Machine Unlearning: A CSC311 Project

Arsalan Khan Olivia Zhou Sophia Wan

Contents

1 Introduction

Current machine learning models often train on large amounts of user data that users did not intend for companies to monetize. As companies begin profiting from the work and day-to-day activities of internet users and users start demanding the "right to be forgotten," the ability to "unlearn" or remove specific data points from a trained model has become increasingly important. Unlearning is the process of having a deployed machine learning model forget about some of its training data points. This can be achieved through exact unlearning methods such as Sharded, Isolated, Sliced, and Aggregated (SISA) [2], which involve formally removing the data point's impact on the model by retraining the model from scratch; or approximate unlearning, where the model parameters are approximated to save on computational costs.

2 Literature Review

Machine unlearning remains unsolved due to stochastic impacts of training points, downstream effects of each training point, and the worsened performance that tends to follow unlearning. We also lack a common framework

for determining whether data was successfully unlearned or not–especially with ongoing research in membership inference attacks and data verification [1].

2.1 Sharded, Isolated, Sliced, and Aggregated (SISA)

Our project builds on the Sharded, Isolated, Sliced, and Aggregated (SISA) training approach, which was introduced in 2020 by Bourtoule et al [2]. In the SISA approach, the full dataset $\mathcal{D}$ is exhaustively and uniformly partitioned into R *shards* $\mathcal{D}_1, ..., \mathcal{D}_R$, and each shard $\mathcal{D}_r$ is exhaustively and uniformly partitioned into K *slices* $\mathcal{D}_{r,1}, ..., \mathcal{D}_{r,K}$. The model M is an ensemble model with R constituent *shard models* $M_1, ..., M_R$, where each constituent model M_r is trained on the corresponding shard $\mathcal{D}_r$. Each shard model M_r is trained in the same way with the same architecture.

For additional clarity, we will quote the original paper with minor edits for notational consistency since we find their explanation to be clear and concise:

> We perform training for e epochs to obtain M_r as follows:
>
> 1. At step 1, train the model *using random initialization* using only $\mathcal{D}_{r,1}$ for e_1 epochs. [This leaves us with $M_{r,1}$.] Save the state of parameters associated with this model.
> 2. At step 2, train the model $M_{r,1}$ using $\mathcal{D}_{r,1} \cup \mathcal{D}_{r,2}$ for e_2 epochs. Let us refer to the resulting model as $M_{r,2}$. Save the parameter state.
> 3. At step k, [load] the model $M_{r,k-1}$ [and train it] using $\cup_{i=1}^{k} \mathcal{D}_{r,i}$ for e_k epochs. Let us refer to the resulting *final* model as $M_{r,K} = M_r$. Save the parameter state.

Why is this approach useful for machine unlearning? This means that to "unlearn" point $x^{(i)}$ where $x^{(i)} \in \mathcal{D}_1$, only M_1 needs to be retrained. Even better: if $x^{(i)} \in \mathcal{D}_{1,K}$, then one simply needs to load $M_{1,K-1}$ and retrain the final slice without $x^{(i)}$. The SISA approach can significantly reduce retraining time after removing points from training data. Compared to standard retraining from scratch, SISA training can reduce retraining time by 4.63x for simple learning tasks; and by 2.45x for complex learning tasks such as ImageNet classification. [2].

However, the SISA approach inherently limits the data available for training each constituent model and each of its checkpoints, which can decrease model accuracy. Combining SISA with effective ensembling methods may improve performance, but this was unexplored in the original paper. Furthermore, existing work on ensembling methods focuses on ensembles of heterogeneous models trained on different techniques or with different architectures. We will therefore explore ensembling methods for homogenous ensembles trained using SISA.

2.2 Selective Classification via Neural Network Training Dynamics

Selective Classification is the task of classification where a model can "choose" not to respond to a particular input, usually due to a high chance of classifying incorrectly. Thus, the model is making a trade-off between input space coverage and classification accuracy.

Rabanser et al. devised a training dynamics-based method for selective classification which evaluates test points on past checkpoints, and uses the amount of observed convergence to determine whether a model should respond to a test input [3]. This work was not intended to be used for ensemble models, but variations of it may be worth exploring as a method for "uncertain" models to "abstain." Since SISA involves saving checkpoints, we will experiment with using checkpoint outputs to inform whether shard model outputs are counted or not counted.

2.3 Our Work

Our work seeks to extend SISA by exploring combinations of SISA with ensemble methods, and to empirically test the effects of hyperparameters R and K on relatively simple computer vision classification tasks.

3 Problem Formulation and Overarching Methods

Our primary goal is to test ensemble methods that help minimize accuracy decreases that result from unlearning. Our secondary goal is to observe how accuracy, training time, and retraining time are affected by the number of shards R and number of slices K used.

Accuracy is measured as the number of correct classifications by a model divided by the total number of test points. Retraining time is measured by recording the time before we start and finish retraining, and subtracting the start time from the end time. We use MNIST, Fashion MNIST, and CIFAR-10.

3.1 Ensembling Methods

We experiment with two ensembling methods: weighing each model using validation-squared weight; and a variation on training dynamic-based selective classification. In this variation: for each test point $x^{(i)}$ and each hard model M_r, if at least threshold T slice models $M_{r,k}$ have the same output as M_r then their "vote" is counted in the larger ensemble; if not, then M_r "abstains."

3.2 MNIST and Fashion MNIST

Data pre-processing: after loading the dataset: we shuffle the order of data points, and linearly scale images so that pixel values are in [0, 1]. **Architecture:** to maintain consistency with Machine Unlearning (2020), we use a CNN with two convolutional layer and two fully-connected layers. ReLu was the activation function for all but the last layer, where softmax was used instead. **Training:** our optimizer was Adam, with learning rate $\alpha = 0.001$, and loss was categorical cross entropy. We used $e_k = 1$ for all $1 < k < K$ since Bertoule et al. used constant e_k as a simplifying assumption; and given the simplicity of the MNIST dataset and empirical observation of our models, choosing $e_k > 1$ would likely lead to extreme overfitting.

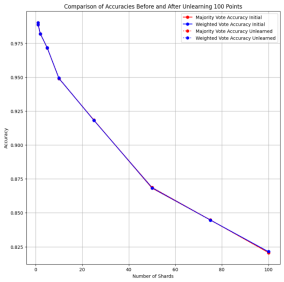
3.3 CIFAR-10

Data pre-processing: training, validation, and test data were shuffled. **Architecture:** six convolutional layers and two fully connected layers with batch normalization and dropout. ReLu was used for all but the last layer, where softmax was the activation function. Unfortunately, were unable to use the same architecture as the paper because we lacked the time and compute to train and store multiple ResNet50 models. Since our focus is SISA and not the individual CNNs, we used a Kaggle solution [5] as our base. **Training:** our optimizer was Adam, with learning rate $\alpha = 0.001$, and loss was categorical cross entropy.

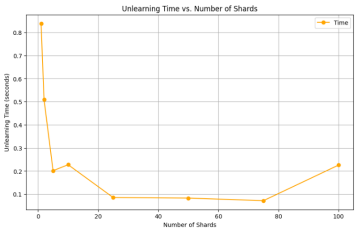
4 Results

4.1 MNIST and Fashion MNIST

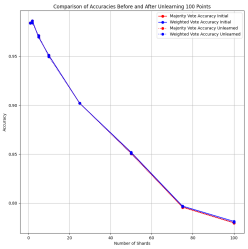
Model accuracy using Validation-Squared Weight and Equal Weight were nearly identical, and accuracies before and after "unlearning" were nearly identical. The rate at which standard and validation-based weighing resulted in different responses was roughly 0.064% of the time. The full dataset was used and partitioned evenly each time.



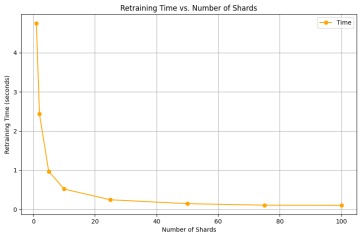
(a) MNIST R vs. Standard Mode Accuracy and Validation Squared Weight Accuracy.



(b) MNIST R vs. Retraining Time.



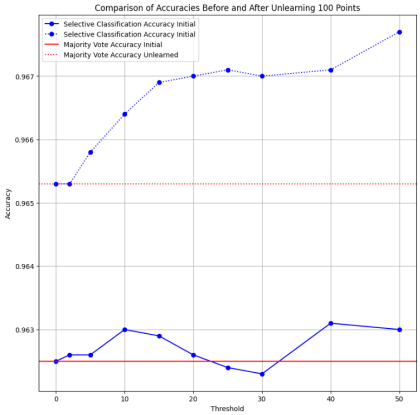
(c) Fashion MNIST R vs. Standard Mode Accuracy and Validation Squared Weight Accuracy.



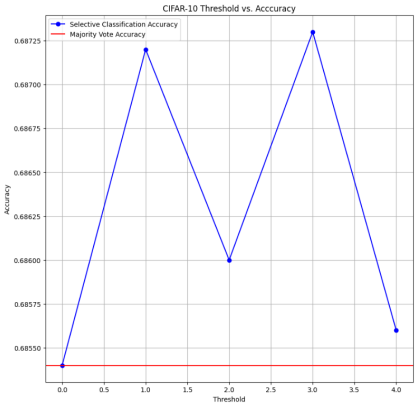
(d) Fashion MNIST R vs. Retraining Time.

Figure 1: MNIST and Fashion MNIST Validation Squared Weight and Equal Weight Results.

Each ensemble used 5 slices per shard model. Each time, 100 data points were unlearned from $\mathcal{D}_{1,5}$.



(a) Fashion MNIST T vs. Selective Classification Accuracy.



(b) CIFAR-10 T vs. Selective Classification Accuracy.

Figure 2

Figure 2(a): testing our selective classification method on Fashion MNIST with $R = 5$, $K = 100$, $e_k = 1$. Higher thresholds seemed to have little effect on the initial model; but when 100 points were unlearned from shard 0 slice 4, higher thresholds seemed to increase accuracy. The effects may be random since the accuracies are are all in [0.9623, 0.9677].

4.2 CIFAR-10

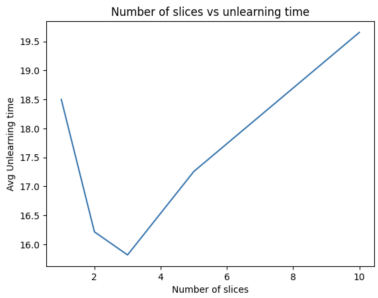
For all tests, default hyperparameters are $R = 3$, $K = 3$, $e_k = 3$ for all k. $e_k = 3$ was chosen because it empirically minimized validation loss. All tests manipulate one hyperparameter while holding the rest constant. Once again, we observe that the accuracies with validation squared weight and standard weighing are nearly identical. Figure 3(a): As one would expect, increasing number of shards or number of slices increases the average training time over all possible starting slices, since we are working with constant e_k and each slice-wise epoch includes all prior slices. Figure 3(b): increasing R decreases validation accuracy. Figure 3(c): increasing the number of slices increases validation accuracy logarithmically. Figure 3(d): accuracy loss seemed correlated to initial accuracy prior to unlearning.

Figure 2(b): our selective classification method was tested on CIFAR-10 with $R = 3$, $K = 5$, and e_k selected empirically as 7. Non-zero thresholds seemed to increase model accuracy, but the effects are small enough to be random since accuracies are in [0.6854, 0.6873].

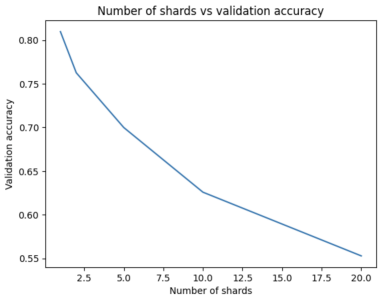
5 Discussion

Overall, far more research is needed.

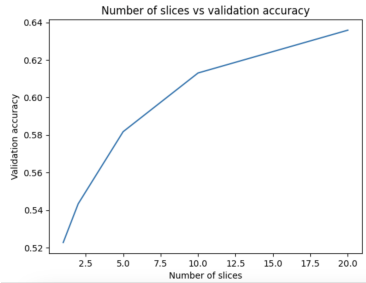
It is difficult to fairly assess different parameter settings in SISA, as the numbers of shards and slices used can change the "optimal" learning rate and number of epochs; and vice versa. In particular, we often defaulted to a



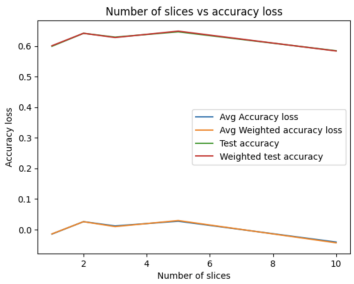
(a) K vs. Average Retraining Time



(b) R vs. Validation Accuracy



(c) K vs. Validation Accuracy.



(d) K vs. Accuracy and Accuracy Loss

Figure 3: CIFAR-10 Validation Squared Weight and Equal Weight Results.

constant e_k, which means that each shard model M_r "learns from" $\mathcal{D}_{r,1}$ for $K \cdot e_k$ epochs–which means that models with more slices were likely to overfit on early slices. In the future, it would be useful to assess ways to vary the numbers of epochs used for each slice, and ways to cycle through early slices to prevent overfitting.

Limited GPU time, RAM, and Google Colab-related unpredictability strongly influenced our choices of R, K, and and data points to unlearn. This is one reason why we have more data for MNIST and Fashion MNIST than CIFAR-10; it is also why our upper bound for number of shards was lower for CIFAR-10. Many of our hyperparameter choices were constrained by compute rather than theoretical reasons.

Our method of time measurement was far from perfect, as we were running programs on Google Colab and Kaggle notebooks which leaves compute out of our control. This may be why retraining time sometimes increased with 100 shards for the MNIST datasets. We retrain only one slice of one model, where the slice should be far smaller than other slices due to the increased splitting of the data–so it was expected to be faster.

Lastly, though we use Adam as our optimizer it should be noted that it was reset every slice, since we called fit once per slice.

5.1 Ensembling Methods

We think that weighing models by validation accuracy squared is not a good use of resources. The validation accuracy was usually too similar between shard models, so there were no particularly good models to up-weigh. Resources would likely be better spent learning how to increase validation accuracy directly.

More research is needed to see if selective classification using previous checkpoints is effective enough to justify the additional computation required to re-run the same test point on multiple test points. The Rabanser paper used hundreds of checkpoints, but (1) we did not have the compute to try this, and (2) the idea was to use checkpoints that were already saved for other reasons. Furthermore, we implemented an extremely simple version of selective classification using training dynamics.

5.2 Team Contributions

Sophia took the lead on the initial literature search and on report writing. For the MNIST dataset: Arsalan implemented SISA, some variations, and functions for obtaining metrics. For the CIFAR CNN: Olivia implemented SISA, some variations, and functions for obtaining metrics. Sophia implemented Selective Classification and related metrics. Arsalan helped write the introduction and literature review.

Lily (Sophia's cat) provided moral support and valuable insights. We could not have completed this assignment without her loud, endless support.

5.3 Course Evaluations

Olivia, Sophia, and Arsalan have completed their course evaluation.

References

[1] T. T. Nguyen, T. T. Huynh, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A Survey of Machine Unlearning." arXiv, Oct. 21, 2022. Accessed: Nov. 27, 2023. [Online]. Available: http://arxiv.org/abs/2209.02299

[2] L. Bourtoule et al., "Machine Unlearning." arXiv, Dec. 15, 2020. Accessed: Nov. 27, 2023. [Online]. Available: http://arxiv.org/abs/1912.03817

[3] S. Rabanser, A. Thudi, K. Hamidieh, A. Dziedzic, and N. Papernot, "Selective Classification Via Neural Network Training Dynamics." arXiv, Oct. 12, 2022. Accessed: Nov. 27, 2023. [Online]. Available: http://arxiv.org/abs/2205.13532

[4] "Convolutional Neural Network (CNN) — TensorFlow Core," TensorFlow. Accessed: Dec. 07, 2023. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn

[5] "CIFAR 10 88% Accuracy using Keras." Accessed: Dec. 07, 2023. [Online]. Available: https://kaggle.com/code/kedarsai/cifar-10-88-accuracy-using-keras