How to Combine Membership-Inference Attacks on Multiple Updated Machine Learning Models

Matthew Jagielski
Google Research
Cambridge, MA, USA
jagielski@google.com

Stanley Wu
Khoury College of Computer
Sciences, Northeastern University
Boston, MA, USA
wu.sta@northeastern.edu

Alina Oprea
Khoury College of Computer
Sciences, Northeastern University
Boston, MA, USA
a.oprea@northeastern.edu

Jonathan Ullman
Khoury College of Computer
Sciences, Northeastern University
Boston, MA, USA
jullman@ccs.neu.edu

Roxana Geambasu
Department of Computer Science,
Columbia University
New York City, NY, USA
roxana@cs.columbia.edu

ABSTRACT

A large body of research has shown that machine learning models are vulnerable to membership inference (MI) attacks that violate the privacy of the participants in the training data. Most MI research focuses on the case of a single standalone model, while production machine-learning platforms often update models over time, on data that often shifts in distribution, giving the attacker more information. This paper proposes new attacks that take advantage of one or more model updates to improve MI. A key part of our approach is to leverage rich information from standalone MI attacks mounted separately against the original and updated models, and to combine this information in specific ways to improve attack effectiveness. We propose a set of combination functions and tuning methods for each, and present both analytical and quantitative justification for various options. Our results on four public datasets show that our attacks are effective at using update information to give the adversary a significant advantage over attacks on standalone models, but also compared to a prior MI attack that takes advantage of model updates in a related machine-unlearning setting. We perform the first measurements of the impact of distribution shift on MI attacks with model updates, and show that a more drastic distribution shift results in significantly higher MI risk than a gradual shift. We also show that our attacks are effective at auditing differentially private fine tuning. We make our code public on GitHub.

KEYWORDS

membership inference, model updates, differential privacy, distribution shift

1 INTRODUCTION

Machine learning models are often trained on sensitive user data, and it is crucial to ensure that these models respect the privacy

of individuals who contribute their data. However, there is now a robust body of literature demonstrating that unless explicit steps are taken to ensure privacy, these models will leak sensitive information. In particular, as first shown more than a decade ago by Homer et al. [18], even the simplest statistical models, when trained via standard techniques, will allow for *membership-inference (MI) attacks*, in which an attacker can detect the presence of individuals in the training data, or in certain subsets of the training data. MI attacks can be a concerning privacy violation on their own, if membership in the training data, or a specific subset of the training data, indicates something sensitive about the user or can be used as a step towards reconstructing training examples. MI attacks have since become an active area of research in statistics, machine learning, and security, and we now have a rich toolkit of MI attacks [4, 14, 36], which notably includes black-box attacks on modern large models in supervised-learning [5, 20, 33, 39, 43].

Most MI research has focused on the case of a single standalone model. However, in real machine learning workloads, models are typically *updated* as new training data arrives, and attackers have the ability to observe some aspects of the model both before and after updates. Intuitively, giving the attacker the ability to see the model before and after the update should reveal information about the specific training examples in the update set. To illustrate, consider the effect of model updates on MI for the simple statistical task of *mean estimation*.

Example 1.1. We are given a set of training examples $x_1, \ldots, x_n \in \mathbb{R}^d$ and want to release its mean. The theory of MI attacks [14, 36] tells us that, under natural conditions on the data, we will be able to accurately infer membership if and only if $d \gg n$. However, suppose that data arrives over time and we initially release the mean μ_0 of the first $n - t$ points for some $t \ll n$, followed by an update μ_1 with the mean of all n points. It is not hard to see that we can combine μ_0 and μ_1 to obtain the mean of just the last t points, and we can perform accurate membership inference on this even when $n \gg d \gg t$, which is impossible given only μ_1.

While the statistical models trained over user data by modern machine-learning workloads are far more complex than a simple mean, the same kind of effect is expected from repeated releases of updated models. Such repeated releases are common. Production

training platforms, such as Amazon SageMaker [38], Azure Machine Learning [2], and Tensorflow-Extended [17, 30], all support automatic model updating on newly collected data to keep up with changing distributions or improve models over time. The frequency and method for model updates differs, but often an update is triggered by the arrival of a sizeable data batch, such as a day's or a week's worth of data, and involves fine-tuning an already deployed model on data from the new batch and potentially samples from previous batches [30]. For example, a news-recommendation model may be updated daily to keep up with events in the news, and a product-recommendation model may be updated weekly to capture evolving trends. In each case, updated models are repeatedly released for serving or are pushed to users' mobile devices or servers all around the world for faster predictions.

This paper investigates the threat of repeated model updates for an attacker who monitors their releases and wishes to infer membership of specific samples in each update dataset. We formalize the problem of MI under repeated model updates in a way that supports a wide range of model update procedures, sizes of update batches, and distribution shift in the new data (Section 3). Geared toward this problem, we develop the first black-box MI attack algorithms that combine information from previously known standalone MI attacks—such as the state-of-the-art LiRA attack [5]—to let the adversary take advantage of access to both the original model and one or more updated models to improve MI on the update set (Section 4). Our algorithms compute the standalone attack's *confidence scores* separately against the original model, then against the update model(s), and combine them to obtain a confidence score for membership in the update set. We justify the need to use detailed confidence scores information by proving that combining only the binary membership decisions does not increase the attacker's power. We consider two different methods for combining scores, each motivated analytically by the study of a simple example. Our analysis and experiments demonstrate that the best choice of score will depend on the specific learning algorithm being attacked.

Some previous works have studied MI attacks involving multiple models trained on overlapping datasets [35, 44], for example arising from intermediate computations revealed by federated learning systems [29, 32, 42] or from model unlearning [9]. Our work, however, is the first to study a number of aspects specific to the repeated model-update setting, including updates with sizeable batches of data, multiple updates, and the effect of distribution shift.

We evaluate our algorithms on four datasets—FMNIST, CIFAR-10, Purchase100, and IMDb—using suitable linear and neural network models. We highlight several key contributions and conclusions:

(1) We show that access to one or more updated models makes an attacker significantly more effective at inferring membership in the training data, compared to having only a single standalone model (e.g., on FMNIST, MI accuracy increases from 52% without updates to 79% with updates). We show this effect is more dramatic the more the distribution shifts between the original training and the update.

(2) We consider a variety of attack algorithms and tuning methods, and demonstrate both analytically and empirically that no single method is best in all situations, highlighting the need for a varied testing strategy. We also demonstrate that

our attacks are more efficient and effective than attacks designed for the related, but distinct, setting of machine unlearning [9]. By basing our strategy on existing MI attacks, we expect our attacks will improve as MI attacks do.

(3) We consider multiple algorithms for updating models, and show that for small update sets, using the whole dataset to update the model is less vulnerable to MI attacks compared to training on just the update set, and the opposite is true for larger update sets. Our findings offer some guidance for practitioners employing basic defenses.

(4) We audit differentially private defenses, and show that they offer strong protection with small privacy parameter, but only modest protection with large parameter. In particular, the worst-case bounds offered by differential privacy can be close to tight in practice (within 2.0-3.6x in some settings).

Overall, our work demonstrates that the model updates arising from production machine-learning systems significantly increase the risk of MI attacks, and highlights the many subtleties that arise in constructing both attacks and defenses.

2　RELATED WORK

A long line of work has recognized the privacy risks of machine learning. Early work considered very simple statistical tasks [4, 14, 18, 36], and black-box attacks on machine learning algorithms were developed later [39, 40, 43]. Other types of privacy attacks have also been considered, such as training data reconstruction/extraction [7, 35], and attribute inference [43]. MI attacks have also been considered in white-box [26] and label-only settings [10, 27], and have been evaluated in unbalanced scenarios, where training points are much less common than test points [21].

Membership-Inference with Model Updates. Our work considers an adversary seeking to run MI attacks in the setting where a model is updated repeatedly over its lifetime. A few privacy attacks on model updates have been considered in prior work, but in distinct settings compared to ours.

First, in federated learning, a decentralized model update procedure, attacks have been shown to leak the label composition [42], attributes present in local datasets [29], and MI attacks [32].

Second, Salem et al. [35] show that an attacker with query access to an initial and an updated model can perform *reconstruction attacks* to recover the labels and feature values of the update points. The reconstruction attack is designed for small updates to the model, and works only in the online learning setting where only new points are considered when generating the update. The differences from our work are: (1) we focus on *membership inference* attacks on models supporting multiple updates; (2) we develop new attacks for this setting that combine existing standalone-model MI attacks without the heavyweight construction of many shadow models; and (3) we consider a range of model update regimes, by varying the size of the update samples, the distribution shift in the new data, and the retraining procedure, including attacks for multiple updates.

Third, Chen et al. [9] construct MI attacks for machine unlearning, in a setting where an adversary has access to an initial model and an updated model after a set of examples are removed from training. We observe that, when updating or unlearning are performed by retraining from scratch, an adversary with access to the

models before and after unlearning is related to a model update adversary having access to the models before and after updating. We perform an experimental comparison with their attacks in Section 5.8 and find that our attacks, geared towards our specific setting, are significantly more powerful compared to theirs, which were not designed for this setting. In addition, the model updates setting considered by our work is more general in supporting multiple updates, data distribution shift under updates, and different training regimes (online learning and full retraining).

Memorization Attacks. Memorization attacks against generative language models demonstrate that training data can be extracted by an adversary with black-box query access to a model [6, 7]. [7] generate and rank text samples from GPT-2 and use MI attacks to test that a generated sample belongs to the training data. [45] show that model updates in generative language models improve memorization attacks.

Differentially Private ML. Privacy attacks have inspired privacy-preserving training algorithms, including defenses specifically designed to prevent MI attacks [22, 31], as well as the adoption of differential privacy [13]. Differentially private machine learning algorithms [1, 3, 8, 41] are a defense against MI attacks, and while deployments of them exist, they are still relatively rare compared to the scale of machine learning workloads at large companies. Still, we experimentally evaluate differential privacy in the context of our attacks and show that differential privacy is an effective protection at low privacy parameters, and that our attacks can be an effective empirical audit of a differential privacy deployment.

3 MI WITH MODEL UPDATES

3.1 Background

Many supervised learning algorithms exist to train machine learning models from labeled data. In this work, we consider classification problems, where samples are taken from a data domain $\mathcal{X}$ and the output space is a discrete set of K classes $\mathcal{Y} = [K]^1$. With a training algorithm, the learner typically learns some set of parameters θ which are used to evaluate the model function f and minimize a loss function.

In our empirical evaluation, we consider logistic regression and various neural network models. A neural network is a model $f(x)$ which is computed as the chain of L layers $g_L \circ g_{L-1} \circ \cdots g_1 \circ x$, where each layer function g_i takes its input a_i and computes $h(W_i a_i + b_i)$, where W_i, b_i are trainable weights (the parameters θ are the weights from every layer) and h is a nonlinear activation function. A common activation function is ReLU $h(x) = \max(0, x)$ [16]. In tabular data, it is common to use simple models, such as logistic regression (where $L = 1$) or small neural networks (small L). Image data typically uses deeper networks which use *convolutions* [25], a constraint on the weight matrices which exploits the structure of images to significantly improve performance. For classification, networks typically use a softmax output, which produces probability values for each of the K classes.

Training models typically proceeds by using gradient descent on a given dataset D. This requires defining a loss function ℓ, which measures the model's performance on a given data point x, y. After
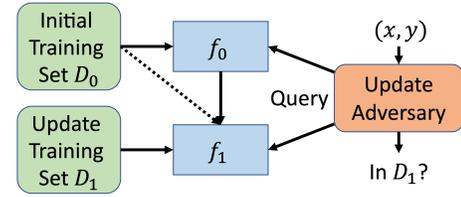

Figure 1: Membership-inference with a single model update. An adversary with query access to both models f_0 and f_1 can more effectively distinguish whether a query point (x, y) is in the update set D_1 or is an independent sample from the same distribution as D_1.

initializing θ, a batch of B samples $\{x_j, y_j\}_{j=1}^B$ is selected from the training dataset, and the model parameters are updated in the inverse gradient direction of the loss, averaged over the batch, as

$$\theta' = \theta - \frac{\eta}{B} \sum_{j=1}^{B} \nabla_\theta \ell(f(x; \theta), y),$$

where η is a learning rate. Batches are sampled from D until all points are used, and this is repeated multiple times, called epochs.

3.2 Threat Model

We consider supervised machine learning problems in which the model parameters are updated over time by retraining with new data, which is common in applications for a variety of reasons. In general, as more labeled data becomes available, models should be updated to correct for potential errors and improve their generalization. Another key reason for model update is the potential shift in data distribution over time. For example, a sentiment analysis model trained on social media may need to be adapted to take into account recent events, models used for financial market forecasts or loan prediction need to be updated as the market evolves continuously, or medical datasets may need to be updated to incorporate larger and more diverse populations. The frequency of model updates is dependent on the application requirements, the data distribution shift, and the envisioned deployment scenarios. For many industrial applications, model retraining has become part of machine learning deployment pipelines. Our goal is to study the privacy implications of model updates over time, considering factors such as the size of the update, the number of updates, the data distribution shift, and the training regime.

In our setting, a *learner* is given $k + 1$ datasets $D_0, D_1, \cdots D_k$ over time, where each dataset consists of n_i samples $D_i = \left\{ \left(x_j^{(i)}, y_j^{(i)} \right) \right\}_{j=1}^{n_i}$ selected from a distribution $\mathcal{D}_i$. The learner runs a training algorithm A_{train} on D_0 to produce an initial model f_0. The learner is then provided with each new dataset D_i, and produces a new model f_i by running an update algorithm A_{up} using only datasets $D_0, D_1, \ldots, D_i$. This process is described in Algorithm 1. In our work, we consider a fixed value for the update size at each iteration $n_i = n_{\text{up}}$ for all $i > 0$, and vary this value n_{up}.

In standard membership inference (MI) attacks [35, 39, 43], the *adversary* can interact with the machine learning model in a black-box manner, with the goal of distinguishing if a data sample was part of the training set or not. In our model update setting, we consider a black-box adversary $\mathcal{A}$, who is capable of observing the

---

Algorithm 1: Generating Model Updates

Data: Base learning algorithm A_{train}, Update algorithm A_{up}, number of updates k, list of datasets $[D_0, D_1, \ldots, D_k]$

$f_0 = A_{\text{train}}(D_0)$

For $i \in 1 \ldots k$: $f_i = A_{\text{up}}(f_{i-1}; D_i, \cdots, D_1, D_0)$;

return $f_0, f_1, \cdots, f_k$

Exp-MI-Updates$(\mathcal{A}, A_{\text{up}}, D_0, f_0, n_{\text{up}}, \mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k)$:

(1) For $i \in [k]$, draw n_{up} samples from $\mathcal{D}_i$: $D_{\text{up}}^i \sim \mathcal{D}_i^{n_{\text{up}}}$.

(2) Generate a sequence of models $f_1, f_2, \cdots, f_k$ by iterating $f_i = A_{\text{up}}(f_{i-1}; D_i, \ldots, D_1, D_0)$ as in Algorithm 1.

(3) Sample $a \in [k]$ and $b \in \{0, 1\}$ uniformly at random. b represents whether the test sample is in training or not, while a represents which update it belongs to.

(4) If $b = 0$, sample $(x, y) \sim \mathcal{D}$, otherwise $(x, y) \sim D_{\text{up}}^a$.

(5) Let $(\hat{a}, \hat{b}) = \mathcal{A}(x, y; f_0, f_1, \ldots, f_k)$ be the output of the attacker. Let s_a indicate if $\hat{a} = a$, let s_b indicate if $\hat{b} = b$, and return (s_a, s_b).

Figure 2: Membership-inference with model updates.

output of each model f_i on multiple query points, but does not have knowledge of the specific models architecture or parameters. We assume the adversary knows when the model is updated, potentially by querying the model and noticing when it changes its predictions. As models are retrained with new data, the adversary's goal is to infer if a data sample was part of the update set or not. In the setting with multiple model updates, the adversary is also interested in inferring at which time epoch the data sample was used to update the model. Figure 1 visualizes this threat model. We remark that the adversary here only makes inferences about the update set, not the initial training set. In general, the update set will generally be smaller than the initial training set, so our attacks are designed to cause maximum leakage on a small subset of the data.

3.3　Formalization of MI under Model Updates

We formalize the problem of membership inference with model updates by adapting the membership inference experiment of [43] to the model update setting. We present the experiment in full generality, and introduce specific contexts for which we subsequently develop specific attack algorithms. Using our experiment, we also suggest a new form of leakage that can occur in the multiple update setting, called entry inference.

The membership-inference experiment. Let A_{up} be a model update algorithm, D_0 an initial training set, f_0 an initial model, n_{up} an update set size, and each $\mathcal{D}_i$ with $i \in [k]$ a distribution over samples (x, y). We define the MI experiment in Figure 2. The experiment allows the adversary access to all updated models, and requires it to distinguish between update and test data.

We instantiate this experiment in three settings, for attack algorithm development (Section 4) and evaluation (Section 5): single update, multiple updates, and single update distribution shift.

Single model update. In the single update setting, we consider $k = 1$, $\mathcal{D}_1 = \mathcal{D}$, so the initial training distribution does not change to sample updates. This setting lets us understand the difference between having access to the models before and after an update compared to only the final model. To evaluate the performance of our membership inference attacks, we can measure *accuracy*, $\mathbb{E}[s_a]$, and *precision*, $\mathbb{E}[s_a | \hat{b} = 1]$. An attack maximizing precision may differ from one maximizing accuracy. We also measure *recall*, $\mathbb{E}[\hat{b} = 1 | b = 1]$, but, as noted in [26] and [21], a MI attack achieving high precision is likely to be more harmful than one achieving high recall.[2] An attack which classifies every sample as appearing in the training set is not harmful, but obtains a high recall; meanwhile, an attack correctly identifying a single sample as appearing in training set has tiny recall, but harms that sample.

Multiple model updates. The multiple update setting considers $k > 1$ and $\mathcal{D}_i = \mathcal{D}$ for all i, so the training distribution remains constant. This lets us understand the difference between access to multiple models and only the last model. In this setting there is also a richer set of attacks that can be run.

- *Membership Inference*: As before, we measure the membership inference accuracy. This is the success rate at inferring if a data sample is part of any of the k update datasets, $\mathbb{E}[s_b]$. The harm here is identical to the standard harms suggested for membership inference: membership in a medical study may indicate that somebody has a disease, or membership in a loan dataset might indicate that someone applied for a loan.

- *Entry Inference*: We also introduce this novel threat, where the adversary guesses precisely at which update step a user enters the dataset, with success rate $\mathbb{E}[s_a]$. This measures how well the adversary can determine *when* a user started being used in training, which itself may be sensitive. For example, when a patient enters a medical study may leak when they contracted a disease, or knowing when a loan was issued may leak sensitive information about someone's financial status. There are also cases where membership inference is more of a threat than entry inference; for example, if high membership inference success leads to social security numbers being extractable from a large language model [6], it is much more important that the social security numbers are leaked than when they entered the dataset, making membership inference more scary than entry inference. However, in other cases, membership inference may also be less of a concern than entry inference. For example, membership in a commerce dataset may indicate that someone has been to a certain coffee shop, which itself may not be risky. However, knowing *when* the person entered the dataset may allow an adversary to link this time with other people's entry into the dataset, and learn about meetings between people. Ultimately, membership inference and entry inference are both

---

[2]Carlini et al. [5] suggests measuring true positive rate (TPR) and false positive rate (FPR). TPR is identical to recall, and FPR can be computed easily from precision, given that b takes the value 0 and 1 equally often.

risks and can lead to harms in meaningfully different ways. Finally, we note that any attack which achieves a membership inference accuracy of p can be converted to an entry inference attack by randomly selecting one of the k update sets, achieving entry inference accuracy of p/k. A standard random guessing baseline therefore reaches a membership inference accuracy of $1/2$ and an entry inference accuracy of $1/2k$.

Distribution shift. In this setting, we consider $k = 1$ and $\mathcal{D}_1 \neq \mathcal{D}$. While distribution shift may happen over several updates, we elect to isolate the impact that distribution shift has on our attacks. To measure attack performance, we use the same metrics as in the single update setting: membership inference accuracy, precision, and recall.

Retraining methods. There are multiple ways to use the datasets $D_0, D_1, \cdots D_i$ to update model f_{i-1} at epoch i, which we call:

- **SGD-New.** This strategy updates the model f_{i-1} using only the new training set $D = D_i$. To prevent forgetting earlier datasets, one must use a small learning rate and few epochs.
- **SGD-Full.** This strategy updates the model f_{i-1} using the entire training set available at epoch i, $D = [D_0, D_1, \cdots D_i]$. With a larger dataset, one can increase the learning rate and number of epochs, at the cost of using less recent data. This setting is practical for models with modestly sized training sets, and may be impractical for some models, like large language models.

These training strategies have also been used by previous work considering model updates. Zanella-Béguelin et al. [45] compare both strategies for extracting training data from generative language models, while Salem et al. [35] use a variant of continual training for mounting reconstruction attacks.

4 ATTACK ALGORITHMS

We develop attacks for the single and multiple update instantiations of the MI under model update problem introduced in the preceding section. Our attacks are generic with respect to distribution shift and retraining methods, so we discuss those topics directly as part of our evaluation of the proposed attacks (Section 5). We first focus on attacks for the single update setting (Section 4.1). We propose multiple options and subsequently justify analytically both their designs and the need for the options (Section 4.2). Finally, we propose several attack options for the multiple update setting (Section 4.3). Table 1, placed at the end of this section, summarizes the various attacks and their options for easy access.

4.1 Single Update Attacks

Given a single model f_0 trained on a dataset D_0, and an individual target example (x, y), the standard black-box way to test membership of x in D_0 is to compute an appropriate *score function* $\ell(x, y; f_0)$ and apply some *threshold* to this score.

Now suppose we are given two models f_0, f_1 trained on datasets D_0 and $D_0 \cup D_1$ respectively, and a target example (x, y), whose membership in D_1 we want to infer. Intuitively, being a member of D_1 is equivalent to being a member of $D_0 \cup D_1$ and a non-member of D_1. So, a first attempt is simply to infer membership in $D_0 \cup D_1$ and membership in D_0 and decide membership in D_1 appropriately.

However, as we show in Section 4.2, if the score information is binary (e.g. it is the output of some membership-inference attack for each standalone model), then model updates *do not* increase the accuracy of membership-inference attacks.

However, score-based membership-inference attacks give more information than just the binary outcome, and a key contribution of our work is to show how to strictly outperform the preceding baseline by *combining* the two scores, $\ell(x, y; f_0)$ and $\ell(x, y; f_1)$. Given these two scores, there are multiple logical ways we could combine them to produce a single score for membership in D_1. In this work, we introduce two main strategies: SCOREDIFF and SCORERATIO. As their names suggest, we define

$$\text{SCOREDIFF}(x, y, f_0, f_1, \ell) = \ell(x, y, f_1) - \ell(x, y; f_0) \text{ and}$$

$$\text{SCORERATIO}(x, y, f_0, f_1, \ell) = \frac{\ell(x, y, f_1) + c}{\ell(x, y; f_0) + c}$$

where $c > 0$ is a damping constant to avoid instability when the denominator is close to 0. As we will show empirically in Section 5, neither of these two methods for combining scores dominates the other, and in Section 4.2 we give analytical justification for why each score can sometimes be superior.

While we could take the approach of Chen et al. [9] and learn how best to combine scores from scratch, our evaluation will show that choosing a fixed combiner, such as SCOREDIFF or SCORERATIO, is both more efficient and more effective.

To use these methods for combining scores, we need to do two things: (1) instantiate these strategies by choosing the score function, and (2) determine a threshold to apply to convert the real-valued scores into a binary membership decision. As score functions, in this paper we use the standard cross-entropy loss [43] and the state-of-the-art LiRA score function [5]. These are both computable with only class probabilities, but LiRA requires training shadow models. Future single-model MI attacks might provide even better score functions that our strategies can incorporate.

In this work we consider multiple ways to set the threshold. To motivate these methods, we return to the standard interpretation of membership inference as a hypothesis-testing problem. Once the score function is fixed, any query point gets mapped to some value v. Typically, v is compared to a threshold T: v membership as IN if $v \geq T$, and OUT otherwise. This performs well, because the distributions P_{IN} of IN scores and P_{OUT} of OUT scores will differ. In practice, however, the attacker does not know the IN and OUT distributions, and so needs some side information to find a good threshold. All membership-inference attacks give the attacker some side information for this goal. In this work we consider a few different types of side information the attacker can have, corresponding to different ways of setting the threshold:

- *Batch Strategy.* The adversary has access to a dataset containing both update points (IN) and test points (OUT), but does not know which are IN. Given these points, the adversary can compute scores and thereby see samples from the distributions P_{IN} and P_{OUT} and can then find an optimal threshold for distinguishing these points. For example, if the attacker has k update points and k test points for large k, then the quantiles of these $2k$ scores will give a good threshold. In our work we choose the median to maximize accuracy and the 10th percentile to maximize precision. The

assumption that the attacker has access to many update points is strong, but a useful thought experiment, since the attacker can try to approximate these samples on their own using other forms of side information, as our next approach does.

- *Transfer Strategy ("Shadow Models").* The adversary has access to some set of test points. Using these test points, the attacker trains *shadow models* [26, 35, 39] $\hat{f}_0$ and then updates these models using a random half of the test points. Provided that the test points are drawn from the same (or similar) distribution and the attacker can train the models in the same (or similar) method as the real models, this should allow us to approximate P_{IN} and P_{OUT}. Since the attacker knows which points were included in the update set, the attacker now has a batch of update points and test points, so the attacker can use the batch strategy to obtain a good threshold for these points and then transfer that threshold to the models they are trying to attack. This only uses one shadow model, as we identify a threshold on a scalar value, rather than train a classifier, as many shadow model attacks do [9, 35].

- *Rank Strategy.* The adversary again has access to some set of test points. The attacker can use these to generate samples from P_{OUT}, and given these samples, can use the q-quantile of the distribution as a threshold. This strategy will achieve a false-negative rate of q. This strategy was employed by Leino et al. [26].

4.2　Analytical Justification

The preceding single-update attack algorithms rely on two key insights that our work contributes: (1) that taking advantage of model updates requires combining rich score information from individual-model MI attacks against the model and its update; and (2) that the two specific methods, ScoreDiff and ScoreRatio, that we propose for combining two scores into a single MI attack in the model-update setting are both needed and justified. This section provides analytical justification for these two insights. Section 5 provides empirical evidence in support of these claims.

The need for rich score information. We justify our score combination approach by showing that model updates can only increase accuracy of membership-inference when the attacker can obtain rich (non-binary) score information. We prove that, under reasonable assumptions, an adversary with only access to the binary scores does not improve when given access to the initial model.

Here, we consider a threat model where a MI attack $\mathcal{A}$ returns only binary IN/OUT predictions on a point. Any MI attack eventually returns this, and some MI attacks (e.g. the "gap attack" [10]) only use binary information. Given access to two models, there are four possible outcomes for the MI attack: $\{\mathcal{A}(f_0) = IN, \mathcal{A}(f_0) = OUT\} \times \{\mathcal{A}(f_1) = IN, \mathcal{A}(f_1) = OUT\}$. The adversary can make a decision for each of these outputs, the frequencies of which we write below (as is standard, we assume any point has a .5 probability of being included in the update or test set, so these frequencies are the frequencies for an update or a test example).

	Update		Test	
	$\mathcal{A}(f_1)$: IN	$\mathcal{A}(f_1)$: OUT	$\mathcal{A}(f_1)$: IN	$\mathcal{A}(f_1)$: OUT
$\mathcal{A}(f_0)$: IN	p_{11}^u	p_{10}^u	p_{11}^t	p_{10}^t
$\mathcal{A}(f_0)$: OUT	p_{01}^u	p_{00}^u	p_{01}^t	p_{00}^t

Theorem 4.1 proves that, under realistic assumptions, knowledge of f_0 does not improve the adversary's attack. The assumptions are: (1) The attack on f_0 returns IN equally often on update and test points: $p_{11}^u + p_{10}^u = p_{11}^t + p_{10}^t$. This is realistic, as both sets of points are not in f_0's training set. (2) The attack on f_1 returns IN more frequently on update than test points, for both points classified as IN and OUT by the attack on f_0: $p_{11}^u > p_{11}^t$, $p_{01}^u > p_{01}^t$. This is realistic, as the attack should return IN frequently on training data.

THEOREM 4.1. *For any single-model membership inference attack $\mathcal{A}$ returning only a binary IN/OUT prediction satisfying the above assumptions, there exists an adversary receiving the output of $\mathcal{A}$ on f_1 that obtains at least as high accuracy as any adversary with access to the output of $\mathcal{A}$ on both f_0 and f_1.*

PROOF SKETCH. The proof identifies optimal decisions for the adversary with and without updates. The optimal strategy without updates follows the guesses of $\mathcal{A}$. We show that, because the update leads to more IN guesses on update points, the optimal strategy with updates is the same as without. The full proof is in the Appendix. □

The main takeaway of Theorem 4.1 is that for an attack to successfully use model updates, it must exploit some rich score information, rather than binary IN/OUT information, which includes the model's generalization gaps. We note that it is impossible to prove that richer score information will lead to successful update-based attacks; as a simple counterexample, the richer score could take one of three values, but only two are observed in practice. In this case, Theorem 6 would apply, and updates would not help. As a result, we see it is important to design setting-dependent score combination approaches, as we do in Section 4.1.

Justifying ScoreDiff and ScoreRatio. We justify our choice of score combination functions by studying the example of computing the mean and the median of the training data. These two examples will show both that our choices are well motivated, and also demonstrate that the right approach depends on the specific learning problem being solved, and thus there is likely not a single best method. Section 5.5 confirms these claims empirically.

The earliest work in membership-inference [19, 36], in the single-model/no-update setting, justified specific membership-inference attacks by exploiting a connection to hypothesis testing and use the Neyman-Pearson Lemma to devise optimal attacks. Proving exact optimality typically requires making strong distributional assumptions, and being able to reason explicitly about the exact distribution of the outputs of the learning algorithm. In our work, we mostly consider learning algorithms that are too complex for this sort of precise analysis (such as neural networks), so we settle for a more heuristic justification instead. In particular, we will consider a learning algorithm that outputs the exact minimizer of the loss function to obtain the initial model f_0, then performs an update on a single point x_i^1 by performing a single gradient step from f_0 with fixed step size to obtain the updated model f_1. This update strategy corresponds to what we call SGD-New, but with a single step of training. We then analyze how the loss on the point x_i^1 changes as a result of the update and will see that the change is best reflected by either ScoreDiff or ScoreRatio, *depending on the loss function.*

Algorithm 2: Multiple Update Attacks

Data: Test Sample x, y, Models $f_0, f_1, \cdots f_k$, Privacy Score
Function ℓ, Threshold T or Thresholds $\{T_i\}_{i=1}^k$,
Strategy ATTACK (SCOREDIFF or SCORERATIO)

Function BackFront$(x, y, f_0, f_1, \cdots, f_k, \ell, T, \text{ATTACK})$:

  return $\mathbb{1}(\text{ATTACK}(x, y, f_0, f_k, \ell) < T)$

Function Delta$(x, y, f_0, f_1, \cdots, f_k, \ell, \{T_i\}_{i=1}^k, \text{ATTACK})$:

  For $i \in 1 \ldots k$

    $g_i = \mathbb{1}(\text{ATTACK}(x, y, f_{i-1}, f_i, \ell) < T_i)$

    If $g_i = \text{IN}$ **return** (IN, i)

  return OUT

In the Appendix, we show that SCORERATIO is a perfect membership inference attack for the ℓ_2 loss, and SCOREDIFF is a perfect attack for the ℓ_1 loss. In the Appendix, we study mean estimation under updates, showing that model updates provably improve accuracy, and lower bounding the accuracy of SCOREDIFF.

4.3 Multiple Update Attacks

Multiple updates can allow leakage in two ways: an adversary can learn both *whether* a user is contained in a training set (membership inference), but also *when* they begin participating in that dataset (entry inference). The former is the standard membership inference task, but entry inference may also be sensitive in several settings, such as in medical datasets, where someone could learn when a patient contracted a disease. We construct an attack for each goal: the Back-Front attack for membership inference and the Delta attack for entry inference, in Algorithm 2.

Back-Front attack. This attack ignores all information except for the first and last model update, and is designed for the membership inference case. This attack is the natural adaptation of the single model attacks to the multiple update setting, as it treats the sequence of k updates as a single, large, update algorithm. The attack uses either the score difference or ratio between the original model f_0 and the final model f_k after k updates.

Delta attack. This attack is designed to perform entry inference. In this attack, we identify samples to a specific update i when they have a large loss difference (or ratio) on the two consecutive models f_{i-1}, f_i produced by the update. Notice that we can also adapt this attack (or any entry inference attack) to a membership inference attack: a sample which is predicted to be in any update is predicted to be IN. As a result, we measure this attack's performance on both membership and entry inference.

Threshold setting. The three thresholding strategies we defined for the single-update setting, Batch, Transfer, and Rank, can be adapted to configure the threshold for the multi-update attacks. However, for simplicity, we focus on the Batch strategy. With Batch, the adversary is given a dataset D which contains each update set $\{D_i\}_{i=1}^k$ of size n_{up}, as well as a test set of size kn_{up}. For the Back-Front attack, we set the threshold to the median value of the entire dataset D. For the Delta attack, we set each threshold T_i so that n_{up} points are classified into each update index.

Method	Function	Description
SCOREDIFF	single-update attack	compute difference of scores
SCORERATIO	single-update attack	compute ratio of scores
Back-Front	multi-update attack	ignore intermediate models
Delta	multi-update attack	analyze adjacent models
Transfer	setting threshold	transfer from a shadow model
Batch	setting threshold	calibrate with update/test points
Rank	setting threshold	calibrate with test points

Table 1: Proposed attacks and thresholding strategies.

5 EVALUATION

We next evaluate our proposed algorithms for the single- and multi-update settings by answering seven key questions in the context of the datasets described in Section 5.1:

Q1: Does access to one model update give the attack an advantage over models with no access to updates? How does the update set size impact this advantage? (Section 5.2)

Q2: Does attack success improve with more updates? (Section 5.3)

Q3: How does the training strategy—SGD-New or SGD-Full—impact attack performance? (Section 5.4)

Q4: How do the various attacks and thresholding choices impact attack performance? (Section 5.5)

Q5: How does distribution shift impact attack performance? (Section 5.6)

Q6: How would adoption of differential privacy impact attack performance? (Section 5.7)

Q7: How do our attacks compare to those developed for unlearning in Chen et al. [9]? (Section 5.8)

Since we propose multiple attacks for two different settings – single-update and multiple updates – and each attack can be instantiated with multiple score functions and thresholding methods, the space of experimentation is quite large. Moreover, performance of an attack can be measured in different ways, such as with accuracy, precision, and recall for the single-update setting, and membership inference accuracy and entry inference accuracy for the multiple-update setting. To tame this large experimentation space, we answer different questions in the context of different algorithms, methods, settings, and metrics that are most relevant for the specific question. Q1-Q3 choose the best-performing attack relevant to the considered setting – single or multiple updates – and focus on membership inference accuracy and entry inference accuracy, respectively. Q4 compares key pairs of algorithms and mechanisms under multiple performance metrics. And Q5-Q7 focus on the single-update setting and best-performing algorithms.

5.1 Datasets

In this section we describe the datasets used in our evaluation. We use the Adam optimizer for transfer learning datasets (CIFAR-10 and IMDb) and SGD otherwise, as we find this works best.

FMNIST. FMNIST is a 10-class dataset consisting of 28x28 pixel grayscale images of different clothing items. On this dataset, the initial model is a logistic regression model, trained on an initial dataset of 1000 data points for 50 epochs at a learning rate of 0.01. We use 1000 data points to be able to train many of these models, including running LiRA calibration. On average, this achieves 82.5% training accuracy and 79.5% test accuracy. SGD-New trains for 10 epochs at a learning rate of 0.001, and SGD-Full trains for 10 epochs at a learning rate of 0.01.

CIFAR-10. CIFAR-10 is a 10-class dataset of 32x32 pixel RGB images of various animals and vehicles. This dataset is harder than FMNIST, and requires more complex models to achieve reasonable accuracy. We fine-tune a VGG-16 network which was pretrained on the ImageNet dataset. The initial model is trained with 12 epochs over a training set of 25000 points at a learning rate of 10^{-4}. On average, this achieves 98.2% training accuracy and 82.6% test accuracy. SGD-New trains for 4 epochs at a learning rate of 10^{-5}, and SGD-Full trains for 2 epochs at a learning rate of 10^{-4}.

Purchase100. Purchase100 is a 100-class purchase history dataset. The task is to classify a shopper into one of 100 clusters. Here, our initial model is a single layer neural network trained on an initial dataset of 25000 samples for 95 epochs at a learning rate of 0.01. On average, this achieves 94.1% training accuracy and 84.1% test accuracy. SGD-New trains for 5 epochs at a learning rate of 0.01, and SGD-Full trains for 10 epochs at a learning rate of 0.1.

IMDb. IMDb is a text dataset of movie reviews, where the task is to classify a movie review as either positive or negative. Here, we fine tune the BERT base model (uncased) which was pretrained on a large collection of English data. The initial model is trained with 4 epochs over a training set of 25000 points at a learning rate of 10^{-5}. On average, this achieves 99.4% training accuracy and 93.8% test accuracy. SGD-New trains for 6 epochs at a learning rate of 10^{-6}, and SGD-Full trains for 3 epochs at a learning rate of 10^{-5}.

5.2 Advantage from a Single Update

Q1: Does access to one model update give the attack an advantage over models with no access to updates? How does the update set size impact this advantage?

To evaluate MI advantage from a single update, we compare with three baseline attacks that use only the updated model f_1. The first baseline, called *Loss*, uses the approach of Yeom et al. [43], which compares the loss on a point to the average training loss. The second baseline, called *Gap*, uses the gap attack of [10], which classifies correctly classified points as training and incorrectly classified points as test. These strategies capture the role of *label memorization* as a cause for vulnerability [15], and so attacks which perform better than these cannot rely solely on such memorization. For all datasets except IMDb, we also evaluate use the LiRA attack [5] as a baseline, which trains shadow models to compute sample-specific baseline loss values to compare to.

Figure 3 shows the accuracy of the best of our single-update attacks, compared to the best baseline without access to updates. We show these accuracies for update sizes varying from 1% to 32% of the original training set (except IMDb, where we use a fixed 10-320 points for acceptable running time). The best attack differs in each setting. For updates, fixing the Batch thresholding strategy, we choose the best of {ScoreDiff, ScoreRatio} × {loss score, LiRA score}. For no updates, we choose the best of {Gap, Loss, LiRA} baselines. For both update and no-update, we show accuracy for both training strategies SGD-Full and SGD-New.

For all datasets, update sizes, and training strategies, attacks with model updates outperform the no-update attacks. As expected, the gap between updates and no updates decreases as the update set gets larger. On FMNIST, for example, at 10 update points, the Batch attack achieves 79% accuracy, while the Batch attack achieves

70% accuracy at 320 update points. The gap between the update and no update attacks decreases from 27% to 18%. As an additional exploration of this effect, we compare the accuracy of our attacks with the success rate of single model attacks when the training set is the same size as the update in Appendix C, and find that our attacks are often more effective than this setting as well. Also in Appendix C, we evaluate disparate impact, finding some classes are somewhat more vulnerable to our attacks.

Q1 answer: Overall, our results show that updates give the adversary significant advantage to identify training set members.

5.3 Advantage from Multiple Updates

Q2: Does advantage improve with the number of updates?

To evaluate the threat of MI and entry inference with the number of updates, we run our multi-update attacks on a sequence of 1 to 10 updates, and observe how each attack's performance changes. To isolate the role of multiple updates, we fix n_{up} to 250 for CIFAR-10, 100 for Purchase100, and 10 for FMNIST and IMDb (1%, .4%, 1%, and .04% of the initial dataset, respectively). We measure both membership and entry inference accuracies, which are the metrics relevant for the multi-update attack, however we focus in this section on entry inference accuracy and evaluate membership inference accuracy in a subsequent section. However, we remind the reader that a membership inference attack can be used to construct an entry inference attack, which we will use as a baseline.

We compare against two baselines. The first, called *Random*, is random guessing. It randomly selects IN or OUT with probability $1/2$ and selects the update index uniformly at random. This results in a success probability of $1/2k$ when there are k updates. We also define a stronger baseline, called *Generic*, which runs a membership inference attack to answer IN or OUT, and selects the update uniformly at random. We ignore the details of membership inference attacks for this section. If the membership inference accuracy of the attack is p, then it obtains entry inference accuracy p/k. When membership inference accuracy is better than random chance ($p > 1/2$), Generic outperforms Random.

Figure 4 shows the multiplicative improvement of entry inference accuracy over random guessing as a function of the number of updates. As before, we show the best attack for each case, fixing the thresholding strategy to Batch. Our best attacks significantly outperform the baselines for entry inference, with the gap increasing with the number of updates. For example, for SGD-New on CIFAR-10, at 2 updates, our best attack (in this case, ScoreRatio with cross-entropy loss as the score) outperforms the baseline by a factor of 2.03×, while at 8 updates, it outperforms the baseline by 6.10×. Still, for both the baselines and our attacks, the absolute value of entry inference accuracy decays with the number of updates. This is natural, as increasing the number of updates makes it more difficult to guess which of the many updates a point was used in—this is why the Random and Generic baselines' success probabilities – $1/2k$ and p/k, respectively – decay with k. For our attack, the absolute value of entry inference accuracy for CIFAR-10 is 50.8% at 2 updates and 38.1% at 8 updates. Yet, our attacks fare much better compared to baselines, whose entry inference accuracy for CIFAR-10 is 25% at 2 updates and 6.25% at 8 updates. For a real adversary, it is unclear whether the absolute accuracy level or the

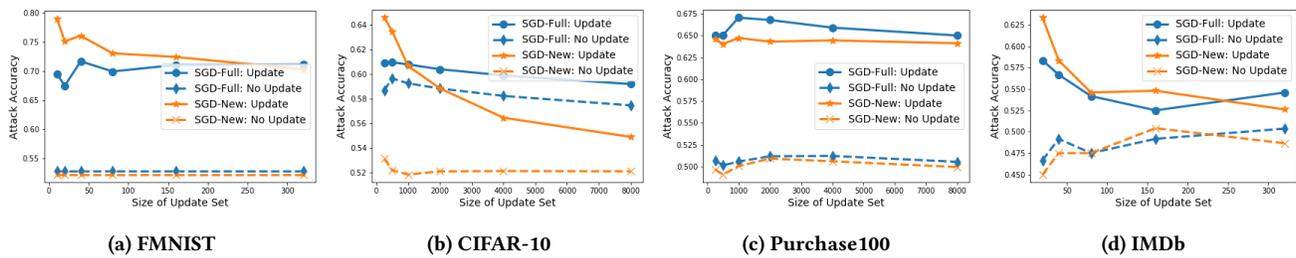(a) FMNIST   (b) CIFAR-10   (c) Purchase100   (d) IMDb

Figure 3: (Q1, Q3) Accuracy of attacks with access to one update (solid lines) vs. without access to updates (dashed lines). Each line corresponds to the best attack for each setting (see text). Even one update gives the adversary significant power to identify training set members.
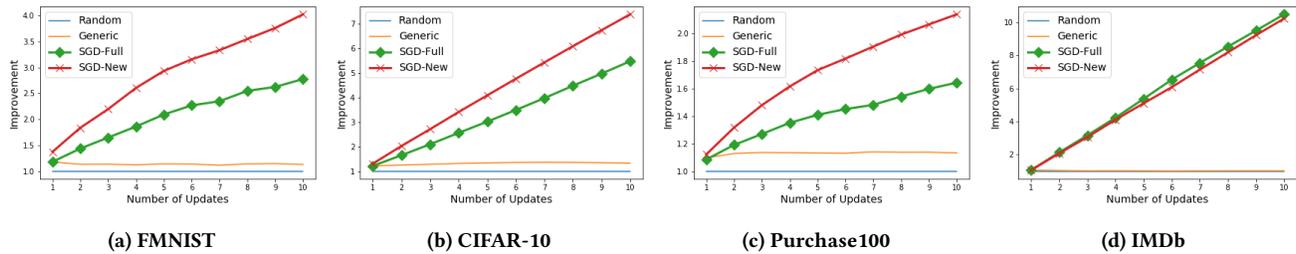

(a) FMNIST   (b) CIFAR-10   (c) Purchase100   (d) IMDb

Figure 4: (Q2, Q3) Improvement of entry inference accuracy over random guessing (y axis), as a function of the number of updates (x axis). For k updates, Random guessing achieves $1/2k$ accuracy. We present the best attack strategy for each dataset: for FMNIST and Purchase100, this is loss difference, but loss ratio for CIFAR-10. As the number of updates increases, the attack performs significantly better than random guessing, although the absolute accuracy decays.

improved attack performance will be more relevant, and future work may be required to increase the absolute attack success rate further, especially for very long training processes.

Q2 answer: Thus, as the number of updates increases, the absolute accuracy fundamentally decreases for all cases, but our attack significantly outperforms appropriate baselines.

5.4 Impact of Training Strategy

Q3: How does the training strategy—SGD-New or SGD-Full—impact attack performance?

The success of the MI attack depends on the training strategy, which is a choice the learner makes that the adversary cannot influence. If one training strategy were consistently less vulnerable to attack than the other, then the learner could choose the former, as a heuristic defense. We find that such a heuristic exists, but it depends on the update size and dataset, making it difficult for the learner to configure without experimentation with our attacks.

We revisit Figures 3 and 4, which already illustrate the effect of SGD-New and SGD-Full on attack performance with one and multiple updates, respectively. For a single update, Figure 3 shows the attack is more effective with SGD-New when the update set is small. However, the attack's performance on SGD-New degrades faster than on SGD-Full as the update set size increases. For example, on CIFAR, the gap between 250 and 8000 update points is 10% for SGD-New but only 2% for SGD-Full. This faster drop in performance with SGD-New on larger update sets makes SGD-Full,

in fact, more vulnerable to attack for large update sets on CIFAR-10 compared to SGD-New. The reason for this inversion is that SGD-Full would naturally use a higher learning rate for the update set compared to SGD-New. Interestingly, the inversion already happens on Purchase100 at the smallest update set size.

For multiple updates, we observe a similar and even more consistent effect. Figure 4, which uses a very small update set of 1% of the original training set, shows the attack as most effective with SGD-New. This is true even more consistently across datasets than with a single update. In the multiple update setting, SGD-New only uses each update point once, so an update point will observe a larger loss decrease in the update in which it appears compared to the loss in updates in which it does not appear. SGD-Full, by contrast, should observe a loss decrease in each update. We include results in Appendix C (Figure 10a) showing that for a larger update set size of 10% of the original training set, attacks on SGD-Full are more effective relative to attacks on SGD-New.

Q3 answer: Thus, as a rule of thumb, when update sets are small, training on the whole dataset is less vulnerable to MI attack; but when update sets are larger, training on only the update set is less vulnerable. The point at which the inversion happens depends on the dataset, so a learner should tune this heuristic, for example, by running our attack. Section 5.6 evaluates differential privacy as a far more principled approach for defense against MI attacks, yet still one that will likely require some tuning or auditing in practice, for which our attacks could still prove useful.

5.5 Impact of Attack Strategy

Q4: How do the various attacks and thresholding choices impact attack performance?

In the preceding questions, we picked the best performing attack algorithm and thresholding strategy from the suite we are proposing. But are all these attacks/thresholds really needed, or do some of them outperform others consistently? The answer is, indeed, that different algorithms and thresholds perform better in different settings, so they are all relevant in an attacker's toolkit. For example, the LiRA score function is more difficult to compute than loss, so should be used when it can be reliably estimated. Recall that the adversary's knowledge determines their choice of threshold. We show here a few main comparisons.

SCOREDIFF vs. SCORERATIO in terms of accuracy. Focusing first on the single update setting, Table 2 shows attack accuracy with an update size of 1% of the original training set. We compare SCOREDIFF and SCORERATIO when using the standard cross-entropy loss and either the Batch or Transfer thresholding strategy (Batch and Transfer are the only ones relevant for accuracy). We show results for both SGD-New and SGD-Full training strategies. The SCORERATIO strategy typically outperforms SCOREDIFF. However, there are exceptions, especially at large update set sizes. The two threshold selection strategies, Batch and Transfer, have comparable performance in the cases we show. We also ran the same evaluation with LiRA as the score in SCOREDIFF and SCORERATIO. LiRA consistently performs better than traditional loss, except on CIFAR-10, where there is not enough data to train sufficient shadow models for it to perform well. We find LiRA tends to be strongest with SCORERATIO, but SCOREDIFF performs comparably.

For the multiple update setting, there is similarly no single best choice for the attacker. For example, loss ratio outperforms loss difference on CIFAR-10, while loss difference outperforms loss ratio on FMNIST and Purchase100, and there is little difference on IMDb.

For a specific use case, an adversary might experiment on their own shadow updated model to determine whether to use SCOREDIFF or SCORERATIO, and which privacy score to use. Thresholding strategies are determined by what the adversary has access to.

SCOREDIFF vs. SCORERATIO in terms of precision/recall. The broad takeaways for precision are similar to those for accuracy. SGD-New outperforms SGD-Full, as we can see from comparing Table 3 and a table we included in the Appendix, Table 7. At $n_{up} = 0.01n_0$, Batch achieves 90% precision on FMNIST for SGD-New, while the best attack achieves 86% precision for SGD-Full. This difference is more pronounced for other datasets. Precision is larger when fewer update points are used: Transfer achieves 86% precision with $n_{up} = 0.01n_0$ on FMNIST, but only 78% precision with $n_{up} = 0.08n_0$. For precision, we have three strategies for selecting the threshold: Batch, Transfer, and also Rank. While Rank often achieves a higher recall than the other strategies, Batch and Transfer typically achieve a higher precision. Thus, once again, the choice of algorithm and its configurations requires experimentation.

Delta vs. Back-Front in terms of membership inference accuracy. For the multiple update setting, we proposed two membership inference strategies for the attacker: Delta, which compares each adjacent pair of models; and Back-Front, which only compares the first and last models. Figure 5 compares these two strategies for SGD-Full in terms of membership inference accuracy. As before, no strategy strictly dominates. The Delta attack performs the best for CIFAR-10. On Purchase100, the Back-Front attack is best. On FMNIST, the Back-Front and Delta attack perform comparably. As before, an adversary could decide between one strategy versus another based on experiments with a shadow model that they train themselves.

Q4 answer: These results show that the variety of algorithms and configurations that we proposed is truly needed, because no single algorithm and configuration will work best in all situations.

5.6 Impact of Distribution Shift

Q5: How does distribution shift impact attack performance?

An important characteristic of the model retraining setting we tackle in this paper is that models are updated to keep up with a shifting distribution. How distribution shift impacts MI attack performance has been evaluated very little in prior literature, hence its investigation, in the context of our proposed attacks, is a significant contribution of our work. The contribution consists of two components: (1) a new methodology that we developed for evaluating impact of distribution shift on MI attacks and (2) the evaluation of our proposed attack algorithms with this methodology.

Methodology. We consider two types of distribution shift: subpopulation shift and covariate shift.

We consider *subpopulation shift* using the BREEDS framework [37]. The BREEDS framework generates subpopulation shift by generating a hierarchy of classes and shifting between classes which are close in the hierarchy. For example, a "dog" class trained on images of dalmatians may struggle to recognize poodles as dogs. We adapt BREEDS to CIFAR-10 with an "animal vs. vehicle" binary task, and vary the animals and vehicles to simulate distribution shift. "Animals" are { bird, cat, deer, dog, frog, horse}; "vehicles" are { airplane, automobile, ship, truck }. For the "animal" class, we consider a source class s_a and a target class t_a (both class 0 in the binary task). Likewise, the "vehicle" class has a source class s_v and target class t_v (both class 1). We write the distribution of a given class c as $\mathcal{D}^c$. We consider balanced classes, so that the source distribution is $\mathcal{D}_S = \frac{1}{2}(\mathcal{D}^{s_a} + \mathcal{D}^{s_v})$ and the target distribution $\mathcal{D}_T = \frac{1}{2}(\mathcal{D}^{t_a} + \mathcal{D}^{t_v})$. The original training distribution is $\mathcal{D}_S$, and the update distribution is $\mathcal{D}_1 = (1 - \alpha)\mathcal{D}_S + \alpha\mathcal{D}_T$, where the shift ratio α controls the strength of the distribution shift.

For *covariate shift*, we use the CINIC-10 dataset [11]. CINIC-10 is a combination of the CIFAR-10 dataset and the ImageNet dataset, filtered to the 10 classes of the CIFAR-10 dataset. The covariate shift we consider is from a $\mathcal{D}_S$ consisting of purely CIFAR-10 data (or purely ImageNet data) to a mixture of an α fraction of CIFAR-10 and a $1 - \alpha$ fraction of ImageNet data. This is not a form of subpopulation shift as CIFAR-10 and ImageNet do not represent different subpopulations of each class, rather a different distribution of examples for each class (with ImageNet consisting of harder data). We also describe in Appendix C an experiment using CINIC-10 to confirm our subpopulation shift experiments on a larger dataset.

We run our single update attacks from Section 4.1 on the preceding distribution shift methodologies. Our goal is to isolate the role of distribution shift on attack performance. For both settings, we vary

Dataset	SGD-New						SGD-Full					
	No Update		ScoreDiff (loss)		ScoreRatio (loss)		No Update		ScoreDiff (loss)		ScoreRatio (loss)	
	Gap	Loss	Batch	Transfer	Batch	Transfer	Gap	Loss	Batch	Transfer	Batch	Transfer
FMNIST	0.51	0.53	0.68 *	0.67 *	0.71 **	0.72 **	0.52	0.51	0.63	0.55	0.57	0.58
CIFAR-10	0.50	0.53	0.61 **	0.62 **	0.65 **	0.68 **	0.50	0.59	0.58	0.59	0.61	0.63 **
Purchase100	0.51	0.50	0.56 **	0.56 **	0.57 **	0.58 **	0.51	0.51	0.54 *	0.54 *	0.55 **	0.54 *
IMDb	0.50	0.45	0.63 **	0.60 *	0.63 **	0.60 *	0.50	0.47	0.58	0.55	0.58	0.55

Table 2: (Q4) Accuracy for single update models with smallest update size. Values with * improve over the best no update attack ($p < 0.05$), values with ** improve with higher significance ($p < 0.01$).

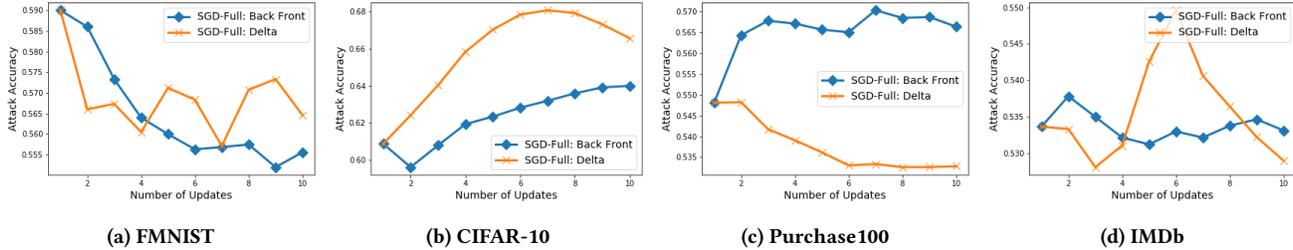**(a) FMNIST** 　　　　 **(b) CIFAR-10** 　　　　 **(c) Purchase100** 　　　　 **(d) IMDb**

Figure 5: (Q4) Membership inference accuracy for attacks with multiple updates. We select the best strategy for each case: for FMNIST, this is always loss difference (although ratio performs similarly), and is loss ratio for CIFAR-10 and Purchase100.

Dataset	n_{up}	Loss	Batch	Transfer	Rank
FMNIST	10	0.53/0.11	0.78/0.16	0.86/0.24	0.82/0.28
CIFAR-10	250	0.54/0.10	0.85/0.17	0.82/0.22	0.69/0.46
Purchase100	250	0.53/0.11	0.60/0.12	0.63/0.07	0.60/0.14
IMDb	10	0.58/0.12	0.58/0.12	0.56/0.32	0.52/0.25
FMNIST	80	0.52/0.10	0.74/0.15	0.78/0.13	0.69/0.25
CIFAR-10	2000	0.55/0.11	0.83/0.17	0.79/0.25	0.69/0.44
Purchase100	2000	0.49/0.10	0.63/0.13	0.65/0.10	0.61/0.16
IMDb	80	0.46/0.09	0.67/0.13	0.70/0.12	0.59/0.25

Table 3: (Q4) Precision/recall results for single update models on SGD-Full. Rank sets the threshold to the top 10% of the test data. We report values for two update set sizes. It is well known that no update loss obtains poor precision [5, 26].

the parameter α and the "hardness" of shift, keeping everything else constant. For subpopulation shift, we consider two settings of $(s_a \rightarrow t_a)$, $(s_v \rightarrow t_v)$. The first, which we call *Hard*, is (Airplane $\rightarrow$ Automobile), (Cat $\rightarrow$ Bird). The second, which we call *Easy*, is (Automobile $\rightarrow$ Truck), (Cat $\rightarrow$ Dog). Hard is a distribution shift where the original model will not perform well on the new data, due to the dissimilarity between original and update classes. Easy is a distribution shift where the original model will perform well. For covariate shift, we call starting with CIFAR-10 as the *Hard* setting, as this model will perform poorly on ImageNet data, and starting with ImageNet is the *Easy* setting, as this model will perform better on CIFAR-10 data.

Importantly, our methodology does *not* measure the ability of an adversary to distinguish the old and new distributions. This is because our MI game formulation from Section 3 samples test points and update points identically, from the same distribution. Instead, we are measuring the ability to distinguish *shifted training points* from *shifted testing points*. Intuitively, a large distribution shift requires the model to fit to the specific update points to accommodate the new distribution, thereby making the update points vulnerable to MI. It is worth noting that the only prior work investigating distribution shift's impact on privacy, namely Zanella-Béguelin et

Metric	Shift	$\alpha = 0.2$		$\alpha = 0.6$		$\alpha = 1.0$	
		Diff	Ratio	Diff	Ratio	Diff	Ratio
Accuracy	Hard	0.551	0.559	0.602	0.634	0.600	0.687
	Easy	0.685	0.685	0.574	0.581	0.560	0.562
Prec./Recall	Hard	.73/.15	.90/.18	.94/.19	.96/.19	.98/.20	.94/.19
	Easy	.64/.13	.74/.15	.58/.12	.90/.18	.71/.14	.94/.19

Table 4: (Q5) Accuracy and precision/recall of attacks after subpopulation shift on SGD-New. Ratio, Diff stand for ScoreRatio, ScoreDiff, respectively, with loss score. Ratio performs best, a Hard distribution shift results in better attacks as α increases, and SGD-New is typically more vulnerable than SGD-Full.

Metric	Shift	$\alpha = 0.2$		$\alpha = 0.6$		$\alpha = 1.0$	
		Diff	Ratio	Diff	Ratio	Diff	Ratio
Accuracy	Hard	0.628	0.718	0.615	0.694	0.596	0.673
	Easy	0.595	0.614	0.588	0.616	0.589	0.652
Prec./Recall	Hard	.93/.19	.87/.17	.89/.18	.83/.17	.80/.16	.79/.16
	Easy	.89/.18	.90/.18	.91/.18	.89/.18	.87/.17	.84/.17

Table 5: (Q5) Accuracy and precision/recall of attacks after co-variate shift on SGD-New. Ratio, Diff stand for ScoreRatio, ScoreDiff, respectively, with loss score. Here, Ratio performs best for accuracy, and a Hard distribution shift results in more accurate attacks than an Easy shift, but α no longer has as significant an impact as with subpopulation shift.

al. [45], shows that some outputs are more likely after a distribution shift, but their experiments cannot isolate whether this is a privacy violation or just the model adapting to the new distribution. We thus believe that our methodology can constitute a better platform for future measurements of MI under distribution shift.

Evaluation. The second component of our distribution-shift contribution is the evaluation of our proposed algorithms using the preceding methodologies. We focus on the single update attacks, and use the Batch thresholding strategy. For the subpopulation shift setup, we use $n_{up} = 50$ and $n_0 = 5000$. The covariate shift is a more complex task, so we set $n_{up} = 1000$ and $n_0 = 10000$. We evaluate accuracy and precision/recall for both shifts with $\alpha \in \{0.2, 0.6, 1.0\}$,

with results for subpopulation shift shown in Table 4 and results for covariate shift in Table 5. We show results for SGD-New, using both ScoreDiff and ScoreRatio with the loss score (SGD-Full results for subpopulation shift can be found in Table 10 in the Appendix).

Accuracy: ScoreRatio is the most effective strategy in all cases. For the Hard subpopulation shift, the faster the change in distribution (larger α), the more effective MI is. As α increases from 0.2 to 1.0, accuracy on SGD-Full increases by 0.07, and accuracy increases by 0.13 on SGD-New. Interestingly, there is no strong trend with α for covariate shift, likely because the task is complex enough that it leads to high risk regardless (our attacks with covariate shift tend to have high accuracy and precision). Similar to how SGD-New is more influenced by n_{up} in Section 5.4, SGD-New is also more heavily influenced by α. Interestingly, accuracy decreases as α grows for the Easy shift. This shows that a drastic shift requires significant changes to be made to the model, leading the model to overfit to the specific update points and make them more vulnerable.

Precision/Recall: The lessons for precision/recall are similar to those for accuracy, but precision tends to be much higher than accuracy, often reaching >90% (and as high as 98%), especially at more drastic subpopulation shifts. For subpopulation shift, we still find that ScoreRatio is generally more effective, and Hard shifts and large α result in higher precision. For covariate shift, neither attack clearly outperforms the other, and precision is always high, even with small α and Easy shifts. The key difference with accuracy we note is that even the Easy shift results in large precision at high α (and even at low α for covariate shift); this is likely because, even when the distributions are similar, there are some samples which still require the model to change significantly to learn them.

Disparate Impact: Finally, we consider whether our attacks have *disparate impact* [23, 46]. That is, we ask whether some subpopulations are more impacted by our attacks. We focus on this question here, due to the explicit distinctions made between source and target distributions. For covariate shift, we find no disparate impact. This is likely due to the lack of distinction between subpopulations. For subpopulation shift with CIFAR, we do find disparate impact: with $\alpha = .2$, new examples from the Hard shift reach 78% accuracy, while old examples are less vulnerable, with 42% accuracy. The Easy shift is less drastic: new examples reach 62% accuracy, while old examples reach 53% accuracy.

Q5 answer: A drastic subpopulation shift can result in higher MI risk than a gradual shift, but not if MI risk is already high.

5.7 Differential Privacy

Q6: How does differential privacy impact our attacks?

A rigorous strategy for preventing our attacks is by training with differential privacy. Indeed, training with (ε, δ)-differential privacy, imposes some upper bound on the accuracy of *any* MI attacks and also on the precision at a fixed level of recall. We evaluate the effectiveness of this strategy by training with DP-SGD [1, 3, 41], the standard algorithm for training differentially private models. This algorithm modifies standard SGD by clipping gradient norms and adding noise. We use the implementation provided by Tensorflow Privacy [28]. We fix $\delta = 10^{-4}$, and vary the noise multiplier to reach fixed values of ε, computed with the accounting provided by the repository. We focus on the Fashion-MNIST dataset, with an update

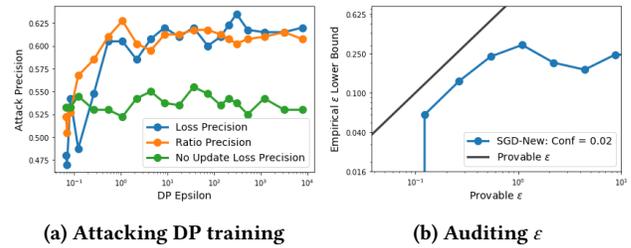(a) Attacking DP training   (b) Auditing ε

Figure 6: (Q6) (a) Effectiveness of differential privacy at defending against single update attacks, as ε varies for SGD-New. Accuracy and precision for both SGD-New and SGD-Full observe similar behavior, where no update attacks are similar at small ε, but fall behind at larger ε. (b) Using our attacks to audit differentially private SGD-Full and SGD-New. All lower bounds for SGD-Full are 0. For multiple values of ε, SGD-New lower bounds are a 2.0-3.6x factor from upper bounds.

size of 100, a clipping norm of 0.5, and we fix all other parameters to be the same as in previous sections. We focus on the single update setting for simplicity. This model retains accuracy similar to the non-private model (around 75% test accuracy) when epsilon is at or above 0.2. However the test accuracy decreases below 65% at an epsilon of 0.06.

Protection wanes with ε. We present the results of this experiment in Figure 6a. As expected, as ε increases, its protection from our attacks decreases. For example, at $\varepsilon = 0.26$, our best attack reaches a precision of only 0.59, but at $\varepsilon = 1.1$, it reaches a precision of 0.63. The precision levels off as ε increases. The gap between the attacks with and without access to updates is also largest at moderate ε values. At $\varepsilon < 1$, there is little difference. The no-update attacks also catch up at extremely large ε, where the noise addition is minor, and gradient-clipping is the major difference between DP-SGD and the standard implementation of SGD.

Auditing differentially private deployments with our attacks. Following [20, 33], we can use our results to provide empirical lower bounds on the privacy of each update algorithm, as a means of understanding how worst-case upper bounds on ε correspond to practical privacy against state-of-the-art attacks. We can convert the bound used by either [20] or [33] to a bound on precision, giving that precision p should be bounded by $e^{\varepsilon}/(1 + e^{\varepsilon})$, or, identically, that ε is lower bounded by $p/(1 - p)$. Since we cannot measure p directly, we follow [20] and compute conservative estimates via Clopper-Pearson confidence intervals. We have 400 trials for each ε, as we train 20 models with 20 points in each trial. We report in Figure 6b these computed ε values for both SGD-New and SGD-Full. We enforce a confidence of 98% for each reported value.

We highlight two key takeaways. First, as in non-private training, SGD-New empirically offers less privacy protection than SGD-Full for those points in the update. In fact, our attack does not refute the possibility that SGD-Full satisfies differential privacy with $\varepsilon = 0$, although we stress that this is not robust evidence that privacy is not a concern when retraining with SGD-Full. The second takeaway is that the provable upper bounds on the privacy of SGD-New are nearly tight for moderate values of ε. With provable ε in .12-1.09,

our lower bounds are within a 2.0-3.6x factor of the theoretical upper bound. A gap smaller than 3.6x is perhaps remarkable, since state-of-the-art attacks on standalone models trained with DP-SGD have gaps of 5-10x [20, 33], suggesting that model updates represent an especially risky scenario for private model training.

Q6 answer: Our results show that differential privacy is an effective protection at lower ε, and also that our attacks can be an effective method of empirically auditing a differential privacy deployment.

5.8 Comparison with Chen et al. [9]

Q7: How do our attacks compare to those developed for unlearning in Chen et al. [9]?

We observe that machine unlearning can be viewed as the "reverse operation" of our model update setting. Then we can adapt the MI attacks designed for machine unlearning in Chen et al. [9], and compare them to ours. In their strategy, the adversary trains shadow models to learn an "attack model". This attack model takes as input some combination of the probability vectors returned by the two models, and outputs a prediction for whether the point was unlearned. They experiment with different instantiations of the attack, and we reproduce their SortedDiff attack in our setting, which takes the difference between sorted probability vectors before and after deletion. They note SortedDiff is their best attack on well-generalized models, as our models are. We run their attack with up to 30 shadow models, with each of the attack model architectures tested, and report the best attack from these. We focus on our loss score attacks with Transfer thresholds, as this fits the threat model they consider (LiRA requires training different types of shadow models, so we avoid this comparison for simplicity). Our attacks here will therefore only use a single shadow model (to set the threshold), while their attacks will be allowed up to 30.

We compare ScoreDiff and ScoreRatio with the Chen et al. [9] attack in Figure 7, for both SGD-Full and SGD-New, on the FMNIST dataset. We observe that both ScoreDiff and ScoreRatio always outperform their attack, although the gap can be somewhat small, depending on the update size. We are able to do this with a single shadow model, because it is easier to identify a good threshold on a single feature, than learn a good function on 10 features[3]. We corroborate this on Purchase100.

To further demonstrate the strength of the test statistics we compute, we run an experiment allowing the Chen et al. attack model to access our ScoreDiff and ScoreRatio features, in addition to the features they use (this results in 12 features, two of ours, and 10 SortedDiff features from their paper). We allow this improved Chen et al. attack 30 shadow models, and have it learn a logistic regression attack model on these 12 features. We inspect the weights the model learns, as a way to measure how useful the features are, and find that the average weight assigned to our features is, on average, 7.5x higher than the weight assigned to one of the Sorted-Diff features! This speaks to the value of carefully constructing a useful test statistic, rather than attempting to learn one from a high dimensional space. Shadow models are more useful when used to improve a simple test statistic, as our results with LiRA show.

---

[3]We also note that their attacks are much stronger on overfit models, and the attack they performed on MNIST models that do not exhibit a large amount of overfitting only achieved an AUC of .51.
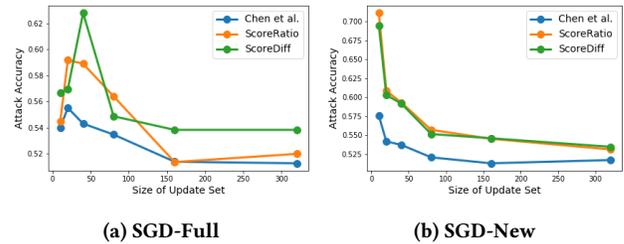

(a) SGD-Full **(b) SGD-New**

Figure 7: (Q7) Performance of Chen et al. [9] relative to our attacks. We consistently outperform their attack, despite only using a single shadow model to set our attack threshold.

Q7 answer: ScoreDiff and ScoreRatio are more efficient (fewer shadow models) and more effective (higher accuracy) than [9].

6 CONCLUSIONS

We have presented and evaluated MI attacks which leverage model updates. Our attacks apply to a variety of settings, including when models are repeatedly updated and when the distribution shifts over time. Our strategies are theoretically justified and empirically achieve both high accuracy and high precision. Empirically, we find the role of the update set size, the training algorithm, and any distribution shift to be key factors impacting our attacks' performance. As a general rule, the smaller the update set size, the more effective our attacks are. This holds true for attacks on single updates as well as multiple updates, and for both SGD-New and SGD-Full. It is well known that MI attacks are more successful when training sets are smaller [14, 36], so our results confirm this in the model updates setting. Since model accuracy also improves more with larger updates, this is a nice win-win for privacy and utility! A drastic distribution shift also improves the performance of an attack, as learning the new distribution requires fitting heavily to the update points. Zanella-Béguelin et al. [45] also notice that distribution shift results in memorization in the generative language model setting, but our results are the first to identify this as privacy leakage, rather than the model adapting to a new distribution. Finally, the specific training setup used by the learner can impact the accuracy of MI. Models updated repeatedly can be used to improve MI, and can also leak the time that a data point appeared in the training set. Learners training with SGD-New are more vulnerable than SGD-Full at small update set sizes and when distributions shift significantly, but this trend tends to reverse as the update set gets larger and the distribution is more stable.

ACKNOWLEDGMENTS

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications*

security, pages 308–318, 2016.

[2] Microsoft Azure. https://docs.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets?tabs=python.

[3] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[4] Mark Bun, Jonathan Ullman, and Salil Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *ACM Symposium on the Theory of Computing*, STOC '14, pages 1–10, New York, NY, USA, 2014. https://arxiv.org/abs/1311.3158.

[5] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914. IEEE, 2022.

[6] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pages 267–284, 2019.

[7] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[8] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[9] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 896–911, 2021.

[10] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[11] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[12] Luc Devroye, Abbas Mehrabian, and Tommy Reddad. The total variation distance between high-dimensional gaussians. *arXiv preprint arXiv:1810.08693*, 2018.

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[14] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, 2015.

[15] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[17] Google. Tensorflow extended. https://www.tensorflow.org/tfx.

[18] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[19] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS genetics*, 4(8):e1000167, 2008.

[20] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

[21] Bargav Jayaraman, Lingxiao Wang, Katherine Knipmeyer, Quanquan Gu, and David Evans. Revisiting membership inference under realistic assumptions. *Proceedings on Privacy Enhancing Technologies*, 2021(2):348–368, 2021.

[22] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 259–274, 2019.

[23] Bogdan Kulynych, Mohammad Yaghini, Giovanni Cherubin, Michael Veale, and Carmela Troncoso. Disparate vulnerability to membership inference attacks. *arXiv preprint arXiv:1906.00389*, 2019.

[24] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5):1302–1338, 2000.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *29th USENIX Security Symposium*, pages 1605–1622, 2020.

[27] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021.

[28] Google LLC. Tensorflow privacy. https://github.com/tensorflow/privacy.

[29] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, 2019.

[30] Akshay Naresh Modi, Chiu Yuen Koo, Chuan Yu Foo, Clemens Mewald, Denis M. Baylor, Eric Breck, Heng-Tze Cheng, Jarek Wilkiewicz, Levent Koc, Lukasz Lew, Martin A. Zinkevich, Martin Wicke, Mustafa Ispir, Neoklis Polyzotis, Noah Fiedel, Salem Elie Haykal, Steven Whang, Sudip Roy, Sukriti Ramesh, Vihan Jain, Xin Zhang, and Zakaria Haque. Tfx: A tensorflow-based production-scale machine learning platform. In *KDD 2017*, 2017.

[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 634–646, 2018.

[32] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[33] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *IEEE Symposium on Security & Privacy*, IEEE S&P '21, 2021. https://arxiv.org/abs/2101.04535.

[34] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[35] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th USENIX Security Symposium*, pages 1291–1308, 2020.

[36] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, 2009.

[37] Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. {BREEDS}: Benchmarks for subpopulation shift. In *International Conference on Learning Representations*, 2021.

[38] Amazon Web Services. Sagemaker model monitor. https://aws.amazon.com/sagemaker/model-monitor/.

[39] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[40] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[41] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[42] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Eavesdrop the composition proportion of training labels in federated learning. *arXiv preprint arXiv:1910.06044*, 2019.

[43] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[44] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM Conference on Computer and Communications Security*, CCS '20, pages 363–375, 2020. https://arxiv.org/abs/1912.07942.

[45] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM Conference on Computer and Communications Security*, CCS '20, pages 363–375, 2020.

[46] Da Zhong, Haipei Sun, Jun Xu, Neil Gong, and Wendy Hui Wang. Understanding disparate effects of membership inference attacks and their countermeasures. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 959–974, 2022.

	Update		Test	
	$\mathcal{A}(f_1)$: IN	$\mathcal{A}(f_1)$: OUT	$\mathcal{A}(f_1)$: IN	$\mathcal{A}(f_1)$: OUT
$\mathcal{A}(f_0)$: IN	p_{11}^u	p_{10}^u	p_{11}^t	p_{10}^t
$\mathcal{A}(f_0)$: OUT	p_{01}^u	p_{00}^u	p_{01}^t	p_{00}^t

Table 6: Probabilities used in Theorem 4.1

A PROOF OF THEOREM 4.1

We now prove Theorem 4.1, which shows that rich score information is required to take advantage of updates. We begin by restating the theorem and assumptions.

- The attack run on f_0 returns IN equally often on update and test points: $p_{11}^u + p_{10}^u = p_{11}^t + p_{10}^t$. This is a reasonable assumption as both sets are not in f_0's training set and are identically distributed, so the attack should not perform differently.
- The attack run on f_1 is more likely to return IN on points in the update than on points in the test set, for both points with IN and OUT guesses from the attack on f_0: $p_{11}^u > p_{11}^t$, $p_{01}^u > p_{01}^t$. This is realistic, as the membership inference attack should return IN more frequently on training data than on testing data.

THEOREM A.1 (THEOREM 4.1, RESTATED). *For any single-model membership inference attack $\mathcal{A}$ returning only a binary IN/OUT prediction satisfying the above assumptions, there exists an adversary receiving the output of $\mathcal{A}$ on f_1 that obtains at least as high accuracy as any adversary with access to the output of $\mathcal{A}$ on both f_0 and f_1.*

PROOF. Given only knowledge of f_1, the adversary may only make a decision based on $\mathcal{A}(f_1)$. Then a uniform decision must be made for points with $\mathcal{A}(f_1) = $ IN, representing a $p_{11}^u + p_{01}^u$ fraction of update points and a $p_{11}^t + p_{01}^t$ fraction of test points. Because there is a balance between update and test points, there will be more updates points in this set when $p_{11}^u + p_{01}^u > p_{11}^t + p_{01}^t$, so the optimal attack should guess IN when this inequality holds and OUT otherwise. The fraction of overall points which are correctly classified by this rule will be $\frac{1}{2}(\max(p_{11}^u + p_{01}^u, p_{11}^t + p_{01}^t))$. Similarly, points for which $\mathcal{A}(f_1) = $ OUT should be guessed as IN if $p_{10}^u + p_{00}^u > p_{10}^t + p_{00}^t$ and OUT otherwise. Then the optimal attack achieves accuracy

$$\frac{1}{2}\left[\max(p_{11}^u + p_{01}^u, p_{11}^t + p_{01}^t) + \max(p_{10}^u + p_{00}^u, p_{10}^t + p_{00}^t)\right].$$

Now, Assumption 2 states that the attack guesses IN on update data more than on test data, so $\max(p_{11}^u + p_{01}^u, p_{11}^t + p_{01}^t) = p_{11}^u + p_{01}^u$; the optimal attack should guess IN when $\mathcal{A}$ does. Similarly, $\max(p_{10}^u + p_{00}^u, p_{10}^t + p_{00}^t) = p_{10}^t + p_{00}^t$, so the optimal attack should guess OUT when $\mathcal{A}$ does. The optimal attack here is to simply use predictions returned by $\mathcal{A}$. This optimal attack strategy reaches an accuracy of $\frac{1}{2}(p_{11}^u + p_{01}^u + p_{10}^t + p_{00}^t)$.

Using Updates. Now, an attack with access to both f_0 and f_1 may make decisions based on both attack results. Following the same argument as before, we see that a p_{11}^u fraction of update points are guessed as IN for both models, while a p_{11}^t fraction of test points are, so the optimal attack should classify these points as update if $p_{11}^u > p_{11}^t$ and test otherwise. Applying the same logic to all four possible attack results, we see that the optimal attack achieves

accuracy

$$\frac{1}{2}\left[\max(p_{11}^u, p_{11}^t) + \max(p_{01}^u, p_{01}^t)\right] + \frac{1}{2}\left[\max(p_{10}^u, p_{10}^t) + \max(p_{00}^u, p_{00}^t)\right].$$

Again, Assumption 2 gives $\max(p_{11}^u, p_{11}^t) = p_{11}^u$ and $\max(p_{01}^u, p_{01}^t) = p_{01}^u$. Assumption 1 states that the attack returns IN equally often on update and test points on f_0, so combining Assumption 1 and Assumption 2 gives $p_{10}^u < p_{10}^t$, so that $\max(p_{10}^u, p_{10}^t) = p_{10}^t$. Finally, Assumption 1 implies that $p_{01}^u + p_{00}^u = p_{01}^t + p_{00}^t$. Combining this with Assumption 2 gives us that $p_{00}^u < p_{00}^t$, so $\max(p_{00}^u, p_{00}^t) = p_{00}^t$. Then the accuracy of the optimal attack when given both f_0 and f_1 is

$$\frac{1}{2}\left[p_{11}^u + p_{01}^u + p_{10}^t + p_{00}^t\right],$$

equivalent to the attack which did not use updates. The attack in both cases is identical - follow the guesses made by $\mathcal{A}$ when run on f_1. □

B ADDITIONAL THEORETICAL ANALYSIS

B.1 Justifying SCOREDIFF and SCORERATIO

We consider two loss functions: (1) the ℓ_2^2 loss, which is denoted $\ell^m(x, f) = \|f - x\|_2^2$, and whose minimizer is the dataset's mean, and (2) the ℓ_2 loss $\ell^g(x, f) = \|f - x\|_2$, whose minimizer is the dataset's geometric median. For simplicity we also consider a single update point x.

For the ℓ_2^2 loss, the update rule is

$$f_1^m = f_0^m - \eta\nabla\ell^m(x, f_0^m) = f_0^m - 2\eta(f_0^m - x).$$

Now we have,

$$\begin{aligned}\ell^m(x, f_1^m) &= \langle(f_1^m - f_0^m) + (f_0^m - x), (f_1^m - f_0^m) + (f_0^m - x)\rangle \\ &= \|f_0^m - x\|_2^2 + \|f_1^m - f_0^m\|_2^2 + 2\langle f_1^m - f_0^m, f_0^m - x\rangle \\ &= \ell^m(x, f_0^m) + (4\eta^2 - 4\eta)\|f_0^m - x\|_2^2 \\ &= (1 - 2\eta)^2\ell^m(x, f_0^m).\end{aligned}$$

When recomputing the mean, the loss after update is a fixed ratio decrease from the loss before the update. Meanwhile, the probability that a randomly drawn test point will have the same loss ratio is 0. Then, with a single update point and a known learning rate, loss ratio is a *perfect* membership test.

For the ℓ_2 loss, the update rule is

$$f_1^g = f_0^g - \eta\nabla\ell^g(x, f_0^g) = f_0^g - \eta\frac{f_0^g - x}{\|f_0^g - x\|_2}.$$

Now we have,

$$\begin{aligned}\ell^g(x, f_1^g) &= \sqrt{\|(f_1^g - f_0^g) + (f_0^g - x)\|_2^2} \\ &= \sqrt{\|f_0^g - x\|_2^2 + \|f_1^g - f_0^g\|_2^2 + 2\langle f_1^g - f_0^g, f_0^g - x\rangle} \\ &= \sqrt{\ell^g(x, f_0^g)^2 + \frac{\eta^2\|f_0^g - x\|_2^2}{\|f_0^g - x\|_2^2} - 2\eta\frac{\|f_0^g - x\|_2^2}{\|f_0^g - x\|_2}} \\ &= \sqrt{\ell^g(x, f_0^g)^2 + \eta^2 - 2\eta\ell^g(x, f_0^g)} \\ &= \ell^g(x, f_0^g) - \eta,\end{aligned}$$

which shows that recomputing the geometric median results in a fixed constant decrease from the loss before the update, making it also a perfect membership test in this setting.

B.2 Mean Estimation

In this section we give a more detailed treatment of Example 1.1 from the Introduction. Namely, we analyze the effect of updates in a very simple case: a single update of a rudimentary "model" that estimates the mean over a multi-dimensional dataset.

We consider the task of estimating the mean of samples drawn from a d-dimensional spherical Gaussian distribution $\mathcal{D} = \mathcal{N}(\mu, \mathbb{I}_{d \times d})$. We consider a learner $A_{\text{train}} = A_{\text{up}}$ which simply outputs the sample mean of its training data, and which produces two models—the first, $\hat{\mu}_0$, is computed on a dataset D_0 of n_0 samples, and the second, $\hat{\mu}_1$, is computed with an additional n_1 samples. The total dataset $D = [D_0; D_1]$ contains $n = n_0 + n_1$ samples. We consider an adversary who seeks to identify whether a given v was contained in D_1. In this setting, we can upper bound the performance of *any* attack when the adversary has no access to model updates (that is, it only has access to $\hat{\mu}_1$). Next, we show that, when given access to model updates (both $\hat{\mu}_0$ and $\hat{\mu}_1$), the adversary can outperform this upper bound.

Theorem B.1. *Consider membership-inference using mean estimation in $\mathbb{R}^d$ with an initial dataset of size n_0 and a single model update with a set of size n_1. If*

$$\frac{1}{2} + \sqrt{\frac{2d}{n-1}} < \Phi\left(\sqrt{\frac{d}{80(n_1-1)}}\right) \tag{1}$$

then there is an attacker with access to both $\hat{\mu}_0$ and $\hat{\mu}_1$ that outperforms every adversary with access to only $\hat{\mu}_1$. Here $\Phi(z) = \Pr[N(0,1) \leq z]$ is the Gaussian CDF.

The condition (1) is always satisfied when $n \gg d \gg n_1$, in which case the left-hand side is close to $1/2$ and the right-hand side is close to 1. We now prove this statment, by breaking it into two lemmas. Lemma B.2 upper bounds all adversaries with access to only $\hat{\mu}_1$. Lemma B.3 analyzes a specific attack using $\hat{\mu}_0$ and $\hat{\mu}_1$ together. Combining these lemmas proves Theorem B.1.

Lemma B.2. *For the task of membership inference of a sample v on mean estimation, with probability $> 1 - \exp(-d)$ over the selection of v, all adversaries $\mathcal{A}$ have success rate bounded above by*

$$Acc(\mathcal{A}) = \frac{1}{2} + \frac{1}{2}\left(\sqrt{\frac{5d}{n-1}} + \frac{\sqrt{d}}{n-1}\right).$$

Proof. Notice that $\hat{\mu}_0$ is distributed as $\mathcal{N}(\mu, \frac{\sigma^2}{n_0}\mathbb{I})$ and $\hat{\mu}_1$ as $\mathcal{N}(\mu, \frac{\sigma^2}{n_0+n_1}\mathbb{I})$. An adversary seeking to distinguish between the cases where a sample v is contained in D_1 or not must distinguish between two distributions: the distribution over means where v is not contained in D_1, $\mathcal{D}_{\text{OUT}} = \mathcal{N}(\mu, \frac{\sigma^2}{n}\mathbb{I})$, and the distribution over means where v is contained in D_1, $\mathcal{D}_{\text{IN}} = \frac{1}{n}v + \mathcal{N}(\frac{n-1}{n}\mu, \frac{\sigma^2(n-1)}{n^2}\mathbb{I})$. We can upper bound the success of this adversary by a function of the total variation (TV) distance between the two distributions: $\frac{1}{2} + \frac{1}{2}TV(\mathcal{D}_{\text{OUT}}, \mathcal{D}_{\text{IN}})$.

Recall that the TV distance between $\mathcal{N}(x_0, \Sigma_0)$ and $\mathcal{N}(x_1, \Sigma_1)$ can be upper bounded [12] by

$$TV(\mathcal{N}(x_0, \Sigma_0), \mathcal{N}(x_1, \Sigma_1)) \leq \left\|\Sigma_0^{-1/2}(x_0 - x_1)\right\|_2 + \left\|\mathbb{I} - \Sigma_0^{-1/2}\Sigma_1\Sigma_0^{-1/2}\right\|_F.$$

We use this to bound $TV(\mathcal{D}_{\text{OUT}}, \mathcal{D}_{\text{IN}})$ as follows:

$$TV(\mathcal{D}_{\text{OUT}}, \mathcal{D}_{\text{IN}}) \leq \left\|\left(\frac{n}{\sigma\sqrt{n-1}}\mathbb{I}\right)\left(\frac{1}{n}v - \frac{1}{n}\mu\right)\right\|_2 +$$
$$\left\|\mathbb{I} - \frac{n}{\sigma\sqrt{n-1}}\mathbb{I}\left(\frac{\sigma^2}{n}\mathbb{I}\right)\frac{n}{\sigma\sqrt{n-1}}\mathbb{I}\right\|_F$$
$$= \frac{\|v - \mu\|_2}{\sigma\sqrt{n-1}} + \left\|\frac{1}{n-1}\mathbb{I}\right\|_F$$
$$= \frac{\|v - \mu\|_2}{\sigma\sqrt{n-1}} + \frac{\sqrt{d}}{n-1}.$$

Because $v - \mu \sim \mathcal{N}(0, \sigma^2\mathbb{I})$, we have $\|v - \mu\|_2 < \sigma\sqrt{5d}$ [24] except with probability at most $\exp(-d)$, which gives us

$$TV(\mathcal{D}_{\text{OUT}}, \mathcal{D}_{\text{IN}}) \leq \sqrt{\frac{5d}{n-1}} + \frac{\sqrt{d}}{n-1}.$$

This completes the proof. □

Lemma B.3. *For the task of membership inference of a sample v on mean estimation with model updates, there exists an adversary $\mathcal{A}$ with success rate*

$$Acc(\mathcal{A}) \geq \Phi\left(\sqrt{\frac{d}{80(n_1-1)}}\right).$$

This holds with probability $> 1 - 2\exp(-d/16)$ over the choice of v, and when $n_1 > 1$.

Proof. We consider an adversary with access to model updates, receiving two quantities. The first is the mean of D_0, $\hat{\mu}_0 \sim \mathcal{N}(\mu, \frac{\sigma^2}{n_0}\mathbb{I})$. The next is the overall mean $\hat{\mu}_1$. When v is not contained in D_1, $\hat{\mu}_1$ is distributed as $\frac{n_0}{n}\hat{\mu}_0 + \frac{n_1}{n}\mathcal{N}(\mu, \frac{\sigma^2}{n_1}\mathbb{I})$. When v is found in D_1, $\hat{\mu}_1$ is distributed as $\frac{n_0}{n}\hat{\mu}_0 + \frac{1}{n}v + \frac{n_1-1}{n}\mathcal{N}(\mu, \frac{\sigma^2}{n_1-1}\mathbb{I})$.

With both of these quantities, the adversary computes the mean of only D_1: $\hat{\mu}_{\text{Delta}} = \frac{n}{n_1}\hat{\mu}_1 - \frac{n_0}{n_1}\hat{\mu}_0$. The task of determining whether v is contained in D_1 can now be written as the task of distinguishing between the distribution of $\hat{\mu}_{\text{Delta}}$ when v is not included, $\mathcal{D}_{\text{Delta,OUT}}$, and the distribution when it is included, $\mathcal{D}_{\text{Delta,IN}}$, both written below:

$$\mathcal{D}_{\text{Delta,OUT}} = \mathcal{N}\left(\mu, \frac{\sigma^2}{n_1}\mathbb{I}\right) \text{ and } \mathcal{D}_{\text{Delta,IN}} = \frac{1}{n_1}v + \mathcal{N}\left(\frac{n_1-1}{n_1}\mu, \frac{\sigma^2(n_1-1)}{n_1^2}\mathbb{I}\right).$$

Now, the adversary computes $s(\hat{\mu}_{\text{Delta}}, v) = (\hat{\mu}_{\text{Delta}} - \mu) \cdot (v - \mu)$. In the OUT case, $s(\hat{\mu}_{\text{Delta}}, v)$ is distributed as $\mathcal{N}(0, \frac{\|v-\mu\|_2^2\sigma^2}{n_1})$. In the IN case, it is distributed as $\mathcal{N}(\frac{1}{n_1}\|v-\mu\|_2^2, \frac{\|v-\mu\|_2^2\sigma^2(n_1-1)}{n_1^2})$.

The adversary guesses OUT if $s(\hat{\mu}_{\text{Delta}}, v)$ is below $T = \frac{1}{2n_1}\|v - \mu\|_2^2$ and IN otherwise. If $\mathcal{D}_{\text{Delta,IN}}$ and $\mathcal{D}_{\text{Delta,OUT}}$ had equal variance, this would be the optimal Neyman-Pearson distinguisher [34]; because the variances are similar, the test will still be effective. For convenience, we write $c = \|v - \mu\|_2^2\sigma^2$.

The probability the adversary succeeds when v is OUT is

$$Acc_{\text{OUT}} = \Pr\left[\mathcal{N}\left(0, \frac{c}{n_1}\right) \le T\right],$$

and the probability of success when v is IN is

$$Acc_{\text{IN}} = \Pr\left[T \le \mathcal{N}\left(2T, \frac{c(n_1-1)}{n_1^2}\right)\right] = \Pr\left[\mathcal{N}\left(0, \frac{c(n_1-1)}{n_1^2}\right) \le T\right].$$

The adversary achieves accuracy $Acc = \frac{1}{2}(Acc_{\text{OUT}} + Acc_{\text{IN}})$.

We have $Acc_{\text{IN}} \le Acc_{\text{OUT}}$ because of its higher variance. Then we can use proceed by computing Acc_{IN} as a lower bound for Acc. To compute this probability, we notice that, due to the distribution of v, we have $\|v - \mu\|_2 < \sigma\sqrt{5d}$ with probability $> 1 - \exp(-d)$, and $\|v - \mu\|_2 > \sigma\sqrt{d/2}$ with probability $> 1 - \exp(-d/16)$ [24]. We can therefore lower bound $T > d\sigma^2/4n_1$ and upper bound $c < 5d\sigma^4$, in all giving

$$Acc(\mathcal{A}) > \Pr\left[\mathcal{N}\left(0, \frac{5d\sigma^4(n_1-1)}{n_1^2}\right) \le \frac{d\sigma^2}{4n_1}\right]$$

$$= \Phi\left(\sqrt{\frac{d}{80(n_1-1)}}\right)$$

This completes the proof. □

B.3 ScoreDiff Achieves High Accuracy

We now show that ScoreDiff with the loss score on mean estimation achieves high accuracy when $d >> n_1$.

Theorem B.4. *Suppose f_0 is the mean of D_0, and f_1 is produced by taking a single gradient step from f_0 with a learning rate of η on the ℓ_m loss. Then there is some constant C and threshold T such that, if $d > Cn_1$, running ScoreDiff with a threshold of T reaches a membership inference accuracy of $>90\%$ for both SGD-Full and SGD-New.*

In the following, we write the mean of D_1 as μ_{up}. When excluding a sample x, we write the mean of D_1/x as μ_{rest}. We begin by proving Lemma B.5.

Lemma B.5. *For the task of mean estimation with an update D_1, when f_0 is the mean of the original dataset D_0, a gradient step with learning rate η using SGD-Full is equal to a gradient step using SGD-New with a learning rate of $\eta' = \frac{n_1\eta}{n_0+n_1}$.*

Proof. In SGD-New, the gradient step is performed on D_1 is

$$-2\eta \sum_{x_i \in D_1} \frac{f_0 - x_i}{n_{\text{up}}} = 2\eta(\mu_{\text{up}} - f_0).$$

In SGD-Full, the gradient step on $D_0 \cup D_1$ is

$$-2\eta \sum_{x_i \in D_0 \cup D_1} \frac{f_0 - x_i}{n_0 + n_{\text{up}}} = \frac{2n_{\text{up}}\eta}{n_0 + n_{\text{up}}}(\mu_{\text{up}} - f_0),$$

as the gradient on D_0 adds to 0, because f_0 is the minimizer of ℓ_m on D_0. We see that these gradient steps are rescalings of each other, as we wanted to show. □

Having proven Lemma B.5, we can now prove Theorem B.4, by fixing an η and analyzing the loss difference with SGD-New.

Proof. We write the loss for a fixed x and η, and will later consider the cases where x is IN D_1 and where x is a test point, OUT of D_1.

$$\ell^m(f_1, x)$$
$$= \langle f_1 - x, f_1 - x \rangle$$
$$= \langle (f_1 - f_0) + (f_0 - x), (f_1 - f_0) + (f_0 - x) \rangle$$
$$= \|f_1 - f_0\|_2^2 + \|f_0 - x\|_2^2 + 2\langle f_1 - f_0, f_0 - x \rangle$$
$$= \ell^m(f_0, x_i) + \|2\eta(\mu_{\text{up}} - f_0)\|_2^2 - 4\eta\langle \mu_{\text{up}} - f_0, x - f_0 \rangle.$$

Now, the norm $\|2\eta(\mu_{\text{up}} - f_0)\|_2^2$ can be computed from f_0 and f_1, so we consider only the distribution of the rightmost term $d = -4\eta\langle \mu_{\text{up}} - f_0, x - f_0 \rangle$, showing that this is smaller for update points than for test points. In the OUT case, we write d as d_{OUT}, which is distributed as

$$d_{\text{OUT}}$$
$$= -4\eta\left(\langle \mu_{\text{up}} - \mu, x - \mu \rangle + \|f_0 - \mu\|_2^2 - \langle x - \mu, f_0 - \mu \rangle - \langle \mu_{\text{up}} - \mu, f_0 - \mu \rangle\right)$$
$$= -4\eta\left(\frac{\sigma^2}{n_0}X_1 + \sigma^2\sqrt{\frac{n_0+n_1}{n_0 n_1}}(X_2 - X_3) + \frac{\sigma^2}{\sqrt{n_0 n_1}}(X_4 - X_5)\right),$$

where each of the X_i are independent samples from a Chi square distribution χ_d^2. The mean of d_{OUT} is $\frac{-4\eta}{n_0}d\sigma^2$, and its variance is $32d\eta^2\sigma^4\left(\frac{n_1+2n_0^2+2n_1n_0+2n_0}{n_0^2 n_1}\right) = O\left(d\eta^2\sigma^4\left(\frac{1}{n_0} + \frac{1}{n_1}\right)\right)$.

In the IN case, we write d as d_{IN}, distributed as

$$d_{\text{IN}}$$
$$= -4\eta\left(\langle \mu_{\text{up}} - \mu, x - \mu \rangle + \|f_0 - \mu\|_2^2 - \langle x - \mu, f_0 - \mu \rangle - \langle \mu_{\text{up}} - \mu, f_0 - \mu \rangle\right)$$
$$= -4\eta\left(\frac{1}{n_1}\|x - \mu\|_2^2 + \frac{\sigma^2}{n_0}X_1 + \frac{\sigma^2(n_1+1)}{n_1\sqrt{n_0}}(X_2 - X_3)\right.$$
$$\left. \frac{\sigma^2(n_1-1)}{n_1\sqrt{n_1}}(X_4 - X_5) + \frac{\sigma^2(n_1-1)}{n_1\sqrt{n_0 n_1}}(X_6 - X_7)\right),$$

where each of the X_i are independent samples from a Chi square distribution χ_d^2. The mean of d_{IN} is $\frac{-4\eta}{n_0}d\sigma^2 - \frac{4\eta}{n_1}\|x - \mu\|_2^2$, and its variance is $O\left(d\eta^2\sigma^4\left(\frac{1}{n_0} + \frac{1}{n_1}\right)\right)$.

Now, the difference in means for d_{OUT} and d_{IN} is $4\eta\|x - \mu\|_2^2/n_1$, which is distributed as $4\eta\chi_d^2/n_1$, which we can bound below by $d\eta\sigma^2/n_1$ with probability $> 1 - \exp(-d/16)$ [24] over the randomness of selecting x. Then we can consider a threshold of $\frac{1}{2}(\mathbb{E}[d_{\text{OUT}}] + \mathbb{E}[d_{\text{IN}}]) \ge \mathbb{E}[d_{\text{OUT}}] + 0.5d\eta\sigma^2/n_1$. The success at distinguishing between the IN and OUT case is then bounded below by $Acc > \Pr[d_{\text{OUT}} < \mathbb{E}[d_{\text{OUT}}] + 0.5d\eta\sigma^2/n_1]$. To measure this probability, notice that the variance of d_{OUT} is $Var[d_{\text{OUT}}] = O\left(\frac{d\eta^2\sigma^4}{n_1}\right)$, if we assume $n_1 < n_0$. Then there is some C_{OUT} for which the variance $Var[d_{\text{OUT}}] < \frac{C_{\text{OUT}}d\eta^2\sigma^4}{n_1}$. Then we can use Chebyshev's Inequality[4]

---

[4] For a random variable Y, $\Pr[|Y - \mathbb{E}[Y]| \ge k\sqrt{Var(Y)}] \le \frac{1}{k^2}$.
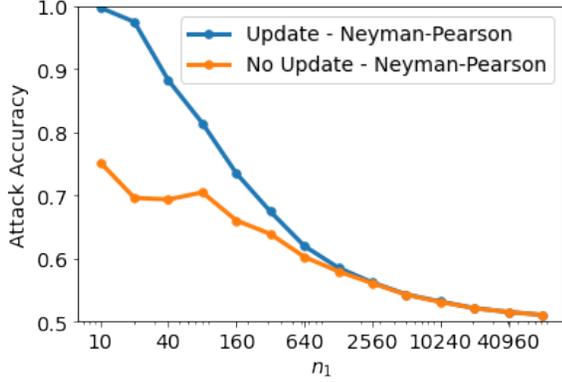
Figure 8: Demonstrating the power of updates. Here, we use $n_0 = 200$, $d = 250$, $\mu = 0$, $\sigma = 0.1$, and vary n_1 along the x axis, reporting performance averaged over 60 trials.

to bound our success probability from below as

$$
Acc = \Pr\left[d_{\text{OUT}} < \mathbb{E}[d_{\text{OUT}}] + \sqrt{Var[d_{\text{OUT}}]} \sqrt{\frac{d}{8C_{\text{OUT}}n_1}} \right]
$$

$$
\geq 1 - \frac{8C_{\text{OUT}}n_1}{d}.
$$

The analysis for the IN case is identical, although requires a different constant C_{IN}, due to the different variance term. Then, to guarantee $> 90\%$ accuracy as in the theorem statement, we can take $K = 80 \max(C_{\text{IN}}, C_{\text{OUT}})$. □

C EXTENDED EXPERIMENTS

Algorithm 3: Neyman-Pearson Optimal [34] Distinguisher for Mean Estimation

---
Data: Test Sample x, Estimated Mean $\hat{\mu}$, True Mean μ, True Variance σ^2, Dimensions d, Dataset Size n

Function NeymanPearson($x, \hat{\mu}, \mu, \sigma, d, n$):

$\quad p_{\text{IN}} = \left(\frac{n-1}{2\pi\sigma^2}\right)^{d/2} \exp\left(-\frac{n-1}{2\sigma^2}\|\hat{\mu} - \frac{n-1}{n}\mu - \frac{1}{n}x\|_2^2\right)$

$\quad p_{\text{OUT}} = \left(\frac{n}{2\pi\sigma^2}\right)^{d/2} \exp\left(-\frac{n}{2\sigma^2}\|\hat{\mu} - \mu\|_2^2\right)$

$\quad$ **return** $\mathbb{1}(p_{\text{IN}} > p_{\text{OUT}})$

Mean Estimation

To validate the improved performance of model updates for mean estimation, we test attacks on the mean estimation setup described earlier in the Appendix. For these experiments, we set $\mu = 0$, $n_0 = 200$, $d = 250$, $\sigma = 0.1$, and we vary n_1. We experiment an attack which leverages updates, and one which doesn't.

In the no update case, we run the Neyman-Pearson optimal attack [34], as described in Algorithm 3. This attack, when provided a $\hat{\mu}_1$, computes the PDF values $p_{\text{OUT}} = \mathcal{D}_{\text{OUT}}(\hat{\mu}_1)$ and $p_{\text{IN}} = \mathcal{D}_{\text{IN}}(\hat{\mu}_1)$, and returns IN ($g_0 = 1$) if $p_{\text{IN}} > p_{\text{OUT}}$ and OUT ($g_0 = 0$) otherwise. For Algorithm 3, we set $\hat{\mu} = \hat{\mu}_1$ and $n = n_0 + n_{\text{up}}$.
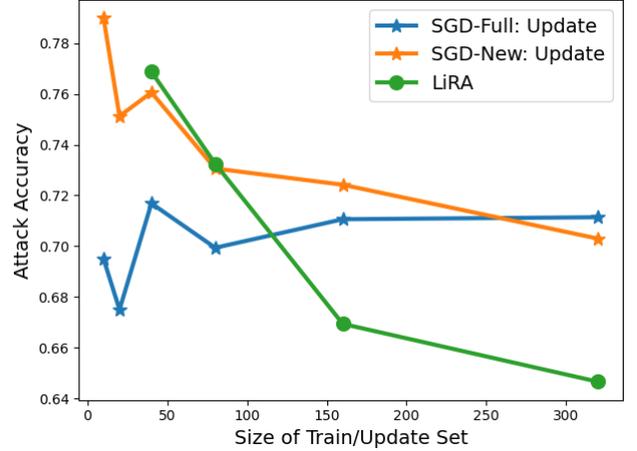

Figure 9: Membership inference on non-updated models trained on small datasets (SGD-Full and SGD-New) vs. models updated on small datasets (LiRA). Attacks on models updated on small datasets appear to be more resilient to dataset size than models trained on small datasets.

In the update case, we run the corresponding Neyman-Pearson optimal attack, which is equivalent to the no update Neyman-Pearson optimal attack, replacing $\hat{\mu}_1$ with $\hat{\mu}_{\text{Delta}}$. When provided $\hat{\mu}_1$ and $\hat{\mu}_0$, the attack computes the mean of D_1 as $\hat{\mu}_{\text{Delta}} = \frac{n}{n_1}\hat{\mu}_1 - \frac{n_0}{n_1}\hat{\mu}_0$, and then computes the PDF values $p_{\text{OUT}} = \mathcal{D}_{\text{Delta,OUT}}(\hat{\mu}_1)$ and $p_{\text{IN}} = \mathcal{D}_{\text{Delta,IN}}(\hat{\mu}_1)$, where the distributions $\mathcal{D}_{\text{Delta,OUT}}$ and $\mathcal{D}_{\text{Delta,IN}}$ are defined in the proof of Lemma B.3. Then the attack returns IN ($g_0 = 1$) if $p_{\text{IN}} > p_{\text{OUT}}$ and OUT ($g_0 = 0$) otherwise. For Algorithm 3, we set $\hat{\mu} = \hat{\mu}_{\text{Delta}}$ and $n = n_{\text{up}}$.

In Figure 8, we empirically evaluate these attacks. The immediate takeaway is that using model update never results in worse performance than not using updates. However, notice that this message holds in a variety of scenarios, even those which are not covered by our theoretical analysis. Because $d = 250$ and $n_0 = 200$, when $n_1 < 50$, we see that even when $d > n$, the model update attack outperforms the no update attack, a setting our analysis performs poorly in. Also, even when n_1 is much larger than n_0, we see the attack which uses updates does not perform poorly relative to the attack which does not. The improved performance from using model updates is robust to a wide range of parameter settings.

Section 5.2 Experiments - Single Update. We report the precision and recall for SGD-New attacks in Table 7. These precision values are even higher than precisions reached by SGD-Full presented in Table 3.

Section 5.2 Experiments - Comparison to Single Update Attacks with Small Datasets. A model trained on a small dataset is more likely to overfit than one trained on a larger dataset, naturally leading to increased membership inference risk. As a result, it is perhaps unsurprising that we find that larger update set sizes are less vulnerable to our attacks. Here, we experiment on FMNIST to see if the increased vulnerability on small datasets is different between the two settings: a model trained on a small dataset, or a

Dataset	$n_{up} = 0.01n_0$				$n_{up} = .08n_0$			
	Loss	Batch	Transfer	Test	Loss	Batch	Transfer	Test
FMNIST	.53/.11	.90/.18	.80/.48	.82/.38	.51/.10	.71/.14	.83/.09	.66/.20
CIFAR-10	.47/.09	.92/.18	.90/.06	.83/.46	.49/.10	.83/.17	.84/.14	.77/.36
Purchase100	.54/.11	.70/.14	.77/.09	.68/.18	.49/.10	.54/.11	.54/.22	.54/.12
IMDb	.50/.10	.83/.17	.70/.34	.75/.22	.52/.10	.70/.14	.74/.14	.69/.16

Table 7: Precision/recall results for single update models on SGD-New. Loss, Batch, Transfer, and Rank are defined in Table 2. We report for two n_{up} values.
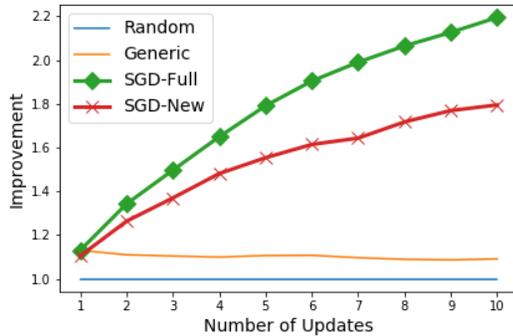
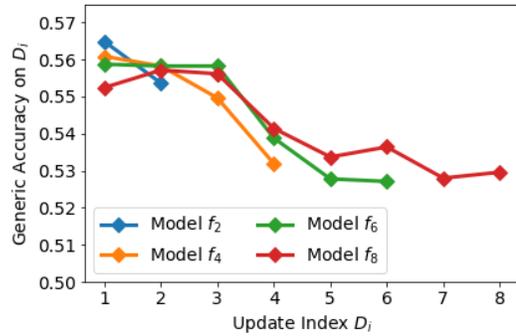
(a) Entry Inference Accuracy
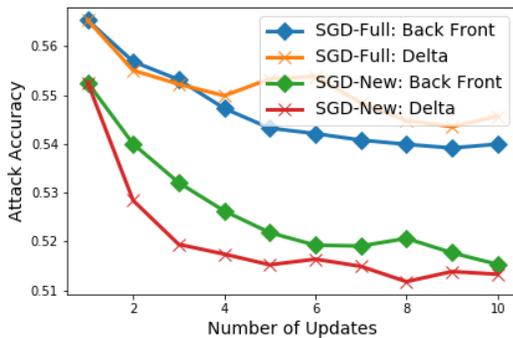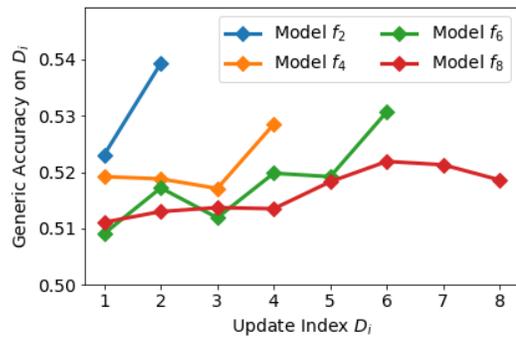

(b) Membership Inference Accuracy

Figure 10: Entry inference and membership inference accuracy of multiple update attacks on FMNIST with $n_{up} = 100$, using loss difference. As in Section 4.1, SGD-New attacks perform worse relative to SGD-Full.

(a) SGD-Full

(b) SGD-New

Figure 11: Membership inference accuracy for each update index used to train a given model f_i. Results shown for the Back-Front attack with loss difference on FMNIST with SGD-Full and SGD-New at $n_{up} = 100$.

model updated with a small dataset. The results of our experiment can be found in Figure 9. Here, we find that, while the standard LiRA attack is as strong as our adaptation on SGD-New, its performance drops much more quickly than our update attacks, so that the update setting leaks (according to our attacks) more than the fixed dataset setting. Furthermore, in practice, updating a model with a small dataset is likely to be tolerable, as it will not worsen an already accurate model, while training a model entirely on a small dataset will likely lead to an inaccurate model, which is unlikely to be deployed, making our finding here more worrisome.

Section 5.2 Experiments - Disparate Impact of Single Update Attacks. To evaluate the disparate impact of our attacks in the

absence of distribution shift, we measure per-class attack accuracy on FMNIST in Figures 12a and 12b. We report single update SGD-Full and SGD-New accuracies with $n_{up} \in \{10, 160\}$. While there are no strong differences, Class 1 (trouser), is reliably easy to attack, with perfect accuracy at $n_{up} = 10$ for both SGD-Full and SGD-New. Class 7 (sneaker) also has somewhat higher vulnerability to our attacks.

Section 5.5 Experiments - Multiple Updates. We present in Figures 10a and 10b the results for attacks on FMNIST when $n_{up} = 100$, to compare against the results from Section 4.3, which use $n_{up} = 10$.

In Figure 10a, we show that, at larger update set sizes, attacks on SGD-Full can outperform attacks on SGD-New, in line with experiments from Section 4.1. The entry inference accuracy of attacks with $n_{up} = 100$ is also smaller than for attacks with $n_{up} = 10$, which also corroborates our experiments in Section 4.1.

In Figure 10b, we make similar observations to those in Figure 10a. Attacks are less powerful at large update set sizes, and attacks on SGD-Full perform better relative to attacks on SGD-New.

Section 5.6 Experiments - Distribution Shift. We now reproduce our subpopulation shift findings in the CINIC-10 dataset. Here, we use CINIC-10 to construct a dataset of ImageNet images corresponding to 9 classes from CIFAR-10 (all but bird). Again using the BREEDS framework, and we select source and target subclasses which should have "Easy" transfer based on how close they are in the ImageNet sysnet tree (we select leaf classes which share a parent node), and we select "Hard" subclasses by selecting leaf classes which only share the CIFAR-10 label, making them far apart in the tree. We detail these classes in Table 8. We also need to make sure these classes are well represented, so we only select subclasses with more than 200 images in CINIC-10 (due to the large number of bird classes, we could not find bird subclasses fitting this requirement, so we omit it). We then use an update set size of 900, and vary the shift ratio α between 0.2, 0.6, 1.0, as before. We report accuracy and precision of our attacks in Table 9. Interestingly, the correlations we observed in our CIFAR-2 dataset for BREEDS do not hold up, similarly to the covariate shift case with CINIC-10. This is likely to be because the attacks are already highly successful without much shift, making it harder for these shifts to improve upon them.
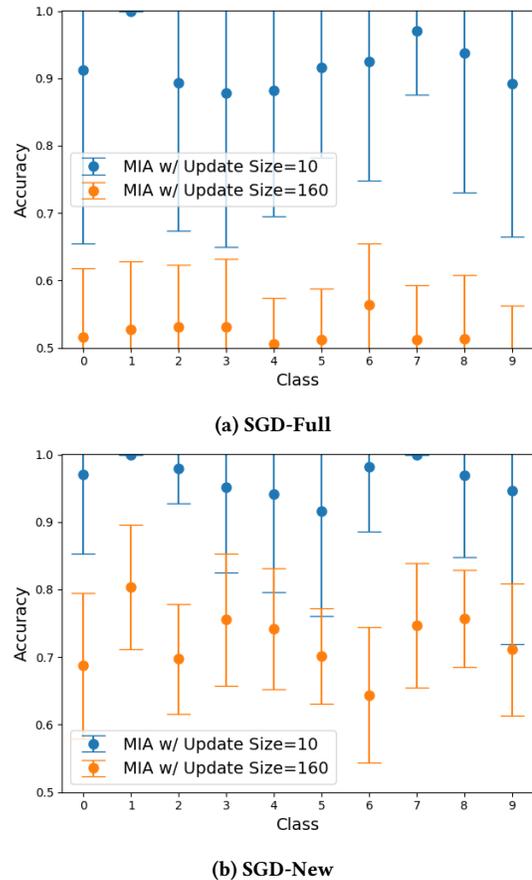

(a) SGD-Full

(b) SGD-New

Figure 12: Membership inference accuracy for each class in the FMNIST dataset on $n_{up} = 10$, and $n_{up} = 160$.

CIFAR-10 class	ImageNet Source	Easy Target	Hard Target
Airplane	airbus	widebody aircraft	amphibious aircraft
Automobile	race car	stock car	limousine
Cat	Abyssinian cat	Siamese cat	lion
Deer	musk ox	bison	white-tailed deer
Dog	sled dog	husky	Newfoundland dog
Frog	green frog	leopard frog	European toad
Horse	carthorse	shire horse	Arabian
Ship	dugout canoe	kayak	speedboat
Truck	passenger van	delivery truck	fire truck

Table 8: Classes used for the CINIC subpopulation shift experiment.

Metric	Shift	$\alpha = 0.2$		$\alpha = 0.6$		$\alpha = 1.0$	
		DIFF	RATIO	DIFF	RATIO	DIFF	RATIO
Accuracy	Easy	0.587	0.643	0.566	0.587	0.604	0.625
	Hard	0.593	0.630	0.583	0.592	0.607	0.630
Prec./Recall	Easy	.75/.15	.69/.14	.79/.16	.77/.15	.88/.18	.87/.17
	Hard	.88/0.18	.83/.17	.84/0.17	.85/.17	.89/.18	.88/.18

Table 9: Accuracy and precision/recall of attacks after sub-population shift on SGD-New with CINIC-10.

Metric	Shift	SGD-New						SGD-Full					
		$\alpha = 0.2$		$\alpha = 0.6$		$\alpha = 1.0$		$\alpha = 0.2$		$\alpha = 0.6$		$\alpha = 1.0$	
		Diff	Ratio	Diff	Ratio	Diff	Ratio	Diff	Ratio	Diff	Ratio	Diff	Ratio
Accuracy	Hard	0.551	0.559	0.602	0.634	0.600	0.687	0.574	0.597	0.578	0.585	0.595	0.666
	Easy	0.685	0.685	0.574	0.581	0.560	0.562	0.597	0.606	0.570	0.570	0.598	0.598
Prec./Recall	Hard	.73/.15	.90/.18	.94/.19	.96/.19	.98/.20	.94/.19	.63/.13	.76/.15	.69/.14	.95/.19	.80/.16	.94/.19
	Easy	.64/.13	.74/.15	.58/.12	.90/.18	.71/.14	.94/.19	.54/.11	.64/.13	.47/.09	.70/.14	.62/.12	.86/.17

Table 10: (Q5) Accuracy and precision/recall of attacks after subpopulation shift. Ratio, Diff stand for ScoreRatio, ScoreDiff, respectively, with loss score. Ratio performs best, a Hard distribution shift results in better attacks as α increases, and SGD-New is typically more vulnerable than SGD-Full.