# Gradient Matching for Categorical Data Distillation in CTR Prediction

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

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

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

## ABSTRACT

The cost of hardware and energy consumption on training a click-through rate (CTR) model is highly prohibitive. A recent promising direction for reducing such costs is data distillation with gradient matching, which aims to synthesize a small distilled dataset to guide the model to a similar parameter space as those trained on real data. However, there are two main challenges to implementing such a method in the recommendation field: (1) The categorical recommended data are high dimensional and sparse one- or multi-hot data which will block the gradient flow, causing backpropagation-based data distillation invalid. (2) The data distillation process with gradient matching is computationally expensive due to the bi-level optimization. To this end, we investigate efficient data distillation tailored for recommendation data with plenty of side information where we formulate the discrete data to the dense and continuous data format. Then, we further introduce a one-step gradient matching scheme, which performs gradient matching for only a single step to overcome the inefficient training process. The overall proposed method is called Categorical data distillation with Gradient Matching (CGM), which is capable of distilling a large dataset into a small of informative synthetic data for training CTR models from scratch. Experimental results show that our proposed method not only outperforms the state-of-the-art coreset selection and data distillation methods but also has remarkable cross-architecture performance. Moreover, we explore the application of CGM on model retraining and mitigate the effect of different random seeds on the training results.

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*RecSys '23, September 18–22, 2023, Singapore, Singapore*
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0241-9/23/09…$15.00
https://doi.org/10.1145/3604915.3608769

## CCS CONCEPTS

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

## KEYWORDS

Recommendation System; Dataset Distillation; Data Efficient Training; CTR Prediction.

**ACM Reference Format:**
Cheng Wang, Jiacheng Sun, Zhenhua Dong, Ruixuan Li, and Rui Zhang. 2023. Gradient Matching for Categorical Data Distillation in CTR Prediction. 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/10.1145/3604915.3608769

## 1 INTRODUCTION

Click-through rate (CTR) has been widely used by content providers for their advertising services, such as Google [3], YouTube [6], and TikTok [17]. 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.

Modern commercial advertising or shopping systems are notoriously data-hungry, i.e., they require massive datasets to learn powerful representations. At the same time, due to the dynamic nature of item features like popularity and user preferences, it is critical for a CTR prediction model to fit the newest user-item preferences [2, 19, 36]. In such circumstances, an offline CTR prediction model requires frequent retraining on a large amount of recent data [35, 41], which has become very computationally expensive. Hence, there is a considerable demand that can lower the computational cost of training models on the proxy dataset with a minimal performance drop. To this end, a straightforward idea to lower the training cost is from the data-efficient perspective, which aims to select a representative subset without significantly degrading the overall performance. Coreset selection [8, 9, 26], as a traditional solution to reduce the data size, can sample representative data based on some heuristic criteria. Unfortunately, the coreset selection methods have the following deficits when applicated to categorical recommendation data: (1) Unlike classification data, where each class of data has a similar distribution space, categorical recommendation data

consists of numerous side information. This multimodal data adds many dimensions to sample data, making it more difficult to construct meaningful samplers. (2) Categorical recommendation data is well-known for its sparsity, long-tail distribution, and missing-not-at-random. Sampling on such skewed data will result in additional problems, especially in the worse case in the proxy-based sampling strategy [26] might even exacerbate existing biases in the data or even aberrant data samples. (3) Most coreset selection methods are shortsighted in their incremental and greedy selection strategy. This strategy may lead to a worse situation when the information of a dataset is uniformly distributed over all of them.

To address the aforementioned shortcomings, instead of selecting representative samples from the original data, an alternative method is to synthesize informative samples based on the overall original data [40]. In particular, Data Distillation [32], as the recent data efficiency training method, is the task of "distilling " a small synthetic dataset such that a model trained on the synthetic set will match the test accuracy of the model trained on the full dataset (as shown in Figure 1(a)). The core idea of data distillation is to synthesize informative data from random noise such that the model trained on these synthetic data can minimize the training loss over the original data. In particular, DC [40] is one of the representative methods, which formulates the distillation goal as matching the gradient of the model parameters between small synthetic data and large real data. It has a remarkable effect on reducing the data size of image datasets with minor performance drops. However, directly applying existing data distillation [1, 15, 38, 40] methods to categorical recommendation data has several challenges. First, recommendation data are high-dimension and sparse one- or multi-hot data, which will block the gradient flow, causing backpropagation-based data distillation to be invalid. Second, the class-wise data split strategy of traditional data distillation intrinsically assumes that the data in the same class have a similar distribution. Such an assumption is inappropriate for recommendation data. Third, most traditional data distillation methods involve a complex bi-level optimization which is computationally expensive. To this end, we first 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 data format can directly transfer to different embedding-based CTR models for training. Then, we find that the class-wise data split is inappropriate for recommendation data and introduce an easy but effective random-wise data split strategy. Finally, we propose a one-step gradient matching scheme to overcome the inefficient bi-level optimization and remove the burden of tuning hyper-parameters. We name the overall method as Categorical data distillation with Gradient Matching (CGM). The experiment results show that our CGM can distill prosperous information from prior training data into a tiny synthetic dataset (3-4 orders of magnitude). In a nutshell, this work makes the following main contributions:

- We study a novel problem of learning continuous synthetic data for distilling discrete recommendation data. These synthetic data can effectively learn a CTR model from scratch and outperform the state-of-the-art (SOTA) coreset selection and data distillation methods.
- We propose a new form of data for our synthetic data to be optimized in the gradient matching process rather than the discrete one- or multi-hot data. Based on that, we introduce a one-step gradient matching scheme for learning more efficiently.
- Extensive experiments show that our method has good generality, which can implement in different deep recommendation models. At the same time, the synthetic data has remarkable cross-architecture performance. Finally, we demonstrate some promising applications of CGM - (1) model retraining scenario, (2) To assist the training process more stable with our synthetic data.

## 2 RELATED WORK

### 2.1 CTR Prediction

CTR prediction is the essential solution for the online advertising system. Most existing approaches consider CTR prediction as binary classification problems [5, 42]. It estimates the probability of exhibited items with or without click response by a user. The evolution of the CTR model has experienced blooming from classic machine learning, such as Logistic Regression (LR) [33] and Factorization Machine (FM) [24], to deep learning, which is capable of automatically learning feature interaction. Wide&Deep [3] was first productionized and evaluated the system on Google Play, which jointly trained wide linear models and deep neural networks. A recent work, DeepFM [10], learns low- and high-order feature interactions from wide component-FM and deep component-neural networks. To overcome the inefficient and implicit feature interactions in the deep neural network, DCN [31] proposed a novel cross-network to explicitly applies feature crossing. In this paper, We use these three widely used baselines as our base model to verify our GCM approach's generality, which can theoretically be applied to any embedding-based recommendation models.

### 2.2 Data Distillation

Data distillation was first proposed by Wang et al. [32], which is a related but orthogonal task to Knowledge Distillation [4]. Unlike classical data compression, data distillation aims for a small task-related synthetic dataset so that models trained on it can generalize to unseen test data with similar performance as training on the original dataset. To this end, most works formulate this problem as the optimal matching problem that narrows the parameter distance between the model trained on synthetic data and trained on real large data. [1, 15, 38, 40]. Cazenavette et al. [1] propose to optimize distilled data to a similar parameter state as those expert parameter trajectories trained on real data. DM [39] tries formulating the data distillation as feature distributions of the synthetic and original training set. To overcome the fine-grained classification, DCC [15] utilizes contrastive signals to capture the differences between classes effectively.

The above works are all based on image data which is low-dimensional and dense data. The most recent work [25] proposes Distill-CF to condense large categorical data into small synthetic data. However, Distill-CF only works with linear models ($\infty$-AE) and can only distill the collaborative filtering data which is the

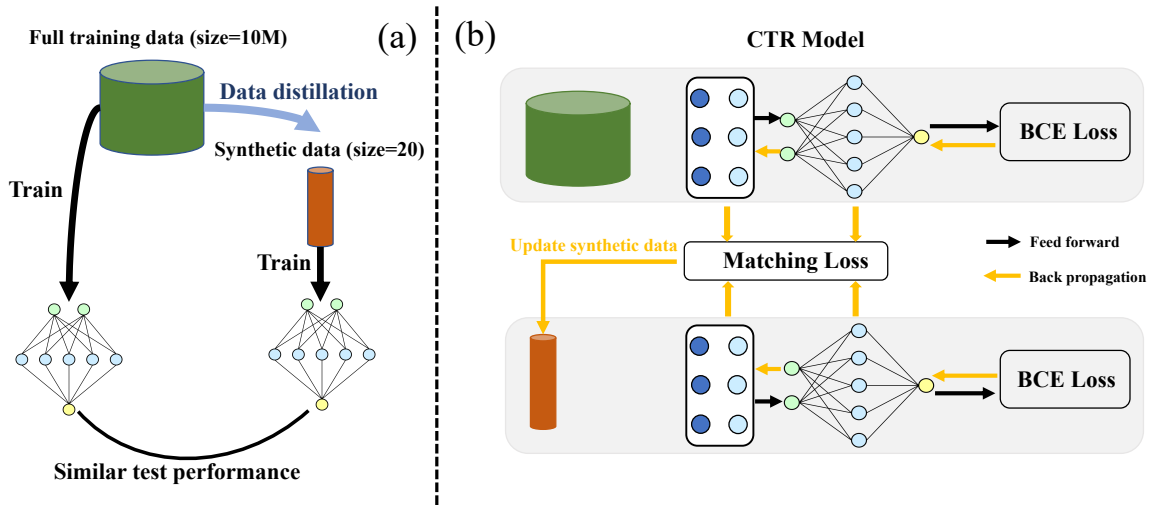


**Figure 1: (a): Data distillation , (b) :Illustration of CGM**

triple form (users, items, ratings) of RS data. Our main contribution is a general framework that can fit different embedding-based recommendation models and distill data with extra categorical side information. The work DC [40] is similar to ours, which also uses gradient matching to optimize synthetic data. However, DC is class-wise gradient matching based on the similarity of distribution spaces between the same class of image data. The categorical data with much side information cannot be distinguished only by labels.

## 2.3 Coreset Selection

Coreset selection is a long-standing learning problem that attempts to choose a subset of the most informative training samples that can benefit many downstream tasks such as data-efficient learning, continual learning, neural architecture search, active learning, etc [20, 22, 23]. Some classical data selection methods [8, 27, 34] focus on selecting the closest samples to the cluster center. Other lines of research aim to understand which training samples are 'valuable' for modern machine learning. Some literature selects coreset based on the assumption that some samples contribute more to the error or loss when training models [22]. A similar idea like FORGETTING EVENT [29] counts how often the "forgetting" happens during the training. Some works have recently used proxy functions to select data points. For example, SVP-CF [26] uses matrix factorization [14] as a proxy to select a representative collaborative filtering subset. Although coreset selection methods can be very computationally efficient, they have two major limitations. First, most methods incrementally and greedily select samples rely on some heuristics, which do not guarantee any optimal solution for the downstream task. Second, their efficiency is upper bound by the information in the selected samples in the original dataset.

# 3 PRELIMINARIES

Suppose we are given a large recommendation data consisting of $|\mathcal{T}|$ pieces of training data $\mathcal{T} = (\mathcal{X}, y)$, where $\mathcal{X}$ is a $m$-fields data records that may include categorical fields (e.g., item brand, location) and numeric fields (e.g., user age). $y \in \{0, 1\}$ is the associated label indicating user click responses ($y = 1$ means the user clicked the item, and $y = 0$ otherwise). Then, each instance is converted to $(x, y)$ where $x = [x_1, x_2, ..., x_m]$ can be represented as the concatenation of binary vectors from all fields:

$$x = \underbrace{[1, 0, ..., 0]}_{x_1:userid} \underbrace{[1, 0]}_{x_2:gender} \underbrace{[0, 1, 0, 0]}_{x_3:age} \overbrace{.....}^{otherfield} \underbrace{[0, 1, ..., 0]}_{x_m:itemid}, \quad (1)$$

where $m$ is the number of feature fields, $x_i$ is the vector representation of the $i$-th field of $\mathcal{X}$. The categorical data are transformed into binary vectors via one-hot encoding, e.g., $[0, 1]$ for $gender = Female$ and $[1, 0]$ for $gender = Male$. The numeric data are first partitioned into buckets, and then we have a binary vector for each bucket, e.g., we can use $[0, 0, 0, 1]$ for a child whose $age \in [0, 14]$, $[0, 0, 1, 0]$ for a teenager whose $age \in [15, 24]$, $[0, 1, 0, 0]$ for adult whose $age \in [25, 64]$, and $[1, 0, 0, 0]$ for seniors whose $age \geq 65$.

In real-world CTR tasks, the input instances $x$ are discrete high-dimensional and extremely sparse. A common practice is to map each field into low-dimensional latent space (i.e., embeddings) as:

$$e_i = x_i V_i, \quad (2)$$

where $V_i \in R^{u_i \times k}$ denotes the embedding matrix of field $i$. $u_i$ is the number of features in the $i$-th feature field, and $k$ is predefined dimension of embedding vectors. Then the feature embeddings of instance $x$ is $e = [e_1, e_2, ..., e_m]$, which will be adopted in CTR models for prediction.

CTR prediction is to learn the parameters of the function $f$ over data $\mathcal{T}$ with parameters $\theta$ to minimize the following objective function:

$$\theta^{\mathcal{T}} = \arg\min_{\theta} \mathcal{L}^{\mathcal{T}}(\theta), \quad (3)$$

where $\mathcal{L}^{\mathcal{T}}(\theta) = \frac{1}{|\mathcal{T}|}\sum_{(x,y)\in\mathcal{T}}\ell(f_\theta(x), y)$, $\ell(\cdot,\cdot)$ is the binary cross-entropy loss and $\theta^{\mathcal{T}}$ is the minimizer of $\mathcal{L}^{\mathcal{T}}$.

# 4 CATEGORICAL DATA DISTILLATION WITH GRADIENT MATCHING

In this section, we introduce the details of our proposed Categorical data distillation with Gradient Matching (CGM). First, we overview the traditional gradient matching and discuss the existing problems that directly implement in recommendation fields. Then, we propose a new data format to overcome the difficulty brought by the categorical data and illustrate the forward and backpropagation mechanism with the new data format. Finally, we introduce a one-step gradient matching strategy to overcome the bi-level optimization and remove the burden of tuning hyper-parameters, such as the number of outer/inner iterations of the bi-level optimization as required by DC. The overall architecture of our proposed CGM is illustrated in Figure 1(b).

## 4.1 Overview of Traditional Gradient Matching

To distill a tiny synthetic set $\mathcal{S}$ that model train on them has comparable performance than train on original data $\mathcal{T}$ is as follows:

$$\mathbb{E}_{x\sim P_{test}}[\ell(f_{\theta^{\mathcal{T}}}(x), y)] \simeq \mathbb{E}_{x\sim P_{test}}[\ell(f_{\theta^{\mathcal{S}}}(x), y)]. \tag{4}$$

Where $p_{test}$ is the test data distribution. $f_{\theta^{\mathcal{T}}}$ and $f_{\theta^{\mathcal{S}}}$ denote the model with parameters $\theta^{\mathcal{T}}$ and $\theta^{\mathcal{S}}$ respectively. A promising solution to achieve comparable performance in Equation 4 is to allow the model trained on synthetic data $\mathcal{S}$ to imitate the parameters on the original data $\mathcal{T}$ (i.e., $\theta^{\mathcal{T}} \approx \theta^{\mathcal{S}}$). It is because similar parameters indicate similar mappings in a local neighbourhood and thus generalization performance. Therefore, some work [1, 40] formulate the goal of data distillation to a class-wise *curriculum gradient matching* problem such that model $f_{\theta^{\mathcal{S}}}$ trained on synthetic data not only have comparable performance to model $f_{\theta^{\mathcal{T}}}$ but also converges to similar parameter space. The *curriculum* means the parameter distance w.r.t large-real data training and small-synthetic data training will be close at each training epoch. We denote the model parameters at $t$-th epoch as $\theta_t$ and $f_{\theta_t}$ indicate the model parameterized by $\theta_t$. The curriculum gradient matching is expressed as:

$$\min_{\mathcal{S}} \sum_{t=0}^{T-1}\sum_{c=0}^{C-1} D(\nabla_\theta \ell(f_{\theta_t}(\mathcal{S}_c), \mathcal{Y'}_c), \nabla_\theta \ell(f_{\theta_t}(\mathcal{T}_c), \mathcal{Y}_c)), \quad s.t. \quad \theta_{t+1} = opt_\theta(\theta_t, \mathcal{S}), \tag{5}$$

where $T$ is the number of training epochs, and $C$ is the whole class of original data. $D(\cdot,\cdot)$ is the distance function and $\mathcal{Y'}$ indicate the label of synthetic data. We denote $opt_\theta(\theta_t, \mathcal{S})$ as the optimization operator for updating parameter $\theta$ on synthetic data $\mathcal{S}$. To generate synthetic data $\mathcal{S}$ that can perform well within a distribution of random initialization $P_{\theta_0}$, we sample $\theta_0 \sim P_{\theta_0}$ and modify Equation 5 as following:

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0\sim P_{\theta_0}}[\sum_{t=0}^{T-1}\sum_{c=0}^{C-1} D(\nabla_\theta \ell(f_{\theta_t}(\mathcal{S}_c), \mathcal{Y'}_c), \nabla_\theta \ell(f_{\theta_t}(\mathcal{T}_c), \mathcal{Y}_c))], \quad s.t. \quad \theta_{t+1} = opt_\theta(\theta_t, \mathcal{S}). \tag{6}$$

**Discussion.** The Equation 6 has demonstrated promising performance on distilling image datasets. However, the discrete recommendation data will block the gradient flow, causing backpropagation training to be invalid. Moreover, Equation 5 is a bi-level optimization problem where the synthetic data $\mathcal{S}$ is learned at the outer optimization and model parameters $\theta_t$ are learned at inner optimization, which will inevitably hinder the scalability on the real large-scale CTR prediction datasets.

## 4.2 Learning Discrete Recommendation Data with New Data Format

The standard approach to solving Equation 6 is to optimize the synthetic data $\mathcal{S}$ by narrowing the parameter distance between $\theta^{\mathcal{S}}$ and $\theta^{\mathcal{T}}$ in each iteration. However, the categorical recommendation data are discrete and sparse one- or multi-hot data, rendering backpropagation-based gradient matching useless [30]. Therefore, we reconstruct the data as continuous tensors so that our synthetic data can successfully obtain the gradient. At the same time, considering the generality, the new data format should adapt to different deep CTR models. To be more specific, the dimension of each new data format is identical to the one-hot representation of the original real data. The form of initial synthetic data can be recognized as:

$$s = \underbrace{[\omega_1^1, \omega_1^2, ..., \omega_1^{u_1}]}_{x_1:userid}\underbrace{[\omega_2^1, \omega_2^2]}_{x_2:gender}\overbrace{.....}^{otherfield}\underbrace{[\omega_m^1, \omega_m^2, ..., \omega_m^{u_m}]}_{x_m:itemid}, \quad s.t. \quad \text{For any field-}i, \sum_{j=1}^{u_i}\exp(\omega_i^j) = 1, \omega \sim U(0,1). \tag{7}$$

Where each $\omega$ is sampled from the uniform distribution for initialization. It can be seen that a data instance from $\mathcal{T}$ is the special case of our new data format (iff for any $i$-th field $\omega_i^j = 1$ and $\omega_i^{j'} = 0$ for any $j' \neq j$). We limit the sum of the randomly initialized synthetic data in each field to 1 because we try to union the synthetic data and original data on the same scale. Such normalization can stabilize the update of synthetic data.

**Feed-forward and back-propagation mechanism of the new data format.** To allow the CTR model to be trained directly on the synthetic data without changing the model architecture, each field of the synthetic data is first mapped to low-dimensional embedding vectors as follows:

$$e_i^{'} = s_i V_i. \tag{8}$$

Unlike original data $\mathcal{T}$, each field is mapped into $e_i \in R^{1\times k}$ as in Equation 2, each field in the new data format $\mathcal{S}$ is mapped into $e_i^{'} \in R^{u_i\times k}$. The feature embeddings of one piece of synthetic data are $e^{'} = [e_1^{'}, e_2^{'}, \ldots, e_m^{'}]$ which cannot directly be used in

**Algorithm 1** CGM for distilling Recommendation Data

**Input:** Training data $\mathcal{T}$
**Output:** Small batch of $\mathcal{S}$

Pre-set Chunks $B$, $N$ number of synthetic data for each block, the probability distribution of initialized weights $P_{\theta_0}$, a random CTR model $f_\theta$, number of epochs $K$, learning rates for updating synthetic samples $\eta_{\mathcal{S}}$.
Randomly split data $\mathcal{T}$ into $B$ blocks $(\mathcal{T}_1, ..., \mathcal{T}_B)$
Initialize $N * B$ synthetic data $\mathcal{S}$ with Equation 6
**for** $k \leftarrow 0$ to $K$ **do**
  Initialize model parameters $\theta_0 \sim P_{\theta_0}$
  **for** $b \leftarrow 0$ to $B$ **do**
    Fetch data $\mathcal{T}_b$ and $\mathcal{S}_b$
    compute $\mathcal{L}^{\mathcal{T}_b} = \frac{1}{|\mathcal{T}_b|} \sum_{(x,y)\in\mathcal{T}_b} \ell(f_{\theta_t}(x), y)$
    compute $\mathcal{L}^{\mathcal{S}_b} = \frac{1}{N} \sum_{(s,y')\in\mathcal{S}_b} \ell(f_{\theta_t}(s), y')$
    $\mathcal{S}_b \leftarrow \nabla_{\mathcal{S}_b}(D(\nabla_{\theta_0}\mathcal{L}^{\mathcal{S}_b}, \nabla_{\theta_0}\mathcal{L}^{\mathcal{T}_b}))$
  **end for**
**end for**

the following feed-forward training. Hence, in order to make the dimension of $e_i^{'}$ consistent with $e_i$, we compress each field's feature embedding matrix $e_i^{'} \in R^{u_i \times k}$ to dimension $1 \times k$ as:

$$agg(e_i^{'}) = \sum_{u_i} s_i V_i. \tag{9}$$

After that, the training process of feature embedding of a synthetic data instance $e^{'}$ is the same as the original data's feature embedding. At the same time, the weight $\omega$ of synthetic data $s$ can be updated by backpropagation:

$$(\omega_i^j)_{t+1} = (\omega_i^j)_t - \eta_{\mathcal{S}} \nabla \ell(f_{\theta_t}(s, y')), \tag{10}$$

where $t$ denotes the $t$-th iteration and $\eta_{\mathcal{S}}$ is the learning rate of synthetic data.

## 4.3 One-step Gradient Matching with Random-wise Data Split

According to Equation 6, the gradient calculated on synthetic data $\mathcal{S}_c$ and original data $\mathcal{T}_c$ is with the same class separately. The intuition is image data with the same class that imply similar data distribution. This will help the training process become more stable. However, such class-wise gradient matching strategy is inappropriate for categorical recommendation data. Hence, we try the following three data split strategies: (1) class-wise, (2) user-wise, (3) random-wise. The best result is the random-wise data split strategy which randomly splits original data into $B$ blocks, and each block corresponds to one or several synthetic data. Then, the gradient matching goal is to reduce the gradient distance between the model trained on synthetic data $\mathcal{S}_b$ and the model trained on original data $\mathcal{T}_b$ as follows:

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0\sim P_{\theta_0}}[\sum_{t=0}^{T-1}\sum_{b=0}^{B-1} D(\nabla_\theta \ell(f_{\theta_t}(\mathcal{S}_b), \mathcal{Y'}_b), \nabla_\theta \ell(f_{\theta_t}(\mathcal{T}_b), \mathcal{Y}_b))], \quad s.t. \quad \theta_{t+1} = opt_\theta(\theta_t, \mathcal{S}). \tag{11}$$

The synthetic data $\mathcal{S}_b$ can be recognized as a highly condensed version of the original data $\mathcal{T}_b$. The vanilla gradient matching presents a nested optimization, which optimizes synthetic data $\mathcal{S}$ at the outer loop and the network parameter $\theta_t$ at the inner loop. The nested loop severely hinders the scalability of the distillation method, which motivates us to propose a new scheme for efficient distillation. We introduce *one-step* gradient matching where we only match the gradient on the model initializations $\theta_0$ while ignoring the subsequent training trajectory of $\theta_t$. Specifically, instead of approximating the overall gradient matching loss on $T$ iterations, we only adopt the initial matching loss at the first epoch. We drop the summation symbol $\sum_{t=0}^{T-1}$ and modify the Equation 11. **The final optimized objective of proposed CGM is**:

$$\min_{\mathcal{S}} \mathbb{E}_{\theta_0\sim P_{\theta_0}}[\sum_{b=0}^{B-1} D(\nabla_\theta \ell(f_{\theta_0}(\mathcal{S}_b), \mathcal{Y'}_b), \nabla_\theta \ell(f_{\theta_0}(\mathcal{T}_b), \mathcal{Y}_b))]. \tag{12}$$

The intuition is that we assume the largest gradient gap happens at 0-th epoch, and the initial matching loss at the first epoch sufficiently informs us about the direction to update the synthetic data [13]. Experimental results show the effectiveness of one-step gradient matching in Section 5.3.

**The Distance Function.** The matching loss $D(\cdot,\cdot)$ in Equation 12 measures the gradient between $\mathcal{L}^{\mathcal{S}}$ and $\mathcal{L}^{\mathcal{T}}$. Suppose we have a vanilla deep CTR model which can be decomposed to embedding matrix layer and multilayer perception (MLP). The matching loss is the sum of the layerwise loss as:

$$D(\nabla_\theta \mathcal{L}^{\mathcal{S}}, \nabla_\theta \mathcal{L}^{\mathcal{T}}) = \sum_{l=1}^{L}(1 - \frac{\nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{S}} \odot \nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{T}}}{||\nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{S}}||||\nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{T}}||}). \tag{13}$$

Where $l$ is the layer index, $L$ is the number of layers. The symbol $\odot$ is the element-wise product. Experimental results show that this is a better distance for gradient matching and enables using a single learning rate across all layers.

## 4.4 CGM Algorithm

The synthetic data optimization algorithm is depicted in Algorithm 1. Recent data distillation methods [40] are class-wise, i.e., when generating a small synthetic data for class "a" ($\mathcal{S}_a$), these methods use only the training data of class "a" ($\mathcal{T}_a$) to match gradients. For categorical recommendation data, we randomly (ignore what the label is) and evenly (each block has the same data size) split the data $\mathcal{T}$ into $B$ blocks. We match gradients and update synthetic data for each block separately in order to make matching easier. Specifically, we initialize the model parameters $\theta_0$ for $K$ times to perform one-step gradient matching. For block $b$, we first retrieve the synthetic data of that block, denoted as $(\mathcal{S}_b, \mathcal{Y'}_b) \sim \mathcal{S}$, and retrieve all original data $(\mathcal{T}_b, \mathcal{Y}_b)$ from that block. Then, we forward these data to the CTR prediction models and calculate the one-step gradient matching loss with distance function $D(\nabla_{\theta_0}\mathcal{L}^{\mathcal{S}}, \nabla_{\theta_0}\mathcal{L}^{\mathcal{T}})$. Note that the matching process for each block can be run in parallel since the model updates for one block is independent of other blocks.

**Table 1: Statistics of Criteo, Avazu, and Movielens datasets**

| Dataset | Criteo | Avazu | Movielens |
|---|---|---|---|
| #. of features (C + N) | 22 + 14 | 21 + 0 | 2 + 0 |
| #. of total records | 13.8M | 12.1M | 10M |
| #. of distinct features | 428.4K | 653.1K | 82.2K |

## 5 EXPERIMENTAL RESULTS

We conduct experiments in this section to evaluate the performance of our proposed CGM. The majority of our experiments will focus on the following research questions: **RQ1:** Can our data distillation approach effectively condense the training data information into a small set of informative synthetic data? How does the performance of our proposed CGM compare with the baselines? **RQ2:** How is the cross-architecture performance of the synthetic data generated by CGM? **RQ3:** How is the performance of proposed tricks such as data split strategy, distance function and one-step gradient matching? **RQ4:** What is the potential application of our informative synthetic data?

**Datasets and metrics.** We evaluate CGM on three publicly available datasets, namely Avazu[1], Criteo[2], and Movielens10M[3]. The details of the data statistic are depicted in Table 1. Specifically, we convert the 14 numerical feature fields of Criteo to categorical. Criteo and Avazu are online advertisement competition datasets on Kaggle, which contain chronologically ordered click-through records. We use their top 30% records for our experiments [16]. Movielens10M 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. Table 1 shows the statistics of the datasets. The ratio of train, test, and validation set sizes is 8:1:1.

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.

**Implementation Details** We conduct all experiments with PyTorch [21] on a single RTX 5000 GPU. The source code[4] of the publicly accessible library is modified to fit our proposed method. In this paper, we select three Deep CTR models (i.e., DCN, DeepFM, and Wide&Deep) to implement our CGM. In each model, we use three hidden layers (100-100-100) MLP with 10 dimensions of embedding and ReLU [18] activation for the MLP layer. We adopt early stopping when AUC scores do not improve in consecutive epochs on the validation set. We employ Binary Cross-entropy (BCE) as the optimizer. The learning rates are slightly different with respect to distinct models or datasets. We do a grain search on the learning rate of synthetic data $\eta = \{1e-5, 5e-5, 5e-6, 1e-6, 5e-7, 1e-7\}$. All experimental results are reported by the average performance of 10 runs.

[1]https://www.kaggle.com/c/avazu-ctr-prediction
[2]http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/
[3]https://grouplens.org/datasets/movielens/
[4]https://rixwew.github.io/pytorch-fm/

**Baselines.** To evaluate the generalization of the proposed CGM, we implement it on three different architectures of CTR prediction models. (1) Deep & Cross Network (DCN) ensemble CrossNet and Deep Neural Network (DNN) [31]. (2) Deep Factorization Machine (DeepFM) [10] ensemble Factorization Machine and DNN. (3) Wide&Deep [3] integrates Logistic Regression with DNN. For a fair comparison, we omitted the hand-crafted cross features. After selecting the implemented models, we compare our CGM with the following baselines:

- **Random**: We randomly select data from training samples as coreset.
- **K-center** [27]: It picks multiple center points such that the largest distance between a data point and its nearest center is minimized. We use K-prototype [11] algorithm to calculate the distance of categorical data. To extract features and compute the distance to centers, we employ models trained on the whole dataset.
- **SVP-CF** [26]: It is a proxy-based coreset sampling stragety for collaborative filtering datasets. We use Factorization Machine (FM) as the proxy model to calculate the "forgetting event" as in SVP-CF. At the same time, the variation (SVP-CF-Prop) of SVP-CF will not be considered here since it is not suitable for data with many categorical and numerical side information.
- **Distill-CF** [25]: A data distillation method for creating tiny, high-fidelity data summaries of large datasets. It adopts Gumbel sampling [12] to distill semi-structured and sparse collaborative filtering data. The synthetic data from Distill-CF is a continuous user-item rating matrix. Each synthetic user has a continuous type of rating for each item.

For all strategies, we exclusively sample/distill on the train set and never touch the validation and test sets. Note that to keep comparisons fair, the synthetic data from Distill-CF is converted to user-item-rating pairs for training on DCN model instead of the $\infty$-AE model.

### 5.1 Performance Comparison (RQ1)

To validate the effectiveness of the CGM framework, we measure the AUC score and LogLoss of CTR prediction models trained on our synthetic data. We compare our CGM to the selected baselines on Avazu, Criteo, and Movielens10M in Table 2, and data generated from baselines are trained on the DCN model which has the best performance among DeepFM and Wide&Deep. Specifically, the selection-based baselines use data sizes of 10K, 20K, and 30K to compare distillation-based methods with data sizes of 5, 20, and 30 condensed data (three orders of magnitude) generated from ours or Distill-CF. We name the distilled data of our framework as CGM-DCN while using DCN as the base model for gradient matching. Similarly, we also have CGM-DFM and CGM-Wide&Deep. The whole data indicates model training on the entire raw training set, which serves as an approximate upper-bound performance. From the table, we have the following observations: (1) Our method outperforms all the baselines significantly, and even the AUC scores on 5 distilled data are better than baselines under the data size of 30K in some cases. At the same time, the best result of Movielens10M dataset is CGM-DeepFM which can achieve 95.13% upper-bound

**Table 2: Performance comparisons of CGM and baseline methods on AUC and LogLoss. We report the mean results over ten runs. The notation - denotes the selected baseline that can not implement on the dataset.**

| Datasets | Metrics | Data Size | Selection-based | | | Distillation-based | | | | Whole Data |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Random | K-center | SVP-CF | Distill-CF | CGM-DCN | CGM-DeepFM | CGM-Wide&Deep | |
| Avazu | AUC | 10K vs. 5 | 0.6766 | 0.6254 | 0.6766 | - | **0.7124** | 0.6951 | 0.6912 | 0.7635 |
| | | 20K vs. 20 | 0.6986 | 0.6319 | 0.6994 | - | **0.7323** | 0.7301 | 0.7085 | |
| | | 30K vs. 30 | 0.7038 | 0.6422 | 0.7041 | - | 0.7283 | **0.7313** | 0.7066 | |
| | LogLoss | 10K vs. 5 | 0.4382 | 0.9238 | 0.4453 | - | **0.4359** | 0.562 | 0.9356 | 0.3651 |
| | | 20K vs. 20 | 0.4181 | 0.9215 | 0.4187 | - | **0.4119** | 0.4365 | 0.8589 | |
| | | 30K vs. 30 | 0.4162 | 0.9186 | 0.4181 | - | **0.4154** | 0.4196 | 0.5936 | |
| Criteo | AUC | 10K vs. 5 | 0.7049 | 0.6352 | 0.7016 | - | 0.7252 | 0.7224 | **0.7337** | 0.7887 |
| | | 20K vs. 20 | 0.7119 | 0.6433 | 0.7122 | - | 0.7505 | **0.7512** | 0.7344 | |
| | | 30K vs. 30 | 0.7202 | 0.6576 | 0.7217 | - | 0.7533 | **0.7562** | 0.7386 | |
| | LogLoss | 10K vs. 5 | 0.5279 | 1.3243 | 0.5394 | - | 0.5412 | 0.6177 | **0.5135** | 0.4504 |
| | | 20K vs. 20 | 0.5422 | 1.2716 | 0.5387 | - | 0.5259 | **0.5113** | 0.6924 | |
| | | 30K vs. 30 | 0.5547 | 1.2446 | 0.5533 | - | 0.5085 | **0.5012** | 0.5296 | |
| Movielens | AUC | 10K vs. 5 | 0.6298 | 0.5733 | 0.6313 | 0.7234 | 0.7483 | **0.7529** | 0.7446 | 0.8136 |
| | | 20K vs. 20 | 0.6447 | 0.5737 | 0.643 | 0.7357 | 0.7634 | **0.7712** | 0.7609 | |
| | | 30K vs. 30 | 0.6592 | 0.5865 | 0.6613 | 0.7469 | 0.7685 | **0.774** | 0.7616 | |
| | LogLoss | 10K vs. 5 | 1.0511 | 1.4757 | 0.9981 | 0.7311 | 0.6847 | **0.6653** | 0.7321 | 0.5136 |
| | | 20K vs. 20 | 1.0326 | 1.3657 | 1.2704 | 0.6923 | 0.6491 | 0.6352 | **0.6358** | |
| | | 30K vs. 30 | 0.9959 | 1.7253 | 1.1657 | 0.6845 | 0.6476 | **0.6361** | 0.6454 | |

**Table 3: Cross-architecture performance for condensed 20 synthetic data in Avazu, Criteo and Movielens10M.**

| CGM variants | Avazu | | | Criteo | | | Movielens10M | | |
|---|---|---|---|---|---|---|---|---|---|
| | DCN | DeepFM | Wide&Deep | DCN | DeepFM | Wide&Deep | DCN | DeepFM | Wide&Deep |
| CGM-DCN | **0.7323** | 0.6514 | 0.7321 | **0.7505** | 0.7267 | 0.7493 | **0.7634** | 0.7636 | 0.7621 |
| CGM-DeepFM | 0.7282 | **0.7301** | 0.7136 | 0.7486 | **0.7512** | 0.7484 | 0.7613 | **0.7712** | 0.7638 |
| CGM-Wide&Deep | 0.7097 | **0.7107** | 0.7085 | **0.7379** | 0.7229 | 0.7344 | 0.7363 | 0.7401 | **0.7609** |

**Table 4: Ablation study on distinct data split strategies.**

| Split strategies | CGM-DCN AUC/LogLoss | CGM-DeepFM AUC/LogLoss | CGM-Wide&Deep AUC/LogLoss |
|---|---|---|---|
| user-wise | 0.7379/0.6572 | 0.7369/0.6365 | 0.7021/0.6665 |
| class-wise | 0.7132/0.6877 | 0.7091/0.6945 | 0.6934/0.7015 |
| random-wise | **0.7634/0.6491** | **0.7712/0.6352** | **0.7609/0.6358** |

**Table 5: Ablation study on distinct distance functions.**

| Distance functions | CGM-DCN AUC/LogLoss | CGM-DeepFM AUC/LogLoss | CGM-Wide&Deep AUC/LogLoss |
|---|---|---|---|
| Euclidean | 0.6573/0.7245 | 0.6368/0.7371 | 0.6121/0.7590 |
| Cosine | 0.7231/0.6903 | 0.7121/0.7015 | 0.7184/0.6968 |
| ours | **0.7634/0.6491** | **0.7712/0.6352** | **0.7609/0.6358** |

performance with only 30 pieces of data. (2) Increasing the number of synthetic data can further improve the overall performance. It is because more synthetic data indicates more learnable information preserved in the original data (Further discussion about the performance of synthetic data scale with data size is in Section 5.3). The performance gap between CGM and the upper bound is bigger on Criteo and Avazu datasets, which contain more side information. (3) For the coreset selection baselines, we also observe that SVP-CF has no significant improvement over Random, and K-center has the worst performance. Like in [26], the random coreset method is still a strong baseline. There may be two rational reasons. First, unlike in CV, the categorical information cannot distinguish the data distribution of RS data, which leads to poor performance of K-center. Second, the core idea of SVP-CF is to select the samples which cause a large loss (hard sample). However, recent work [28] claims that selecting samples that cause a small loss is sometimes better. (4) The data type of synthetic data from Distill-CF is also the continuous tensor, which can contain informative training data information with a small batch of distilled data. However, our data distillation framework has two advantages: (i) our synthetic can be generalised to different models for training. (ii) our framework can distill more complex tabular data such as Avazu and Criteo.

## 5.2 Cross-architecture Performance of CGM (RQ2)

Another critical advantage of CGM is that the distilled data learned from one model can be used to train another unseen one. Here we learn 20 pieces of distilled data for Avazu, Criteo, and Movielens10M over a diverse set of models, including DCN, DeepFM, and Wide&Deep. As depicted in Table 3, we train every model on all corresponding sets separately from scratch and evaluate their cross-architecture performance in terms of AUC scores on the Avazu,

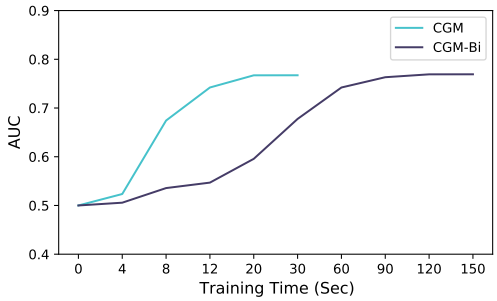


**Figure 2: One-Step vs. Bi-Level matching**

Criteo, and Movielens10M test set. Table 3 shows that all the distilled data perform well in different models or datasets. The best performance of DCN and DeepFM is the distilled data synthesized from themself. Interestingly, CGM-DCN and CGM-DeepFM outperformed CGM-Wide&Deep, possibly due to the extra information provided by CrossNet or FM in the distilled data are more effective than the linear net in Wide&Deep model. The best results are obtained in most cases are CGM-DCN, which is in line with its performance when trained on the original dataset.

## 5.3 Ablation study (RQ3)

**Different data split strategies:** We study the performance of distinct raw data split strategies to evaluate the effectiveness of our random-wise data split strategy. In Algorithm 1, one of the steps is to split raw data into different chunks (line 2). For image/graph data distillation, the most commonly used data split strategy is class-wise which can stabilize the final gradient matching results. However, the recommendation data with the same class do not imply the property of similar data distribution. Hence, it is inappropriate to directly graft the class-wise data split strategy on the recommendation data. We try the following three strategies to find the most effective one: (1) user-wise strategy (allocating all rating data of a user to an identical block), (2) class-wise strategy, (3) random-wise strategy (Randomly selecting data into a randomly indicating block). As shown in Table 4, the random-wise split strategy outperforms others on different Deep CTR models, which means it is more effective and robust.

**Different distance function:** We also discuss the effectiveness and robustness of our distance function. Our distance function is the element-wise product of two gradient matrices in layer $l$ ($\nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{S}} \odot \nabla_{\theta^{(l)}}\mathcal{L}^{\mathcal{T}}$) without flattening the tensors over all layers into one vector. We compare with the Euclidean distance $||G^{\mathcal{T}}, G^{\mathcal{S}}||^2$ and Cosine distance $1 - cos(G^{\mathcal{T}}, G^{\mathcal{S}})$ $G^{\mathcal{T}}, G^{\mathcal{S}} \in R^D$ which vectorizes and concatenates the whole gradient [44], where $D$ is the number of all network parameters. We separately adopt the Euclidean distance and Cosine distance to distill raw data into 20 synthetic data on Movielens with three selected models. Table 5 shows that our distance function achieves a better performance.

**One-step vs. bi-level:** To evaluate the effectiveness of the one-step gradient matching strategy. We create an ablation of our proposed framework, namely CGM-Bi, which adopts the traditional gradient matching that involves a bi-level optimization. We compare the AUC scores and training time of CGM and CGM-Bi in the setting of learning 20 synthetic data on the DCN model with 10M Movielens datasets. As depicted in Figure 2, our one-step gradient matching strategy needs approximately 20 Sec to reach its performance, while the bi-level strategy needs almost 120 Sec. It demonstrates the efficiency of our proposed one-step gradient matching strategy.

**Table 6: The AUC scores and training time of different training data in model retraining scenario**

| Training Data | DCN | DFM | Wide&Deep |
|---|---|---|---|
| $raw_{T1} \cup raw_{T2}$ | 0.8143 | 0.8169 | 0.8129 |
| $raw_{T2}$ | 0.7931 | 0.7958 | 0.7940 |
| $random_{T1} \cup raw_{T2}$ | 0.7954 | 0.7971 | 0.7967 |
| $K\text{-}center_{T1} \cup raw_{T2}$ | 0.7952 | 0.7966 | 0.7960 |
| $SVP\text{-}CF_{T1} \cup raw_{T2}$ | 0.7956 | 0.7976 | 0.7979 |
| $Distill\text{-}CF_{T1} \cup raw_{T2}$ | 0.8023 | 0.8021 | 0.7988 |
| $CGM\text{-}DCN_{T1} \cup raw_{T2}$ | **0.8051** | **0.8057** | **0.8015** |
| $CGM\text{-}DeepFM_{T1} \cup raw_{T2}$ | 0.8049 | 0.8053 | 0.8012 |
| $CGM\text{-}Wide\&Deep_{T1} \cup raw_{T2}$ | 0.8048 | 0.8046 | 0.8011 |
| $raw_{T1} \cup raw_{T2}$ | 356.91s | 386.61s | 377.81s |
| $Coreset_{T1} \cup raw_{T2}$ | 238.32s | 259.72s | 243.41 |
| $CGM_{T1} \cup raw_{T2}$ | **207.91s** | **220.57s** | **211.62s** |

**Table 7: The mean and standard variance of AUC scores in 10 experiments with different random seeds.**

| Model Parameters | DCN | DFM | Wide&Deep |
|---|---|---|---|
| Random Initialization | 0.8159 ± 0.0018 | 0.8152 ± 0.0017 | 0.8125 ± 0.0016 |
| CGM Initialization | **0.8167±0.0006** | **0.8157±0.0008** | **0.8145±0.0007** |

**Scaling with different data size:** We study our synthetic data's absolute and relative upper-bound performance on selected three datasets by increasing the number of distilled data. With increasing the number of distilled data, the AUC improves in all models and further closes the gap with the whole datasets' upper-bound performance. However, After the data volume exceeds 30, the improvement of AUC becomes slow. The best AUC scores of our synthetic data to that of whole data differ by 5%∼6%. Interestingly, the relative performance gap on CGM-DCN and CGM-DeepFM does not decrease significantly for Avazu and Criteo datasets. However, the relative performance of CGM-Wide&Deep has declined more severely because the CrossNet in the DCN model and the FM in DeepFM can learn more feature interaction so that more informative information of raw data can be distilled into synthetic data. With the data scale increasing, AUC starts to drop slightly. It is because an excess of data will add redundant information to the process of gradient matching, which decreases model performance.

## 5.4 Case Studies about Applications of CGM (RQ4)

**Model Retraining:** Modern commercial CTR models collect overwhelming data daily. To fit the latest user-item preference, CTR models will frequently retrain the models with the latest collected data to provide timely personalization [37]. The former (historical) data can also benefit model performance since it gives a long-term preference signal. However, a full model retraining could be very time-consuming and memory-costly when the scale of historical

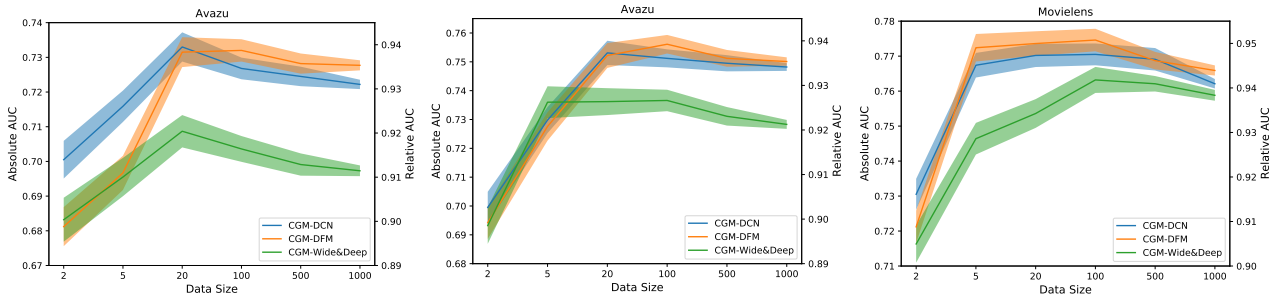


**Figure 3: Absolute and relative AUC scores for varying the number of distilled synthetic data on Avazu, Criteo, and Movielens**

data is large. Hence, it is vital to make a trade-off between high performance (more data for training) and real-time deployment (fewer data but reducing the training time). Instead of directly using all collected data, which will damage the real-time deployment, our method can distill the raw data into small yet efficient distilled data to improve model performance without significantly increasing training time. To simulate a real-world scenario, we split the MovieLens10M into three different time windows according to the timestamp. Following [37], the data from the time window one is adopted as historical data, time window two is the latest user behaviour directly used for model retraining, and the data from time window three is the test set for simulating the future preferences of users. We denote the original training data from the time window one as $raw_{T1}$, and distilled data synthesized from $raw_{T1}$ by the DCN model as $CGM - DCN_{T1}$. In this experiment, we compare with other baselines on the model retraining setting. The coreset selection methods sample data size of 10K from the time window one, and the distillation methods synthesize 10 synthetic data. We record the average retraining time on different datasets to illustrate the effectiveness of our proposed CGM. From Table 6, we have the following observations: (1) Our CGM can preserve informative training data information from the time window one's training data into a small batch of synthetic data. We no longer need to retrain the model with massive historical data. We only need to use CGM to synthesize a highly condensed version of historical data for data-efficient training. (2) In the model retraining procedure, although a full retraining with $raw_{T1} \cup raw_{T2}$ can get the best AUC scores, it needs the longest training time, which is intolerant for the timely model deployment. However, distilled data $CGM_{T1} \cup raw_{T2}$ for the retraining model slightly increase the training time while significantly improving model performance than the coreset selection methods with data $Coreset_{T1} \cup raw_{T2}$.

**Stable the training result:** In this part, we study the effect of distilled data on improving the resulting stability of CTR model training. In the CTR prediction task, a slight increase in AUC at .001-level is known to be a significant improvement. Therefore, in recent years, some of the work with significant CTR performance improvements are at .001-level. However, this improvement will become less significant with a random seed change or even worse than the past works [43]. Hence, to validate that our distilled data can assist the stability of training results, in each model, we run ten experiments with ten different seeds and record the mean and standard variance in Table 7. The model parameters are initialized under the following setting: (1) Random initialization and (2) Training on a few distilled data as initialization. We obverse that with the help of our distilled data, the variation of standard variance among the three models is from .001-level decreased to .0001-level without harm to the overall performance.

## 6 ETHICAL STATEMENT

With our proposed CGM, it might involve some new biases in the recommendation area. One is inherited from the original data since the synthetic data is generated by imitating the training gradient of the original data. Another bias is from information lost during the gradient matching process since the performance of synthetic data is hard to surpass the original data. This may be because the matching focuses more on the features with the fastest gradient changes, while neglecting some features with small gradient changes yet located near the decision boundary.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we first explore the data distillation for the categorical recommendation data with plenty of side information on the task of CTR prediction. We attempt to distill the large categorical recommendation data into a small set of synthetic data for data-efficient training. We propose a data distillation method named Categorical data distillation with gradient matching (CGM). With the help of the proposed new data format, our CGM can successfully narrow the parameter distance of synthetic data to that of original data. Then to overcome the nested optimization, we introduce the one-step gradient matching strategy. These optimized synthetic data are significantly more data-efficient than other coreset and data distillation methods which can be generalized to train other unseen models.

For future work, we are interested in exploring the privacy-preserving issues with our condensed synthetic data [7]. It is also a promising solution for discrete data like graphs, speech, and sentiment for data-efficient training by data distillation with our new data format.

## 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] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4750–4759.

[2] 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.

[3] 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.

[4] Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4794–4802.

[5] 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.

[6] 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.

[7] Tian Dong, Bo Zhao, and Lingjuan Lyu. 2022. Privacy for free: How does dataset condensation help privacy?. In *International Conference on Machine Learning*. PMLR, 5378–5396.

[8] Dan Feldman, Melanie Schmidt, and Christian Sohler. 2020. Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering. *SIAM J. Comput.* 49, 3 (2020), 601–657.

[9] Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*. Springer, 181–195.

[10] 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.

[11] Zhexue Huang. 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery* 2, 3 (1998), 283–304.

[12] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[13] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. 2022. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 720–730.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[15] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. 2022. Dataset condensation with contrastive signals. In *International Conference on Machine Learning*. PMLR, 12352–12364.

[16] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 313–321.

[17] 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.

[18] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. Citeseer, 3.

[19] Rui Maia and Joao C Ferreira. 2018. Context-aware food recommendation system. *Context-aware food recommendation system* (2018), 349–356.

[20] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2020. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*. PMLR, 6950–6960.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[22] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems* 34 (2021), 20596–20607.

[23] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

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

[25] Noveen Sachdeva, Mehak Dhaliwal, Carole-Jean Wu, and Julian McAuley. 2022. Infinite recommendation networks: A data-centric approach. *Advances in Neural Information Processing Systems* 35 (2022), 31292–31305.

[26] Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. 2022. On Sampling Collaborative Filtering Datasets. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 842–850.

[27] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

[28] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems* 35 (2022), 19523–19536.

[29] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *International Conference on Learning Representations*.

[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] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[32] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2018. Dataset distillation. *arXiv preprint arXiv:1811.10959* (2018).

[33] 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.

[34] Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 1121–1128.

[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] 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*.

[37] 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.

[38] Bo Zhao and Hakan Bilen. 2021. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*. PMLR, 12674–12685.

[39] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.

[40] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset Condensation with Gradient Matching. *ICLR* 1, 2 (2021), 3.

[41] Weijie Zhao, Jingyuan Zhang, Deping Xie, Yulei Qian, Ronglai Jia, and Ping Li. 2019. Aibox: Ctr prediction model training on a single node. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 319–328.

[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.

[44] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems* 32 (2019).