Towards Efficient Machine Unlearning with Data Augmentation: Guided Loss-Increasing (GLI) to Prevent the Catastrophic Model Utility Drop

Supplementary Material

1. Security Threat and The Need of Unlearning

Increasing awareness of individuals' personal information protection [16] is invoking the importance of deep learning security. Deep learning demonstrates impressive performance, especially in security-related fields such as identity verification in financial transactions. Despite the advanced performance and capabilities, deep neural networks still have vulnerabilities. They are susceptible to input data and tend to fail to predict even fine variations [13]. Previous research has introduced adversarial examples that are difficult to resolve because humans cannot perceive the fine variation [17]. Since the advent of adversarial examples, many researchers have become aware of issues with the robustness and security of models and have studied deep learning system attacks. The Fast Gradient Sign Method(FGSM) effectively generates adversarial examples and is utilized for training. These adversarial training methods can resolve adversarial example problems and strengthen the model [7, 11].

In addition to adversarial attacks [11] on deep learning models, previous work has studied a variety of attacks such as a backdoor attack [1] and a clean-label poisoning attack [15] and so on. A backdoor attack is an attack that generates a backdoor by using a backdoor key and creates input instances on the label. A backdoor attack focuses on creating a backdoor into the model using single instance keys and pattern keys, rather than reducing the performance of the model. A backdoor attack enables attacks by inserting a small amount of data without any knowledge of the model or training data [1]. A clean-label poisoning attack involves injecting misleading examples to confuse the model. This attack does not require control of the labeling function. Therefore, individuals with no prior research involvement can become attackers. Previous work shows that this attack can be successful without internal access to the data collection or labeling process [15].

In addition to these system attack issues, deep learning models have issues with training data. Therefore, deep learning can also result in intellectual property infringement on deep learning training data and illegal disclosure of personal data [3, 10, 14]. Therefore, machine unlearning is increasingly important for securing systems and protecting privacy. However, despite its importance, machine unlearning is still under-researched. We believe our research reflects this necessity.

2. Comparison of Distance Loss

In our Guided Loss Increasing (GLI) method, we increase the distance loss d, which is crucial for achieving a high forgetting score. We have extensively experimented with various distance algorithms to increase the feature differentiation between the augmented image and forget data x_forget. We have explored various loss functions including L_1 norm, L_2 norm, and cosine similarity.

2.1. Distance Loss

The L_1 norm is sensitive to outliers, emphasizing differences when the values of a particular component are significantly distinct. It measures distance by summing the absolute values of each component like the following equation.

$$\|L\|_1 = \sum_i |A - B_i| \tag{1}$$

In contrast, L_2 norm is less sensitive to outliers and can be used when considering all components as equally important. It measures distance by summing the squares of each component and then taking the square root according to the following equation.

$$\|L\|_2 = \sqrt{\sum_i (A - B)_i^2} \tag{2}$$

Cosine similarity is effective when the size of the vectors isn't important because it only measures directional similarity, not the size of the vectors. Cosine similarity demonstrates that the closer two vectors are to 0 degrees, the more similar they are. To adapt cosine similarity for representing vector dissimilarity, we utilize it according to the following equation.

$$-\cos(\theta) = -\frac{A \cdot B}{\|A\|\|B\|} \tag{3}$$

Where A and B are two vectors, and $\|A\|$ and $\|B\|$ representing the vector lengths or magnitudes of A and B, respectively. $A \cdot B$ is the dot product of the two vectors. For cosine similarity, a larger magnitude indicates greater similarity between two vectors. By multiplying the cosine similarity by -1, we redefine it as *distance loss*, where a larger magnitude indicates a greater difference between the two vectors. Therefore, the closer the *distance loss* is to zero, the greater the difference between the feature of the augmented image and forget data x_{forget}.

Table 1. Performance comparison using various distance loss Functions.

	Metric	Cosine Similarity	L1-norm	L2-norm
MUFAC (multi-class)	Test Acc. ↑	0.5601	0.5620	0.5685
	Top-2 Test Acc.↑	0.8226	0.8050	0.8362
	Forgetting Score↓	0.0744	0.0375	0.0305
	Final Score ↑	0.7056	0.7435	**0.7537**
MUCAC (multi-label)	Average Test Acc. ↑	0.8955	0.9066	0.9108
	Forgetting Score ↓	0.0314	0.0299	0.0032
	Final Score ↑	0.9163	0.9234	**0.9522**

2.2. Experimental Results for Each Distance Loss

In Table 1, we present the results for all three distance metrics – cosine similarity, L_1-norm, and L_2-norm – on both the MUFAC and MUCAC datasets. Overall, the L_2-norm demonstrates the highest performance in terms of the final score, consistently outperforming the other distance measures in both datasets. Notably, the GLI method using the L_2-norm distance achieves the best accuracy and forgetting scores across both tasks. This superiority is particularly evident in the multi-label task using the MUCAC dataset, where the L_2-norm shows a significantly improved forgetting score. On the other hand, cosine similarity results in the lowest overall performance, highlighting its inadequacy in comparison to the norm-based distance measures.

3. Visual Comparison of GLI-Based Images

In our work, we present a novel method, **GLI**, that generates mixed images that are adversarial against the images to be forgotten. The augmented images generated from GLI are semantically different from the forgotten images. Figure 1 shows a comparison between an original image from the Machine Unlearning for Facial Age Classifier (MUFAC) and its GLI-augmented counterpart. Figure 2 shows a comparison of the Machine Unlearning for Celebrity Attribute Classifier (MUCAC) with its GLI-augmented version. These examples demonstrate the effectiveness of the GLI method in generating mixed images that are adversarial against the images to be forgotten while being semantically different from the forgotten images.

4. Performance of SOTA Methods

This section presents a comprehensive analysis of the performance of various machine learning models over training iterations. We experiment with SOTA methods such as Fine-tuning, CF-K, NegGrad, UNSIR, EU-K, and SCRUB. The analysis is structured into several key areas:

4.1. Graphical Analysis of Model Performance

We present the performance of these models across training iterations through graphs, illustrated in Figures 3. Each graph measures performance using metrics such as accuracy, MIA (Membership Inference Attacks), and final scores, providing a visual representation of each model's performance trajectory. This approach provides insights into the learning dynamics of each model, highlighting how their performance evolves based on accuracy, resistance to inference attacks, and overall effectiveness.

4.2. Detailed Graph Analysis

Examining the training graphs, models like Fine-tuning, CF-K, UNSIR, Bad Teaching, and EU-K show similar learning patterns, characterized by fluctuating within a certain range without significant overall performance improvement. In contrast, NegGrad exhibits a catastrophic model utility drop during training. As a simple loss-increasing method, NegGrad experiences a catastrophic decrease in test accuracy on the original task after the 100-th training iteration. The SCRUB method displays the most dynamic changes in its learning graph. Initially, there is a sharp drop in accuracy, followed by a gradual upward trend, indicating a slight improvement in the final score over time. However, the performance is still not satisfactory.

On the other hand, our GLI method shows a relatively stable learning graph. A continuous decrease in MIA values indicates effective forgetting. While there is an inevitable trade-off in accuracy due to improved forgetting performance, the overall final score consistently remains the highest among the compared methods.

5. Additional Experiment Results

5.1. Results for Various Architectures

In our experiments, we have trained three state-of-the-art (SOTA) deep learning models– ResNet18 [8], WideResNet [20], and EfficientNet [18] – to tackle original classification tasks. In Table 2 and Table 3, we present results using WideResNet and EfficientNet that are not included in the main body of the paper due to space limitations. In both tables, other unlearning algorithm methods tend to exhibit lower forgetting scores when the model utility is high, or lower model utility when the forgetting score is high. In contrast, our method consistently achieves higher forgetting scores while preserving model utility. These results align with the goals of machine unlearning. Furthermore, while the results for WideResNet and EfficientNet indicate that the fine-tuning method performs well, our GLI method consistently shows superior or comparable performance, particularly in forgetting scores while maintaining model utility.

5.2. Results of Single Task Learning

In our primary experiments with the MUCAC dataset, which includes a multi-label task with three labels, our GLI model shows superior performance. To validate the effec-

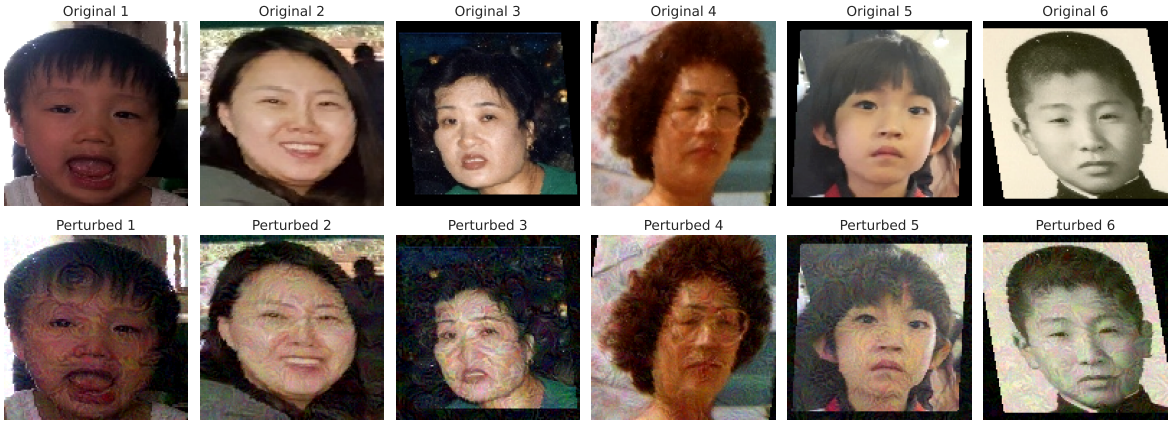


Figure 1. Example comparison of MUFAC original image and MUFAC perturbed image.

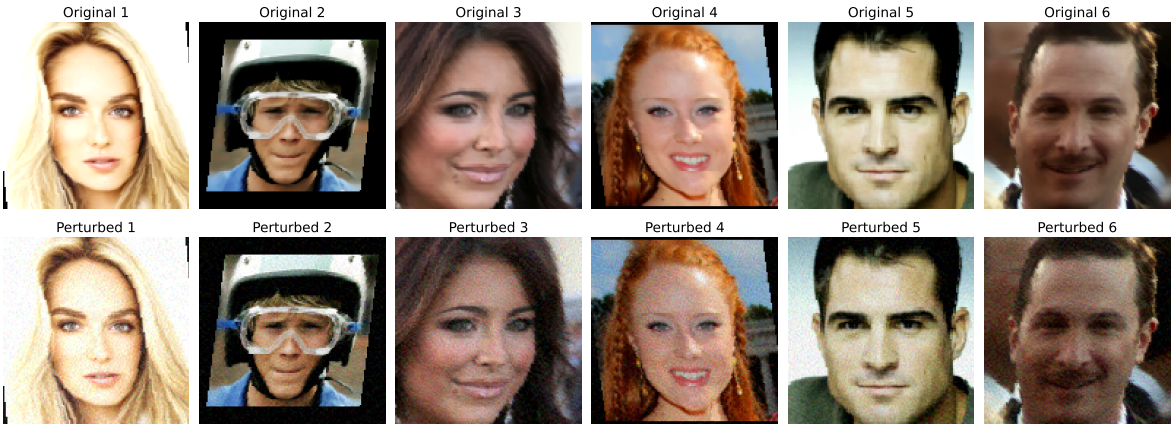
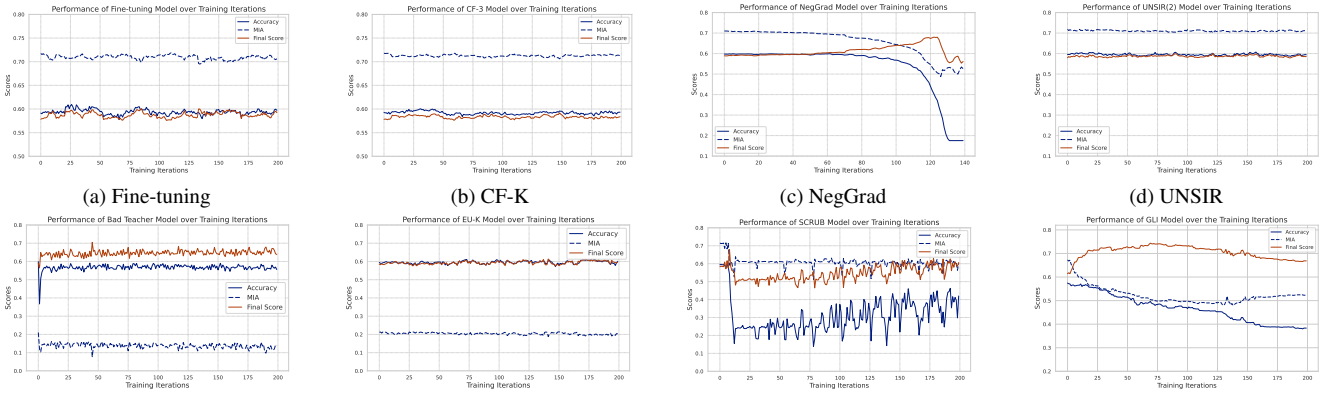


Figure 2. Example comparison of MUCAC original image and MUCAC perturbed image.

tiveness of our method in a more simplified context, we have extended our experiments to single-task learning scenarios using the same dataset. Table 4 shows the results of one of the representative MUCAC tasks, *facial young*, which involves classifying faces as young or old. The GLI model achieves a test accuracy of 0.9065, closely matching the 0.9017 accuracy of the original model. Notably, the forgetting score of our model is 0.0355, which is higher than the 0.0145 of the retrained model (our ground truth) but lower than the 0.0462 of the original model. This indicates that while our model effectively forgets, it does so without significantly compromising model utility. Moreover, the final score, which considers both model utility and forgetting ability, further supports our approach. Our model scores the highest with a final score of 0.9177, the highest among the models tested. This highlights the GLI method's capability to achieve effective unlearning while maintaining high model utility in both multi-label and single-task scenarios.



(a) Fine-tuning (b) CF-K (c) NegGrad (d) UNSIR

(e) Bad Teaching (f) EU-K (g) SCRUB (h) GLI

Figure 3. Overall Performance Trajectory of SOTA Models During Training Iterations.

Table 2. The results of WideResNet50-2's performance for two main classification tasks.

	Metrics	Original	Retrained	Fine-tunning [6]	CF-K [4]	NegGrad [5]	UNSIR [19]	SCRUB [9]	Bad Teaching [2]	EU-K [4]	**Ours (GLI)**
MUFAC (multi-class)	Test Acc. ↑	0.5984	0.5828	0.6042	0.5893	0.601	0.3242	0.6023	0.5802	0.5984	0.5763
	Top-2 Acc. ↑	0.8622	0.8648	0.8654	0.8401	0.8576	0.5464	0.8726	0.8239	0.8551	0.8180
	Forgetting Score ↓	0.2504	0.0817	0.2458	0.2521	0.2056	0.0700	0.2212	0.2225	0.2528	0.1328
	Final Score ↑	0.9099	0.9063	0.9174	0.9200	0.9220	0.9206	0.8916	0.9170	0.9183	**0.9376**
MUCAC (multi-label)	Average Test Acc.↑	0.8921	0.8590	0.8957	0.8953	0.8805	0.8950	0.8071	0.8799	0.8949	0.8921
	Forgetting Score ↓	0.0361	0.0232	0.0304	0.0276	0.0182	0.0269	0.0119	0.0229	0.0291	0.0084
	Final Score ↑	0.9217	0.9403	0.9504	0.9388	0.8523	0.9445	0.9273	0.8906	0.8982	**0.9522**

Table 3. The results of EfficientNetB0's performance for two main classification tasks.

	Metrics	Original	Retrained	Fine-tunning [6]	CF-K [4]	NegGrad [5]	UNSIR [19]	SCRUB [9]	Bad Teaching [2]	EU-K [4]	**Ours (GLI)**
MUFAC (multi-class)	Test Acc. ↑	0.5789	0.5425	0.5854	0.5880	0.1890	0.5010	0.5951	0.5607	0.5906	0.5724
	Top-2 Acc. ↑	0.8317	0.8265	0.8297	0.8284	0.3827	0.7582	0.831	0.7855	0.8187	0.8057
	Forgetting Score ↓	0.2784	0.0541	0.2744	0.2691	0.0162	0.0727	0.2621	0.2245	0.2744	0.0654
	Final Score ↑	0.51105	0.71715	0.5183	0.5249	0.5783	0.6768	0.53545	0.5558	0.5209	**0.7208**
MUCAC (multi-label)	Average Test Acc. ↑	0.9251	0.8978	0.9230	0.9236	0.9046	0.9241	0.8205	0.8933	0.9254	0.9096
	Forgetting Score ↓	0.0236	0.0142	0.0291	0.0324	0.0179	0.0316	0.0142	0.0114	0.0299	0.0107
	Final Score ↑	0.9389	0.9347	0.9324	0.9294	0.9344	0.9304	0.8961	0.9352	0.9328	**0.9441**

Table 4. The results of performance for binary classification tasks.

	Metrics	Original	Retrained	Fine-tunning [6]	CF-K [4]	NegGrad [5]	UNSIR [19]	SCRUB [9]	Bad Teaching [2]	EU-K [4]	**Ours (GLI)**
Facial Young (binary-class)	Test Acc. ↑	0.9017	0.8493	0.9215	0.9196	0.1733	0.9142	0.9214	0.9046	0.9213	0.9065
	Forgetting Score ↓	0.0462	0.0145	0.0488	0.0581	0.0895	0.0663	0.0615	0.0453	0.0575	0.0355
	Final Score ↑	0.9047	0.9097	0.9119	0.9017	0.5001	0.8908	0.8992	0.9070	0.9031	**0.9177**

6. Comparison Unlearning Algorithm

In our experimental setup, we adopt several SOTA methods for machine unlearning, such as CF-K [4], EU-K [4], NegGrad [12], UNSIR [19], SCRUB [9] and Bad Teaching [2], which are widely recognized in this field. However, these recently developed techniques predominantly focus on a class-unlearning approach. Therefore, we have modified certain aspects of these methods to better suit a task-agnostic framework. In cases where an existing machine unlearning technique is designed to create synthetic data that maximize the loss for specific *classes to forget*, we modify this approach. Our adaptation involves generating synthetic data that increases the divergence from the *instances to forget*, thereby providing an enhanced variation of the original methods.

6.1. CF-K Training Method

CF-K method [4] utilizes the phenomenon known as "catastrophic forgetting" in neural networks, where the network tends to lose information about previously learned samples. In this method, the last k layers of the model are fine-tuned using the remaining data set after excluding specific data. The earlier layers are again kept frozen. The aim is to make the model forget the information related to the data that has been excluded.

Algorithm 1 CF-K Training Epoch

Input:
  Retain dataset D_{retain}
  Number of epochs N
Output:
  Fine-tuned model θ_{ft}
Load and initialize model θ
Define CrossEntropyLoss L, SGD optimizer O
Unfreeze last layers of θ for fine-tuning
for $iteration = 1, \ldots, N$ **do**
  for each batch (x, y) in D_{retain} **do**
    Compute outputs $\theta(x)$
    Calculate loss $L_{\text{batch}} = L(\theta(x), y)$
    Update θ with O
  end for
end for

6.2. Negative Gradient Ascent (NegGrad) Method

The NegGrad unlearning method can optimize the model by maximizing the loss [12]. We call this method **NegGrad** following the previous work [5]. The NegGrad unlearning method Fine-tunes the dataset by moving in the direction of increasing the loss of forget data samples. This is equivalent to using a negative gradient to make the samples of forget data $x_{forget} \in \mathcal{D}_{forget}$. This method aims to corrupt the feature's prediction to correctly predict forget data samples.

Algorithm 2 Negative Gradient Ascent (NegGrad) Training Epoch

Input:
  Retain dataset D_{retain}
  Forget dataset D_{forget}
  Number of epochs N
  Model architecture function M
  Learning rate α
Output: The unlearned model $\theta_{\text{unlearned}}$
Load and initialize $\theta_{\text{unlearned}}$
Define criterion L_{cls}, optimizer O with α
Create iterator for D_{forget}
for $iteration = 1, \ldots, N$ **do**
  Set $L_{\text{running}} = 0$
  for each batch b, $(x_{\text{retain}}, y_{\text{retain}})$ in D_{retain} **do**
    Calculate L_{ascent}
    $= -L_{\text{cls}}(\theta_{\text{unlearned}}(x_{\text{forget}}), y_{\text{forget}})$
    Perform gradient ascent: O.zero_grad(),
    L_{ascent}.backward(), O.step()
    Update L_{running} with L_{ascent}
  end for
  Compute L_{epoch}
end for
Compute performance metrics with $\theta_{\text{unlearned}}$

6.3. UNSIR Method

UNSIR (Unlearning by Selective Impair and Repair) [19] method facilitates the removal of data from one or multiple classes without the need to access the data being excluded. It strategically increases noise in parts of the model that are not directly related to the target classes, helping the model to recognize and ignore patterns linked to these classes.

Originally designed for unlearning from specific classes, we have adapted UNSIR to focus more on maximizing noise in relation to the model's loss in classes not central to the learning objective. This modification assists in identifying patterns that are conducive to unlearning across a diverse array of tasks. Essentially, this process resembles creating a counter-sample for different classes, which is then used to disrupt and unlearn previously acquired knowledge. This adaptation allows UNSIR to be more flexible and effective for a broader, task-agnostic application.

6.4. SCRUB Method

SCRUB [9] is based on a novel teacher-student formulation in which the student model selectively disobeys the all-knowing teacher, inheriting only knowledge that is irrelevant to the data to be deleted. SCRUB can produce very high errors on deleted data, which is desirable in some scenarios but can make it vulnerable to membership inference

Algorithm 3 UNSIR Training Epoch

Stage 1: Impair
Input:
  Unlearned model architecture $\theta_{\text{unlearned}}$
  Retain dataloader D_{retain}
  Forget dataloader D_{forget}
  Number of epochs N
Output: Impaired model parameters θ_{impaired}
Load and initialize $\theta_{\text{unlearned}}$
Define CrossEntropyLoss criterion L_{CE}, SGD optimizer O with learning rate α
for $iteration = 1, \ldots, N$ **do**
  For each batch, extract $(X_{\text{retain}}, Y_{\text{retain}})$, $(X_{\text{forget}}, Y_{\text{forget}})$
  Initialize and optimize noise to increase loss with forget data
  Train $\theta_{\text{unlearned}}$ with noise-enhanced data and retain data
end for

Stage 2: Repair
Re-initialize L_{CE}, O
for $iteration = 1, \ldots, N$ **do**
  For each batch, extract $(X_{\text{retain}}, Y_{\text{retain}})$
  Compute classification loss L_{CE}, update $\theta_{\text{unlearned}}$ parameters
end for

attacks. To mitigate this where appropriate, we extended SCRUB with a new "rewinding" procedure that pinpoints "checkpoints" in the unlearning process to use so that the error on deleted data is close to a "high enough" threshold.

6.5. Bad Teaching Method

The concept of the Bad Teaching [2] method illustrates that unlearning for single-class or multi-class scenarios can be achieved through a teacher-student framework. In this setup, the teacher model transfers knowledge to the student model, but with a twist: it either removes or overwrites the data meant to be unlearned while maintaining previously learned information. The student model is then trained to discard the old information while assimilating this new, selectively filtered knowledge. This framework is adept at facilitating the forgetting of data across various scales, including single classes, multiple classes, or even subclasses.

6.6. EU-K Method

EU-K Method [4] involves retraining the last k layers of a model from the beginning using the remaining data set after excluding specific data that needs to be forgotten. The earlier layers of the model are kept unchanged (or frozen) during this process. The goal is to make the model precisely forget the information related to the excluded data.

Algorithm 4 SCRUB Training Epoch

Input: Pretrained teacher model θ_{teacher} from original_model, Initialized student model θ_{student} from scrub_model, Retain dataloader D_{retain}, Forget dataloader D_{forget}, Number of epochs N
Output: Fine-tuned student model $\theta^*_{\text{student}}$
Initialize SCRUB Training: θ_{teacher}, θ_{student}, D_{retain}, D_{forget}
Define Cross-entropy loss L_{CE}, Distillation divergence L_{div}
Initialize SGD optimizer O with learning rate $\alpha = 0.001$ for θ_{student}
for $iteration = 1, \ldots, N$ **do**
  Set θ_{student} to training mode and θ_{teacher} to evaluation mode
  Initialize $Loss_{\text{retain}} = 0$, $Loss_{\text{forget}} = 0$
  for each batch $(X_{\text{retain}}, Y_{\text{retain}})$ in D_{retain} **do**
    Compute $O^{\text{retain}}_{\text{student}} = \theta_{\text{student}}(X_{\text{retain}})$
    Compute $O^{\text{retain}}_{\text{teacher}} = \theta_{\text{teacher}}(X_{\text{retain}})$
    Compute $L^{\text{retain}}_{\text{CE}} = L_{\text{CE}}(O^{\text{retain}}_{\text{student}}, Y_{\text{retain}})$
    Compute $L^{\text{retain}}_{\text{div}} = L_{\text{div}}(O^{\text{retain}}_{\text{student}}, O^{\text{retain}}_{\text{teacher}})$
    Compute total loss $L^{\text{retain}}_{\text{total}} = L^{\text{retain}}_{\text{CE}} + L^{\text{retain}}_{\text{div}}$
  end for
  for each batch $(X_{\text{forget}}, Y_{\text{forget}})$ in D_{forget} **do**
    Compute $O^{\text{forget}}_{\text{student}} = \theta_{\text{student}}(X_{\text{forget}})$
    Compute $O^{\text{forget}}_{\text{teacher}} = \theta_{\text{teacher}}(X_{\text{forget}})$
    Compute negative loss $L^{\text{forget}}_{\text{div}} = -L_{\text{div}}(O^{\text{forget}}_{\text{student}}, O^{\text{forget}}_{\text{teacher}})$
  end for
end for

Algorithm 5 Bad Teacher Training Epoch

Input:
    Retain dataset D_{retain}
    Forget dataset D_{forget}
    Number of epochs N
    Good teacher model θ_{good}, loaded from $\text{path}_{\text{original}}$
    Bad teacher model θ_{bad}, initialized with random weights.
    Student model θ_{student}, cloned from θ_{good}
    Cross-Entropy loss function $\mathcal{L}_{\text{CE}}$
    Knowledge Distillation loss function $\mathcal{L}_{\text{KL}}$
    SGD optimizer $\mathcal{O}$ with learning rate α
Output: The trained student model θ_{student}
Set θ_{good} and θ_{bad} to evaluation mode; θ_{student} to training mode.
Define set of batches B for D_{retain}, D_{forget}
for $iteration = 1, \ldots, N$ **do**
    for each batch $b \in B$ **do**
        Extract $(x_{\text{retain}}, y_{\text{retain}})$ from D_{retain},
        $(x_{\text{forget}}, y_{\text{forget}})$ from D_{forget}
        Compute student logits $\ell_{\text{student}} =$
        $\theta_{\text{student}}(x_{\text{retain}}, x_{\text{forget}})$
        Without gradients, compute teacher logits
        $\ell_{\text{good}}, \ell_{\text{bad}} = \theta_{\text{good}}(x_{\text{retain}}), \theta_{\text{bad}}(x_{\text{forget}})$
        Calculate classification and distillation losses
        $\mathcal{L}_{\text{CE}}, \mathcal{L}_{\text{KL}}$
    end for
end for

Algorithm 6 EU-K Model Unlearning Epoch

Input:
    Retain dataset D_{retain}
    Unlearning model θ_{euk}
    Initial model θ_{initial} before unlearning
    Set of layers to unlearn $\mathcal{L}_{\text{unlearn}}$
Output: The unlearned model θ_{euk}
Disable gradients for all parameters in θ_{euk}.
Restore and enable gradients for weights of specific layers:
for each layer $l \in \mathcal{L}_{\text{unlearn}}$ **do**
    $\theta_{\text{euk}}[l] \leftarrow \theta_{\text{initial}}[l]$
    Enable gradients for $\theta_{\text{euk}}[l]$
end for
Fine-tune θ_{euk} using SGD optimizer $\mathcal{O}$ with $\alpha = 0.01$ and $\mathcal{L}_{\text{CE}}$ on D_{retain}
Set θ_{euk} to training mode
for $iteration = 1, \ldots, 2$ **do**
    for each batch $(x, y) \in D_{\text{retain}}$ **do**
        Compute logits $\ell = \theta_{\text{euk}}(x)$
        Compute loss $\mathcal{L} = \mathcal{L}_{\text{CE}}(\ell, y)$
        Backpropagate $\mathcal{L}$ and update θ_{euk} using $\mathcal{O}$
    end for
end for
return θ_{euk} after unlearning

References

[1] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017. 1

[2] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7210–7217, 2023. 4, 5, 6

[3] Luigi De Angelis, Francesco Baglivo, Guglielmo Arzilli, Gaetano Pierpaolo Privitera, Paolo Ferragina, Alberto Eugenio Tozzi, and Caterina Rizzo. Chatgpt and the rise of large language models: the new ai-driven infodemic threat in public health. *Frontiers in Public Health*, 11, 2023. 1

[4] Shashwat Goel, Ameya Prabhu, and Ponnurangam Kumaraguru. Evaluating inexact unlearning requires revisiting forgetting. *arXiv preprint arXiv:2201.06640*, 2022. 4, 5, 6

[5] 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. 4, 5

[6] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *ECCV 2020: 16th European Conference, Glasgow, UK, 2020, Proceedings*, pages 383–398. Springer, 2020. 4

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. 1

[8] 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. 2

[9] Meghdad Kurmanji, Peter Triantafillou, and Eleni Triantafillou. Towards unbounded machine unlearning. *arXiv preprint arXiv:2302.09880*, 2023. 4, 5

[10] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt, 2023. 1

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. 1

[12] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33:16025–16036, 2020. 5

[13] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016. 1

[14] Radi P. Romansky and Irina S. Noninska. Challenges of the digital age for privacy and personal data protection. *Mathematical Biosciences and Engineering*, 17(5):5288–5303, 2020. 1

[15] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *CoRR*, abs/1804.00792, 2018. 1

[16] H Jeff Smith, Sandra J Milberg, and Sandra J Burke. Information privacy: Measuring individuals' concerns about organizational practices. *MIS quarterly*, pages 167–196, 1996. 1

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. 1

[18] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2

[19] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 4, 5

[20] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019. 2