ERM-KTP: Knowledge-level Machine Unlearning via Knowledge Transfer

Shen Lin[1*], Xiaoyu Zhang[1*], Chenyang Chen[1], Xiaofeng Chen[1†], Willy Susilo[2]

[1]Xidian University, [2]University of Wollongong
linshen@stu.xidian.edu.cn, xiaoyuzhang@xidian.edu.cn, cychen_1@stu.xidian.edu.cn,
xfchen@xidian.edu.cn, wsusilo@uow.edu.au

Abstract

Machine unlearning can fortify the privacy and security of machine learning applications. Unfortunately, the exact unlearning approaches are inefficient, and the approximate unlearning approaches are unsuitable for complicated CNNs. Moreover, the approximate approaches have serious security flaws because even unlearning completely different data points can produce the same contribution estimation as unlearning the target data points. To address the above problems, we try to define machine unlearning from the knowledge perspective, and we propose a knowledge-level machine unlearning method, namely ERM-KTP. Specifically, we propose an entanglement-reduced mask (ERM) structure to reduce the knowledge entanglement among classes during the training phase. When receiving the unlearning requests, we transfer the knowledge of the non-target data points from the original model to the unlearned model and meanwhile prohibit the knowledge of the target data points via our proposed knowledge transfer and prohibition (KTP) method. Finally, we will get the unlearned model as the result and delete the original model to accomplish the unlearning process. Especially, our proposed ERM-KTP is an interpretable unlearning method because the ERM structure and the crafted masks in KTP can explicitly explain the operation and the effect of unlearning data points. Extensive experiments demonstrate the effectiveness, efficiency, high fidelity, and scalability of the ERM-KTP unlearning method. Code is available at https://github.com/RUIYUN-ML/ERM-KTP

1. Introduction

In recent years, many countries have raised concerns about protecting personal privacy. The privacy legislation, *e.g.*, the well-known European Union's GDPR [19], have been promulgated to oblige information service providers to remove personal data when receiving a request from the data owner, *i.e.*, the *right-to-be-forgotten*. Besides, the GDPR stipulates that the service providers should remove the corresponding impact of the data requested by the data owner, in which the machine learning models are the most representative. Many kinds of research demonstrate that the machine learning models can memorize knowledge of the data points, *e.g.*, the membership inference attack [7, 14, 15] can infer whether a data point is in the training set or not.

A naive approach is retraining the model after removing the target data points from the training set, but the human resources and materials consumed are costly. Thus, aiming to efficiently remove data as well as their generated contribution to the model, a new ML privacy protection research direction emerged, called *machine unlearning*. A good deal of related work attempted to solve the data-removing challenge of inefficient retraining, and there are two representative research fields, including *exact unlearning* [1] and *approximate unlearning* [3–5]. Unfortunately, these approaches usually require huge computational, storage overhead, and memory requirements for class-specific machine unlearning tasks on the complicated convolutional neural networks (CNNs). Furthermore, they may compromise the model's performance and even cause disastrous forgetting. Most significantly, Thudi et al. [18] believed that such approximate unlearning approaches have serious security flaws because they usually define *machine unlearning* as the distribution difference between the unlearned model and the retrained model. According to this definition, even unlearning completely different data points can produce the same contribution estimation as unlearning the target data points.

Unlearning algorithms are difficult to implement on deep learning models because these models are often seen as a black box and lack interpretability, which makes data points' contributions challenging to estimate. Though many related works [2, 10, 16, 21] attempted to improve the interpretability of deep learning models, they can not apply to machine unlearning directly. For example, Zhou et al. [22]

*Both authors contributed equally
†Corresponding author

leveraged the global average pooling (GAP) in CNNs to generate a class activation mapping (CAM) to indicate the discriminative image regions used by the CNNs to identify the class. Liang et al. [12] proposed class-specific filters to transform the complex representations in convolutional layers into interpretable graphs.

To address the above problems, for the first time, we consider the need for subsequent unlearning tasks during the model training phase. We define machine unlearning in the knowledge perspective, *i.e.,* a reliable machine unlearning approach should satisfy that the knowledge of the unlearned model should be identical to the retrained model. Furthermore, we propose an interpretable unlearning method, *i.e.,* ERM-KTP. The ERM structure can interpret the relation between data points and channels in the activation maps. Such a relationship can describe which filters the data points contribute to the convolutional layer. Then, KTP utilizes the relation to transfer the knowledge from corresponding filters to the unlearned model. Furthermore, the crafted masks in KTP interpret the operation of the knowledge transfer. With these interpretable methods, we can precisely unlearn data points and enhance the reliability of our proposed unlearning approach. We summarize our contributions of this paper as follows:

- We give a novel definition of machine unlearning from the knowledge perspective, and we further propose an interpretable knowledge-level machine unlearning method, *i.e.,* ERM-KTP. Especially, the interpretability of ERM-KTP enhances the reliability of the unlearning operation.
- We introduce an ERM structure that reduces the knowledge entanglement among classes to get a pretrained model. When receiving the unlearning requests, we use the proposed KTP method to transfer the knowledge of the remaining set from the original model to the unlearned model and, meanwhile, prohibit the knowledge of the unlearning set. Finally, we will get the unlearned model as the result and delete the original model to complete the unlearning process.
- We conduct experiments on three different scales of image classification datasets and three complicated CNNs. Extensive experimental results demonstrate the effectiveness, efficiency, high fidelity, and scalability of ERM-KTP.

2. Problem Formulation

In this paper, we primarily consider a class-specific machine unlearning scenario. We assume a sample space $\mathcal{X} \subseteq \mathbb{R}^d$ and corresponding labels $\mathcal{Y} = \{1, 2, \cdots, C\}$, where d is the number of dimensions and C is the number of classes. The training set can be represented as $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_C\}$. To unlearn the target class's data points $\mathcal{D}_u \subseteq \mathcal{D}$ exactly, we need to apply a removal method, which should be equivalent to applying the training algorithm to the dataset without target data points.

Definition 1. We define a *learning algorithm*, $\mathcal{A} : \mathcal{D} \to \mathcal{W}$, as a function from a dataset to a model in hypothesis space $\mathcal{W}$. A *removal method*, $\mathcal{R} : \mathcal{A}(\mathcal{D}) \times \mathcal{D} \times \mathcal{D}_u \to \mathcal{W}$, is a function from an original model $\mathcal{A}(D)$, training dataset $\mathcal{D}$, an unlearning dataset $\mathcal{D}_u$ to remove from the training dataset $\mathcal{D}$ to an unlearned model in $\mathcal{W}$, and a remaining dataset $\mathcal{D}_r = \mathcal{D} \backslash \mathcal{D}_u$. We define equivalence as having identical knowledge for each model in $\mathcal{W}$:

$$\mathbb{K}(\mathcal{A}(\mathcal{D}_r)) = \mathbb{K}(\mathcal{R}(\mathcal{A}(\mathcal{D}), \mathcal{D}, \mathcal{D}_u)), \tag{1}$$

in which $\mathbb{K}(\cdot)$ is a knowledge measuring function. If an exact unlearning method satisfy the above definition, it can be defined as *a knowledge-level unlearning method*.

3. Related Work

Exact unlearning. The exact unlearning approaches can provide algorithmic unlearning proof. The most representative exact unlearning approach is retrain-from-scratch (RfS), *i.e.,* remove the target data points from the training set and retrain the model. Bourtoule et al. proposed SISA to improve efficiency by retraining the sub-model instead of retrain-from-scratch. These exact unlearning approaches are knowledge-level because they remove all target data points in the input phase so that the model can not learn any knowledge of them. However, their computational overhead is still significant because they need to relearn the remaining set's knowledge through retraining.

Approximate unlearning. The main idea of approximate unlearning approaches is to estimate the target data points' contribution to the model and update the parameters for unlearning. For example, Graves et al. [4] leveraged gradients to estimate the contributions. They stored gradients of batches consisting of target data points and then subtracted the target gradients to update the model's parameters. For class-specific unlearning tasks or large-scale unlearning requests, it needs to store gradients of almost every batch, and the model will degenerate to the initialization state. Guo et al. [5] utilized the Influence Theory [8], and Golatkar et al. [3] used the Fisher Information [13] to estimate the contributions. However, the above two methods need to calculate the hessian matrix, which is very expensive in deep learning models. Besides, they may compromise the model's performance due to imprecise estimates of contributions on CNNs and even cause disastrous forgetting. Most significantly, Thudi et al. [18] pointed out that different data points can also generate the same network, making approximate unlearning approaches unable to provide an algorithmic unlearning proof.

4. Proposed Method

In this section, we will explain the ERM-KTP unlearning method in detail. We will describe how the proposed ERM structure reduces the entanglement of knowledge in Section 4.1. Then, we will explain how the proposed KTP method transfers knowledge in Section 4.2. Finally, we will explain how we combine the ERM and the KTP method to construct the ERM-KTP unlearning method in Section 4.3.

4.1. Entanglement-Reduced Mask (ERM) Structure

The most representative knowledge the ML models learn is the features extracted by the convolutional layers. However, the features of different classes are entangled, so the key features of different classes are always highly overlapping. As a result, it is difficult to transfer the knowledge of target classes without affecting others.

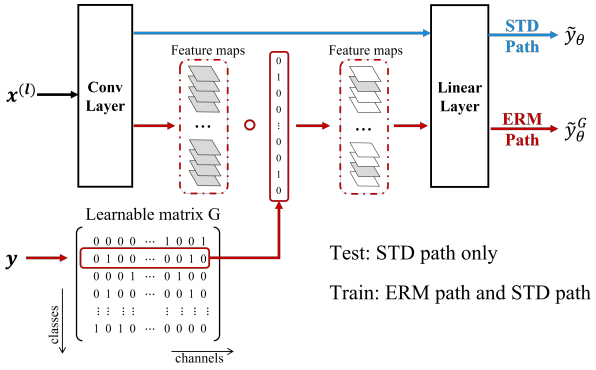


Figure 1. An overview of our proposed ERM structure.

In order to reduce the entanglement of knowledge, we propose an Entanglement-Reduced Mask (ERM) structure as a learnable layer of the CNNs inspired by the model interpretability work [12]. The learnable ERM structure can be represented as a matrix $G \in \mathbb{R}^{C\times K}$, where C is the number of classes and K is the number of filters. Each element $G_c^k \in [0,1]$ represents the relevance between the k-th filter and the c-th class, and a higher value of G_c^k denotes a closer correlation. Given a data point $(x \in \mathcal{X},\ y \in \mathcal{Y})$ as an input, we use the y-th row $G_y \in \mathbb{R}^{1\times K}$ as a mask vector multiplied to the activation maps $\mathcal{M} = (\mathcal{M}_1, \cdots, \mathcal{M}_K)$ of the last convolution layer (before the fully connected layer) to shut down irrelevant channels. Formally, we can define the above multiply operation as

$$A \circ B = (A_1 B^1, \cdots, A_{d_1} B^{d_1}), \tag{2}$$

in which $A \in \mathbb{R}^{d_1\times d_2\times d_3}$, $B \in \mathbb{R}^{1\times d_1}$, $A_i \in \mathbb{R}^{d_2\times d_3}$, and $B^i \in \mathbb{R}$.

For a CNN with an ERM structure (ERM-CNN), we set two forward propagation paths of an ERM-CNN with parameters θ: (1) the STD path with the predicting probability vector $\tilde{y}_\theta$; (2) the ERM path with mask matrix G predicting $\tilde{y}_\theta^G$, as shown in Fig. 1. In the training phase, we train the network alternately through the STD and ERM paths, while only the STD path is used in the validation phase.

Then, we will discuss the details of training an ERM-CNN. We first define an ideal ERM structure in which each row of the matrix G is at least one element equal to 1, and each row is orthogonal. In this ideal state, the knowledge of each class learned by the model is not entangled and independent of each other. Thus, the training objectives are to train an ERM-CNN with an ideal ERM structure, and its classification performance is comparable to the STD-CNNs. To train a model with the ideal ERM, we first consider its validation accuracy through two paths and ensure that at least one element of each row of G equals 1, *i.e.,* $\|G_c\|_\infty = 1$. We can get the following formulation:

$$\min_{\theta,G} L_0(\theta) = \text{CE}(y||\tilde{y}_\theta) + \lambda_1 \text{CE}(y||\tilde{y}_\theta^G) \quad s.t.\ G \in V_G, \tag{3}$$

where CE denotes the cross entropy loss and $V_G = \{G \in \mathbb{R}^{C\times K} : \|G_c\|_\infty = 1\}$. Furthermore, we introduce a regulation term to encourage the sparsity of the matrix G, and encourage the L_1-norm of G not to exceed the preset upper bound α and has no effect when $\|G\|_1 < \alpha$. Apparently, we should set $\alpha \geqslant K$ because $\|G_c\|_\infty = 1$ ensures $\|G\|_1 \geqslant K$. As described above, we can formulate the regulation term as

$$\min_G L_1(\theta) = \|\text{ReLU}(\|G\|_1 - \alpha)\|_p \quad s.t.\ G \in V_G, \tag{4}$$

in which $\|\cdot\|_p$ indicates p-norm. Moreover, we introduce the inner product regulation to encourage each row of G to be orthogonal to the other as

$$\text{InP} = \sum_{c_1=0}^{C} \sum_{\substack{c_2=0,\\ c_1<c_2}}^{C} (G_{c_1} \cdot G_{c_2}). \tag{5}$$

To facilitate the solution in practice, we replace Eq. (5) with the following equivalent form as our inner product regulation term:

$$\min_G L_2(\theta) = \sum_{c_1=0}^{C} \sum_{\substack{c_2=0,\\ c_1<c_2}}^{C} (GG^T)_{c_1}^{c_2} \quad s.t.\ G \in V_G, \tag{6}$$

in which $(GG^T)_{c_1}^{c_2}$ denotes the c_1-th row and the c_2-th column of the matrix (GG^T). When we minimize $L_2(\theta)$, it will tend to 0 if $G_c^k \in [0,1]$, *i.e.,* each row of G will be orthogonal to the other. Essentially, the inner product describes the similarity between two vectors as $\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| cos\langle \vec{a}, \vec{b}\rangle$. When we minimize $L_2(\theta)$, the cosine value and their L_2-norm are minimized simultaneously.

To sum up, we formulate an optimization problem to train an ERM-CNN by combining Eq. (3), (4), and (6):

$$\begin{aligned}\min_{\theta,G} L(\theta) = &\text{CE}(y||\tilde{y}_\theta) + \lambda_1 \text{CE}(y||\tilde{y}_\theta^G) \\ &+ \lambda_2 \|\text{ReLU}(\|G\|_1 - \alpha)\|_p \\ &+ \lambda_3 \sum_{c_1=0}^{C} \sum_{\substack{c_2=0, \\ c_1<c_2}}^{C} (GG^T)_{c_1}^{c_2} \quad s.t.\ G \in V_G,\end{aligned} \tag{7}$$

in which the first and the second term ensure the classification performance, the third term ensures the sparsity of the matrix G, and the last term imposes a rigorous constraint on each row of G.

Algorithm 1: Alternate Training (AT)

for *e in epochs* **do**
  if $e \in ERM's\ epoch$ **then**
    $\tilde{y}_\theta^G \leftarrow$ prediction through the ERM path
    $\mathcal{L} \leftarrow \lambda_1 CE(y||\tilde{y}_\theta^G) + \lambda_2 L_1 + \lambda_3 L_2$
    $G \leftarrow G - \eta \frac{\partial \mathcal{L}}{\partial G}$
    $G_c \leftarrow G_c / \|G_c\|_\infty$
    $G \leftarrow G.clip(0, 1)$
  else
    $\tilde{y}_\theta \leftarrow$ prediction through the STD path
    $\mathcal{L} \leftarrow \text{CE}(y||\tilde{y}_\theta)$
  end
  $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$
end

To solve the above optimization problem in Eq. (7), we apply an approximate projected gradient descent (PGD) algorithm. After the matrix G is updated, G_c will be normalized by $G_c/\|G_c\|_\infty$ to ensure $\|G_c\|_\infty = 1$, and then each element of G_c will be clipped into the range $[0, 1]$. However, the joint training scheme may have some difficulties due to poor convergence. The forward propagation through the ERM path can make most features block, leading to a much poorer convergence than the STD path because of the weaker gradient. Therefore, we propose an alternate training (AT) scheme instead of the normal joint training scheme in that the forward propagation uses the STD path and ERM path alternately in different epochs. Specifically, we update G and θ by descend the gradient of $\lambda_1 \text{CE}(y||\tilde{y}_\theta^G) + \lambda_2 L_1 + \lambda_3 L_2$ in the epoch for ERM path, and we update θ with the gradient of $\text{CE}(y||\tilde{y}_\theta)$ in the STD path, as shown in Algorithm 1.

4.2. Knowledge Transfer and Prohibition (KTP)

After training an ERM-CNN as a pre-trained model, different classes' knowledge (features and parameters) is independent. When receiving the unlearning requests, we can transfer the knowledge of a single class separately via our proposed knowledge transfer and prohibition (KTP) method. Specifically, KTP transfers the knowledge of the non-target class of data in the origin model θ_T to the unlearned model θ_S while prohibiting the knowledge of the target data points. We can describe the KTP method as the following three processes: (1) convolution layers (θ_S^{cv} and θ_T^{cv}) knowledge transfer (CKT); (2) fully connected layers (θ_S^{fc} and θ_T^{fc}) knowledge transfer (FKT); (3) matrix G (G_S and G_T) knowledge transfer (GKT). Formally, we can define the KTP method as

$$\begin{aligned}&\text{CKT}: \theta_S^{cv} \overset{trans}{\longleftarrow} \theta_T^{cv}, \\ &\text{FKT}: \theta_S^{fc} \overset{trans}{\longleftarrow} \theta_T^{fc}, \\ &\text{GKT}: G_S \overset{trans}{\longleftarrow} G_T.\end{aligned} \tag{8}$$

In the CKT process, we transfer the knowledge of features extracted by the convolutional layers by a crafted channel mask $\bar{G}_{\mathbf{c}} \in \mathbb{R}^{1\times K}$. For the single target class $\mathbf{c}$ unlearning task, the mask can be defined as

$$\bar{G}_{\mathbf{c},i} = \begin{cases} 0, & G_{\mathbf{c},i} = 1 \\ 1, & G_{\mathbf{c},i} = 0 \end{cases}. \tag{9}$$

For the multiple target classes $\mathbf{c} = \{\mathbf{c}_0, \cdots, \mathbf{c}_{|c|}\}$ unlearning task, we calculate $\bar{G}_{\mathbf{c}_j}$ for each class j and then aggregate them as

$$\bar{G}_{\mathbf{c},i} = \begin{cases} 1, & (\sum_j^{|c|} \bar{G}_{\mathbf{c}_j})_i = |c| \\ 0, & (\sum_j^{|c|} \bar{G}_{\mathbf{c}_j})_i < |c| \end{cases}. \tag{10}$$

Formally, the CKT process can be formulated as an optimization problem:

$$\min_{\theta_S^{cv}} L_3(\theta_S^{cv}) = \text{MSE}(f(x; \theta_T^{cv}), g(x; \theta_S^{cv}) \circ \bar{G}_{\mathbf{c}}), \tag{11}$$

where $\text{MSE}(\cdot)$ is Mean Square Error, x is data points, $\circ$ is the multiply operation (Eq. (2)), and $f(;\theta_S^{cv})$ and $g(;\theta_S^{cv})$ denote the output of the origin model's last convolution layer with parameters θ_S^{cv} and the output from unlearned model's last convolution layer with parameters θ_S^{cv}, respectively. Note that the knowledge of the corresponding channels in the feature maps multiplied by the matrix G where the value equals one has been transferred. In contrast, the knowledge associated with values equal to 0 has been prohibited.

For the FKT process, we transfer the knowledge of parameters in fully connected (FC) layer $\theta^{fc} = \{\theta_\beta^{fc}, \theta_\gamma^{fc}\}$, in which $\theta_\beta^{fc} \in \mathbb{R}^{K\times C}$ denotes the weight, and $\theta_\gamma^{fc} \in \mathbb{R}^{1\times C}$ denotes the bias. The FKT process can be executed by directly modifying the original model's parameters of fully

connected layers as

$$\theta^{fc}_{S,\beta} = \theta^{fc}_{T,\beta} * \mathcal{A}_\beta, \quad \mathcal{A}_\beta = (\underbrace{(\bar{G}_c)^T, \cdots, (\bar{G}_c)^T}_{C}), \\ \theta^{fc}_{S,\gamma} = \theta^{fc}_{T,\gamma} * \mathcal{A}_\gamma, \quad \mathcal{A}_\gamma \in [0,1]^{1\times C}, \tag{12}$$

where all elements of $\mathcal{A}_\gamma$ are equal to 1 except the $\mathbf{c}$-th column, $(\cdot)^T$ denotes the matrix transpose operation, and $*$ denotes the hardamad product operation. Similar to the channel mask $\bar{G}_c$, the FC mask $\mathcal{A} = \{\mathcal{A}_\beta, \mathcal{A}_\gamma\}$ prohibits the knowledge of the FC layer related to the target class of data points.

Finally, we update the matrix G_S of the unlearned model through the GKT process by the following equation:

$$G_S = G_T * \mathcal{A}_G, \quad \mathcal{A}_G \in [0,1]^{C\times K}, \tag{13}$$

where only the $\mathbf{c}$-th row of the mask matrix $\mathcal{A}_G$ is equal to 0, and the others are equal to 1. The updated matrix G_S can visualize which classes of data points have been removed by checking which rows are equal to 0. We summarize the KTP method in Algorithm 2.

Algorithm 2: Knowledge Transfer and Prohibition (KTP)

// CKT
for *e in epochs* **do**
  $f(x;\theta^{cv}_T) \leftarrow$ forward propagation through θ_T
  $g(x;\theta^{cv}_S) \leftarrow$ forward propagation through θ_S
  $\mathcal{L} \leftarrow \text{MSE}(f(x;\theta^{cv}_T) \circ \bar{G}_c, g(x;\theta^{cv}_S))$
  $\theta^{cv}_S \leftarrow \theta^{cv}_S - \eta \frac{\partial \mathcal{L}}{\partial \theta^{cv}_S}$
end
// FKT
$\theta^{fc}_{S,\beta} = \theta^{fc}_{T,\beta} * \mathcal{A}_\beta$
$\theta^{fc}_{S,\gamma} = \theta^{fc}_{T,\gamma} * \mathcal{A}_\gamma$
$\theta^{fc}_S = \{\theta^{fc}_{S,\beta}, \theta^{fc}_{S,\gamma}\}$
// GKT
$G_S = G_T * \mathcal{A}_G$
// Unlearned Model
$\theta_S = \{\theta^{cv}_S, G_S, \theta^{fc}_S\}$

4.3. Machine Unlearning via ERM-KTP

Here, we will explain how we combine the ERM structure and KTP method to construct a class-specific knowledge-level machine unlearning approach. We first train the original ERM-CNNs by Algorithm 1. When receiving unlearning requests, we then transfer the remaining dataset's knowledge to the unlearned model by Algorithm 2. Finally, we complete the unlearning process by deleting the original model. The detailed ERM-KTP unlearning algorithm is shown in Algorithm 3.

Algorithm 3: ERM-KTP Unlearning

$\theta_T \leftarrow \text{AT}(\theta)$
while *receiving unlearning requests* **do**
  $\theta_S \leftarrow \text{KTP}(\theta_T)$
  Delete θ_T
  $\theta_T = \theta_S$
end

5. Experiments

In this section, we empirically evaluate the performance of the proposed unlearning method on three popular image classification benchmarks. All experiments are implemented with Python 3.8 and PyTorch 1.10.1 on a machine with Intel(R) Core(TM) i9-10980XE CPU, 32GB RAM, and a Nvidia 3090 GPU.

5.1. Datasets & Models

We evaluate the proposed unlearning method in three public image classification datasets, *i.e.,* CIFAR10, CIFAR100 [9], and Tiny-ImageNet [11]. The input size of CIFAR-10 and CIFAR-100 is 32×32, and they both contain $50,000$ training images and $10,000$ for validation. For Tiny-ImageNet, the size of image samples is 224×224. It contains 200 classes of samples, and each class has 500 samples for training, 50 for validation, and 50 for testing.

Moreover, the model structure is three common image classification deep neural networks, ResNet-20, ResNet-50 [6], and ResNeXt-50 [20]. The original models are trained for 200 epochs using Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, weight decay of 5e-4, and an initial learning rate of 0.1, divided by 10 after 100 and 150 epochs, respectively.

5.2. Effectiveness of the ERM Structure

In order to evaluate the effectiveness of ERM structure, we propose the following metrics:

- **Cosine similarity (CSI)** to analyze the orthogonality of each row of the matrix G as follows,

$$\text{CSI} = \sum_{c_1=0}^{C} \sum_{\substack{c_2=0,\\ c_1\neq c_2}}^{C} \left(\frac{G_{c_1} \cdot G_{c_2}}{\|G_{c_1}\|\|G_{c_2}\|}\right). \tag{14}$$

- L_1**-density** to evaluate the sparsity of the matrix G as

$$L_1\text{-density} = \frac{\|G\|_1}{CK}. \tag{15}$$

Table 1 shows that the CNNs with the ERM structure have slightly better classification performance than the STD

CNNs. Moreover, the L_1-density convergences to α, and CSI convergences close to 0, meaning each class of data is independent to the other. These metrics quantitatively demonstrate ERM's effectiveness in learning a sparse mask matrix to reduce the entanglement for the following unlearning task without sacrificing classification accuracy.

Table 1. Metrics of the STD-CNNs and ERM-CNNs. The first row of each two stand for STD-CNNs and the second one stand for ERM-CNNs. Acc_{val} is the original model's classification accuracy on the validation dataset.

Dataset	Model	C	K	CSI	L_1-density	Acc_{val}
CIFAR10	ResNet-20	–	–	–	–	89.6%
		10	64	0.0	0.1	92.6%
CIFAR100	ResNeXt-50	–	–	–	–	76.2%
		100	2,048	0.0	0.01	77.6%
Tiny-ImageNet	ResNet-50	–	–	–	–	49.3%
		200	2,048	0.0	0.005	53.4%

Moreover, we evaluate if the ERM structure can reduce or eliminate the knowledge entanglement across classes by prohibiting each class's knowledge successively during the ERM path's forward propagation. As shown in Table 2, the classification accuracy will degrade into 0.0% for the knowledge-prohibited classes, while the accuracy will remain the same for the knowledge-permitted classes.

Table 2. Accuracy of the data points with prohibited or permitted Knowledge. $Acc^{pro}_{(\cdot)}$ and $Acc^{per}_{(\cdot)}$ denote the accuracy of the knowledge-prohibited classes and the knowledge-permitted classes, respectively.

Dataset	Model	Acc^{pro}_{train}	Acc^{per}_{train}	Acc^{pro}_{val}	Acc^{per}_{val}
CIFAR10	ResNet-20	0.0%	97.1%	0.0%	91.8%
CIFAR100	ResNeXt-50	0.0%	99.9%	0.0%	78.1%
Tiny-ImageNet	ResNet-50	0.0%	99.9%	0.0%	53.4%

We also evaluate the time overhead and memory requirements of the ERM CNNs in the training phase. The computation overhead is the activation map multiplied by the corresponding y-th row of the matrix G and computing the regulation terms L_1 and L_2. As shown in Table 3, the time overhead of the ERM structure is comparable to the standard (STD) CNNs, and the additional memory requirements are negligible, which demonstrates that ERM structure has great scalability and can be applied to any CNN structures.

5.3. Effectiveness of the ERM-KTP Unlearning Method

We utilize some common metrics to evaluate the effectiveness of the ERM-KTP unlearning method as follows.

- Acc_{D_r} is the original model's classification accuracy on the training set of the remaining classes.
- Acc_{val} is the original model's classification accuracy on the validation dataset of the remaining classes.
- Acc'_{D_r} measures the unlearned model's performance on the training set of the remaining classes.
- $\Delta Acc_{D_r} = Acc'_{D_r} - Acc_{D_r}$ indicates the difference between the accuracy of the remaining set before and after the unlearning operation.
- $\Delta Acc_{val} = Acc'_{val} - Acc_{val}$ indicates the difference between the accuracy of the validation set before and after the unlearning operation.

Table 3. Time overhead and memory requirements of the ERM-CNNs and the STD-CNNs in the training phase.

Dataset	Model	Structure	Time	Memory
CIFAR10	ResNet-20	STD	1206s	2867MiB
		ERM	1541s	2867MiB
CIFAR100	ResNeXt-50	STD	9447s	19223MiB
		ERM	9983s	19275MiB
Tiny-ImageNet	ResNet-50	STD	5025s	6769MiB
		ERM	6164s	6803MiB

Then, we conduct experiments in the case of unlearning 10%, 40%, and 80% of classes, respectively. Table 4 shows that Acc_{D_u} equals 0.0% in all cases, demonstrating that ERM-KTP can effectively unlearn target data points. Furthermore, ΔAcc_{D_r} are close to 0.0% and ΔAcc_{D_r} is greater than 0.0% in most cases. These show that ERM-KTP can unlearn target data without affecting the model classification performance, demonstrating the high fidelity of ERM-KTP. In summary, ERM-KTP performs well on all these datasets and model structures, demonstrating the scalability of ERM-KTP.

5.4. Comparison with Baselines

To further prove the advantage of ERM-KTP, we compare ERM-KTP with the following baseline approaches:

- **Retrain from scratch (RfS):** Retrain model from scratch with remaining data points $\mathcal{D}_r$.
- **SISA:** Retrain the sub-model whose training set contains the target data points [1].
- **Fine-tune:** Fine-tune the model on the remaining data $\mathcal{D}_r$ using a slightly larger learning rate.
- **Random Labels (RL):** Fine-tune the model on $\mathcal{D}$ by randomly resampling labels corresponding to images belonging to the remaining data points $\mathcal{D}_r$.
- **Amnesiac Unlearning:** Track gradient in every training batch. Then, update the model by subtracting the gradients when the batch is marked for removal [4].

Table 4. Performance evaluation of ERM-KTP unlearning method. #Classes denotes the number of the unlearned classes of data points

Dataset	Model	#Classes	Acc_{D_r}	Acc_{val}	Acc_{D_u}	Acc'_{D_r}	ΔAcc_{D_r}	ΔAcc_{val}
CIFAR10	ResNet-20	0	96.8%	92.6%	–	–	–	–
		1	96.6%	91.3%	0.0%	95.3%	-1.3%	-1.3%
		4	97.4%	96.1%	0.0%	97.3%	-0.1%	+3.5%
		8	98.4%	97.8%	0.0%	98.5%	+0.1%	+3.1%
CIFAR100	ResNeXt-50	0	99.9%	77.6%	–	–	–	–
		10	99.9%	76.1%	0.0%	98.1%	-1.8%	-0.6%
		40	99.9%	79.4%	0.0%	98.4%	-1.5%	+1.1%
		80	99.9%	86.4%	0.0%	99.8%	-0.1%	+7.6%
Tiny-ImageNet	ResNet-50	0	99.9%	53.4%	–	–	–	–
		20	99.9%	52.6%	0.0%	97.6%	-2.3%	-0.3%
		80	99.9%	53.8%	0.0%	98.3%	-1.6%	+1.5%
		160	99.9%	59.3%	0.0%	97.6%	-2.3%	+4.4%

- **Fisher Unlearning:** Perform a corrective Newton step using the Fisher Information Matrix. Add fisher noises sampled from Gaussian distribution to the weight to scrub the information [3].

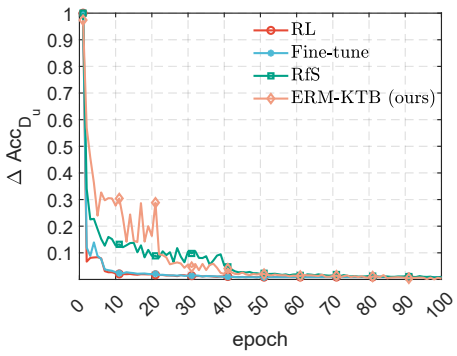
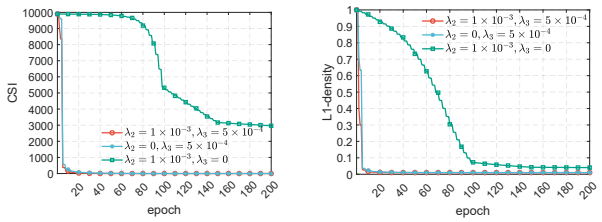


Figure 2. Train the unlearned model with the training datasets, and ΔAcc_{D_u} denotes the difference between the accuracy of the unlearning dataset on the relearned model and the accuracy on the original model on CIFAR10 and ResNet-20.

Then, we conduct the comparison experiments for 40% classes of data points removed. As shown in Table 5, the Amenesiac method [4] stores the gradients of almost every batch, and the model's performance degrades to random guess because it makes the model return to its initial state. For SISA [1], we need to retrain all shards to unlearn target classes of data points because every shard contains the target data points. Also, SISA's validation accuracy is lower by about 6% than STD CNNs due to the aggregation operation. Though the literature [3] reports that the Fisher method performs well on All-CNN [17] and CIFAR-10, our experimental results show that it is not suitable for complicated CNNs, *e.g.,* ResNet-20, ResNet-50 [6], and ResNeXt-50 [20].

Note that fine-tune and random labels (RL) methods show great performance on ΔAcc_{D_r} and ΔAcc_{val}. However, these two methods are only *an output-level unlearning*, *i.e.,* they only change the model's output of the target data points but not remove the corresponding knowledge. To prove that, we fine-tune the unlearned model to observe how long the model can relearn the *forget* knowledge. As shown in Fig. 2, the Fine-tune and RL method restore the performance on the unlearning dataset at about the 50th epoch, while the relearn time of ERM-KTP method is similar to the RfS method, *i.e.,* they both recover the performance around the 100th epoch. It demonstrates that ERM-KTP can completely remove the target data points like the RfS method, *i.e.,* ERM-KTP is *a knowledge-level unlearning method*.



(a) CSI (b) L_1-density

Figure 3. Ablation experiments of our proposed regulation terms on CIFAR100 and ResNet-50.

For the efficiency comparison, we conduct the experiments on CIFAR10 and ResNet-20 because some methods can not be implemented on the other two settings, *e.g.,* Fisher method needs more than 200 hours in Tiny-ImageNet. As shown in Table 6, ERM-KTP improves by 5× compared to RfS, 5.4× compared to SISA, and 42× compared to Fisher. Though the time overhead of Fine-tune is less than ERM-KTP, Fine-tune method is only an output-level unlearning method, as mentioned above. In summary, compared with these baseline approaches, ERM-KTP is more effective, more efficient, and has better scalability.

Table 5. Unlearning performance comparison with baseline approaches when unlearning 40% classes of data points.

Dataset	Model	Approach	Acc_{D_r}	Acc_{val}	Acc_{D_u}	Acc'_{D_r}	ΔAcc_{D_r}	ΔAcc_{val}
CIFAR10	ResNet-20	STD	99.9%	93.7%	–	–	–	–
		RfS	99.9%	96.3%	0.0%	99.8%	-0.1%	+2.6%
		Fine-tune	99.9%	97.0%	0.0%	99.8%	-0.1%	+3.3%
		RL	99.9%	97.0%	0.0%	99.8%	-0.1%	+3.3%
		SISA	92.4%	91.0%	0.0%	92.8%	+0.4%	+0.2%
		Fisher	99.9%	54.0%	48.4%	53.0%	-46.9%	-25.1%
		Amnesiac	99.9%	9.0%	0.0%	15.4%	-84.5%	-84.7%
		ERM-KTP (ours)	97.4%	96.1%	0.0%	97.3%	-0.1%	+3.5%
CIFAR100	ResNeXt-50	STD	99.9%	78.9%	–	–	–	–
		RfS	99.9%	79.6%	0.0%	99.9%	0.0%	+0.7%
		RL	99.9%	78.1%	0.0%	99.7%	0.0%	-0.8%
		Fisher	99.9%	1.0%	0.0%	1.0%	-98.9%	-77.9%
		Fine-tune	99.9%	83.4%	0.0%	99.9%	0.0%	+4.5%
		Amnesiac	99.9%	0.89%	0.0%	1.0%	-98.9%	-77.8%
		ERM-KTP (ours)	99.9%	79.4%	0.0%	98.4%	-1.5%	+1.1%
Tiny-ImageNet	ResNet-50	STD	99.9%	48.3%	–	–	–	–
		RfS	99.9%	45.1%	0.0%	99.9%	0.0%	-3.2%
		RL	99.9%	43.4%	0.0%	96.7%	-3.2%	-4.9%
		SISA	67.6%	52.8%	0.0%	68.0%	+0.4%	-1.3%
		Fine-tune	99.9%	50.0%	0.0%	99.9%	0.0%	+1.7%
		Amnesiac	99.9%	0.3%	0.8%	0.7%	-99.2%	-48.0%
		ERM-KTP (ours)	99.9%	53.8%	0.0%	98.3%	-1.6%	+1.5%

Table 6. Unlearn time comparison with baselines on CIFAR10 and ResNet-20.

Approach	Time	ΔAcc_{D_r}	ΔAcc_{val}
RfS	668s	-1.7%	+1.7%
SISA	672s	+0.4%	+0.2%
Fine-tune	492s	-0.2%	+2.4%
RL	68s	-1.5%	+1.4%
Fisher	5162s	-46.9%	-25.1%
ERM-KTP (ours)	124s	-1.6%	+2.0%

5.5. Ablation Experiments

To further demonstrate that our proposed regulation terms are effective, we conduct an ablation experiment on CIFAR100 and ResNet-50 by setting the hyper-parameters as: (1) $\lambda_1 = 1$, $\lambda_2 = 1 \times 10^{-3}$, and $\lambda_3 = 0$; (2) $\lambda_1 = 1$, $\lambda_2 = 0$, and $\lambda_3 = 5 \times 10^{-4}$; (3) $\lambda_1 = 1$, $\lambda_2 = 1 \times 10^{-3}$, and $\lambda_3 = 5 \times 10^{-4}$. Fig. 3 demonstrates that applying both L_2 and L_3 achieves the best performance and convergence to the optimal value. Only using the L_2 regulation term (Eq. (4)), the L_1-density tends to 0.01, but the CSI convergences to around 3,500. It illustrates that the L_1-norm regulation term can find the spars solution, but the L_1-norm does not constrain the relative position, which leads to the high value of CSI. However, the L_1-norm can make both CSI and L_1-density convergence to a lower value when the inner-product regulation term (Eq. (6)) is used. Furthermore, the inner-product can make both two values descend more quickly.

6. Conclusion

In this paper, we propose an interpretable knowledge-level unlearning method, ERM-KTP, for the class-specific unlearning task. We propose the ERM structure to tackle the challenge of high knowledge entanglement. Then, we transfer the knowledge of the remaining set from the original model to the unlearned model via our proposed KTP method. Finally, we delete the original model to accomplish the unlearning task. The interpretability brought by the EMR structure and the crafted masks in KTP enhances the reliability of the unlearning. Extensive experiments demonstrate the effectiveness, fidelity, efficiency, and scalability of our proposed ERM-KTP unlearning method. For future work, we attempt to solve the entanglement of each single data point to tackle the sample-specific unlearning task.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant 61960206014, 62102300, and 62121001, in part by the China 111 Project under Grant B16037, in part by the Fundamental Research Funds for the Central Universities under Grant XJS211510.

References

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021. 1, 6, 7

[2] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019. 1

[3] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020. 1, 2, 7

[4] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021. 1, 2, 6, 7

[5] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3832–3842, 2020. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 7

[7] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022. 1

[8] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017. 2

[9] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 5

[10] Himabindu Lakkaraju, Nino Arsov, and Osbert Bastani. Robust and stable black box explanations. In *International Conference on Machine Learning*, pages 5628–5638. PMLR, 2020. 1

[11] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 5

[12] Haoyu Liang, Zhihao Ouyang, Yuyuan Zeng, Hang Su, Zihao He, Shu-Tao Xia, Jun Zhu, and Bo Zhang. Training interpretable convolutional neural networks by differentiating class-specific filters. In *European Conference on Computer Vision*, pages 622–638. Springer, 2020. 2, 3

[13] James Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020. 2

[14] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018. 1

[15] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017. 1

[16] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 1

[17] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 7

[18] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4007–4022, 2022. 1, 2

[19] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017. 1

[20] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5, 7

[21] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8827–8836, 2018. 1

[22] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 1