ARCANE: An Efficient Architecture for Exact Machine Unlearning

Haonan Yan[1,4] , **Xiaoguang Li**[1] , **Ziyao Guo**[1] , **Hui Li**[1*] , **Fenghua Li**[2,3] and **Xiaodong Lin**[4]
[1]State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, China
[3]School of Cyber Security, University of Chinese Academy of Sciences, China
[4]School of Computer Science, University of Guelph, Canada
yanhaonan.sec@gmail.com

Abstract

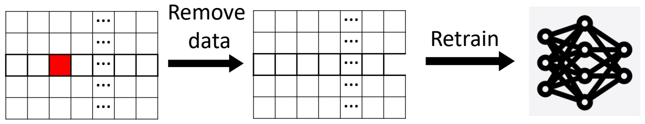
Recently users' *right-to-be-forgotten* is stipulated by many laws and regulations. However, only removing the data from the dataset is not enough, as machine learning models would memorize the training data once the data is involved in model training, increasing the risk of exposing users' privacy. To solve this problem, currently, the straightforward method, naive retraining, is to discard these data and retrain the model from scratch, which is reliable but brings much computational and time overhead. In this paper, we propose an exact unlearning architecture called ARCANE. Based on ensemble learning, we transform the naive retraining into multiple one-class classification tasks to reduce retraining cost while ensuring model performance, especially in the case of a large number of unlearning requests not considered by previous works. Then we further introduce data preprocessing methods to reduce the retraining overhead and speed up the unlearning, which includes representative data selection for redundancy removal, training state saving to reuse previous calculation results, and sorting to cope with unlearning requests of different distributions. We extensively evaluate ARCANE on three typical datasets with three common model architectures. Experiment results show the effectiveness and superiority of ARCANE over both the naive retraining and the state-of-the-art method in terms of model performance and unlearning speed.

1 Introduction

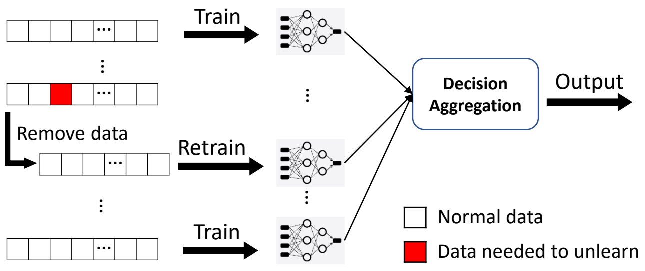
1.1 Background

Recently, the issue of over-collection of private information has raised the concern of regulators in many countries, and then related laws and regulations, such as European Union's GDPR [Mantelero, 2013], the California Consumer Privacy Act in USA [Harding *et al.*, 2019], and Personal Information Protection Law of China [Chen and Sun, 2021], have been promulgated to stipulate that service providers must fulfill the obligation to remove data of users when they receive data owners' requests, which is known as the *right-to-be-forgotten*.

[*]Corresponding Author



(a) Naive retraining



(b) Our architecture ARCANE.

Figure 1: An intuitive example to illustrate the biggest difference between naive retraining and our architecture ARCANE. When unlearning one data (red square), naive retraining requires retraining the model on the entire dataset, while ARCANE only needs to retrain one affected sub-model, whose retraining cost is much lower.

However, it is not enough for the provider to only remove these data from datasets, as the data that has already participated in the training would potentially exist in the memory of trained models. For example, the poisoning attack [Jagielski *et al.*, 2018] illustrates that information of the training data (i.e., trigger) is kept in the network. The model inversion attack [Fredrikson *et al.*, 2015] is able to recover samples in the training dataset only through the label and model as the pipeline. The membership inference attack [Rahman *et al.*, 2018] can determine whether one certain data is contained in the model's training dataset. Therefore, this spawns a new research direction of model privacy, model unlearning, to help service providers delete users' data as well as the contributions that have been generated to the model.

1.2 Previous Works and Problems

There are two main research routes for model unlearning, exact unlearning and approximate unlearning. In exact unlearning, the straightforward approach is to directly remove the data to be unlearned and retrain the model from scratch, also known as naive retraining. This solution often entails significant computational and time overhead. Subsequent researches follow this line and try to optimize model retraining to reduce the cost of retraining [Cao and Yang, 2015; Ginart *et al.*, 2019; Bourtoule *et al.*, 2021]. In contrast, approximate unlearning is a modification in model parameter space so that to remove the contribution of certain data to the parameter update, thus achieving an effect similar to retraining [Golatkar *et al.*, 2020; Guo *et al.*, 2020; Graves *et al.*, 2021]. Despite approximate unlearning schemes having significant advantages in terms of unlearning cost, [Shumailov *et al.*, 2021; Thudi *et al.*, 2021] recently show that such approximate schemes have serious security flaws. The effects of approximate unlearning could not prove their reliability and could be faked to deceive users. In contrast, it is naturally reliable for the exact unlearning to directly remove the data needed to unlearn from the dataset. Therefore, we consider the exact unlearning and try to decrease the retraining cost.

1.3 Our Work

In this work, we design an efficient **A**rchitecture fo**R** exa**C**t m**A**chine u**N**l**E**arning called ARCANE. We divide the dataset into several isolated sub-datasets, and independently train sub-models on each sub-dataset to participate in decision making. The final judgment is output by decision aggregation, as shown in Figure 1. Instead of uniform division, we divide the dataset by class (Section 3.1) to keep the impact of unlearning into one class, which reduces the accuracy loss. Then ARCANE utilizes the one-class classifier that performs well with unbalanced training data to improve the accuracy of sub-models. Especially in cases where a large number of data (more than 10% of the training dataset) is requested to unlearn, training dataset shrinking would degrade the performance of the unlearned model, which has not been considered by previous works, but the accuracy of the model using ARCANE is still competitive.

In addition, we preprocess each sub-dataset to further accelerate the model retraining, including representative data selection (Section 3.2), model training state saving (Section 3.3), and data sorting by erasure probability (Section 3.4). Data selection aims to select the most informative subset of the training set, and remove the redundancy and duplication, This would also significantly reduce the training volume of the model. Training state saving allows the model to retrain from the last state where there is no sample to unlearn, avoiding repeated computations. Sorting takes into account the non-uniform distribution of unlearning requests and places the sample most likely to be removed at the last saved training state to reuse the previous results as much as possible.

To demonstrate that ARCANE is efficient, we analyze and show the acceleration achieved by the service provider in handling unlearning requests after users send requests to delete their data. Our experimental results demonstrate that compared with the naive method of retraining from scratch, as well as the State-of-the-art (SOTA) method [Bourtoule *et al.*, 2021], ARCANE guarantees the accuracy of the retrained model and significantly reduces the number of samples that need to retrain, thus shortening the retraining time. The above results are all shown in Section 4.

Contribution. The main contributions are summarized as follows:

1. **Propose an exact unlearning architecture** ARCANE. Utilizing the one-class classifier, ARCANE significantly accelerates the exact unlearning and ensures the accuracy of the retrained model, even with a large number of unlearning requests, which is not considered by previous works, making ARCANE more practical.
2. **Introduce dataset preprocessing methods.** Representative data selection, training state saving, and sorting are considered in ARCANE to further speed up retraining. The experiment results show that these methods are effective in training speed not only for ARCANE, but also for other schemes.
3. **Evaluate** ARCANE. We conduct a comprehensive experiment to demonstrate that our architecture ARCANE is effective. Compared with naive retraining and the SOTA method, ARCANE is more advantageous in terms of both performance and time overhead of model retraining.

2 Related Work

Representative Data Selection. Data reduction could be roughly divided into two categories based on whether the model is involved. For model interactive works, [Orekondy *et al.*, 2019; Du *et al.*, 2019] all filter or sample the dataset by interacting with the model. For model non-interactive works, [Chouvatut *et al.*, 2015; Zheng *et al.*, 2017] could reduce the dataset without the need for model involvement. This kind of methods relies on no information from the model output or internal state and can be executed before model training. In this work, we directly consider the data selection from the perspective of the overall information amount of subset, which is more suitable for our problem.

One-class Classification. The existing methods for end-to-end learning mainly fall into two categories. The former's representative work is deep one-class classification [Ruff *et al.*, 2018]. The representative work of the latter is outlier exposure [Hendrycks *et al.*, 2019]. Afterward, [Sohn *et al.*, 2021] present a two-stage framework for one-class classification. They combine the two types of method and achieve the state-of-the-art performance. In our work, we transform the classification task into multiple one-class anomaly detection tasks and utilize the deep one-class method to train classifiers after preprocessing the training dataset.

3 Our Architecture ARCANE

We introduce each step in turn and show how our architecture solves the problem in previous unlearning methods. Figure 2 shows the workflow of model training using ARCANE.

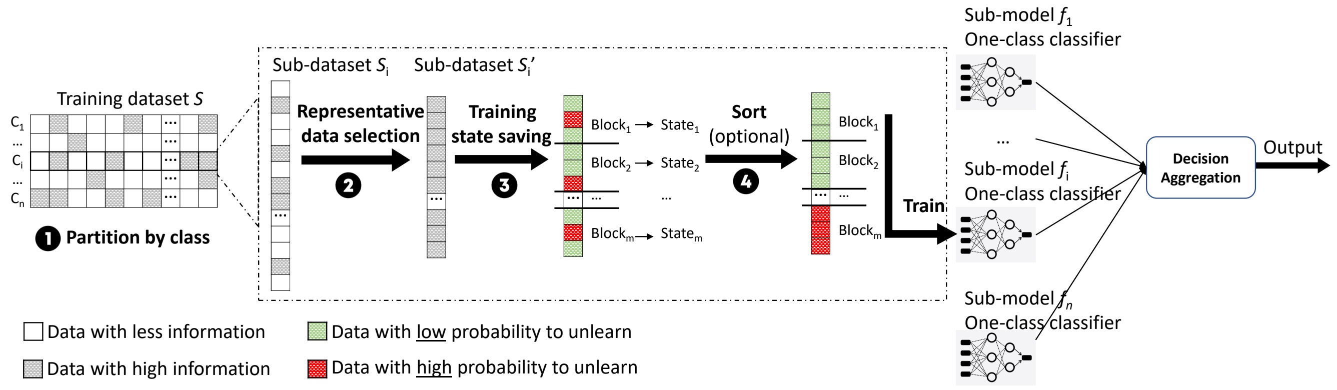

Training dataset S
C1
Ci
Cn
1 Partition by class
Sub-dataset Si
Sub-dataset Si′
Representative data selection
2
Training state saving
3
Block1 → State1
Block2 → State2
Blockm → Statem
Sort (optional)
4
Block1
Block2
Blockm
Train
Sub-model f1 One-class classifier
Sub-model fi One-class classifier
Sub-model fn One-class classifier
Decision Aggregation
Output
Data with less information
Data with high information
Data with low probability to unlearn
Data with high probability to unlearn

Figure 2: The overall model training process of our proposed architecture ARCANE.

3.1 Partition by Class

For a training set $\mathcal{S}$ with c labels in output domain $\mathcal{Y}$, c also denotes the number of class in $\mathcal{S}$. Here we choose to split the dataset by class instead of split uniformly. The reason is that the specific sub-model trained on the entire one-class dataset performs better than the weaker learner trained on a uniform dataset of the same small size. $\mathcal{S}$ is partitioned into c sub-datasets, $size(\mathcal{S}) = \sum_{i=1}^{c} size(\mathcal{S}_i)$. Each sub-dataset $\mathcal{S}_i$ only contains all data of one class, i.e., $\cap \mathcal{S}_i = \emptyset$ and $\cup \mathcal{S}_i = \mathcal{S}$, where $i \in \{0, 1, ..., c-1\}$. The sub-model f_i is trained using the sub-dataset $\mathcal{S}_i$. Benefiting from parallel computing, splitting datasets disjointly and training sub-models on each sub-datasets can distribute training costs and improve training efficiency. Note to be fair we do not take this into consideration in our analysis and experiments.

Here we clarify that ARCANE indeed needs more sub-models compared with the naive method. This is an inherent cost. However, we believe that the required cost is lower than expected in practice. Compared with one complex task, multiple simpler sub-tasks may be easier to directly deploy for unlearning. It also provides the possibility for distributed computing. Besides, it is a good deal to trade preparation cost for the performance gain (unlearning acceleration, usually hundreds vs. tens of thousands), as previous representative exact unlearning methods [Cao and Yang, 2015; Bourtoule *et al.*, 2021] all adopted, and we provided a more efficient option for model owners to unlearn data.

When one sample needed to unlearn, due to each sub-model f_i is trained in isolation, the model owners only need to retrain the corresponding sub-model after the unlearning sample is removed. Suppose that the amount of data in each class is similar in ordinary circumstances, partitioning the datasets could help naive retraining speed up c times with no other overhead.

3.2 Representative Data Selection

Training model on an adequate training dataset would achieve better performance, but the redundant information in the dataset would also increase, which severely slows down the training of the model. Thus, in this part, we use information theory to select the most informative subset from the original training set to remove redundant data, which speeds up retraining without reducing the prediction accuracy. Suppose n samples are selected from original sub-dataset $\mathcal{S}_i$ to form new sub-dataset $\mathcal{S}_i'$, i.e., $\mathcal{S}_i' \subset \mathcal{S}_i$ and $n = size(\mathcal{S}_i')$. This task is let the subset $\mathcal{S}_i'$ contain as much information as possible.

We use the joint entropy to measure the information amount contained in the subset $\mathcal{S}_i'$. For samples $X_1, ..., X_n \in \mathcal{S}_i'$, there is

$$\begin{aligned} H(\mathcal{S}_i') &= H(X_1, ..., X_n) \\ &= -\sum_{x_1 \in X_1} ... \sum_{x_n \in X_n} P(x_1, ..., x_n) \log P(x_1, ..., x_n) \end{aligned}$$

where $x_1, ..., x_n$ are the different values in samples $X_1, ..., X_n$ respectively, $P(x_1, ..., x_n)$ is the probability that these values appear together at the same time. We set $P(x_1, ..., x_n) \log P(x_1, ..., x_n) = 0$ if $P(x_1, ..., x_n) = 0$.

When there is one unlearning request, the model owners determine which sub-dataset $\mathcal{S}_i$ contains this sample. There would be two cases: (a) the unlearning data is in $\mathcal{S}_i'$, the probability is $n/size(\mathcal{S}_i)$, which will result in retraining n samples, or (b) the request is not in $\mathcal{S}_i'$, the probability is $(size(\mathcal{S}_i) - n)/size(\mathcal{S}_i)$, with no need to retrain, so it finally needs to retrain $n^2/size(\mathcal{S}_i)$ samples. For naive retraining, it needs to retrain $size(\mathcal{S}_i)$ samples.

3.3 Training State Saving

In each sub-model training and retraining procedure, we further divide the sub-dataset $\mathcal{S}_i$ into m disjoint data blocks B_{ij} uniformly, i.e., $\cap B_{ij} = \emptyset$ and $\cup B_{ij} = \mathcal{S}_i$, where $j \in \{0, 1, ..., m-1\}$. The blocks maintain a constant sequential relationship with each other during training. In the training of sub-model f_i, the data block B_{ij} is sequentially fed into the model for iterative update and the corresponding parameters $\boldsymbol{\theta}_{ij}$ are saved after training. The final result $\boldsymbol{\theta}_{i(m-1)}$ is the sub-model's parameter $\boldsymbol{\theta}_i$.

When one sample is needed to unlearn, the model owners determine which data block the sample is in. Let the sample is in the j-th block of sub-dataset $\mathcal{S}_i$, referred to as B_{ij}, the model owners could remove the sample from the j-th block to get a new data block, referred to as B_{ij}', and then start training from the saved state $\boldsymbol{\theta}_{i(j-1)}$ with B_{ij}'. Previous $j-1$ blocks do not need to join in the sub-model's retraining. Suppose unlearning sample may span from 0-th to $(m-1)$-th block, it may need to retrain from m to 1 blocks. In the case of a statistical average of m, a total of $(m+1)m/2$ blocks are required to participate in retraining, and for naive retraining, it always requires m^2 blocks. Thus, training state saving could help naive retraining speed up $2m/(m+1)$ times with only the overhead of space occupation for saving the model state.

3.4 Sorting by Erasure Probability

We take the distribution of the unlearning requests into consideration. Note that this step is not necessary but can further speed up the unlearning when the probability distribution of the data being requested to be unlearned is known for model owners. In practice, the data collector can determine which data in the dataset are more likely to be erased by users' requests based on some auxiliary information. Then the data with the highest probability are placed into the last blocks as much as possible so that when unlearning is needed, the model only needs to retrain on the last blocks, which reuses the previous results to the greatest extent and saves time.

When k samples with higher erasure probability are determined and would be unlearned, the model owners sort the datasets and then divide them into m blocks. Suppose these samples are randomly distributed. When $k < m$, which is more in line with the actual situation according to the report [Bertram *et al.*, 2019], naive retraining generally need to retrain k blocks. However, after sorting, it only needs $\lceil k/size(B_{ij}) \rceil$ blocks (normally is 1). Thus, sorting by erasure probability could normally help naive retraining speed up k times with negligible sorting consumption.

3.5 Model Architecture

In this part, we firstly introduce the one-class classifier used in ARCANE and then show how to aggregate the multiple decisions from each classifier to reach the final decision.

One-class Classifier

The task of a one-class classifier $f_i(\boldsymbol{x};\boldsymbol{\theta})$ in this work is to learn feature representations of its one-class data distribution and detect the data of other classes as out of distribution. $\boldsymbol{x}$ is input. The training goal of $f_i(\boldsymbol{x};\boldsymbol{\theta})$ is to learn a set of parameters $\boldsymbol{\theta}$ that make the sample in class i as close to the class representation r_i as possible, which force the network to learn to extract the common features of the samples in class i, i.e., the feature representations of class i. The objective function of the one-class classifier is defined as

$$\underset{\boldsymbol{\theta}}{\arg\min} \quad \frac{1}{n}\sum_{i=1}^{n}\left\|f_i(\boldsymbol{x}_i;\boldsymbol{\theta})-r_i\right\|^2+\frac{\lambda}{2}\sum_{l=1}^{L}\left\|\boldsymbol{W}^{[l]}\right\|^2, \tag{1}$$

where $\boldsymbol{x}_i \in \mathcal{S}'_i, n = size(\mathcal{S}'_i)$. The second term is a weight decay regularizer, where λ is a hyperparameter greater than 0 and $\boldsymbol{W}^{[l]}$ denotes the l-th layer weight.

Using adaptive optimizer Adam [Kingma and Ba, 2015] for weight $\boldsymbol{\theta}$ is computationally efficient and has little memory requirements. We empirically set r_i as the average of the representations of samples in one batch. Meanwhile, r is in the neighborhood of the initial network outputs, which also makes the model converge more faster and robust. After training, when a test sample $\boldsymbol{x}_i$ inputs to the classier $f_i(\boldsymbol{x};\boldsymbol{\theta})$, the sample's anomaly score can be given based on the distance between its feature representation and the class representation r_i, i.e.,

$$anomaly\_score = \left\|f_i(\boldsymbol{x}_i;\boldsymbol{\theta})-r_i\right\|^2.$$

Decision Aggregation

The strategy adopted in this work is that when only one classifier judges that the sample belongs to this class, the final result output is as this class. If there are multiple judgments for the respective class, the class with the lowest anomaly score, i.e., the highest confidence, is selected to be output as the result. Although the approach seems to jeopardize the performance of the model since each sub-model is trained on an extremely unbalanced dataset, only have one class, we evaluate our multiple one-class models in experiments to show that the loss of performance is very limited and can be mitigated when using an appropriate aggregation strategy.

4 Experiment

In this section, we firstly introduce our experiment setting in Section 4.1, and then evaluate our architecture's performance in Section 4.2. Next, we separately discuss the impact of representative data selection, training state saving, and sorting on the architecture for ARCANE. Finally, we show the comparison results with the SOTA mechanism.

4.1 Setup

The datasets used in this work are simple MNIST, moderate CIFAR-10, and complex Imagenet. The choice of datasets varies in the dataset size, input dimension, and task complexity. We use one subset of Imagenet, used in previous work [Hendrycks *et al.*, 2019], because of device limitations. There are 30 classes with 1,300 training images and 100 test images per class in this subset.

For models, we also vary the backbone network to show the generalization of our architecture, i.e., LeNet for MNIST, Wide ResNet for CIFAR-10, and ResNet-18 for Imagenet. Here we choose the same network for the baseline model and all sub-models for the one-class task. Each sub-model is trained only on the data of the corresponding one of the c classes. The unseen data of the remaining $c-1$ classes in test datasets are used as test data, which does not participate in the training. The number of epochs is 50. For the task of MNIST, we set the batch size to 256. For the task of CIFAR-10 and Imagenet, the batch size is 64.

We run all experiments on an Ubuntu 16.04.1 LTS 64 bit server, with one Intel(R) Xeon(R) Silver 4214 CPU, 128GB RAM, and one RTX 2080 Ti GPU with 11GB dedicated memory. Note that we have not chosen the SOTA models nor performance tuning for simplicity. This is also not the focus of this work. Compared with the accuracy performance of the original model, we pay more attention to the reduction in time and accuracy of model retraining after adopting ARCANE.

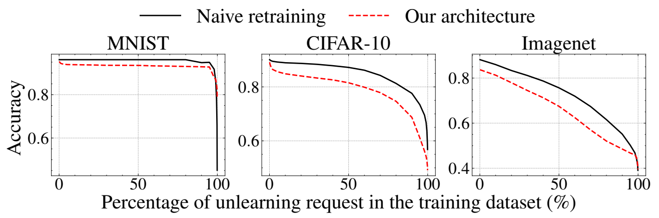


Figure 3: Accuracy comparison between naive retraining and our architecture on three datasets after different percentages of unlearning requests.

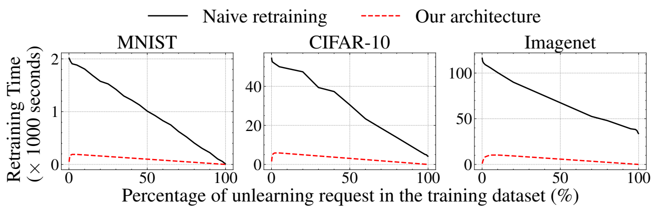


Figure 4: Retraining time comparison between naive retraining and our architecture on three datasets after different percentages of unlearning requests.

4.2 Overall Evaluation

We compare our architecture with naive retraining in terms of accuracy and retraining time. The different number of unlearned data is involved in the evaluation, and the data requiring unlearning is randomly selected. The parameters of ARCANE are set to default values, i.e., $n/S = 0.1$ and $m = 5$, unless specified otherwise. Each result is averaged over multiple repetitions.

Evaluation of Accuracy

In Figure 3, we set the percentage of the unlearning request in the training dataset from 0.1% to 99.9%. When there is no unlearning request, the accuracy of the original model on MNIST, CIFAR-10, and Imagenet are 96.13%, 90.10%, and 89.70% respectively, and the accuracy of models using our architecture are 95.28%, 89.3%, and 85.49%, respectively, which illustrates that our architecture would not cause the performance of the original model to drop too much. With the gradual reduction of training datasets, the accuracy of our architecture and naive training are also gradually declining, the gap is small and the trend remains the same.

Evaluation of Retraining Time

Figure 4 shows the comparison of the time cost of training models on datasets of different sizes. As the number of unlearning requests increases, the training dataset is shrinking, so the time of both two retraining decreases linearly. The results show that using our architecture has a very significant advantage in retraining time, especially when the number of unlearning requests is few.

The significant lead of architecture is owing to two reasons. The first one is the architecture of ARCANE. We divide the dataset into c sub-dataset at the beginning, and it only needs to train a few sub-models instead of the entire large model. The second one is our preprocessing methods. The representative

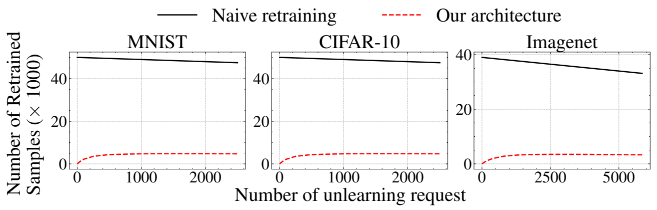


Figure 5: Number of retrained samples comparison between naive retraining and our architecture on three datasets after different small number of unlearning request ($\leq$ 5% for MNIST and CIFAR-10, $\leq$ 15% for Imagenet).

data selection algorithm significantly reduces the amount of training, and training data saving also drastically reduces the number of retrained samples when the proportion of unlearning requests to the total dataset is small (less than 5%).

In Figure 5, to further demonstrate the reasons for the reduction of retraining time in a small number of requests, we show the number of retrained samples with respect to the number of unlearning requests. As expected, the trend of Figure 5 is basically consistent with the one of Figure 4, which shows that our proposed scheme can truly reduce the number of samples that need to participate in retraining due to unlearning requests.

4.3 Impact of Data Selection

Figure 6 shows the impact of different proportions of selected informative data on retrained model's performance and time. We can see that the retraining time for all three tasks grows linearly with the increase of the number of selected samples. Therefore, when the model's accuracy is enough, the fewer data selected, the faster the retraining. For the simple task of MNIST, selecting 5% of the most informative data according to our algorithm is enough to obtain a high-accuracy model of 92.76%. For the moderate task of CIFAR-10, 30% data is enough for a model with an accuracy of 85.11%. For the hard task of Imagenet, it may need 65% for a model with an accuracy of 81.25%. Note here we do not tune the models and only focus on the accuracy gap between the original model and retrained model. The results indicate that our representative data selection method is feasible.

We test different proportion of unlearning request from 0.02% to 10% in the evaluation of two components (data selection in Figure 6 and state saving in Figure 7), and the final trends are all same. Therefore, we heuristically select two proportions to show results under various setting in limited length. We would like to clarify that this does not affect the final conclusion.

4.4 Impact of Training State Saving

Figure 7 shows the impact of training state saving on retraining time. The more data blocks used to retain the model state, the more frequently the model state is saved, and the less data needs to be involved in the retraining. Besides, when the number of unlearning requests is small, using our architecture would also apparently reduce the retraining number. When unlearning requests increase, almost every block contains samples needed to unlearn, and the reduction of training

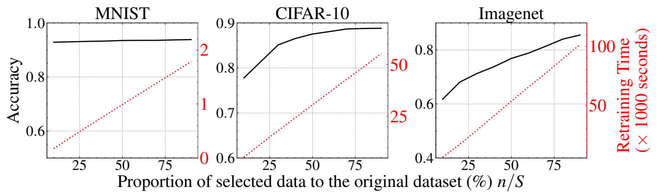


Figure 6: With different proportion of selected data, the accuracy and retraining time of our architecture on three datasets. We set the percentage of unlearning request in the training dataset to 1%, and set the number of blocks to 1, i.e., $m = 1$.

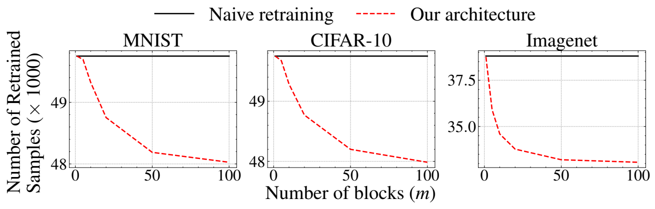


Figure 7: With different number of blocks, the number of retrained samples comparison between naive retraining and our architecture on three datasets. The proportion of unlearning requests is 1% to the training dataset.

datasets leads to a shortening of training data size, which both weaken the improvement of training state saving.

4.5 Impact of Sorting

Figure 8 shows the comparison of the number of samples needed to retrain with sorting (Our architecture) and without sorting (Naive retraining). In this test, the most common normal distribution is used to randomly sample unlearning requests. Other distribution can also be generalized into our architecture ARCANE. We can see that sorting is able to reduce the amount of retrained samples on all three datasets. In addition, the larger the training set to sort, the better the results would be, so the effect of MNIST and CIDFAR-10 are better than the one of Imagnet as $size(\mathcal{S}_{MNIST}) = size(\mathcal{S}_{CIFAR-10}) = 50,000 > size(\mathcal{S}_{Imagenet}) = 39,000$.

4.6 Comparison with SOTA

The parameter setting of comparison experiment follows the latest work [He *et al.*, 2021] at that time, which empirically studied and proposed the best setting of SISA in most practical cases. Then we set ARCANE consistent with the SISA, e.g., the parameter $m = 20$ (block number) is consistent with the R (slice number, which performs the similar function) of SISA, and the S of SISA is equivalent to the number of submodels in ARCANE.

Table 1 shows the comparison between our architecture and the naive retraining and SISA[Bourtoule *et al.*, 2021], which is considered as the SOTA method in exact unlearning. For SISA, in MNIST the number of shards is 50 and the number of slices is 20, and in Imagenet the number of shards is 10. For our architecture, we set $m = 20$ for fair, and c is set automatically based on the dataset. Overall, our architecture can reach a faster retraining with similar accuracy compared to SISA, especially when we perform selection, i.e., set $n/\mathcal{S} \le 1$, our architecture can significantly decrease the retraining time. To ensure the accuracy of Imagenet, we increase the proportion of selected data, which reduces the acceleration effect but is still faster than SISA. Note SISA also incorporates the state saving and sorting, but ARCANE still has the advantages due to data selection and one-class classification.

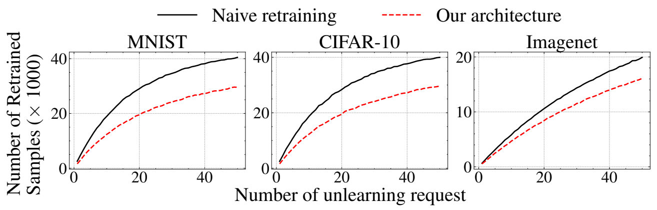


Figure 8: With or without sorting, the number of retrained samples comparison between naive retraining (without sorting) and our architecture (sorting) on three datasets after a different small number of unlearning requests. We suppose the unlearning requests follow the most common normal distribution.

Datasets	Method	Paramenter	Accuracy	Retraining Time
MNIST	Naive	None	96.13%	1
	SISA	S=50, R=20	93.32%	1/73
	Our	m=20, $n/\mathcal{S} = 1$	94.65%	1/29
	Our	m=20, $n/\mathcal{S} = 0.1$	93.69%	1/2891
Imagenet	Naive	None	89.70%	1
	SISA	S=10, R=20	82.04%	1/15
	Our	m=20, $n/\mathcal{S} = 1$	85.36%	1/68
	Our	m=20, $n/\mathcal{S} = 0.6$	82.70%	1/204

Table 1: Comparison results of naive retraining, SISA, and our architecture. The retraining time is shown as a multiple of the time of the naive method. Only 1 sample is requested to be unlearned.

With the increasing number of unlearning requests, ARCANE shows more superiority in terms of retraining speed and accuracy over SISA. In this procedure, the data selection of ARCANE plays an increasingly important role. The more requests, the more retrained data filtered out by the selection, and the faster the retrain. Besides, one obvious trend is that the accuracy of model would decrease due to the missing training data. Compared with naive retraining, ARCANE remains competitive (robustness from one-classier) but the degrade of sub-models in SISA damages the final aggregation accuracy more as SISA needs each data shard balanced enough. Besides, the main advantages of ARCANE compared with SISA are: 1) with one-class, ARCANE performs well on unbalanced training data, but SISA needs each data shard balanced enough. When large data unlearning, especially focusing on one specific class, the accuracy of ARCANE would not degrade too much but SISA cannot handle this situation well. 2) with data selection, the training and unlearning of ARCANE is much faster than SISA.

5 Discussion

ARCANE is exact unlearning. The difference between exact unlearning and approximate unlearning is how unlearning is done [Thudi *et al.*, 2021]. Exact unlearning is based on retraining at the algorithmic level, and approximate unlearning typically directly modify the parameters of the trained model, which is at the parameter level. In our work, the data selection is treated as part of model training at the algorithmic level, so ARCANE comply with the definition of exact unlearning.

Marginal benefits. The retraining speed of data selection outperforms state saving by several orders of magnitude. In overall, data selection indeed brings the best benefits on the unlearning speed. Other components play a role in different aspects. The one-class architecture lets the overall model more stable when facing lots of non-deterministic unlearning requests. State saving could avoid repetitive retraining and sorting could effectively cope with high unlearning probability data. They are also very useful in specific problems.

Role of data selection. SISA could also adopt the data selection to save retraining time, but there is a big challenge that its accuracy would drop a lot compared with ARCANE. The reason is that SISA divides the dataset uniformly and requires each shard has balanced and enough data for sub-model to maintain high accuracy. After data selection, too little data in each shard may be unbalanced and not enough, which would significantly reduce the accuracy of the sub-models. Consequently, many weak sub-models would result in a more accuracy drop in decision aggregation of SISA. However, one-class model does not require balanced dataset and is effective to bound the impact of the reduction of dataset amount in each specific class, which would not affect the decision of other class too much as the uniform partition.

Side effect. Using ARCANE would drop some accuracy, since ARCANE replaces model architecture with multiple sub-models using ensemble learning, which induces some accuracy drop for complex tasks (e.g., ImageNet). We would like to clarify that suitable aggregation strategies and augmentation methods could bridge this accuracy gap without varying the existing model hyperparameters and architecture based on [Bourtoule *et al.*, 2021]. Besides, stronger one-class classifiers in the future would also improve this situation a lot.

6 Conclusion

In this work, we devise an efficient model architecture for exact machine unlearning. By transforming the learning goal into multiple one-class classification tasks, we execute dataset prepossession to speed up model training and also reuse previous unaffected results to avoid repeated computing. Through the evaluation of typical datasets and models, the results show that using our architecture could remove the data needed to unlearn in a very fast retraining manner without loss of performance. We hope our architecture could help user to protect their privacy and promote model governance for service providers as an effective unlearning tool in reality.

Acknowledgments

We thank all the anonymous reviewers for their valuable guidance and constructive feedback. Part of Hui Li's work is supported by the National Natural Science Foundation of China (61932015), Shaanxi Innovation Team project (2018TD-007), Higher Education Discipline Innovation 111 project (B16037). Part of Haonan Yan's work is done when he visits the School of Computer Science at the University of Guelph.

References

[Bertram *et al.*, 2019] Theo Bertram, Elie Bursztein, Stephanie Caro, Hubert Chao, Rutledge Chin Feman, Peter Fleischer, Albin Gustafsson, Jess Hemerly, Chris Hibbert, Luca Invernizzi, et al. Five years of the right to be forgotten. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 959–972, 2019.

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

[Cao and Yang, 2015] 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.

[Chen and Sun, 2021] Jihong Chen and Jiabin Sun. Understanding the chinese data security law. *International Cybersecurity Law Review*, 2(2):209–221, 2021.

[Chouvatut *et al.*, 2015] Varin Chouvatut, Wattana Jindaluang, and Ekkarat Boonchieng. Training set size reduction in large dataset problems. In *2015 International Computer Science and Engineering Conference (ICSEC)*, pages 1–5. IEEE, 2015.

[Du *et al.*, 2019] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 477–487, 2019.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

[Ginart *et al.*, 2019] Antonio A. Ginart, Melody Y. Guan, Gregory Valiant, and James Y. Zou. Making ai forget you: Data deletion in machine learning. In *NeurIPS*, 2019.

[Golatkar *et al.*, 2020] 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.

[Graves *et al.*, 2021] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *AAAI*, 2021.

[Guo *et al.*, 2020] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *ICML*, 2020.

[Harding *et al.*, 2019] Elizabeth Liz Harding, Jarno J Vanto, Reece Clark, L Hannah Ji, and Sara C Ainsworth. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy*, 2(3):234–253, 2019.

[He *et al.*, 2021] Yingzhe He, Guozhu Meng, Kai Chen, Jinwen He, and Xingbo Hu. Deepobliviate: A powerful charm for erasing data residual memory in deep neural networks. *arXiv preprint arXiv:2105.06209*, 2021.

[Hendrycks *et al.*, 2019] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Neural Information Processing Systems*, 2019.

[Jagielski *et al.*, 2018] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[Mantelero, 2013] 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.

[Orekondy *et al.*, 2019] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[Rahman *et al.*, 2018] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11(1):61–79, 2018.

[Ruff *et al.*, 2018] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[Shumailov *et al.*, 2021] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross Anderson. Manipulating sgd with data ordering attacks. *arXiv preprint arXiv:2104.09667*, 2021.

[Sohn *et al.*, 2021] Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and evaluating representations for deep one-class classification. *ICLR*, 2021.

[Thudi *et al.*, 2021] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. *arXiv preprint arXiv:2110.11891*, 2021.

[Zheng *et al.*, 2017] Jian Zheng, Wei Yang, and Xiaohua Li. Training data reduction in deep neural networks with partial mutual information based feature selection and correlation matching based active learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2362–2366. IEEE, 2017.