

Understanding Unintended Memorization in Federated Learning

Om Thakkar Swaroop Ramaswamy Rajiv Mathews Françoise Beaufays

Google LLC,

Mountain View, CA, U.S.A.

{omthkkr, swaroopram, mathews, fsb} @google.com

June 16, 2020

Abstract

Recent works have shown that generative sequence models (e.g., language models) have a tendency to memorize rare or unique sequences in the training data. Since useful models are often trained on sensitive data, to ensure the privacy of the training data it is critical to identify and mitigate such *unintended* memorization. Federated Learning (FL) has emerged as a novel framework for large-scale distributed learning tasks. However, it differs in many aspects from the well-studied *central learning* setting where all the data is stored at the central server. In this paper, we initiate a formal study to understand the effect of different components of canonical FL on unintended memorization in trained models, comparing with the central learning setting. Our results show that several differing components of FL play an important role in reducing unintended memorization. Specifically, we observe that the clustering of data according to users—which happens by design in FL—has a significant effect in reducing such memorization, and using the method of Federated Averaging for training causes a further reduction. We also show that training with a strong user-level differential privacy guarantee results in models that exhibit the least amount of unintended memorization.

1 Introduction

There is a growing line of work [FJR15, WFJN16, SSSS17, CLK⁺18, SS19] demonstrating that neural networks can leak information about the underlying training data in unexpected ways. In particular, many of these works show that generative sequence models, which include commonly-used language models, are prone to *unintentionally memorize* rarely-occurring phrases in the data. Large-scale learning often involves training over sensitive data, and such memorization can result in blatant leaks of privacy (e.g., [Mun19]). Thus, for any novel learning framework of interest, it is crucial to test the resilience of models trained in the framework against such memorization. Techniques to mitigate memorization must be identified to ensure the privacy of training data.

The framework of Federated Learning (FL) [MMR⁺17, MR17] has emerged as a popular approach for training neural networks on a large corpus of decentralized on-device data (e.g., [KMRR16, KMY⁺16, BIK⁺17, HRM⁺18, BEG⁺19]). FL operates in an iterative fashion: in each round, sampled client devices receive the current global model from a central server to compute an update on their locally-stored data, and the server aggregates these updates using the Federated Averaging algorithm [MMR⁺17] to build a new global model. A hallmark of FL is that each participating device only sends model weights to the central server; raw data never leaves the device, remaining locally-cached. Although this, by itself, is not sufficient to provide formal privacy guarantees for the training data, it is important to note that the canonical setting of FL [MMR⁺17] does differ in many aspects from the well-studied *central learning* setting where all the data is stored at a central server. In this work, we initiate a formal study to understand the effect of the different components of FL, compared to the central learning setting, on unintended memorization in trained models.

We also investigate the effect of using a training procedure, with a formalized privacy guarantee, on such memorization. To this end, we use Differential Privacy (DP) [DMNS06, DKM⁺06a], which has become the standard for performing learning tasks over sensitive data. DP has been adopted by companies like Google [EPK14, BEM⁺17, EFM⁺20], Apple [App17], Microsoft [DKY17], and LinkedIn [RSP⁺20], as well as the US Census Bureau [KCK⁺18]. Intuitively, DP prevents an adversary from confidently making any conclusions about whether any particular user’s data was to train a model, even while having access to the model and arbitrary external side information.

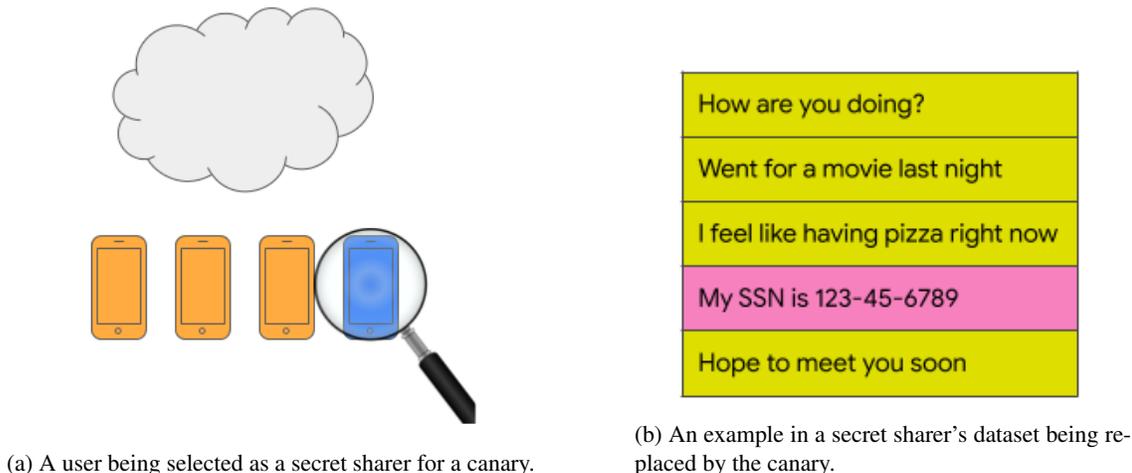


Figure 1: An illustration of our federated secret-sharer framework, using the canary phrase “My SSN is 123-45-6789” with a user-selection probability $p_u = 0.25$, and an example-replacement probability $p_e = 0.2$. Thus, in expectation, 1 out of every 4 users will be a secret sharer for this canary, and 1 out of every 5 examples in each such secret sharer’s data will be replaced by the canary.

We build on the “secret sharer” framework from [CLK⁺18] that was designed to measure the unintended memorization in generative models. At a high-level, *out-of-distribution* examples (called canaries) are inserted into a training corpus, and a model trained on this corpus is then evaluated using various techniques to measure the extent to which the model has *memorized* the canaries. Since datasets in FL are inherently partitioned according to users, we adapt this framework to the FL regime by introducing two parameters to control the presence of a canary in such settings. An illustration of our federated secret sharer framework is shown in Figure 1. Given a canary with parameters p_u and p_e , we let p_u be the probability with which each user in a dataset is selected to be a secret sharer of the canary (Figure 1a), whereas p_e denotes the probability with which each example in such a secret sharer’s data is replaced by the canary (Figure 1b).

Our empirical evaluations (Section 4) for this paper demonstrate the following key contributions. First, we show clustering the training data according to users, (i.e., having heterogeneous users) has a significant effect in reducing unintended memorization. Note that such a clustering of the data happens by design in FL settings. Next, given data clustered according to users, we show that replacing the learning optimizer from SGD to Federated Averaging provides a further reduction in such memorization. Lastly, we demonstrate that training in FL with a strong user-level DP guarantee results in models that exhibit the least amount of unintended memorization.

Organization of the paper: We provide a formal definition of differential privacy in Section 2. In Section 3, we identify the main components separating central learning from the canonical setting of FL. Section 4 contains the results of our empirical evaluation. We state the conclusions of this work in Section 5.

1.1 Related Work

There is a wide variety of work demonstrating unexpected information leakage from datasets in unexpected ways: [DN03] design a general *reconstruction attack* whereas there are other works (e.g., [HSR⁺08, SOJH09, BUV14, DSS⁺15, SSSS17, MSDS18]) that design *membership inference attacks*. Apart from [CLK⁺18] (which this work builds upon), other works [SS19] have also studied memorization in generative text models.

The FL paradigm, which is a major focus of this work, has been used to train multiple production scale models [HRM⁺18, RMRB19, CSM⁺]. We refer the reader to [KMA⁺19] which provides an excellent overview of the state-of-the-art in the field, along with a suite of interesting open problems.

This work also studies the effectiveness of a user-level DP guarantee in reducing unintended memorization. While many works on DP focus on *record-level* DP guarantees (which usually cannot be directly extended to strong user-level DP guarantees), recent works (e.g., [MRTZ17, JTT18, AMR⁺20, TAM19]) have designed techniques tailored to user-level DP guarantees.

2 Background on Differential Privacy

To establish the notion of differential privacy [DMNS06, DKM⁺06b], we first define neighboring datasets. We will refer to a pair of datasets D, D' as neighbors if D' can be obtained by the addition or removal of all the examples associated with one user from D .

Definition 2.1 (Differential privacy [DMNS06, DKM⁺06b]). *A randomized algorithm \mathcal{A} is (ϵ, δ) -differentially private if, for any pair of neighboring datasets D and D' , and for all events \mathcal{S} in the output range of \mathcal{A} , we have*

$$\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^\epsilon \cdot \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta$$

where the probability is taken over the random coins of \mathcal{A} .

For meaningful privacy guarantees, ϵ is assumed to be a small constant, and $\delta \ll 1/|D|$.

To train models with DP guarantees, we follow the variant of DP Federated Averaging (DP-FedAvg) [MRTZ17] used in [AMR⁺20], where the only change is sampling fixed-sized minibatches in each training round.¹ We provide more details of this technique, its privacy analysis, and a pseudo-code for reference in Appendix A.

3 Contrasting Federated Learning with Central Learning

Now, we take a deeper look at how the well-studied central learning framework differs from the canonical setting of Federated Learning (FL) [MMR⁺17]. Specifically, we are interested in differences that might have an effect on unintended memorization. We identify three such components²:

1. *Learning Technique*: In central learning, the model is updated via an SGD step on a minibatch of records. In the canonical setting of FL, a model update typically corresponds to Federated Averaging over a minibatch of users: an average of the differences between the current model and the model obtained after several SGD steps on the local data of a user.
2. *Data Processed per Update*: Central learning typically ingests data as *records*. For instance, the data for training a language model in central learning could be stored a set of sentences. On the other hand, FL operates at the granularity of a *user*, with each user having their own set of records locally. In our example, the data in an FL setting would be a set of users, with each user having their own set of sentences locally. Typically, the amount of data processed per model update in central learning is much smaller in comparison to FL.

¹Due to a technical limitation of the simulation framework, our experiments use sampling with replacement instead of without replacement; this should have negligible impact on the metrics of the trained models.

²We do not discuss unbalanced datasets, i.e., the fact that users can have varying amounts of local data, since FedAvg deals with such imbalances by weighing each client update according to the size of its local data.

3. *Independent and Identically Distributed (IID) Data*: To reduce variance in learning, the data in central learning is shuffled before training (and/or each update involves a randomly sampled minibatch). Thus, each minibatch can be estimated to be drawn IID from the data. However, the data in FL is naturally grouped according to heterogeneous users, resulting in non-IID data even though each minibatch of users may be randomly sampled.

Unintended memorization typically corresponds to the model memorizing information pertaining to a specific individual, or a very small group of individuals in the dataset. Given the above-stated differences between the two settings, each of the three components could *intuitively* play a significant role on such memorization. For instance, it is conceivable that the higher amount of data processed per update in FL is better at concealing a targeted *signal*, since each model update is averaged over effectively many more records than in central learning. Moreover, signals restricted to a small group of users may be encountered less frequently during FL training, since the data is clustered according to users. On the flip side, such heterogeneity (non-IIDness) of the data could result in a *boosted* targeted signal whenever it is encountered in training. Additionally, since FedAvg involves several steps of SGD over user data in an update, the signal boost could be magnified due to factors such as “client-drift” [KKM⁺19]. Thus, it is unclear what effect each (or, any combination) of these differing components may have on such memorization in models trained via FL.

4 Empirical Evaluation

4.1 Experimental Setup

Model Architecture: We use a word-level language model based on a recurrent neural network. Our model architecture mirrors the one used in [HRM⁺18]. The model comprises a CIFG-LSTM [SSB14] with shared weights between the input embedding layer and the output projection layer. A vocabulary of size $\approx 10\text{K}$ is used for both input and output vocabularies. The input embedding dimension and the output projection size are set to 96, and a single-layer CIFG cell with 670 units is used as the recurrent layer. The overall number of parameters in the model is $\approx 1.3\text{ M}$.

Dataset Construction: We create a modified version of the Stack Overflow questions and answers dataset [Ove18] hosted by TensorFlow Federated [IO19]. The original dataset is keyed by a unique ID associated with the user who typed a question/answer. This ID can be used to separate the dataset into users for Federated Learning. To compare the extent of memorization among different canary configurations in our federated secret sharer framework (Figure 1), we modify the dataset to have balanced users as follows. First, we discard all users that have fewer than 2K words. For the remaining users, we split the data associated with each user into (multiple) “balanced” users having $\approx 2\text{K}$ words each. We also discard any “remainder data”, i.e, if a user originally has 4030 words, we create 2 balanced users having 2K words each and discard the remaining 30 words. The modified version of the dataset has $\approx 392\text{K}$ users, each with $\approx 2\text{K}$ words, and a total of $\approx 93\text{K}$ examples. For an IID version of this dataset, we randomly shuffle all the records in the dataset, and create *synthetic* users having $\approx 2\text{K}$ words assigned sequentially from the shuffled records.

Canary Construction: Since our model is a word-level language model, we follow the methodology used by [CLK⁺18] for their experiments with the Gmail Smart Compose model [CLB⁺19]. We opt for inserting 5-word canaries instead of 7-word canaries because our model is smaller³ than the Smart Compose model, and it is not as efficient at encoding longer contexts. Our canaries are constructed by choosing each word uniformly at random from the 10K model vocabulary. To measure *unintended* memorization, it is important to ensure that canaries comprise of out-of-distribution phrases. Randomly sampled 5-word canaries produces phrases that are highly unlikely to be in-distribution. For instance, our inserted canaries consist of phrases like “qualifier winded alike configs txt”, “gentle originally saml likewise notified”, “dns scoring investigate compact auto”, etc.

Canary Insertion: Using our federated secret sharer framework (Figure 1), we insert various canaries into the dataset. Recall that each canary is parameterized by two parameters: a user-selection probability denoted by p_u , and an

³Our model is smaller because it is designed to run on mobile devices [HRM⁺18].

example-replacement probability denoted by p_e . We use Poisson sampling for both user-selection and example-replacement. We insert canaries with configurations in the cross product of $p_u \in \{1/5K, 3/50K, 1/50K\}$ and $p_e \in \{1\%, 10\%, 100\%\}$. To limit the variance in our measurements, we insert 10 different canaries for each (p_u, p_e) configuration. These parameters result in the insertion of 90 different canaries, each shared among at least 4 users and at most 103 users in the dataset, with their overall insertion frequencies ranging from 13 to 24.5K.

Evaluation Methods: We use two methods of evaluation to understand the extent of unintended memorization exhibited by a model.

1. **Random Sampling (RS) [CLK⁺18]** It is a sampling-based method that captures how strongly the model favors the canary as compared to random chance. More specifically, we first define the log-perplexity of a model θ on a sequence $s = s_1, \dots, s_n$ given context p as $P_\theta(s|p) = \sum_{i=1}^n \left(-\log \Pr_\theta(s_i|p, s_1, \dots, s_{i-1}) \right)$. Next, this method requires i) a model θ , ii) an inserted canary $c = (p|s)$ where p is a prefix and s is the remaining sequence, and iii) a set R that consists of $|s|$ -length sequences with each word sampled uniformly at random from the vocabulary. Now, the rank of the canary c can be defined as $\text{rank}_\theta(c; R) = |\{r' \in R : P_\theta(r'|p) \leq P_\theta(s|p)\}|$. Using this method, we consider a canary c as “memorized” by a model θ if given a set R , we get that $\text{rank}_\theta(c; R) = 1$. For our experiments, we consider the size of R to be $2M$.
2. **Beam Search (BS)** Given a prefix, and the total length of the phrase to be extracted, this method conducts a greedy beam search on a model. As a result, this method functions without the knowledge of the whole canary. Using this method, we consider a canary as “memorized” by a model if given a prefix, the canary is the most-likely continuation. For our experiments, we use a beam search width of 5 for this method.

We perform measurements for both the methods using a prefix length of 1. It is important to observe that since the BS method can function without complete knowledge of the inserted canary, it requires *strictly* less information than the RS method. Thus, if a canary does not show up as memorized via the RS method, it will almost certainly not show up as memorized via the BS method.⁴

Along with evaluating for unintended memorization, we measure the utility of a model with accuracy (Recall@1) and perplexity on the test partition of the unmodified StackOverflow dataset.

4.2 Empirical Results

We present the results of our experiments on evaluating unintended memorization under different training regimes ranging from the well-studied central learning setting to canonical FL. For all our experiments that use SGD, we train models for 37.5M steps, whereas we train for 8000 rounds for the experiments using FedAvg. For the largest minibatch sizes used in both the settings (256 records for SGD, and 5000 users for FedAvg), these checkpoints correspond to training for 100 epochs. Table 1 shows the number of canaries (out of 90) that show up as memorized via both the RS and BS techniques. For both the methods, we also provide the lowest canary configuration (ordered by expected insertion frequency, and then by expected number of users sharing the canary) in terms of (p_u, p_e) that had at least one of the ten inserted canaries as memorized. It is important to note that the utility of all the evaluated models is similar; the Recall@1 varies by less than 1% across all the models. We conduct experiments for the following training regimes.

1. **Central Learning:** First, we present the results for the central training regime where data is accessed as records, shuffled (IID), and SGD is used as the update step. This setting is the closest to the one evaluated in prior work [CLK⁺18]. We start by keeping the minibatch size $b_r = 32$ records with a tuned learning rate of 0.005, and we double the minibatch size until it is 256.⁵ For all the configurations, we observe that 52-54 canaries show up as memorized via the RS method, and 42-45 via the BS method.

⁴Since beam search is essentially a greedy algorithm, it is possible to construct contrived examples where the BS method classifies a phrase as memorized whereas the RS method does not. However, given the typical log-perplexities of trained models on random phrases, the chance of observing such cases is very small.

⁵In Appendix B, we provide results for the setting where we increase the learning rate with the minibatch size, and stop training at 10 epochs for every minibatch size. We see that the amount of unintended memorization is relatively unaffected, but the model utility significantly decreases with increasing minibatch size.

2. **FedAvg in Central Learning Setup:** Next, we evaluate the regime where the learning technique of SGD in the setup for central learning is replaced by FedAvg. Since data is operated at the granularity of “users” for FedAvg, we create *synthetic users*, each containing 2000 words, from the shuffled dataset for SGD. As a consequence, in this experiment we effectively have FedAvg operating over IID users. For all the configurations, 65-69 canaries show up as memorized via the RS method, and 56-58 via the BS method.
3. **Non-IIDness in Central Learning:** In this experiment, we evaluate the effect of Non-IID data in the setup of central learning. Since data is accessed as records for SGD but the natural grouping in the dataset is by heterogeneous users, we run the steps of SGD sequentially through the user-grouped variant of the dataset in each epoch of training. Here, we observe that the grouping of the data according to users exhibits a significant reduction in the unintended memorization by a trained model: for all the configurations, we observe that 37-51 canaries show up as memorized via the RS method, and 19-39 via the BS method.
4. **Federated Learning:** Now, we evaluate the standard setting of FL where data is grouped by users (Non-IID), and FedAvg is used as the update step. The amount of unintended memorization in this setting significantly drops compared to any of the three settings evaluated above: for all the configurations, we observe that only 19-26 canaries show up as memorized via the RS method, whereas at most 2 canaries are extracted via the BS method. Notice that for the RS method, the lowest canary configuration that gets memorized is when 1 in every 5K users shares the canary, and further, for the BS method, only when all of the data of such users is replaced by the canary.

| Setting Data, Optimizer, Batch Size | RS /90 | RS Lowest (p_u, p_e) | BS /90 | BS Lowest (p_u, p_e) | Acc. % | Perp. |
|--|-----------|-----------------------------|-----------|-----------------------------|-----------|-------|
| IID, SGD, $b_r = 32$ | 54 | 1/50K, 10% | 42 | 3/50K, 10% | 24 | 62.2 |
| IID, SGD, $b_r = 64$ | 54 | 1/50K, 10% | 42 | 3/50K, 10% | 24.1 | 61.5 |
| IID, SGD, $b_r = 128$ | 52 | 1/50K, 10% | 45 | 3/50K, 10% | 24 | 62 |
| IID, SGD, $b_r = 256$ | 53 | 1/50K, 10% | 43 | 1/50K, 10% | 24.1 | 61.1 |
| IID, FedAvg, $b_u = 500$ | 66 | 1/50K, 10% | 56 | 1/50K, 10% | 24.6 | 57.5 |
| IID, FedAvg, $b_u = 1000$ | 69 | 3/50K, 1% | 58 | 1/50K, 10% | 24.6 | 57.3 |
| IID, FedAvg, $b_u = 2000$ | 67 | 3/50K, 1% | 56 | 1/50K, 10% | 24.6 | 57.4 |
| IID, FedAvg, $b_u = 5000$ | 65 | 3/50K, 1% | 58 | 1/50K, 10% | 24.6 | 57.3 |
| Non-IID, SGD, $b_r = 32$ | 37 | 1/50K, 10% | 19 | 3/50K, 10% | 23.7 | 64.3 |
| Non-IID, SGD, $b_r = 64$ | 49 | 1/50K, 10% | 36 | 1/50K, 10% | 24.1 | 61.8 |
| Non-IID, SGD, $b_r = 128$ | 48 | 1/50K, 10% | 34 | 1/50K, 10% | 24.1 | 61.5 |
| Non-IID, SGD, $b_r = 256$ | 51 | 1/50K, 10% | 39 | 3/50K, 10% | 24.1 | 61.3 |
| Non-IID, FedAvg, $b_u = 500$ | 21 | 1/5K, 1% | 0 | - | 24.4 | 58.8 |
| Non-IID, FedAvg, $b_u = 1000$ | 23 | 1/5K, 1% | 1 | 1/5K, 100% | 24.3 | 59.5 |
| Non-IID, FedAvg, $b_u = 2000$ | 19 | 1/5K, 1% | 1 | 1/5K, 100% | 24.5 | 58.3 |
| Non-IID, FedAvg, $b_u = 5000$ | 26 | 1/5K, 1% | 2 | 1/5K, 100% | 24.5 | 58.2 |

Table 1: Results for the number of inserted canaries (out of 90) memorized via the RS and BS methods, the lowest (by insertion frequency) canary configurations that show up as memorized for each of the methods, and utility metrics for various models evaluated at 37.5M steps when sampling records (SGD), and 8000 rounds when sampling users (FedAvg).

4.2.1 Training with user-level DP

Next, we evaluate the effect of training with a guarantee of DP [DMNS06, DKM⁺06b], on the unintended memorization in a trained model. To be able to provide the strongest DP parameters while obtaining reasonable utility from the trained models, we conduct experiments only for our largest minibatch size of 5000 users per training round. The results are presented in Table 2.

Only Clipping: To bound the contribution by each participating user, DP-FedAvg clips each user update before aggregating them from a minibatch of users and adding calibrated noise to guarantee DP. Following [CLK⁺18], we present results (rows containing “FedAvg+Clip” in Table 2) for the case when user updates are clipped to a value of 0.2, but no noise is added. This results in an (∞, δ) -DP guarantee for any $\delta \in (0, 1)$. This experiment helps us observe the effect of only clipping on the unintended memorization exhibited by trained models. With IID data, for the setting evaluated in Section 4.2 (8000 rounds, i.e., 100 epochs for minibatch size of 5000 users), we observe that the RS method extracts 58 canaries with clipping, which is 7 fewer canaries when compared to without clipping. The BS method extracts 49, which is 9 fewer than without clipping. If we pick a model after training for 10 epochs, the amount of memorization significantly reduces to 28 canaries via the RS method, and 18 via the BS method, but the utility of the resulting model is also significantly reduced. For Non-IID data, we observe a similar trend but it is more pronounced: the RS method extracts 11 canaries with clipping, which is 15 fewer canaries when compared to without clipping, whereas the BS method is not able to extract any of the inserted canaries. For the measurement at 10 epochs of training, both of our methods are able to extract no canary, but the utility of the model is reduced as well for this case.

| Setting ($b_u = 5000$) Data, Optimizer, Epochs | RS /90 | RS Lowest (p_u, p_e) | BS /90 | BS Lowest (p_u, p_e) | Acc. % | Perp. |
|---|-----------|-----------------------------|-----------|-----------------------------|-----------|-------|
| IID, FedAvg, 100 | 65 | 3/50K, 1% | 58 | 1/50K, 10% | 24.6 | 57.3 |
| IID, FedAvg+Clip, 100 | 58 | 1/50K, 10% | 49 | 1/50K, 10% | 24.2 | 60 |
| IID, DP-FedAvg, 100 | 48 | 1/50K, 10% | 42 | 3/50K, 10% | 23.9 | 63 |
| IID, FedAvg+Clip, 10 | 28 | 1/50K, 100% | 18 | 3/50K, 100% | 21.9 | 83.2 |
| IID, DP-FedAvg, 10 | 26 | 1/50K, 100% | 18 | 3/50K, 100% | 21.8 | 84.2 |
| Non-IID, FedAvg, 100 | 26 | 1/5K, 1% | 2 | 1/5K, 100% | 24.5 | 58.2 |
| Non-IID, FedAvg+Clip, 100 | 11 | 1/5K, 10% | 0 | - | 24 | 61.5 |
| Non-IID, DP-FedAvg, 100 | 12 | 1/5K, 10% | 0 | - | 23.3 | 68.5 |
| Non-IID, FedAvg+Clip, 10 | 0 | - | 0 | - | 20.8 | 95.9 |
| Non-IID, DP-FedAvg, 10 | 0 | - | 0 | - | 20.7 | 97.1 |

Table 2: Unintended memorization, lowest (by insertion frequency) canary configuration memorized, and utility for models trained with Clipping/DP and 5000 users/round. For 100 epochs, the models trained with DP-FedAvg satisfy $(18.8, 10^{-7})$ -DP, and $(5.6, 10^{-7})$ -DP for 10 epochs.

DP: For obtaining a strong user-level DP guarantee, we add Gaussian noise with a noise multiplier of 1 to the per-round aggregate of clipped updates, which results in $(18.8, 10^{-7})$ -DP for 100 epochs of training. We also provide measurements for 10 epochs of training, which results in a stronger $(5.6, 10^{-7})$ -DP guarantee. Compared to the results with only clipping, we observe a significant drop in memorization for the setting with the highest amount of memorization with only clipping, i.e., the setting with IID data evaluated at 100 epochs.

4.3 Discussion

1. **Effect of clustering data according to users:** The results from our experiments strongly indicate that clustering data according to users significantly reduces unintended memorization. This is evident by considering the measurements in Table 1 in pairs where the only differing component among them is whether the data is IID or not. The number of epochs taken over the dataset to train the models on which we measure memorization is the same for any particular minibatch size, irrespective of whether the data is IID. Thus, the number of times the inserted canaries were encountered during training is also comparable. However, the amount of memorization observed is always lower when the data is Non-IID. This effect is more pronounced in the settings where FedAvg is used as the training method. For instance, for a minibatch size $b_u = 500$ users, training with FedAvg on IID data results in 66 canaries showing up as memorized via the RS method, and 56 via the BS method. However, the same configuration on Non-IID data results in the RS method classifying only 21 canaries as memorized, and the BS method not being able to extract any of the inserted canaries even after 8000 rounds of training. In addition to the data being clustered, the inserted canaries are clustered as well, which seems to play a crucial

role in reducing such memorization. It is important to note that the utility of the models is similar for non-IID data even as the memorization drops significantly.⁶

2. **Effect of varying data processed per update:** From the results presented in Table 1, fixing the optimizer to be from SGD/FedAvg and the data to be IID/non-IID, we do not observe any significant effect of varying the batch size, i.e., the data processed per update, on the unintended memorization of a model.
3. **Effect of training heterogeneous user data with FedAvg and *larger* minibatches:** The smallest minibatch size used for our experiments using FedAvg is 500 users per round, and as each user contains ≈ 250 records, the *effective* minibatch size for this setting is $\approx 125K$ records. In comparison, the largest minibatch size we are able to conduct training for our experiments using SGD is 512 records. Focusing our attention on the results in Table 1 using Non-IID data, we find that using FedAvg as an optimizer⁷, and consequently having larger effective minibatches per round training causes a significant reduction in unintended memorization when compared to training with SGD.⁸ Even from the lowest canary configurations memorized in both the RS and the BS methods, it can be seen that this setting can tolerate a larger frequency of inserted canaries and/or canaries shared among a higher proportion of users.
4. **Effect of adding Differential Privacy to FL training:** Some of our inserted canaries consist of being shared by as many as 103 users (with more than 24.5K occurrences in the training data). By definition, a user-level DP guarantee is intended to be resilient to changes in the trained models w.r.t. any one user’s data. Moreover, it is clear from the results in Table 1 that the models trained in the FL regime exhibit the least amount of unintended memorization. In spite of these, we observe that training with a user-level DP guarantee for our largest minibatch size of 5000 users results in a significant further reduction in such memorization. With Non-IID data (which is what we expect naturally in a real-world implementation of FL), both of our RS/BS methods fail to extract any of the inserted canaries at 10 epochs of training. Here, we obtain a guarantee of $(5.6, 10^{-7})$ -DP. Even at 100 epochs of training, where the guarantee becomes $(18.8, 10^{-7})$ -DP, only the RS method is able to classify 12 canaries as memorized. For the lowest canary configuration memorized in the RS method, notice that this setting can tolerate a magnitude larger frequency insertion of canaries than the setting without DP-FedAvg. It may not be surprising to believe the argument that as more privacy-preserving noise is added during training, less unintended memorization takes place. However, our results are noteworthy as, in spite of our DP models exhibiting the least amount of unintended memorization, they also provide a utility comparable to that of the models trained in Section 4.2, along with providing a strong user-level guarantee of $(18.8, 10^{-7})$ -DP.

5 Conclusion

In this work, we conduct a formal study to understand the effect of the different components of Federated Learning, on the unintended memorization in trained models, as compared to the well-studied central learning. From our results, we observe that the components of FL exhibit a synergy in reducing such memorization. In particular, user-based heterogeneity of data (which occurs as a natural consequence in the FL setting) has a significant effect in the reduction, and training using Federated Averaging reduces it further. Moreover, we observe the least amount of memorization in the models where we train in FL with strong user-level differential privacy guarantees.

Recent work [KKM⁺19] has shown that, in general, such heterogeneity in the training data can result in a slower and unstable convergence due to factors such as “client-drift”. For all of the experiments with non-IID data, we observe that the utility of the trained models is comparable to those trained on IID data, and we leave further exploration into why client-drift may not play a significant role in our experiments for future work. Lastly, the secret-sharer line of methods for measuring unintended memorization operate at the granularity of a record. For future work, it will be

⁶It might appear from Table 1 that using FedAvg consistently provides better utility than SGD. However, SGD is sensitive to the tuning of the learning rate parameter [MMR⁺17], and with further fine-tuning we expect SGD to provide the same utility as FedAvg.

⁷We also conducted experiments using Momentum [Qia99] and Adam [KB15] optimizers, but did not observe a strong effect on reducing memorization while maintaining comparable utility.

⁸Looking at the same set of results for IID data, the trend seems to be moving in the direction of increasing memorization. However, since the magnitude of the effect is much smaller, we deem that further investigation is required for this case, which leave for future work.

interesting to design stronger attacks targeting data at the granularity of a user, and measure the resilience of models trained via FL, against such memorization.

Acknowledgements

The authors would like to thank Mingqing Chen, and Andrew Hard for their helpful comments towards improving the paper.

References

- [AMR⁺20] Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.
- [App17] Differential Privacy Team Apple. Learning with privacy at scale, 2017.
- [BEG⁺19] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.
- [BEM⁺17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 441–459. ACM, 2017.
- [BIK⁺17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 Association for Computing Machinery (ACM) Special Interest Group on Security, Audit and Control (SIGSAC) Conference on Computer and Communications Security, CCS '17*, pages 1175–1191, New York, NY, USA, 2017. ACM.
- [BUV14] Mark Bun, Jonathan Ullman, and Salil Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *Proceedings of the Forty-sixth Annual Association for Computing Machinery (ACM) Symposium on Theory of Computing, STOC '14*, pages 1–10, New York, NY, USA, 2014. Association for Computing Machinery (ACM).
- [CLB⁺19] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295, 2019.
- [CLK⁺18] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *Computing Research Repository (CoRR)*, abs/1802.08232, 2018.
- [CSM⁺] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*.
- [DKM⁺06a] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

- [DKM⁺06b] Cynthia Dwork, Krishnaram Kenthapadi, Frank Mcsherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.
- [DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3571–3580, 2017.
- [DMNS06] 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.
- [DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second Association for Computing Machinery (ACM) Special Interest Group on Algorithms and Computation Theory (SIGACT)-Special Interest Group on Management of Data (SIGMOD)-Special Interest Group on Artificial Intelligence (SIGART) Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 202–210, 2003.
- [DSS⁺15] C. Dwork, A. Smith, T. Steinke, J. Ullman, and S. Vadhan. Robust traceability from trace amounts. In *2015 Institute of Electrical and Electronics Engineers (IEEE) 56th Annual Symposium on Foundations of Computer Science*, pages 650–669, Oct 2015.
- [EFM⁺20] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618, 2020.
- [EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 Association for Computing Machinery (ACM) SIGSAC conference on computer and communications security*, pages 1054–1067. Association for Computing Machinery (ACM), 2014.
- [FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd Association for Computing Machinery (ACM) SIGSAC Conference on Computer and Communications Security, CCS '15*, pages 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery (ACM).
- [HRM⁺18] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.
- [HSR⁺08] 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.
- [IO19] Alex Ingerman and Krzys Ostrowski. Introducing tensorflow federated, 2019.
- [JTT18] Prateek Jain, Om Thakkar, and Abhradeep Thakurta. Differentially private matrix completion revisited. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, pages 2220–2229, 2018.
- [KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [KCK⁺18] Yu-Hsuan Kuo, Cho-Chun Chiu, Daniel Kifer, Michael Hay, and Ashwin Machanavajjhala. Differentially private hierarchical count-of-counts histograms. *PVLDB*, 11(11):1509–1521, 2018.

- [KKM⁺19] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019.
- [KMA⁺19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D’Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.
- [KMRR16] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *ArXiv*, abs/1610.02527, 2016.
- [KMY⁺16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *Computing Research Repository (CoRR)*, abs/1610.05492, 2016.
- [Mir17] I. Mironov. Rényi differential privacy. In *2017 Institute of Electrical and Electronics Engineers (IEEE) 30th Computer Security Foundations Symposium (CSF)*, pages 263–275, Aug 2017.
- [MMR⁺17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017.
- [MR17] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3, 2017.
- [MRTZ17] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. *CoRR*, abs/1710.06963, 2017.
- [MSDS18] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. *arXiv e-prints*, page arXiv:1805.04049, May 2018.
- [Mun19] Randall Munroe. xkcd: Predictive models, 2019. <https://xkcd.com/2169/>.
- [Ove18] Stack Overflow. The Stack Overflow Data, 2018. <https://www.kaggle.com/stackoverflow/stackoverflow>.
- [Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.
- [RMRB19] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard, 2019.
- [RSP⁺20] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. LinkedIn’s audience engagements api: A privacy preserving data analytics system at scale, 2020.
- [SOJH09] 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, 2009.

- [SS19] Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 196–206. ACM, 2019.
- [SSB14] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014.
- [SSSS17] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 Institute of Electrical and Electronics Engineers (IEEE) Symposium on Security and Privacy (SP)*, pages 3–18, May 2017.
- [TAM19] Om Thakkar, Galen Andrew, and H. Brendan McMahan. Differentially private learning with adaptive clipping. *CoRR*, abs/1905.03871, 2019.
- [WBK19] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 1226–1235, 2019.
- [WFJN16] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 Institute of Electrical and Electronics Engineers (IEEE) 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, June 2016.

A DP Federated Averaging with Fixed-size Rounds

We now present the technique used to train our DP NWP model. It closely follows the DP-FedAvg technique in [MRTZ17], in that per-user updates are clipped to have a bounded L_2 norm, and calibrated Gaussian noise is added to the weighted average update to be used for computing the model to be sent in the next round. A slight difference between the DP-FedAvg algorithm in [MRTZ17] and our approach is the way in which client devices are sampled to participate in a given federated round of computation. DP-FedAvg uses Poisson sampling, where for each round, each user is selected independently with a fixed probability. In this work (also, following [AMR⁺20]), we instead use fixed-size federated rounds, where a fixed number of users is randomly sampled to participate in each round. For reference, we provide a pseudo-code for the technique in Algorithm 1.

Main training loop:

parameters: round participation fraction $q \in (0, 1]$, total user population $N \in \mathbb{N}$, noise scale $z \in \mathcal{R}^+$, clip parameter $S \in \mathcal{R}^+$

Initialize model θ^0 , moments accountant \mathcal{M}

Set $\sigma = \frac{zS}{qW}$

for each round $t = 0, 1, 2, \dots$ **do**

$C^t \leftarrow$ (sample without replacement qN users from population)

for each user $k \in C^t$ **in parallel do**

$\Delta_k^{t+1} \leftarrow$ UserUpdate(k, θ^t)

$\Delta^{t+1} = \frac{1}{qN} \sum_{k \in C^t} \Delta_k^{t+1}$

$\theta^{t+1} \leftarrow \theta^t + \Delta^{t+1} + \mathcal{N}(0, I\sigma^2)$

$\mathcal{M}.\text{accum_priv_spending}(z)$

print $\mathcal{M}.\text{get_privacy_spent}()$

UserUpdate(k, θ^0):

parameters: number of local epochs $E \in \mathbb{N}$, batch size $B \in \mathbb{N}$, learning rate $\eta \in \mathcal{R}^+$, clip parameter $S \in \mathcal{R}^+$, loss function $\ell(\theta; b)$

$\theta \leftarrow \theta^0$

for each local epoch i from 1 to E **do**

$\mathcal{B} \leftarrow$ (k 's data split into size B batches)

for each batch $b \in \mathcal{B}$ **do**

$\theta \leftarrow \theta - \eta \nabla \ell(\theta; b)$

$\Delta = \theta - \theta^0$

return update $\Delta_k = \Delta \cdot \min\left(1, \frac{S}{\|\Delta\|}\right)$ // Clip

Algorithm 1: DP-FedAvg with fixed-size federated rounds, used to train our DP NWP model.

Privacy analysis: Following the analysis of this technique in [AMR⁺20], we obtain our DP guarantees by using the following:

1. the analytical moments accountant [WBK19] to obtain the Rényi differential privacy (RDP) guarantee for a federated round of computation that is based on the subsampled Gaussian mechanism,
2. Proposition 1 [Mir17] for computing the RDP guarantee of the composition involving all the rounds, and
3. Proposition 3 [Mir17] to obtain a DP guarantee from the composed RDP guarantee.

B Additional Empirical Evaluation

In this section, we present the results of our additional empirical evaluation that was omitted from the main body.

Using Different Optimizers: In Table 3, we provide the results for using different optimizers like Momentum [Qia99] and Adam [KB15] for training. We conduct experiments using only the smallest batch size in both the granularities (32 records, or 500 users). For Momentum, we set the momentum parameter to 0.9, and for Adam, we set the learning rate to 10^{-4} . First, we observe that using Momentum increases the observed unintended memorization but has a similar utility as SGD. On the other hand, we see that using Adam decreases such memorization, but the utility of the models is also noticeably reduced as compared to SGD. We observe a similar trend when Adam is combined with FedAvg.

Evaluating for Same Training Epochs: In Table 4, we provide the results for evaluating models trained for the same number of epochs over the training data. For the runs using SGD, we start with a batch size of 32 records and a tuned

| Setting Data, Optimizer, Batch Size | RS /90 | RS Lowest (p_u, p_e) | BS /90 | BS Lowest (p_u, p_e) | Acc. % | Perp. |
|--|-----------|-----------------------------|-----------|-----------------------------|-----------|-------|
| IID, SGD, $b_r = 32$ | 54 | 1/50K, 10% | 42 | 3/50K, 10% | 24 | 62.2 |
| IID, Momentum, $b_r = 32$ | 64 | 3/50K, 1% | 50 | 1/50K, 10% | 24.2 | 60.6 |
| IID, Adam, $b_r = 32$ | 56 | 1/50K, 10% | 42 | 3/50K, 10% | 22.7 | 70.8 |
| IID, FedAvg, $b_u = 500$ | 66 | 1/50K, 10% | 56 | 1/50K, 10% | 24.6 | 57.5 |
| IID, FedAvg+Adam, $b_u = 500$ | 60 | 1/50K, 10% | 46 | 1/50K, 10% | 23.7 | 63.7 |
| Non-IID, SGD, $b_r = 32$ | 37 | 1/50K, 10% | 19 | 3/50K, 10% | 23.7 | 64.3 |
| Non-IID, Momentum, $b_r = 32$ | 48 | 3/50K, 1% | 36 | 1/50K, 10% | 24.3 | 59.9 |
| Non-IID, Adam, $b_r = 32$ | 21 | 1/50K, 10% | 13 | 1/50K, 10% | 21.5 | 84.1 |
| Non-IID, FedAvg, $b_u = 500$ | 21 | 1/5K, 1% | 0 | - | 24.4 | 58.8 |
| Non-IID, FedAvg+Adam, $b_u = 500$ | 8 | 1/5K, 10% | 0 | - | 23.3 | 66.5 |

Table 3: Unintended memorization, lowest (by insertion frequency) canary configuration memorized, and utility metrics for models using different optimizers evaluated at 37.5M steps when sampling records (e.g., SGD), and 8000 rounds when sampling users (e.g., FedAvg).

learning rate of 0.005, and we increase the learning rate by $\approx \sqrt{2}$ for every 2x increase in the batch size. For all the experiments with FedAvg, we find that using a constant learning rate provides the best utility across the different batch sizes, and thus, we keep it fixed. For the models trained with SGD, for both IID and non-IID data we observe that unintended memorization remains comparable for models trained with different batch sizes. However, we see a decrease in the utility as the batch size increases. The decrease in utility is observed for models trained using FedAvg as well, but we also observe a significant drop in the such memorization when training is performed with at least 1000 users per round. Moreover, once the training involves at least 2000 users on non-IID data, both the RS and BS methods are unsuccessful in classifying any of the 90 inserted canaries as memorized.

| Setting Data, Optimizer, Batch Size | RS /90 | RS Lowest (p_u, p_e) | BS /90 | BS Lowest (p_u, p_e) | Acc. % | Perp. |
|--|-----------|-----------------------------|-----------|-----------------------------|-----------|-------|
| IID, SGD, $b_r = 32$ | 49 | 1/50K, 10% | 43 | 3/50K, 10% | 23.9 | 63 |
| IID, SGD, $b_r = 64$ | 51 | 1/50K, 10% | 39 | 3/50K, 10% | 23.5 | 65.1 |
| IID, SGD, $b_r = 128$ | 46 | 3/50K, 10% | 36 | 1/50K, 100% | 23.2 | 67.6 |
| IID, SGD, $b_r = 256$ | 46 | 3/50K, 10% | 32 | 1/50K, 100% | 23 | 69.7 |
| IID, FedAvg, $b_u = 500$ | 66 | 1/50K, 10% | 56 | 1/50K, 10% | 24.6 | 57.5 |
| IID, FedAvg, $b_u = 1000$ | 27 | 1/50K, 100% | 18 | 3/50K, 100% | 24.5 | 58 |
| IID, FedAvg, $b_u = 2000$ | 28 | 1/50K, 100% | 18 | 3/50K, 100% | 24.4 | 59.3 |
| IID, FedAvg, $b_u = 5000$ | 28 | 1/50K, 100% | 18 | 3/50K, 100% | 24 | 62.5 |
| Non-IID, SGD, $b_r = 32$ | 40 | 1/50K, 10% | 29 | 1/50K, 10% | 23.7 | 64.2 |
| Non-IID, SGD, $b_r = 64$ | 38 | 1/50K, 10% | 32 | 1/50K, 10% | 23.6 | 65.6 |
| Non-IID, SGD, $b_r = 128$ | 35 | 1/50K, 10% | 26 | 1/50K, 100% | 23.2 | 68.2 |
| Non-IID, SGD, $b_r = 256$ | 40 | 3/50K, 10% | 31 | 1/50K, 100% | 23 | 69.8 |
| Non-IID, FedAvg, $b_u = 500$ | 21 | 1/5K, 1% | 0 | - | 24.4 | 58.8 |
| Non-IID, FedAvg, $b_u = 1000$ | 10 | 1/5K, 10% | 0 | - | 24 | 61.2 |
| Non-IID, FedAvg, $b_u = 2000$ | 0 | - | 0 | - | 24 | 61.9 |
| Non-IID, FedAvg, $b_u = 5000$ | 0 | - | 0 | - | 23.3 | 67.9 |

Table 4: Results for the number of inserted canaries (out of 90) memorized via the RS and BS methods, the lowest (by insertion frequency) canary configurations that show up as memorized for each of the methods, and utility metrics for various models evaluated at ≈ 10 epochs of training.