Federated Variational Autoencoder for Collaborative Filtering

Federated Variational Autoencoder for Collaborative Filtering

Mirko Polato
Department of Mathematics, University of Padova
Padova, Italy
mpolato@math.unipd.it

***Abstract*—Recommender Systems (RSs) are valuable technologies that help users in their decision-making process. Generally, RSs are designed with the assumption that a central server stores and manages historical users' behaviors. However, users are nowadays more aware of privacy issues leading to a higher demand for privacy-preserving technologies. To cope with this issue, the Federated Learning (FL) paradigm can provide good performance without harming the users' privacy. Some efforts have been devoted to adapt standard collaborative filtering methods (e.g., matrix factorization) into the FL framework in recent years.**

In this paper, we present a Federated Variational Autoencoder for Collaborative Filtering (FedVAE), which extends the state-of-the-art MultVAE model. Additionally, we propose an adaptive learning rate schedule to accelerate learning. We also discuss the potential privacy-preserving capabilities of FedVAE. An extensive experimental evaluation on five benchmark data sets shows that our proposal can achieve performance close to MultVAE in a reasonable number of iterations. We also empirically demonstrate that the adaptive learning rate guarantees both accelerated learning and good stability.

***Index Terms*—federated learning, variational autoencoder, collaborative filtering, recommender systems, top-n item recommendation**

I. Introduction

With the introduction of the General Data Protection Regulation [1] (GDPR[1]) in 2018, both service providers and users have become more concerned about privacy. This concern also had a considerable impact on the scientific community that intensified the efforts to find privacy-preserving solutions. The privacy is particularly important in personalized services such as recommender systems where data regards users' preferences. For these reasons, in the last few years, many privacy-preserving recommender systems have been proposed. The first attempts were mostly based on data perturbation [2]. Instead of sharing their private data in clear, users first add some noise to make it "safe" to share. Unfortunately, this methodology has an evident drawback: the spurious data negatively affects the recommendations' quality. However, this drawback can be addressed using cryptography, specifically homomorphic encryption [3]–[6]. Homomorphic encryption (HE) allows computation on encrypted data so that, when decrypted, matches the result of the operations as if they had been performed on the plain data. However, HE is computationally expensive, especially for "limited" devices such as smartphones.

More recently, decentralized approaches, such as secure multi-party protocols [7]–[9], seem to have attracted researchers attention. In particular, the Federated Learning (FL) scheme proposed by Google [10], [11] has opened new and exciting possibilities. The main idea behind FL is that the training process is computed across multiple decentralized parties holding private data that do not need to be shared. Users share parameters/gradients with a central server that updates and share a global model without having access to the users' private data. FL is not a new line of research in the Recommender Systems community; however, all the proposed approaches are based on matrix factorization [3], [12]–[14].

In this paper, we propose a Federated Variational Autoencoder (FedVAE) for top-N item recommendation. FedVAE uses the same VAE architecture proposed in [15] and trains it using the FL framework. A peculiar characteristic of a Federated RS is the client's side dataset, which consists of a single example, that is the user's ratings themselves. This fact can cause a slow learning pace, especially in the first rounds. For this reason, we propose an adaptive learning rate schedule (FedVAE$_{\text{dec}}$), showing that the simple increase in the learning rate is not enough. We show that the proposed schedule enhances the efficiency and also the stability of the training. We also discuss how FedVAE deals with security and privacy issues. Security against malicious clients is guaranteed thanks to a filtering approach able to detect byzantine clients. Moreover, differential privacy is also implemented "for free" thanks to the VAE architecture that contains a dropout layer. The dropout acts like input perturbations on the input data, which is an effective way to perform differential privacy. An extensive experimental evaluation on five benchmark datasets shows that FedVAE achieves performance similar to MultVAE. The experimental results also underline that FedVAE$_{\text{dec}}$ increases both the effectiveness and the efficiency of FedVAE in terms of the number of rounds to reach high accuracy values.

The remainder of the paper is structured as follows. Section II discusses the related work, while Section III presents the background knowledge useful to grasp the paper's content. Section IV describes the main contributions of the paper. Section V shows the carried out experimental evaluation. Finally, Section VI provides possible future research directions.

[1] https://gdpr-info.eu/

II. Related work

In this section, we focus on recent approaches for privacy-preserving collaborative filtering. For a more in-depth analysis of privacy risks and standard methods, we refer the reader to [16].

Historically, the most used techniques for building privacy-preserving CF was based on cryptographic approach, random perturbations, and differential privacy. Cryptographic methods [3]–[5], [17], [18], are usually based on homomorphic encryption, which is effective but expensive in terms of computational cost. On the other hand, techniques based on random perturbations [2], [7], [8], [19] are cheap, but the perturbations can harm the quality of the recommendations. However, they can guarantee good differential-privacy.

In CF contexts, also the use of Federated Learning is not new. In [20], the Federated scheme is applied to the popular WRMF [21] recommender for top-N item recommendation. The method, dubbed FCF, guarantees privacy by design, and the authors also showed that FCF achieves similar performance w.r.t. WRMF. However, the authors do not provide any insight about possible attacks like reconstruction attak [22] or membership inference attack [23]. In [3], Chai et al. showed that a federated approach for performing matrix factorization could leak users' information. To address this problem, the authors enhanced the method (called FedMF) with the introduction of homomorphic encryption. A closely related method is the one by Jalalirad et al. [12] in which they proposed yet another federated MF approach for rating prediction.

Ribero et al. [24] proposed yet another federated matrix factorization for rating prediction. The method uses a shared item representation approximately computed over the client's prototypes. These prototypes are designed to contain similar information as representing the actual item but in a lower-dimensional space to minimize the communication load.

Recently, FedRec [25] has been proposed. It implements a classic matrix factorization for rating prediction [26] in a federated setting. FedRec protects the user's privacy through the addition of noise to the user's ratings. Specifically, FedRec samples negative feedback and associates to them a fake random rating.

Chen et al [13] proposed a federated version of the famous FISM algorithm [27] by Karypis et al. The optimization is performed using an Adam procedure [28] similarly as it will be shown in this paper. Authors also show that an Adam-based federated recommender is susceptible to attacks from byzantine clients that can camouflage their gradients to launch a poisoning attack. For this reason, they propose an Adam-based robust federated recommendation system, called A-RFRS, in which byzantine clients are detected by checking their gradients.

In [14], the authors consider a slightly different federated setting. Users' ratings are spread over small datasets, and two servers are responsible for applying the secure multi-party computation. The proposed protocol also establishes more security layers through the use of garbled circuits [29] and secret sharing.

TABLE I: Notations table.

Variable	Definition
$\mathcal{U}$	Set of users
$\mathcal{I}$	Set of items
$\mathcal{U}_i$	Set of users who rate the item $i \in \mathcal{I}$
$\mathcal{I}_u$	Set of items rated by $u \in \mathcal{U}$
n	Number of users, $\lvert\mathcal{U}\rvert = n$
m	Number of items, $\lvert\mathcal{I}\rvert = m$
$\mathcal{R}$	Ratings set
$\mathbf{R}$	Binary rating matrix s.t. $r_{ui} = 1 \iff (u,i) \in \mathcal{R}$
$\mathbf{r}_u$	User u (column) rating vector
r_{ui}	Rating matrix entry
$\boldsymbol{\phi}$	Encoder's parameters
$\boldsymbol{\theta}$	Decoder's parameters
$\boldsymbol{\omega}$	VAE's parameters
$\mathbf{z}_u$	Latent user's representation
T	Number of federated rounds
R	Number of users per round
$\mathbb{I}[x]$	Indicator function s.t. $\mathbb{I}[x] = 1 \iff x$ is true
$\mathbf{I}_k$	Identity matrix in $\mathbb{R}^{k \times k}$
$\lVert\mathbf{x}\rVert_p$	p-norm of $\mathbf{x}$, i.e., $\sqrt[p]{\sum_i \mathbf{x}_i^p}$
$\langle \cdot, \cdot \rangle$	Dot-product

All the just mentioned works are based on matrix factorization techniques, while in this paper, we propose a Federated recommender system based on a state-of-the-art neural network-based model (i.e., [15]). There are some analogies between our work and [13]. We will be discussed these analogies later in the paper, although the overall approach is very different. Moreover, in this paper, we focus on top-N item recommendation, as in [20], while all the other works deal with rating prediction. Accuracy-wise we will not compare our solution with FCF [20] since both FedVAE and FCF reflect the performance of their corresponding "batch" versions, namely MultVAE and WRMF. MultVAE has shown of being superior to WRMF [15].

III. Background

This section briefly goes through the background material useful to understand the paper fully. To ease the reading, in the following, we provide a notations table (Table I), which summarizes the most used notation throughout the paper.

A. Variational Autoencoder for CF

In this paper, we focus on the top-N recommendation. We assume of knowing the users' implicit preferences, that is, ratings are binary ($\mathbf{r}_u \in \{0, 1\}^m$). The recommendation task consists of, given a user u, computing a ranking over the items where the relevant items for u should be put in the top.

Mult-VAE [15] is a state-of-the-art CF for top-N item recommendation. It is based on the VAE framework, and its two main peculiarities are: (i) it assumes a multinomial prior on the input instead of the classic Gaussian prior used in standard VAEs; (ii) it employs the β-VAE [30] architecture in which a hyper-parameter β acts as a trade-off parameter between the reconstruction loss and the Kullback-Leibler (*KL*) loss.

As just said, the basic idea of Mult-VAE is similar to VAE but with the multinomial distribution as the likelihood function instead of Gaussian distributions usually used in VAE. For each user u, the model starts by sampling a k-dimensional latent representation $\mathbf{z}_u$. Then, the model transforms it via a non-linear function $f_{\boldsymbol{\theta}}(\cdot) : \mathbb{R}^k \to \mathbb{R}^m$ to produce a probability distribution over the m items $\pi(\mathbf{z}_u)$ from which the user rating history $\mathbf{r}_u$ is assumed to have been drawn:

$$\begin{aligned} \mathbf{z}_u \sim \mathcal{N}\left(0, \mathbf{I}_k\right), \quad \pi\left(\mathbf{z}_u\right) \propto \exp\left\{f_\theta\left(\mathbf{z}_u\right)\right\} \\ \mathbf{r}_u \sim \operatorname{Mult}\left(\|\mathbf{r}_u\|_1, \pi\left(\mathbf{z}_u\right)\right). \end{aligned} \tag{1}$$

The non-linear function f is parametrized by a deep neural network with parameters $\boldsymbol{\theta}$, while $\pi(\mathbf{z}_u) \in \mathbb{R}^m$ is the softmax function applied to the transformed latent representation.

The log-likelihood for a user u conditioned on the latent representation $\mathbf{z}_u$ is:

$$\log p_{\boldsymbol{\theta}}(\mathbf{r}_u|\mathbf{z}_u) = \sum_{i=1}^{m} \mathbf{r}_{ui} \log \pi_i(\mathbf{z}_u) \tag{2}$$

To estimate the model parameters $\boldsymbol{\theta}$, for each data point $\mathbf{r}_u$, one needs to approximate the intractable posterior distribution $p(\mathbf{z}_u|\mathbf{r}_u)$. Likewise standard VAE, the true intractable posterior is approximated with simpler variational distribution $q(\mathbf{z}_u)$ that is set to be a fully factorized (diagonal) Gaussian distribution, $q(\mathbf{z}_u) = \mathcal{N}(\boldsymbol{\mu}_u, \langle \mathbf{I}_k, \boldsymbol{\sigma_u}^2\rangle)$.

The objective of variational inference is then to optimize the free variational parameters $\{\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u^2\}$ so that the *Kullback-Leibler* divergence $\mathrm{KL}(q(\mathbf{z}_u)||p(\mathbf{z}_u|\mathbf{r}_u))$ is minimized. However, the number of parameters to optimize grows with the number of users and items in the data set. In order to solve this problem, the VAE replaces the individual variational parameters with a data-dependent function $g_{\boldsymbol{\phi}}\left(\mathbf{r}_u\right) = \left[\boldsymbol{\mu}_{\boldsymbol{\phi}}\left(\mathbf{r}_u\right), \boldsymbol{\sigma}_{\boldsymbol{\phi}}\left(\mathbf{r}_u\right)\right] \in \mathbb{R}^{2k}$ parametrized by $\boldsymbol{\phi}$ and where both $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{r}_u)$ and $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{r}_u)$ k-dimensional vectors. Thus, the variational distribution is set to be $q_{\boldsymbol{\phi}}(\mathbf{z}_u|\mathbf{r}_u) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{r}_u), \langle \mathbf{I}_k, \boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{r}_u)\rangle)$. Putting together $q_\phi(\mathbf{z}_u|\mathbf{r}_u)$ and $p_\theta(\mathbf{r}_u|\mathbf{z}_u)$ we end up with a an autoencoder-like architecture.

Finally, the model's parameters are estimated by constructing an evidence lower bound (ELBO). The resulting Mult-VAE ELBO [15] is defined as:

$$\begin{aligned} &\mathcal{L}_{\text{Mult-VAE}}\left(\boldsymbol{\phi}, \boldsymbol{\theta}; \mathbf{r}_u\right) = \\ &\mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{r}_u)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{r}_u|\mathbf{z}_u\right) - \beta \cdot \mathrm{KL}\left(q_{\boldsymbol{\phi}}\left(\mathbf{z}_u|\mathbf{r}_u\right) \| p_{\boldsymbol{\theta}}\left(\mathbf{z}_u\right)\right)\right], \end{aligned} \tag{3}$$

where $\beta \geq 0$ is a trade-off hyper-parameter between the reconstruction loss and the closeness of the approximate posterior w.r.t. the Gaussian prior.

It is possible to obtain an unbiased estimate of the ELBO by sampling $\mathbf{z}_u \sim q_{\boldsymbol{\phi}}$ and performing stochastic gradient ascent to optimize it. However, the sampling operation has not derivative. Thanks to the *reparameterization trick* [31] we can sidestep this issue. Once the model is trained, the recommendation can be computed as follows: (i) the user rating history $\mathbf{r}_u$ is fed to the encoder network, which outputs the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{r}_u)$ and the standard deviation $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{r}_u)$ of the variational distribution; (ii) $\mathbf{z}_u = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{r}_u)$ is fed into the decoder network; (iii) the decoder outputs the predicted the distribution $f_{\boldsymbol{\theta}}(\mathbf{z}_u)$ that is eventually sorted.

B. Federated Learning

Federated learning (FL) [10], [32], [33] (also known as collaborative learning) is a machine learning technique in which the training process is computed across multiple decentralized parties (e.g., devices or servers) holding private local data samples that are never exchanged with other parties. The general idea consists of training local models on the local private data and then sharing the parameters (e.g., the weights of a neural network) or the computed gradients to generate a global model (usually on a centralized server). Thus, the learning is decentralized for "guaranteeing" privacy, but the global model stays on the server with only a view of the model and not the data used during training. Some federated models work entirely peer-to-peer, but we focus on a server-based federated scheme in this work. It is important to underline that Federated learning differs from distributed learning in terms of the assumptions about the data. Distributed machine learning aims at parallelizing the computation, but the data are shared between the different parties (and they usually used the same data/same distribution). Federated learning instead aims at training on heterogeneous data sets. In FL, the training happens in rounds. At each round t: (i) the server shares the model with all the participants (i.e., clients); (ii) each client computes the gradient w.r.t. her data, and send back this information to the server; (iii) the server collects and aggregates all the participants' contribution and updates its model. In the context of FL for recommendations, the data owned by a user is her ratings.

IV. Federated VAE

In this section, we describe our Federated Variational Autoencoder for collaborative filtering (FedVAE). We first introduce the Federated learning algorithm in general (Section IV-A), and afterward (Section IV-B), we introduce our learning rate schedule (FedVAE$_{\text{dec}}$) to improve the efficiency and effectiveness of the learning process. Finally, we discuss the privacy and security capabilities of FedVAE (Section IV-C).

A. FedVAE algorithm

FedVAE aims at computing a Mult-VAE [15] model using the Federated Learning (FL) framework. In the FL paradigm, users ($\mathcal{U}$) are the data owners who wish to train a model by consolidating their respective data, i.e., ratings ($\mathbf{r}_u$). The learning process is carried out collaboratively by the users without exposing their private data. In the client-server setting, FL also considers an additional actor: a server $\mathcal{S}$, which holds the updated global model, but it does not have direct access to any of the data owners, i.e., users. The challenge is to maintain an up to date model by keeping all the user ratings private.

The Mult-VAE model is a (deep) neural network in which the training is performed using the Adam optimization algorithm. Adam [28] is a Stochastic Gradient Descent (SGD) based algorithm, and thus it is suitable for performing distributed machine learning. At the beginning of the training

Algorithm 1: FedVAE

```
Input: $R$: number of users per round, $T$: number of rounds, $\eta$: learning rate, $\beta_1$: Adam's first moment weight, $\beta_2$: Adam's second moment weight, $\epsilon$: Adam's correction value
Output: $(\boldsymbol{\phi}_T, \boldsymbol{\theta}_T)$: trained parameters of the global model
initialization
    $\boldsymbol{\omega}_0 = (\boldsymbol{\phi}_0, \boldsymbol{\theta}_0) \leftarrow$ random initialized from $\mathcal{N}(0, 1)$
end
for $t \in [1, \dots, T]$ do                                  ▷ Federated rounds
    $\mathcal{U}_t \leftarrow$ randomly pick $R$ users from $\mathcal{U}$
    for $u \in \mathcal{U}_t$ in parallel do
        Send: $\boldsymbol{\omega}_{t-1}$ to $u$
        Receive: $g_{\boldsymbol{\omega}_u} \leftarrow$ LocalGradient($\mathbf{r}_u$)    ▷ Client
    end
    $\mathbf{G}_t \leftarrow \{g_{\boldsymbol{\omega}_u}\}_{u \in \mathcal{U}_t}$                  ▷ Set of gradient updates
    $\boldsymbol{\omega}_t \leftarrow$ GlobalUpdate($\boldsymbol{\omega}_{t-1}$, $\mathbf{G}_t$, $\eta$, $\beta_1$, $\beta_2$, $\epsilon$)
end
return $\boldsymbol{\omega}_T = (\boldsymbol{\phi}_T, \boldsymbol{\theta}_T)$
```

phase, the server $\mathcal{S}$ randomly initializes the model's parameters. Then, the learning process proceeds in rounds, where at each round t:

1) the server $\mathcal{S}$ shares a copy of the model parameters, i.e., $\boldsymbol{\omega} := (\boldsymbol{\phi}, \boldsymbol{\theta})$, to a subset of users $\mathcal{U}_t \subseteq \mathcal{U}$;
2) (in parallel) each user u compute the gradient update (on the Mult-VAE loss, Eq. (3)) on its own copy of the model and sends it back to the server;
3) the server collects and aggregates (e.g., average) all the users' contributions;
4) the server updates the model according to the computed aggregated gradient.

In Algorithm 1, the just mentioned procedure is described in detail. The LocalGradient($\mathbf{r}_u$) procedure is computed (in parallel) client's side and it returns gradient of the loss (3) w.r.t. the user's ratings. Users can perform one or more backward pass on the input to compute the gradient. Experimentally, we found that a single iteration is enough to guarantee a decent convergence rate of FedVAE. From a practical perspective, this is good because it reduces the client's computational burden.

The GlobalUpdate function is defined in Algorithm 2, and it follows the Adam optimization steps with the addition of a users' filtering function (SecureFilter). The variables $\mathbf{m}_t$ and $\mathbf{v}_t$ are the gradient's first and the second momentum estimate, respectively. Both are initialized to 0 at the beginning of the training process. The function SecureFilter implements the gradient-based Krum filtering [34] useful to avoid poisoning attacks by malicious users. A more in-depth discussion about both the security and privacy capabilities of FedVAE is present in Section IV-C.

Algorithm 2: GlobalUpdate

```
Input: $\boldsymbol{\omega}_{t-1}$: current model's parameters, $\mathbf{G}_t$: set of gradient updates, $\eta$: learning rate, $\beta_1$: Adam's first moment weight, $\beta_2$: Adam's second moment weight, $\epsilon$: Adam's correction value
Output: $\boldsymbol{\omega}^t$: updated model's parameters
1 $\mathbf{G}_t \leftarrow$ SecureFilter($\mathbf{G}_t$)      ▷ Filter byzantine clients
2 $g_{\boldsymbol{\omega}_t} \leftarrow |\mathbf{G}_t|^{-1} \sum_{g_{\boldsymbol{\omega}_u} \in \mathbf{G}_t} g_{\boldsymbol{\omega}_u}$      ▷ Average gradient step
3 $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_{\boldsymbol{\omega}_t}$
4 $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}^2_{\boldsymbol{\omega}_t}$
5 $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$
6 $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$
7 $\boldsymbol{\omega}_t \leftarrow \boldsymbol{\omega}_{t-1} - \eta \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + \epsilon)$
8 return $\boldsymbol{\omega}_t = (\boldsymbol{\phi}_t, \boldsymbol{\theta}_t)$
```

B. FedVAE with learning rate decay

In each round of FedVAE, only a relatively small subset of users participates in the gradient step's computation causing a slow learning pace. To mitigate this issue, we can increase the learning rate "hoping" that bigger gradient steps can help the training. As we will see in the experimental section (Section V), this is sub-optimal, and it generally harms the training as the number of rounds increases. We propose a decaying learning rate schedule to make the learning stable and efficient. In particular, we modify Algorithm 2 by adding the following update just before the return

$$\boldsymbol{\omega}_t \leftarrow \boldsymbol{\omega}_t + \gamma_t (\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}) \tag{4}$$

where γ_t is defined as $\gamma_t = \gamma \delta^t, \quad \gamma \geq 0, \delta \in [0, 1]$.

We call γ the learning rate multiplier and δ the decay factor. Now we show that the parameters update in Equation (4) is equivalent to changing the learning rate according to the formula $\eta_t = \eta(1 + \gamma \delta^t)$. Let us recall the weights update of the Adam optimization procedure (Line 5 in Algorithm 2):

$$\boldsymbol{\omega}_t \leftarrow \boldsymbol{\omega}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}. \tag{5}$$

Then, by substituting (5) in (4) we obtain

$$\begin{aligned}
\boldsymbol{\omega}_t &\leftarrow \boldsymbol{\omega}_t + \gamma_t (\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t-1}) \\
&= \boldsymbol{\omega}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} + \gamma_t \left(\boldsymbol{\omega}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} - \boldsymbol{\omega}_{t-1} \right) \\
&= \boldsymbol{\omega}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} - \gamma_t \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \\
&= \boldsymbol{\omega}_{t-1} - \underbrace{(1 + \gamma_t)\eta}_{\eta_t} \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}.
\end{aligned}$$

When $\delta = 1$, no decay is applied, and it is equivalent to fix the learning rate to $\eta(1 + \gamma)$ from the beginning of the training. Similarly, if $\delta = 0$, then it is equal to use the initial learning rate η throughout all the training (i.e., Algorithm 1). With $0 < \delta < 1$, then given $t \to \infty$ the learning rate will converge to η. In practice, even with low decay values, such as $\delta = 0.9$, reaching learning rate values close to η requires less than an hundred of epochs.

C. Security and privacy of FedVAE

It is well known that the Federated approach helps but does not guarantee privacy [35]. It has been demonstrated that an attacker (potentially the server itself) through the gradient information can partially reconstruct the input data (i.e., the user's rating). An off-the-shelf solution may be to employ a Secure Aggregation approach [36], [37]. Secure Aggregation would allow the clients to share their gradient updates securely, even in the presence of an honest-but-curious server. The communication happens on secure channels, and the encrypted gradients are shared securely. The server can only access the decrypted version of the aggregated gradient, while it has no way to decrypt the single clients' contributions. The main practical problem of such an approach is that it requires a specific set of clients to participate in a particular round. Some methods try to mitigate this requirement [36], [37], but the applicability of such protocol is not that obvious. Moreover, it is computationally more expensive, especially from the clients' perspective. However, by design, the FedVAE model is based on Mult-VAE which applies a dropout layer [38] to the input for regularization purposes. Formally, given a user u and a dropout rate $p \in [0, 1]$, the dropout operation is defined as $\forall i \in \mathcal{I}$ s.t. $r_{ui} = 1$ then $\hat{r}_{ui} = b\mathbf{r}_{ui}$, where $b \sim \text{Ber}(1-p)$. In other words, the (positive) rating of the user u for the item i is put to zero with probability p.

Computing the dropout can be considered an operation that performs input perturbation [39]. Input perturbation is a well-known technique to achieve differential privacy [40]. In particular, it has been shown that adding noise to the input data leads to perturbation on the gradient. Applying Gaussian noise to gradients of an SGD-based optimization algorithm is demonstrated to be (ϵ, δ)-differential private for some $\epsilon, \delta > 0$.

The dropout-based perturbation of FedVAE can be approximated by a clipped Gaussian noise on the positive ratings, that is, for $r_{ui} = 1$, $\hat{r}_{ui} = br_{ui} \approx \min(\max(0, r_{ui} + \xi), 1)$, where $b \sim \text{Ber}(1-p), \xi \sim \mathcal{N}(-2p, 1)$. Even though this approximation does not allow to directly exploit the results in [40], it gives some hints about the potential privacy-preserving capabilities of FedVAE. However, providing a formal analysis of the differential-privacy of FedVAE is out of the scope of this paper, and we reserve this for future work.

Regarding the robustness against poisoning attacks (i.e., malicious users that inject fake ratings), FedVAE can rely on the detection mechanism proposed in [13]. Since the server receives the client's gradient updates, we can detect byzantine clients using the gradient-based Krum [34] filter. In Algorithm 2, we refer to this filtering with the function SecureFilter. The proof of the robustness of this defensive measure is reported in [34] and [13]. The gradient-based Krum filtering, in conjunction with the in-design input perturbation, makes FedVAE robust and secure without adding any particular computational burden. These characteristics allow FedVAE to be practically usable both in terms of recommendation quality and security/privacy.

V. Experiments

In this section, we describe the performed experiments to assess the efficiency and effectiveness of the proposed federated learning approach. The source code of the experiments is available as Google Colaboratory notebook [2].

A. Datasets and setting

We compared the different approaches on five benchmark datasets that are described in the following:

- `ml1m`: The MovieLens 1M[3] data set contains user-movie ratings collected from a movie recommendation service. We binarize the explicit ratings by keeping ≥ 4 stars ratings. Users with less than five ratings are removed.
- `citeulike-a`: This dataset[4] was collected from CiteULike and Google Scholar. The dataset contains articles associated with users and those are considered as positive ratings. Users with less than two ratings are removed.
- `steam`: This dataset[5] contains a list of user behaviors provided by the Steam PC Gaming platform. The behaviors included are 'purchase' and 'play', but we keep, for each user, a single behaviour with a specific item disregarding the behaviour's type. The kept behaviours are treated as implicit feedback. Users with less than two feedbacks are removed.
- `lastFM`: This dataset[6] contains social networking, tagging, and music artist listening information taken from Last.fm online music system. Users with less than two feedbacks are removed.
- `filmtrust`: FilmTrust[7] is a dataset crawled from the FilmTrust website 2011. Ratings are in a 1-5 scale, and no thresholding is applied. Users with less than two ratings are removed.

Table II reports the details about the datasets. In the experiments, we held out a set of users ($\mathcal{U}_{\text{test}} \subset \mathcal{U}$) to test/validate the models, and for each test user $u \in \mathcal{U}_{\text{test}}$ 80% of her ratings are used as input, i.e., the known u's ratings, and the remaining 20% as testing ratings.

To reduce the number of variables between experiments, we fixed the variational autoencoder architecture to the one reported in [15]. The Mult-VAE neural network architecture is symmetric, with the encoder network having the following structure $[|\mathcal{I}| \Rightarrow 600 \Rightarrow 200]$, where 200 is the size of the latent space. In each hidden layer the activation function is a *tanh*. We also fixed $\beta = 0.2$, and the learning rate $\eta = 0.001$ that are the best performing ones in [15]. Differently from [15], β is not annealed but kept constant during all the training. Moreover, the dropout rate is fixed to 0.5. For the optimization,

[2] https://tinyurl.com/155zn06i

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

[4] https://github.com/js05212/citeulike-a

[5] https://www.kaggle.com/tamber/steam-video-games

[6] https://grouplens.org/datasets/hetrec-2011/

[7] https://guoguibing.github.io/librec/datasets.html

TABLE II: Data sets information after the pre-processing: number of users, number of items, number of ratings, number of users in the test set, and the size of each batch in each federated round. Numbers inside the parentheses indicate the percentage with respect to the total number of users.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{U}_{\text{test}}\rvert$	R
`ml1m`	6,034	3,520	575,259	400(6.6%)	250(4%)
`citeulike-a`	5,551	16,950	210,495	800(14%)	250(4.5%)
`steam`	6,693	4,943	122,893	10^3(14%)	250(3.5%)
`lastFM`	1,884	16,519	91,694	200(10.6%)	150(7.9%)
`filmtrust`	1,400	1,939	35,252	200(14%)	150(10.7%)

we used the standard values for the Adam's hyper-parameters, that is, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

We compare results of the following variants:

- **Mult-VAE** [15]: that is the non-federated baseline upon which our proposal is based on.
- **FedVAE**: this is the standard FedVAE variant (Section IV-A).
- **FedVAE$_{\text{dec}}$**: this is the FedVAE variant with the decaying learning rate. In all the experiments concerning the decay factor we fixed $\delta = 0.9$.

The learning rate schedule is defined w.r.t. the value of the unit t, that in Algorithm 1 is defined as a round. In our experiments, to keep consistencies between rounds and epochs, we consider the unit t as an epoch. In the federated setting, an epoch is defined as a sequence of M rounds such that $MR \geq |\mathcal{U} \setminus \mathcal{U}_{\text{test}}| > (M-1)R$, where R is the number of users per round (see Table II). A user can contribute more than once, and others may not contribute at all. Thus, E epochs correspond to EM rounds. We report the results achieved with the factor γ in the set $\{1, 3, 5\}$. Higher values have shown to be not effective.

It is worth to notice that both FedVAE and Mult-VAE have not been fine-tuned for these experiments. Most of the hyper-parameters have been set to the best values reported in [15]. These experiments do not aim to make a thorough comparison between the batch version (Mult-VAE) with the Federated version (FedVAE). FL theory already showed that with $t \to \infty$, the Federated model converges to the same solution as the non-Federated version. The purpose of the reported experiments is to show that FedVAE is effective/efficient and the benefit of using the proposed adaptive learning rate.

To measure the recommendation quality we used two widely used ranking-based metrics:

- **nDCG@k** measures the quality of the prediction based on the position of an item in the recommended list. In particular, it uses a monotonically increasing discount to emphasize the importance of higher ranks versus lower ones.
- **recall@k**, differently from the nDCG, considers all items ranked within the first k to be equally important.

In the experiments we fix the value $k = 100$ for all dataset but `filmtrust` for which $k = 20$. This difference is due to the size of the item set, which is rather small compared to the other considered datasets.

B. *Results and discussion*

We compared FedVAE and FedVAE$_{\text{dec}}$ in terms of the top-N accuracy as the number of federated rounds increases. The results are reported in Figure 1. In the plots, we report FedVAE with different γ values, which means changing the learning rate of FedVAE according to the rule described in Section IV-B. Regarding FedVAE$_{\text{dec}}$, the decay value is fixed to $\delta = 0.9$.

First of all, we can observe that FedVAE (black curve) in all datasets but `filmtrust` in 100 epochs converges to the same nDCG and recall as Mult-VAE. We argue that the training issue on `filmtrust` is due to the few ratings (and users/items). Maybe with a smaller network, e.g., fewer neurons in the hidden layer, the convergence could have been faster and more stable. Regarding the comparison between FedVAE and FedVAE$_{\text{dec}}$, we highlight the following interesting behaviors:

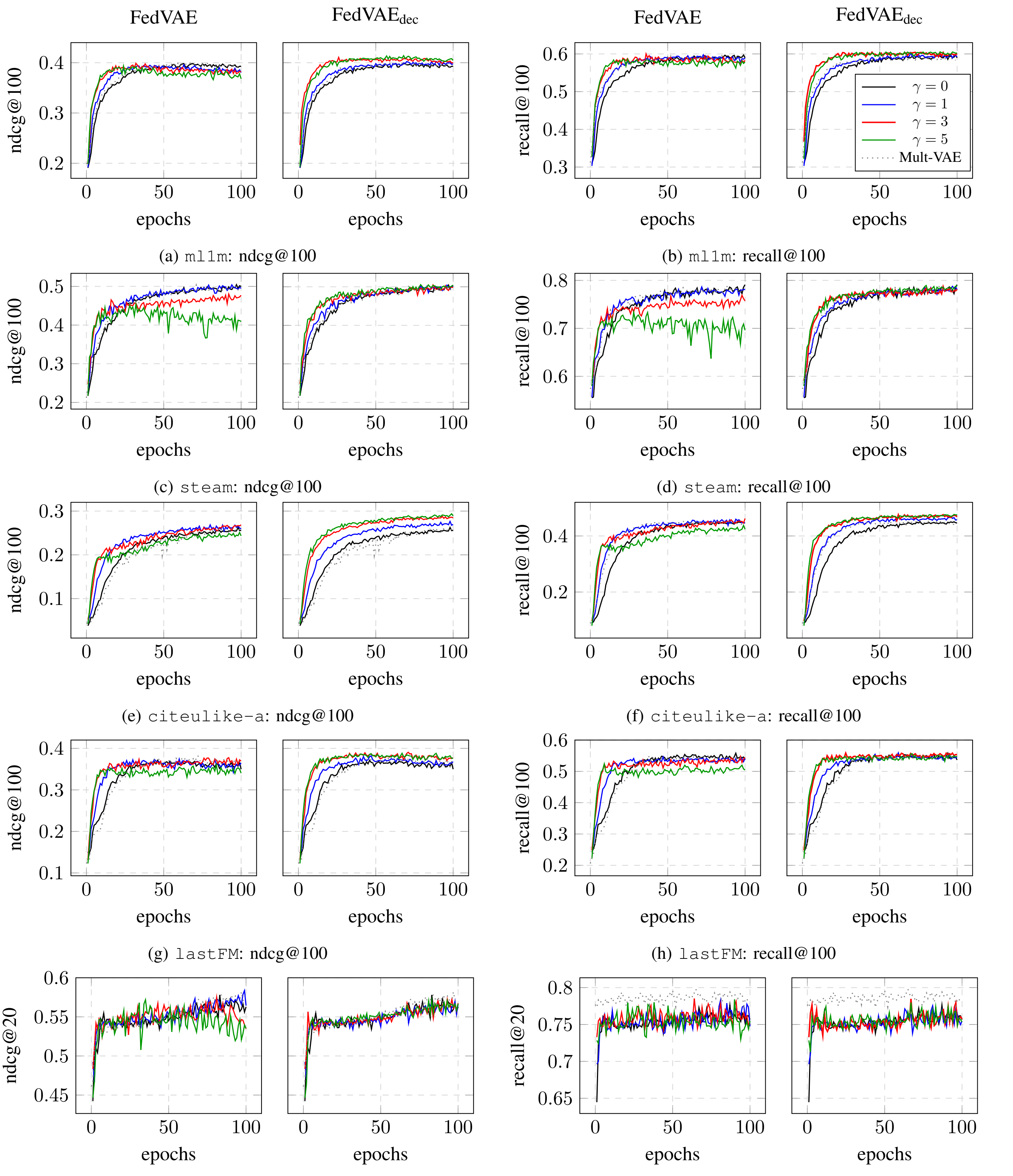
- FedVAE$_{\text{dec}}$ consistently achieves better or comparable performance in less epochs/rounds. This is particularly evident on `ml1m`, `citeulike-a` and `lastFM`. In this experimental setting, FedVAE$_{\text{dec}}$ is also able to achieve better performance than Mult-VAE. However, we have to stress that neither method has been fine-tuned.
- The simple increase of the learning rate (via γ) brings benefit in the first rounds, while it deteriorates the performance in the long run. Moreover, learning is highly unstable. The introduction of the decay schedule (FedVAE$_{\text{dec}}$) allows retaining the first rounds' benefit while keeping stability through the training rounds. This can be easily seen on `steam`, `lastFM` and `filmtrust`.
- The number of users per round (4% up to $\sim$ 11%) does not seem to affect FedVAE. We do not try higher values because we wanted to test reasonable scenarios where only a small subset of users participate in each round.

It is also worth to notice that the gain in performance of FedVAE$_{\text{dec}}$ is more noticeable in terms of nDCG. This means that FedVAE$_{\text{dec}}$ helps rearrange the top part of the rankings, which is a desirable feature. In general, a γ value of 5 has shown to achieve consistent performance throughout the datasets. Higher values (not reported here) have shown highly inconsistent results. From the results, we can observe that γ represents a trade-off between stability and convergence speed: values close to zero favor the stability of the learning, while higher values (close to 5) favor the convergence speed.

VI. Conclusions and future work

We proposed a new Federated Learning approach for top-N item recommendation, dubbed FedVAE. FedVAE is a simple yet effective and efficient adaptation of the state-of-the-art Mult-VAE model. Moreover, FedVAE implements effective measures to cope with potential privacy and security issues. We also propose an adaptive learning rate schedule that has empirically shown to improve the efficiency and stability of the FedVAE training.

In the future, we plan to theoretically study the differential privacy capabilities of FedVAE. In the paper, we argue that



(a) `ml1m`: ndcg@100

(b) `ml1m`: recall@100

(c) `steam`: ndcg@100

(d) `steam`: recall@100

(e) `citeulike-a`: ndcg@100

(f) `citeulike-a`: recall@100

(g) `lastFM`: ndcg@100

(h) `lastFM`: recall@100

(i) `filmtrust`: ndcg@20

(j) `filmtrust`: recall@20

Fig. 1: For each dataset: (left) FedVAE vs. (right) FedVAE$_{\text{dec}}$ using different values of γ. nDCG and recall metrics are reported. Side-by-side plots have the same scale (the one on the left side). In all the plots, we reported also the performance of Mult-VAE (gray dotted curve). In FedVAE$_{\text{dec}}$, $\delta = 0.9$.

the dropout to the input layer can ensure good differential privacy. However, this must be confirmed through theoretical analysis. Further analysis needs to be done on the effect of using an alternative schedule, such as different decays. It will also be interesting to analyze whether adding a further layer of perturbations, e.g., directly at the gradient's level, can improve privacy without harming the method's accuracy. We additionally aim to search for efficient and effective ways to perform hyper-parameters tuning (especially β and γ) in the Federated setting. This would hugely improve the practical applicability of FedVAE. The easiest approach could be to use a similar annealing schedule, for β, used in [15] at the server level.

References

[1] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, pp. 50–57, Oct 2017.

[2] N. Polatidis, C. K. Georgiadis, E. Pimenidis, and H. Mouratidis, "Privacy-preserving collaborative recommendations based on random perturbations," *Expert Syst. Appl.*, vol. 71, pp. 18–25, Apr. 2017.

[3] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *ArXiv*, vol. abs/1906.05108, 2019.

[4] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, "Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pp. 617–628, 2016.

[5] J. Wang, Q. Tang, A. Arriaga, and P. Y. A. Ryan, "Novel collaborative filtering recommender friendly to privacy protection," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4809–4815, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[6] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Comput. Surv.*, vol. 51, July 2018.

[7] E. Shmueli and T. Tassa, "Secure multi-party protocols for item-based collaborative filtering," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pp. 89–97, 2017.

[8] E. Shmueli and T. Tassa, "Mediated secure multi-party protocols for collaborative filtering," *ACM Trans. Intell. Syst. Technol.*, vol. 11, Feb. 2020.

[9] H. Kaur, N. Kumar, and M. S. Obaidat, "Multi-party secure collaborative filtering for recommendation generation," *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[10] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and A. y. B. Arcas, "Communication-efficient learning of deep networks from decentralized data," *AISTATS*, pp. 1273–1282, 2017.

[12] A. Jalalirad, M. Scavuzzo, C. Capota, and M. Sprague, "A simple and efficient federated recommender system," in *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, BDCAT '19, pp. 53–58, 2019.

[13] C. Chen, J. Zhang, A. K. H. Tung, M. Kankanhalli, and G. Chen, "Robust federated recommendation system," 2020.

[14] L. Wang, Z. Huang, Q. Pei, and S. Wang, "Federated cf: Privacy-preserving collaborative filtering cross multiple datasets," in *2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[15] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pp. 689–698, 2018.

[16] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2nd ed., 2015.

[17] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[18] S. Badsha, X. Yi, and I. Khalil, "A practical privacy-preserving recommender system," *Data Science and Engineering*, vol. 1, pp. 161–177, 2016.

[19] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky, "Applying differential privacy to matrix factorization," in *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pp. 107–114, 2015.

[20] M. A. ud din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," 2019.

[21] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*, pp. 263–272, 2008.

[22] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 2512–2520, 2019.

[23] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017.

[24] M. Ribero, J. Henderson, S. Williamson, and H. Vikalo, "Federating recommendations using differentially private prototypes," 2020.

[25] G. Lin, F. Liang, W. Pan, and Z. Ming, "Fedrec: Federated recommendation with explicit feedback," *IEEE Intelligent Systems*, pp. 1–8, 2020.

[26] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, (Red Hook, NY, USA), pp. 1257–1264, Curran Associates Inc., 2007.

[27] S. Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pp. 659–667, 2013.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[29] A. C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science*, pp. 162–167, 1986.

[30] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[32] K. A. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konecný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *SysML 2019*, 2019.

[33] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[34] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NIPS*, 2017.

[35] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, vol. 34, no. 4, pp. 242–248, 2020.

[36] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[37] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 1175–1191, 2017.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[39] P. Jain, V. Kulkarni, A. Thakurta, and O. Williams, "To drop or not to drop: Robustness, consistency and differential privacy properties of dropout," *ArXiv*, vol. abs/1503.02031, 2015.

[40] Y. Kang, Y. Liu, B. Niu, X.-Y. Tong, L. Zhang, and W. Wang, "Input perturbation: A new paradigm between central and local differential privacy," *ArXiv*, vol. abs/2002.08570, 2020.