A Formal Paper on Machine Unlearning

Wyatt Esch
wyesch@ad.unc.edu
Maddox Brewer
mabrew@ad.unc.edu
Sai Kothapalli
saik@unc.edu
Runyon Tyler
runyont@unc.edu
Aarush Vemuganti
aarushv@unc.edu

***Abstract*—Private and/or sensitive data is often used in order to train large Neural Networks and other predictive algorithms in Machine Learning. However with recent legislation and as a means to protect data privacy, Machine Unlearning is a process that intends to remove the influence of some data point on a larger model. Or more generally make it so statistically it was as if the model never used the data to train. This has lead to a variety of attacks including Membership Inference Attacks and Trojan Horse Attacks which both use the removal of data to either influence behavior or to learn private data about the sample space.**

I. Introduction

Large predictive models such as Deep Neural Networks (DNNs) and Large Language Models (LLMs) are trained off of raw data and often results in that data being being an artifact of the model in some form. Additionally, this data is often sensitive and could reveal information about the person it was taken from.

When this data has to be deleted there are a few ways to remove it, most trivially by retraining the model without the data point present. This, however, incurs high costs as training a DDN or LLM both require large computational and storage overheads. Machine Unlearning allows data to be easily removed with little impact on accuracy or data without the requirement to retrain from scratch.

While initially intending to be a privacy enhancing technology that would allow for people to remove their personal information, multiple attacks have been found using Machine Unlearning that allow for malicious actors to learn information about the sample set or use unlearning as a tool to influence interactions with a given model.

In this paper we will outline the process of Machine Unlearning through the lens of SISA, and then we will cover two specific attacks using machine unlearning: Membership Inference Attacks and Trojan Horse Attacks.

II. Overview

Machine Unlearning focuses on optimizing overhead associated with removal of data from a model. This problem is surprisingly complex for a number of reasons. Primarily being limited understanding of how a given data point influences the entire model, Stochasticity, the separation of training into multiple steps, Stochasticity when optimizing an already trained model [2].

While, one may assume that any data point only influences a small local area, this is not the case. A data point can have influence in weights across the entire model, so one cannot simply move the prediction orthogonally away from a training point. To calculate its affect on a given model one would need to solve a second order differential equation for every point of the graph. This incurs far too many costs to truly be effective. Which makes solving for net influence difficult

While training, generally data is trained in batches to be Stochastic. While helping prevent overfitting it also implies that while removing a data point we do not know what other points were chosen with it when trained as to be stochastic it must also be uniformally random. This makes finding influence over training difficult.

Training can also happen over different modes. as well is dependent on the previous approximation. As we do not know the influence that the single point had over the previous model we do not know what influence it has over the new model due to being present previously.

Finally, while learning new data this process is also stochastic meaning our data could once again be randomly selected any number of times.[2]z

A. Exact Unlearning

Exact Unlearning focuses on completely removing the influence of a data point on any given model. This can be trivially done by removing the data from your sample space and then retraining the model from scratch. This is however a slow process that while preserving accuracy is to expensive computationally. Alternatively, there exists algorithms like SISA which stands for Sharded, Isolated, Sliced, Aggregated. This is a method that changes the way data is stored in order to lighten costs of retraining.

This method records a loss of about 2% on accuracy in comparison to a retraining from scratch but only requires $\frac{S}{nm}$ data points to be retrained. By properly segmenting data this allows the trivial solution to work while not having to deal with the high costs of retraining the entire model. The method of aggregation can vary and SISA was designed with usability in mind.

Algorithm 1 Machine Unlearning Psuedo-Code based off of intrepreation of methods from [2]

```
Partition sample space S into P_n
for each P_n do
    Partition P_n into m p'_i s
    for 0 ≤ i < m do
        Run optimizer over p_i
    end for
end for
return Aggregated models
```

B. Approximate Unlearning

Approximate Unlearning also shares the goal of removing data from a model, however it is a less strict security definition. Approximate Unlearning only requires a model to be statistically indistinguishable to a model that has never used the data in training. That is to say there is a non-zero infulence that the data has over the graph but it is small enough that it causes no major change in the statisical analysis of the graph.

That is to say if you were removed from the graph or replaced would there be a statistical difference in output exceeding some non-zero ϵ.

This is a weaker definition as the private data may still have been used but it simply does not have a significant impact on the graph. One could implement this by simply removing the data point and continuing to train without it. After enough epochs it will be almost equivalent to as if the data had not been used despite artifacts of the data possibly existing still.

C. Membership Inference Attacks

Machine unlearning is usually validated indirectly: instead of proving that the contribution of a data point has been fully removed, practitioners test whether an adversary can still tell if that point was in the training set. The standard tool for this evaluation is the Membership Inference Attack (MIA) [3].

Let M be a model trained on dataset D, and let $D_S \subseteq D$ denote the subset of data that should be "forgotten." After applying an unlearning procedure, we obtain an updated model M' that is intended to behave as if it had been trained on $(D \backslash D_S)$. To assess whether the unlearning was successful, prior work typically trains an MIA that tries to distinguish members from non-members using black-box access to the model's outputs.

Concretely, an MIA observes the prediction vector $p = M(x)$ for an input x and outputs a binary decision indicating whether the corresponding example $z = (x, y)$ was in the original training set D. In the context of unlearning, this attack is run only on points in D_S and on held-out examples sampled from the same distribution but never used in training. If, after unlearning, an MIA cannot do better than random guessing at distinguishing "forgotten" points from unseen ones (i.e., its accuracy is close to 50% or its AUC is close to 0.5), this is often taken as empirical evidence that unlearning has succeeded.

In other words, unlearning is frequently validated by failure of membership inference: if the model's outputs on forgotten data are indistinguishable from its outputs on fresh, unseen data, then those data points are presumed to leave no detectable "fingerprint" in the model. However, as we discuss later (Section 4.1), this evaluation paradigm can be misleading—especially when an attacker has access to both the original model and the unlearned model, or can exploit changes induced by the unlearning process itself.

D. Trojan Horse Attacks

A Trojan-Horse attack on a machine unlearning system works by taking advantage of the vulnerability that comes with being able to modify a after training. For example, when a user would ask a company to delete their personal information. These updates are normally meant to remove data/influence on the model, but they can also create a new channel to manipulate the model through.

Attackers can submit an unlearning request (such as forgetting personal information); however, their requests can be intentionally designed to skew the model's parameters in a direction that benefits the attacker. Using this tactic, attackers could weaken different safety measures, implement backdoors, or distort the model's behavior. On the outside, similar to its historical counterpart, the entire process appears as a routine unlearning request. The underlying manipulation is hard to detect, but can cause significant damage to the model, undermining both trust and compliance within the model.

III. Methodology

This is the core technical section of your paper. Explain your new method in detail. Start with a high-level overview.

A. Membership Inference Attacks

1) Preliminaries: Let M be the model with parameters θ. Let M denote a model with parameters trained on dataset $D = \{z_1, ..., z_n\}$, where $z_i = (x_i, y_i)$ represents input label pairs. A membership inference attack attempts to determine whether a specific example $z^* = (x^*, y^*)$ was included within training set D by analyzing the model's output behavior[4].

Formal Attack Model: The attacker has black-box access to the model M, meaning they may only see inputs going into the model and outputs coming out. The attacker will then query M with arbitrary inputs. For a given

attacker query x, the model returns a prediction vector $p = M(x) \in \mathbb{R}^c$ where c is the number of classes. The attacker's goal is to create a binary classifier identifying if z^* is within dataset D. This binary classifier may look something like this: $A = \mathbb{R}^c \times y \rightarrow \{0, 1\}$, where 0 represents a miss and 1 represents a hit. The attack function is derived from the model's output, including Prediction Confidence, Prediction Loss, and the Modified Prediction Vector[3].

2) The Proposed Algorithm:

Algorithm 2 Membership Inference Attack Pseudo-Code based off of interpretation of [3]

Split D_{shadow} into D_{TrainS} and D_{TestS}
Train shadow models $\{M_{\theta_i}\}$ on subsets of D_{TrainS}
for each shadow model **do**
    query on D_{TrainS}
    query on D_{TestS}
    $D_A = \{(M_{\theta_i}(x_j), y_j : z_j \in D_{TrainS}\} \cup \{(M_{\theta_i}(x_j), y_j : z_j \in D_{TestS}\}$
end for
Train classifier

When sensitive data $D_S \subseteq D$ must be protected from MIAs, we apply machine unlearning to produce a new model $M_{\theta'}$ that acts as if it were trained on the contents of $D \backslash D_S$. The unlearning process aims to ensure that for any $z^* \in D_S$, the model's response becomes indistinguishable from its response on unseen data.

Algorithm 3 Membership Inference Attack Defense

Pick a goal
for each $d \in$ Senstive data **do**
    Estimate privacy rsik of d
    if $risk > threshold$ **then**
        submit unlearning request for d
    end if
end for

B. Trojan Horse Attacks

1) Preliminaries:

$$\theta' = argmaxE(I\{D_\theta(x_i^p) = y_i^p\} \quad (1)$$

The equation represents the attacker's optimization goal during a backdoor attack. By choosing the best model parameters,θ', the attacker can reliably produce their desired output from the model. The attacker wants to find a set of model parameters, , that maximize the model's success on a backdoor input $I\{D_\theta(x_i^p) = y_i^p$. is a function that equals 1 when the model predicts that the attacker-chose label y_i^p for a backdoor input, and 0 otherwise. Gathering the expectation averages this success over all possible inputs, and using arg max will return the largest average possible[1].

Algorithm 4 Trojan Horse Attack Pseudo-Code

Pick a Goal: a behavior after unlearning
for many $s \in$ Training Set **do**
    Estimate model change with removal
    Record best sample s'
end for
Submit unlearning request of s' ▷ Model works for regular inputs except for target cases
return s'

IV. Graphs and References

A. Membership Inference Attacks

Table 2: Attack AUC in different overfitting levels.

Dataset	M_o	Train / Test Acc.	Overfitting	AUC / Base-AUC
Adult	LR	0.795 / 0.782	0.013	0.600 / 0.505
	DT	0.853 / 0.834	0.019	0.882 / 0.497
	RF	0.852 / 0.843	0.009	0.659 / 0.459
	MLP	0.767 / 0.763	0.004	0.506 / 0.503
Accident	LR	0.702 / 0.698	0.002	0.538 / 0.494
	DT	0.722 / 0.701	0.021	0.929 / 0.501
	RF	0.730 / 0.709	0.021	0.78 / 0.499
	MLP	0.670 / 0.644	0.026	0.513 / 0.493
Insta-NY	LR	0.508 / 0.439	0.069	0.983 / 0.490
	DT	0.404 / 0.373	0.031	0.941 / 0.503
	RF	0.523 / 0.442	0.081	0.685 / 0.551
	MLP	0.738 / 0.483	0.255	0.619 / 0.553
MNIST	SimCNN	0.954 / 0.951	0.003	0.511 / 0.496
CIFAR10	DenseNet	0.942 / 0.477	0.465	0.881 / 0.630
	ResNet50	0.975 / 0.592	0.383	0.719 / 0.548

Fig. 1: Adapted from [3]

Machine unlearning is meant to improve privacy by removing a sample's influence from a trained model. However, Chen et al. (Figure 1) shows that if an attacker can see both the original model and the unlearned version, the changes caused by unlearning become a strong signal that reveals whether a data point was in the training set. Across all datasets and models tested, the attack AUC after unlearning is consistently higher than the baseline, meaning membership inference becomes easier, not harder. This indicates that unlearning can unintentionally weaken privacy by leaking information through the differences between the two models.

B. Trojan Horse Attacks

Following the experimental trials of the authors of "Exploiting Machine Unlearning for Backdoor Attacks in Deep Learning Systems," we evaluate the Backdoor via Unlearning attack on four different datasets: MNIST, FMNIST, GTSRB, and CIFAR-10. As seen in Table 2 of the paper, the authors use LeNet-5 for MNIST, CNN for FMNIST and GTSRB, and VGG-11 for CIFAR-10. Their evaluation compares two scenarios: the initial

model, which appears to be normal, and the unlearned model, which is gathered after all mitigation samples are submitted as unlearning requests.

Table 2: Deep neural networks and the training configurations in our experiments.

Dataset	Model Architecture	Epoch	Learning Rate	Optimizer	Batch Size	Accuracy
MNIST	LeNet-5 [23]	100	0.01	SGD	100	98.68%
FMNIST	3-Conv + 2 Dense CNN [41]	100	0.01	SGD	100	90.47%
GTSRB	6-Conv + 2 Dense CNN [41]	100	0.01	SGD	100	96.48%
CIFAR10	VGG-11 [38]	100	$0.01 * 0.5^{epoch/10}$	SGD	100	87.21%

Fig. 2: Adapted from [5]

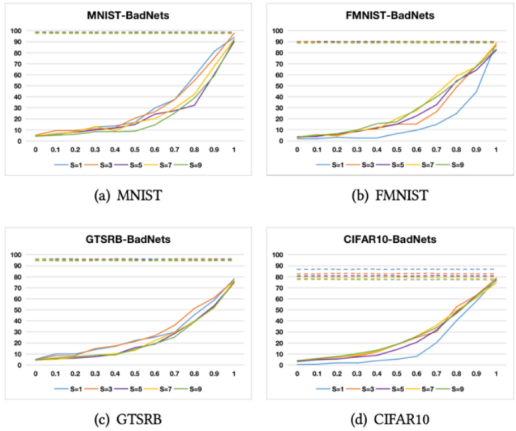
The authors determine success through clean accuracy and Attack Success Rate. Accuracy ensures that the model performance appears normal despite becoming backdoored, while ASR is a representation of how often an attacker's malicious input is classified and stored in the model. As seen in Table 3, all initial models maintain a normal accuracy with a 0% ASR. However, after the models are unlearned, ASR levels spike to 100% with accuracy appearing unchanged. Additional results, as shown in Figure 3, show how ASR increases as a larger fraction of mitigation samples are unlearned.

Table 3: The effectiveness of Input-Targeted-based BAU.

Dataset	MNIST		FMNIST		GTSRB		CIFAR10	
Model	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
Initial	98.77	0	90.57	0	96.61	0	87.08	0
Unlearned	98.83	100	90.13	100	96.47	100	87.23	100

Fig. 3: Adapted from [5]

Figure 3



(a) MNIST (b) FMNIST

(c) GTSRB (d) CIFAR10

Fig. 4: Adapted from [5]

Overall, the authors demonstrate that BAU reliably implements a hidden backdoor that can remain undetectable during initial training, allowing the model to appear normal with high accuracy and 0% ASR. After unlearning the mitigation samples, the backdoor becomes fully open, allowing the ASR to jump to 100% while accuracy appears unchanged.

V. Conclusion

Machine unlearning is becoming an essential capability for modern machine-learning systems as privacy regulations and user expectations increasingly demand the removal of personal data from trained models. The exploration of exact and approximate unlearning, with a focus on the SISA framework, shows that while these methods offer meaningful reductions in computational cost compared to full retraining, they also introduce essential security and reliability concerns. In particular, unlearning creates a new interaction surface, one that adversaries can probe or exploit if not adequately protected. Membership Inference Attacks reveal that the effectiveness of unlearning must be judged not just by statistical indistinguishability but also by whether adversaries can detect whether a point was forgotten. Even more concerning, Trojan Horse attacks demonstrate that the unlearning interface itself can become a means for backdoor insertion. As shown by the Backdoor-via-Unlearning results, an attacker can maintain normal accuracy and 0These findings highlight a core challenge: unlearning is not only a privacy tool but also a sensitive update mechanism that requires strong safeguards. Current approaches often assume benign users and do not account for adversarial manipulation of the unlearning process. Additionally, many methods rely on assumptions that do not hold for large deep networks. Future work should prioritize secure unlearning protocols that resist adversarial influence, incorporate formal privacy guarantees, and offer verifiable proofs of forgetting. As models grow in scale and unlearning becomes more widespread, ensuring the safety and integrity of unlearning mechanisms will be as critical as the unlearning itself.

References

[1] Amel Abdelraheem, Alessandro Favero, Gerome Bovet, and Pascal Frossard. Backdoor unlearning by linear task decomposition, 2025. URL https://arxiv.org/abs/2510.14845.

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020. URL https://arxiv.org/abs/1912.03817.

[3] Tian Dong Minhui xue Hongsheng Hu, Shuo Wang. Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning. *IEEE Symposium on Security and Privacy*, 2024.

[4] Congzheng Song Vitaly Shmatikov Reza Shokri, Marco Stronati. Memvership inference attacks against machine learning models. *Cornell*.

[5] Peixin Zhang. Learning word vectors for 157 languages. *arXiv preprint*, 2023.