Countering Relearning with Perception Revising Unlearning

Chenhao Zhang CHENHAO.ZHANG@UQ.EDU.AU
University of Queensland

Weitong Chen T.CHEN@ADELAIDE.EDU.AU
Wei Zhang WEI.E.ZHANG@ADELAIDE.EDU.AU
University of Adelaide

Miao Xu* MIAO.XU@UQ.EDU.AU
University of Queensland

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

Unlearning methods that rely solely on forgetting data typically modify the network's decision boundary to achieve unlearning. However, these approaches are susceptible to the "relearning" problem, whereby the network may recall the forgotten class upon subsequent updates with the remaining class data. Our experimental analysis reveals that, although these modifications alter the decision boundary, the network's fundamental perception of the samples remains mostly unchanged. In response to the relearning problem, we introduce the Perception Revising Unlearning (PRU) framework. PRU employs a probability redistribution method, which assigns new labels and more precise supervision information to each forgetting class instance. The PRU actively shifts the network's perception of forgetting class samples toward other remaining classes. The experimental results demonstrate that PRU not only maintains unlearned models' classification effectiveness but also significantly reduces the risk of relearning, suggesting a robust approach to class unlearning tasks that depend solely on forgetting data. The experimental code is available[1].

Keywords: Weakly-supervised learning; Machine Unlearning

1. Introduction

Machine unlearning (Brophy and Lowd, 2021; Nguyen et al., 2020; Sekhari et al., 2021), which allows a model to forget certain data, has grown in importance due to increasing data privacy concerns and legal protections (BUKATY, 2019; European Parliament and Council of the European Union). This concept is crucial in the era of widespread machine learning services (Ribeiro et al., 2015), which offer convenience but also necessitate adherence to privacy agreements and the protection of users' "right to be forgotten". Classification tasks, such as image classification (Lu and Weng, 2007), spam email detection (Guzella and Caminhas, 2009), and customer behavior prediction (Li et al., 2019), are all important machine learning services where class unlearning has also received much attention. Class unlearning generally requires a well-trained model to forget or disregard certain classes that it has previously learned. The data of the classes to be forgotten is termed "forgetting data", while the data of the classes to be retained is called "remaining data".

* Corresponding author

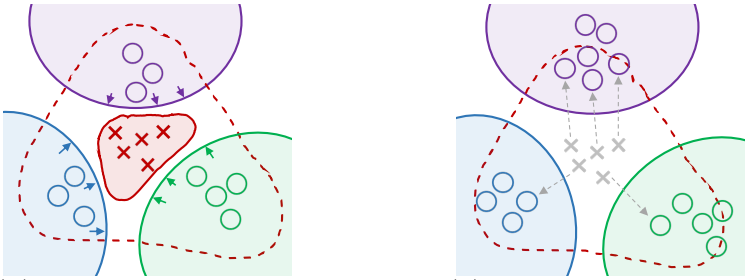
1. https://github.com/DATA-Transpose/PRU

Existing class unlearning methods fall into two main categories: *retrain based* and *non-retrain based*. Retrain based methods (Bourtoule et al., 2021; Chen et al., 2022; Liu et al., 2021) are time-consuming and impractical in many applications, leading to greater interest in non-retrain based methods (Graves et al., 2021; Tarun et al., 2021; Thudi et al., 2022) that don't require retraining a model from scratch. However, these non-retrain based methods typically need both forgetting and remaining class data for unlearning, a requirement not always feasible in real-world applications where historical data may be deleted after training due to storage constraints or privacy concerns.

Newly uploaded data may be able to help in such a situation of lacking historical data, however, not every class has newly uploaded samples when the unlearning request is initiated. For example, in face recognition applications (Masi et al., 2018), when a user's latest facial features are acquired and an unlearning request is made, other users' data may not be freshly uploaded, leaving only the forgetting data for the unlearning process. Existing non-retrain unlearning methods (Graves et al., 2021; Tarun et al., 2021; Thudi et al., 2022; Shen et al., 2024a), which focus on the parameter changes brought about by gradients, may fail in such scenarios as they require a fine-tuning on remaining class data to maintain performance on those classes. Boundary Unlearning (Chen et al., 2023) can unlearn using only forgetting class data. It shifts focus from parameter space to output space by identifying the nearest remaining class for each forgetting data and extending its decision boundary to encompass the forgetting data. Thus, the model classifies forgetting data as some remaining class. Boundary Unlearning not only addresses the challenge of unlearning with only forgetting data but also offers a more straightforward and comprehensible approach to understanding unlearning.

The shifted boundary is vulnerable to being reversed, especially if the boundary is just shifted by tuning the model with newly labeled forgetting data. This is because, if the decision boundary of the remaining classes can be shifted and extended into that of the forgetting class in such a way, the extended boundary can also be refined back by tuning it on data from the remaining classes. This enables the model to readily recall the forgetting class and again to correctly determine the samples that match the characteristics of the forgetting class. This scenario, in which the model continues to train on the remaining class data after unlearning, is a common occurrence. For example, in real-world applications, the model needs to be updated with new data to adapt to possible changes in the data distribution. Obviously, after unlearning, it is still possible to train the model on new data from the remaining classes. We present the **relearning** problem, at the first time, of recalling the forgetting class when the model continues to learn from only the remaining class data. This **relearning** problem differs from the term used in Tarun et al. (2021), where the unlearned model relearns from a dataset containing both forgetting and remaining data. Therefore, how to avoid the **relearning** has emerged as a novel challenge for unlearning that relies solely on forgetting class data.

According to the common neural network structure used in classification tasks (He et al., 2016; Springenberg et al., 2015), the output layer is basically a linear layer, which is used to map the sample features extracted by the upper layers into the output space, as shown in Figure 2(*a*). For simplicity, we name the last linear layer *classifier*, and all the layers above form the *feature extractor*. We found that simply using the relabeled forgetting data to guide the shift of the decision boundary made the changes in the neural network very subtle, with



(a) Boundary shifting. (b) Perception revising.

Figure 1: Schematic representation of the network perceptual space. All data points (circles and crosses) are forgetting class samples. Dashed and solid lines indicate decision boundaries before and after unlearning, respectively. Red lines mark the forgetting class's boundary, while other colors represent other classes. Circles are successfully unlearned samples, and crosses are samples that still be classified as the forgetting class after unlearning.

most of the changes occurring in the classifier and less in the feature extractor. This means that the network's perception of the sample has not changed much. For instance, a cat observed by an unlearned model is not significantly different from the cat observed prior to unlearning, except that the model says "that's an aeroplane". Therefore, we believe that it is the subtle changes in the parameters, most of which occur in the classifier, that make the model susceptible to relearning.

In this work, we propose a new unlearning framework, named Perception Revising Unlearning (PRU), to address the relearning challenge. The PRU can use only forgetting data in unlearning and can overcome the relearning problem by explicitly moving the network's perception of forgetting data towards the remaining classes, as shown in Figure 1(b). Unlike searching for the nearest remaining class for forgetting data, as in Boundary Shrink (Chen et al., 2023), we redistribute the decision probability on the forgetting class of a sample proportionally to other classes, which is faster. We use the redistributed probabilities to find new labels for each forgetting data and use the new probabilities as important supervisory information in unlearning. These redistributed forgetting samples can be divided into clusters according to their new labels. Then, in the network's perception space (feature extractor's output space), we explicitly increase the gap among these clusters and decrease the gap among the samples within a cluster. To allow more changes in the feature extractor, we update the feature extractor and classifier alternately during unlearning, i.e., fixing the parameters of one while updating the other.

Our contribution can be summarized as follows: 1) To solve the relearning problem encountered by existing methods when only forgetting data is available, we propose the Perception Revising Unlearning (PRU) framework with a simple strategy to revise the network's perception rather than shift the network's decision boundary. 2) We novelly propose a simple strategy that can explicitly move the network's perception of forgetting data toward the remaining classes. We also novelly split a neural network into the feature extractor and the classifier, and propose a new flexible unlearning framework that alternately updates these two parts of a network in unlearning for more significant changes in the feature extractor. 3) Experimental results illustrate that existing methods, which can only

use forgetting data for unlearning, run into the relearning problem. In contrast, PRU can overcome the relearning problem and shows superior performance over these baselines.

2. Related Work

2.1. Machine unlearning

Machine unlearning for traditional machine learning methods (Brophy and Lowd, 2021; Li et al., 2021) have found analytical optimization solutions by identifying the impact of data on model. However, for deep learning techniques (Cheng et al., 2016; Masi et al., 2018; Mehrish et al., 2023; Shen et al., 2024b), the non-convex nature of the optimization problem and the stochasticity of the learning process make it hard to identify the impact of data on the trained model and further eliminate such impact from the model. Retraining a new model from scratch with remaining data is straightforward but time-consuming. To accelerate the retraining, retain-based methods (Bourtoule et al., 2021; Chen et al., 2022; Liu et al., 2021) split the complete training dataset into partitions and train models one for each partition, when unlearning, only the partitions that contained forgetting data need to be retrained. However, these methods require as much data as possible in retraining to restore the model's performance on the remaining data, which makes them infeasible when training data becomes less or unavailable. Non-retrain based methods mainly attempt to identify the impact of data on the model and then eliminate these impacts. For example, Golatkar et al. (2020) uses Fisher information (Martens, 2020) to identify data impact on the model, in which the calculation of the Hessian matrix is complicated. The Unrolling SGD (Thudi et al., 2022) and Amnesiac unlearning (Graves et al., 2021) record the changes in network's parameters during the training and recover these changes during unlearning. The UNSIR (Tarun et al., 2021) uses the Impair-Repair two-step method, first using the negative gradient of forgetting data to impair the network and then repairing the network through remaining data. Although these non-retrain methods avoid retraining the model and only require part of the training data, they still need the remaining class data to maintain the performance of the model. In contrast, our proposed method will only require the forgetting class data for unlearning.

2.2. Unlearning with only forgetting data

As aforementioned, in practical applications, the data of the remaining class is usually unavailable and existing non-retrain based unlearning may fail in such a situation. For example, Unrolling SGD (Thudi et al., 2022) first performs incremental learning on the forgetting class data, records the direction of the gradient, and then impairs the network by reloading a larger gradient in the reversed direction. If there is no remaining class data to repair the network, the network may perform poorly on remaining classes.

Boundary Unlearning (Chen et al., 2023) focuses on the model's output space and proposes to achieve unlearning by shifting the model's decision boundary with two specific methods: 1) BoundaryShrink updates the input samples by fixing the network parameters and searches the nearest incorrect class on the gradient direction for each forgetting data as their new label. The original samples and their new labels are then used to update the model; 2) BoundaryExpanding updates an expanded model by creating a dummy class in

the output space, which dummy class will be used as the new label for forgetting data and be directly pruned when the updating is completed. Although these methods only require forgetting class data for unlearning, they result in the majority of the changes in the network occurring in the output layer, with minimal alterations in the upper layers. That is, only the decision boundary of the model changes, but the model's perception of the sample does not change much. This will make the unlearned model encounter the relearning problem. In contrast, in our proposed method, the relearning problem will be solved by actively moving the model's perception of the forgetting class data toward that of the remaining classes.

3. Preliminaries

3.1. Class unlearning

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \in \mathcal{X} \times \mathcal{Y}$ be a dataset containing n data samples that belong to K classes. The i-th pair of the data sample and its associated label can be denoted as (x_i, y_i), where $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y_i \in \mathcal{Y} = \{1, \ldots, K\}$. We denote $\mathcal{D}^k = \{(x_i, y_i) | y_i \in \mathcal{Y}^k\}$ as a subset of $\mathcal{D}$ that contains samples of the k-th class, where $\mathcal{Y}^k$ is a set that only contains class label k. When a class unlearning request is issued, it requires the model to forget knowledge on the forgetting class $\mathcal{Y}_f$ and maintain knowledge learned on the retain class $\mathcal{Y}_r$, where $\mathcal{Y}_f, \mathcal{Y}_r \subset \mathcal{Y}, \mathcal{Y}_f \cap \mathcal{Y}_r = \varnothing$ and $\mathcal{Y}_r \cup \mathcal{Y}_f = \mathcal{Y}$. Then, we can further denote their corresponding dataset $\mathcal{D}_f$ and $\mathcal{D}_r$, where $\mathcal{D}_f \cup \mathcal{D}_r = \mathcal{D}$ and $\mathcal{D}_f \cap \mathcal{D}_r = \varnothing$. In the context of unlearning, an original model $g(\cdot, \theta_{or})$ is trained with $\mathcal{D}$. A retrained model $g(\cdot, \theta_{re})$ is trained with $\mathcal{D}_r$. An unlearning method $\mathcal{U}$ is expected to make $g(\cdot, \theta_{or})$ forget the knowledge about $\mathcal{D}_f$ by outputting an unlearned model $g(\cdot, \theta_{un})$, i.e., $g(\cdot, \theta_{un}) = \mathcal{U}(\mathcal{D}, g(\cdot, \theta_{or}))$, which unlearned model has the similar performance as a retrained model, i.e., $g(\cdot, \theta_{un}) \approx g(\cdot, \theta_{re})$.

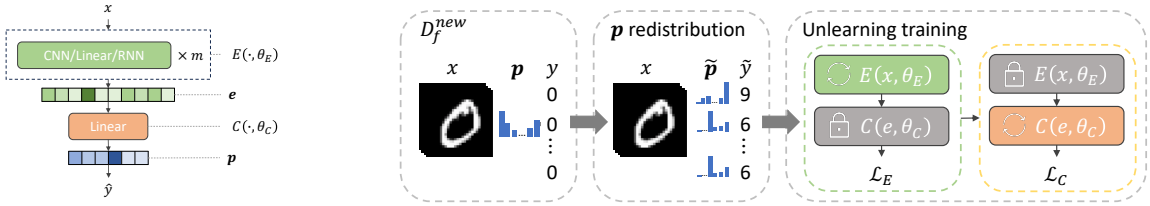
In this work, we focus on the setting, where there is only forgetting data available for unlearning. This is a more restricted condition than where both forgetting and remaining data are available. In such a case, the unlearning method $\mathcal{U}$ is required to be able to use only the forgetting class data for unlearning, i.e., $g(\cdot, \theta_{un}) = \mathcal{U}(\mathcal{D}_f, g(\cdot, \theta_{or}))$

3.2. Deep neural network

As shown in Figure 2(*a*), a deep learning neural network for classification tasks $g(x, \theta)$ is usually composed of multiple feature extraction layers followed by a linear layer. The feature extraction layers can extract the features of the input sample x and output a feature vector $\boldsymbol{e}$. The structure and number of the feature extraction layers vary according to specific tasks, and we define these feature extraction layers collectively as *feature extractors* $E(x, \theta_E)$. The last linear layer will map the extracted features $\boldsymbol{e}$ into the output space and get a probability vector $\boldsymbol{p} \in (0, 1)^K$, where $\boldsymbol{p}_k \in (0, 1)$ is the possibility of x belonging to the k-th class. Therefore, we define this linear layer *classifier* $C(\boldsymbol{e}, \theta_C)$, and the network's output of a given sample x can be presented as $g(x, \theta) = C(E(x, \theta_E), \theta_C)$.

4. Method

In this section, we first introduce the main processes of the proposed PRU framework (shown in Figure 2(*b*)). In PRU, we first use a probability redistribution strategy to find new labels



(a) Network $g(x, \theta)$ (b) PRU framework.

Figure 2: The overview of PRU. As shown in Figure 2(a), we divide the neural network architecture commonly used in classification tasks into feature extractor $E(\cdot, \theta_E)$ and classifier $C(\cdot, \theta_C)$. Forgetting samples will be re-labeled and be used for unlearning training, in which the $E(\cdot, \theta_E)$ and $C(\cdot, \theta_C)$ are alternately updated, as shown in Figure 2(b).

for each forgetting data and use the new decision probability as important supervisory information to update the network. We use an alternating training method to alternately update the feature extractor $E(\cdot, \theta_E)$ and classifier $C(\cdot, \theta_C)$. Specifically, in each training epoch, we first fix the θ_C and update the θ_E, then fix the θ_E and update the θ_C. When updating θ_E, we designed a new objective function to allow the model's perception of these samples, i.e., the $\boldsymbol{e}$ to actively move to their new classes. When updating θ_C, we use the redistributed probability to slightly refine the model's decision boundaries. We will detail these steps in the following subsections.

4.1. Probability redistribution

An unlearned model that is retrained from scratch will classify the forgetting class data as other remaining classes, and such new classes of each forgetting sample may be different, even if they originally belonged to the same class. Intuitively, this new class should be the one that is most difficult to distinguish from the original class of the forgetting sample before unlearning. We hope that the model can transfer the ability once used to distinguish forgetting classes to distinguish new classes after unlearning. For example, in the handwritten digit, "4" and "9" are difficult to distinguish because they both have a tail and a circle at the top-left (the "4" looks like it has a circle because it is handwritten), then if the model is asked to forget "9", it will say "4" without confusion when it sees such a pattern with a tail and a circle at the top-left. Therefore, in PRU, we innovatively redistribute the decision possibilities for not only finding the nearest incorrect class for each forgetting class sample but also help the model better distinguish the remaining classes after unlearning.

We first use the original model $g_{or}(\cdot, \theta)$ to obtain the possibility vector $\boldsymbol{p}^i \in (0, 1)^K$ of the i-th sample. When the model is asked to forget the f-th class, we will redistribute the $\boldsymbol{p}^i_f$ to other classes according to proportion. Specifically, an assignment proportion $\boldsymbol{a}^i \in (0, 1)^K$ can be prepared by

$$\boldsymbol{a}^i_k = \frac{\boldsymbol{p}^i_k}{\sum_k^K \boldsymbol{p}^i_k}, k \neq f \;\; and \;\; \boldsymbol{a}^i_f = 0, \tag{1}$$

then, the redistributed probability $\tilde{\boldsymbol{p}}^i$ can be calculated as

$$\tilde{\boldsymbol{p}}^i_k = \boldsymbol{p}^i_k + \boldsymbol{p}^i_f \cdot \boldsymbol{a}^i_k, k \neq f \;\; and \;\; \tilde{\boldsymbol{p}}^i_f = 0. \tag{2}$$

Therefore, the new label of a forgetting sample can be obtained by $\tilde{y} = \arg\max(\tilde{\boldsymbol{p}}^i)$. Although $\tilde{y}$ can be obtained directly by finding the second largest $\boldsymbol{p}_k^i$, the more important role of $\tilde{\boldsymbol{p}}^i$ is to provide supervision information in the unlearning training of PRU. We will discuss this more detailed in the following subsection.

4.2. Unlearning training

We believe that the reason why existing unlearning methods (Chen et al., 2023) encounter relearning problems when only using forgetting data is because the unlearned networks only reshape the decision boundary and the network's perception of the sample does not change much. Therefore, we innovatively divide the network into a feature extractor $E(\cdot, \theta_E)$ followed by a classifier $C(\cdot, \theta_C)$ and update them alternately with different objectives, in the hope that more changes in the feature extractor will lead to more changes in the network's perception of the samples. When updating one, the parameters of the other will be fixed.

Inspired by knowledge distillation (Hinton et al., 2015), the output of a trained network carries more information than finite discrete labels can and is usually more useful for training other networks. Therefore, it is worth noting that, unlike existing methods that use the new labels found for the forgetting class sample as supervision information, we use the redistributed probability $\tilde{\boldsymbol{p}}^i$ in our objective.

4.2.1. Updating feature extractor

Since the purpose of updating the feature extractor $E(\cdot, \theta_E)$ is to revise the network's perception of the samples, we introduce a new objective to make the network's perception of the forgetting class samples move towards their new label (some remaining class).

Forgetting class samples can get their new labels after probability redistribution (Section 4.1), that is, these samples that originally belonged to one class are divided into different clusters (new classes). Thus, the first goal is to separate these different clusters as much as possible. We calculate the centroid o_k of k-th class's cluster through the perception $\boldsymbol{e}$ output by $E(\cdot, \theta_E)$, i.e., $o_k = mean(\{e^i | \tilde{y}^i = k\}) = \frac{1}{N_k} \sum_{\tilde{y}=k} e^i$, where the o_k is the centroid of the k-th class's cluster and N_k represents the number of samples in this cluster. We then maximize the average Euclidean distance d_o between each pair of these centroids

$$\mathcal{L}_o = \frac{1}{d_o + \epsilon}, \tag{3}$$

where ϵ is a very small number for calculation stability. While increasing the distance between these clusters, we hope that the samples within each cluster can become closer. Therefore, the second goal is to minimize the average Euclidean distance d_s between each pair of samples within the same cluster, i.e.,

$$\mathcal{L}_s = mean(\{\| e^i - e^j \|_2 \, | i \neq j, \tilde{y}^i = \tilde{y}^j\}), \tag{4}$$

where the $\| \cdot \|_2$ is the L_2 norm.

We also need to take the goal of the classification task into account and help the model distinguish the remaining class by minimizing the difference between the distribution of $\hat{\boldsymbol{p}}^i$

output by the unlearn model and the distribution of redistributed $\tilde{\boldsymbol{p}}^i$, specifically, we use the Kullback-Leibler divergence to measure the difference

$$\mathcal{L}_p = \sum_k \tilde{\boldsymbol{p}}_k^i \log \frac{\tilde{\boldsymbol{p}}_k^i}{\hat{\boldsymbol{p}}_k^i} \tag{5}$$

Finally, we have the objective function $\mathcal{L}_E$ for updating the feature extractor $E(\cdot, \theta_E)$, that is, $\mathcal{L}_E = \mathcal{L}_p + \mathcal{L}_s + \mathcal{L}_o$.

4.2.2. Updating classifier

Since the classifier $C(\cdot, \theta_C)$ is usually a linear layer used as the output layer by a neural network, changes in the classifier will have more effect on the network's decision boundary. The purpose of updating $C(\cdot, \theta_C)$ is to match the revised perception by refining the decision boundary. Considering this purpose and the computational complexity, only $\mathcal{L}_p$, in Eq. 5, is used as the objective when updating $C(\cdot, \theta_C)$, i.e., $\mathcal{L}_C = \mathcal{L}_p$.

5. Experiment

We evaluate the effectiveness of the proposed PRU on two benchmark datasets, i.e., Digit-MNIST[2] and CIFAR-10[3], and two nerual network architectures, i.e., AllCNN (Springenberg et al., 2015) and ResNet18 (He et al., 2016). We use both architectures on the CIFAR-10 and use only the AllCNN on the Digit-MNIST. For ease of tabular presentation, we use an abbreviation of the form "**Data-Model**" to denote the dataset and model structure setup in tables. In this format, we use **DM** to denote the Digit-MNIST dataset and **C10** denotes CIFAR-10. For the model structure, we use **A** for AllCNN and **R** for ResNet18.

5.1. Settings

Datasets. For each dataset, we conducted experiments using all ten classes separately as forgetting classes. To simulate the model update in the real-world application, we split 20% data from the training set of the raw dataset to form the future set $\mathcal{D}^{new}$ and use only the $\mathcal{D}_r^{new}$ when updating. We conduct two groups of experiments: 1) to facilitate a more fair comparison between comparison methods, we use all the forgetting data in the original training data, i.e., the $\mathcal{D}_f$, when unlearning; 2) we simulate the unlearning with newly uploaded forgetting class data, the $\mathcal{D}_f^{new}$ when unlearning. Since the $\mathcal{D}^{new}$ is randomly sampled from the raw dataset, we can assume that $\mathcal{D}^{new}$ and $\mathcal{D}$ are identically distributed, except that the $|\mathcal{D}_f^{new}| = |\mathcal{D}_f|/4$. Unless otherwise specified, all experiments are conducted using $\mathcal{D}_f$. Experiments on more datasets are placed in Appendix C.

Baselines. We conduct comparison experiments on two types of unlearning methods, one requires both data of forgetting and remaining classes, like the Unrolling SGD (Thudi et al., 2022), and the other can use only forgetting class data that includes Boundary Shrink and Boundary Expanding (Chen et al., 2023). *1) Retrain:* The unlearned model is trained from scratch with remaining data, which model can be regarded as the optimal unlearned

2. https://yann.lecun.com/exdb/mnist/
3. https://www.cs.toronto.edu/~kriz/cifar.html

model. *2) Boundary Shrink (Chen et al., 2023):* It finds the nearest but incorrect label for a forgetting data by using its neighbor searching, then, tunes the model with newly labeled forgetting data. *3) Boundary Expanding (Chen et al., 2023):* Instead of finding a new label, Boundary Expanding creates a dummy class by adding an extra output neuron and assigns the dummy class to forgetting data. After tuning with such newly labeled forgetting data, the output neuron of the dummy class will be pruned. *4) Unrolling (Thudi et al., 2022):* When unlearning, Unrolling SGD performs incremental training and learns all forgetting data before remaining data. It records gradients when learning the forgetting data and adds recorded gradients on weights after the incremental training. *5) UnrollingOF:* UnrollingOF refers to a variant of Unrolling SGD using only forgetting class data. We implement it to show that this type of method requires retaining class data to ensure their effectiveness of unlearning.

Evaluation metrics *Classification Accuracies:* We test unlearned models on the forgetting test data and the retaining test data for obtaining A_f and A_r, respectively. Both A_f and A_r are the closer to that of the *Retrain* model the better. *Relearning Accuracies:* Unlearned models are continuously trained on the remaining only data for several epochs, and the A_f is observed during this relearning process. The faster the A_f increases, the less effect the unlearning. Unlearning time cost results are placed in Appendix D.

Implementation details. Following the baselines' setting, for the training of the original model, the training batch size for all datasets is set as 64 and the SGD optimizer with 0.9 momenta is used. We train the 1) AllCNN on the MINST using a learning rate of 0.01 for 10 epochs; 2) AllCNN on the CIFAR-10 using a learning rate of 0.1 for 30 epochs; 3) ResNet18 using a learning rate of 0.01 for 20 epochs. For the relearning test, we train an unlearned model on $\mathcal{D}_r^{new}$ with a learning rate of 0.001 for 10 epochs, and all other training settings remain the same as the original training. To make the comparison fair, we select parameters for all methods by conducting unlearning on the 0-th class and following the principle of touching the forgetting accuracy of less than 0.01 while maintaining the remaining accuracy as high as possible. Since PRU updates feature extractor E and classifier C alternately, we can set the learning rate of E and C more flexibly (the parameter setting has been included in the supplementary material). We set each class as the forgetting class for experiments. When a class is set as a forgetting class, all other classes in the dataset are used as remaining classes. The reported results are averages across experiments when each class is set as the forgetting class. All the experiments are conducted for five trials and the reported results are also the average across all five trials of experiments using different seeds.

5.2. Unlearning effectivness

We first evaluate the effectiveness of the unlearned methods and report the results in terms of classification accuracy. The A_f denotes the test accuracy on the forgetting class and the A_r denotes the test accuracy on the remaining classes, the closer the two are to the retrained model, the better the performance.

Let's first take a look at the results when using $\mathcal{D}_f$ for unlearning. As shown in Table 1, the classification utility of our proposed PRU on the MNIST dataset outperforms that of other methods that only use forgetting data for unlearning (marked in *italic*). On

Table 1: Classification accuracy of unlearned models. The methods marked in *italic* letter use only forgetting data.

Dataset	Model	Acc	Original	Retrain	Unroll		*UnrollOF*		*BoundShrink*		*BoundExpand*		*PRU (OURS)*	
					$\mathcal{D}_f$	$\mathcal{D}_f^{new}$	$\mathcal{D}_f$	$\mathcal{D}_f^{new}$	$\mathcal{D}_f$	$\mathcal{D}_f^{new}$	$\mathcal{D}_f$	$\mathcal{D}_f^{new}$	$\mathcal{D}_f$	$\mathcal{D}_f^{new}$
DMNIST	AllCNN	$A_r \uparrow$	99.40	99.28	93.78	93.79	24.08	88.92	79.54	89.30	82.53	79.22	84.98	86.45
		$A_f \downarrow$	99.38	0.00	0.000	0.000	5.170	0.090	13.18	12.98	12.23	11.89	0.000	0.000
CIFAR-10	AllCNN	$A_r \uparrow$	84.17	84.11	89.59	88.87	64.97	66.33	78.82	81.19	81.62	80.84	77.86	84.59
		$A_f \downarrow$	84.17	0.00	0.260	3.370	0.250	0.250	0.630	1.800	8.150	8.180	0.000	0.090
	ResNet	$A_r \uparrow$	85.55	86.65	85.32	80.30	39.15	24.95	81.03	81.94	82.74	81.34	80.35	84.39
		$A_f \downarrow$	85.55	0.00	0.000	0.000	0.000	0.000	1.270	15.75	8.390	17.41	2.490	12.10

Table 2: Relearning results when unlearning using $\mathcal{D}_f$. Unlearned models are updated with $\mathcal{D}_r^{new}$ in 10 epochs, and the results of the 1/5/10-th epochs are reported. The first column tells the dataset and model structure settings in abbreviations, please refer to the first paragraph in Section 5 for a detailed description.

Data-Model	Acc	Original	Retrain	Unroll	*UnrollOF*	*BoundShrink*	*BoundExpand*	*PRU (OURS)*
DM-A	$A_r \uparrow$	98.6/99.5/99.6	99.5/99.5/99.6	99.1/99.5/99.5	98.8/99.4/99.5	99.1/99.5/99.6	98.8/99.5/99.6	**99.1/99.4/99.5**
	$A_f \downarrow$	88.8/85.3/85.3	0.0/0.0/0.0	0.0/0.0/0.0	8.1/12.7/8.4	75.6/72.7/71.7	76.2/83.4/82.6	**9.0/4.8/2.7**
C10-A	$A_r \uparrow$	87.6/89.4/89.9	88.0/89.2/89.7	90.7/90.9/91.0	87.6/89.4/89.9	85.1/88.8/89.6	87.1/89.2/89.8	**82.4/88.4/89.4**
	$A_f \downarrow$	86.9/75.4/68.0	0.0/0.0/0.0	0.8/0.9/0.8	86.3/75.0/67.5	45.4/51.3/50.1	81.7/70.3/62.7	**1.5/8.9/11.3**
C10-R	$A_r \uparrow$	89.7/90.4/90.5	89.4/90.1/90.3	89.4/90.0/90.2	89.7/90.4/90.6	89.5/90.4/90.5	89.6/90.4/90.6	**89.2/90.4/90.5**
	$A_f \downarrow$	67.7/49.8/40.4	0.0/0.0/0.0	0.0/0.0/0.0	65.1/51.4/42.8	57.3/43.6/35.4	60.2/40.5/30.3	**27.0/10.2/5.2**

the CIFAR-10 dataset, PRU's classification utility exhibits comparable performance with Boundary Unlearning. Although Boundary Expanding has a slightly higher A_r than PRU, its A_f is significantly higher than PRU, indicating that Boundary Expanding is not as effective as PRU in terms of forgetting. Table 2 shows the performance of the unlearned model obtained by these methods in the relearning test. It can be observed that when updating unlearned models using the $\mathcal{D}_r^{new}$, the A_f of PRU does not exhibit the same degree of improvement as the other compared methods, suggesting that PRU can effectively address the relearning challenge. Unrolling's classification utility is the most effective and its A_f remains unchanged during relearning, this is because it can use $\mathcal{D}_r$ to repair the network more accurately. In contrast, UnrollOF, which is a variant of Unrolling that only uses $\mathcal{D}_f$, does not avoid the relearning problem while not retaining a good A_r in utility.

We also conducted experiments when unlearning using only $\mathcal{D}_f^{new}$. It is noteworthy that we kept the parameter settings for all methods as they were when we conducted the experiments using $\mathcal{D}_f$, and $|\mathcal{D}_f^{new}| = |\mathcal{D}_f|/4$. The results in Table 1 demonstrate that the A_r and A_f of the unlearned model obtained by each method increase when the amount of used forgetting data decreases. This indicates that the degree of unlearning of each method is weakening in this case. The results of the relearning tests presented in Table 3 also provide evidence of this, with accelerated increasing of A_f for all methods. Although this, the A_f rebound of PRU remains relatively slow in comparison to other methods, with the utility demonstrating superior performance. The UnrollOF's A_f shows no growth in this case. This is because, lacking access to retaining data, the UnrollOF relies only on forgetting data to impair the network, which excessively damages the model. This results in poor performance on the original task (as shown by the 24.95% accuracy in Table 1). Thus, during relearning, its A_f does improve as expected. As previously stated, the PRU allows for more flexible tuning of the learning parameters. Thus, we further tune the learning rate

Table 3: Relearning results when unlearning using $\mathcal{D}_f^{new}$. Unlearned models are updated with $\mathcal{D}_r^{new}$ in 10 epochs, and the results of the 1/5/10-th epochs are reported.

Data-Model	Acc	Original	Retrain	Unroll	*UnrollOF*	*BoundShrink*	*BoundExpand*	*PRU (OURS)*
DM-A	$A_r \uparrow$	98.6/99.5/99.6	99.5/99.5/99.6	99.3/99.4/99.5	99.5/99.6/99.6	99.5/99.6/99.6	99.5/99.6/99.6	**99.3/99.5/99.6**
	$A_f \downarrow$	88.8/85.3/85.3	0.0/0.0/0.0	0.0/0.0/0.0	99.4/98.5/98.1	99.0/98.1/97.4	98.6/97.6/97.1	**19.4/26.1/30.3**
C10-A	$A_r \uparrow$	87.6/89.4/89.9	88.0/89.2/89.7	90.0/90.3/90.6	88.0/89.8/90.1	86.2/89.4/89.9	87.4/89.3/89.9	**85.4/89.3/89.8**
	$A_f \downarrow$	86.9/75.4/68.0	0.0/0.0/0.0	9.0/10.0/10.3	43.8/44.2/43.3	57.5/60.8/57.1	80.1/69.9/61.4	**46.2/51.8/48.0**
C10-R	$A_r \uparrow$	89.7/90.4/90.5	89.4/90.1/90.3	88.1/89.2/89.6	88.4/90.0/90.2	90.0/90.6/90.7	89.9/90.5/90.7	**90.0/90.6/90.7**
	$A_f \downarrow$	67.7/49.8/40.4	0.0/0.0/0.0	0.0/0.0/0.0	0.0/0.0/0.0	67.6/49.3/40.0	61.6/42.3/31.7	**63.0/41.9/30.6**

of the PRU for the scenario where the amount of $\mathcal{D}_f^{new}$ is less than that of $\mathcal{D}_f$. This enabled the PRU to obtain utility and relearning test results comparable to those using $\mathcal{D}_f$ in the case of using $\mathcal{D}_f^{new}$. We also tried to tune parameters for other comparison methods, but their results remained unchanged. Due to space constraints, this part of the experiment has been included in the supplementary material.

5.3. Changes in model

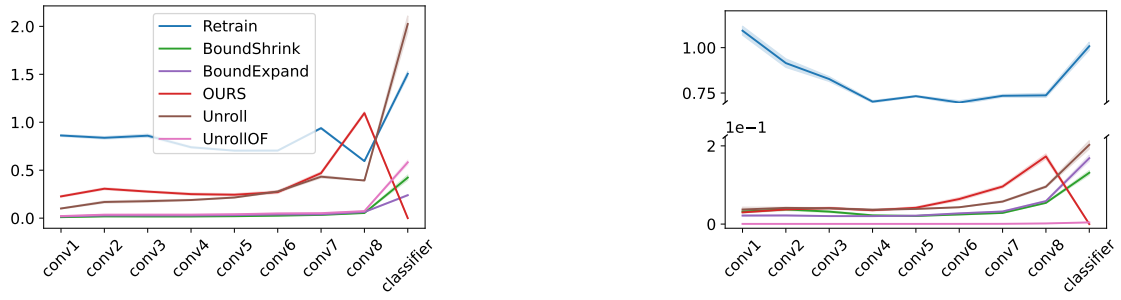
In this section, we investigate changes in these unlearned models, including changes in model parameters and differences in their output spaces.

5.3.1. Parameters changes

We mainly observe how much the parameters of the unlearned model have changed relative to the parameters of the original model. Specifically, we first calculate the difference between the parameters of the l-th layer of an unlearned model and that of the original model, i.e., $\delta^l = \theta_{un}^l - \theta_{or}^l$, and then normalize the modulus of this difference with the parameter modulus of the corresponding original model layer to obtain the relative parameter changes Δ, that is,

$$\Delta^l = \frac{\| \delta^l \|_2}{\| \theta_{or}^l \|_2}, \Delta^l \geq 0. \tag{6}$$

The closer the Δ^l is to 0, the less the unlearned model has changed from the original model. On the contrary, the larger the Δ^l, the more significant the change.



(*a*) DMNIST (*b*) CIFAR-10

Figure 3: The Δ (Eq.6) of each layer in the model. The left side shows the result on the DMNIST dataset, and the right side shows the result on the CIFAR-10 dataset.

We show the AllCNN parameter changes as examples. As shown in Figure 3, the retrained model has the most significant parameter change when compared to the original model. The unlearned model obtained by the proposed PRU and the one obtained by the unrolling method exhibited comparable parameter changes, particularly in the layers preceding the final layer, i.e., the feature extractor. The Unrolling requires both forgetting and remaining class samples for its unlearning, but PRU only requires forgetting class samples. The other three methods, which only use forgetting class samples, have subtle changes in their feature extractor. By combining the results in Table 2 and Figure 3, we can find that unlearned models with more alterations in the feature extractor are more robust to the challenges of relearning. It is noteworthy that PRU exhibits almost no change in the final classifier, yet it offers superior utility compared to other methods while effectively mitigating the issue of relearning.

5.3.2. Perception changes

It is difficult to visualize the data distribution and decision boundaries by directly reducing the dimension with PCA or T-SNE, since the dimensionality of image data is too high and many samples are difficult to distinguish by lines or surfaces in 2-D or 3-D space. Therefore, we regard the representation of a sample in the output space (perception space) of the model's feature extractor as the model's perception of the sample, and observe 1) the distribution of forgetting class samples in this space and 2) the distribution of their new labels after unlearning.

1) Distribution of forgetting samples. Let's first observe the relationship between forgetting class samples and forgetting class boundary in the perception space. Intuitively, if there is still a decision boundary for forgetting class in the perception space of an unlearned model, the closer a sample is to this boundary, the greater the probability that it will be classified as a forgetting class. For methods that mainly change the final classifier layer, the boundaries of other classes are more likely to expand inward from the edge of this cluster. Referring to the Figure 1(a), we use the output of the model's feature extractor, i.e., $e^i \in \mathbb{R}^d$, to represent data points (circles and crosses) in this figure. The closer to the centroid of the red solid line area, the greater the probability of being classified as a forgetting class, i.e. the greater the $\boldsymbol{p}_f^i$. Therefore, we are going to observe the relationship between the distance from the forgetting class samples in the test set to their centroid and the probability of them being classified as a forgetting class.

In detail, we have the centroid $o_f = mean(\{e^i | y^i = f\})$ and the Euclidean distance from a sample to the o_f, i.e., $Dist^i = \parallel e^i - o_f \parallel, y^i = f$. Since different networks have different perceptual spaces, distances in different spaces cannot be compared directly, so we will normalize these distances by the mean distance in their own space, and the normalized distance is

$$nDist^i = \frac{Dist^i}{mean(\{Dist^j | y^j = f\})}. \tag{7}$$

We take the unlearning of the number "9" in MNIST dataset as an example, as shown in Figure 4, when unlearning via Boundary Expanding, there are still 17% of "9" being classified as "9" after unlearning. Although no forgetting data is being correctly classified again in Boundary Shrink, there are still some samples that have a few probabilities on the $\boldsymbol{p}_f^i$. This indicates that in these methods, the decision boundary of the forgetting class

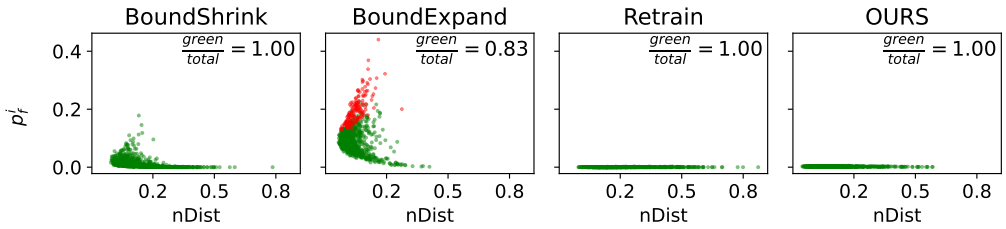


Figure 4: The probability of forgetting samples being determined as a forgetting class after unlearning. The x-axis is the normalized distance from forgetting samples to their centroid. Samples in red are still being determined as forgetting class. Samples in green are determined as some other classes.

still remains. The results of Boundary Unlearning demonstrate that the centroid of the test forgetting data is not the closest to the center of the forgetting class boundary. This may be because the generalization ability of the original model makes the original boundary of the forgetting class broader, and the distribution of the samples involved in modifying the classification boundary has limited coverage, resulting in a limited modification of the classification boundary. Our method behaves similarly to the retrained model in that none of the test samples are classified or have the tendency to be classified as forgetting classes.

2) New labels of forgetting samples. We still want to ask, where do all these forgetting class samples go after unlearning? We observe the classification results of all the test forgetting samples predicted by the unlearned model and count how many percent of them were classified in each class.

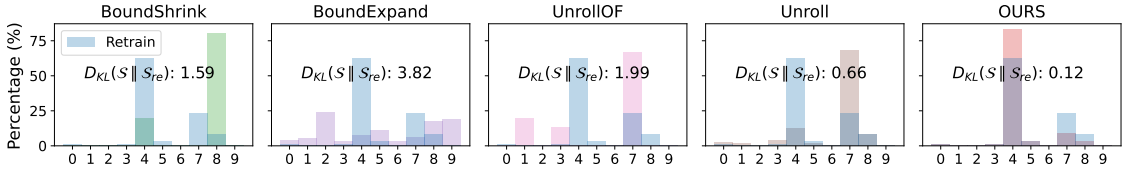


Figure 5: Prediction distribution on test forgetting samples. The x-axis is classes.

Detailedly, we compare the distribution $\mathcal{S}$, which is an unlearned model's prediction distribution on test forgetting samples, with the $\mathcal{S}_{re}$ obtained by the retrained model, and evaluate the gap between the $\mathcal{S}$ and $\mathcal{S}_{re}$ using the Kullback-Leibler divergence (Kullback and Leibler, 1951), i.e., $D_{KL}(\mathcal{S} \parallel \mathcal{S}_{re})$. We take the unlearning of the number "9" in MNIST dataset as an example. As shown in Figure 5, when unlearning class "9" in MNIST, the retrained model classifies forgetting samples into class "4" at most, with class "7" being the second most common. Among all the compared methods, the unlearning result of our method is the closest to that of the retrained model as our method has the smallest $D_{KL}(\mathcal{S} \parallel \mathcal{S}_{re})$. This is because the PRU benefit from using the redistributed probability as supervisory information and relabeling forgetting data, which allows for a more appropriate destination for the forgetting data and further reduces the impact on the remaining classes.

5.4. Ablation study

To explore the contribution of each objective in PRU to the unlearned model, we conducted an ablation study, whereby $\mathcal{L}_o$ and $\mathcal{L}_s$ are removed, and $\mathcal{L}_p$ is replaced with CELoss. The

Table 4: Ablation result on PRU's losses. Results in the table show the 0/1/5/10-th relearning epoch result.

Data-Model	Acc	PRU	PRU w/o $\mathcal{L}_p$	replace $\mathcal{L}_p$ with CELoss	w/o $\mathcal{L}_o$	w/o $\mathcal{L}_s$
DM-A	$A_r \uparrow$	85.0/99.1/99.4/99.5	87.3/99.1/99.4/99.5	62.3/96.0/98.1/98.5	74.8/98.8/99.3/99.4	90.9/99.1/99.4/99.5
	$A_f \downarrow$	0.0/9.0/4.8/2.7	0.0/0.0/3.7/4.7	0.0/0.0/0.0/0.0	0.0/0.0/0.8/2.7	0.0/0.0/0.3/0.5
C10-A	$A_r \uparrow$	77.9/82.4/88.4/89.4	68.5/83.4/88.5/89.5	75.7/82.0/88.4/89.4	54.3/79.1/88.2/89.4	84.4/84.8/88.9/89.7
	$A_f \downarrow$	0.0/1.5/8.9/11.3	20.8/72.7/55.8/47.0	0.0/2.1/10.6/12.5	0.0/0.2/4.0/7.3	0.0/23.4/35.8/35.4
C10-R	$A_r \uparrow$	80.3/89.2/90.4/90.5	80.6/84.8/88.5/89.4	79.2/89.2/90.4/90.5	79.9/89.1/90.3/90.6	85.2/89.5/90.4/90.6
	$A_f \downarrow$	2.5/27.0/10.2/5.2	11.1/75.2/60.1/48.5	2.1/28.0/10.2/5.2	2.5/26.2/9.3/5.1	3.2/52.3/28.2/19.3

results of the utility and relearning tests of the unlearned model are then observed. As shown in Table 4, the unlearned model is more susceptible to relearning when $\mathcal{L}_s$ is not used. A comparison of the unlearned model (0-th relearning epoch) reveals that $\mathcal{L}_p$ and $\mathcal{L}_o$ play a pivotal role in maintaining A_r. This is because the new labels assigned for the samples through our designed probability redistribution are more aligned with the current perception of the network, thereby better maintaining the network's performance on remaining classes.

5.5. Case study on ViT architecture

Table 5: ViT results. Results in the table show the 0/1/2/3-th relearning epoch result.

Acc	Retrain	Unroll	*UnrollOF*	*BoundShrink*	*BoundExpand*	*PRU (OURS)*
$A_r \uparrow$	94.73	88.29/96.9/97.0/97.1	77.68/96.5/96.8/96.9	87.85/96.6/96.9/97.0	79.09/96.6/96.8/97.0	**91.63/96.6/96.8/96.9**
$A_f \downarrow$	0.0	1.74/7.8/9.9/11.5	0.35/6.4/9.0/11.1	0.72/10.2/14.0/16.6	3.78/19.7/22.3/24.1	**0.27/6.9/10.8/13.2**

We try to apply PRU in a larger network architecture. We selected the contemporary Vision Transformer (ViT) (Dosovitskiy et al., 2021) for this case study. Settings are in the supplementary material. The results in Table 5 demonstrate that PRU can still outperform the comparison methods in both the utility after unlearning and the relearning test.

6. Conclusion

We focus on class unlearning tasks that use only forgetting class data. Existing methods that only use forgetting data primarily achieve unlearning by modifying the decision boundary of the network, and are susceptible to the *relearning* problem, in which the network may recall the forgetting class when it is further updated with the remaining class data. We analyzed this experimentally and found that although the decision boundary of the network changed the network's perception of the samples remained largely unaltered. In response to the relearning problem, we designed the Perception Revising Unlearning (PRU) framework. In the PRU, we design a probability redistribution method that can assign a new label and more appropriate supervision information to each instance of forgetting data. During the unlearning, the network's perception of forgetting class samples can actively move to other remaining classes. Experimental results show that PRU not only has superior unlearning effectiveness but also effectively mitigates the effect of relearning problems.

Acknowledgments

The work is supported by the Australian Research Council DE230101116 and DP240103070.

References

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *IEEE SP*, 2021.

Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *ICML*, 2021.

PRESTON BUKATY. *The California Consumer Privacy Act (CCPA): An implementation guide*. IT Governance Publishing, 2019.

Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *WWW*, 2022.

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *CVPR*, 2023.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *DLRS*, 2016.

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 *ICLR*, 2021.

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

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*, 2020.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *AAAI*, 2021.

Thiago S. Guzella and Walmir M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Syst. Appl.*, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, 2015.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

Jing Li, Shuxiao Pan, Lei Huang, et al. A machine learning based method for customer behavior prediction. *Tehnički vjesnik*, 2019.

Yuantong Li, Chi-Hua Wang, and Guang Cheng. Online forgetting process for linear regression models. In *AISTATS*, 2021.

Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *IWQOS*, 2021.

Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *Int. J. Remote Sens.*, 2007.

James Martens. New insights and perspectives on the natural gradient method. *J. Mach. Learn. Res.*, 2020.

Iacopo Masi, Yue Wu, Tal Hassner, and Prem Natarajan. Deep face recognition: A survey. In *SIBGRAPI*, 2018.

Ambuj Mehrish, Navonil Majumder, Rishabh Bharadwaj, Rada Mihalcea, and Soujanya Poria. A review of deep learning techniques for speech processing. *Inf. Fusion*, 2023.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. In *NeurIPS*, 2020.

Mauro Ribeiro, Katarina Grolinger, and Miriam A. M. Capretz. Mlaas: Machine learning as a service. In *ICMLA*, 2015.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *NeurIPS*, 2021.

Shaofei Shen, Chenhao Zhang, Alina Bialkowski, Weitong Chen, and Miao Xu. Camu: Disentangling causal effects in deep model unlearning. In *SDM*, 2024a.

Shaofei Shen, Chenhao Zhang, Yawen Zhao, Alina Bialkowski, Tony Weitong Chen, and Miao Xu. Label-agnostic forgetting: A supervision-free unlearning in deep models. In *ICLR*, 2024b.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan S. Kankanhalli. Fast yet effective machine unlearning. *CoRR*, 2021.

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: understanding factors influencing machine unlearning. In *IEEE SP*, 2022.