A Supplementary Material

A.1 Related works

Machine Unlearning. Machine unlearning requires removing information of forgetting data in the original model while preserving the knowledge contained in the remaining data (Bourtoule et al., 2021; Xu et al., 2024). Currently works on machine unlearning can be summarized into two branches based on the unlearning objectives. The first type is *exact unlearning*, which requires achieving the same model as the train-from-scratch model on remaining data. Exact unlearning is mainly applied to classical machine learning models (Bourtoule et al., 2021; Kim & Woo, 2022). In deep models, the parameters and model structures can be much more complex. Therefore, the exact unlearning is hard to be realized. The current works of exact unlearning on deep models usually take the retraining strategy and focus on improving the algorithm efficiency, for example, the SISA algorithm which retrains the model via a distributed approach on different devices (Bourtoule et al., 2021). The second type is *approximate unlearning*, which requires the unlearned model to get similar performances to the retrained model on both the remaining data and the forgetting data. Approximate unlearning methods are widely applied to deep models (Nguyen et al., 2020; Tarun et al., 2023; Golatkar et al., 2020b; Thudi et al., 2022; Graves et al., 2021; Chen et al., 2023; Kurmanji et al., 2023; Chundawat et al., 2023; Golatkar et al., 2020a; Liu et al., 2021). The cutting-edged works on approximate unlearning includes: (Chundawat et al., 2023) which employs two teacher models that are trained on the remaining and forgetting data to guide the unlearning; (Chen et al., 2023) which explores a new perspective of unlearning by shifting the decision boundary of different classes for unlearning, and (Thudi et al., 2022) which recovers the changes of parameters occurring in the training of data to be forgotten. Compared with the exact unlearning on deep models, approximate unlearning has wider applications.

Unsupervised Representation Learning. Unsupervised representation learning aims to learn the representations of the input data without using labels. The unsupervised representation learning has attracted more attention from researchers. The variational autoencoder (VAE) (Kingma & Welling, 2014) and contrastive learning (van den Oord et al., 2018) are two critical techniques. The VAE can project the input features into low-dimensional Gaussian representations. Currently, the strong ability for representation learning makes the VAE have wide applications on deep learning tasks. For instance, (Liang et al., 2018) explores the application of VAE on representation learning in collaborative filtering while (Kipf & Welling, 2016) applies the VAE on the representation learning of graph data. Contrastive learning reduces the distances of the embeddings of data that share similar characteristics and increases the distances of the embeddings of data that are dissimilar from each other. For instance, (van den Oord et al., 2018) introduces the Noise Contrastive Estimation to differentiate the distance between the similar and dissimilar samples and (Chen et al., 2020) employs cosine similarity during contrastive learning.

A.2 Notations

We provide a table of all notations of the main paper in Table 1.

A.3 Implementation Details

A.3.1 Overall Workflow

Figure. 1 presents the workflow of the whole LAF framework. The LAF first trained two VAEs h and h_f on the representations of training data $\mathbf{X}$ and representations of forgetting data $\mathbf{X}_f$. Then by fixing the parameters of h and h_f, Next, to align the representation distribution of g_U^e with the classifier, LAF compares the similarities between the representations of remaining data and forgetting data in the model before and after unlearning and maximizes the representation alignment loss L_{RA}. L_{UE} and L_{RA} can be updated alternately. We output the updated model as the final model g_U^e.

Subsequently, the LAF framework focuses on aligning the representation distributions between the post-unlearning extractor g_U^e and the classifier g_D^c. This is achieved by the representation alignment loss L_{RA}, aligning the representations of the remaining data before and after the unlearning process

Table 1: Table of Notations Used in The Main Paper

Notation	Explanation
D	Training data
$\mathcal{P}$	Training data distribution
D_r	Remaining data
$\mathcal{P}_r$	Training data distribution
D_f	Forgetting data
$\mathcal{P}_f$	Training data distribution
x	Instance of data
$\mathcal{X}$	Instance space
y	Label of data
$\mathcal{Y}$	Label space
g_D	Trained deep model
g_D^e	Extractor of the trained deep model
g_D^c	Classifier of the trained deep model
g_U	Post-unlearning deep model
g_U^e	Extractor of the post-unlearning deep model
$Q(D_r)$	Distribution that post-unlearning deep model follows on D_r
$Q(D_f)$	Distribution that post-unlearning deep model follows on D_f
$\Delta(\cdot,\cdot)$	Distribution discrepancy
h	VAE that learns the distribution of the training data representations
h_f	VAE that learns the distribution of the forgetting data representations
$\mathcal{N}(0,\mathcal{I})$	Standard Gaussian distribution
μ_h, σ_h	Mean and std estimated by h on its encoding layer for $g_D^e(x), x \in D_r$
$\tilde{\mu}_h, \tilde{\sigma}_h$	Mean and std estimated by h on its encoding layer for $g_U^e(x), x \in D_r$
μ_{h_f}, σ_{h_f}	Mean and std estimated by h_f on its encoding layer for $g_D^e(x), x \in D_f$
$\tilde{\mu}_{h_f}, \tilde{\sigma}_{h_f}$	Mean and std estimated by h_f on its encoding layer for $g_U^e(x), x \in D_f$

and differentiating the representations of the forgetting data before and after the unlearning. The L_{UE} and L_{RA} losses are updated in an alternating fashion.

The culmination of this process is the final updated model, denoted as g_U^e, which effectively embodies the refined balance between learning and forgetting, as dictated by the LAF framework.

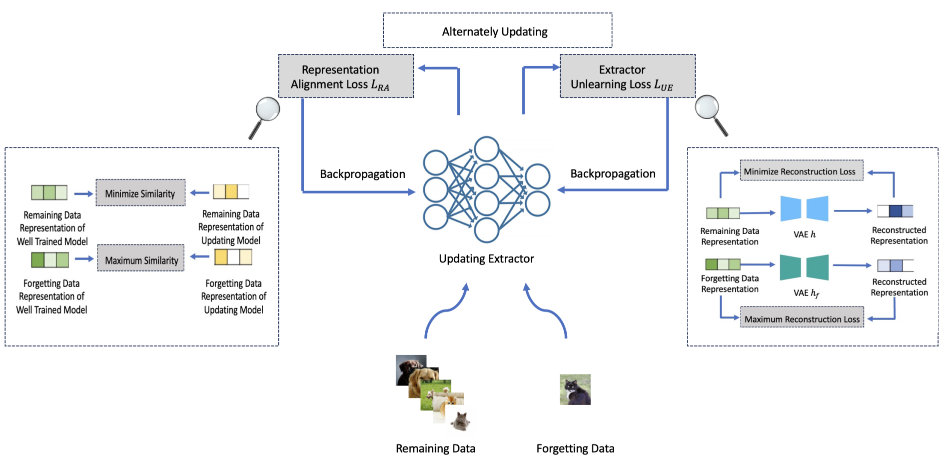


Figure 1: Workflow for LAF consisting of VAE training, extractor unlearning and representation alignment stages.

A.3.2 ENVIRONMENT

All the experiments are conducted on one server with NVIDIA RTX A6000 GPU (48GB GDDR6 Memory) and 12th Gen Intel(R) Core(TM) i9-12900K (16 cores and 128GB Memory) and two servers with NVIDIA RTX A5000 GPUs (24GB GDDR6 Memory) and 12th Gen Intel Core i7-12700K CPUs (12 cores and 128GB Memory). The code of LAF was implemented in Python 3.9.16 and

Cuda 11.6.1. The main Python packages' versions are the following: Numpy 1.23.5; Pandas 2.0.1; Pytorch 1.13.1; Torchvision 0.14.1. The datasets in experiments: **DIGITS** (LeCun, 1998), **FASHION** (Xiao et al., 2017), **CIFAR10** (Krizhevsky et al., 2009), and **SVHN** dataset (Netzer et al., 2011) are all downloaded from the Torchvision library. Moreover, all the comparison methods provide open resources for their implementation code: **Boundary** [1], **T-S** [2], **SCRUB** [3], **SISA** [4], **Unrolling** [5].

A.3.3 INITIALIZATIONS

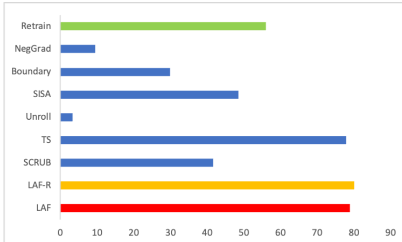
For the experiment models, we choose the **CNN**(LeCun et al., 1995) with two convolutional layers for the two MNIST datasets. The output channels for the two convolutional layers are 16 and 32 respectively. Then the other parts of the CNN consist of three linear layers with the output dimensions 256, 128 and 10. For the two CIFAR datasets, we choose an **18-layer ResNet** (He et al., 2016) with two linear layers with the output dimensions 256, and 10 and the **ResNet** does not contain the pre-trained weights. We construct two VAEs with three and four linear layers in the encoders and decoders. The first type of VAE is used for the two MNIST datasets consisting of three linear layers' encoder with the input dimensions 256, 128, and 32 and a three linear layers' decoder with the input dimensions 8, 32, and 128. The second type of VAE is used for the other two datasets consisting of the same structure encoder as the first one and a three linear layers' decoder with the input dimensions 16, 32, and 128.

All the experiments are based on the original models trained in the four datasets. We train two CNN models on two MNIST datasets for 10 epochs with a learning rate of 1e-3 while we train another two 18-layer ResNet models on two CIFAR datasets for 20 epochs with a learning rate of 5e-5. For the golden standard baselines **Retrain**, we retrain the CNN models on two MNIST datasets for 20 epochs with a learning rate of 1e-3. We retrain the 18-layer ResNet models on two CIFAR datasets for 40 epochs with a learning rate of 5e-5. Then for the other six comparison baselines:**NegGrad**, **Boundary**(Chen et al., 2023), **T-S**(Chundawat et al., 2023), **SCRUB**(Kurmanji et al., 2023), **SISA**(Bourtoule et al., 2021), **Unroll**(Thudi et al., 2022), we keep the hyperparameters of the unlearning process the same as in the original paper and adjust other necessary parameters for the unlearning stage to get as high performances as we can. **NegGrad** adjusts the deep model parameters with positive gradients on remaining data and negative gradients on forgetting data; **Boundary** (Chen et al., 2023) shift the decision boundaries of the forgetting data and remaining data to eliminate the forgetting data information; **SISA** (Bourtoule et al., 2021) proposes to retrain the model using the small data shards from the remaining dataset and ensemble the final results; **Unroll** (Thudi et al., 2022) records gradients when learning the first epoch and adds recorded gradients on weights after the incremental training; **T-S** (Chundawat et al., 2023) proposes to retrain two teacher models on forgetting data and remaining data and adjust the student model through the differences between the output space of the two teacher models; **SCRUB** (Kurmanji et al., 2023) force the model to be consistent with the teacher model trained on remaining data and inconsistent with another teacher model trained on forgetting data.

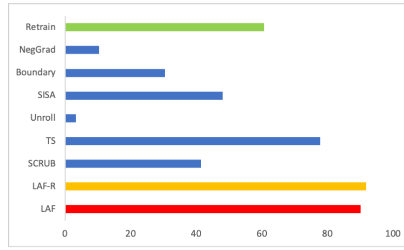
A.3.4 HYPERPARAMETERS

In all experiments, we configure the batch size to 32. During the training of VAEs, we assign the latent dimensions as 8 for the DIGITS and FASHION datasets and 16 for the CIFAR10 and SVHN datasets. The learning rate for VAE training is established at 1e-3, with the number of training epochs set to 10. For representation alignment, we assign the value of τ as 2, 20, and 20 for data removal, class removal, and noisy label removal tasks, respectively for CNN. We assign the value of τ as 20, 20, and 5 for ResNet. Subsequently, in the supervised repairing stage, we designate the repairing epoch as 1, applying a learning rate of 1e-3 for all tasks on the DIGITS and FASHION datasets, and 5e-5 on the CIFAR10 and SVHN datasets.

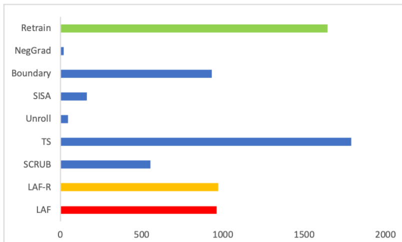
[1] https://www.dropbox.com/s/bwu543qsdy4s32i/Boundary-Unlearning-Code.zip?dl=0
[2] https://github.com/vikram2000b/bad-teaching-unlearning
[3] https://github.com/meghdadk/SCRUB
[4] https://github.com/cleverhans-lab/machine-unlearning
[5] https://github.com/cleverhans-lab/unrolling-sgd



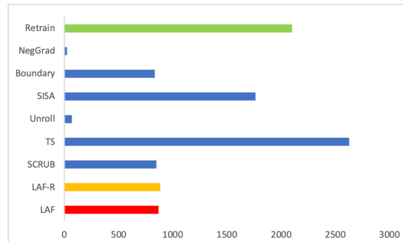
(a) Time cost on DIGITS



(b) Time cost on FASHION



(c) Time cost on CIFAR10



(d) Time cost on SVHN

Figure 3: Time cost comparison in the data removal task. The red columns stand for the time costs of the proposed LAF and the orange columns stand for LAF-R. The green columns denote the retraining and the blue columns denote other methods.

A.4 Efficiency analysis

A.4.1 Time cost analysis

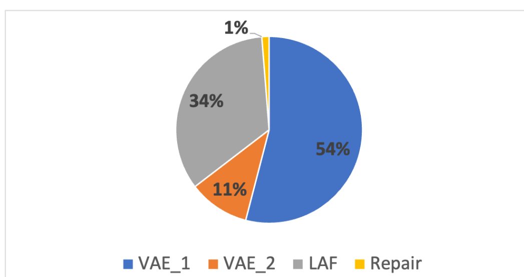
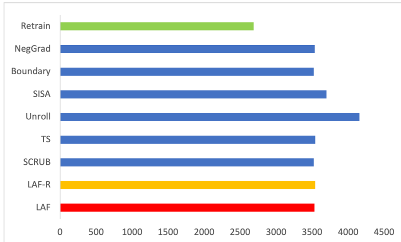


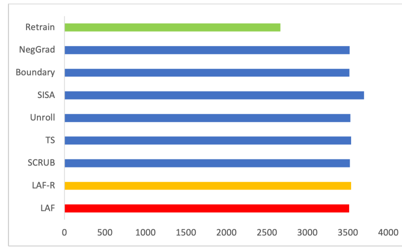
Figure 2: Time cost proportion. VAE_1 stands for the training of h and VAE_2 stands for the training of h_f

Figure 3 presents a comparative analysis of the time efficiency of our LAF framework against other methods in data removal tasks. The results indicate that LAF does not hold a distinct advantage in terms of efficiency. Specifically, in experiments conducted on two MNIST datasets, LAF exhibits a slightly higher time cost compared to the seven other evaluated methods. However, in trials involving the CIFAR10 and SVHN datasets, LAF's time consumption is close to the average time cost of other methods and is notably less than that required for retraining and the TS (Teacher-Student) approaches.

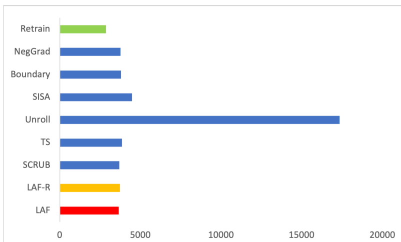
This variation in time efficiency primarily stems from the time-intensive process of training the VAEs. As illustrated in Figure 2, the training phase of VAE h accounts for nearly half of the total algorithm runtime, pinpointing a key area for future enhancements. It's important to note, though, that the training of h is conducted on the entire training dataset and is independent of the selection of data to be forgotten. Hence, this training phase can be executed separately from the unlearning process, offering a substantial opportunity to reduce overall time expenditure.



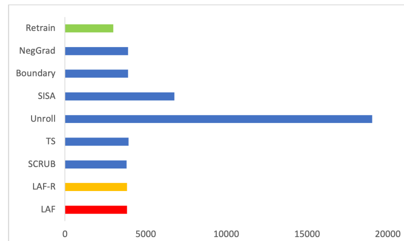
(a) Storage cost on DIGITS



(b) Storage cost on FASHION



(c) Storage cost on CIFAR10



(d) Storage cost on SVHN

Figure 4: Storage workload comparison in the data removal task. The red columns stand for the time costs of the proposed LAF and the orange columns stand for LAF-R. The green columns denote the retraining and the blue columns denote other methods.

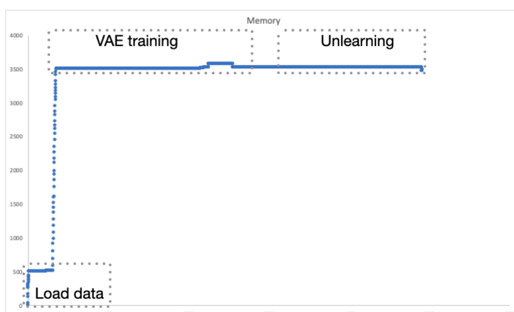


Figure 5: Memory workload changes of LAF during the whole procedure on the random data removal task on DIGITS

A.4.2 Storage workload analysis

Figure 4 provides a comparative overview of the storage workload associated with our LAF framework and other data unlearning methods. The analysis indicates that LAF's storage demands are broadly comparable to those of most other unlearning methods. Notably, the Retrain method exhibits the lowest storage workload, as it does not necessitate any additional memory-intensive components. Conversely, while the Unroll method achieves the lowest time cost, it demands the most storage, particularly in experiments involving ResNet. This increased requirement is due to Unroll's need to store gradients for all parameters across the entire training dataset. Moreover, the SISA approach involves training multiple models concurrently, each mirroring the structure of the original model, thereby escalating the storage requirements. In contrast, our LAF framework avoids the need to store extensive gradients or maintain complex additional models. Although LAF includes the training of two additional VAEs, these are structurally simple, comprising merely five or four linear layers each. For context, the CNN model encompasses 450K parameters, and ResNet-18 contains 11.3M parameters, while the two VAEs collectively have only 150K parameters.

To provide a clearer depiction of the storage workload dynamics within LAF, Figure 5 visualizes the changes in storage requirements throughout the entire LAF process. It reveals that the peak workload occurs during the VAE training stage, after which the storage demands stabilize during the actual unlearning phase.

A.5 Additional Experiments

In this section, we add three parts of additional experiments. A.5.1 is to evaluate the impact of the two approximations in the extractor unlearning process: replacing D by D_r for the training of the first VAE, and dropping of the two KL divergence terms in Eq.8. A.5.2 is to evaluate two different optimization strategies, alternately updating and two-stage updating. A.5.3 is set up to examine the efficacy of the proposed methods on the low-quality representations.

Table 2: Ablation study results in data removal. 'Add KL' adds two KL divergence terms in Eq.8 in the main paper for optimization and D_r denotes training the VAE h using the remaining data. The bold results stand for the best. The following tables take the same notations.

Method	**Data**	**Train$_r$**	**Train$_f$**	**Test**	**ASR**	**Data**	**Train$_r$**	**Train$_f$**	**Test**	**ASR**
Retrain	DIGITS	99.56±0.05	98.84±0.10	99.04±0.10	49.80±0.53	FASHION	96.43±0.35	92.15±0.41	90.23±0.22	47.32±0.76
Add KL		53.36±3.54	85.78±5.14	58.90±1.40	41.19±0.32		59.97±0.06	11.93±2.94	48.93±0.23	41.57±0.06
D_r		**99.52±0.01**	**99.43±0.30**	**98.98±0.09**	56.67±2.61		**92.49±0.37**	90.17±1.57	**88.22±0.42**	44.57±0.87
LAF		98.03±0.68	97.29±1.43	97.30±0.78	**47.92±0.84**		91.54±2.67	**90.91±7.00**	87.53±3.26	**46.89±0.88**
Retrain	CIFAR10	84.03±0.20	78.05±1.34	87.20±0.65	57.48±0	SVHN	83.88±0.23	75.16±0.76	93.41±0.40	58.76±0.48
Add KL		44.88±32.38	40.81±39.45	46.33±36.33	57.30±5.20		**81.92±0.30**	75.79±0.34	91.93±0.32	58.09±0.29
D_r		77.70±0.67	**75.59±1.81**	81.79±0.84	55.73±0.73		81.77±0.36	**75.37±0.82**	91.88±0.13	**58.19±0.12**
LAF		**78.03±1.55**	73.30±3.96	**82.22±2.57**	**57.65±0.70**		81.63±0.49	76.11±1.49	**92.32±0.58**	57.85±0.89

Table 3: Ablation study results in class removal.

Method	**Data**	**Test$_r$**	**Test$_f$**	**ASR**	**Data**	**Test$_r$**	**Test$_f$**	**ASR**
Retrain	DIGITS	98.81±0.15	0±0	26.49±1.41	FASHION	92.66±0.29	0±0	38.24±3.13
Add KL		98.16±0.18	**0.26±0.05**	24.83±0.76		88.43±1.24	**0.75±0.44**	**31.71±0.74**
D_r		**98.18±0.17**	0.31±0.10	**24.74±0.80**		89.78±0.39	2.35±1.04	31.45±0.19
LAF		98.03±0.68	0.26±0.11	52.25±2.61		**91.54±2.67**	2.46±1.46	31.35±0.71
Retrain	CIFAR10	86.01±0.64	0±0	67.76±1.58	SVHN	94.07±0.67	0±0	59.33±1.31
Add KL		47.11±36.02	**0.10±0.05**	48.45±1.67		91.14±0.76	2.38±2.32	54.33±2.47
D_r		76.83±0.38	2.05±1.65	47.14±1.48		90.76±0.14	6.31±3.89	47.19±3.63
LAF		**82.38±0.97**	2.15±1.96	50.46±1.96		85.80±1.14	**0.33±0.51**	**56.33±0.49**

A.5.1 Further Ablation Study

Tables 2, 3, and 4 present the findings from our expanded ablation study, focusing on various unlearning tasks. The results highlight that LAF, both in its standard form and with D_r utilized during VAE training, achieves comparable outcomes across most unlearning scenarios. This is particularly evident in tasks involving random data removal. Such consistency validates our approach of substituting D with D_r, which offers the advantage of pre-training the VAE, thereby reducing time costs associated with unlearning requests.

Furthermore, upon integrating two KL divergence terms into the optimization process, we observe that performance in class removal and noisy label removal tasks remains similar to both the standard LAF and the LAF with D_r in VAE training. However, a notable difference emerges in random data removal tasks, where we witness a marked decline in performance for the remaining data and test data, along with a greater deviation in attack success rates compared to retrained models. This phenomenon can be attributed to the KL divergence term of the VAE, which, when trained on the entire dataset, acts as a regularization component. This effect makes unlearning more challenging, inadvertently preserving information about the remaining data. It is this observation that led us to exclude these two KL divergence terms from the final extractor unlearning loss formulation.

Table 4: Ablation study results in noisy label removal.

Method	Data	Train_r	Train_f	Test	ASR	Data	Train_r	Train_f	Test	ASR
Retrain	DIGITS	99.75±0.12	0.17±0.01	98.83±0.05	39.26±0.01	FASHION	97.04±0.83	2.16±0.06	88.15±0.45	37.65±1.88
Add KL		90.15±1.12	3.66±0.17	84.40±1.74	**29.31±0.75**		87.82±0.47	**4.68±0.22**	78.33±0.75	30.19±0.58
D_r		90.60±0.59	3.53±0.03	84.84±1.18	28.85±0.66		87.74±0.44	4.70±0.19	78.27±0.81	**30.35±0.22**
LAF		**96.46±0.67**	**2.70±0.59**	**91.48±1.49**	18.51±0.57		**92.32±0.66**	4.80±0.71	**81.21±1.22**	22.36±0.72
Retrain	CIFAR10	73.33±0.89	7.74±0.23	64.74±1.26	57.04±0.99	SVHN	82.46±0.15	2.37±0.23	93.38±0.35	59.55±1.22
Add KL		**77.67±0.90**	**2.80±0.35**	82.48±0.66	**51.82±4.76**		**78.06±2.71**	**3.49±6.43**	89.11±0.48	49.58±0.50
D_r		78.31±1.20	2.80±0.31	**82.65±0.56**	47.18±1.14		78.02±0.08	3.53±0.03	89.15±0.56	50.71±1.16
LAF		57.44±1.11	10.60±0.20	47.57±0.63	53.18±0.68		77.87±0.35	3.59±0.20	**89.33±0.32**	**51.50±1.17**

Table 5: Optimizing strategy comparison in data removal.

Method	Data	Train_r	Train_f	Test	ASR	Data	Train_r	Train_f	Test	ASR
Retrain	DIGITS	99.56±0.05	98.84±0.10	99.04±0.10	49.80±0.53	FASH	96.43±0.35	92.15±0.41	90.23±0.22	47.32±0.76
Two Stage		88.63±7.06	69.22±19.74	84.22±9.45	44.01±1.29		81.82±0.16	71.26±1.37	**91.28±0.30**	56.92±0.96
LAF		**98.03±0.68**	**97.29±1.43**	**97.30±0.78**	**47.92±0.84**		**91.54±2.67**	**90.91±7.00**	87.53±3.26	**46.89±0.88**
Retrain	CIFAR10	84.03±0.20	78.05±1.34	87.20±0.65	57.48±0	SVHN	83.88±0.23	75.16±0.76	93.41±0.40	58.76±0.48
Two Stage		**78.62±0.79**	**80.05±1.11**	**83.51±0.5**	56.46±0.30		**81.82±0.16**	71.26±1.37	91.28±0.30	56.92±0.96
LAF		78.03±1.55	73.30±3.96	82.22±2.57	**57.65±0.70**		81.63±0.49	**76.11±1.49**	**92.32±0.58**	**57.85±0.89**

Table 6: Optimizing strategy comparison in class removal.

Method	Data	Test_r	Test_f	ASR	Data	Test_r	Test_f	ASR
Retrain	DIGITS	98.81±0.15	0±0	26.49±1.41	FASH	92.66±0.29	0±0	38.24±3.13
Two Stage		**98.84±0.13**	1.02±0.31	**23.59±0.28**		91.17±0.17	9.05±0.55	30.58±0.09
LAF		98.03±0.68	**0.26±0.11**	52.25±2.61		**91.54±2.67**	**2.46±1.46**	**31.35±0.71**
Retrain	CIFAR10	86.01±0.64	0±0	67.76±1.58	SVHN	94.07±0.67	0±0	59.33±1.31
Two Stage		82.27±1.06	**1.15±0.55**	46.20±0.72		**91.95±0.11**	2.67±1.98	54.78±1.13
LAF		**82.38±0.97**	2.15±1.96	**50.46±1.96**		85.80±1.14	**0.33±0.51**	**56.33±0.49**

Table 7: Optimizing strategy comparison in noisy label removal.

Method	Data	Train_r	Train_f	Test	ASR	Data	Train_r	Train_f	Test	ASR
Retrain	DIGITS	99.75±0.12	0.17±0.01	98.83±0.05	39.26±0.01	FASH	97.04±0.83	2.16±0.06	88.15±0.45	37.65±1.88
Two Stage		90.42±0.15	3.79±0.16	84.12±0.30	**58.54±0.01**		85.52±1.03	5.94±0.35	73.86±1.63	**30.08±0.53**
LAF		**96.46±0.67**	**2.70±0.59**	**91.48±1.49**	18.51±0.57		**92.32±0.66**	**4.80±0.71**	**81.21±1.22**	22.36±0.72
Retrain	CIFAR10	73.33±0.89	7.74±0.23	64.74±1.26	**57.04±0.99**	SVHN	82.46±0.15	2.37±0.23	93.38±0.35	59.55±1.22
Two Stage		**80.04±0.33**	**2.67±0.17**	83.93±0.70	57.31±0.22		15.70±0.75	12.41±0.18	9.65±0.50	**55.84±0.69**
LAF		57.44±1.11	10.60±0.20	**47.57±0.63**	53.18±0.68		**77.87±0.35**	**3.59±0.20**	**89.33±0.32**	51.50±1.17

A.5.2 Optimizing Strategy

Table 5, 6, 7 presents the results using two different optimizing strategies, alternately updating and two-stage updating. On the DIGITS, FASHION, and SVHN datasets, the alternately updating can reach better forgetting performances and knowledge preservation performances for all three unlearning tasks. In addition, although the two-stage updating can achieve closer results to the retrained models on the preservation of the knowledge from the remaining data, the performances on the forgetting data and the ASR show large differences to the results of alternately updating. Therefore, the experiment results can demonstrate the reasonability and correctness of alternately updating instead of updating in two stages.

A.5.3 Experiment On low-quality representations

To further examine the efficacy of the proposed LAF, we test LAF with low-quality representations on the different unlearning tasks. Considering that deep models can easily to reach high prediction performances on the two MNIST datasets, we choose the other two datasets: CIFAR10 and SVHN and train two insufficiently trained ResNet-18 models for the experiments. We set the training epochs as 1 and keep the same values of the other hyperparameters as the experiment settings in the main paper. The results are presented in Table 8 and 9.

The sufficiently retrained model and sufficiently trained SISA always reach significantly better performances than all the post-unlearning models because the models provided for unlearning are insufficiently trained. Therefore, the retrained results do not have much reference value in this experiment setting. The results of the original model can prove that all the original models are sufficiently trained and can provide baselines of the performances on the remaining and forgetting data.

Then for the remaining approaches, the results demonstrate that LAF can LAF-R can achieve much better performances than other methods. This can support that LAF can also work on low-quality representation extractors.

Table 8: Comparison results with other state-of-the-art methods in data removal (avg%±std%).

Method	Data	Train_r	Train_f	Test	ASR	Data	Train_r	Train_f	Test	ASR
Retrain	CIFAR10	84.03±0.20	78.05±1.34	87.20±0.65	57.48±0	SVHN	83.88±0.23	75.16±0.76	93.41±0.40	58.76±0.48
Original		45.59±2.77	46.12±2.78	48.76±3.95	-		63.21±1.66	63.04±1.73	72.70±3.18	-
NegGrad		20.27±0.93	0±0	16.20±0.64	51.38±0.96		22.42±0.10	0±0	19.73±0.15	60.34±0.06
Boundary		21.32±1.34	10.40±0.37	19.62±2.03	54.72±0.81		42.09±1.31	12.66±0.19	47.21±2.78	55.53±1.73
SISA		66.78±0.10	53.12±0.74	54.30±0.05	37.53±0.02		82.48±0.17	67.79±0.34	82.57±0.83	50.19±0.38
Unrolling		27.02±0.16	2.28±2.23	29.72±0.16	57.25±0.87		49.74±1.16	14.78±4.44	53.43±1.35	56.34±0.13
T-S		46.48±1.87	50.20±4.59	50.61±3.14	52.98±0.52		64.52±2.20	55.16±2.13	73.13±4.59	55.02±0.25
SCRUB		30.00±0.12	0±0	26.84±0.84	53.86±0.55		30.23±0.17	0±0	27.82±1.04	60.30±0.05
LAF+R		48.11±1.36	44.19±1.00	52.32±0.50	53.43±0.34		68.31±0.55	54.77±5.10	78.75±0.96	55.70±0.57
LAF		43.55±0.75	44.51±0.21	46.06±0.87	54.95±0.66		63.89±1.20	53.94±2.81	72.30±3.05	54.23±1.41

Table 9: Comparison results with other state-of-the-art methods in class removal (avg%±std%)

Method	Data	Test_r	Test_f	ASR	Data	Test_r	Test_f	ASR
Retrain	CIFAR10	86.01±0.64	0±0	67.76±1.58	SVHN	94.07±0.67	0±0	59.33±1.31
Original		63.86±9.61	47.08±5.27	-		61.27±17.36	73.52±3.22	-
NegGrad		17.87±0.34	0±0	46.46±0.43		34.36±0.21	0±0	64.13±1.74
Boundary		35.24±9.22	1.52±2.89	49.03±1.64		54.36±0.71	12.13±1.81	61.99±1.20
SISA		99.10±0.03	0±0	50.12±0.23		92.14±0.07	0±0	50.00±0.02
Unrolling		42.35±0.67	0±0	58.55±0.01		60.78±2.68	0±0	56.85±2.59
T-S		48.81±3.05	32.30±10.20	45.86±2.26		72.10±3.09	25.83±14.94	57.77±8.49
SCRUB		31.45±1.56	0±0	51.57±0.29		22.53±1.54	0±0	68.76±2.94
LAF+R		47.04±0.16	0±0	47.34±2.35		76.34±0.10	0±0	56.81±0.65
LAF		43.07±4.63	1.3±0.20	43.29±0.34		61.51±4.63	0.06±0.06	56.67±2.61

References

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

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.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan S. Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *AAAI*, 2023.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *ECCV*, 2020a.

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

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

Junyaup Kim and Simon S. Woo. Efficient two-stage model retraining for machine unlearning. In *CVPR*, 2022.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *Arxiv*, 1611.07308, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/cifar.html*, 2009.

Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *Arxiv*, 2302.09880, 2023.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *WWW*, 2018.

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.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

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

Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan S. Kankanhalli. Fast yet effective machine unlearning. *TNNLS*, 2023.

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

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *Arxiv*, 1807.03748, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *Arxiv*, 1708.07747, 2017.

Heng Xu, Tianqing Zhu, Lefeng Zhang, Wanlei Zhou, and Philip S. Yu. Machine unlearning: A survey. *ACM Comput. Surv.*, 2024.