Data-free Knowledge Distillation for Reusing Recommendation Models

Cheng Wang*
Huazhong University of Science and Technology
Wuhan, China
wangcheng20@hust.edu.cn

Jiacheng Sun
Huawei Noah's Ark Lab
Shenzhen, China
xysunjiacheng@gmail.com

Zhenhua Dong
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

Jieming Zhu
Huawei Noah's Ark Lab
Shenzhen, China
jiemingzhu@ieee.org

Zhenguo Li
Huawei Noah's Ark Lab
Shenzhen, China
Li.Zhenguo@huawei.com

Ruixuan Li†
Huazhong University of Science and Technology
Wuhan, China
rxli@hust.edu.cn

Rui Zhang†
ruizhang.info
Shenzhen, China
rayteam@yeah.com

ABSTRACT

A common practice to keep the freshness of an offline Recommender System (RS) is to train models that fit the user's most recent behaviours while directly replacing the outdated historical model. However, many feature engineering and computing resources are used to train these historical models, but they are underutilized in the downstream RS model training. In this paper, to turn these historical models into treasures, we introduce a model inversed data synthesis framework, which can recover training data information from the historical model and use it for knowledge transfer. This framework synthesizes a new form of data from the historical model. Specifically, we 'invert' an off-the-shield pretrained model to synthesize binary class user-item pairs beginning from random noise without requiring any additional information from the training dataset. To synthesize informative data from a pretrained model, we propose a new continuous data type rather than the original one- or multi-hot vectors. An additional statistical regularization is added to further improve the quality of the synthetic data inverted from the deep model with batch normalization. The experimental results show that our framework can generalize across different types of models. We can efficiently train different types of classical Click-Through-Rate (CTR) prediction models from scratch with significantly few inversed synthetic data (2 orders of magnitude). Moreover, our framework can also work well in the knowledge transfer scenarios such as model retraining and data-free knowledge distillation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Online advertising**.

KEYWORDS

Recommendation System, Model Reusing, Data Synthesis, Model Inversion

ACM Reference Format:
Cheng Wang, Jiacheng Sun, Zhenhua Dong, Jieming Zhu, Zhenguo Li, Ruixuan Li, and Rui Zhang. 2023. Data-free Knowledge Distillation for Reusing Recommendation Models. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3604915.3608789

1 INTRODUCTION

Recommendation has been widely used by various kinds of content providers such as Amazon [27], YouTube [7], and TikTok [18]. They make customized products or service recommendations to consumers based on the assumption that users' interests may be derived from their prior activities or other users with similar preferences.

Due to the dynamic nature of item features like popularity and user preferences, it is critical for a model-based recommender system (RS) to fit the newest user-item preferences. It is challenging to apply real-time updates on the models in an online fashion due to the increasing complexity of recommendation models [40]. *Therefore, it is more common to adopt an offline training strategy that retrains a recommendation model with massive data to handle the various user-item behaviour from a global perspective* [35, 41]. Unlike online models that directly update the parameters in real-time

*Work done as an intern in Huawei Noah's Ark Lab.
†Corresponding authors.

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09…$15.00
https://doi.org/10.1145/3604915.3608789

with new data, offline models often need to use a certain period of data to train a new model from scratch to replace the outdated historical models [40]. These historical models, which cost a lot of computational resources and manual feature engineering [4, 39], are directly replaced by newly trained models to maintain the freshness of the recommendation model. However, there is much valuable information in historical models since massive of prior user-item behaviours are used to train these models. These prior behaviours can enrich the user's behaviour information to improve the performance of the recommendation model. *Hence, in this paper, we investigate an interesting but seldomly discussed problem, i.e., how to reuse and transfer the valuable information that is included in historical models for various downstream applications.*

Knowledge Distillation (KD) is a pervasive way for knowledge transfer [13], where the initialized model (student) is trained to imitate the outputs of the pretrained model (teacher) and labeled raw data. Through the additional supervision of the teacher model, the student model can achieve comparable performance with the teacher model in a shorter time [28]. However, KD still has several limitations. First, real user-item interactions are necessary during KD, which is very restrictive in practice. Second, the KD method transfers knowledge indirectly and inflexibly since it should compare the divergence of both outputs between the teacher and student. Hence, we need a more persuasive and flexible way to extract knowledge from the historical models.

To solve the challenge we mentioned above, we propose an insightful model inversed data synthetic framework for recovering informative training data information from the pretrained historical recommendation models and using it for knowledge transfer. By fixing the parameters of the pretrained model and using backpropagation to optimize the randomly initialized inputs continuously (without any additional training data information), we can successfully distill the training data information encoded in the recommendation model's parameters by our model inversed data synthetic framework. However, the particularity of the recommendation model and data makes the implementation of model inversion have the following difficulties: (1) general recommendation data are always mixed with the numerical and categorical features, which are inappropriate as optimization objects. (2) The information we distilled from a specific model should have cross-architecture performance, i.e., inverted information can easily transfer to other unseen RS models for training. (3) To recover the original training set information from the pretrained model as much as possible, it is imperative to ensure that the inversed information has enough diversity. To address these issues, we design a new form of data, which firstly transfers the non-derivable sparse and discrete recommendation data to dense and continuous data, while the new form of data can directly transfer to different embedding-based recommendation models. To fortify the diversity of the synthetic data, we fix a section of vector in synthetic data not updating in the process of optimization. An important observation about the deep model in practice is that almost all of them use batch normalization layer. Hence, we further improve the quality of the synthetic data from the deep recommendation model by adding an additional regularization term that utilizes the running means and variances [14] stored in the batch normalization layers. The experiment results show that our framework can distill prosperous information from prior training data that was used to train this model. At the same time, our synthetic data can significantly reduce the amount of data required to train a RS model.

In a nutshell, this work makes the following main contributions:

- We first adopt the standard linear regression to formally explain why model inversion can recover informative training data knowledge with only the pretrained model. Then, We introduce a general model inversed data synthesis framework which can recover training data information from the pretrained model without any real data.
- We propose a new form of data for our inverse-synthetic data to be optimized in the pretrained model rather than the one- or multi-hot data. It can successfully train a model within significantly small inverse-synthetic data (2 orders of magnitude). Further, we improve the synthetic data quality that inverses from the deep recommendation model by adding an additional regularization term.
- We conduct extensive experiments on three different types of CTR prediction models (from linear to deep) to validate the effectiveness and efficiency of the inverse-synthetic data. Moreover, we explore the application of knowledge transfer, such as data-free knowledge distillation and model retraining. Our inverse-synthetic data shows highly competitive with other state-of-the-art.

2 RELATED WORK

Data-free knowledge transfer. To better leverage the potential of well-trained models, knowledge transfer aims to distill valuable knowledge from these pretrained models to an intra-domain or cross-domain model [19]. Then, to overcome the barrier of large datasets or privacy concerns, data-free knowledge transfer deals with knowledge transfer via pseudo-data synthesis to train students without using any real data [22]. This type of method has not attracted much attention in recommendation systems. Yue et al. [38] propose data-free model extraction to launch profile pollution and poisoning attack over sequential recommendation models.

Model inversion was first proposed by [10], aiming to steal recognizable images from models, especially when attackers lack training data information. Subsequent works found that it can inverse training data information starting from random noise with additional image prior [21] and statistical regularization [36] which highly fulfill the data-free setting. However, model inversion has two shortcomings: (1) Although the inversed information as the auxiliary prior has good performance in the downstream tasks, it cannot train a model from scratch with only the inversed information [8]. (2) Inversion-based methods generate high sparse one- or multi-hot data poorly since discrete user or item vector would block the gradient flow, causing backpropagation training invalid. To the best of our knowledge, we are the first paper that uses the model inversion paradigm for data synthesis in the recommendation and overcomes all the deficits mentioned above which further benefit the model reuse.

Data Synthesis for recommendation. Data synthesis is an influential data augmentation technology widely used in Computer Version (CV) and Natural Language Processing (NLP) for solving

privacy-preserving, class imbalance, model unrobustness, etc. However, synthesizing recommendation data is challenging due to its high-dimensional and sparse properties. Generative Adversarial Networks (GAN) have been wildly adapted in the recommendation field for data synthesis. It uses the generator to increase the amount and diversity of data to alleviate the data sparsity problem [2, 25, 30]. IRGAN [29] makes the first attempt by adopting a minimax game to unify the generative and discriminative models in information retrieval. AugCF [30] uses GAN to directly generate training data to reduce data sparsity. Meanwhile, there are some GAN-based strategies for specific scenarios, such as synthetic data for privacy-preserving [17], or in sequential prediction tasks [34]. In recommendation field, another correlative line of work is data imputation technique [33] which aims to generate missing user-item ratings in the rating matrix. The deficit of such a method is that it is not end-to-end and requires extra manual work. One main barrier of the above data synthesis methods to apply in practice is that they assume the real data is available. However, this setting is restrictive and unrealistic.

Click-Through Rate Prediction (CTR). Most existing approaches consider CTR prediction as binary classification problems. The evolution of the CTR model [43] has experienced blooming from classic machine learning such as LR, FM, to deep learning such as DNN [6], Wide&Deep [5], DeepFM [11], DIN [42], etc. As the classic machine learning model for CTR prediction, LR [32] is still a widely used method due to its strong interpretability and engineering requirements. However, the weak generalization ability and feature interaction of linear models makes it easier to cause the loss of training information. In order to solve the problem that the LR model cannot perform feature interaction through suitable methods, Rendle et al. [26] proposed FM, which learns latent vectors for each user and item. At the same time, it reduces the complexity of straightforward computation of degree-2 polynomial LR model from $O(n^2)$ to $O(kn)$ ($k \in \mathbb{N}^+$ is a hyperparameter that defines the dimensionality of the factorization.). Since FM only solves poly-2 feature interactions. To learn high-order feature interactions. DeepFM was proposed [11], it uses the FM component to capture poly-2 feature interactions and a feed-forward neural network component for the higher feature interactions. This paper studies these three classic CTR prediction models to evaluate our model inversed data synthesis approach.

3 PRELIMINARIES

In this section, we formulate the recommendation task and perform a formal analysis of model inversion. To discuss the problem of reusing recommendation models. The recommendation task we formulate thereafter is the CTR prediction model, which is a core solution of advertising recommender systems.

3.1 Task Formulation

Suppose we have a CTR prediction task [15] with a user set $U = \{u_1, u_2, ..., u_n\}$ and an item set $V = \{v_1, v_2, ..., v_m\}$, where n (or m) is the number of users (or items). We denote the boldface $\mathbf{u}_i$ (or $\mathbf{v}_j$) as the one-hot representation of user u_i (or item v_j). Let $r \in R$ denotes the feedback given by a user to an item, while exhibited events with and without click response are classified as positive and negative instances. The collected user-item behaviour data D can be denoted as a set of triplets (u_i, v_j, r) over the user-item-label space $U \times V \times R$ [3]. We denote $\ell(\cdot, \cdot)$ as the loss function between the prediction and ground-truth label. Formally, under a certain dataset D, the goal of CTR prediction is to learn the parameters of the function $f_\theta : U \times I \rightarrow R$ to minimize the following objective function:

$$\min_\theta L(u_i, v_j, r) = \frac{1}{|D|} \sum_{h=1}^{|D|} \ell(f_\theta(u_i, v_j), r). \tag{1}$$

Where f_θ is denoted as a specific CTR model learned with parameters θ. Typically, the categorical user/item features are mapped into low-dimensional latent vectors through embedding techniques. We denote $\mathbb{E}$ as the embedding matrix that learned in f_θ on data D.

$$\mathbb{E} = \left(\mathbf{e}_1^u \dots \mathbf{e}_n^u, \mathbf{e}_1^v \dots \mathbf{e}_m^v\right)^T, \tag{2}$$

$\mathbf{e}_i^u$ (or $\mathbf{e}_j^v$) denotes the embedding vector of user u_i (or item v_j). After some epochs of training, the CTR prediction model can accept a pair of user-item (u_i, v_j) to make a prediction whether or not the user u_i is going to click on the recommended item v_j.

3.2 Motivation

Model inversion is a novel way to recover informative training data with similar distribution to raw training data [9]. Take linear regression as an example (Deep models have similar properties, which have been proven in [36]), let $f_\theta(x) = \theta^\top x$ be the convergent regression model under some training samples, where $x = (u, v)$, θ is the least square solution of training samples, loss function $\ell = \ell_2$ be the square loss. The *model inversion* optimization problem is defined as:

$$x^* = \arg\min_x \ell(\theta^\top x, r) := (\theta^\top x - r)^2. \tag{3}$$

Instead of optimizing model parameters in training process, model inversion updates the inputs x iteratively to generate samples x^* which minimize the loss under a random set label of r. In Figure 1, for linear regression problem, it is well known that the optimal solution of Eq.(3) is a linear space with normal direction θ orthogonal to it, i.e., $P = \{x : \theta^\top x - r = 0\}$. If we used gradient descent to optimize Eq. (3), we could get the optimal solution x^* to be the linear projection of initialization x_0 onto the linear space P. Specifically, we can update the sample x_0 iteratively by

$$x_{k+1} = x_k - \eta \frac{\partial \ell}{\partial x_k}, \tag{4}$$

where the gradient can be calculated from Eq. (3) as:

$$\frac{\partial \ell}{\partial x_k} = 2(\theta^\top x_k - r)\theta. \tag{5}$$

From Eq. (5) we know that gradient descent moves towards the linear solution space P orthogonal in the direction θ. As moving towards the plane P, the gradient $\frac{\partial \ell}{\partial x_k}$ becomes smaller and smaller, until x^* exactly lies on P. Thus, model inversion can generate new sample x^* lies in the plane P which training data determined. This toy example motivates us to use model inversion as an synthesis method to generate synthetic samples.

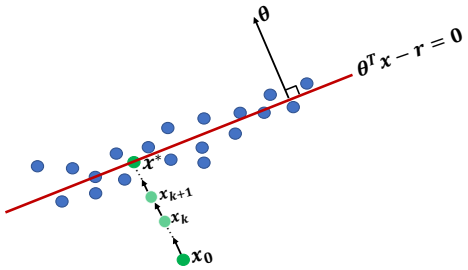

Figure 1: Model inversion on the linear regression model. θ is the normal direction of P, the updated sample approaches P in the direction of θ.

4 MODEL INVERSED DATA SYNTHESIS FRAMEWORK

Although the reutilization of historical data can enhance the distribution of users' behaviour, thus improving the performance of recommendation models, the inclusion of excessive historical data alongside the latest data in training can significantly hamper real-time recommendation. Therefore, it becomes imperative to strike a balance between real-time deployment and performance. Furthermore, conventional knowledge distillation (KD) methods fail to adequately address the challenges posed by massive data management and privacy concerns. To overcome these challenges, we propose a novel model inversed data synthesis framework that extracts valuable knowledge from historical models without the need for additional real data. Our preliminary experiments observe that we can use a few inverse-synthetic data to efficiently train a comparable model and boost the model retraining task with performance improvement. The workflow of our framework is by fixing the parameters in the pretrained model and using backpropagation to optimize the randomly initialized inputs continuously, as illustrated in Figure 2.

As described in Section 3.2, when we specify the target label, the initialized random input can be gradually moved to the distribution of the raw training set by gradient descent. In the CTR prediction task, the goal is to optimize the user $\mathbf{u}_i$'s vector and item $\mathbf{v}_j$'s vector as follow:

$$(\mathbf{u}_i, \mathbf{v}_j) = \arg\min_{\mathbf{u}_i, \mathbf{v}_j} \ell(f_\theta(\mathbf{u}_i, \mathbf{v}_j), r). \tag{6}$$

However, the discrete one- or multi-hot recommendation data would block the gradient flow, causing backpropagation training invalid [30]. Hence, we reconstruct the data as continuous tensors so that our input can get the back-propagated gradient. The form of our new inverse-synthetic data can be recognized as:

$$\begin{aligned} \boldsymbol{\mu} &= (\omega_1^{\boldsymbol{\mu}}, \omega_2^{\boldsymbol{\mu}}, ..., \omega_n^{\boldsymbol{\mu}}), \omega^{\boldsymbol{\mu}} \sim U(0,1), \\ \boldsymbol{\nu} &= (\omega_1^{\boldsymbol{\nu}}, ..., \omega_2^{\boldsymbol{\nu}}, ..., \omega_m^{\boldsymbol{\nu}}), \omega^{\boldsymbol{\nu}} \sim U(0,1), \end{aligned} \tag{7}$$

subject to:

$$\sum_{i=1}^{n} \omega_i^{\boldsymbol{\mu}} = 1, \sum_{j=1}^{m} \omega_j^{\boldsymbol{\nu}} = 1. \tag{8}$$

Where $\omega^{\boldsymbol{\mu}}$ and $\omega^{\boldsymbol{\nu}}$ are sampled from the uniform distribution. It can be seen that the one-hot represents of user vector $\mathbf{u}_i$ and item vector $\mathbf{v}_j$ are the special case of our new form of inverse-synthetic data (if and only if $w_i^{\boldsymbol{\mu}} = w_j^{\boldsymbol{\nu}} = 1$ and $\omega_{i'}^{\boldsymbol{\mu}} = \omega_{j'}^{\boldsymbol{\nu}} = 0$ for any $i' \neq i$ and $j' \neq j$). Hence, given an indicated target label $r \in R$, a pretrained model f_θ with derivable new data form can adapt our model inversed data synthesis framework to find a user-item pair of vector $(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ to minimize the loss function ℓ, i.e.,

$$(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*) = \arg\min_{(\boldsymbol{\mu}, \boldsymbol{\nu})} \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}), r). \tag{9}$$

The generated new samples can be interpreted as a prototype of real data that is used to train the historical models. They are required to preserve informative knowledge for training recommendation models rather than interpretability for humans. The experiments (Section 5.2, 5.3) show that the synthetic data generate from our framework has valuable information to train a model from scratch or help the downstream tasks.

4.1 Differences Between Raw Data Training and inverse-synthetic Data Training

Assuming that we have a piece of raw data about user u_i's preference about item v_j, in the processing of training a model, the feature embedding matrix of data (u_i, v_j) will be obtained by following:

$$\begin{aligned} (\mathbf{u}_i, \mathbf{v}_j)\mathbb{E} &= (\omega_i^u e_i^u, \omega_j^v e_j^v), \\ s.t. & \\ \omega_i^u &= \omega_j^v = 1. \end{aligned} \tag{10}$$

When obtaining the corresponding feature embedding matrix $(e_i^u, e_j^v) \in R^{2\times k}$, where k is the predefined dimension of each embedding vector. As the state-of-the-art deep CTR prediction model, the Deep Factorization Machine (DeepFM) [11] will feed the feature embedding matrix (e_i^u, e_j^v) into the wide component-FM and deep component-neural network, enabling DeepFM to learn low- and high-order feature interactions simultaneously from the input. The DeepFM model will continually optimize this pair of feature embedding vectors via gradient descent.

Feed-forward and back-propagation mechanism of the new data format: Specifically, in the training process of feed-forward, a piece of raw data will assign weight 1 to user u_i and item v_j's feature embedding vector. In contrast, a piece of inverse-synthetic data is the strategy to assign weight to all feature embedding vectors. Hence, in our inverse-synthetic data training scenario, the case will be as follow:

$$(\boldsymbol{\mu}, \boldsymbol{\nu})\mathbb{E} = (\omega_1^{\boldsymbol{\mu}} e_1^u, \omega_2^{\boldsymbol{\mu}} e_2^u, ..., \omega_n^{\boldsymbol{\mu}} e_n^u, \omega_1^{\boldsymbol{\nu}} e_1^v, \omega_2^{\boldsymbol{\nu}} e_2^v, ..., \omega_m^{\boldsymbol{\nu}} e_m^v). \tag{11}$$

With this operation, we obtain a feature embedding matrix $(\boldsymbol{\mu}, \boldsymbol{\nu})\mathbb{E} \in R^{(m+n)\times k}$. Such embedding matrix can not be directly adapted in the following feed-forward process. In order to allow end-to-end training, we aggregate the weighted embedding vectors according to the user and item field as follows:

$$agg[(\boldsymbol{\mu}, \boldsymbol{\nu})\mathbb{E}] = (\sum_{i}^{n} \omega_i^{\boldsymbol{\mu}} e_i^u, \sum_{j}^{m} \omega_j^{\boldsymbol{\nu}} e_j^v). \tag{12}$$

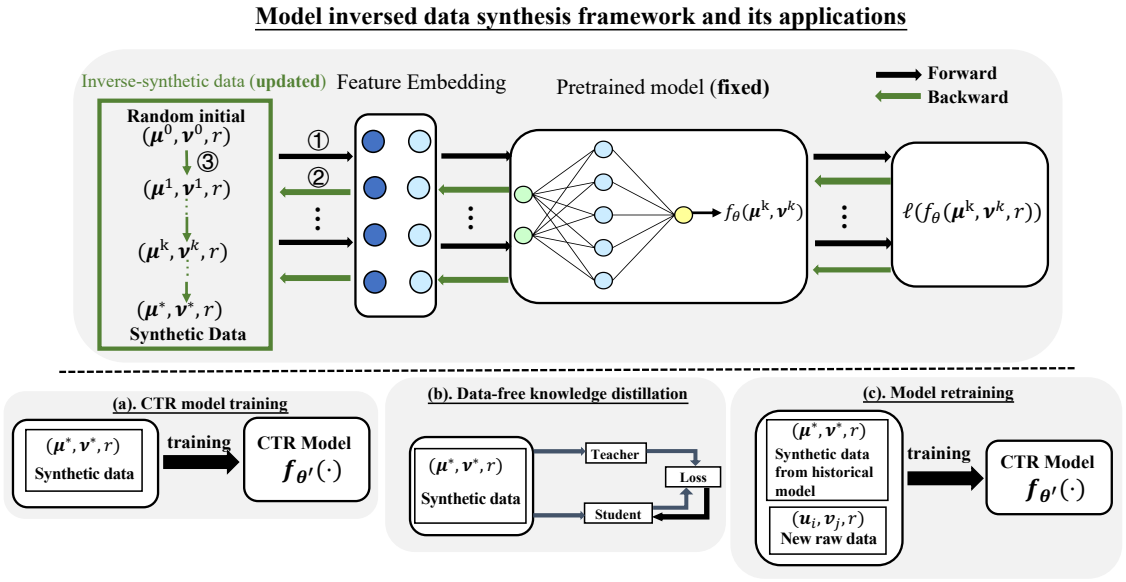

Figure 2: The model inversed data synthesis framework optimizes random noise into binary class synthetic data given just a pretrained CTR model. It has following applications: (a) using the synthesized data, we enable directly training an CTR model starting from scratch (Section 5.2); (b) Our inverse-synthetic data are applicable to data-free knowledge distillation (Section 5.3); (c) the inverse-synthetic data can improve the efficiency and performance in the model retraining scenario (Section 5.3).

After that, the dimension of the feature embedding matrix from synthetic data is identical to a piece of raw data's feature embedding matrix. Finishing the feed-forward process, the synthetic user vector $\boldsymbol{\mu}$ and item $\boldsymbol{\nu}$ vector can update parameter with gradient descent as follow:

$$
\begin{aligned}
(\omega_i^{\boldsymbol{\mu}})_{t+1} &= (\omega_i^{\boldsymbol{\mu}})_t - \eta \nabla \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu})_t, y), \\
(\omega_j^{\boldsymbol{\nu}})_{t+1} &= (\omega_j^{\boldsymbol{\nu}})_t - \eta \nabla \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu})_t, y).
\end{aligned} \tag{13}
$$

Where t denotes the t-th iteration, and η is the learning rate of synthetic data.

4.2 Diversity of The Inverse-synthetic Data with Gradient Freezing

Aside from quality, diversity also plays a crucial role in avoiding repeated and redundant synthetic data since the lack of diversity can easily lead to model overfitting and low-performance issues [1]. To overcome such issues, various strategies, such as the min-max training competition [29] and the truncation trick [1] have been proposed. However, these methods, which rely on the joint training of two networks over raw data, are consequently inapplicable to our data-free case. So far, we have two tricks to guarantee the diversity of synthetic data. One is that synthetic data is randomly initialized from a uniform distribution, and another is that the labels are randomly assigned to synthetic data. To further avoid the repeated and redundant synthetic data, we add an additional gradient freezing trick, i.e., we freeze the gradient of either the synthetic vector of user μ or the synthetic vector of item ν during

Algorithm 1: Model Inversed Data synthesis

input : Pretrained recommendation model f_θ; Training epochs T; Synthetic data size N, Learning rate η, Binary cross entropy ℓ, Softmax function $\sigma(\cdot)$; Loss threshold ϵ;
output: N pieces of inverse-synthetic data $(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*, r)$;

for *epoch* $\leftarrow 0$ **to** T **do**
  initialize one batch of user vectors $\mathcal{U}^0$ and item vectors $\mathcal{V}^0$ from random noise;
  /* Normalize synthetic user and item vector to fulfill the constraint of Eq.(8) */
  foreach $\boldsymbol{\mu}^0 \in \mathcal{U}^0$ **do** $\boldsymbol{\mu}^0 \leftarrow \sigma(\boldsymbol{\mu}^0)$;
  foreach $\boldsymbol{\nu}^0 \in \mathcal{V}^0$ **do** $\boldsymbol{\nu}^0 \leftarrow \sigma(\boldsymbol{\nu}^0)$;
  initialize one batch of random label $\mathbf{R}$;
  while *True* **do**
    /* Feed-forward with Eq.(12) */
    $\hat{\mathbf{R}} \leftarrow f_\theta(\mathcal{U}^k, \mathcal{V}^k)$;
    $loss_k \leftarrow \ell(\hat{\mathbf{R}}, \mathbf{R})$;
    if $loss_k \geq \epsilon$ **then**
      $(\mathcal{U}^{k+1}, \mathcal{V}^{k+1}) \leftarrow (\mathcal{U}^k, \mathcal{V}^k) - \eta(\frac{\partial loss_k}{\partial \mathcal{U}^k}, \frac{\partial loss_k}{\partial \mathcal{V}^k})$;
    else
      $(\mathcal{U}^*, \mathcal{V}^*) \leftarrow (\mathcal{U}^k, \mathcal{V}^k)$;
      break;
    end
  end
end

the training process. Therefore, the optimization problem of inverse-synthetic data is changed to the following:

$$(\boldsymbol{\nu}^*|\boldsymbol{\mu}, r) = \arg\min_{\boldsymbol{\nu}} \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}), r) \quad or \quad (\boldsymbol{\mu}^*|\boldsymbol{\nu}, r) = \arg\min_{\boldsymbol{\mu}} \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}), r). \tag{14}$$

After that, the synthetic data will optimize under the condition of both gradient freezing vector and randomly assigned label. The intrinsic behind the gradient freezing trick to enhance diversity is that the freezing vector will constrain the direction of parameter update of the unfrozen vector. It is because the freezing vector is randomly initialized, and it will generate a unique feature embedding matrix when it feed-forward in the CTR model. Then, the feature embedding matrix obtained from the unfrozen vector will combine the freezing embedding matrix to the following computation. When back-propagation updating, the unfrozen synthetic vector will be constrained under this freezing embedding matrix and randomly assigned label. We detailedly discuss the gradient freezing trick in Section 5.5.

4.3 Batch Norm Statics for Deep Model

We extend a new feature distribution regularization term to further improve the quality of our inverse-synthetic data. We strive to reduce the difference of feature map statistics w.r.t embedding (μ, ν) and (u, v). In order to successfully enforce the similarity of feature statistics at all multi-layer perceptron (MLP) layers utilized in deep recommendation models. We assume that feature statistics follow the Gaussian distribution across batches of data and can be defined by mean δ and variance σ^2. The feature distribution regularization term can therefore be written as follows:

$$R_{feature} = \sum_l ||\delta_l(\mu, \nu) - \mathbb{E}(\delta_l(u, v)|(U, V)||+ \sum_l ||\sigma_l^2(\mu, \nu) - \mathbb{E}(\sigma_l^2(u, v)|(U, V))||. \tag{15}$$

Where $\delta_l(\mu, \nu)$ and $\sigma_l^2(\mu, \nu)$ denote as the batch-wise mean and variance to that of l^{th} MLP layer. We denote $\mathbb{E}(\cdot)$ and $||\cdot||$ operators as the expected value and ℓ_2 norm calculations. It might seem we need mean and variance of original data to obtain $\mathbb{E}(\delta_l(u, v)|(U, V))$ and $\mathbb{E}(\sigma_l^2(u, v)|(U, V))$. Fortunately, the running average statistics stored in the widely-used BatchNorm (BN) layers are more than sufficient. BN as the key tool to alleviate covariate shifts [14] during training are implicitly capture the running mean and variance. Hence, the mean and variance of batches of data can be represented as follow:

$$\mathbb{E}(\delta_l(u, v)|(U, V)) \simeq BN_l(running\_mean), \quad \mathbb{E}(\sigma_l^2(u, v)|(U, V)) \simeq BN_l(running\_variance). \tag{16}$$

This regularization term customized for the deep CTR model can effectively help improve the quality of inverse-synthetic data generation and the performance of downstream tasks such as data-free knowledge distillation and model retraining. With the additional feature distribution regularization term, the recommendation data is synthesized by optimizing the following term:

$$(\boldsymbol{\mu}^*, \boldsymbol{\nu}^*) = \arg\min_{(\boldsymbol{\mu}, \boldsymbol{\nu})} \ell(f_\theta(\boldsymbol{\mu}, \boldsymbol{\nu}), r) + \alpha_f R_{feature}. \tag{17}$$

Where α_f is a hyperparameter.

Table 1: Statistics of the datasets.

Dataset	user	item	Interactions
Yahoo! R3 uniform	5,400	1,000	54,000
Yahoo! R3 selected	15,400	1,000	311,704
MovieLens1M	6,040	3,952	1,000,209

4.4 Model Inversed Data Synthesis Algorithm

For model inversed data synthesis, we adopt mini-batch training in our implementation. Specifically, in each epoch, we first initialize a batch of user-item feature embedding vector pairs $(\mathcal{U}^0, \mathcal{V}^0)$ from the random noise, then we generate the same amount indicated binary-class labels. Keeping the proportion of negative labels higher than positive labels is better because user preferences for positive and negative behaviors are naturally imbalanced. Each user vector $\boldsymbol{\mu}^0$ and item vector $\boldsymbol{\nu}^0$ should go through a softmax function [24], to normalize the sum of the weights of each vector, which can further help the inverse-synthetic data converge to satisfactory loss values faster. We use the pretrained CTR model parameters to update the inverse-synthetic data using gradient descent gradually. We stop training when the loss is smaller than an arbitrarily specified loss threshold b. The pseudo-code for the entire model inversed data synthesis is given in Algorithm 1.

5 EXPERIMENTS

We conduct experiments in this section to evaluate the performance of our proposed model inversed data synthetic method. The majority of our experiments will focus on the following research questions: **RQ1:** Compared with other data synthesis SoTA, how is the quality of the inverse-synthetic data generated from our framework? **RQ2:** Compared with other data-free knowledge transfer SoTA in the scenario of data-free knowledge distillation, how effective of our framework can distill informative training information from pretrained model? **RQ3:** How is the performance of the inverse-synthetic data compared with other sample-based model retraining methods?

5.1 Experimental Setup

To evaluate the generalization and efficiency of the model inverse-based data synthetic method from different perspectives, our experiments compare with different state-of-the-art methods under varying scenarios on two public datasets.

5.1.1 Datasets. We evaluate our approach with the following two datasets, and the statistics are shown in Table 1.

Yahoo! R3 [20]: This dataset is provided by Yahoo! music recommendation service. We follow [37] to divide this dataset into two parts according to Yahoo!'s different collection strategies. Hereafter, these two parts are named as Yahoo uniform dataset and Yahoo selected dataset. The uniform dataset is carried out by 5400 survey participants who rate 1000 randomly selected songs during an

Table 2: The AUC scores and LogLoss of our inverse-synthetic data compare with other generative models on Yahoo R3 and Movielens. Boldface denotes the best result, underline is secondary.

	Yahoo R3! uniform						**Yahoo R3! selected**						**Movielens**					
	LR		FM		DeepFM		LR		FM		DeepFM		LR		FM		DeepFM	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
raw data (full)	0.7512	0.3906	0.7544	0.3891	0.7561	0.3839	0.8218	0.4907	0.8231	0.4902	0.8321	0.5314	0.7935	0.538	0.8063	0.525	0.8156	0.5148
raw data (10K)	0.6424	0.5948	0.6401	0.8753	0.6328	0.7823	0.6731	0.7269	0.6637	0.7374	0.6595	0.8227	0.6636	0.7621	0.6458	0.7935	0.6524	0.9382
IRGAN (10K)	0.6475	0.5996	0.6479	0.8188	0.6403	0.7746	0.6888	0.7182	0.6715	0.7284	0.6753	0.8123	0.6652	0.7502	0.6519	0.7894	0.6556	0.9371
AugCF (10K)	0.6487	0.5961	0.6493	0.8152	0.6411	0.7721	0.6894	0.7175	0.6745	0.7261	0.6773	0.8105	0.6673	0.7493	0.6544	0.7826	0.6577	0.9307
LRSD (2K)	0.7442	**0.4608**	0.7093	0.4849	0.6884	0.6025	0.8128	**0.5117**	**0.8107**	0.5751	0.7934	0.6301	0.7442	**0.5868**	0.7435	0.5864	0.6912	0.7093
FMSD (2K)	0.6998	0.5472	**0.7265**	**0.3875**	0.6886	0.6012	0.7917	0.5571	0.7967	**0.5569**	0.7944	**0.6088**	**0.7138**	0.7799	0.7293	0.6075	0.6706	0.7338
DeepFMSD (2K)	0.7427	0.4944	0.6916	0.6752	0.6895	0.5993	0.8117	0.5359	0.7957	0.6421	0.7968	0.6641	0.7149	0.6513	0.7207	0.6193	0.6889	0.7376
DeepFMSD-BN (2K)	**0.7461**	0.4702	0.6988	0.6546	**0.6919**	**0.5938**	**0.8135**	0.5241	0.7965	0.6351	**0.8008**	0.6226	0.7224	0.7752	**0.7619**	**0.5806**	**0.7367**	**0.6396**

Table 3: The AUC scores and LogLoss of our inverse-synthetic data compare with DFME in the scenario of data-free knowledge distillation on Yahoo R3 and Movielens.

	DFME (10K)			Ours (2K)			
	LRME AUC/LL	FMME AUC/LL	DeepFMME AUC/LL	LRSD AUC/LL	FMSD AUC/LL	DeepFMSD AUC/LL	DeepFM-BN AUC/LL
YahooR3 uniform	0.6863/0.7678	0.6777/0.6525	0.6836/0.6487	0.6893/0.5736	0.6923/0.5674	0.6937/0.5522	**0.7091/0.5391**
YahooR3 selected	0.7515/0.8142	0.7173/0.7659	0.7371/0.7451	0.7945/0.6123	0.7952/0.6101	0.7953/0.6057	**0.7989/0.6035**
Movielens	0.7191/0.8391	0.6688/0.6915	0.6981/0.6811	0.7424/0.5878	0.7204/0.6092	0.7192/0.6106	**0.7499/0.5989**

online survey. Each participant was instructed to rate at least ten songs chosen at random by the system from the 1000 selected songs. The selected dataset is conducted by 10,000 non-survey participants who choose and rate the 1,000 songs according to their own wishes. The reason for dividing the Yahoo! R3 dataset is to verify that our proposed method can decouple the potential training data from the model trained with uniform data or selected data.

MovieLens [12]: Movielens consists of users' tagging records on movies. We focus on personalized tag recommendation by converting each tagging record (user ID, movie ID, rating) to a feature vector as input.

We formalize the task as a regression problem, in which all ratings for 1-3 are normalized to be 0, 4-5 ratings to be 1. The ratio of train, test, and validation set sizes is 8:1:1.

5.1.2 Evaluation metrics. We adopt two metrics for performance evaluation of CTR prediction tasks: AUC (Area Under the ROC curve) and Logloss (cross-entropy). The higher AUC scores or the lower Logloss represents the better performance of the CTR prediction model.

5.1.3 Baselines. To evaluate the generalization of our framework, we implement it on three different types of CTR prediction models (from linear to deep model). (1) Logistic Regression (LR) [23] is a popular linear recommendation model; (2) Factorization Machine (FM) [26] is the combination of linear regression and matrix factorization; (3) Deep Factorization Machine (DeepFM) [11] is a well-known deep model which consists of wide component-FM and deep component-neural network.

SoTA for RQ1: Our framework can be realized as a generative model that can synthesize new data from a pretrained model. Hence, we first compare our framework with other data synthesis methods.

- **IRGAN** [29]: It is a classic min-max Generative Adversarial Network (GAN) for information retrieval, and we use the converged generative model that is trained on the whole raw data to synthesize new data.
- **AugCF** [30]: It is based on a Conditional Generative Adversarial Net that is trained on whole raw data to augment reliable user interaction.

SoTA for RQ2: Our framework can extract informative information from a pretraind model for knowledge transfer without any raw data. Hence, we compare our method with other data-free model extraction methods in the application of data-free knowledge distillation.

- **Data-free model extraction (DFME) [38]:** It adopts the data-free model extraction to launch a Black-Box attack on sequential recommenders. It contains two modules, one is the model extraction module, and another is the downstream attacks module. The model extraction module synthesizes informative data via API queries. Then, use model distillation to minimize the difference between the victim and surrogate model by training with generated data. We select the data-free model extraction module as the SoTA and abbreviate it as DFME.

SoTA for RQ3: Our framework can reuse historical models by generating synthetic data as a memorization of historical user interest. Hence, following [40], we compare our synthetic data with other representative sample selection methods as the replay of historical user interest in the application of model retraining.

- **Random**: It randomly sample data instance from historical data as the replay for the retraining model.
- **Fine-tune**: This method updates the CTR model with newly collected data.
- **SPMF** [31]: This is a state-of-the-art streaming recommendation method that belongs to the category of sample-based retraining.

5.1.4 Implementation Details. In this part, we briefly introduce the types of our generated data, parameters setting.

The types of inverse-synthetic data: We have four types of synthetic data, which are inversed from the pretrained LR, FM, DeepFM, and DeepFM with BN loss respectively. The synthetic data we inversed from the LR model will be abbreviated as LRSD. Similarly, we also have FMSD and DeepFMSD, and the BN loss used in DeepFM is abbreviated as DeepFMSD-BN.

Parameters setting: We implement all algorithms in PyTorch and train on a single NVIDIA Quadro RTX5000 (16GB memory). The overall experiments are of two stages. (1) *data synthetic:* Adopting our framework to synthetic data from a pretrained model. (2)*model training:* Adopting the synthetic data to train models over different downstream tasks. In the first stage, the pretrained models (LR, FM, DeepFM) are trained with Binary Cross Entropy (BCE) loss and Adam optimizer (learning rate 0.01). Then, we use Adam (a grid search of learning rate $[1e-4, 1e-5, 1e-6]$ and weight decay $[0.1, 0.05, 0.001, 0.005]$) for optimization the synthetic data. While synthetic data from deep model with BN loss in Eq.(17), the $\alpha_f = [0.1, 0.05, 0.01]$ for the BN loss term. In the second stage, we use Adam optimizer (a grid search of learning rate $[0.1, 0.01, 0.001]$ and weight decay $[1e-5, 1e-6, 1e-7]$) for training different type of synthetic data. While in the data-free knowledge distillation scenario, the temperature $\tau = [2, 3, 4]$.

5.2 Performance Comparison with Data Synthesis SoTA (RQ1)

Table 2 is the comparison of data that from data synthetic SoTAs and our framework with metrics of AUC scores and LogLoss. In order to make the results as fair as possible, we generate 5x data size from SoTAs (10K) than ours (2K) since the form of our synthetic data is continuous and dense. "raw data (full)" in the table indicates model training on the whole raw training set, which serves as an approximate upper-bound performance. As shown in Table 2, our methods perform better than all the compared data synthetic SoTAs. More specifically, we have the following observations: (1) From the row perspective, when we directly train the model with inverse-synthetic data, the results show that different inverse-synthetic data have distinct advantages, i.e., LRSD is better at training LR and FM models, and FMSD and DeepFMSD with/without BN loss are more advantageous for training their original models. (2) The ablation study of BN loss (DeepFMSD vs. DeepFMSD-BN) shows when we generate synthetic data from DeepFM model implemented with BN loss has better AUC scores when training all types of models. (3) Our framework has a remarkable cross-architecture performance that the synthetic data from a model can be adapted to train other models successfully. (4) The trends of AUC and LogLoss metrics may be inconsistent. For example, some of our strategies have a better AUC value but a poor LogLoss value. Since the LogLoss value is susceptible to the difference in label distribution between the training and test sets, we mainly consider AUC as in [16].

The best training performance of synthetic data on different datasets has an absolute 0.7%-7.89% gap compared with full raw data training. Note that such a result is the amount of raw data has over 120x and 400x than our inverse-synthetic data on the Yahoo R3 selected and Movielens.

Table 4: The AUC scores and LogLoss of synthetic data compare with the representative sample selection SoTAs in model retraining scenario.

Training Data	LR AUC/LL	FM AUC/LL	DeepFM AUC/LL
$Random_{T1} \cup raw_{T2}$	0.7775/0.5661	0.7802/0.5723	0.7768/0.5758
Fine-tune	0.7835/0.5566	0.7847/0.5618	0.7850/0.5603
$SPMF_{T1} \cup raw_{T2}$	0.7885/0.5562	0.7868/0.5613	0.7878/0.5601
$LRSD_{T1} \cup raw_{T2}$	**0.7950**/0.5480	0.7882/0.5737	0.7921/**0.5538**
$FMSD_{T1} \cup raw_{T2}$	0.7941/**0.5478**	**0.7906**/**0.5694**	0.7915/0.5547
$DeepFMSD_{T1} \cup raw_{T2}$	0.7926/0.5485	0.78810.5749	0.7912/0.5572
$DeepFMSD\text{-}BN_{T1} \cup raw_{T2}$	0.7947/0.5486	0.7897/0.5719	**0.7933**/0.5588

Table 5: Ablation study on with/without gradient freezing for enhancing the diversity of inverse-synthetic data.

	YahooR3 uniform AUC/LL	YahooR3 selected AUC/LL	Movielens AUC/LL
without gradient freezing	0.6587/0.6998	0.6468/0.4627	0.6901/0.6456
with gradient freezing	0.6919/0.5938	0.8008/0.6226	0.7367/0.6396

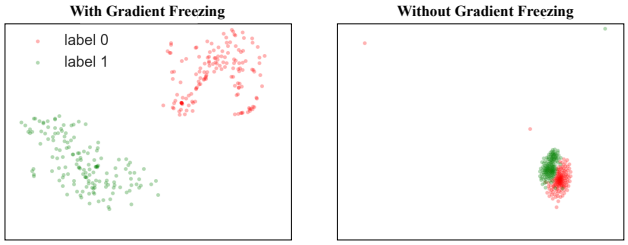

Figure 3: t-SNE visualization of synthetic data with/without gradient freezing.

5.3 Application: Data-free Knowledge Distillation (RQ2)

In this section, we demonstrate the applicability of our proposed method to the data-free knowledge transfer scenario.

Data-free knowledge transfer aims at distilling information from a pretrained teacher model to a student model without using any raw data, which is a more realistic setting in practice. On the one hand, data is a valuable asset and core competitiveness for a commercial company. On the other hand, the privacy of user interest is another primary concern. In this experiment, we apply our framework to synthesize LRSD, FMSD, DeepFMSD, and DeepFM-BN from the models trained on Yahoo R3! and Movielens, respectively. Then, the compared method DFME is with the same operation to synthesize data. We name the synthetic data from LR as LRME over the method of DFME. Similarly, we have FMME and DeepFMME. We use the synthetic data from our framework and DFME to train student DeepFM models with knowledge distillation separately.

Table 3 shows that inverse-synthetic data performs better than data from DFME. It successfully trains a comparable CTR prediction model by knowledge distillation without any help from raw training data. While the inverse-synthetic data generated from the DeepFM model implemented with BN loss further improved AUC scores by 0.36%-3.07% than DeepFMSD.

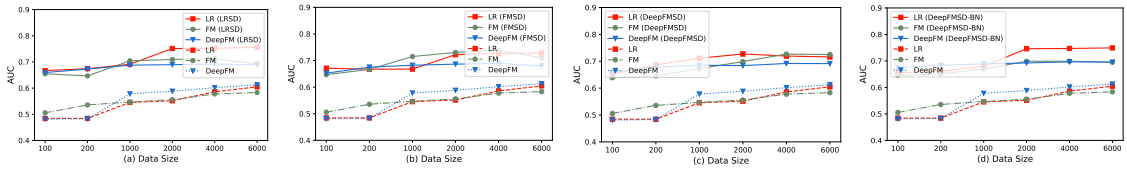

Figure 4: The performance of our framework scale with varying synthetic data size vs. randomly sampled data on Yahoo R3!'s uniform data.

5.4 Application: model retraining (RQ3)

Time sensitivity is critical for CTR prediction to operate well in a real-world environment. Hence, it is necessary for model retraining with the latest user-item interaction. In practice, selecting some representative samples from historical data to incorporate with the latest user-item interaction for model retraining can further benefit the retraining model with a more global perspective of user preference. Instead of directly using historical raw data, our framework can generate informative synthetic data from historical models, which can be realized as the memorization of historical user interest. We compare our method with other core sample selection methods with data size 10K than ours 2K. To simulate a real-world scenario such that a CTR prediction model always uses recorded data to predict future user preferences. we split the MovieLens into three different time windows according to the timestamp. Following [40], the data from time window one is adopted as historical data, time window two is the latest user behaviour directly used for training, and the data from time window three is the test set for simulating the future preferences of users. For example, we denote the synthetic data inversed from the historical LR model in time window one as $LRSD_{T1}$. raw_{T2} is the whole training set on time window two. All results in Table 4 are obtained with testing on raw_{T3}.

In Table 4, the overall quality of our synthetic data is better than the representative sample selection SoTAs in the model retraining scenario. It shows the new data format can perform well when directly combined with raw data. Therefore, in the model retraining scenario, our framework utilizes the potential of the historical model to synthesize data. These synthetic data can be realized as the memorization of historical data. Not only do we reuse the out-of-date models, but we also protect the privacy of historical user interests.

5.5 Ablation Study

Gradient Freezing. We conduct an ablation study to explore whether the gradient freezing trick for increasing the data diversity is desired to be introduced. Table 5 shows the performance of the ablated model where the gradient freezing trick is removed compared with unremoved. From this table, we can conclude that the performance is significantly dropped without gradient freezing. To more specifically discuss the benefit of the gradient freezing trick, we adopt t-SNE to visualize the distribution of synthetic data with/without the gradient freezing trick. As shown in Figure 3, when synthesize data with gradient freezing, the distribution of data is highly dispersed and evenly distributed than without gradient freezing trick.

Scaling with Different Data Size. To validate the availability of inverse-synthetic data when scale with different data size, we employ off-the-shelf pretrained LR, FM, and DeepFM models to generate several forms of inverse-synthetic data, i.e., LRSD, FMSD, DeepFMSD, DeepFMSD-BN and then use these data to train the LR, FM, and DeepFM from scratch. For instance, using LRSD as the training data for training initial LR is denoted by LR (LRSD). Each results are obtained by ten times running. We compare the models' AUC scores trained by inverse-synthetic data with the model trained by the same amount of raw data randomly sampled from the training set of the Yahoo uniform dataset. Across all the cases in Figure 4, the AUC scores of inverse-synthetic data training are better than the directly raw data training. Even the performance with one hundred pieces of inverse-synthetic is still outperformed six thousand pieces of raw data. When the number of inverse-synthetic data exceeds two thousand, the improvement of the AUC scores through synthetic data training is bogged down, and even in some cases, AUC starts to drop slightly. It is because an excess of data will add redundant information to that inverse-synthetic dataset, decreasing model performance.

6 CONCLUSION

In this paper, we are the first to attempt model inversion to achieve data synthesis via a pretrained model on the recommendation field. We propose a universal model inversed data synthesis framework that can be well-performed in different classic CTR models. The inverse-synthetic data has the following characteristics: (1) It can recover training data information used to train a model. (2) Our synthetic data shows effectiveness and efficiency in the knowledge transfer scenarios.

In the future, we are interested in exploring the new data format from the privacy-preserving perspective. The new data format offers a promising solution for discrete data like graphs, and sentiment. At the same time, the data-free property of our framework has the potential in some other applications, such as data-free quantization and pruning for model compression and data-free model attack.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under grants 62206102, U1836204, U1936108, and Science and Technology Support Program of Hubei Province under grant 2022BAA046.

REFERENCES

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations*.

[2] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In *The World Wide Web Conference*. 2616–2622.

[3] Jiawei Chen, Hande Dong, Yang Qiu, Xiangnan He, Xin Xin, Liang Chen, Guli Lin, and Keping Yang. 2021. AutoDebias: Learning to debias for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 21–30.

[4] Suming J Chen, Zhen Qin, Zac Wilson, Brian Calaci, Michael Rose, Ryan Evans, Sean Abraham, Donald Metzler, Sandeep Tata, and Michael Colagrosso. 2020. Improving recommendation quality in google drive. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2900–2908.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[7] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. 293–296.

[8] Xin Dong, Hongxu Yin, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. 2021. Deep Neural Networks are Surprisingly Reversible: A Baseline for Zero-Shot Inversion. *arXiv preprint arXiv:2107.06304* (2021).

[9] Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. 2022. Up to 100x faster data-free knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6597–6604.

[10] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[12] 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.

[13] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[14] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[15] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2021. Towards personalized fairness based on causal notion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1054–1063.

[16] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 831–840.

[17] Fan Liu, Zhiyong Cheng, Huilin Chen, Yinwei Wei, Liqiang Nie, and Mohan Kankanhalli. 2022. Privacy-Preserving Synthetic Data Generation for Recommendation Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1379–1389.

[18] Yang Liu, Cheng Lyu, Zhiyuan Liu, and Dacheng Tao. 2019. Building effective short video recommendation. In *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 651–656.

[19] Yuang Liu, Wei Zhang, Jun Wang, and Jianyong Wang. 2021. Data-free knowledge transfer: A survey. *arXiv preprint arXiv:2112.15278* (2021).

[20] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*. 5–12.

[21] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).

[22] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. 2019. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*. PMLR, 4743–4751.

[23] Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. 2002. An introduction to logistic regression analysis and reporting. *The journal of educational research* 96, 1 (2002), 3–14.

[24] Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[25] AV Prosvetov. 2019. GAN for recommendation system. In *Journal of Physics: Conference Series*, Vol. 1405. IOP Publishing, 012005.

[26] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[27] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon. com. *Ieee internet computing* 21, 3 (2017), 12–18.

[28] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2289–2298.

[29] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 515–524.

[30] Qinyong Wang, Hongzhi Yin, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2019. Enhancing collaborative filtering with generative augmentation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 548–556.

[31] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming ranking based recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 525–534.

[32] Yaozheng Wang, Dawei Feng, Dongsheng Li, Xinyuan Chen, Yunxiang Zhao, and Xin Niu. 2016. A mobile recommendation system based on logistic regression and gradient boosting decision trees. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1896–1902.

[33] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 347–356.

[34] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[35] 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.

[36] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8715–8724.

[37] Bowen Yuan, Jui-Yang Hsia, Meng-Yuan Yang, Hong Zhu, Chih-Yao Chang, Zhenhua Dong, and Chih-Jen Lin. 2019. Improving ad click prediction by considering non-displayed events. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 329–338.

[38] Zhenrui Yue, Zhankui He, Huimin Zeng, and Julian McAuley. 2021. Black-Box Attacks on Sequential Recommenders via Data-Free Model Extraction. In *Fifteenth ACM Conference on Recommender Systems*. 44–54.

[39] Jianyu Zhang and Françoise Fogelman-Soulié. 2018. KKbox's music recommendation challenge solution with feature engineering. In *11th ACM International Conference on Web Search and Data Mining WSDM*.

[40] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? A sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1479–1488.

[41] Erheng Zhong, Nathan Liu, Yue Shi, and Suju Rajan. 2015. Building discriminative user profiles for large-scale content recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2277–2286.

[42] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[43] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open benchmarking for click-through rate prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2759–2769.