Efficient Two-stage Model Retraining for Machine Unlearning

Junyaup Kim
PurpleChips, South Korea
yaup22cc@purplechips.ai

Simon S.Woo
Sungkyunkwan University, South Korea
swoo@g.skku.edu

Abstract

With the rise of the General Data Protection Regulation (GDPR), user data holders should guarantee the "individual's right to be forgotten". It means user data holders must completely remove user data when they receive the request. However, enabling a deep learning model to exclude specific data used during training is challenging. We cannot easily define the meaning of "forgetting" in deep learning and how to achieve it. To address this issue, we propose an efficient machine unlearning architecture to be used for computer vision classification models. Our approach consists of two-stage models, where in the first stage we enables a deep learning model that loses information with contrastive labels in the requested dataset. Second, we retrain the first stage output model with knowledge distillation (KD). Using this two-stage approach, we can substantiate the removal or forgetness of the requested dataset in the deep learning model. With various datasets used for multimedia applications, we demonstrate that our approach achieves performance on par or even higher accuracy than the original model, while effectively removing the requested data.

1. Introduction

With the development of the Internet, various data are being shared, collected by internet participants. Many internet service providers are collecting user information to improve user-friendly services and highly interested in analyzing user information to provide customized services or to capture hidden business opportunities. Most of the dataset are copyrighted or sensitive such as health checkup information, GPS location, and financial information. As information technology evolves in the future, considerable amount of personal data will be sent and received and more businesses will utilize sensitive personal information in their services. But the problem with these collected personal information is that from an individual's point of view, it is difficult to claim rights to their data because it is being collected in fine detail and difficult to notice how much their data is being used.

The General Data Protection Regulation (GDPR) [11] provided regulatory guidance on this data management situation. The GDPR guarantees the "individual's right to be forgotten" that is individuals can limit the scope of commercial use on their personal information, and entities utilizing personal data must faithfully respond to individual data related requests. These rights are not limited to simply erasing the corresponding data records from the database, but also to all entities that indirectly utilize the data. Examples include machine learning algorithm because it performs high-level abstraction through representations of given data [15]. For compliance with the regulation, data managers must process the individual data according to individual's requests.
It is not problematic to modify or erase a user's records from a database management system. But making deep learning model erase the information of the requested data is troublesome. The reasons are mentioned in a previous study [1]. Frist,

we could not fully measure contribution of each data point in training deep learning model. Second, stochasticity is inevitably occur in training. Lastly, stochasticity is leveraged updating model parameters

1.1. Our Contribution

In this paper, we propose a efficient two-stage model retraining algorithm, which enables a deep learning model to efficiently erase the information of the requested dataset on deep learning model. First, we make the model forget the subset of the training dataset requested by users. We will denote this forgetting procedure as "neutralization". Second, we retrain the first stage output model with remaining dataset using knowledge distillation. In the first stage, we force the original model to partially erase the information of the requested data. We proposed practical measurement method of forgetness in deep learning model and validated the forgetness of requested dataset by using augmentation. In the next stage, we retrain the neutralized model to recover the performance using knowledge distillation. By this, we assure that the output of our procedure does not utilize the requested dataset but produces same or superior performance than the original model. Our contributions are summarized as follows:

- We propose an efficient model retraining algorithm that is composed with two stages, neutralization and retraining using knowledge distillation.
- Our algorithm can be applied in existing deep learning models and can deal with the number of data deletion requests in deep learning model with empirically low computational cost.
- We conducted our experiment in real world situations and showed precise ablation studies to prove feasibility of our approach.

2. Related Work

Machine Unlearning. Machine unlearning is the field that making efficient algorithm to forget the information of specific subset data from a deep learning model. To forget the specific data points, the model remove the feature of the subset dataset information in the model. Many researchers are actively studying Machine Unlearning, as academic and industrial interest in privacy has increased [2, 5, 6, 9, 10, 12, 14, 17]. Cao et al. [3] showed an unlearning algorithm using a small number of summations in the restricted setting of statistical query learning, where the learning algorithm cannot access to specific samples. This approach showed the machine learning model is possible to retrain using small amount of samples, quickly. Other researches require to involve splitting the data into multiple subsets to retrain models separately on combinations of them. Lucas et al. [1] proposed an unlearning algorithm that reduces the computational cost by slicing the dataset into smaller sharded-data and retraining on each the sharded-data. However, it require to set the number of sharding and slicing properly. In contrast, our approach is not required the any hyper-parameters. Wu et al. [16] propose the algorithm for rapid retraining machine learning models based on information cached during the training phases to reduce the learning time. In this paper, they proposed the DeltaGrad algorithm for rapid retraining machine learning models based on information cached during the training phase. Shokri et al. [13] showed that classifying the data used in training can be done by inference on the model and measure probability.

Knowledge Distillation. Knowledge distillation(KD) is a model compression algorithm introduced by Hinton et al. [8], which can transfer knowledge from a large model to a small one. The goal of KD is that a small model called student mimics the behaviors conducted by a large model called teacher. Cho et al. [4] proposed KD is an alternative way to improve the student model by ensembling method. In this work, we use KD to efficiently retrain the model that forgets the requested data. By using the KD algorithm, we measure of how one probability distribution is different from a second, reference probability distribution to mimic the teacher model.

3. Our Approach

3.1. Notations

In this paper, we define terms that will be used to explain our algorithm as follows: $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3).., (x_n, y_n)\}$ denotes the original dataset that has the number of the data point as number of n and (x_i, y_i) is the pair of image inputx_i and image classy_i. Also, $\mathcal{M}_D$ denotes the deep learning model that trained by using the dataset D. In our situation, we have to exclude subset of data in D and make deep learning model $\mathcal{M}_D$ forget the extracted dataset in D. Let us assume that individuals requested to data manager to remove the data from the dataset D. Requested dataset should be removed in D will be notated as $P(P \subset D)$. In this scenario, we will reproduce deep learning model D neutralized on P so that the performance of the model on P is equivalent as randomness. Neutralized model will be denoted as $M_D^{'}$ and $C_D = \{c_1, c_2, c_3, ..., c_m\}$ denotes the set of classes in the dataset when m means the number of classes in the dataset D.

3.2. Characterize the Machine Unlearning

Revisiting Machine Unlearning concept Here, we arranged commitment on providing fundamental explanations about Machine Unlearning concept and procedure. Not only researchers but also developers, general users on data analytic service and member of society should understand our proposed concept without needs of deep background knowledge on statistics and machine learning. By cause of previous motivation, We loosely defined the boundaries of the Machine Unlearning as the accuracy of $M_D^{'}$ on P is should be same as $\frac{1}{C_P}$. It implies the model's prior on P should be equivalent as randomly selecting classes on each input. If the deep learning model accuracy on P is higher than random, than the deep learning model can be considered to have prior on the corresponding dataset P. We acknowledge that even though the deep learning model has not been trained with dataset P, the accuracy of the corresponding dataset P could be higher than random. However, in order to explain that we have eliminated the prior on dataset P, we could not accept the accuracy that higher than random on dataset P and vice versa. Even if the deep learning model has a lower accuracy than random, it also implies that model has a prior on the corresponding dataset P. The gist is that training deep learning model positively or negatively with dataset P can be considered to have information about dataset P. Considering this background, we thoughtfully propose the goal of Machine Unlearning should be rearrange original deep learning model to have approximately same performance as random selection on user-requested dataset P.

Model neutralization with Contrastive label Model neutralization start with model output accumulation on dataset P. In this algorithm, we accumulate the model output of each data in P to calculate global minimum value on given output. The goal of this part is to find pair of classes that have most distances based on the output of the model. To do that, we first define 2 dimensional array that have $n(Y_D)$ size. Since the objective of the deep learning model $\mathcal{M}$ is tries to classify given class set C_D, the output will be same as the number of element in C_D and we are calculating per class minimum label, we have to accumulate per class output. First we calculate the output of each data on

Algorithm 1: Model output accumulation

```
   Input: $\mathcal{M}_D, P, C_D, C_P$
   Output: $Matrix_c$
   Initialize: zero initialized 2-dimensional
               array $Matrix_c[n(Y_D)][n(Y_D)]$
1  foreach $(x_i, y_i) \in P$ do
2      $Matrix_c[y_i] += \mathcal{M}_D(x_i)$
   /* divide values by count   */
3  foreach $label \in Y_P$ do
4      $Matrix[label] =$
         $Matrix[label]/$(number of $label$ data
         in $P$)
5      $Matrix[label][label] \leftarrow \infty$
6  foreach $label \in Y_D/Y_P$ do
7      $Matrix_c[label] \leftarrow \infty$
8  return $Matrix_c$
```

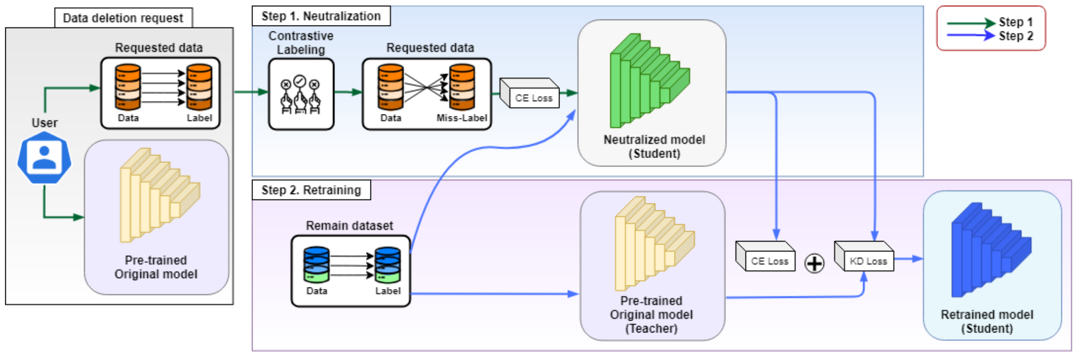


Figure 1. Overall procedure of two-stage model retraining. First, we retrain the original deep learning model by using contrastive labels mapped in requested dataset which described as neutralization. And we retrain the neutralized deep learning model with remaining dataset using kl-divergence loss only with remaining dataset.

model P and save it on $matrix_c$. In the next stage of algorithm, we average each label output based on the number of label in P because in the dataset, the number of data in given class can be different from each other. Also, we make unit matrix coordinates to go ∞ because we have to relocate each label to different label. lastly, we also make unused label coordinates to go ∞. By this, we can relocate the labels only using C_P labels. If the dataset P has only one class, we use all the classes. In the next algorithm, we calculate contrast labels in given matrix $matrix_c$. First, we find minimum value in 2 dimensional array $matrix_c$. first dimension will be named as from class because this class has minumum value in given matrix. Next, we calculate minimum value index in $matirx_c[from_class]$ because this class should be matched to furthermost class. and after we match two labels fill the designated coordinate as ∞ so that we can find another globally minimum value in given matrix.

Model neutralization The concept of neutralization is to make our model loss the critical information about the specific subset of dataset P. In this stage, we train the model M_D by using stored label list $Mat[0..len(C_X)]$. When each epoch increases, the model forced to trained with mislabled dataset P continuously unless the accuracy of the dataset P is below random prediction.

Retraining the Neutralized Model. The next stage is shown in Figure 1 as light blue box. In this

Algorithm 2: Contrastive label extraction

```
   Input: $Matrix_c$
   Output: $ContrastLabelList[m]$
   Initialize: $arraylength = n(Y_P)$
1  while $arraylength > 0$ do
2      init $templist[m]$
3      for index in range($Matrix_c$) do
4          $templist[index]$ =
             minvalue($Matrix_c[index]$)
5      $fromclass$ =
         index($templist$,$min(templist)$)
6      $toclass$=
         index($Matrix_c$,$min(Matrix_c[fromclass])$)
7      $ContrastLabelList[fromclass] =$
         $toclass$
8      for $label \in Y_P$ do
9          $Matrix_c[label][toclass] \leftarrow \infty$
10     arraylength -= 1
11 return $ContrastLabelList[m]$
```

Algorithm 3: Model neutralization

```
   Input: $\mathcal{M}_D, P, ContrastLabelList[m]$
   Output: $\mathcal{M}'_D$
1  while accuracy of $\mathcal{M}'_D$ on dataset
     $P < 1/n(Y_D)$ do
2      foreach $(x_p, y_p)$ in $P$ do
3          $train(\mathcal{M}'_D(x_p, ContrastLabelList[y_p]))$
```

stage, we are retraining the neutralized model with dataset $\mathcal{D} \setminus P$. We retrain the neutralized model using the remaining data except the requested data.

During training time, we apply the KL divergence loss with Teacher and Student model that is proposed by Hinton et al [8] to train the model faster and ensure the stability of training. In this training phase, the neutralized model is student model, and the original model is the teacher model. The student model is trained with soft label generated by original model, and use from the softmax function :

$$\delta(x) = \frac{exp(x_i/\mathcal{T})}{\sum_i exp(x_i/\mathcal{T})} \tag{1}$$

The temperature $\mathcal{T}$ in E.q 1 helps that the student model can mimic information of the teacher model by softening the label. That is, the higher $\mathcal{T}$, the more information it provides by softening the probability distribution. In the experiment, Our retraining part uses two loss function at the same time. the final loss function is combined with soft label knowledge distillation loss2 and cross-entropy loss3 with bias factor as follows.

$$\mathcal{L}_{KD} = \sum_{i=1}^{N} \delta(\hat{y}_i)\delta(\log \frac{\hat{y}_i}{y_i}) \tag{2}$$

$$\mathcal{L}_{CE} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}), \quad (M > 2) \tag{3}$$

$$\mathcal{L}_{TOTAL} = \alpha\mathcal{L}_{CE} + \beta\mathcal{L}_{KD} \tag{4}$$

For training efficiency, we set alpha and beta value as 1.1, 0.9 each in 4 To validate our training efficiency that can merge faster, we stopped using knowledge distillation loss after 10th of epochs and used only cross entropy loss with no bias factor.

4. Experimental setting

We run our experiments using P100 and TITAN RTX GPUs, with 24 GB of dedicated memory, respectively. We use Intel Xeon Gold 6230 CPUs with 8 cores each and 24GB of RAM. The underlying OS is Ubuntu 18.04.2 LTS 64 bit. We use PyTorch v1.4.0 with CUDA 10.1 and Python 3.7. For reproductivility, we fixed random seed, disabled the benchmarking feature and did not used non deterministic algorithms in PyTorch.

4.1. Datasets and Setup

We use several public benchmark datasets in machine learning field to demonstrate that our algorithm works well in general situation. Also, we deploy medical image dataset related to personal information for demonstrating our method as practical and its applicability in real-world scenarios. We represent the details in Table1.

Table 1. Dataset characteristics and model architecture.

Dataset description		
Name	**# of Instance**	**Model Architecture**
MNIST	70,000	2 Conv.layers followed by 1 FC layer
Fashion-MNIST	70,000	2 Conv. layers followed by 1 FC layer
CIFAR-10	60,000	2 Conv. layers followed by 1 FC layer
CIFAR-100	60,000	ResNet50 followed by 1 FC layer
SVHN	99,289	ResNet50 followed by 1 FC layer
HAM10000	10,015	ResNet50 followed by 1 FC layer

We build a deep learning model for each dataset as simple as possible, as the objective of our method is to unlearn the requested dataset in efficient way not performing well for the specific tasks, which does not require a complicated classification model. The details of these models are represented in Table 1, where Conv. layer represents the convolutional layer in the deep neural network and FC layer means fully connected layer. For relatively complicated dataset such as SVHN or CIFAR-100, we deploy the most popular convolutional neural network, Resnet-50 [7].

Pre-Training. For preparing an original model, we train the model with Adam optimizer with the setting of learning rate as 0.001. The number of epochs is 100 for CIFAR-10, CIFAR-100, and HAM10000 (Skin Cancer MNIST), and 50 for MNIST, SVHN, Fashion-MNIST.

Retraining Neutralized model. To verify the efficiency of our method, we retrain the neutralized model with the setting of the epoch as half to the pre-training epoch. This is because we assume that the neutralized model is faster to train than an end-to-end scratch model based on the conjecture that the neutralized model could be already familiar to other information excluding the requested data. While we retrain the neutralized model, we apply the cross-entropy and KL-divergence which is proposed by Hinton et al. [8].

Test-Time Augmentation. Since there is no precise measurement to prove certainty about forgetting requested data, we have to show practical output of the neutralized model forget the requested data if it does not perform well on both requested data and data with similar features. Thus, we leverage TTA to demonstrate that our retrained model does not remember the requested data points, and even the similar data points with maintaining the model performance. We note that we only use TTA as horizontal flip, vertical flip, and a center crop that magnifying 90% around the center of the image, as using various augmentations harms the diversity of samples.

5. Results

We compared the performance of the models as follows: original, retrained and scratch model. We note that the retrained model is the model trained with our proposed method. We conducted 100 experiments with varying the portion of the requested data from range 0.1% to 10% with step size 0.1% based on each total dataset. We used accuracy as performance metric. In Table 2, we represented the average accuracy of the experiments for all datasets. While our model produced 65.25%, 41.90%, 87.51%, 99.20%, 91.14% and 74.39% accuracy, the scratch model produced 64.43%, 38.15%, 87.85%, 98.89%, 91.09% and 68.42% accuracy for all datasets, respectively. For all datasets, our model proved the superiority than the scratch model. Also, the variance of accuracy is significantly smaller than scratch model with the varying portion of the requested data, which means our model is robust against the number of the requested data. Thus, we demonstrated that our method successfully retrained the neutralized model efficiently and effectively. With the observations, we conducted various ablation studies including the effect of the constrastive labeling and the impact of the increasing amount of the requested data.

Table 2. The Average accuracy for all datasets. The blue text indicates the variance for accuracy.

Model	CIFAR-10	CIFAR-100	SVHN	MNIST	Fashion MNIST	HAM 10000
Original	64.35	41.61	86.54	99.20	90.63	72.25
Retrained (Ours)	**65.25(0.24)**	**41.90(0.71)**	**92.64(0.32)**	**99.20(0.03)**	**90.94(0.15)**	**74.39(1.61)**
Scratch	63.43(0.79)	38.15(4.9)	89.18(6.83)	98.89(0.15)	90.23(0.47)	68.42(1.64)

Table 3. Average epoch of saving point from the requested data ranges from 1 to 10 percent of total training data.

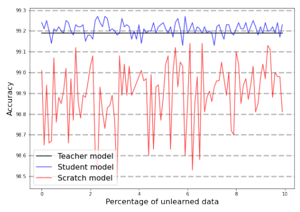
Model	CIFAR-10	CIFAR-100	SVHN	MNIST	Fashion MNIST	HAM 10000
Neutralized (Ours)	**12.4 (11)**	**8.6 (35)**	**10.9 (33.2)**	**11.50 (24)**	**10 (1.3)**	**29 (48.6)**
Scratch	23.4	43.6	44.1	35.50	11.3	77.6

5.1. Measuring efficiency of neutralization

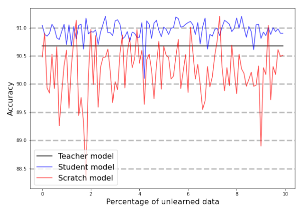
To demonstrate the efficiency of our method, we compared the average epoch for saving points when the models produced the best performance on the validation dataset. We recorded each saving point for each experiment, where the portion of the requested data varies in the range from 1 to 10 percent of total dataset. After recording the points, we calculate the average points for each dataset. As we can see in Table 3, the saving points for the neutralized model are earlier on all datasets prominently except for the Fashion-MNIST. The blue text indicates the average gap between our neutralized model and scratch model. This indicates that our method can efficiently retrain the original model and time-saving in Machine Unlearning.

5.2. Feasibility of contrastive labeling

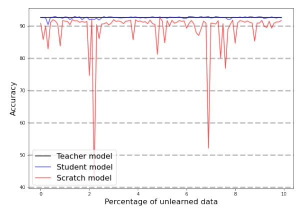
To validate the effect of the contrastive learning in the neutralization phase, we compare two



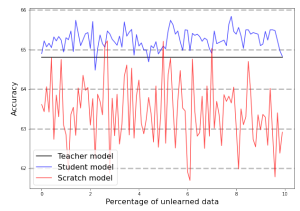
(a) MNIST



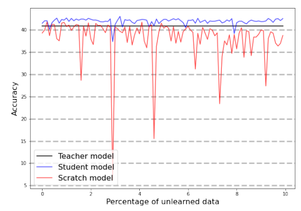
(b) Fashion-MNIST



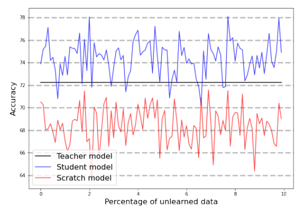
(c) SVHN



(d) CIFAR-10



(e) CIFAR-100



(f) HAM10000

Figure 2. The performance results for all datasets used in our experiments. The X-axis represents the percentage of each dataset we try to unlearn for our neutralization procedure, and the Y-axis represents the test accuracy of our neutralized model.

Table 4. Experiment of the average accuracy between minimum mislabel and maximum mislabel

Label type	Minimum Mislabel			Maximum Mislabel		
Dataset	Horizontal flip	Vertical flip	Center crop	Horizontal flip	Vertical flip	Center crop
CIFAR-10 (10%)	3.85	6.11	3.00	14.2	18.36	7.29
CIFAR-100 (1%)	0.58	0.76	0.49	5.31	3.8	2.37
SVHN (10%)	6.95	4.66	0.45	16.43	4.98	3.73
MNIST (10%)	7.28	8.54	1.95	10.37	9.2	5.68
Fashion-MNIST (10%)	1.27	6.37	0.56	10.55	9.44	10.43
HAM10000 (14.3%)	5.43	6.23	2.46	10.81	11.33	7.3
AVG.	**4.23**	**5.45**	**1.49**	11.28	9.52	6.13

learning methods: minimum mislabel and maximum mislabel. For minimum mislabel, we saved the predicted logits for each class on the requested dataset. Then, we averaged the logits and selected the class having the lowest predicted score for each class, which means the selected class is the most far from the actual label in the embedding space. For maximum mislabel, we conducted the same procedure as the minimum mislabel except selecting the class with the second-highest predicted score for each class, which means the selected class is the closest to the actual label in the embedding space. Thus, we assumed that minimum mislabel would produce a lower performance than maximum mislabel. In table 4 We conducted experiments, where the requested data ranges from 1 to 10 percent of total training data and marked random accuracy rate on the right side of dataset name. The model trained with minimum mislabel produced lower accuracy than maximum mislabel for all sort of augmentation on all datasets. Also, it produced lower accuracy than random on all sort of augmentation on datasets. However, the model trained with maximum mislabel produced higher accuracy than random, which is critical in machine unlearning as the model partially remembers the features of the requested dataset. Thus, we empirically demonstrated that using minimum mislabel is important to effectively delete the prior the requested data in deep learning model.

5.3. Impact of the Amount of the Given Data While Unlearning

To empirically show the rapidness of our neutralization against the amount of the requested dataset, we experiment with the point of stopping epoch as increasing the requested data while training the origin model.

Table 5. List of the last epoch in neutralization stage based on each ratio of data.

Ratio of data (%)	CIFAR-10	CIFAR-100	SVHN	MNIST	Fashion -MNIST	HAM10000
1	37	5	11	16	28	16
2	57	279	5	15	12	3,689
3	33	5	4	14	19	4,208
4	41	58	3	9	12	3,535
5	35	6	3	9	8	15
6	39	20	3	6	9	2,325
7	34	6	4	6	9	2,084
8	37	11	2	5	12	1,819
9	31	8	2	5	10	1,559
10	33	13	1	3	14	1,421

In Table 5, we measured the epoch number when the accuracy of the neutralized model is zero percent. Even considering that we randomly extract the data for each percentage and some extraordinary values, we can see that there is almost no significant change in the number of epochs for neutralization as the percentage of the data we extract increases. which means our method is applicable for various unlearning scenarios.

5.4. limitations

In applications where personal information is notably important, random accuracy on model is weak. In addition, our algorithm is difficult to apply in applications other than supervised multiclass classification. Lastly, Machine Unlearning has a relatively short history compared to other computer vision fields, and criteria, datasets, and experimental conditions that can be used as baseline are not shared and researched deeply.

6. Conclusions and Future Work

In this paper, we propose simple but notable retraining architecture to do machine unlearning and retraining with two stages. First stage is model neutralization that make the model forget about requested dataset by using contrastive labes. In this stage, the original model trained using contrastive labels in the requested dataset. contrastive label means that we pick the class that has least probability in certain data. To measure the forgetness, we used accuracy of requested dataset that should be same or below randomness. In particular, picking leaset probability class outperform higher than the label based on second maximum loss. Second stage is retraining the output of first stage model using dataset $\mathcal{D} \setminus \mathcal{P}$. In this stage, we used half of training epoch compared with scratch model and also used half of data in each epoch compared with scratch model. Experiment result showed that same or better accuracy performance than the original model. We also show the neutralized model learn quickly than both the scratch model and original model.

Acknowledgment

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)), (No. 2019-0-01343, Regional strategic industry convergence security core talent training business) and the Basic Science Research Program through National Research Foundation of Korea (NRF) grant funded by Korea government MSIT (No. 2020R1C1C1006004). Also, this research was partly supported by IITP grant funded by the Korea government MSIT (No. 2021-0-02309 and No. 2021-0-02157, Core Technology Development of Artificial Intelligence Industry).

References

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020. 1, 2

[2] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In 2015 IEEE Symposium on Security and Privacy, pages 463–480, 2015. 2

[3] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In 2015 IEEE Symposium on Security and Privacy, pages 463–480. IEEE, 2015. 2

[4] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4794–4802, 2019. 2

[5] 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, editors, Computer Vision – ECCV 2020, pages 383–398, Cham, 2020. Springer International Publishing. 2

[6] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3832–3842. PMLR, 13–18 Jul 2020. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 2, 5, 6

[9] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models: Algorithms and evaluations, 2020. 2

[10] Yang Liu, Zhuo Ma, Ximeng Liu, Jian Liu, Zhongyuan Jiang, Jianfeng Ma, Philip Yu, and Kui Ren. Learn to forget: Memorization elimination for neural networks, 2021. 2

[11] Alessandro Mantelero. The eu proposal for a general data protection regulation and the roots of the 'right to be forgotten'. Computer Law & Security Review, 29(3):229 – 235, 2013. 1

[12] S. Schelter. "amnesia" - machine learning models that can forget user data very fast. In CIDR, 2020. 2

[13] Reza Shokri, Marco Stronati, and Vitaly Shmatikov. Membership inference attacks against machine learning models. CoRR, abs/1610.05820, 2016. 2

[14] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards probabilistic verification of machine unlearning, 2020. 2

[15] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. Computer Law & Security Review, 34(2):304–313, 2018. 1

[16] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In International Conference on Machine Learning, pages 10355–10366. PMLR, 2020. 2

[17] Minhui Xue, Gabriel Magno, Evandro Cunha, Virgilio Almeida, and Keith W. Ross. The right to be forgotten in the media: A data-driven study. Proceedings on Privacy Enhancing Technologies, 2016(4):389–402, Oct. 2016. 2