This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

CLUE: Bringing Machine Unlearning to Mobile Devices

A.Q.M. Sazzad Sayyed*, Nathaniel D. Bastian†,
Michael De Lucia‡, Ananthram Swami‡ and Francesco Restuccia*

*Northeastern University † United Stated Military Academy ‡Army Research Laboratory

Corresponding Author: sayyed.a@northeastern.edu

Abstract

Class-level machine unlearning has been proposed to address security and privacy issues of deep neural networks (DNNs). However, existing approaches either exhibit low performance or have excessive computation/storage requirements. This makes them inapplicable in mobile computing scenarios, where computation and memory are severely constrained yet unlearning has to be performed frequently and effectively. This limitation is mainly due to the usage of a retain dataset, i.e., a sub-dataset containing the knowledge that the DNN should maintain after the unlearning. In this paper, we propose CLUE, an unlearning algorithm that does not require a retain dataset. Our key idea is to treat inputs coming from the forget class as out-of-distribution data and to use knowledge distillation to impose this constraint on the updated DNN. We have experimentally evaluated CLUE on Resnet-20, ViT-Base, and ViT-Large DNNs trained on CIFAR10, CIFAR100, and VGGFace2 datasets. We have also implemented CLUE on Raspberry PI and compared the power consumption and latency of CLUE with respect to several existing baselines. We show that CLUE improves power consumption by 68% and latency by 90% while improving the unlearning performance by up to 4.74%. The implementation code can be found in this github repository[1].

1. Introduction and Motivation

Addressing security and privacy concerns of DNNs is of fundamental importance to guarantee continued acceptance of artificial intelligence by the broader population. Indeed, as companies collect user data to improve their DNNs, attacks based on membership inference [18] and model inversion [39] can lead to theft of personal information [31]. To ensure privacy, existing laws such as the California Consumer Privacy Act (CCPA) in the US [23] and the General Data Protect Regulation (GDPR) in the EU [33, 37] require companies to delete the data of individual consumers on request, also known as the *right to be forgotten*. This issue has led to the emergence of *machine unlearning*, which aims at removing from a trained DNN the influence of a subset of data used for training without requiring complete retraining [3, 28]. Specifically, machine unlearning can be used to remove a subset of the training data that has been identified as corrupted or poisoned (*item removal*) [3], removing specific features (*feature removal*) [14, 35], removing an entire class (*class removal*) [30], and forgetting specific tasks (*task removal*) [24].

In this paper, we focus on class-level machine unlearning (in short, referred to as "unlearning"). One of the most important applications of class-level machine unlearning is continual learning in constrained mobile systems, where the neural network must forget outdated object classes to remove irrelevant knowledge and make room for new tasks in order to adapt to new environments [13]. Beyond mobile systems, prior work has proposed class unlearning to remove facial identification [6, 12, 30], defense against backdoor attacks [26, 36], and malicious classes [5]. Not surprisingly, removing classes in real time on mobile devices address several real-world applications. For example, in smart-home scenarios where cameras are used to identify household members, sometimes a particular face needs to be removed while still recognizing other household members – for example, parents may invoke the Children's Online Privacy Protection Act (COPPA) to immediately delete a minor's data [1]. Another example regards sensors deployed for wildlife monitoring, where conservation policies often mandate discarding images of endangered species after rapid annotation while minimizing bandwidth [33].

These scenarios demonstrate that being able to jettison a class quickly, entirely on device, and with no access to the retain set is a pressing requirement for several applications. In this scenarios, unlearning needs to be seen not as a scheduled maintenance task, but something that needs to be performed as fast as possible and with as low resource consumption as possible. As explained in Section 1 of the Supplement, prior work has focused on addressing settings where unlearning is performed on devices where computation and memory resources are not an issue. As such, they assume access to a "retain dataset", which is the original dataset minus the data to be unlearned. However, using a

[1] https://github.com/Restuccia-Group/CLUE.git

retain set increases the complexity and energy consumption [2] of the unlearning process. Moreover, data access may be restricted. Many real-world deployments discard training data due to privacy laws (e.g., GDPR) or operational policies (e.g., healthcare, defense). In such cases, the retain set is no longer available; this is especially true for pretrained models. Fetching or storing it again may be infeasible when performing on-device unlearning in data-sparse, privacy-sensitive, and offline environments.

Summary of Major Contributions

• We propose *Class-Label Unlearning for Efficiency* (CLUE) that *does not require a retain dataset yet outperforms the state of the art*. Our key intuition is that since out-of-distribution (OOD) inputs are by definition drawn from a distribution different than the training distribution, we can modify the DNN so that inputs coming from a target class to be unlearned (which constitute the forget set) will be treated as OOD. We believe, this re-conceptualization of the class unlearning adds a novel perspective to the problem and to our knowledge, this has not been explored by prior work. Further, to achieve this goal, as explained in Section 3, CLUE uses a unique and innovative combination of energy-based loss function [7], knowledge distillation [17] and gradient masking [34].

• We supplement the conceptual novelty of CLUE extensively benchmarking its performance on both Resnet-20 and Vision Transformer (ViT) such as ViT-Base and ViT-Large trained on CIFAR10, CIFAR100, and VGGFace2 datasets. The performance of CLUE has been characterized on Raspberry Pi 5 and its power consumption and latency compared with respect to several existing baselines. Our results show that CLUE improves by up to 68%, 90% and 4.74% (difference in average performance gap with "retrain from scratch") in terms of power consumption, latency and unlearning performance while requiring up to 30% less memory than existing approaches. When evaluated against the nearest state-of-the-art method, CLUE achieves an average relative improvement of 27.28%, with gains as high as 70% on the ViT-Base architecture with the CIFAR-10 dataset. These results demonstrate that CLUE substantially outperforms existing approaches in both efficacy and efficiency, in data access constrained setting without access to retain set.

Paper Organization. We first introduce preliminary notions and definitions in Section 2, and then provide details of the proposed CLUE in Section 3. Next, we present the experimental setup in Section 4 and the related experimental evaluation in Section 5. We compare energy consumption and latency of CLUE in Section 6 and discuss future directions in Section 7. We defer the discussion of related works to supplementary Section 1.

2. Preliminary Notions and Definitions

Class-Level Machine Unlearning. We consider class-level machine unlearning class-level machine unlearning (CMU) in the context of supervised learning. We denote a DNN by $\mathcal{F}_\theta$, where θ is the set of parameters. We define a learning algorithm as $\mathcal{A} : \theta \times \mathbf{D} \rightarrow \theta^*$, which maps a dataset $\mathbf{D}$ and DNN parameters θ to a set of parameters optimized for the corresponding dataset θ^*. The dataset $\mathbf{D}$ consists of N input-label pairs $\{\mathbf{x_i}, y_i\}_{i=1}^{i=N}$ sampled from the space $\mathcal{X} \times \mathcal{Y}$ which follows distribution $\mathcal{D}_{train}$. Here, $\mathcal{X}$ denotes the input space and $\mathcal{Y} = \{1, 2, \cdots, K\}$ represents the label space. We denote *forget dataset* with $\mathbf{D}_f$ so that the *retain dataset* is $\mathbf{D}_r = \mathbf{D} \backslash \mathbf{D}_f$. We also define the *forget class* $c_f \in \mathcal{Y}$ (assuming $\mathbf{D}_f$ constitutes of a single class following the literature) as the class to be unlearned. Given an input $\mathbf{x}_i$, vector $\mathbf{z}_i = \mathcal{F}_{\theta^*}(\mathbf{x_i})$ represents the logits of the DNN.

Our goal is to remove the information related to the forget dataset $\mathbf{D}_f$ from the trained DNN $\mathcal{F}_{\theta^*}$ *without retraining from scratch* and *without accessing* the retain dataset $\mathbf{D}_r$ and without significantly affecting performance on $\mathbf{D}_r$. Let us denote the parameters of the DNN retrained only on the retain-set by θ^r and an unlearning algorithm with $\mathcal{U}(\theta^*, \mathbf{D}_f)$. The goal of $\mathcal{U}(\cdot)$ is to map the θ^* to another set of parameters θ^f. Due to the stochastic nature of DNN training, θ^r will be different for different training configurations or weight initializations - even for the same performance of the DNN - forming a distribution for a given dataset. As a result, *exact unlearning* is defined as the case in which θ^f comes from the same distribution as θ^r. We define as *approximate unlearning* the case where $d(\mathcal{F}_{\theta^f}(\mathbf{x}), \mathcal{F}_{\theta^r}(\mathbf{x})) \simeq 0$, where $d(\cdot)$ is an appropriate distance metric.

OOD Detection. Since the training data follow distribution $\mathcal{D}_{train}$, an OOD sample is defined as an input-label pair $\{\mathbf{x_i}, y_i\}$ that does not follow $\mathcal{D}_{train}$. This may happen when (i) label $y_i \notin \mathcal{Y}$, meaning the class does not belong to the label space and thus a shift in the semantic content of the input has happened (e.g. emergence of a novel class) – also known as *semantic OOD* which indirectly entails change in input distribution $\mathcal{D}_{\mathcal{X}}$; (ii) label $y_i \in \mathcal{Y}$ and $\mathbf{x}_i \notin \mathcal{X}$, meaning the distribution of the input does not follow the training distribution – also known as *covariate-shifted OOD* or *non-semantic OOD*.

Energy Function in OOD Detection. The energy function introduced by Liu et al. in [25] is a common approach for OOD detection. Specifically, $p(\mathbf{x})$ - the likelihood that an input $\mathbf{x}$ comes from In-Distribution (ID) should be low for an OOD input. As such, Liu et al. drew on the similarity of the formulation of the softmax probability of the DNN to the Gibbs distribution. Specifically, Liu et al. [25] have shown that the log-probability of the input $p(\mathbf{x})$ is affinely

related to the *Helmholtz free energy* (HFE) defined in [22] and given by Eq. (1).

$$E(\mathbf{x}; \mathcal{F}_\theta) = -T \cdot \log \left(\sum_{y_i} e^{\mathcal{F}_\theta^{y_i}(\mathbf{x})/T} \right). \quad (1)$$

As such, HFE can be used to characterize the OOD samples, as the ID samples usually have lower HFE than the OOD samples. As explained in Section 3, CLUE uses this as a proxy for the approximate unlearning outcome.

3. Overview of CLUE

Fig. 1 presents an overview of CLUE, which is based on (i) an *energy loss function* $\mathcal{L}_E$ inspired by the HFE for OOD detection; (ii) a *knowledge distillation (KD) loss function* $\mathcal{L}_{MU}$ and (iii) a *gradient masking* process that excludes the gradients while unlearning with a mask M which is constructed based on the weight salience of the forget dataset. We describe the three parts next.

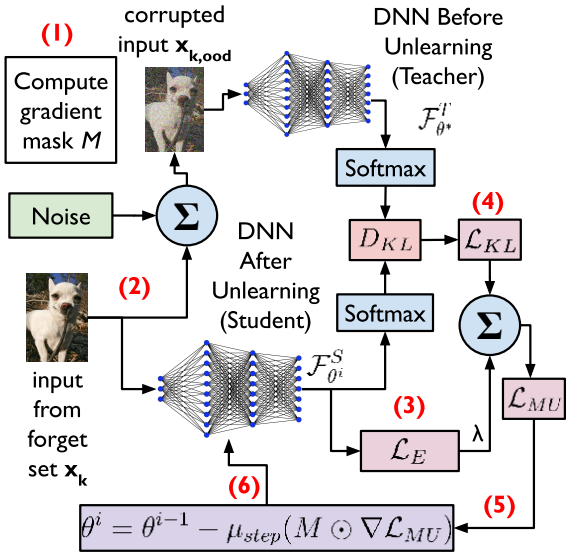


Figure 1. Overview of CLUE. We highlight the six unlearning steps (1)-(6), which are summarized in Section 3.1. After E unlearning iterations, the resulting $\mathcal{F}_{\theta_E}^S$ becomes the final unlearned model $\mathcal{F}_{\theta_f}^S$.

Energy Loss. An exact unlearning algorithm $\mathcal{U}$ maps the optimized parameters of a DNN $\mathcal{F}_{\theta^*}$ into θ^r. Since exact unlearning is challenging, we consider approximate unlearning. Our approach is to match the output of the unlearned DNN $\mathcal{F}_{\theta^f}$ to that of the DNN $\mathcal{F}_{\theta^r}$ trained on the retain dataset only. The key issue is that $\mathcal{F}_{\theta^r}$ is not available since we do not assume to have access to the retain-set. As such, we notice that after unlearning a particular class $c_f \in \mathcal{Y}$, the samples from the forget dataset $\mathbf{D}_f$ should behave the same as any other OOD samples. As a result, we design an energy loss in Eq. (2) to make the samples from $\mathbf{D}_f$ behave like OOD for the updated model:

$$\mathcal{L}_E = \frac{1}{|\mathbf{D}_f|} \sum_{\mathbf{x}_k \in \mathbf{D}_f} \sum_{y_i} e^{\mathcal{F}_{\theta^*}^{y_i}(\mathbf{x}_i)/T}, \quad (2)$$

where $|\mathbf{D}_f|$ denotes the cardinality of the forget dataset. From Section 2, Eq. (1), we know that OOD samples have higher HFE than ID samples – in other words, they have lower value of the partition function given by $\sum_{y_i} e^{\mathcal{F}_{\theta^*}^{y_i}(\mathbf{x}_i)/T}$. Thus, the forget dataset can be approximated as OOD data by minimizing this partition function for the forget dataset $\mathbf{D}_f$.

KD Loss. While the energy loss can induce OOD behavior on the forget dataset, it does not provide any direct guidance regarding the posterior probability of the DNN. This adversely affects the retain classes as we show in Section 5. We hypothesize that this adverse effect is elicited from the unguided change in the logit distribution of the retain classes during unlearning. To make the update of the DNN smoother by introducing guidance, we use KD by setting the DNN before and after unlearning as the teacher and student DNN, respectively. Using the KD loss provides the unlearned model information about the retain set embedded in the logits while aligning the representation of the unlearn class with that of heavily corrupted OOD-like inputs. This should counter balance any abrupt change in the logit space introduced by the energy loss. We define the parameters of the student DNN at i-th iteration as θ^i, where $\theta^0 = \theta^*$. The KD loss can be written as in Eq. (3):

$$\mathcal{L}_{KL} = \frac{\sum_{\mathbf{x}_i \in \mathbf{D}_f} D_{KL}(\sigma_s(\mathcal{F}_{\theta^*}^T(\mathbf{x}_{i,ood})), \sigma_s(\mathcal{F}_{\theta^i}^S(\mathbf{x}_i)))}{|\mathbf{D}_f|} \quad (3)$$

where σ_s denotes the softmax activation and D_{KL} denotes the Kullback-Leibler (KL) divergence. Variable $\mathbf{x}_{i,ood}$ denotes the corrupted version of the input $\mathbf{x}_i$ obtained with:

$$\mathbf{x}_{i,ood} = \mathbf{x}_i + \mathbf{n}; \ \mathbf{n} \sim \mathcal{P}, \quad (4)$$

Here, $\mathcal{P}$ denotes the noise distribution (for example, Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ or bimodal Bernoulli distribution $\mathcal{B}(p_s, p_p)$ for salt-and-pepper noise with probability of salt and pepper noise p_s and p_p respectively). The key intuition is that for sufficiently high noise power σ^2, the input $\mathbf{x}_i$ becomes akin to an OOD sample. In turn, this translates to guiding the student DNN to align its posterior with the posterior of $\mathcal{F}_{\theta^*}^T$ DNN fed with OOD data. Although $\mathcal{F}_{\theta^r}$ would be the ideal teacher DNN, we approximate the posterior $\mathcal{F}_{\theta^r}$ with that of $\mathcal{F}_{\theta^*}$, as we cannot access the former. Minimizing the KD loss can help the student learn the posterior distribution of the OOD data which is not possible with energy loss alone. With the two losses, our final loss function $\mathcal{L}_{MU}$ for unlearning is given by Eq. (5) where λ is a hyper-parameter which sets the relative contribution of the two losses.

$$\mathcal{L}_{MU} = \mathcal{L}_{KL} + \lambda \cdot \mathcal{L}_E \quad (5)$$

Gradient Masking. One of the key challenges of approximate unlearning is preserving the accuracy on the retain dataset. Hence, we adopt an interpretability based approach inspired by [9]. Specifically, we create a gradient mask M based on the saliency of the weights of the forget dataset. To find the latter, we employ the importance estimation tool proposed in [27]. We measure the importance R of a weight w according to Eq. (6):

$$R(w) = \left| w \frac{\partial \mathcal{L}_{CE}}{\partial w} \right|; \qquad M = 1(R(w) > \tau), \tag{6}$$

where $\mathcal{L}_{CE}$ denotes the cross entropy loss of the DNN for the forget dataset. Next, we employ a threshold τ to create M. The value in a position of the mask corresponding to a weight w of the DNN is 1 if its importance is above the threshold, and zero otherwise. We apply this mask to the gradient while updating the parameters of the student model. Masking the gradient ensures that the parameters containing the most information about the forget dataset are getting updated. This also helps preserve the performance of the DNN on the retain-set by leaving most of the weights which contain information about the retain-set untouched.

3.1. The CLUE Unlearning Algorithm

We summarize the unlearning steps in Algorithm 1. Initially, both teacher and student DNNs are initialized with $\mathcal{F}_{\theta^*}$. At the i-th update step, we compute the mask M for the forget dataset using $\mathcal{F}^S_{\theta^i}$ (**step 1**). Next, we feed a batch of samples from the forget dataset to the student DNN, as well as the corresponding corrupted version to the teacher DNN (**step 2**). We compute the energy loss $\mathcal{L}_E$ for the student DNN $\mathcal{F}^S_{\theta^i}$ (**step 3**) as well as the KD loss $\mathcal{L}_{KL}$ using the outputs of both the teacher and student DNNs (**step 4**), and combine them to get the total loss $\mathcal{L}_{MU}$ (**step 5**). Finally, we update θ^i as follows (**step 6**):

$$\theta^i = \theta^{i-1} - \mu_{step}(M \odot \nabla \mathcal{L}_{MU}) \,. \tag{7}$$

Here we abuse the notation a bit denoting both the updated student model after each batch and after each iteration epoch with θ^i. We perform these steps for E number of epochs. The end result is the unlearned model $\mathcal{F}_{\theta^f}$. Details about the hyper-parameters (e.g. number of unlearning epochs, noise configuration, gradient threshold etc.) are provided in Section 4 of the supplementary materials.

4. Experimental Setup

Datasets. Similar to existing work [6, 13, 30], we have conducted experiments on CIFAR10 [21] and VGGFace2 [4] to benchmark unlearning performance on the tasks of image classification and face recognition. We have also experimented with CIFAR100 [21]. For VGGFace2 dataset, we use a subset of 480 classes from [16]. We show the results in Section 5 for class#2 and defer the results on additional classes to supplementary Section 7.

Algorithm 1 The CLUE Unlearning Algorithm

Initialize: Teacher and Student DNNs with $\mathcal{F}_{\theta^*}$
for $i = 1$ to E **do**
  // θ^i is the Student DNN at the i-th iteration
  // Compute each element M_{θ_j} of mask M
  for each $\theta_j \in \theta^i$ **do**
    $M_{\theta_j} = 1(|\theta_j \cdot \frac{\partial \mathcal{L}_{CE}}{\partial \theta_j}| > \tau)$
  end for
  for each input $\mathbf{x}_k$ in batch $\mathbf{B}^i$ of $\mathbf{D}_f$ **do**
    $\mathbf{x}_{k,ood} = \mathbf{x}_k + \mathbf{n}$ // Noise $\mathbf{n} \sim$ Distribution $\mathcal{P}$
    $\mathbf{o}^T_k = \mathcal{F}^T_{\theta^*}(\mathbf{x}_{k,ood})$ // Teacher DNN $\mathcal{F}^T_{\theta^*}$
    $\mathbf{o}^S_k = \mathcal{F}^S_{\theta^i}(\mathbf{x}_k)$ // Student DNN $\mathcal{F}^S_{\theta^i}$
  end for
  $\mathcal{L}_{KL} = \frac{1}{|\mathbf{B}^i|} \sum_{\mathbf{x}_k} D_{KL}(\sigma_s(\mathbf{o}^T_k), \sigma_s(\mathbf{o}^S_k))$
  $\mathcal{L}_E = \frac{1}{|\mathbf{B}^i|} \sum_{\mathbf{x}_k} \sum_y e^{o^S_{k,y}(\mathbf{x}_k)/T}$
  $\mathcal{L}_{MU} = \mathcal{L}_{KL} + \lambda \mathcal{L}_E$
  $\theta^i = \theta^{i-1} - \mu_{step}(M \odot \nabla \mathcal{L}_{MU})$
end for
Output: Unlearned model $\mathcal{F}_{\theta^f} = \mathcal{F}_{\theta^E}$

DNN Architectures. For both CIFAR10 and CIFAR100, we have performed experiments with Resnet-20 [15] and ViT [8]. For VGGFace2, we have used the ViT-Large DNN architecture. The rationale behind choosing the DNNs is to show that CLUE can generalize across different types of DNN-namely Convolutional Neural Network (CNN) and transformer architecture. We choose a small CNN in this respect to investigate the impact of DNN capacity on CMU. Details of the training procedures of the DNNs can be found in Section 4 of the supplementary materials.

Baselines. For comparison purposes, we have implemented six machine unlearning methods along with the golden standard (retrain from scratch) - *Retrain:Random Labels (RL) [13]*, *Gradient Ascent (GA) [32]*, *Influence Unlearning (IU) [19, 20]*, *Boundary Unlearning (BU) [6]*, *Lipschitz Unlearning (LU) [11]*, and *Unlearning by Selective Impair and Repair (UNSIR) [30]*. We provide the details of the approaches in the supplementary Section 2. As only the BU and LU consider limited access to data, we adapt the rest of the baselines to use only the forget set. Bayesian Uncertainty (BU) encompasses two methods - Boundary Shrinking (BS) and Boundary expanding (BE).

Performance Metrics. Although there is no generally-adopted performance metric for CMU, we use the metrics (1)-(5) adopted in recent works [9, 20], plus edge computing metrics (6) and (7):

1. *Unlearning Accuracy (UA)* - measures the accuracy of the DNN on the forget dataset;
2. *Remaining Accuracy (RA)* - the accuracy of the DNN

on the retain-set;

3. *Testing Accuracy (TA)* - the accuracy of the DNN on the test set of the retained classes;
4. *Membership Inference Attack (MIA)* - infers whether a sample belongs to the training set or not;
5. *Run Time Efficiency (RTE)* - time (in seconds) taken to perform the unlearning process;
6. *Energy* - the amount of energy consumption for performing unlearning measured in joules;
7. *Latency* - the time taken (in seconds) to perform unlearning in the edge device.

For the MIA, following [20], we implement a confidence-based attack [29, 38] and use the percentage of forget samples correctly predicted as non-training samples as the metric of unlearning performance. We note that the first four metrics are reported as percentages. We compute the gap between the performance of Retraining from scratch (the golden standard) and the other unlearning methods for each of the metrics from 1-4 and report their average to summarize the overall performance of the methods.

5. Experimental Results

5.1. Unlearning Efficacy

Tables 1 and 2 compare the performance of CLUE against the baselines on the ViT-Base architecture trained on CIFAR10 and CIFAR100. As discussed in Sec. 4, we measure the performance of unlearning using the gap between the four metrics UA, RA, TA, MIA and *Retrain*. We also report the average of the gaps. We can observe from the results that CLUE outperforms all the baselines. Specifically, Boundary Shrinking (BS) and Boundary Expanding (BE) are the closest to CLUE in terms of average gap, while CLUE yields performance improvement by 4.44% and 3.95% on CIFAR10 benchmark and 2.78% and 4.6% on CIFAR100 benchmark for ViT-base.

Tab. 3 and Tab. 4 compare the performance of CLUE against the baselines on the Resnet-20 architecture trained on CIFAR10 and CIFAR100. We observe that CLUE generalizes across different architectures and performs better than the baselines. CLUE improves upon the nearest baseline by 4.74% on CIFAR10 and by 0.4% on CIFAR100. We observe that approaches such as *LU* [11] and *UNSIR* consistently perform poorly. This is expected, as the UNSIR approach forgets by directly learning from noise. A strong noise impairs the information about the forget dataset yet it also hampers the overall performance. LU processes each sample separately, and as such, the parameters of the batch-norm layers cannot capture the characteristics of the dataset, thus hurting generalization. We add further discussion on the results in supplementary Section 3. In summary, *although CLUE does not necessarily perform the best on each metric, its average performance across all considered metrics is the best among the baselines.* To emphasize the performance of CLUE, when evaluated in terms of relative improvement from the nearest state-of-the-art method, CLUE achieves 27.28% gains, reaching as high as 70% on the ViT-Base architecture with the CIFAR-10 dataset.

Table 1. Performance on ViT-Base trained on CIFAR10. The values in the bracket represent the gap with *Retrain*.

Unlearning Methods	UA	RA	TA	MIA	Average Gap
Retrain	0	99.8	96.92	100	
GA	94.57 (94.57)	**99.28 (0.52)**	96.55 (0.37)	5.42 (94.48)	47.485
RL	0.02 (0.02)	84.11 (15.69)	81.56 (15.4)	99.62(0.38)	7.8725
IU	97.26 (97.26)	99.35 (0.45)	97.10 (-1.08)	2.70 (97.3)	49.02
BE	**0(0)**	89.27 (10.53)	85.12 (11.8)	100(0)	5.58
BS	4.73 (4.73)	92.33 (7.47)	89.55 (7.37)	95.26 (4.74)	6.07
LU	**0 (0)**	11.11 (87.66)	11.11 (85.81)	**100 (0)**	43.36
UNSIR	98.91 (98.91)	99.26 (0.54)	**97.01 (-0.09)**	1.08 (98.92)	49.61
CLUE (Ours)	2.04 (2.04)	98.42 (1.38)	95.86 (1.06)	97.95 (2.05)	**1.63**

Table 2. Performance on ViT-Base trained on CIFAR100. The values in the bracket represent the gap with *Retrain*.

Unlearning Methods	UA	RA	TA	MIA	Average Gap
Retrain	0	99.24	85.72	100	
GA	0.44 (0.44)	**92.03 (7.21)**	**81.50 (4.22)**	99.55 (0.45)	**3.08**
RL	25.99 (25.99)	91.63 (7.59)	81.08 (4.68)	74 (26)	16.06
IU	86.22 (86.22)	92.72 (6.48)	82.18 (3.54)	13.77 (86.33)	45.64
BE	**0 (0)**	81.84 (17.60)	71.39 (14.33)	**100 (0)**	7.97
BS	**0 (0)**	85.32 (13.92)	75.01 (10.71)	**100 (0)**	6.15
LU	0 (0)	1.01 (98.23)	1.01 (84.71)	0 (0)	45.73
UNSIR	26.22 (26.22)	41.99 (57.25)	36.12 (84.60)	73.77 (26.33)	48.60
CLUE (Ours)	0.22 (0.22)	91.24 (8)	80.77 (4.95)	99.77 (0.33)	3.37

5.2. Computational Complexity

We provide details of the hardware specifications for these experiments in Section 4 of the supplementary material. Fig. 3 shows the run-time efficiency (RTE) results in logarithmic scale. We notice that every baseline is faster than *Retrain*. Even when fine-tuning a ViT-Base model pre-trained on Imagenet on CIFAR10 dataset, it takes about 1035 seconds for the performance to stabilize on the retain dataset. Compared to *Retrain*, CLUE provides $28.7\times$ better RTE and has $5.77\times$ better RTE than BE and BS. The reason CLUE is faster than BS is that it does not need to

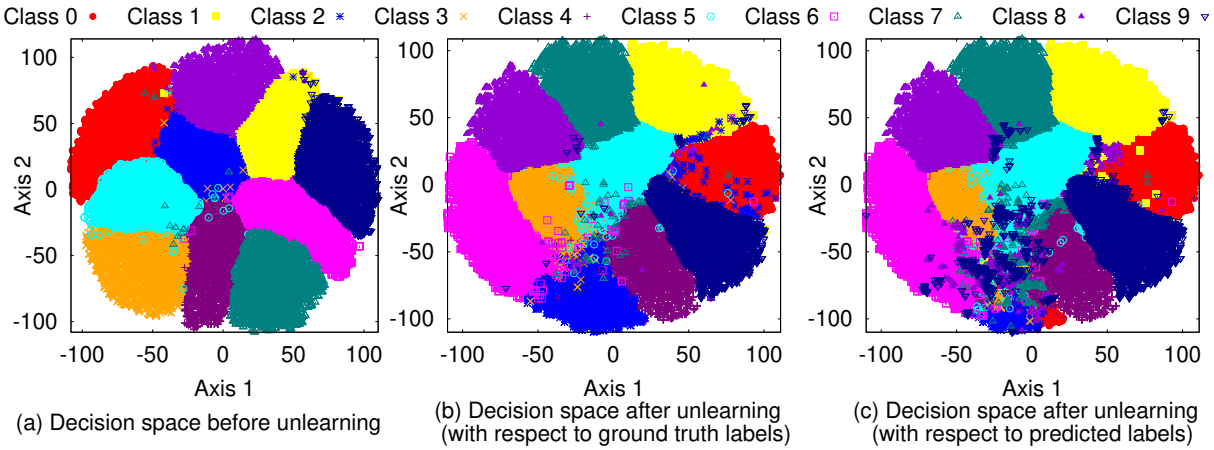


Figure 2. Visualization of the decision space of Resnet-20 (a) before unlearning class #2, (b) after unlearning class #2 with respect to the ground truth labels, and (c) after unlearning class #2 with respect to the predicted labels.

Table 3. Performance on Resnet-20 trained on CIFAR10. The values in the bracket represent the gap with *Retrain*.

Unlearning Methods	UA	RA	TA	MIA	Average Gap
Retrain	0	99.75	90.18	100	
GA	8.66 (8.66)	79.43 (20.32)	76.54 (13.64)	91.33 (8.77)	12.83
RL	20 (20)	89.50 (10.25)	82.22 (7.96)	78.50 (21.5)	14.92
IU	17.77 (17.77)	94.05 (5.70)	87.87 (2.31)	82.22 (17.78)	10.89
BE	24 (24)	91.63 (8.12)	83.33 (6.85)	76 (24)	15.74
BS	46.17 (46.17)	93.26 (6.49)	86.58 (3.7)	53.82 (46.18)	25.63
LU	**0 (0)**	11.11 (88.64)	11.11 (89.07)	**100 (0)**	69.42
UNSIR	**0 (0)**	14.37 (85.38)	14.23 (75.95)	**100 (0)**	40.33
CLUE (Ours)	10.48 (10.48)	**96.61 (3.14)**	**89.7 (0.48)**	89.50 (10.50)	**6.15**

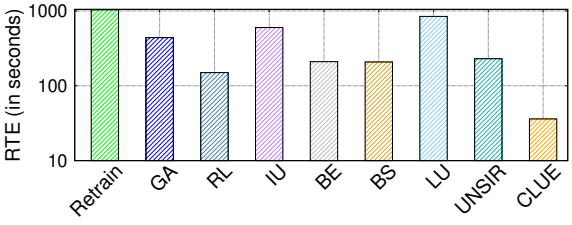


Figure 3. Run-time Efficiency. CLUE provides 28.7× lower RTE and has 5.77× better RTE than BE and BS. This result is obtained for ViT base architecture and CIFAR10 dataset.

search for the nearest class (necessitating the use of Fast Gradient Sign Method (FGSM) or Projected Gradient Descent (PGD)) for each sample, which is computationally expensive. Although CLUE performs inference twice for each input – once for the teacher and once for the student DNN

Table 4. Performance on Resnet-20 trained on CIFAR100. The values in the bracket represent the gap with *Retrain*.

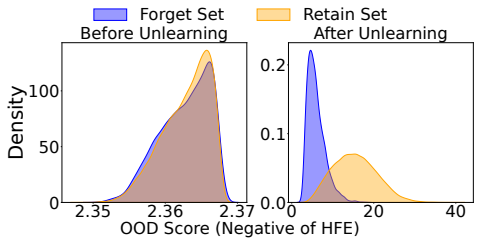
Unlearning Methods	UA	RA	TA	MIA	Average Gap
Retrain	0	87.84	62.85	100	
GA	**0 (0)**	50.18 (37.62)	43.24 (19.61)	**100 (0)**	14.31
RL	8.66 (8.66)	67.52 (20.32)	54.45 (8.4)	91.33 (8.77)	11.53
IU	**0 (0)**	62.16 (25.68)	51.50 (11.30)	100 (0)	9.24
BE	1.11 (1.11)	62.60 (25.24)	51.60 (11.20)	98.88 (1.12)	9.66
BS	2.44 (2.44)	62.95 (24.89)	51.55 (11.25)	96.66 (3.34)	10.48
LU	**0 (0)**	1.20 (86.64)	1.10 (61.75)	0 (100)	62.10
UNSIR	**0 (0)**	1.58 (86.26)	1.58 (61.27)	0 (100)	61.88
CLUE (Ours)	3.77 (3.77)	**68.11 (19.73)**	**54.75 (8.10)**	96.22 (3.78)	**8.84**

– those can be parallelized. In addition, BE modifies the DNN architecture by adding an additional node at the last layer, which requires retraining. Moreover, a large part of the low RTE of CLUE is due to its low number of iterations required for unlearning which is further illustrated in supplementary Section 4.

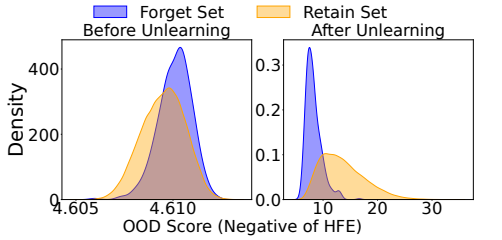
5.3. Understanding the Decision Space of CLUE

To understand the impact of CLUE on the decision space of the DNN, we visualize it in Fig. 2 for the training set before and after the unlearning process. These results are based on ResNet-20 trained on CIFAR-10, as larger datasets are impractical to visualize. The forget class is Class #2.

Fig. 2(a) depicts the decision space before performing the unlearning where well-clustered classes correspond to high model accuracy ($> 99\%$). Fig. 2(b) shows the de-



(a) Density of OOD score for CIFAR10 and Resnet20 Model



(b) Density of OOD score for CIFAR100 and Resnet20 Model

Figure 4. The distribution of HFE before and after unlearning with CLUE. The forget class is treated as OOD.

cision space after unlearning, plotted with respect to the ground-truth labels. This plot helps us understand the current decision boundary of the DNN. We notice that although rearranged, the classes in the retain dataset still form tight clusters which helps preserve accuracy. The rearrangement of the classes in the decision space is due to the stochastic nature of DNN training, which makes the DNN converge to a different local minimum during the unlearning process. Moreover, the area for the forget class in the decision space has shrunk. Ideally, this would vanish entirely but UA = 10.48 indicates retention of information of the forget class. However, the minimal overlap with the retain datasets suggests that performance on the retain set is preserved.

Fig. 2(c) shows that that the inputs from the forget class are being reassigned to different classes indicating the erasure of the information about the forget class which forces the DNN to rely on the remaining classes for inference, demonstrating successful unlearning. Figure 4 shows the distribution of HFE before and after the unlearning with CLUE, where we plot the OOD score proposed in [25], i.e., the negative of the HFE. As shown in Fig. 4a (left) and Fig. 4b (left), before unlearning, the samples of the forget dataset and the retain dataset have indistinguishable OOD scores. After unlearning with CLUE, we observe from Fig. 4a (right) and Fig. 4b (right) that the samples in the forget dataset have a lower OOD score (or higher HFE) than the retain dataset. *In summary, CLUE shrinks the decision boundary of the forget class and forces their samples to appear as OOD.*

5.4. Why Are ViTs better at Unlearning?

Our results show that ViTs outperform ResNets in unlearning performance, with a 4.92% and 5.47% gain on CIFAR10 and CIFAR100, respectively. We attribute this to self-attention mechanism of ViTs and larger capacity, which enable more disentangled class representations. This is supported by improved unlearning from ResNet20 to ViT, though performance drops by 1.74% as dataset complexity increases (Tabs. 1 and 2). On VGG-Face2 with ViT-Large, CLUE doubles baseline performance, but the gap with Retrain widens to 7.77% (Tab. 5).

Table 5. Performance on ViT-Large trained on VGG-Face-2. The values in the bracket represent the gap with *Retrain*.

Unlearning Methods	UA	RA	TA	MIA	Gap
Retrain	0	98.25	89.36	100	0
BE	9.50 (9.50)	80.4 (17.85)	72.35 (17.01)	90.30 (9.70)	13.51
BS	8.70 (8.70)	81.25 (17)	70.43 (18.93)	92.34 (7.66)	13.07
CLUE (Ours)	**3.20 (3.20)**	**85.20 (13.05)**	**76.30 (13.06)**	**98.20 (1.80)**	**7.77**

On the other hand, the attention mechanism in ViT helps it to focus on important parts of the image for inference. To show this point, Tab. 6 shows the attention maps of ViT-Base for three classes of CIFAR10. We notice that CLUE can effectively reshape the attention maps of the forget class #2 to achieve better unlearning performance, as the attention maps are almost unchanged before and after unlearning for class #0 and class #6. However, the attention map for class #2 shows decreased attention and thus cannot capture the structure of the bird. This shows that a successful unlearning approach for ViT selectively modifies the attention heads to lose focus on the forget dataset while retaining focus on the retain dataset. Tab. 6 also serves the purpose of illustrating the effect of CLUE on forget samples instead of forget class - successful unlearning makes it harder for the DNN to capture salient features of the forget samples and focus on generic features learned across different classes of the retain set. This is supported by the fact that in Tab. 6 ViT fails to detect the shape of the bird while focusing on partial features like eyes common to other animal classes.

5.5. CLUE vs Random Soft Labels

Since CLUE uses strong noise to simulate the logit distribution of OOD data, it is possible that the corrupted inputs' logit distribution may resemble that of randomly assigned soft labels. To test whether CLUE is merely equivalent to assigning random soft labels, we compare its performance against a random soft-labeling approach in Table 7. The results reveal a significant performance gap of up to 40%, demonstrating that simply using random soft labels is insufficient for effective unlearning. Intuitively, a significant difference lies in the fact that, unlike CLUE, random soft-labeling significantly degrades the performance on the retain set. This stems from the generation process of random soft labels. As these labels are essentially randomly gener-

ated logits passed through softmax activation, they likely do not resemble any categorical distribution that can be generated by the DNN. As the soft labels essentially encode information about all the classes, anomalous softlabels can be harmful for updating the DNN as evident from Tab. 7. This proves that the soft labels generated by the teacher in CLUE are fundamentally different from random soft labels.

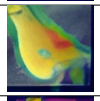
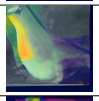
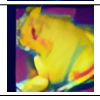
Class ID	Original Image	Attention Map Before Unlearning	Attention Map After Unlearning
Class:0 (Airplane)			
Class:2 (Bird)			
Class:6 (Frog)			

Table 6. Visualization of the attention maps of ViT-Base for three classes of CIFAR10. Class #2 (Bird) is the forget class.

Table 7. CLUE vs assigning random soft labels to forget samples. The values in the bracket represent the gap with *Retrain*.

Dataset/ Architecture	Unlearning Methods	UA	RA	TA	MIA	Average Gap
CIFAR10 ViT Base	Retrain	0	99.8	96.92	100	
	CLUE	2.04 (2.04)	98.42 (1.38)	95.86 (1.06)	97.95 (2.05)	1.63
	Random Softlabel	24.4 (24.4)	97.17 (2.63)	94.44 (2.48)	75.6 (24.4)	13.47
CIFAR100 ViT Base	Retrain	0	99.24	85.72	100	
	CLUE	0.22 (0.22)	91.24 (8)	80.77 (4.95)	99.77 (26.33)	3.37
	Random Softlabel	0 (0)	51.41 (47.83)	47.48 (38.24)	0 (0)	43.04
CIFAR10 Resnet20	Retrain	0	99.75	90.18	100	
	CLUE	10.48 (10.48)	96.61 (3.14)	89.7 (0.48)	89.50 (10.50)	6.15
	Random Softlabel	18.48 (18.48)	76.41 (23.34)	70.54 (19.64)	79.50 (21.50)	20.74
CIFAR100 Resnet20	Retrain	0	87.84	62.85	100	
	CLUE	3.77 (3.77)	68.11 (19.73)	54.75 (8.10)	96.22 (3.78)	8.84
	Random Softlabel	0 (0)	45.71 (42.13)	41.49 (21.36)	100 (0)	15.87

5.6. CLUE Ablation Study

We tested CLUE with (i) KD loss and gradient masking; (ii) energy loss and gradient masking; (iii) KD loss, energy loss, and gradient masking; (iv) KD and energy loss without gradient masking. We ran the ablation study for CIFAR10 dataset and Resnet-20 architecture. The results are shown in Tab. 8. We notice that removing gradient masking has the most significant impact on performance, with the average gap with *Retrain* increasing by 14.41%. Next, removing the KD loss incurs an average gap increase of almost 4%, highlighting its role in learning the posterior distribution of OOD data. Without KD loss, the DNN optimizes energy loss, sacrificing retain-set accuracy for unlearning. Finally, the smallest gap was observed without energy loss, although including it improved performance by over 1%.

Table 8. Ablation study for CLUE. The values in the bracket represent the gap with *Retrain*.

Approach	UA	RA	TA	MIA	Avg. Gap
Retrain	0	99.75	90.18	100	
KD Loss +Gradient Masking	13.04 (13.04)	95.58 (4.17)	88.71 (1.47)	87.95 (12.05)	7.6825
Energy Loss +Gradient Masking	16.71 (16.71)	94.64 (5.11)	88.41 (1.77)	83.28 (16.72)	10.0775
KD Loss +Energy Loss +Gradient Masking	10.48 (10.48)	96.61 (3.14)	89.7 (0.48)	89.51 (10.49)	6.1475
KD Loss +Energy Loss	0 (0)	56.4 (43.34)	51.3 (38.88)	100 (0)	20.55

6. Implementation on Mobile Device

We have implemented CLUE on a Raspberry-Pi 5, which was equipped with a quad-core ARM A76 System-on-Chip (SoC) with 2.4 GHz clock speed and 8 GB LPDDR4 memory [10]. We run CLUE and the closest baselines for the CIFAR100 dataset on both Resnet-20 and ViT-Base. Fig. 5(a) shows that CLUE outperforms the baseline approaches. Specifically, CLUE achieves 25× less energy and is 33× faster than *Retrain* and is 68% faster and consumes 90% less energy than BE. Apart from that we observe that for ViT base model and batch size of 32, CLUE uses 2598 MB peak memory while state-of-the-art approach BS takes 3729 MB which marks about 30% improvement in terms of memory consumption. For Resnet 20, memory consumption of CLUE is 69.5MB while BS requires 116.81 MB.

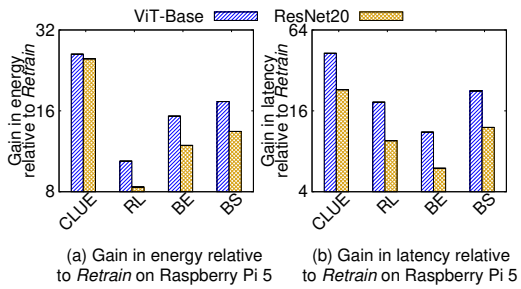


Figure 5. Latency and energy comparison.

7. Conclusions

We propose CLUE, a novel class-level unlearning method for mobile devices that modifies the DNN to treat forget class samples as OOD, without requiring access to the retain dataset. Experiments across multiple architectures and a real-world edge device show that CLUE improves power consumption, latency, and unlearning performance over existing baselines. Future work will explore data-free unlearning and extend CLUE to tasks such as object detection and semantic segmentation.

Acknowledgements

This work has been supported by the National Science Foundation under grants CNS-2312875 and OAC-2530896; by the Air Force Office of Scientific Research under grant FA9550-23-1-0261; by the Office of Naval Research under grant N00014-23-1-2221; and by the Defense Advanced Research Projects Agency under Cooperative Agreement D25AC00374-00.

References

[1] Children's online privacy protection act of 1998. Pub. L. No. 105–277, div. C, title XIII, 112 Stat. 2681–728, codified at 15 U.S.C. §§ 6501–6506, 1998. Enacted October 21, 1998. 1

[2] Immanuel Amirtharaj, Tai Groot, and Behnam Dezfouli. Profiling and improving the duty-cycling performance of Linux-based IoT devices. *Journal of Ambient Intelligence and Humanized Computing*, 11(5):1967–1995, 2020. 2

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

[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFace2: A Dataset for Recognising Faces Across Pose and Age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018. 4

[5] W. Chang, T. Zhu, H. Xu, W. Liu, and W. Zhou. Class machine unlearning for complex data via concepts inference and data poisoning. *arXiv preprint arXiv:2405.15662*, 2024. 1

[6] Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary Unlearning: Rapid Forgetting of Deep Networks via Shifting the Decision Boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7766–7775, June 2023. 1, 4

[7] Dasol Choi and Dongbin Na. Towards Machine Unlearning Benchmarks: Forgetting the Personal Identities in Facial Recognition Systems. *arXiv preprint arXiv:2311.02240*, 2023. 2

[8] 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. *ArXiv*, abs/2010.11929, 2020. 4

[9] Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. SalUn: Empowering Machine Unlearning via Gradient-based Weight Saliency in Both Image Classification and Generation. In *The Twelfth International Conference on Learning Representations*, 2024. 4

[10] Mohapmed Fezari and Ali Al-Dahoud. Raspberry Pi 5: The new Raspberry Pi family with more computation power and AI integration, 2023. 8

[11] Jack Foster, Kyle Fogarty, Stefan Schoepf, Cengiz Öztireli, and Alexandra Brintrup. Zero-shot Machine Unlearning at Scale via Lipschitz Regularization. *arXiv preprint arXiv:2402.01401*, 2024. 4, 5

[12] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast Machine Unlearning without Retraining through Selective Synaptic Dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12043–12051, 2024. 1

[13] 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, 4

[14] Tao Guo, Song Guo, Jiewei Zhang, Wenchao Xu, and Junxiao Wang. Efficient Attribute Unlearning: Towards Selective Removal of Input Attributes from Feature Representations. *arXiv preprint arXiv:2202.13295*, 2022. 1

[15] 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 (CVPR)*, June 2016. 4

[16] hearfool. VGGFace2 Dataset. https://www.kaggle.com/datasets/hearfool/vggface2/data, 2023. Accessed: 2024-10-05. 4

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *Deep Learning and Representation Learning Workshop: Neural Information Pricessing Systems 2014*, 2014. 2

[18] 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

[19] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Y. Zou. Approximate Data Deletion from Machine Learning Models: Algorithms and Evaluations. *ArXiv*, abs/2002.10077, 2020. 4

[20] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model Sparsity Can Simplify Machine Unlearning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 4, 5

[21] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. 4

[22] Yann LeCun, Sumit Chopra, Raia Hadsell, Aurelio Ranzato, and Fu Jie Huang. A Tutorial on Energy-Based Learning. 2006. 3

[23] California State Legislature. California Consumer Privacy Act of 2018 (CCPA). https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375, 2018. Accessed: 2024-10-12. 1

[24] Bo Liu, Qiang Liu, and Peter Stone. Continual Learning and Private Unlearning. *arXiv preprint arXiv:2203.12817*, 2022. 1

[25] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based Out-of-distribution Detection. In *Thirty-fourth Conference on Neural Information Processing Systems*, pages 21464–21475, 2020. 2, 7

[26] Liu, Yang and Fan, Mingyuan and Chen, Cen and Liu, Ximeng and Ma, Zhuo and Wang, Li and Ma, Jianfeng. Backdoor Defense with Machine Unlearning. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, page 280–289. IEEE Press, 2022. 1

[27] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *International Conference on Learning Representations*, 2017. 4

[28] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A Survey of Machine Unlearning. *arXiv preprint arXiv:2209.02299*, 2022. 1

[29] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy Risks of Securing Machine Learning Models against Adversarial Examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 241–257, New York, NY, USA, 2019. Association for Computing Machinery. 5

[30] Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast Yet Effective Machine Unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):13046–13055, 2024. 1, 4

[31] Thomson Reuters. Identity theft is being fueled by AI & cyber-attacks. https://www.thomsonreuters.com/en-us/posts/government/identity-theft-drivers/, May 2024. 1

[32] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding Factors Influencing Machine Unlearning . In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319, 2022. 4

[33] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016. Accessed: 2024-10-12. 1

[34] Johannes von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. In *Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2

[35] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine Unlearning of Features and Labels. *arXiv preprint arXiv:2108.11577*, 2021. 1

[36] Shaokui Wei, Mingda Zhang, Hongyuan Zha, and Baoyuan Wu. Shared Adversarial Unlearning: Backdoor Mitigation by Unlearning Shared Adversarial Examples. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 1

[37] Herbert Woisetschläger, Alexander Erben, Bill Marino, Shiqiang Wang, Nicholas D Lane, Ruben Mayer, and Hans-Arno Jacobsen. Federated Learning Priorities Under the European Union Artificial Intelligence Act. *arXiv preprint arXiv:2402.05968*, 2024. 1

[38] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting . In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, Los Alamitos, CA, USA, July 2018. IEEE Computer Society. 5

[39] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 253–261, 2020. 1