Machine Unlearning of Pre-trained Model on CIFAR-100

Final Report

Leung King Suen, WANG Zekai
The Hong Kong University of Science and Technology
ksleungac@connect.ust.hk, zwanggd@connect.ust.hk

1. Abstract

In the world of supervised learning, the training and fine-tuning of a model is always backed by more training examples. However, in view of security and privacy challenges, there might be need to 'unlearn' the examples from a machine learning model. This is generally referred as 'Machine Unlearning'. This is a relatively new and undiscovered field. While there are published and known unlearning algorithms such the SISA (Sharded, Isolated, Sliced, and Aggregated training) method. They remained on a conceptual stage and lacked experimentation and evaluations of the effectiveness of the algorithms on the actual models. While almost all models for any task can be unlearned, we decided to experiment unlearning methods on the supervised-trained image classification models on the CIFAR-100 dataset. Several unlearning algorithms are implemented and evaluated against metrics such as classification accuracy on the test set and membership inference attack accuracy. Our experimentation showed that **the SISA method is better than other methods in a general manner. It is effective in unlearning task. With the technique of transfer learning, SISA can overcome the difficulties in training thus achieving a very high test accuracy after unlearning, and resolves many difficulties such as the over-fitting issue. So we believe SISA with transfer learning is a reliable and efficient approach in Machine Unlearning.** Besides the experimental results, our group gained considerable knowledge on machine unlearning by completing the project.

2. Introduction

Machine Unlearning refers to unlearn a subset of examples from a model that have been trained with the training set. While adding training examples to the model has always been easy, removing the influence of the the examples are not intuitive. To achieve such goal, one could always retrain a model from the ground up but that is not always possible given the many challenges in the training process. Thus it is important to experiment if there are better approaches without the need to retrain the model and, to evaluate them by experiments on the real-world CNN image classification models. A more vigorously written problem statement is presented below.

Consider a labelled dataset, the complete training set is used to train an model for image classification task. One could split the examples in the training set to random sets **Forget Set *F*** and **Retain Set *R***. Then for whatever reason, the influence of the **Forget Set *F*** needs to be removed from the models classification results.

There are 3 major ways of achieving this, including fine-tuning, SISA and SISA with transfer learning. However, it is a question to be answered for us on which methods has an better unlearning effectiveness. Effectiveness is defined by two points: better unlearning effect and better test accuracy after unlearning. These are evaluated with established metrics: top-1, top-5 classification accuracy and also MIA (Membership Inference Attack).

Throughout our experiment, We implemented unlearning algorithms like fine-tuning and the SISA. We found that SISA method provided generally more stable classification performance before after the unlearning task. Moreover, it is very effective on the unlearning task.

Effectiveness of each methods		
	Performance	Unlearning
Fine-tuning	Good	Poor
SISA	Poor	Good
SISA with transfer learning	Very Good	Very Good

3. Related Work

3.1. SISA Training

SISA stands for Sharded, Isolated, Sliced, and Aggregated training. It is a training strategy published in the paper [1]. It divides the training process into separate steps, minimizing costs by avoiding re-training on the full set of

data.

First, the training data is divided into multiple disjoint shards, ensuring that each training data is included in only one shard. Then, models are trained separately (meaning isolated) on each of these shards, which limits the scope of a data point only to the sub-model trained on the shard containing the point. Afterward, the training on each shard is further divided into steps. Each training step is on a subset of the data inside each shard, with the subset's size increasing along the training, and saving the weight of each step, this is called slices. Finally, depending on the unlearning instruction, only the affected model will be re-trained. This retraining begins on a particular slice, which should be the step before adding the unlearning data point into the training subset. Since shards are smaller than the entire training set, and slices provide a checkpoint, this decreases the retraining time to achieve effective unlearning.

We implemented and experimented such training strategy to facilitate unlearning to train a ResNet model. While we will be comparing SISA-trained unlearning with other unlearning methods, we will also investigate and conclude the several factors that will affect the unlearning by SISA, including the distribution of the **Forget Set *F*** in the sharded and sliced set, the number of shards, number of slices in each shard on the effectiveness and time-needed in the unlearning.

3.1.1 Membership Inference Attack

Membership Inference Attack (MIA) is a classification model that is able to determine whether an example in the set ***F*** is used in the training process of another machine learning model [7]. We will use a binary classification model based on the idea of this paper and use the accuracy of the MIA attack prediction to measure if the model indeed unlearned the **Forget Set *F***.

3.1.2 ResNet

The image classification model we are going to apply unlearn task is the popular CNN architecture such as ResNet [4]. We used pre-trained ResNet model with CIFAR-100 weights taken from this GitHub Repository [2] for the evaluation of original model before going through unlearning and the unlearning task that is by fine-tuning.

We will stick to ResNet throughout this project. In SISA, we will also use the ResNet model with ImageNet1K weights for transfer learning.

4. Data

The data that will be used in this project is the CIFAR-100 Classification Dataset [5]. It consists of 50000 training images, 10000 test images. It has 100 classes of common objects. Standard Data augmentation such as random transformation, horizontal flip were applied to it. The data was normalized with after it was converted to tensor.

For the specificity of our tasks for implementing the "Sharded, Isolated, Sliced" ideology in the SISA training, and to create a randomly distributed **Forget set *F***, the training set indexes is randomized.

With randomized index, the training set is then divided into **4** non-overlapped shards (parts) for training of **4** independent classification models. Then within each shard, the data is then sliced into **5** slices for the 'step and save' training of each model.

At the end, a **Forget set *F*** is formed by picking 20% of the training examples sequentially in the randomized training indexes.

5. Methods

We will start by using the starter-kit provided by the Neurlps2023 Machine Unlearning Challenge [3]. utilizing its code for the fine-tuning and Membership inference attack evaluation, however modifying it to generate our own **Retain, Forget set *R, F*** based on CIFAR-100 training data. Then we applied both of the following algorithms for unlearning.

5.1. Unlearning algorithms

We will be mainly experimenting the two unlearning algorithms below and compare if one or another is superior. We also be measuring the time needed for each algorithms to complete the task.

5.1.1 Unlearning by fine-tuning

This approach is to fine-tune the pre-trained model solely on the retained set. The intuition behind this method is to force the model to over-fit on the **Retain set *R***, thus overwriting the the old knowledge obtained during the training process.

In the experiments, we first evaluate the pre-trained models on the test set, then fine-tune it on **Retain set *R***, at the end evaluate on the test set again and conduct MIA attack again.

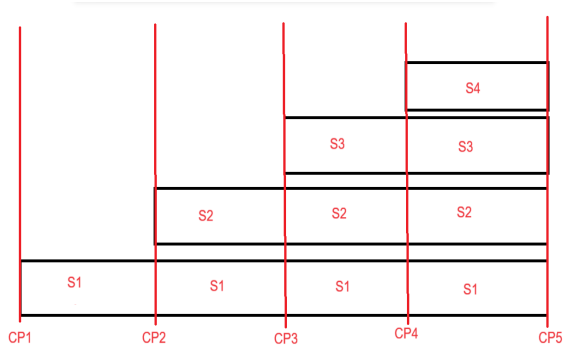

Figure 1. 4-slices training only for illustrations, our actual implementation is 5-slices

5.1.2 Unlearning by SISA training

As introduced in the Related Works section, This method splits the model into many models in training and take majority vote on the prediction.

To implement the SISA method, we first randomize an index mask for the training set, then create sized shards according to number of shards on the training set. This is to ensure that each model is able to be trained on a uniformly distributed subset of the training set. Finally we train model [0 to number of shards-1] consecutively.

For the training of each model, the shard is further splits into some arbitrary number of slices, a model is trained for some epochs on one slice, saved as a checkpoint, before proceeding to next slices, which consisted of new data from new slice and trained data from the old slice. AS illustrated in the drawing, the model was saved with starting checkpoint **CP0**, then proceed to train on slice **S1**, save at checkpoint **CP1**, then train on slice **S1+S2**, back and forth.

In the unlearning phase, we will first determine which of the model that needs retraining, then load the model checkpoint that is before the examples in the forget set, then retrain the model from the checkpoint skipping some examples that the model needs to unlearn. For example, if **Forget set *F*** is on slice **S3, S4**, we directly take the model at checkpoint **CP3**. If **Forget set *F*** is on slice **S1, S2**, we take the model at checkpoint **CP1** and trains it on Slice **S3 and S3+S4**.

We think SISA training is good in comparison to both retraining and fine-tuning. Compare to retraining, it is able to save time to not retrain the models that are not required for unlearning, and save time not to retrain the part of training data that has been trained while not in the **Forget set *F***. Compare to fine-tuning, it does explicitly omit the **Forget set *F*** in the re-training when a unlearning task is being done on the model.

5.2. Evaluation method

We determined the following two metrics be used for evaluation: Image classification accuracy and Membership Inference Attack Accuracy. These two metrics will be tested altogether to measure the performance unlearning algorithm.

5.2.1 Image Classification Accuracy

This metric is used to test that the unlearned model still performs well on the retain set ***R***. Therefore classification accuracy is a good metric to start with image classification task. Both top-1 and top-5 accuracy of the classification results before and after unlearning will be compared. We will test the model on set ***R*** and the test set.

The reason we introduced this accuracy evaluation aside from it is a go-to metric in evaluating a image classification model it that although we are conducting the unlearning task, the model needs to be able to retain knowledge on ***R*** after the unlearning, it should have the same performance on the ***R*** and test set [1].

5.2.2 Membership Inference Attack Accuracy

This metric is used to test that the unlearned model is no longer influenced by **Forget set *F***.

We input the concatenation of model's losses on the **Forget set *F*** and the test set, and the ground truth labelling that is of the same dimension of the concatenation, in which 0 denotes test set, 1 denotes **Forget set *F***. By utilizing the difference in the produced losses produced by the model (If an example is learned by the model, the loss of that example is lower), the MIA can distinguish the example is from test set of **Forget set *F***. The golden standard for this metric is close 0.5, meaning that the MIA classification produces random predictions.

If there is no apparent drop in the MIA accuracy after unlearning process, this indicates an ineffective unlearning algorithm.

5.3. Hypothesis

Our hypothesis for the experimental is, we expect all models are able to illustrate comparable classification accuracy before or after unlearning. However, this is going

to be different if we take MIA accuracy into account. We expect the model unlearned with SISA training strategy will perform the best as it indeed skip the slices in the **Forget set *F***, where the fine-tuning unlearning model will have worse performance, however all of them should be worse than the re-trained model ('the Golden Standard') without **Forget set *F***.

On the time efficient regard, we expect the SISA retraining will be the fastest given multiple models can be trained in parallel, followed by fine-tuning method. All of these method should be quicker than retraining on the **Retain Set *R***.

6. Experiments

6.1. Determining the Forget Set

There are 50000 examples in the train set. We picked 20%, that is 10000 examples as the **Forget set *F***. As we already have a randomized train set indexes, we took 10000 examples from index [20000, 30000) out of the total range [0, 50000). Then the retain set will be a concatenation([10000, 20000), [30000, 50000))

6.2. The Fine-tuning Method

We first experimented on the fine-tuning method. We selected pre-trained ResNet32, ResNet56 models on CIFAR-100 from [2] and test them against the training set and test set to setup a baseline performance before unlearning.

Pre-trained model (before unlearning)		
	Top-1 Accuracy(%)	Top-5 Accuracy(%)
ResNet32		
Train Set	71.7%	90.8%
Test Set	54.8%	80.5%
Forget Set ***F***	71.3%	90.8%
ResNet56		
Train Set	75.6%	91.9%
Test Set	56.0%	81.9%
Forget Set ***F***	76.6%	92.1%

Then we applied the fine-tuning method, that is to fine-tune on the **retain set *R*** for 20 epochs. Done with the following hyper-parameters:

- optimizer = SGD
- epoch = 20
- loss = Cross Entropy Loss
- scheduler = CosineAnnealingLRv
- learning rate = 0.1
- momentum = 0.9
- weight decay = 5e-4

The results were presented at the table below.

Pre-trained model (after unlearning)		
	Top-1 Accuracy(%)	Top-5 Accuracy(%)
ResNet32		
Train Set	77.6%	95.6%
Test Set	65.0%	89.9%
Forget Set ***F***	68.6%	91.7%
ResNet56		
Train Set	79.6%	96.1%
Test Set	66.6%	90.5%
Forget Set ***F***	69.1%	91.8%

From the tables above, we discovered that after 20 epochs of fine-tuning on the **Retain set *R***, the model solidifies its classification capability. This can be seen as the accuracy on the train set and test set increases. The forget set accuracy has dropped a little. Below is a overview of the more important metrics we used to evaluate the unlearning.

Unlearning by Fine-tuning Evaluation			
	Top-1 Test Acc	Top-5 Test Acc	MIA
ResNet32			
Before	54.8%	80.5%	0.581
After	65.0%	89.9%	0.534
ResNet56			
Before	56.0%	81.9%	0.590
After	66.6%	90.5%	0.514

We have three major conclusions based on the results observed:

1. Adopting simple fine-tuning solely based on the retained set shows a limited contribution on the unlearning task on the model. As observed from the evaluation above, the Top-1 accuracy of **Forget Set *F*** comparing to Train Set decreases to a small extent, while the Top-5 accuracy shows no clear decrease.
   Possible solution is to include more datasets like the forget set and validation set into the fine-tuning algorithm. If we cannot 'forget' passively by covering the old memory with newer memory, we have to force the model to 'forget' a particular set of data actively.
   Moreover, since the model we fine-tuned is very deep and has many parameters, it is also possible to achieve unlearning with more epochs on a simpler model. The evidence is that as we tried out different epochs of fine-tuning, the model fits much better. One reason we chose 20 epochs is that the result of 5 epochs is unsatisfactory, with its train set accuracy (70%) much lower than the original accuracy. Also, we tried out a smaller model on CIFAR10, and its forget set accuracy after

fine-tuning dropped about 10% from 100%. Therefore, we believe that with more fine-tuning epochs, it is possible to achieve unlearning to some extent even if we rely solely on the retained set. However, we think 10% is still not a satisfactory result; hence, it is worth exploring more advanced unlearning methods.

2. As observed from the MIA accuracy point of view, this fine-tuning approach shows some positive effect on unlearning. The MIA accuracy of ResNet32 and ResNet56 decrease to 0.534 and 0.514 respectively, which are close to 0.5, meaning that the MIA algorithm could not make correct judgements for distinguishing whether the data is included in or outside the training set.

3. If we take a look of the Top-K test accuracy, we surprisingly find that the test accuracy increases after unlearning by fine-tuning. We believe it is a side-effect of finr-tuning. This could be explained due to the effect of enhancing the **Retained set *R*** outweighed the effect on forgetting the **Forget set *F***. Making the overall performance of the model increases during test set evaluation.

From the above observations, fine-tuning seems to be a decent unlearning method. However, unlearning tasks in real world always come with strict requirements due to privacy or other important reasons, people expect a 100% forgetting rate. Hence, fine-tuning is not a valid approach. As quoted by others, the best way to "forget" data is to not "learn" the data. Therefore, the investigation on SISA method is needed.

6.3. SISA Method

6.3.1 Difficulties in SISA training

In training process by applying the SISA strategy with CIFAR-100, we encountered a pitfall that prevented us to train the model to have comparable performances to the pre-trained CIFAR-100 models.

In the process of training the Model 0 out of the 10 models for the 10 shard-ed experiment, we discovered that the test accuracy quickly saturated with about 20 epochs, the training accuracy continue increases eventually reaching 100% by epochs 60. We later concluded that in the SISA training, because of multiple independent models are trained on only a smaller subset of what it should be the whole training set, the individual model will quickly over-fit on the small training set, leaving test performance behind. For example, in a regular training for CIFAR-100 image classification, the model was fed with 50K training examples with 500 examples for each of the 100 classes. However, with SISA training of 10 shards, although the

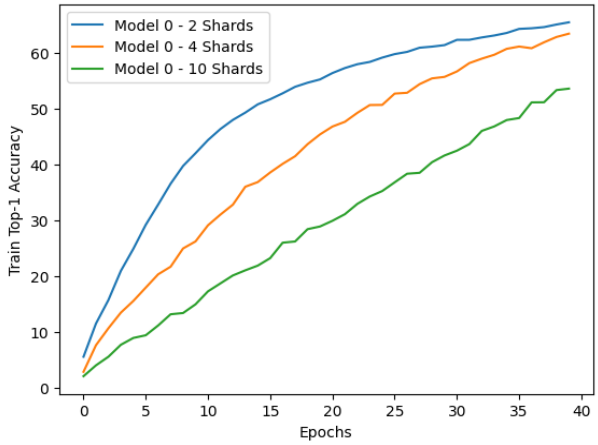

Figure 2. Training accuracy. With less shards, the training is less linear.

subset of which the shard was taken from the training examples are random, each model was only fed with 5K training examples, which is even less examples than that of the test set. Each SISA trained model are getting a fraction of examples over the whole model, the number in the shard-ed training examples is not enough to help the model to obtain enough understanding on the whole CIFAR-100 dataset. This reduction in the data input to the model led to such a performance degradation.

To proof this idea, we ran a quick 40 epochs training on ResNet32 with shards = 2, 4 and 10 with the following parameters

```
SGD(params, lr=0.1, momentum=0.9,
    dampening=0, weight_decay=5e-4,
    nesterov=True)
```

The training and test accuracy is recorded and shown below. The training history is taken by only looking at the Model 0 of the SISA method.

We can see although all of the 3 models showed upward trend in the training accuracy up to 40 epochs, the Model 0 of 10 shards training reached saturated test accuracy after epoch 30. It is worth noting that all of the models here have considerable lower performance to the model that is trained on the whole training set.

Also, for the number of slice, we believe that it is always good to keep as much as slices as possible within a shard. More slices means a lower chance to re-train the shard from the beginning, increasing the time efficiency of SISA.

Wrapping up the things, we decided **4 shards** and **5 slices** training is a good balance to strike as while it spreads

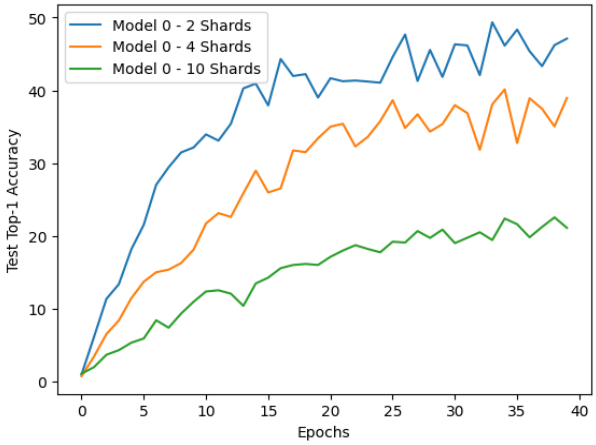

Figure 3. Test accuracy. The more the shards, the lower on test accuracy

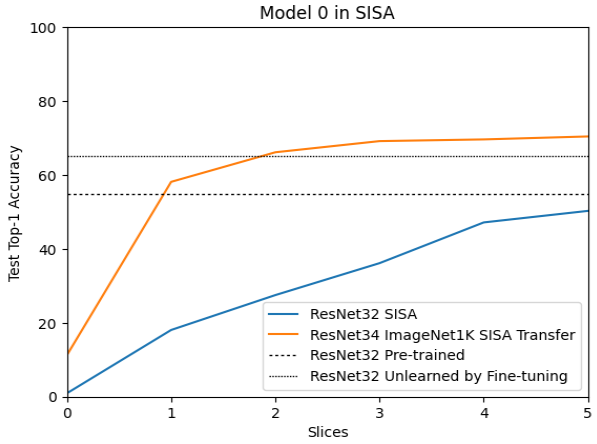

Figure 4. Comparison between SISA Transfer ResNet34 and SISA ResNet32

the training set to multiple models and slices, it does not hurt the performance of each model like the higher shard-ed settings. Then a further 100 epochs training with the introduction of a learning scheduler = CosineAnnealingLR pushed the test set accuracy to a further 50% on the Model 0.

However, we deemed this is still a unsatisfactory test set performance. This is not only because CIFAR-100 is a big dataset, its number of classes is relatively large com-pare with its total number training data. So if we evenly divide the train set T into 4 shards, each shards contains only 50000/100/4 = 125 training images for one class. This greatly affect the performance of the shard-ed model. As suggested in the Machine Unlearning paper, the author mentioned that transfer learning might be a good strategy to compensate for the performance losses in the shard-ed training [1], therefore we introduced *transfer learning* onto our shard-ed models.

6.3.2 Transfer Learning

Transfer learning is a popular technique in machine learning in which knowledge learned from a task is re-used in order to boost performance on a related task. In SISA, we could also take advantage of this technique to boost our test accuracy after unlearning.

In our case, since the target is to "unlearn" a set of data from CIFAR-100, we should not use the model pre-trained on CIFAR-100, but we can use models pre-trained on other dataset with comparable size. Generally speaking, in real world application, we could always use a publicly available pre-trained model as the transferred model in implementing SISA. Our choice is a **ResNet34** model with pre-trained weight on **ImageNet-1K** [4], which is provided by torchvision [6].

However, there are differences between CIFAR-100 and ImageNet1K:

	Dimensionality	Size	#Classes
CIFAR-100	32x32x3	60000	100
ImageNet1K	224x224x3	1281167	1000

Hence, we need to make some adjustments to both training & testing data and the transferred pre-trained model:

- Converting the size of images in CIFAR-100 to match the dimensionality of data in ImageNet1K. We use the technique of upsampling with Nearest-neighbor interpolation. Upscale the original images from 32x32x3 to 224x224x3. Only by doing so, the transferred pre-trained model could interpret the input data correctly.
- Changing the output feature's shape in the last dense layer of the transferred pre-trained model into 100. This is also a must-do step since our CIFAR-100 only have labels up to 100.

After the above adjustments, we gave a quick shot on training the transferred pre-trained model on the first shard Model 0:

It is such a cheering result, we observed a huge improvement in top-1 test accuracy. With more slices being trained, the performance of our transferred **ResNet34** model becomes even better than the original **ResNet32** pre-trained model used in fine-tuning approach. Which means, even the model is sliced, its testing performance does not drop but increases a little. This confirmed our decision in using transfer learning techniques in training SISA.

6.3.3 SISA training

As we have decided to use **4 shards** and **5 slices** settings, we trained 4 models on each 1/4 of the randomized training set. The models are pre-trained ImageNet-1K ResNet34 models, with output layers modified to 100 units for CIFAR-100

- Model 0 = randomized-train[0:12499]
- Model 1 = randomized-train[12500:25499]
- Model 2 = randomized-train[25000:37499]
- Model 3 = randomized-train[37500:49999]

Each model was trained 5 times using index [0:2500), [0, 5000), [0, 7500), [0, 10000),[0, 12500) from each shard using the following hyper-parameters:

- optimizer = Adam
- betas = [0.9, 0.999]
- learning rate = 1e-3 for FC-layers, 1e-4 for other layers
- epochs = 9 - (number of times)
- lr-scheduler = CosineAnnealingLR
- loss = nn.CrossEntropyLoss

6.3.4 Unlearning by SISA

After we have trained 4 models, we need to decide which model to re-train for the unlearning task. As the **Forget set *F*** is formed by sequentially picking indexes [20000, 30000) from randomized train set. This range falls on the last 2 slices of Model 1 and first 2 slices of Model 2. Which means for Model 1, we just took its checkpoint right after finishing the 3rd slices (there are 5 slices). For model 2, we re-trained it on the latter 3 slices.

We ran evaluation for classification accuracy (taking majority vote) and MIA accuracy (taking average), below are the performance of SISA training and unlearning:

Unlearning by SISA		
	Top-1 Test Acc	Top-5 Test Acc
ResNet34		
Before	78.1%	96.2%
After	77.1%	96.0%

Unlearning by SISA (MIA Accuracy)					
ResNet34	Shard0	Shard1	Shard2	Shard3	Mean
Before	0.494	**0.523**	**0.584**	0.506	**0.527**
After	0.490	0.496	0.501	0.507	**0.498**

The test accuracy for the ResNet34 model trained by SISA is almost the same before and after the unlearning, which indicates the algorithm is able to retain its performance after unlearning.

The MIA test showed that before unlearning, the MIA is successful (high accuracy) on shard 1 and 2, this is within our expectation because the **Forget Set *F*** lies in between shard 1 and 2, therefore this is particularly easy to be attacked by MIA. After unlearning, this pattern is no longer there, the MIA accuracy across different shards are similar and with a mean extremely close to 0.500 (the golden standard), indicating that, to MIA, it is impossible to distinguish the forget set and train set. Therefore unlearning with SISA is an effective algorithm to remove the influence of **Forget Set *F***.

6.4. Comparisons

Unlearning Algorithms Comparisons			
	Top-1 Acc	Top-5 Acc	MIA
ResNet32 Finetuning			
Before	54.8%	80.5%	0.581
After	65.0%	89.9%	0.534
ResNet56 Finetuning			
Before	56.0%	81.9%	0.590
After	66.6%	90.5%	0.514
ResNet34 SISA Transfer			
Before	78.1%	96.2%	0.527
After	77.1%	96.0%	**0.498**

6.4.1 MIA Accuracy

From the comparisons across the board, we can see the ResNet34 with SISA method obtained the best MIA accuracy after unlearning across all models, while fine-tuning method performs worse. We think this is because in the SISA unlearning stage, the model has been completely isolated from the **Forget Set *F***, that is to prevent the entrance of the **Forget Set *F*** into training process. This is not possible for the fine-tuning method, therefore they have lower MIA accuracy.

Another finding is that by applying the transfer learning techniques, not only we obtained a better performance on the shard-ed model, we also obtained a better MIA accuracy even before unlearning starts. We think this is due to there are already 'transferred' weights with the model, thus the influence of **Forget Set *F*** is also harder to propagate on the already established weight.

6.4.2 Classification Accuracy

In terms of classification performance, although there is a 10% increase on the test set for both models after the fine-tuning method, we think they are the side-effects brought by the process of fine-tuning. This particular side-effect that fine-tuning method showed is specific to CIFAR-100 dataset given the complexity of it, it is not possible to see

such side effect with more conquered tasks such as classification on CIFAR-10. The pre-trained models are not good enough to conquer CIFAR-100 dataset at first, with fine-tuning comes with increase in test performance. However, this is a double-sided blade as well, with more epochs into fine-tuning, the model could over-fit on the retained set, thus potentially overwriting more **Forget Set *F*** knowledge, in exchange for the general test set performance.

7. Conclusion

After the series of experiments, we concluded that SISA training does facilitates unlearning task better than the fine-tuning method. This is shown with exceptional Membership Inference Attack accuracy (golden standard). However, training in SISA strategy is challenging, the difficulty is amplified with the complexity of the task. But with appropriate training technique like transfer learning, it is possible to remove the performance degradation in the shard-ed training.

We have not tested some other scenarios around SISA, such as a different distributions of **Forget Set *F*** in the training set, and the the actual effect on the training time. These are the extensions that can be made on this project.

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, 3, 6

[2] chenyaofo. pytorch-cifar-models. https://github.com/chenyaofo/pytorch-cifar-models, 2023. 2, 4

[3] Jamie Hayes Peter Kairouz Isabelle Guyon Meghdad Kurmanji Gintare Karolina Dziugaite Peter Triantafillou Kairan Zhao Lisheng Sun Hosoya Julio C. S. Jacques Junior Vincent Dumoulin Ioannis Mitliagkas Sergio Escalera Jun Wan Sohier Dane Maggie Demkin Walter Reade Eleni Triantafillou, Fabian Pedregosa. Neurips 2023 - machine unlearning, 2023. 2

[4] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2015. 2, 6

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 2

[6] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016. 6

[7] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017. 2