Training Differentially Private Ad Prediction Models With Semi-Sensitive Features

Lynn Chua, Qiliang Cui, Badih Ghazi, Charlie Harrison, Pritish Kamath, Walid Krichene, Ravi Kumar, Pasin Manurangsi, Nicolas Mayoraz, Krishna Giri Narra, Steffen Rendle, Amer Sinha, Avinash Varadarajan, Chiyuan Zhang
Google

ABSTRACT

Motivated by problems arising in digital advertising, we study the task of training differentially private (DP) machine learning models with semi-sensitive features. In this setting, a subset of the features is known to the attacker (and thus need not be protected) while the remaining features as well as the label are unknown to the attacker and should be protected by the DP guarantee. This task interpolates between training the model with *full* DP (where the label and all features should be protected) or with *label* DP (where all the features are considered known, and only the label should be protected). We present a new algorithm for training DP models with semi-sensitive features. Through an empirical evaluation on ads datasets, we demonstrate that our algorithm surpasses in utility the baselines of (i) DP stochastic gradient descent (DP-SGD) run on all features (known and unknown), and (ii) a label DP algorithm run only on the known features (while discarding the unknown ones).

KEYWORDS

Differential privacy, model training, ad models, semi-sensitive features

ACM Reference Format:
Lynn Chua, Qiliang Cui, Badih Ghazi, Charlie Harrison, Pritish Kamath,, Walid Krichene, Ravi Kumar, Pasin Manurangsi, Nicolas Mayoraz, Krishna, Giri Narra, Steffen Rendle, Amer Sinha, Avinash Varadarajan, Chiyuan Zhang . 2024. Training Differentially Private Ad Prediction Models With Semi-Sensitive Features. In *Proceedings of (AdKDD '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, large-scale machine learning (ML) algorithms have been adopted and deployed for different ad modeling tasks, including the training of predicted click-through rates (a.k.a. pCTR) and predicted conversion rates (a.k.a. pCVR) models. Roughly speaking, pCTR models predict the likelihood that an ad shown to a user is clicked, and pCVR models predict the likelihood that an ad clicked (or viewed) by the user leads to a *conversion*—which is defined as a desirable action by the user on the advertiser site or app, such as the purchase of the advertised product.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
AdKDD '24, Aug 25, 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Heightened user expectations around privacy have led different web browsers (including Apple Safari [27], Mozilla Firefox [30], and Google Chrome [21]) to the deprecation of third-party cookies (3PC), which are cross-site identifiers that had hitherto allowed the joining in the clear of the datasets on which the pCTR and pCVR models are trained. More precisely, 3PCs previously allowed determining the conversion label for pCVR models as well as the construction of features (for pCTR and pCVR models) that depend on the user's behavior on sites other than the publisher where the ad was shown.

In order to support essential web functionalities that are affected by the deprecation of 3PCs, different web browsers have been building privacy-preserving APIs, including for ads measurement and modeling such as the Interoperable Private Attribution (IPA) developed by Mozilla and Meta [25], Masked LARK from Microsoft [16, 19], the Attribution Reporting API, available on both the Chrome browser [18] and the Android operating system [2], and the Private Click Measurement (PCM) [28] and Private Ad Measurement (PAM) APIs [29] from Apple. The privacy guarantees of several of these APIs rely on *differential privacy* (DP) [6, 7], which is a strong and robust notion of privacy that has in recent years gained significant popularity for data analytics and modeling tasks.

Different DP variants have been studied in the context of supervised ML, depending on the adjacency definition. The standard definition of DP protects the full training example (features and label) and has been extensively studied, e.g., Abadi et al. [1]. On the other hand, *Label* DP (e.g., Chaudhuri and Hsu [4], Ghazi et al. [9], Malek Esmaeili et al. [14]) is a variant that only protects the label of each training example, and is thus suitable in settings where the adversary already has access to the features.

Label DP is a natural fit for the case where the features of the pCVR problem do not depend on cross-site information. However, a common setting, including that of the Protected Audience API on Chrome [5] and Android [3], is where some features depend on cross-site information whereas the remaining features do not. An example is the remarketing use case where a feature could indicate whether the same user previously expressed interest in the advertised product (e.g., added it to their cart) but did not purchase it. Revealing a row of the database that has both contextual features (e.g., the publisher site, or the time of day the ad was served) and features derived on the advertiser (e.g., user presence on a particular remarketing list) could allow an attacker to track a user across sites. In the Protected Audience API, these sensitive features are protected by multiple privacy mechanisms including feature-level randomized response.

The focus of this work is to analyze this setting from the DP perspective; we refer to it as *DP model training with semi-sensitive features*. We formalize this setting, present an algorithm for training private ML models with semi-sensitive features, and evaluate it on

real ad prediction datasets, showing that it compares favorably to natural baselines.[1] We report the effect of certain important parameters on utility, and also study the trade-offs between the size of the private model and its quality – this is motivated by practical settings, in which the private ML model training may happen in Trusted Execution Environments with limited memory.

2 RELATED WORK

Two related notions of private model training with partially private features were recently proposed, although they differ slightly in their adjacency definitions (and hence in what is considered public information): Krichene et al. [11] propose a stronger notion in which the adversary is only assumed to know the *set of distinct values* that the non-sensitive features may take, for example the feature values of all possible ads, (while in our notion, we assume the adversary knows which specific values appeared in the dataset, together with their counts). And in a concurrent work [22], an algorithm based on AdaBoost was proposed under a privacy notion similar to ours; a key difference is that in their setting, the labels are considered public.

3 DP TRAINING WITH SEMI-SENSITIVE FEATURES

We consider the setting of supervised learning, where we assume an underlying (unknown) distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the set of possible inputs and $\mathcal{Y}$ denotes the set of possible labels. In this work, we focus on the binary classification setting where $\mathcal{Y} = \{0, 1\}$. Our goal is to learn a predictor $F : \mathcal{X} \to \mathbb{R}$ that maps the input space $\mathcal{X}$ to $\mathbb{R}$ with the goal of minimizing the expected loss $\mathcal{L}(F; \mathcal{D}) := \mathbb{E}_{(x,y)\sim\mathcal{D}}\, \ell(F(x), y)$, where $\ell(\cdot, \cdot)$ is a suitable loss function, e.g., the binary cross entropy loss.

To capture the setting of *semi-sensitive* features, let $\mathcal{X} = \mathcal{X}^\circ \times \mathcal{X}^\bullet$, where $\mathcal{X}^\circ$ is the set of possible *nonsensitive* feature values, and $\mathcal{X}^\bullet$ is the set of possible *sensitive* feature values. We denote a dataset as $D = ((x_i^\circ, x_i^\bullet, y_i))_{i\in[n]}$, where x_i° denotes the nonsensitive feature value, $x_i^\bullet$ is the sensitive feature value, and y_i is the corresponding (sensitive) label. We use x_i to denote $(x_i^\circ, x_i^\bullet)$ for short. Some problems that motivate the setting above are in ads modeling tasks, where the features can include nonsensitive features such as the browser class, publisher website, category of the mobile app etc., sensitive features such as how long ago and how many times a user showed interest in an advertised product etc., and sensitive labels such as whether the user converted on the ad.

We say that two datasets D, D' are *adjacent*, denoted $D \sim D'$ if one dataset can be obtained from the other by changing the sensitive features and/or the label for a single example, namely replacing $(x_i^\circ, x_i^\bullet, y_i)$ with $(x_i^\circ, \tilde{x}_i^\bullet, \tilde{y}_i)$ for some $(\tilde{x}_i^\bullet, \tilde{y}_i) \in \mathcal{X}^\bullet \times \mathcal{Y}$. Note in particular that x_i° are not allowed to change in the adjacent dataset, and should be considered known to the adversary.[2]

Definition 1 (DP; Dwork et al. [7])**.** For $\varepsilon, \delta \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies (ε, δ)-DP if for all pairs D, D' of adjacent datasets, and for all outcome events E, it holds that $\Pr[\mathcal{M}(D) \in E] \leq e^\varepsilon \cdot \Pr[\mathcal{M}(D') \in E] + \delta$.

[1] A preliminary version of this paper was presented at PPAI-24: The 5th AAAI Workshop on Privacy-Preserving Artificial Intelligence.

[2] Contrast this with the notion of *DP with public features* of [12], in which x_i° *is* allowed to change, as long as it takes values in the publicly known $\mathcal{X}^\circ$.

For an extensive overview of DP, we refer the reader to the monograph of Dwork and Roth [8]. We use the following key properties of DP.

PROPOSITION 2 (COMPOSITION). *If $\mathcal{M}_1$ satisfies $(\varepsilon_1, \delta_1)$-DP, and $\mathcal{M}_2$ satisfies $(\varepsilon_2, \delta_2)$-DP, then the mechanism $\mathcal{M}$ that on dataset D returns $(\mathcal{M}_1(D), \mathcal{M}_2(D))$ satisfies $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP. Furthermore, this holds even in the adaptive case, when $\mathcal{M}_2$ can use the output of $\mathcal{M}_1$.*

PROPOSITION 3 (POST-PROCESSING). *If $\mathcal{M}$ satisfies (ε, δ)-DP, then for all (randomized) algorithms $\mathcal{A}$, it holds that $\mathcal{A}(\mathcal{M}(\cdot))$ satisfies (ε, δ)-DP.*

Randomized Response. Perhaps the simplest mechanism that satisfies DP, even predating its definition, is *Randomized Response*. We state the mechanism in our context as releasing the known features along with the corresponding randomized (binary) label.

Definition 4 (Randomized Response; Warner [26])**.** For $\varepsilon > 0$, the mechanism RR_ε on dataset $D = ((x_i^\circ, x_i^\bullet, y_i))_{i\in[n]}$ returns $((x_i^\circ, \hat{y}_i))_{i\in[n]}$ where each $\hat{y}_i$ is set to y_i with prob. $\frac{e^\varepsilon}{1+e^\varepsilon}$ and to $1 - y_i$ with prob. $\frac{1}{1+e^\varepsilon}$.

PROPOSITION 5. RR_ε *satisfies* $(\varepsilon, 0)$-DP.

SGD *and* **DP-SGD**. Let $F_{\boldsymbol{w}}$ be a parameterized model (e.g., a neural network) with trainable weights $\boldsymbol{w}$, and $\{(x_1, y_1), \ldots, (x_B, y_B)\}$ be a random mini-batch of training examples. Let $L_i = \ell(F_{\boldsymbol{w}}(x_i), y_i)$ be the loss on the ith example and let the average loss be $\bar{L} := \frac{1}{B}\sum_{i=1}^B L_i$. Recall that standard training algorithms compute the *average gradient* $\nabla_{\boldsymbol{w}}\bar{L}$ and update $\boldsymbol{w}$ with an optimizer such as SGD or Adam. Even though various optimizers could be used, we will refer to this class of (non-private) methods as SGD.

DP-SGD [1] is widely used for DP training of deep neural networks, wherein the per-example gradients $\nabla_{\boldsymbol{w}} L_i$ are computed, and then re-scaled to have an ℓ_2-norm of at most C, as $\boldsymbol{g}_i := \nabla_{\boldsymbol{w}} L_i \cdot \min\{1, \frac{C}{\|\nabla_{\boldsymbol{w}} L_i\|_2}\}$. Gaussian noise $\mathcal{N}(\boldsymbol{0}, C^2\sigma^2\boldsymbol{I})$ is then added to the average $\frac{1}{B}\sum_{i=1}^B \boldsymbol{g}_i$ and subsequently passed to the optimizer. As shown by Abadi et al. [1], DP-SGD satisfies (ε, δ)-DP where ε, δ depend on σ, the batch size and number of training steps; this can be computed using the privacy accounting described in [1].

4 ALGORITHMS

We now describe the family of algorithms we use for DP training with semi-sensitive features. Consider a model, such as a deep neural network, parameterized by $\boldsymbol{w}$. We will use the following high-level architecture:

$$F_{\boldsymbol{w}}(x^\circ, x^\bullet) := f_{\boldsymbol{w}^{\mathrm{c}}}\left(g_{\boldsymbol{w}^\circ}(x^\circ), h_{\boldsymbol{w}^\bullet}(x^\bullet)\right),$$

where $\boldsymbol{w} = (\boldsymbol{w}^\circ, \boldsymbol{w}^\bullet, \boldsymbol{w}^{\mathrm{c}})$, $g_{\boldsymbol{w}^\circ} : \mathcal{X}^\circ \to \mathbb{R}^{d_\circ}$ is a *nonsensitive* tower (i.e. the part of the model that acts on the nonsensitive features), $h_{\boldsymbol{w}^\bullet} : \mathcal{X}^\bullet \to \mathbb{R}^{d_\bullet}$ is a *sensitive* tower (acting on the sensitive features), and $f_{\boldsymbol{w}^{\mathrm{c}}} : \mathbb{R}^{d_\circ} \times \mathbb{R}^{d_\bullet} \to \mathbb{R}$ is a *common* tower.

We also consider a *truncated model* that uses the same parameters $\boldsymbol{w}^\circ$ and $\boldsymbol{w}^{\mathrm{c}}$, but does not depend on $\boldsymbol{w}^\bullet$, by eliminating the dependence on $x^\bullet$, defined as follows:

$$F_{\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}}(x^\circ) := f_{\boldsymbol{w}^{\mathrm{c}}}(g_{\boldsymbol{w}^\circ}(x^\circ), \boldsymbol{0}),$$

where $\mathbf{0} \in \mathbb{R}^{d_\bullet}$. For convenience, we use the following notation for the losses of each of these models:

$$L(\boldsymbol{w}; x, y) \;:=\; \ell(F_{\boldsymbol{w}}(x), y),$$
$$L(\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}; x, y) \;:=\; \ell(F_{\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}}(x^\circ), y).$$

Given a total privacy budget of (ε, δ), we consider learning algorithms that execute two phases sequentially that satisfy $(\varepsilon_1, 0)$-DP and (ε_2, δ)-DP respectively such that $\varepsilon_1 + \varepsilon_2 = \varepsilon$ and hence by Proposition 2, the algorithm satisfies (ε, δ)-DP. We refer to this algorithm as Hybrid, and these phases are as follows:

Label-DP Phase. In this phase, we first apply $\mathrm{RR}_{\varepsilon_1}$ to generate $((x_i^\circ, \hat{y}_i))_{i\in[n]}$, i.e., a dataset where the sensitive $x_i^\bullet$'s are *removed* and the labels are randomized. Then, we train the truncated model $F_{\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}}(\cdot)$ on this data for one or more epochs of mini-batch SGD. By Proposition 5 and Proposition 3, this phase satisfies $(\varepsilon_1, 0)$-DP. To remove the bias introduced by the noisy labels, we define $p := 1/(1 + e^{-\varepsilon_1})$ and modify the training loss as follows:

$$\tilde{L}(\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}; x_i^\circ, \hat{y}_i) = \frac{L(\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}; x_i^\circ, 1-\hat{y}_i) - p\sum_{y'\in\{0,1\}} L(\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}; x_i^\circ, y')}{1-2p}.$$

DP-SGD Phase. In this phase, we train the entire model $F_{\boldsymbol{w}}(\cdot)$, by warm-starting it from the $F_{\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}}$ model of the first phase, then training for one or more epochs of DP-SGD. We propose two variants: in the first, we *freeze* the sensitive tower $g_{\boldsymbol{w}^\circ}$, and in the second, we *fine-tune* it. The noise parameter σ is chosen appropriately so that this phase satisfies (ε_2, δ)-DP; in our work, we do this accounting using Rényi DP [1, 17], though other accounting techniques could be used, such as privacy loss distributions (PLD) [15, 23].

5 EXPERIMENTAL RESULTS

We consider two natural baselines: DP-SGD (where all features are treated as sensitive) and RR on the truncated model $F_{\boldsymbol{w}^\circ, \boldsymbol{w}^{\mathrm{c}}}$ (where the sensitive features are discarded and only the labels are protected). Note that both can be viewed as special cases of Hybrid, where we use all the privacy budget in one of the two phases: DP-SGD corresponds to setting $\varepsilon_1 = 0$ and $\varepsilon_2 = \varepsilon$; and RR corresponds to setting $\varepsilon_1 = \varepsilon$ and $\varepsilon_2 = 0$.

The Hybrid algorithm allows using a different split between the two phases. A total privacy budget (ε, δ) will be split into $(\varepsilon_1, 0)$ and (ε_2, δ). Since this budget allocation may have an impact on model quality, we will vary it in our experiments as follows:

$$\varepsilon_1 \;:=\; k\cdot\varepsilon, \qquad \varepsilon_2 \;:=\; (1-k)\cdot\varepsilon, \quad \text{where } k \in \{0, 0.25, 0.5, 0.75, 1\},$$

(the cases $k = 0, k = 1$ correspond to DP-SGD and RR, respectively).

We train binary classification models with binary cross-entropy loss and report it together with the AUC loss (defined as $1 - \mathrm{AUC}$). We study the trade-offs between privacy and utility, as well as model size and utility.

5.1 Models

We evaluate two model classes for $F_{\boldsymbol{w}}$: multilayer perceptrons (MLP) and factorization machines (FM).

Multilayer perceptron. In the MLP models, we concatenate the outputs of the sensitive and nonsensitive towers before feeding them into joint fully connected layers:

$$F_{\boldsymbol{w}}(x^\circ, x^\bullet) := f_{\boldsymbol{w}^{\mathrm{c}}}(g_{\boldsymbol{w}^\circ}(x^\circ) \circ h_{\boldsymbol{w}^\bullet}(x^\bullet)),$$

where $g_{\boldsymbol{w}^\circ} : \mathcal{X}^\circ \to \mathbb{R}^{d_\circ}$, $h_{\boldsymbol{w}^\bullet} : \mathcal{X}^\bullet \to \mathbb{R}^{d_\bullet}$, $f_{\boldsymbol{w}^{\mathrm{c}}} : \mathbb{R}^{d_\circ + d_\bullet} \to \mathbb{R}$ and $u \circ v$ denotes concatenation of the vectors u and v.

Factorization Machine. A factorization machine (FM) [20] embeds each feature into a $d = d_\circ = d_\bullet$ dimensional embedding and builds all pairwise dot products between all features. We shortly summarize how FM can be cast into our notation of $F_{\boldsymbol{w}}$. The parameters of the FM model consist of (i) embeddings for the sensitive and nonsensitive features $\boldsymbol{w}^\circ \in \mathbb{R}^{\mathcal{X}^\circ \times d}$ and $\boldsymbol{w}^\bullet \in \mathbb{R}^{\mathcal{X}^\bullet \times d}$, and (ii) a bias $\boldsymbol{w}^{\mathrm{c}} \in \mathbb{R}$. The combination function $f_{\boldsymbol{w}^{\mathrm{c}}}$ consists of a sum of three different terms: the global bias $\boldsymbol{w}^{\mathrm{c}}$, linear effects that are encoded in the first dimension of g and h, and pairwise interactions of the remaining dimensions:

$$f_{\boldsymbol{w}^{\mathrm{c}}}(g, h) = \boldsymbol{w}^{\mathrm{c}}_1 + g_1 + h_1 + \langle g_{2\ldots d}, h_{2\ldots d}\rangle\,. \tag{1}$$

Unlike an MLP, FM does not have parameters (besides a bias) in the combiner function and does not need to learn how to combine embeddings. The towers are computed by:

$$g_{\boldsymbol{w}^\circ}(x^\circ)_1 = \langle x^\circ, \boldsymbol{w}^\circ_{\cdot,1}\rangle + \sum_j \sum_{k>j} x^\circ_j x^\circ_k \left\langle \boldsymbol{w}^\circ_{j,2\ldots d}, \boldsymbol{w}^\circ_{k,2\ldots d}\right\rangle. \tag{2}$$

$$g_{\boldsymbol{w}^\circ}(x^\circ)_{2\ldots d} = \sum_j x^\circ_j \boldsymbol{w}^\circ_{j,2\ldots d}. \tag{3}$$

The sensitive tower is computed analogously.

See Appendix A for further details about the experimental setup.

5.2 Criteo Display Ads pCTR Dataset

The first benchmark we consider is a pCTR prediction task on the Criteo Display Ads Dataset [10], which contains around 40 million examples. The dataset has a labeled training set and an unlabeled test set. We only use the labeled training set and split it chronologically into a 80%/10%/10% partition of train/validation/test sets. Each example consists of 13 integer features `int-feature-[1-13]` and 26 categorical features `categorical-feature-[14-39]`. Since the precise interpretation of these features is not available, we arbitrarily consider all even-numbered features as sensitive and all odd-numbered features as nonsensitive.

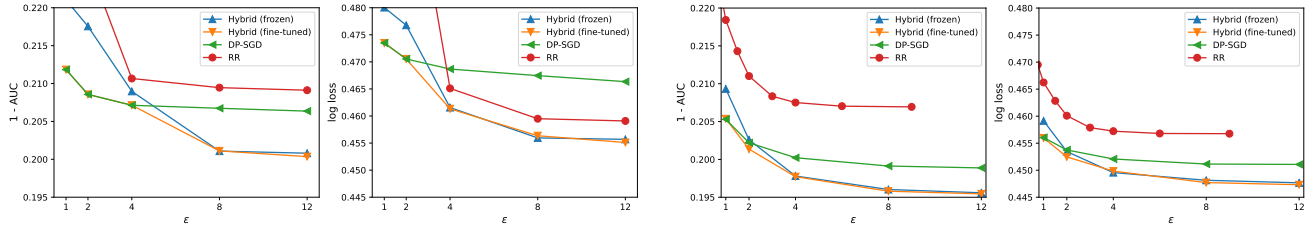
For this dataset, the AUC loss of the non-privately trained baselines is 0.1941 for the MLP model and 0.1930 for the FM model.

5.3 Criteo Spons. Search Conversion Log Dataset

The second benchmark we consider is a pCVR prediction task on the Criteo Spons. Search Conversion Log Dataset [24], which contains 16 millions examples. We used a random 80%/20% partition of train/test sets and the reported metrics are on the test set. The task considered in this work is a conversion prediction task, predicting the binary feature `Sale` (which has 10.8% positive occurrences). The sensitive features are `device_type`, `audience_id`, and `user_id`. We consider all other features to be nonsensitive, except for features denoted *Outcome/Labels* in [24], and `product_price`, all of which are omitted from the model[3].

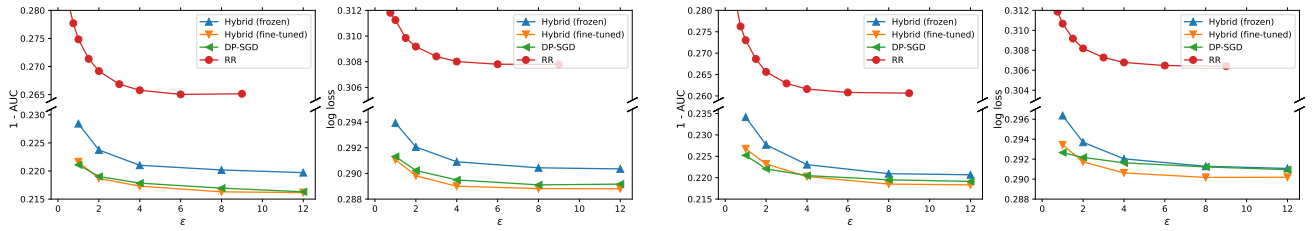
For this dataset, the AUC loss of the non-privately trained baselines is 0.2099 for the MLP model, and 0.2154 for the FM model.

[3]Although `product_price` is not explicitly marked as a label, it has a very high correlation with the label and the prediction task would become significantly easier if we were to include it.



(i) MLP model (ii) FM model

Figure 1: AUC loss and log loss of (i) MLP model and (ii) FM model trained under various privacy budgets ε on the Criteo Display Ads pCTR dataset.



(i) MLP model (ii) FM model

Figure 2: AUC loss and log loss of (i) MLP model and (ii) FM model trained under various privacy budgets ε on the Criteo Sponsored Search Conversion Log dataset.

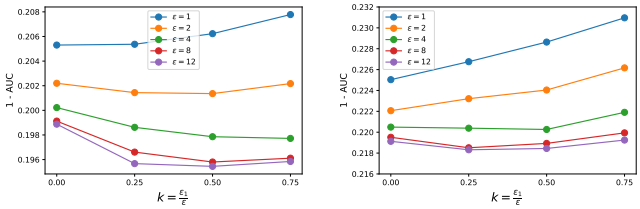
5.4 Results

5.4.1 Improved privacy-utility trade-off. Privacy-utility trade-offs on Criteo Display Ads and Criteo Spons. Search are reported in Figures 1 and 2 respectively. On both benchmarks, we find that Hybrid improves over RR and DP-SGD across a range of privacy budgets. Specifically, we see an improvement in utility for both the MLP and FM models when $\varepsilon \geq 4$. In this regime, there are substantial improvements: for instance, Hybrid achieves a better utility at $\varepsilon = 8$ than DP-SGD at $\varepsilon = 12$ (this is the case for both datasets and both models).

This significantly narrows the gap between the private model and the non-private baselines on Criteo Display Ads. For example, at $\epsilon = 12$, the relative increase in AUC loss (defined as $\frac{1-\text{AUC}}{1-\text{AUC}_{\text{non-private}}} - 1$) goes from 6.3% for MLP-DPSGD to 3.2% for MLP-Hybrid; and it goes from 3.0% for FM-DPSGD to 1.2% for FM-Hybrid. In both cases, the gap to the non-private model is approximately halved.

However, in the higher privacy regime (for $\varepsilon = 1$), the quality of the Hybrid-trained models appears to deteriorate, and in most cases it *no longer improves upon* DP-SGD. We believe this may be because the utility of the RR algorithm significantly deteriorates for small ε, and there may no longer be a benefit to the Label-DP phase in this regime.

It is also worth observing that the loss of the RR model plateaus at a value that is much higher than other methods – recall that the RR model is only trained on the non-sensitive features, hence its quality is limited by the best model one can train on these features alone.



(i) Criteo Display Ads (ii) Criteo Spons. Search

Figure 3: Effect of the budget split $k = \epsilon_1/\epsilon$ on AUC loss for the FM model, with $\varepsilon = 12$, on (i) Criteo Display Ads pCTR dataset and (ii) Criteo Spons. Search Conversion Log dataset.

5.4.2 Freezing vs fine-tuning. Freezing the sensitive tower during the second phase may offer a computational advantage, as one no longer needs to compute/clip gradients of this tower.

To understand the impact this may have on quality, we compare the two Hybrid variants (frozen and fine-tuned). We observe that in some settings (specifically on Criteo Display Ads in the high ϵ regime, see Figure 1), freezing can achieve comparable quality to fine-tuning. In all other cases, fine-tuning generally achieves better utility. In particular, for the Criteo Spons. Search dataset (see Figure 2), freezing leads to a significant degradation across all values of ϵ. This indicates that freezing, while computationally advantageous, may come at a high utility cost in practice.

5.4.3 Effect of budget split. To further understand the effect of the budget split, we report, in Figure 3, the AUC loss of the Hybrid FM models, as we vary the budget allocation ratio $k = \frac{\varepsilon_1}{\varepsilon}$. First,

observe that for $\varepsilon = 1$, the best utility is achieved when $k = 0$ (which corresponds to the special case of DP-SGD); this is consistent with the results of Section 5.4.1. As ε increases ($4 \leq \varepsilon \leq 8$), we observe that the optimal k increases, and the best utility is typically achieved when $k \geq 0.5$, i.e. one benefits from spending a significant part of the budget on the RR phase. Finally, in the high ε regime ($\varepsilon = 12$), the optimal k decreases again, and is equal to $k = 0.25$ in both datasets. This may be explained by the fact that the utility of the RR-trained model plateaus when ϵ grows (see Figure 1-(ii) and Figure 2-(ii)), so one may not benefit from spending a higher budget on the RR phase.

Given the large impact the budget ratio k has on quality, one should generally treat it as an important parameter to tune when using the Hybrid method, and it should be tuned separately for different values of ϵ.

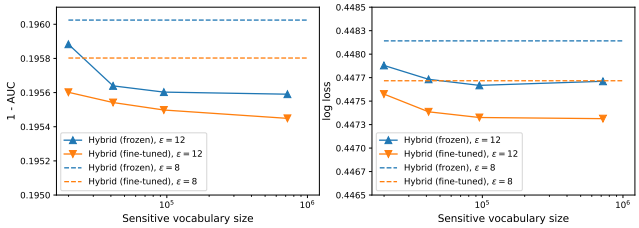


Figure 4: AUC loss and log loss of FM model trained under various private model sizes at $\epsilon = 12$ on the Criteo Display Ads pCTR dataset. The x-axis denotes the vocabulary size of the sensitive tower.

5.4.4 Model-size utility trade-off. In situations in which the DP-SGD phase of training happens in Trusted Executions Environments, one may be faced with stringent memory and compute constraints. In such scenarios, it is important to understand the trade-offs between utility and the size of the private model.

We vary the private model size on the Criteo Display Ads pCTR dataset, by varying the vocabulary size of the sensitive tower (this is done by computing privatized counts of the sensitive feature values, and keeping only features above a threshold. Varying the threshold leads to different model sizes). We report the results in Figure 4, for $\varepsilon = 12$. Here the largest model size corresponds to the results reported in Figure 1.

We observe that for a large range of model sizes, the deterioration in quality is surprisingly low. For example, focusing on the fine-tuned variant, when decreasing the model size ten-fold, the log loss of the FM model increases by 0.027%, and its AUC loss increases by 0.025%. When decreasing the model size fifty-fold, the log loss increases by 0.059% and the AUC loss by 0.078%. The loss remains well below that of the full-sized model at $\epsilon = 8$ (denoted by the dashed lines on the figure). This indicates that for these two benchmarks, one may train significantly smaller models under DP constraints without largely sacrificing quality.

6 CONCLUSION AND FUTURE DIRECTIONS

In this work, we studied training DP models with semi-sensitive features, and presented an algorithm that improves over two natural baselines on real ad modeling datasets.

Our experiments indicate that in the high privacy regime, it is difficult to improve upon DPSGD. This invites further investigation into this regime, either theoretically (by studying utility bounds), or experimentally. An interesting direction to explore is the use of label DP primitives beyond RR, e.g., [9, 14], particularly ones that perform better for smaller ε.

Another open question is the precise characterization of the differences (both in terms of privacy guarantees and potential utility gap) between the notion of "DP with semi-sensitive features" studied in this work, and "DP with public features" from [11].

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *CCS*. 308–318.

[2] Android. 2023. Attribution Reporting. https://developer.android.com/design-for-safety/privacy-sandbox/attribution.

[3] Android. 2023. Protected Audience API on Android developer guide. https://developer.android.com/design-for-safety/privacy-sandbox/guides/protected-audience.

[4] Kamalika Chaudhuri and Daniel Hsu. 2011. Sample complexity bounds for differentially private learning. In *COLT*. 155–186.

[5] Sam Dutton and Kevin K. Lee. 2022. Protected Audience API: On-device ad auctions to serve remarketing and custom audiences, without cross-site third-party tracking. https://developer.chrome.com/docs/privacy-sandbox/protected-audience.

[6] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*. 486–503.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*. 265–284.

[8] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[9] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. 2021. Deep learning with label differential privacy. In *NeurIPS*. 27131–27145.

[10] Olivier Chapelle Jean-Baptiste Tien, joycenv. 2014. Display Advertising Challenge. https://kaggle.com/competitions/criteo-display-ad-challenge

[11] Walid Krichene, Nicolas E Mayoraz, Steffen Rendle, Shuang Song, Abhradeep Thakurta, and Li Zhang. 2024. Private Learning with Public Features. In *AISTATS*. 4150–4158.

[12] Walid Krichene, Nicolas Mayoraz, Steffen Rendle, Shuang Song, Abhradeep Thakurta, and Li Zhang. 2023. Private Learning with Public Features. *arXiv:2310.15454* (2023).

[13] Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*.

[14] Mani Malek Esmaeili, Ilya Mironov, Karthik Prasad, Igor Shilov, and Florian Tramer. 2021. Antipodes of label differential privacy: PATE and ALIBI. In *NeurIPS*, Vol. 34. 6934–6945.

[15] Sebastian Meiser and Esfandiar Mohammadi. 2018. Tight on budget? Tight bounds for r-fold approximate differential privacy. In *CCS*. 247–264.

[16] Microsoft. 2021. MaskedLARk. https://github.com/microsoft/maskedlark.

[17] Ilya Mironov. 2017. Rényi Differential Privacy. In *CSF*. 263–275.

[18] Maud Nalpas and Alexandra White. 2021. Attribution Reporting. https://developer.chrome.com/en/docs/privacy-sandbox/attribution-reporting/.

[19] Joseph J Pfeiffer III, Denis Charles, Davis Gilton, Young Hun Jung, Mehul Parsana, and Erik Anderson. 2021. Masked LARk: Masked learning, aggregation and reporting workflow. *arXiv:2110.14794* (2021).

[20] Steffen Rendle. 2010. Factorization Machines. In *ICDM*. 995–1000.

[21] Justin Schuh. 2020. Building a more private web: A path towards making third party cookies obsolete. https://blog.chromium.org/2020/01/building-more-private-web-path-towards.html.

[22] Zeyu Shen, Anilesh Krishnaswamy, Janardhan Kulkarni, and Kamesh Munagala. 2023. Classification with Partially Private Features. *arXiv:2312.07583* (2023).

[23] David M. Sommer, Sebastian Meiser, and Esfandiar Mohammadi. 2019. Privacy Loss Classes: The Central Limit Theorem in Differential Privacy. *Proc. Priv. Enhancing Technol.* 2019, 2 (2019), 245–269. https://doi.org/10.2478/POPETS-2019-0029

[24] Marcelo Tallis and Pranjul Yadav. 2018. Reacting to Variations in Product Demand: An Application for Conversion Rate (CR) Prediction in Sponsored Search. *arXiv:1806.08211* (2018).

[25] Martin Thomson. 2022. Privacy Preserving Attribution for Advertising. https://blog.mozilla.org/en/mozilla/privacy-preserving-attribution-for-advertising/.

[26] Stanley L. Warner. 1965. Randomized response: a survey technique for eliminating evasive answer bias. *JASA* 60 309 (1965), 63–69.
[27] John Wilander. 2020. Full Third-Party Cookie Blocking and More. https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/.
[28] John Wilander. 2021. Introducing Private Click Measurement, PCM. https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/.
[29] Luke Winstrom. 2023. A proposal for privacy preserving ad attribution measurement using Prio-like architecture. https://github.com/patcg/proposals/issues/17.
[30] Marissa Wood. 2019. Today's Firefox Blocks Third-Party Tracking Cookies and Cryptomining by Default. https://blog.mozilla.org/en/products/firefox/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/.
[31] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive methods for nonconvex optimization. In *NIPS*.

A TRAINING DETAILS

In both Criteo datasets used in this work, the sensitive features (resp. nonsensitive features) are fed into a single embedding layer: each value of a sensitive (resp. nonsensitive) feature is concatenated to its feature name to form a unique string. These string values are then either hashed with a fixed number of hash bins, or a vocabulary of all sensitive (nonsensitive) strings is created, where only frequent values are kept while the remaining values are mapped to a single out-of-vocabulary token. In all our experiments reported here we use an embedding dimension of 32. The model size is therefore controlled either by the number of hash bins for the sensitive and nonsensitive features, or by frequency thresholds defining the the sensitive and nonsensitive vocabularies. In the latter case, we compute the frequency counts of the nonsensitive feature values exactly, while the counts of the sensitive features are computed privately, consuming a portion of the total privacy budget. We report the privacy parameters assuming that the vocabulary is known and only the counts are private; without this assumption the privacy ε increases by at most 0.007 compared to the reported numbers.

We compare these two strategies (hashing vs vocabulary thresholding) on the Criteo Display Ads pCTR dataset, and report the results in Figure 5. We find that the two approaches yield similar quality at the same model size (with a slight advantage to the vocab thresholding approach).

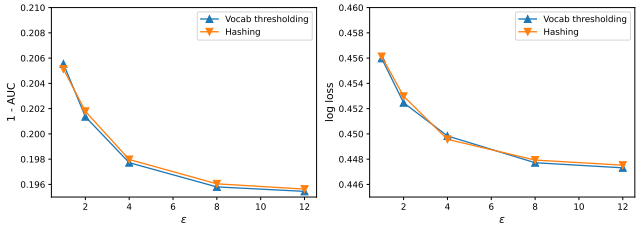


Figure 5: Privacy-utility trade-off of the fine-tuned Hybrid FM model on Criteo Display Ads, under hashing vs vocab thresholding.

When using thresholding in Criteo Display Ads, we used a vocab threshold of 16 for non-sensitive features, and a threshold in $\{16, 64, 256, 1024, 4096\}$ for sensitive features (this controls the sensitive tower size). On the Criteo Sponsored Search Conversion Log dataset, we opt for the simpler hashing approach, with 50k (resp. 100k) hash bins for the sensitive (resp. non-sensitive) features.

All features in both datasets being univalent, each example is transformed into a fixed number of embeddings, 19 sensitive and 20 nonsensitive embeddings for the Criteo Display Ads **pCTR** dataset, and 3 sensitive and 17 nonsensitive[4] embeddings for the Criteo Sponsored Search Conversion Log dataset.

Multilayer Perceptron. For the Criteo Display Ads **pCTR** dataset, the 20 nonsensitive features are concatenated and fed into a single fully connected layer with 598 hidden units and using a ReLU activation function. The output of this layer is concatenated with the 19 embeddings of the sensitive features, and these are fed into two fully connected layers, each also containing 598 hidden units and using a ReLU activation function. The final output is a linear combination of the last layer which produces a scalar logit prediction. We use the Yogi optimizer [31] with a base learning rate of 0.01 and a batch size of 1024 for our baseline and for the RR phase of training. We use SGD with a base learning rate of 0.1, momentum 0.9, and batch size of 16384 for the DP-SGD phase of training. For both, we scale the base learning rate with a cosine decay [13]. We train with 10 RR epochs and 50 or 100 DP-SGD epochs, and we tune the norm bound $C \in [10, 50]$.

For the Criteo Sponsored Search Conversion Log dataset dataset, the 17 nonsensitive features are concatenated and fed into a single fully connected layer with 256 hidden units and using a ReLU activation function. The output of this layer is concatenated with the 3 embeddings of the sensitive features, and these are fed into two fully connected layers, each also containing 256 hidden units and using a ReLU activation function. The final output is a linear combination of the last layer which produces a scalar logit prediction. We use the Adam optimizer with batch size of 1024 for the RR phase of training, and a batch size of 16384 for the DP-SGD phase of training. We train with 16 RR epochs and 64 DP-SGD epochs, and we tune the clipping norm $C \in [10, 30]$.

Factorization Machine. A linear model composed of a bias term and 20+19 (resp. 17+3) linear coefficients complements the above mentioned embeddings for the Criteo Display Ads **pCTR** dataset (resp. the Criteo Sponsored Search Conversion Log dataset). The scalar logit prediction is the sum of this linear model and all the pairwise dot-products of the 39 (resp. 20) embeddings. We use the Adam optimizer with a batch size of 16384 for all our experiments. In an initial hyper-parameter search we also tuned the standard deviation of the random initialization σ of all model parameters, as well as the regularization μ of the embeddings and we settled on $\sigma = 10^{-2}$ and $\mu \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ for all experiments. In practice, the regularization parameter had little impact. The most important parameters to tune are the learning rate (tuned in the range $[10^{-5}, 10^{-3}]$) and the clipping norm (tuned in the range $[10, 30]$. Note that we did not distinguish the dataset used for hyper-parameter tuning from the one used to report the final metrics as our experiments on the Criteo Display Ads **pCTR** dataset showed virtually no difference between metrics measured on either sets.

Finally, we note that the optimal hyper-parameters tend to differ when optimizing for AUC loss vs log loss. In particular, we found that good models in terms of log loss tend to require much larger clipping norms than models optimizing AUC.

[4] `click_timestamp` is replaced by two features: `click_hour_of_day` and `click_day_of_week`, and `nbr_clicks_1week` is replaced by its $\log_2$ transformed value.