EANA: Reducing Privacy Risk on Large-scale Recommendation Models

Devora Berlowitz*
devorab@google.com
Google Research
USA

Mei Chen
meic@google.com
Google
USA

Steve Chien
schien@google.com
Google Research
USA

Lin Ning†
linning@google.com
Google Research
USA

Shuang Song
shuangsong@google.com
Google Research
USA

Yunqi Xue
qqx@google.com
Google
USA

ABSTRACT

Embedding-based deep neural networks (DNNs) are widely used in large-scale recommendation systems. Differentially-private stochastic gradient descent (DP-SGD) provides a way to enable personalized experiences while preserving user privacy by injecting noise into every model parameter during the training process. However, it is challenging to apply DP-SGD to large-scale embedding-based DNNs due to its effect on training speed. This happens because the noise added by DP-SGD causes normally sparse gradients to become dense, introducing a large communication overhead between workers and parameter servers in a typical distributed training framework. This paper proposes embedding-aware noise addition (EANA) to mitigate the communication overhead, making training a large-scale embedding-based DNN possible. We examine the privacy benefit of EANA both analytically and empirically using secret sharer techniques. We demonstrate that training with EANA can achieve reasonable model precision while providing good practical privacy protection as measured by the secret sharer tests. Experiments on a real-world, large-scale dataset and model show that EANA is much faster than standard DP-SGD, improving the training speed by 54X and unblocking the training of a large-scale embedding-based DNN with reduced privacy risk.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Computing methodologies** → **Neural networks**; • **Information systems** → **Recommender systems**; **Personalization**.

KEYWORDS

privacy, embedding-based deep neural networks, recommendation system, large-scale, secret sharer

*Authors are listed by alphabetical order with surnames.
†Corresponding author.

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3546769

ACM Reference Format:
Devora Berlowitz, Mei Chen, Steve Chien, Lin Ning, Shuang Song, and Yunqi Xue. 2022. EANA: Reducing Privacy Risk on Large-scale Recommendation Models. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3523227.3546769

1 INTRODUCTION

Embedding-based deep neural networks have been successfully deployed in large-scale recommendation systems [7, 16, 18, 26, 28, 30]. While these models enrich users' personalized experience, they can also raise privacy concerns as there are potential risks of leaking user information encoded in the model.

A standard technique for mitigating privacy leaks in deep neural networks is differentially private stochastic gradient descent (DP-SGD) [1]. DP-SGD modifies the standard stochastic gradient descent (SGD) algorithm for training machine learning models by clipping the gradients of individual examples and then adding Gaussian noise to the result. This results in provable differential privacy (DP) guarantees for the resulting model, usually with a tradeoff to both model utility and training speed.

Applying DP-SGD to large-scale embedding-based models is particularly challenging due to an extra penalty in training speed. Large-scale models are usually trained in a distributed fashion, with a collection of central parameter servers hosting model parameters and hundreds of workers accessing the parameters and performing the training jobs. Each worker computes the gradients for their input examples locally and sends the model updates back to the parameter servers. A key feature of embedding-based models is that the embedding table comprises most of the parameters of the model. The gradient of the embedding table is thus usually sparse for each training step, being nonzero only for those rows whose corresponding vocabulary items are in an input example. Thus, without DP-SGD, a worker only needs to send a small amount of data to the parameter servers at each step. However, with DP-SGD, noise is added to each parameter, and the previously sparse gradients become dense. The resulting extra traffic between the workers and parameter servers and the extra computation cost significantly slows down the training speed, making it difficult or even prohibitive to train a model with DP-SGD in practice.

We propose a modified version of DP-SGD that we refer to as embedding-aware noise addition, or EANA, to address the training

speed issue. Specifically, EANA only adds noise to embedding parameters with non-zero gradients at each training step, thus keeping the gradients sparse. This eliminates the slowdown caused by traditional DP-SGD, but with the important drawback that EANA no longer guarantees differential privacy. Nonetheless, we demonstrate that EANA still has both analytical and empirical privacy. We also confirm experimentally that EANA effectively resolves the training speed issue on the real-world, large-scale embedding-based model, unblocking the training of large-scale embedding-based models with good practical privacy protection.

More explicitly, our **key contributions** are these:

- Propose EANA to address the slowness issue when training large-scale embedding-based deep neural networks with DP-SGD (see Section 3), making it possible to train a large-scale personalization model with practical privacy protection (see Section 4.2).
- Perform detailed theoretical analysis on the privacy benefits provided by EANA and the constraints of the technique (see Section 3.1).
- Show that models trained with EANA have comparable or acceptable precision (see Figure 6) compared to those trained without privacy.
- Perform empirical evaluation with secret sharer [5] tests on the MovieLens 20M [15] next movie prediction task, and demonstrate that EANA is as effective as DP-SGD regarding reducing the exposure of user's private data (see Section 4.1).
- Evaluate EANA on a real-world large-scale embedding-based DNN model and an industry large-scale dataset (see Section 4.2). We show that EANA is 54X faster than standard DP-SGD in terms of training speed (see Table 3).
- Perform secret sharer test on the real-world, large-scale model and show that EANA significantly reduces the exposure and thus the privacy risk (see Figure 7).

Related Work: Given the challenges of applying DP-SGD in practice, several recent works have tried to modify or improve on it in different ways. Du et al. [9] devise a technique, dynamic differential-privacy preserving SGD, to improve the model accuracy by adjusting the clipping value and the noise level during training, in contrast with the fixed hyperparameters in traditional DP-SGD.

Zhang et al. also consider taking advantage of the sparse nature of gradients in embedding models in [31]. After first showing some results for the broader class of sparse empirical risk minimization problems, they devise a modified version of DP-SGD tailored for embedding models. The modified algorithm involves using a differentially private sampling step to select a small number of embedding rows for noising instead of adding noise to all rows as in standard DP-SGD. Experiments on a small word embedding task show that this approach gives better utility than DP-SGD for the same provable privacy bound.

Other techniques have been proposed to speed up DP-SGD by reducing the cost of the per-example clipping step. Examples include Bu et al., [3] who propose using JL projections to accelerate DP-SGD and make it more memory efficient, and Goodfellow [14], who uses an outer product method on fully connected networks. These are useful in their own right but do not address the issue of communications costs in large-scale models.

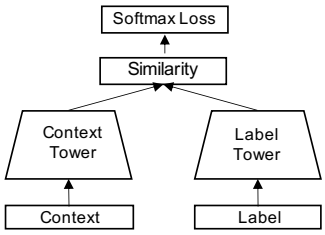


Figure 1: An illustration of the dual encoder model, a typical embedding-based recommendation model.

2 BACKGROUND

2.1 Embedding-based Recommendation Models

Deep neural network models are widely used in large-scale recommendation systems in the industry. Those recommendation systems often involve large numbers of discrete items, such as videos [7], apps [17], etc. A standard approach is to treat the discrete items as sparse categorical features with large vocabularies, and for models to use embeddings to represent them as continuous vectors. These models are referred to as embedding-based models.

The dual encoder model is a typical embeddings-based model for recommendation systems. These are also referred to as deep retrieval, two-tower, or encoder-encoder models depending on the setting [6, 12, 13, 20, 24, 29]. We will focus on this type of embedding-based model in our paper. As illustrated in Figure 1, a dual encoder model consists of two towers (encoders): a context tower and a label tower. Depending on the task, each of them can be a fully-connected network, convolutional neural network, Transformer, and so on. The input consists of *(context, label)* pairs encoded by the left and right towers, respectively. For example, in a movie recommendation use case (see Section 4.1), the context is a sequence of previous movies a user has watched. The label is the next movie they watch. In this setting, a dual encoder model can predict the next movie given an unseen sequence of previous movies and produce embeddings for users' context and movies.

The model outputs the similarity score between the encoded context and label, e.g., the inner product of context and item embeddings. A loss function is applied to enforce that positive examples (i.e., similar context and label pairs) have high similarity, and negative examples have low similarity. Once fully trained, the model can predict relevant items given a new context. The embeddings produced by the context and label towers are general representations of user contexts and label items. They can be used for other downstream applications, such as classification [6, 12].

2.2 DP and DP-SGD

Differential Privacy: Intuitively, differential privacy (DP) [10, 11] requires that one user's record does not affect the algorithm output significantly. The formal definition is stated below.

Definition 2.1 (Differential privacy). *A randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for any pair of neighboring datasets D and D' differing in one record, and for all $\mathcal{S}$ that is a subset*

Algorithm 1 DP-SGD

Require: Data set $D = \{d_1, \cdots, d_n\}$, model with p parameters, loss function: $\ell : \mathbb{R}^p \times D \to \mathbb{R}$, gradient ℓ_2-norm bound: L, number of iterations: T, noise variance: σ^2, learning rate: η.

1: Randomly initialize θ_0.
2: **for** $t = 0, \ldots, T-1$ **do**
3: Randomly sample a batch of examples $B \subseteq D$.
4: $g_t \leftarrow \frac{1}{|B|} \sum_{d \in B} \text{clip}\left(\nabla \ell(\theta_t; d), L\right) + \mathcal{N}\left(0, \sigma^2\right)$ where $\text{clip}\left(x, L\right) = \min\{1, L/\|x\|_2\} \cdot x$.
5: $\theta_{t+1} \leftarrow \theta_t - \eta g_t$.
6: **end for**
7: **return** $\theta_0, \ldots, \theta_T$.

of the range of $\mathcal{A}$, we have

$$\mathbf{Pr}\left(\mathcal{A}(D) \in \mathcal{S}\right) \leq e^{\varepsilon} \mathbf{Pr}\left(\mathcal{A}(D') \in \mathcal{S}\right) + \delta,$$

where the probability is over the randomness of $\mathcal{A}$.

Usually, ε is assumed to be a small constant, and $\delta \ll 1/|D|$ where $|D|$ is the size of dataset D.

Another related definition is called Rényi differential privacy (RDP) [19].

Definition 2.2 (Rényi differential privacy). *A randomized algorithm $\mathcal{A}$ is (α, ε)-RDP if for any pair of neighboring datasets D and D' differing in one record, we have $D_\alpha(\mathcal{A}(D) \| \mathcal{A}(D')) \leq \varepsilon$ where*

$$D_\alpha(\mathcal{A}(D) \| \mathcal{A}(D')) = \frac{1}{\alpha - 1} \log \mathop{\mathbf{E}}_{s \sim \mathcal{A}(D)} \left(\frac{\mathbf{Pr}\left(\mathcal{A}(D) = s\right)}{\mathbf{Pr}\left(\mathcal{A}(D') = s\right)} \right)^{\alpha - 1}$$

is called the Rényi divergence of order α between distributions $\mathcal{A}(D)$ and $\mathcal{A}(D')$.

If an algorithm guarantees (α, ε)-RDP, it also guarantees $\left(\varepsilon + \frac{\log(1/\delta)}{\alpha - 1}, \delta\right)$-differential privacy [1, 19]. Tighter conversions are provided in followup works [2, 4].

DP-SGD: To achieve differential privacy for machine learning models, a standard approach is to use differentially private stochastic gradient descent (DP-SGD), described in Algorithm 1. Its main difference from vanilla SGD is in Step 4, where we clip the per-example gradient and add Gaussian noise to the aggregated gradient. It has been shown that DP-SGD guarantees differential privacy, with ε and δ values determined by the noise value σ, the sampling ratio $|B|/|D|$, and the number of iterations [1]. Note that this guarantee of differential privacy applies to the model parameters after each step, and thus allows for a powerful adversary that can deduce the gradients for each step of training.

2.3 Secret Sharer Attack

Since our algorithm does not guarantee rigorous differential privacy, we evaluate the model's privacy with another metric – the secret sharer attack [5]. The secret sharer attack is designed to measure unintended memorization by a model of its input data. The idea is to purposely inject some out-of-distribution training samples, called "canaries", that do not help with generalization, and check whether the model has memorized them by comparing their losses with those of similar samples that are not injected. For example, in a language model, we can inject "My SSN is 123-45-6789" into the training data, and check if its perplexity is much smaller than those of other sequences of the form "My SSN is □□□-□□-□□□□". If the perplexity for the injected example is indeed much lower, it indicates the model can potentially leak the sequence to an adversary.

More specifically, we compute a metric called exposure for the canary, defined as follows: Suppose we have generated n other sequences and among them, the perplexity of the canary ranks at the m-th position. Then its exposure is defined as $\log_2(n) - \log_2(m)$. One might notice that if n is not large enough, the canary's perplexity can easily rank first, and this formula does not capture the difference between two canaries whose perplexities are both small but different. Therefore, [5] also proposes fitting the perplexities with some distribution and using the CDF at the perplexity of the canary as an analogue for m/n.

The secret sharer attack has been used to evaluate privacy properties of models when there is a lack of good differential privacy guarantee, for example, in [31].

3 EMBEDDING-AWARE NOISE ADDITION

In practice, using DP-SGD has serious negative effects on both model performance and training speed (measured in steps per second). The degradation in utility can be largely attributed to the addition of Gaussian noise, while one major factor in training slowdown is the need to compute per-example gradients. A considerable body of experience with established models on standard datasets has shown that these obstacles frequently make it very difficult to train DP-SGD models with both acceptable privacy and utility efficiently.

Both of these problems take on additional characteristics in models with large embedding layers, where the natural gradient of a training example is zero in all rows of the embedding that are not affected by that example. In terms of utility, adding noise to zero gradients means that model weights are distorted by noise without a chance for the model to correct them. This is necessary to satisfy the threat model of DP-SGD, which assumes that an adversary can observe the state of the model after each update. With this assumption, if noise is *not* added to all model weights in the embedding layer, an adversary can easily infer which items in the embedding vocabulary were present in the input batch.

The impact on training speed is more implementation-specific. One common approach for training large models in a data parallel fashion is to use parameter servers. This approach stores the current model weights on a collection of parameter servers. Worker machines then iteratively read and update these weights using their input data. In standard training for embedding models, the weights that are read and updated are sparse in nature, leading to efficient communication and computation. When using DP-SGD, however, noise is added to every model weight, converting sparse values to large dense values. For large embedding models, this can lead to a prohibitive increase in communication and computation time.

The above considerations motivate a modified version of DP-SGD, which we call embedding-aware noise addition, or EANA (Algorithm 2) for short, in which we only add noise to the nonzero gradients of the embedding layer. This has the simultaneous benefits of keeping the noisy gradients sparse for speed purposes while not harming the parts of the embedding layer that are not affected by

Algorithm 2 EANA

Require: Data set $D = \{d_1, \cdots, d_n\}$, model with p non-embedding parameters and e embedding parameters, loss function: $\ell : \mathbb{R}^{p+e} \times D \to \mathbb{R}$, gradient ℓ_2-norm bound: L, number of iterations: T, noise variance: σ^2, learning rate: η.

1: Randomly initialize θ_0.
2: **for** $t = 0, \ldots, T-1$ **do**
3: Randomly sample a batch of examples $B \subseteq D$.
4: $g_t \leftarrow \frac{1}{|B|} \sum_{d \in B} \text{clip}\left(\nabla \ell(\theta_t; d), L\right) + \delta_E \odot \mathcal{N}\left(0, \sigma^2 \mathbb{I}\right)$ where $\text{clip}\left(x, L\right) = \min\{1, L/\|x\|_2\} \cdot x$, $\delta_E \in \{0,1\}^{p+e}$ is 1 for embedding parameters and 0 elsewhere, and $\odot$ represents the element-wise product between two vectors.
5: $\theta_{t+1} \leftarrow \theta_t - \eta g_t$.
6: **end for**
7: **return** θ_T.

the input batch. The drawback is that, as mentioned above, the resulting algorithm is no longer differentially private against an adversary who can observe the model weights during training.

3.1 Theoretical Analysis

Intuitively, EANA does not guarantee differential privacy against an adversary that can observe intermediate models, since the adversary can use the positions of nonzero gradients to infer which vocabulary items were used in a given time step. More specifically, for zero coordinates, the algorithm is deterministic and thus would not give any DP guarantee. However, in the very common situation where an adversary can only observe the final trained model, we might expect the adversary to learn much less information, because by the time the model is fully trained, almost every coordinate has been updated sometime in the training process. Consequently, the noise added in the update might give a certain level of privacy.

We demonstrate this by a simple linear loss function $\ell(w, x) = \langle w, x \rangle$ where $w \in \mathbb{R}^d$ is the model and $x \in \mathbb{R}^d$ is the data. The gradient of this function is $\nabla \ell(w, x) = x$. Consider a dataset $D = \{x_1, \ldots, x_n\}$ where $\|x_i\|_2 \le L$ (either naturally or through clipping). For $j \in [d]$, let x_i^j denote the j-th coordinate of example x_i. Let $C^j(D) = |\{x_i^j \neq 0 : x_i \in D\}|$ be the number of examples in D that are nonzero in their j-th coordinate. If we run EANA to minimize $\mathcal{L}(w, D) = \frac{1}{|D|} \sum_{x_i \in D} \ell(w, x_i)$ with one pass of the dataset D, we can easily see that the j-th coordinate of w would be the summation of $\sum_{i=1}^n x_i^j$ and $C^j(D)$ pieces of independent Gaussian noise. When some x_i changes such that x_i^j goes from zero to nonzero, then w^j might change by x_i^j with one more piece of Gaussian noise. Intuitively, the larger $C^j(D)$ is, the more w^j behaves similarly as a Gaussian with mean shifted. Therefore, if we know a priori that $C^j(D)$ is large enough for all coordinates, we might be able to get some privacy guarantee.

In contrast to vanilla DP which considers arbitrary pairs of neighboring dataset D and D', here, the privacy guarantee is "*data-dependent*", as we require the dataset D to satisfy a certain property. Such relaxations of DP are not rare in past work [8, 21–23, 27].

Theorem 3.1. *Let EANA(D) denote the final model obtained by running EANA on the linear loss described above. Let* $d(D) = \max_{x_i \in D} |\{x_i^j \neq 0 : j \in [d]\}|$ *be the maximum number of non-zero coordinates over all examples in* D*. For any pair of neighboring datasets* D *and* D' such that for all $j \in [d]$, $C^j(D), C^j(D') \ge C_0$, and $d(D), d(D') \le d_0 \le d$, *we have*

$$D_\alpha(EANA(D) \| EANA(D')) \le \frac{\alpha L^2}{2(C_0 + 1 - \alpha)\sigma^2} + \frac{d_0}{2} \ln\left(1 + \frac{1}{C_0}\right) + \frac{d_0}{2(\alpha - 1)} \ln \frac{C_0 + 1}{C_0 + 1 - \alpha}. \quad (1)$$

for all integers α *such that* $\alpha \le C_0$.

Proof of Theorem 3.1. The gradient of $\ell(w, x) = \langle w, x \rangle$ is x. Given a sample x_i, EANA would add noise $\mathcal{N}(0, \sigma^2)$ at the nonzero coordinate of x_i and 0 elsewhere. Therefore, after enumerating through the dataset D, the final model w is such that $w^j = w_0^j - \eta\left(\sum_i x_i^j + \mathcal{N}(0, C^j(D)\sigma^2)\right)$ where η is the learning rate, and $C^j(D)$ is the number of examples with nonzero j-th coordinate. Denote the term in the bracket as $\Delta^j(D)$ and the vector $\left(\Delta^0(D), \ldots, \Delta^d(D)\right)$ as $\Delta(D)$. For ease of notation, we will abbreviate $C^j(D)$ as C^j.

Consider a dataset $D' = D \cup \{x_*\}$. To analyze the privacy property of w, we need to compare the distribution of $\Delta(D)$ and $\Delta(D')$. Since each coordinate of $\Delta(D)$ is independent, we can consider them separately. Let S_* be the set of indices where x_* is nonzero and let $d_* = |S_*| \le d_0$. For $j \notin S_*$, x_*^j is 0 and $\Delta^j(D')$ follows the same distribution as $\Delta^j(D)$. For $j \in S_*$, $\Delta^j(D') = \sum_i x_i^j + x_*^j + \mathcal{N}(0, (C^j + 1)\sigma^2)$, and we essentially need to compare $\mathcal{N}(0, C^j\sigma^2)$ and $\mathcal{N}(x_*^j, (C^j + 1)\sigma^2)$.

Consider the Rényi divergence between the distribution of $\Delta(D)$ and $\Delta(D')$. We have

$$\begin{aligned} D_\alpha(\Delta(D) \| \Delta(D')) &= \sum_{j \in [d]} D_\alpha(\Delta^j(D) \| \Delta^j(D')) \\ &= \sum_{j \in S_*} D_\alpha(\mathcal{N}(0, C^j\sigma^2) \| \mathcal{N}(x_*^j, (C^j + 1)\sigma^2)) \end{aligned} \quad (2)$$

and similarly for $D_\alpha(\Delta(D') \| \Delta(D))$. We can define the term corresponding to the j-th coordinate as D_α^j, which can be viewed as the privacy guarantee of coordinate j.

According to [25, Eqn (10)], the Rényi divergence between $\mathcal{N}(\mu_0, \sigma_0^2)$ and $\mathcal{N}(\mu_1, \sigma_1^2)$ at integer α is bounded if $\sigma_\alpha^2 = \alpha\sigma_0^2 + (1 - \alpha)\sigma_1^2$ is positive, and the divergence is $\frac{\alpha(\mu_0 - \mu_1)^2}{2\sigma_\alpha^2} + \frac{1}{1-\alpha} \ln \frac{\sigma_\alpha}{\sigma_0^{1-\alpha}\sigma_1^\alpha}$. In our case, $\sigma_\alpha^2 = (C^j + 1 - \alpha)\sigma^2$, and thus the divergence is finite when $\alpha < C^j + 1$. We have

$$\begin{aligned} D_\alpha^j =& D_\alpha(\mathcal{N}(0, C^j\sigma^2) \| \mathcal{N}(x_*^j, (C^j + 1)\sigma^2)) \\ =& \frac{\alpha\left(x_*^j\right)^2}{2(C^j + 1 - \alpha)\sigma^2} + \frac{1}{2(\alpha - 1)} \ln \frac{\left(C^j\right)^{1-\alpha}(C^j + 1)^\alpha}{C^j + 1 - \alpha} \quad (3) \\ \le& \frac{\alpha\left(x_*^j\right)^2}{2(C_0 + 1 - \alpha)\sigma^2} + \frac{1}{2(\alpha - 1)} \ln \frac{C_0^{1-\alpha}(C_0 + 1)^\alpha}{C_0 + 1 - \alpha}. \quad (4) \end{aligned}$$

where the last inequality holds because (3) is decreasing with respect to C^j. Summing up over $j \in S_*$, we have

$$(2) \leq \frac{\alpha \sum_j \left(x_*^j\right)^2}{2(C_0+1-\alpha)\sigma^2} + \frac{d_*}{2(\alpha-1)} \ln \frac{C_0^{1-\alpha}(C_0+1)^\alpha}{C_0+1-\alpha}$$
$$\leq \frac{\alpha L^2}{2(C_0+1-\alpha)\sigma^2} + \frac{d_0}{2(\alpha-1)} \ln \frac{C_0^{1-\alpha}(C_0+1)^\alpha}{C_0+1-\alpha} \quad (5)$$
$$= \frac{\alpha L^2}{2(C_0+1-\alpha)\sigma^2} + \frac{d_0}{2} \ln\left(1+\frac{1}{C_0}\right) + \frac{d_0}{2(\alpha-1)} \ln \frac{C_0+1}{C_0+1-\alpha}. \quad (6)$$

The direction for $D_\alpha(\Delta(D')\|\Delta(D))$ is easier. We have

$$D_\alpha(\mathcal{N}(x_*^j, (C^j+1)\sigma^2)\|\mathcal{N}(0, C^j\sigma^2)) = \frac{\alpha\left(x_*^j\right)^2}{2(\alpha+C^j)\sigma^2} + \frac{1}{2(\alpha-1)} \ln \frac{(C^j+1)^{1-\alpha}\left(C^j\right)^\alpha}{\alpha+C^j}, \quad (7)$$

which is smaller than (3). It is therefore sufficient to analyze $D_\alpha(\Delta(D)\|\Delta(D'))$.

Another way for analyzing the algorithm is to look at each per-coordinate guarantee D_α^j in (3) individually. We have

$$D_\alpha^j \leq \frac{\alpha L^2}{2(C^j+1-\alpha)\sigma^2} + \frac{1}{2}\ln\left(1+\frac{1}{C^j}\right) + \frac{1}{2(\alpha-1)} \ln \frac{C^j+1}{C^j+1-\alpha} \quad (8)$$

which says that coordinate with larger C^j will have less privacy risk. Notice that if we compose (8) over coordinates, we would encounter some overhead (at most an additional d_* factor) in the first term. However, if we know *a priori* that some coordinates have large C^j, such a composition might provide a better tradeoff. □

Remark 1: How does this bound compare to the privacy guarantee of DP-SGD? DP-SGD adds Gaussian noise with the same variance to each coordinate, i.e., $\forall j \in [d]$, $w^j = w_0^j - \eta\left(\sum_i x_i^j + \mathcal{N}(0,\sigma_g^2)\right)$ for some σ_g. Releasing the final w with DP-SGD would guarantee $\left(\alpha, \frac{\alpha L^2}{2\sigma_g^2}\right)$-RDP, or $\left(\frac{L^2}{2\sigma_g^2} + \frac{\sqrt{2\log(1/\delta)}L}{\sigma_g}, \delta\right)$-DP. A natural baseline is to consider the case where all $C^j(D) = C_0$, and set $\sigma_g = \sqrt{C_0}\sigma$, such that the noise added is the same as EANA. In this case, DP-SGD should give better ε, as it is essentially the Gaussian mechanism. Yet we would hope the ε from EANA to be comparable.

In Figure 2, we consider the *data-dependent* (ε, δ) privacy guarantee, i.e., the value of ε such that $\mathbf{Pr}(\text{EANA}(D) \in \mathcal{S}) \leq e^\varepsilon \mathbf{Pr}(\text{EANA}(D') \in \mathcal{S}) + \delta$ for the D, D' pair under the conditions in Theorem 3.1. We fix $d_0 = 10, L = 1$ and vary C_0 and α. We can see that DP-SGD always gives better privacy, while the gap between DP-SGD and EANA shrinks as σ decreases and C_0 increases.

It might look confusing that releasing the same output, i.e, $w^j = w_0^j - \eta\left(\sum_i x_i^j + \mathcal{N}(0, C_0\sigma^2)\right)$ gives different privacy guarantees for the two algorithms. The reason is that the privacy guarantee is a property of an algorithm instead of one particular run of it. We are analyzing the privacy guarantee for two different algorithms, which happen to yield the same output. More specifically, even if their outputs are the same on one particular D, their behavior would be different on a neighboring dataset D'.

Remark 2: Even though the privacy guarantee of EANA is worse than that of DP-SGD under the above setting, EANA might provide better per-instance guarantee. The proof of Theorem 3.1 naturally gives a per-instance RDP guarantee similar to that in [27], which is essentially defined over the $(\mathcal{A}, D, x_*)$-tuple, with $\mathcal{A}$ being the algorithm, and x_* being the sample that differs in D and D', i.e., $D' = D \cup \{x_*\}$. Let S_* be the set of indices where x_* is nonzero and let $d_* = |S_*| \leq d_0$. In such case, we have

$$\max\left\{D_\alpha(\text{EANA}(D)\|\text{EANA}(D')), D_\alpha(\text{EANA}(D')\|\text{EANA}(D))\right\} \leq \sum_{j\in S_*} \frac{\alpha\left(x_*^j\right)^2}{2(C^j(D)+1-\alpha)\sigma^2} + \frac{1}{2}\ln\left(1+\frac{1}{C^j(D)}\right) + \frac{1}{2(\alpha-1)}\ln\frac{C^j(D)+1}{C^j(D)+1-\alpha} \quad (9)$$

where x_*^j is the j-th coordinate of x_*. This bound implies (1) using $C^j(D) \geq C_0$ and $\sum_{j\in[d]}\left(x_*^j\right)^2 \leq L^2$. From the per-instance RDP bound, we can further see that

- If the nonzero coordinates in x_* mostly have large counts, for examples, x_* might be a sentence consisting of popular words, its privacy risk is smaller.
- Even for *data-dependent* privacy that is not specific to one example, if we know the per-coordinate bound C^j, we can still potentially get a tighter bound compared to (1).

4 EVALUATION

We evaluate the efficacy of EANA on two recommendation tasks. The first is the next movie prediction, where we train and evaluate a dual encoder model on a public dataset (MovieLens 20M[1] [15]). The model takes a user's movie-watching history and predicts the next movie for this user. The second task trains a real-world, large-scale dual encoder model on an industry dataset. It generates knowledge graph entity embeddings for downstream recommendation models. Below we provide details on the datasets, model architectures, and experiment settings for each task. We analyze the experiment results on training speed, model quality, and privacy protection.

Table 1: MovieLens 20M Dataset Statistics.

	MovieLens 20M
Ratings	20,000,263
Users	138,493
Movies	27,278

Table 2: Industry Large-scale Dataset Statistics.

	Industry-100K	Industry-5M
History	1 year	1 year
Individuals	4,556,149,471	4,556,149,471
Vocabulary	100K	5M

[1] https://grouplens.org/datasets/movielens/20m/

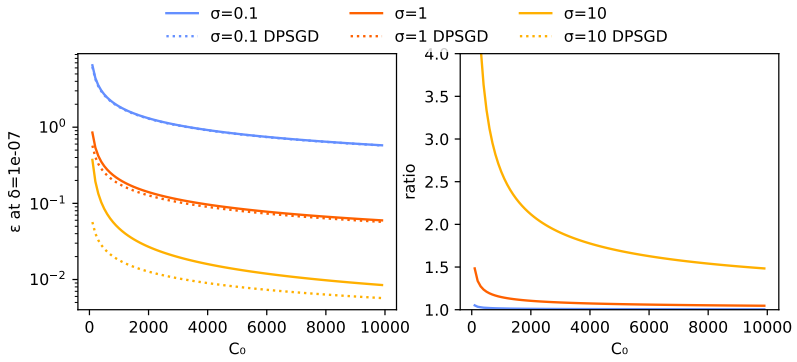


Figure 2: Left: ε at $\delta = 10^{-7}$. Right: ratio between ε of EANA and DP-SGD.

4.1 Next Movie Prediction on MovieLens 20M Dataset

This task examines how noise addition affects the model quality and how much privacy benefit EANA provides. It also studies the differences between EANA and DP-SGD from model quality and privacy benefits perspectives. The vocabulary size for the ID-based next movie prediction model is 27278 (the number of movies in MovieLens 20M as shown in Table 1). The model is of moderate size so that it can be trained with only a few workers or even a single machine. The communication overhead resulting from training with DP-SGD is not huge, and the slowdown is tolerable (see training speed in Table 3). In addition, the time it takes for the model to converge is shorter, making it possible to train with DP-SGD. Therefore, we compare the performance of EANA and DP-SGD on this task.

4.1.1 Dataset. The MovieLens 20M dataset contains approximately 20 million ratings from 138493 users on 27278 movies (Table 1). We generate a movie-watching timeline for each user by stitching the rated movie ID corresponding to that user together, sorted by ascending timestamps. We use timelines from 80% of the users for training and the remaining 20% of users' timelines for testing. To generate an example for training or testing, we randomly sample a movie ID from a user's timeline and use it as a label. The context is a sequence of 20 movie-watches before the label movie ID in the same timeline. The examples are sampled on-the-fly during the training and testing.

4.1.2 Model Architecture. Figure 3 illustrates the ID-based dual encoder model for the next movie prediction task. The model takes a movie ID sequence (watch history) as the context and the next movie ID as the label to form a *(context, label)* input pair. Note that both the context and label contain only movie IDs. The context tower is a bag-of-words encoder, while the label tower performs a simple embedding lookup. The two towers generate context and label embeddings, respectively. The similarity between the context and label embeddings is computed and fed into the softmax loss function to train the model. We use 64 as the batch size for all experiments in this task. The output dimensions of the context and label embedding layers are both 128. The encoded context and label embeddings are also 128-dimensional.

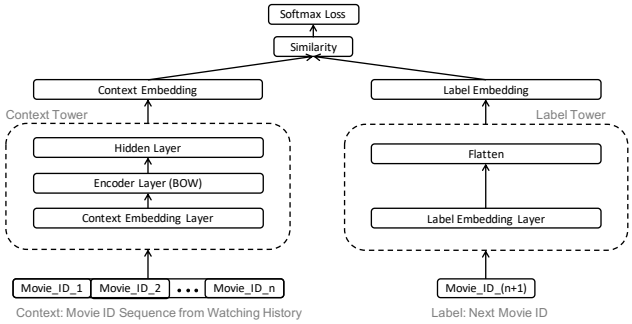


Figure 3: ID-based dual encoder model for next movie prediction. The context tower encodes the user's previous activities (i.e., previous watches), and the label tower encodes the label candidates. During training, items interacting with similar users will be pulled together in the embeddings space. In contrast, embeddings for dissimilar pairs are pushed apart.

4.1.3 Experiment Settings and Results. We train the models with SGD, DP-SGD, and EANA. The baseline is the model performance when training with SGD without adding noise. For DP-SGD and EANA, we experiment with two different noise multipliers: 0.01 and 0.001. In all the experiments, we measure test precision@k and recall@k with $k \in [1, 5, 10, 20]$. This subsection presents precision@k for model quality. The trends in the recall results are similar to what we observed with precision@k. We use secret sharer to measure the model's potential for unintended memorization of unique or rare examples in the training dataset and thus to evaluate the privacy benefit. We inject 9750 canary examples, 50 canaries for each number of occurrences in [1, 2, 3, 4, 5, 10, 20, 50, 100], into the training examples. The number of non-inserted random examples to compute the exposure metric is 16384.

Model Quality: Figure 4 shows the model precision with the settings mentioned above. When the noise multiplier is small (e.g., 0.001), both DP-SGD and EANA lead to similar model performance compared to the baseline model trained with SGD. Increasing the

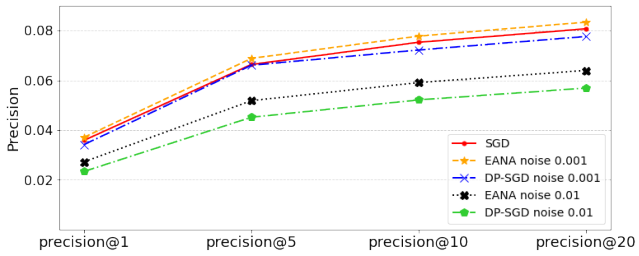


Figure 4: Precision of the models trained with and without DP-SGD, EANA for the next movie prediction task. This figure shows the results of two different noises (0.01 and 0.001) when applying DP and EANA. Note that a small noise with EANA slightly improves model precision, but larger noise reduces precision.

noise (e.g., when the noise multiplier is 0.01) results in slight performance degradation (as illustrated with the black dotted and green dot-dashed curves), but the overall performance is still acceptable. Note that EANA (black dotted curve) leads to slightly better model performance compared to DP-SGD (green dot-dashed curve) when adding more noise to the model training process.

Model Privacy: Figure 5 presents the secret sharer exposure results. When the canaries are rarer (e.g., with an occurrence number of 1 or 2), the measured exposures are small for all settings. The models show no signs of memorization. When the canaries are inserted at a higher frequency (e.g., with an occurrence number of 50 or 100), exposures increase. The inserted canaries become more likely to be memorized by the model than non-inserted canaries. Adding noise with both DP-SGD and EANA significantly reduces the exposures compared to SGD. With a noise multiplier of 0.01, both DP-SGD and EANA demonstrate great empirical privacy protections by looking at the exposure results. Models trained with EANA tend to have higher exposures than those trained with DP-SGD but still show much less unintended memorization than those without noise addition (SGD). Reducing the noise multiplier to 0.001 also reduces the privacy protection, especially for EANA.

We conclude that both DP-SGD and EANA effectively reduce the privacy risk of the movie prediction model. EANA performs slightly worse than DP-SGD but still shows excellent privacy benefits compared to SGD. Therefore, when the model size becomes large such that training with DP-SGD is infeasible, EANA becomes the best choice (see Section 4.2). Decreasing the noise multiplier leads to better model quality but reduces privacy protection. There is always a trade-off between model quality and privacy protection.

4.2 Knowledge Graph Entity Embedding Generation on Industry Large-scale Dataset

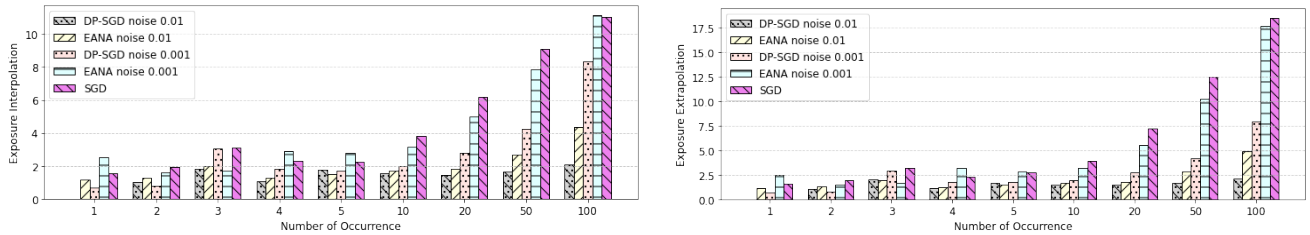
In this subsection, we study how EANA performs on a real-world, large-scale model training with a large-scale industry dataset. Knowledge graphs (KG) are also known as semantic networks. They encode structured information of real-world entities - i.e., objects, events, situations, or abstract concepts - and their rich relations. A KG entity embedding is a condensed vector representation of this entity trained from a large-scale embedding-based model (a dual encoder model in this paper). Each entity is projected to the embedding space such that semantically related entities are clustered together. A typical knowledge graph may contain millions of entities. Therefore, the KG entity embedding generation model needs to have large embedding tables to encode those entities, resulting in a large-scale embedding-based model.

4.2.1 Dataset. We use a large-scale industry dataset (see Table 2) to train the KG embedding generation model. The dataset contains timelines of billions of individuals. Each individual's timeline is a sequence of this individual's daily activities for one year, sorted by descending timestamps. The context of each training example contains three different features extracted from each activity, such as the KG entities, unigrams, or bigrams. The label is a KG entity. The process of generating training and testing examples is similar to the one described in Section 4.1.1. 90% of the timelines are used for training, and the remaining ones are for testing. We experiment with two variations of the dataset: Industry-100K and Industry-5M. The model trained on Industry-100K uses only the top 100K most frequently appearing items for each context feature and learns the embeddings for the top 100K KG entities. The corresponding embedding tables have 100K embeddings for each feature or the label. We refer to the 100K as the vocabulary size. Industry-5M has a vocabulary size of 5 million, so the corresponding embedding table sizes, and thus the model size, are much larger than those used for Industry-100K. The models used for both industry dataset variations are larger than those used for the next movie prediction task.

4.2.2 Model Architecture. The KG entity embedding generation model is also a dual encoder model, similar to the one presented in Section 4.1.2. The key differences are (1) the context input, (2) the model size, and (3) the context tower encoder. As mentioned in Section 4.2.1, the context input consists of three features instead of the activity ID. The model size, determined by the vocabulary size, is much larger than the one for the next movie prediction. Also, the model uses a transformer as the context tower encoder to learn better entity representations with such a huge vocabulary size.

4.2.3 Experiment Settings and Results. The experiment settings for the embedding generation task are similar to the next movie prediction task as described in Section 4.1.3. For embedding generation, we train models on two variations of the dataset: Industry-100K and Industry-5M. When performing the secret sharer tests, we inject 1195000 canaries to the training examples, 1000 canaries for each number of occurrences in $[1, 2, 3, 4, 5, 10, 20, 50, 100, 1000]$. We show the secret sharer test results on the models trained with the Industry-5M dataset in this section.

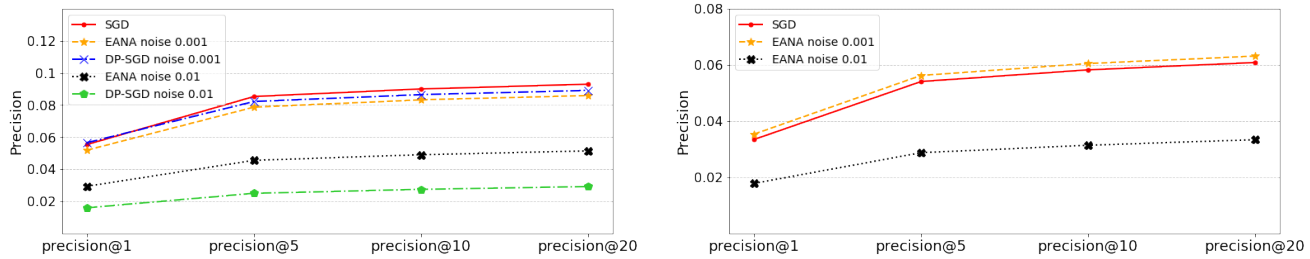
Training Speed: The last two columns in Table 3 present the average training speeds with three different algorithms on the two variations of the industry dataset. Adding noise to model parameters affects the training speed. Thus we see slowdowns for both EANA and DP-SGD compared to SGD. However, the training speed with DP-SGD drops more severely due to the communication overhead between the workers and parameter servers, as explained in Section 3. With a vocabulary size of 100K, training with EANA is 4.7X faster than training with DP-SGD. The difference gets more



(a) Exposure: interpolation.

(b) Exposure: extrapolation.

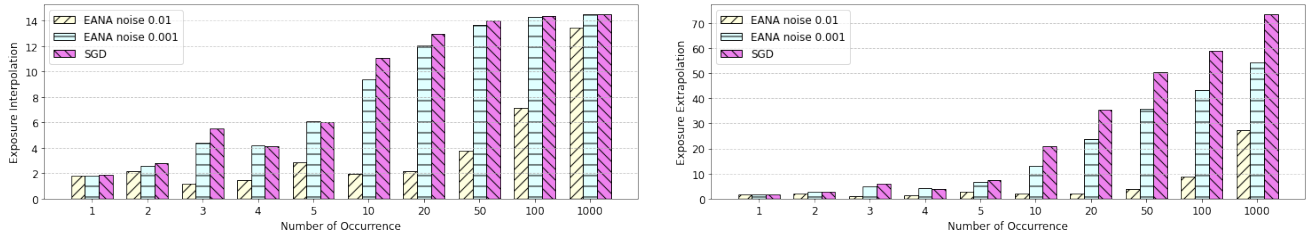
Figure 5: Secret sharer results on models trained with and without DP-SGD and EANA. Observe that for a noise of 0.01, the exposure for DP-SGD and EANA is comparable and is significantly reduced compared with SGD with no added noise.



(a) Industry-100K

(b) Industry-5M

Figure 6: Precision of the models trained with or without DP-SGD and EANA for the KG entity embedding generation task. (a) shows the results for SGD, DP-SGD and EANA with two different noise values when the vocabulary size is 100K. (b) shows the results of two different noise values (0.01 and 0.001) when applying EANA for 5M vocabulary size. Note that we cannot show traditional DP-SGD for 5M since training-speed is too slow to be feasible. When the noise is 0.001, model trained with EANA is as good as the one trained without DP. When we increase the noise to 0.01, the model precision drops, but is still reasonable and acceptable for downstream recommendation tasks.



(a) Exposure: interpolation

(b) Exposure: extrapolation

Figure 7: Secret sharer results on models trained with or without EANA for different noise values on Industry-5M dataset. Note that, we don't have exposure values for traditional DP-SGD for this model size.

significant when the vocabulary size increases to 5M, where EANA is 54X faster than DP-SGD. Note that the real-world production model uses 5M as vocabulary size. In this case, DP-SGD leads to a training speed of 1.3 steps per second. A real-world production model usually trains for millions (e.g., 20 million) of steps to get high-quality embeddings, and those embeddings need to be refreshed periodically. Hence, it is infeasible to train a large-scale production model with DP-SGD due to the extremely slow training speed. Even though the training speed with EANA is slower than training without adding noise (70 steps/s v.s. 232 steps/s), it is still possible to finish the training within a reasonable time. Therefore, EANA helps to unblock training a large-scale embedding-based model for production while still providing privacy protection. We will discuss more on the privacy aspect later in this section.

Model Quality: Figure 6 shows the precision results with the two dataset variations. For Industry-100K, the results are similar to the next movie prediction task (Section 4.2.3). With a noise multiplier of 0.001, both EANA and DP-SGD perform similar to SGD. When increasing the noise multiplier to 0.01, the model quality drops but is still reasonable. Note that EANA leads to better model quality than DP-SGD when the noise multiplier is larger (e.g., 0.01),

Table 3: Training Speed.

	MovieLens 20M	Industry-100K	Industry-5M
SGD	363 steps/s	236 steps/s	232 steps/s
DP-SGD	78 steps/s	37steps/s	1.3 steps/s
EANA	223 steps/s	175 steps/s	70 steps/s

consistent with what we observed in the next movie prediction task. For Industry-5M, we only present the precision results for SGD and EANA because the training processes with DP-SGD are too slow to produce any meaningful results. Again we observe that adding a smaller amount of noise (with a noise multiplier of 0.001) leads to comparable model performance with SGD. A larger noise multiplier (0.01) gives a reasonable but slightly worse model quality.

Model Privacy: Figure 7 gives the exposure results for secret sharer experiments with the Industry-5M dataset. It is easy to see that EANA with a noise multiplier of 0.01 significantly reduces the exposures compared to SGD. When the noise multiplier is smaller (0.001), the model trained with EANA has much higher exposure results but still performs slightly better than the model trained with SGD. It indicates that models trained with EANA are less likely to memorize training examples unintended and thus trigger fewer privacy concerns.

In summary, we demonstrate that EANA is much more efficient than DP-SGD for large-scale embedding-based models, moving the training from the realm of the impossible to the possible. Training with EANA can achieve good model quality while reducing the privacy risk.

5 CONCLUSION

This work proposes embedding-aware noise addition (EANA), a technique that only adds noise to parameters with non-zero gradients at each training step to address the slowness issue when training large-scale embedding-based deep neural networks with DP-SGD. EANA not only significantly improves the training speed but also preserves analytical and empirical privacy. We demonstrate the effectiveness of EANA on both the public dataset and a large-scale industry dataset. This technique makes it feasible to train real-world, large-scale embedding-based models with good practical privacy protection.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS'16)*. 308–318.

[2] Shahab Asoodeh, Jiachun Liao, Flavio P Calmon, Oliver Kosut, and Lalitha Sankar. 2020. A better bound gives a hundred rounds: Enhanced privacy guarantees via f-divergences. In *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 920–925.

[3] Zhiqi Bu, Sivakanth Gopi, Janardhan Kulkarni, Yin Tat Lee, Hanwen Shen, and Uthaipon Tantipongpipat. 2021. Fast and memory efficient differentially private-sgd via jl projections. *Advances in Neural Information Processing Systems* 34 (2021).

[4] Clément Canonne, Gautam Kamath, and Thomas Steinke. 2020. The discrete gaussian for differential privacy. *arXiv preprint arXiv:2004.00010* (2020).

[5] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*. 267–284.

[6] Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836* (2018).

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)*. 191—-198.

[8] Rachel Cummings and David Durfee. 2020. Individual sensitivity preprocessing for data privacy. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 528–547.

[9] Jian Du, Song Li, Moran Feng, and Siheng Chen. 2021. Dynamic Differential-Privacy Preserving SGD. *arXiv preprint arXiv:2111.00173* (2021).

[10] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology—EUROCRYPT*. 486–503.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*. 265–284. http://dx.doi.org/10.1007/11681878_14

[12] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506* (2019).

[13] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space. *arXiv preprint arXiv:1811.08008* (2018).

[14] Ian Goodfellow. 2015. Efficient Per-Example Gradient Computations. *arXiv preprint arXiv:1510.01799* (2015).

[15] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[16] Jyun-Yu Jiang, Tao Wu, Georgios Roumpos, Heng-Tze Cheng, Xinyang Yi, Ed Chi, Harish Ganapathy, Nitin Jindal, Pei Cao, and Wei Wang. 2020. End-to-End Deep Attentive Personalized Item Retrieval for Online Content-sharing Platforms. In *Proceedings of The Web Conference 2020*. 2870–2877.

[17] Manas Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay Adams, Pranav Khaitan, Jiahui Liu, and Quoc Le. 2020. Neural Input Search for Large Scale Recommendation Models. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*. 2387—-2397.

[18] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[19] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 263–275.

[20] Lin Ning, Karan Singhal, Ellie X. Zhou, and Sushant Prakash. 2021. Learning Federated Representations and Recommendations with Limited Negatives. (2021). https://arxiv.org/abs/2108.07931

[21] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 75–84.

[22] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).

[23] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908* (2018).

[24] Pavel Sountsov and Sunita Sarawagi. 2016. Length bias in encoder decoder models and a case for global conditioning. *arXiv preprint arXiv:1606.03402* (2016).

[25] Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

[26] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems.. In *NIPS*. 4957–4966.

[27] Yu-Xiang Wang. 2019. Per-instance differential privacy. *Journal of Privacy and Confidentiality* 9, 1 (2019).

[28] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *Companion Proceedings of the Web Conference 2020*. 441–447.

[29] Yinfei Yanga, Steve Yuanc, Daniel Cera, Sheng-yi Konga, Noah Constanta, Petr Pilarc, Heming Gea, Yun-Hsuan Sunga, Brian Stropea, and Ray Kurzweila. 2018. Learning Semantic Textual Similarity from Conversations. *ACL 2018* (2018), 164.

[30] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, and Ed Chi (Eds.). 2019. *Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations*.

[31] Huanyu Zhang, Ilya Mironov, and Meisam Hejazinia. 2021. Wide network learning with differential privacy. *arXiv preprint arXiv:2103.01294* (2021).