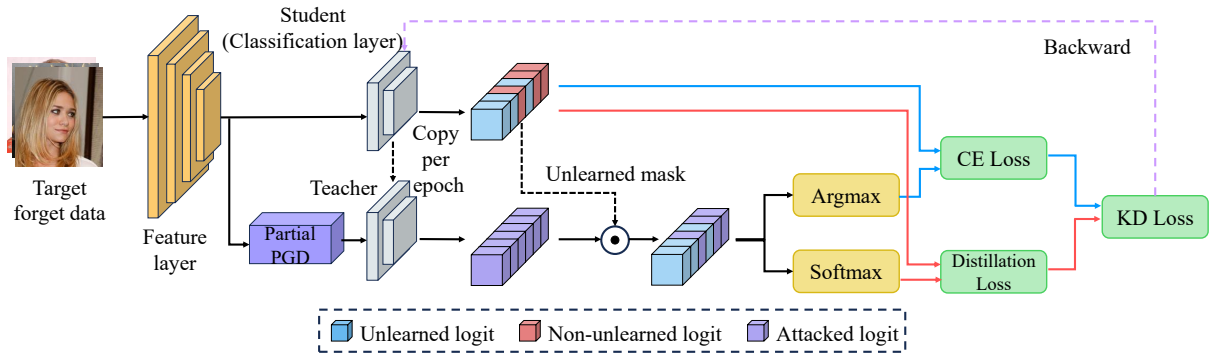


Figure 2: The overall procedure of our approach. Our method involves the unlearning task on the classification layer instead of the entire model, where each classification layer represents the student and the teacher model. For the unlearning task, we perform Knowledge Distillation by combining the teacher 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_{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 $\mathcal{M}_\theta : \mathbb{R}^n \to \mathbb{R}^n$, which comprises a set of feature layers denoted by $\mathcal{F}_\theta^f : \mathbb{R}^n \to \mathbb{R}^n$ and a fully connected layer denoted as $\mathcal{F}_\theta^c : \mathbb{R}^n \to \mathbb{R}^n$, where θ represents the optimal parameters for the model trained on D_{train}. The following provides a compositional representation of the model $\mathcal{M}_\theta$ as $\mathcal{F}_\theta^c \circ \mathcal{F}_\theta^f$. Also, we denote x^{adv} to represent the adversarial examples (Goodfellow, Shlens, and Szegedy 2014) for the input data x. In particular, we define ℓ^{adv} as the adversarial example from Partial-PGD, generated from the intermediate latent feature ℓ obtained from the outputs of $\mathcal{F}_\theta^f$, 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:

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

where the parameter θ 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 Π to calculate x^{t+1}, which is repeated t times. Once x^{t+1} represents x^{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 $\mathcal{F}_\theta^c$ 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:

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

where Partial-PGD applies an adversarial attack to the intermediate latent ℓ obtained from $\mathcal{F}_\theta^f$, where ℓ undergoes gradient computation based solely on passing through $\mathcal{F}_\theta^c$. Then, the result is mapped to the nearby space of a different label of ℓ and becomes ℓ^{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 $\mathcal{F}_\theta^c$ to forget specific classes in the model for class-wise unlearning. Therefore, our layer unlearning focuses on only modifying the parameters of $\mathcal{F}_\theta^c$ tied to classification instead of the entire layers and model $\mathcal{M}_\theta$ to forget D_f effectively.

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

$$\mathcal{M}_{\theta^*} = \mathcal{F}_{\theta^*}^c \circ \mathcal{F}_\theta^f, \tag{3}$$

where θ^* 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 $\mathcal{F}_\theta^c$ serves as our *student* model $\mathcal{S}_\theta$. Additionally, at the beginning of each epoch, we duplicate the $\mathcal{S}_\theta$ as our *teacher* $\mathcal{T}_\theta$. The model uses forgetting data D_f as input to create an intermediate latent feature ℓ_f through the feature layer $\mathcal{F}_\theta^f$. Then, ℓ_f becomes an adversarial example ℓ_f^{adv} after applying a Partial-PGD on the $\mathcal{T}_\theta$.

Next, ℓ_f and ℓ_f^{adv} are passed through $\mathcal{S}_\theta$ and $\mathcal{T}_\theta$, respectively, becoming logits for each student and teacher, as shown in Fig. 2. Then, the logit obtained from $\mathcal{S}_\theta$ 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 $\mathcal{T}_\theta$. This student's logit is used to compute the cross-entropy loss as follows:

$$\mathcal{L}_{CE} = \begin{cases} CE(\mathcal{S}_\theta(\ell_f), y_f^{\text{adv}}) & \text{if } y_{\mathcal{S}_\theta} = y_f \\ CE(\mathcal{S}_\theta(\ell_f), y_{\mathcal{S}_\theta}) & \text{otherwise,} \end{cases} \tag{4}$$

where $y_{\mathcal{S}_\theta}$ represents the predicted label from $\mathcal{S}_\theta(\ell_f)$, 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_f^{adv} of adversarial logit, leading to its unlearning process. Next, let Z be the double Softmax representation, which is defined as:

$$\text{Z} = \begin{cases} \sigma(\mathcal{T}_\theta(\ell_f^{\text{adv}})) & \text{if } y_{\mathcal{S}_\theta} = y_f \\ \sigma(\mathcal{S}_\theta(\ell_f)) & \text{otherwise,} \end{cases} \tag{5}$$

where σ represents Softmax function. In Eq. 5, we performed double Softmax to distill knowledge by adjusting the probability distribution of the output from $\mathcal{T}_\theta$. This approach is intended to convey soft label information to $\mathcal{S}_\theta$. Exclusively unlearning $\mathcal{F}_\theta^c$ 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:

$$\mathcal{L}_{DI} = \text{KL}\left(\sigma(\frac{\mathcal{S}_\theta(\ell_f)}{T}), \sigma(\frac{\text{Z}}{T})\right), \tag{6}$$

where knowledge is distilled from Z of $\mathcal{T}_\theta$ and KL is the KL divergence. During distillation, the computation of loss $\mathcal{L}_{DI}$ between the outputs of $\mathcal{S}_\theta$ and $\mathcal{T}_\theta$ 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 $\mathcal{S}_\theta$ in mimicking $\mathcal{T}_\theta$. The effects of changes in T are described in Suppl. Mat.*

*The full paper with Suppl. Mat. is on arXiv:2312.16823.

Algorithm 1: End-to-End Unlearning Process

Input: $\mathcal{F}_\theta^f$, $\mathcal{F}_\theta^c$, D_f
Parameter: Learning rate η, Hyper-parameters α, Temperature T, Number of Epochs E
Output: $\mathcal{M}_{\theta^*}$

$\mathcal{S}_\theta \leftarrow \mathcal{F}_\theta^c$
$\theta^* \leftarrow \theta$
for e in range E **do**
  $\mathcal{T}_{\theta^*} \leftarrow \mathcal{S}_{\theta^*}$
  $\mathcal{L} \leftarrow (1-\alpha)\cdot\mathcal{L}_{CE} + \alpha\cdot T^2\cdot\mathcal{L}_{DI}$
  $\theta^* \leftarrow \theta^* - \eta\cdot\mathcal{L}$
  if $\mathcal{F}_{\theta^*}^c \circ \mathcal{F}^f(X_f)! = Y_f$ **then**
    break
  end if
end for
$\mathcal{M}_{\theta^*} \leftarrow \mathcal{F}_{\theta^*}^c \circ \mathcal{F}_\theta^f$
return $\mathcal{M}_{\theta^*}$

Using $\mathcal{L}_{CE}$ and $\mathcal{L}_{DI}$, our final loss function is constructed as follows:

$$\mathcal{L} = (1-\alpha)\cdot\mathcal{L}_{CE} + \alpha\cdot T^2\cdot\mathcal{L}_{DI}, \tag{7}$$

where the value of α represents the weight assigned to the loss between $\mathcal{L}_{CE}$ and $\mathcal{L}_{DI}$. As a hyper-parameter, α ranges from 0 to 1. Assigning additional weight to $\mathcal{L}_{CE}$ may boost unlearning time but decrease performance. Conversely, if we provide more weight to $\mathcal{L}_{DI}$, the unlearning speed may slow down but can increase accuracy. We conducted the ablation study for α 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 $\mathcal{T}_\theta$, 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 $\mathcal{M}_{\theta^*}$ by combining $\mathcal{F}_\theta^f$ with the classification layer, $\mathcal{F}_{\theta^*}^c$, 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.

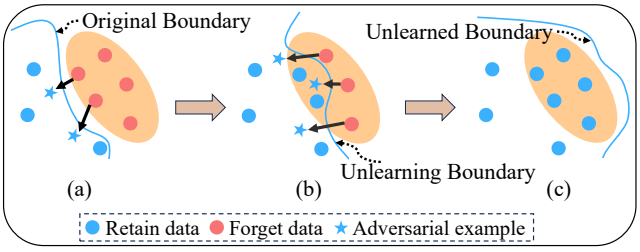


Figure 3: Boundary evolution in the unlearning process. As shown in (a), the original model receives the initial knowledge about the boundary. As the epoch progresses, the boundary information updates as depicted in (b) and (c) from the distilled knowledge.

	Model	VGG16					ResNet18					ResNet50					ViT				
	Metrics	D_r	D_f	D_{tr}	D_{tf}	US	D_r	D_f	D_{tr}	D_{tf}	US	D_r	D_f	D_{tr}	D_{tf}	US	D_r	D_f	D_{tr}	D_{tf}	US
CIFAR-10	Original	99.98	100	92.07	96.70	0.449	99.98	100	93.13	96.60	0.458	99.94	99.96	93.44	95.0	0.465	88.06	93.52	81.48	88.40	0.402
	Retrain	99.89	0	91.98	0	0.939	99.79	0	92.50	0	0.943	99.77	0	92.48	0	0.943	95.0	0	81.0	0	0.863
	NG	88.53	16.96	79.86	17.0	0.732	93.85	28.38	86.30	25.54	0.720	88.75	24.77	82.52	23.30	0.709	85.26	18.69	79.74	16.7	0.733
	FT	99.63	0	90.09	0	0.925	99.63	0	91.25	0	0.934	99.45	0	90.79	0	0.930	90.96	1.77	82.43	1.62	0.860
	RL	80.99	3.56	72.40	3.69	0.781	91.38	11.09	84.00	10.98	0.801	81.30	12.91	76.62	11.84	0.747	77.58	15.10	73.42	14.38	0.709
	FF	46.78	55.24	44.61	52.30	0.341	59.0	52.34	55.57	52.2	0.395	58.17	58.06	55.95	56.20	0.378	42.68	66.34	43.34	62.30	0.291
	BS	90.73	10.16	81.53	9.58	0.794	95.88	9.75	87.91	10.24	0.833	86.03	3.94	80.09	3.46	0.830	85.22	0.61	79.29	0.28	0.850
	IWU	90.81	0	82.35	0.10	0.871	89.41	0	82.55	0	0.873	86.11	0	79.98	0	0.856	82.48	3.92	77.01	2.58	0.817
	Ours	99.97	0	92.18	0	**0.941**	99.97	0	93.53	0	**0.950**	99.92	0	93.52	0	**0.950**	87.51	0	81.14	0	**0.864**
Fashion-MNIST	Original	99.83	100	94.38	99.60	0.458	98.45	99.96	94.71	99.70	0.460	98.49	99.98	94.68	99.6	0.460	91.27	98.71	88.28	97.10	0.421
	Retrain	100	0	93.40	0	0.949	100	0	93.38	0	0.949	100	0	93.28	0	0.949	89.44	0	86.76	0	0.902
	NG	97.77	0	92.63	0	0.944	92.57	1.39	90.04	0.84	0.918	84.44	12.63	81.42	10.22	0.789	71.77	0.10	70.38	0.10	0.796
	FT	99.67	0	93.07	0	0.947	97.23	0	91.93	0	0.939	98.83	0	92.85	0	0.945	96.08	0.01	88.72	0.10	**0.915**
	RL	98.17	8.34	92.43	23.55	0.776	76.80	11.47	74.80	11.54	0.736	75.99	10.77	73.73	10.72	0.737	84.18	11.36	82.10	13.04	0.774
	FF	62.33	28.81	60.32	28.10	0.547	72.78	57.65	71.03	54.10	0.471	60.59	84.01	60.25	82.60	0.296	43.42	88.01	42.60	86.3	0.197
	BS	86.88	1.47	81.66	1.12	0.859	95.78	34.54	92.31	32.40	0.723	83.50	30.23	80.60	27.08	0.673	70.31	2.04	68.74	2.70	0.767
	IWU	99.09	0	93.68	0	0.952	93.82	0	90.80	0	0.930	80.17	0	77.94	0	0.843	82.85	0	81.21	0	0.865
	Ours	99.51	0	93.89	0	**0.953**	97.98	0	94.54	0	**0.958**	98.14	0	94.48	0	**0.956**	90.11	0	87.44	0	0.907
VGGFace2	Original	100	100	96.67	98.41	0.479	100	100	95.88	98.41	0.473	99.12	98.43	93.67	100	0.451	94.71	96.86	95.43	93.82	0.483
	Retrain	99.98	0	96.67	0	0.974	100	0	96.20	0	0.971	99.10	0	94.77	0	0.960	92.63	0	93.32	0	0.947
	NG	96.85	15.67	90.50	4.76	0.892	97.32	9.75	89.55	12.69	0.827	86.80	4.73	78.79	3.17	0.824	91.16	1.63	92.34	0	0.942
	FT	97.86	0	89.87	0	0.942	91.42	0	85.91	0	0.896	95.18	0	90.03	0	0.925	96.91	1.63	84.85	3.70	0.860
	RL	90.32	1.74	79.11	1.58	0.838	96.76	6.44	87.34	0	0.906	88.24	13.19	82.43	9.52	0.801	92.06	9.68	91.04	8.64	0.867
	FF	46.24	31.01	42.72	50.79	0.340	72.78	57.65	71.03	54.10	0.471	76.28	4.52	71.83	7.93	0.746	60.80	71.07	53.58	60.49	0.347
	BS	99.48	17.25	93.04	5.36	0.906	94.02	5.40	86.08	5.36	0.856	93.85	5.36	85.78	5.0	0.857	86.92	6.46	86.81	4.25	0.869
	IWU	99.21	10.80	94.46	4.76	0.865	75.23	0.17	69.77	0	0.794	78.62	0	69.14	0	0.790	76.25	0.27	78.66	0	0.848
	Ours	99.70	0	96.70	0	**0.974**	99.79	0	95.34	0	**0.964**	97.46	0	93.28	0	**0.949**	95.18	0	95.50	0	**0.965**

Table 1: Accuracy and Unlearning Score (US) performance on the CIFAR-10, Fashion-MNIST and VGGFace2 datasets. Bold font highlights the highest performing results, while underlining indicates the second-best performance.

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_{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_r 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: 5×10^{-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 α 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_{tf} corresponds to the test forgetting data, and D_{tr} 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 par 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:

$$\mathrm{US}(\mathrm{acc}_r, \mathrm{acc}_f) = \frac{\exp(\frac{\mathrm{acc}_r}{100}) + \exp(1 - \frac{\mathrm{acc}_f}{100}) - 2}{2 \cdot (\exp(1) - 1)}, \quad (8)$$

where acc_r and 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-tune** 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.

		Retrain	FF	FT	NG	RL	BS	IWU	**Ours**
Data Usage		45k	45k	45k	5k	5k	5k	5k	5k
Time	**VGG16**	3,683	9,710	433	73	24	116	1,351	**3.76**
	ResNet18	2,871	12,526	546	153	30	191	362	**4.37**
	ResNet50	4,705	19,850	1,061	174	57	471	1,513	**7.76**
	ViT	4,441	13,238	479	78	23	163	1,563	**25.93**

Table 2: Total extra data used and time consumption in sec for training different unlearning methods on CIFAR-10.

	Original PGD			Partial-PGD		
	D_{tr}	D_{tf}	Time (s)	D_{tr}	D_{tf}	Time (s)
VGG16	92.03	0	14.18	92.18	0	3.76
ResNet18	92.97	0	18.19	93.53	0	4.37
ResNet50	91.84	0	44.15	93.52	0	7.76
ViT	78.07	0	237.36	81.14	0	25.93

Table 3: Original PGD vs. Partial-PGD.

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

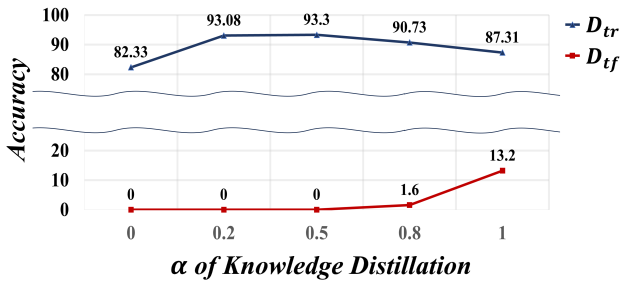
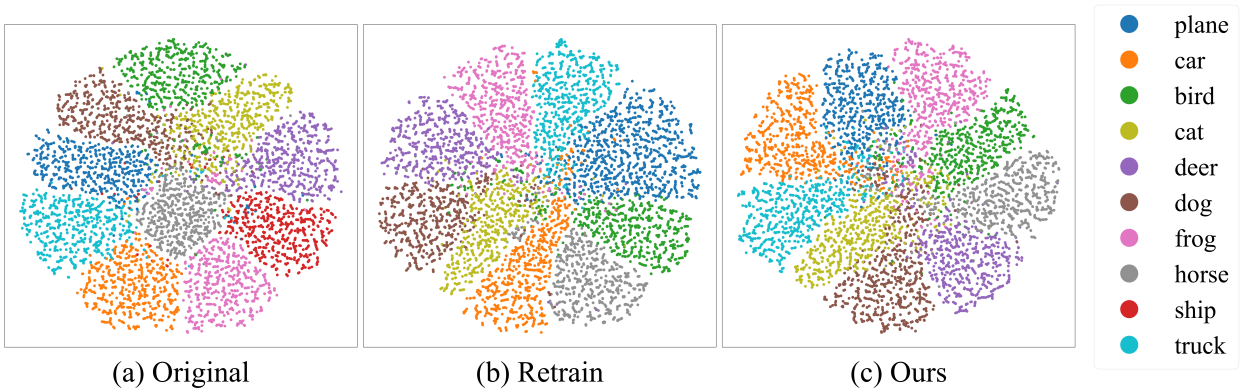


Figure 4: Impact of hyper-parameter α in Knowledge Distillation vs. Accuracy on CIFAR-10 with ResNet18.



(a) Original (b) Retrain (c) Ours

Figure 5: Visualization of decision boundary for the CIFAR-10 dataset in ResNet18, where each point represents a sample colored with the predicted classes. Red dots in (a) are the data to be removed, which are not showing in (b) and (c), indicating the successful unlearning. Similar plots for other models are provided in Suppl. Mat.*

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 α 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 α in Eq. 6. As the α 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 α approaches one, heavily relying on the teacher model for retaining information increases D_{tf} accuracy, indicating ineffective unlearning. Therefore, selecting the appropriate α value can maximize unlearning performance. Consequently, we used α ranging from 0.4 to 0.6 in this work. In more detail, the effects of changes in α are described in Suppl. Mat.*

	w/o Double Softmax			w/ Double Softmax		
	D_{tr}	D_{tf}	Time (s)	D_{tr}	D_{tf}	Time (s)
VGG16	84.74	0	10.9	93.89	0	8.75
ResNet18	91.42	0.1	25.87	94.54	0	5.19
ResNet50	80.91	0	93.49	94.48	0	9.13
ViT	87.01	0	61.37	87.44	0	13.39

Table 4: Effect of Softmax vs. Double Softmax.

Model	VGG16		ResNet18		ResNet50		ViT	
Data Usage	2,500	500	2,500	500	2,500	500	2,500	500
D_{tr}	92.42	92.38	93.51	93.38	93.63	93.37	81.14	81.6
D_{tf}	0	0	0	0	0	0	0	0.1
US	0.94	0.94	0.95	0.95	0.95	0.95	0.86	0.87
Time	1.91	1.21	2.28	1.45	3.81	1.62	25.63	14.55

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. 2021-0-02068, Artificial Intelligence Innovation Hub), (No. 2019-0-00421, AI Graduate School Support Program at Sungkyunkwan University), and (No. RS-2023-00230337, Advanced and Proactive AI Platform Research and Development Against Malicious Deepfakes).

References

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

Burgess, M. 2023. ChatGPT Has a Big Privacy Problem. https://www.wired.com/story/italy-ban-chatgpt-privacy-gdpr/. Accessed: 2023-04-04.

Cao, Q.; Shen, L.; Xie, W.; Parkhi, O. M.; and Zisserman, A. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, 67–74. IEEE.

Cao, Y.; and Yang, J. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, 463–480. IEEE.

Cha, S.; Cho, S.; Hwang, D.; Lee, H.; Moon, T.; and Lee, M. 2023. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. *arXiv preprint arXiv:2301.11578*.

Chen, M.; Gao, W.; Liu, G.; Peng, K.; and Wang, C. 2023. Boundary Unlearning. *arXiv preprint arXiv:2303.11570*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; and Unterthiner, T. 2020. Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

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

Hinton, G. 2022. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Kaye, K. 2022. The FTC's 'profoundly vague' plan to force companies to destroy algorithms could get very messy. https://www.protocol.com/enterprise/ftc-algorithm-data-model-ai/. Accessed:2022-05-17.

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

Kublik, V. 2019. EU/US Copyright Law and Implications on ML Training Data. https://valohai.com/blog/copyright-laws-and-machine-learning/. Accessed:2019-03-01.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mantelero, A. 2013. The EU Proposal for a General Data Protection Regulation and the roots of the 'right to be forgotten'. *Computer Law & Security Review*, 29(3): 229–235.

Nguyen, T. T.; Huynh, T. T.; Nguyen, P. L.; Liew, A. W.-C.; Yin, H.; and Nguyen, Q. V. H. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Shibata, T.; Irie, G.; Ikami, D.; and Mitsuzumi, Y. 2021. Learning with Selective Forgetting. In *IJCAI*, volume 3, 4.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.