# Infinite Recommendation Networks: A Data-Centric Approach

**Noveen Sachdeva**[†] **Mehak Preet Dhaliwal**[†] **Carole-Jean Wu**[♠] **Julian McAuley**[†]

University of California, San Diego[†] Meta AI[♠]

{nosachde,mdhaliwal,jmcauley}@ucsd.edu

carolejeanwu@fb.com

## Abstract

We leverage the Neural Tangent Kernel and its equivalence to training infinitely-wide neural networks to devise $\infty$-AE: an autoencoder with infinitely-wide bottleneck layers. The outcome is a highly expressive yet simplistic recommendation model with a single hyper-parameter and a closed-form solution. Leveraging $\infty$-AE's simplicity, we also develop DISTILL-CF for synthesizing tiny, high-fidelity data summaries which distill the most important knowledge from the extremely large and sparse user-item interaction matrix for efficient and accurate subsequent data-usage like model training, inference, architecture search, *etc.* This takes a data-centric approach to recommendation, where we aim to improve the quality of logged user-feedback data for subsequent modeling, independent of the learning algorithm. We particularly utilize the concept of differentiable Gumbel-sampling to handle the inherent data heterogeneity, sparsity, and semi-structuredness, while being scalable to datasets with hundreds of millions of user-item interactions. Both of our proposed approaches significantly outperform their respective state-of-the-art and when used together, we observe $96 - 105\%$ of $\infty$-AE's performance on the full dataset with as little as $0.1\%$ of the original dataset size, leading us to explore the counter-intuitive question: *Is more data what you need for better recommendation?*

## 1 Introduction

The Neural Tangent Kernel (NTK) has recently advanced the theoretical understanding of how neural networks learn [2, 20]. Notably, performing Kernelized Ridge Regression (KRR) with the NTK has been shown to be equivalent to training infinitely-wide neural networks for an infinite number of SGD steps. Owing to KRR's closed-form solution, these networks can be trained in a fast and efficient manner whilst not sacrificing expressivity. While the application of infinite neural networks is being explored in various learning problems [48, 3], detailed comparative analyses demonstrate that deep, finite-width networks tend to perform significantly better than their infinite-width counterparts for a variety of standard computer-vision tasks [31].

On the contrary, for recommendation tasks, there always has been a debate of linear *vs.* non-linear networks [29, 65], along with the importance of increasing the width *vs.* depth of the network [11, 39]. At a high level, the general conclusion is that a well-tuned, wide and linear network can outperform shallow and deep neural networks for recommendation [50]. We extend this debate by introducing our model $\infty$-AE, an autoencoder with infinitely wide bottleneck layers and examine its behavior on the recommendation task. Our evaluation demonstrates significantly improved results over state-of-the-art (SoTA) models across various datasets and evaluation metrics.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

A rising challenge in recommender systems research has been the increased cost of training models on massive datasets which can involve billions of user-item interaction logs, in terms of time, computational resources, as well as the environmental impact of an increased carbon footprint. Moreover, despite anonymization efforts, privacy risks have been associated with publicly released user data [38]. To this end, we further explore recommendation from a data-centric viewpoint [60], which we loosely define as:

**Definition 1.1. (Data-centric AI)** *Systematic methods for improving the collected data's quality, thereby shifting the focus from merely acquiring large quantities of data; implicitly helping in a learning algorithm's generalization, scalability, and eco-sustainability.*

Previous work on data-centric techniques generally involve constructing terse data summaries of large datasets, and has focused on domains with continuous, real-valued features such as images [66, 41], which are arguably more amenable to data optimization approaches. Due to the heterogeneity, sparsity, and semi-structuredness of recommendation data, such methods are not directly applicable. Common approaches for scaling down such recommendation datasets typically include heuristics such as random, head-user, or k-core sampling. More theoretically and empirically principled methods include coreset construction [57] that focus on optimizing for *picking* the set of data-points that are the most representative from a given dataset, and are generally shown to out-perform various heuristic sampling strategies. However, *synthesizing* informative summaries for recommendation datasets still remains a challenge.

Consequently, we propose DISTILL-CF, a data distillation framework for collaborative filtering (CF) datasets that utilizes $\infty$-AE in a bilevel optimization objective to create highly compressed, informative, and generic data summaries. DISTILL-CF employs efficient multi-step differentiable Gumbel-sampling [23] at each step of the optimization to encompass the heterogeneity, sparsity, and semi-structuredness of recommendation data. We further provide an analysis of the denoising effect observed when training the model on the synthesized versus the full dataset.

To summarize, *in this paper*, we make the following contributions:

- We develop $\infty$-AE: an infinite-width autoencoder for recommendation, that aims to reconstruct the incomplete preferences in a user's item consumption sequence. We demonstrate its efficacy on four datasets, and demonstrate that $\infty$-AE outperforms complicated SoTA models with only a single fully-connected layer, closed-form optimization, and a single hyper-parameter. We believe our work to be the first to demonstrate that an infinite-width network can outperform their finite-width SoTA counterparts for practical scenarios like recommendation.
- We subsequently develop DISTILL-CF: a novel data distillation framework that can synthesize tiny yet accurate data summaries for a variety of modeling applications. We empirically demonstrate similar performance of models trained on the full dataset versus training the same models on $2-3$ orders smaller data summaries synthesized by DISTILL-CF. Notably, DISTILL-CF and $\infty$-AE are complementary for each other's practicality, as $\infty$-AE's closed-form solution enables DISTILL-CF to scale to datasets with hundreds of millions of interactions; whereas, DISTILL-CF's succinct data summaries help improving $\infty$-AE's restrictive training complexity, and achieving SoTA performance when trained on these tiny data summaries.
- Finally, we also note that DISTILL-CF has a strong data denoising effect, validated with a counter-intuitive observation — if there is noise in the original data, models trained on *less* data synthesized by DISTILL-CF can be better than the same model trained on the *entire* original dataset. Such observations, along with the strong data compression results attained by DISTILL-CF, reinforce our data-centric viewpoint to recommendation, encouraging the community to think more about the quality of collected data, rather than its quantity.

## 2 Related Work

**Autoencoders for recommendation.** Recent approaches to implicit feedback recommendation involve building models that can re-construct an incomplete user preference list using autoencoders [32, 59, 54, 33]. The CDAE model [64] first introduced this idea and used a standard denoising autoencoder for recommending new items to users. MVAE [32] later extended this idea to use variational autoencoders, and provided a principled approach to perform variational inference for this model architecture. More recently, EASE [59] proposed using a shallow autoencoder and estimates

only an item-item similarity matrix by performing ordinary least squares regression on the relevance matrix, resulting in closed-form optimization.

**Infinite neural networks.** The Neural Tangent Kernel (NTK) [20] has gained significant attention because of its equivalence to training infinitely-wide neural networks by performing KRR with the NTK. Recent work also demonstrated the double-descent risk curve [4] that extends the classical regime of train *vs.* test error for overparameterized neural networks, and plays a crucial role in the generalization capabilities of such infinite neural networks. However, despite this emerging promise of utilizing NTK for learning problems, detailed comparative analyses [43, 31, 2] for computer vision tasks demonstrate that finite-width networks are still significantly more accurate than infinite-width ones. Looking at recommendation systems, [65] performed a theoretical comparison between Matrix Factorization (MF) and Neural MF [18] by studying their expressivity in the infinite-width limit, comparing the NTK of both of these algorithms. Notably, their settings involved the typical (user-ID, item-ID) input to the recommendation model, and observed results that were equivalent to a random predictor. [48] performed a similar study that performed matrix completion using the NTK of a single layer fully-connected neural network, but assumed meaningful feature-priors available beforehand.

**Data sampling & distillation.** Data sampling is ubiquitous — sampling negatives while contrastive learning [51, 25], sampling large datasets for fast experimentation [57], sampling for evaluating expensive metrics [30], *etc.* In this paper, we primarily focus on sampling at the dataset level, principled approaches of which can be categorized into: (1) coreset construction methods that aim to *pick* the most useful datapoints for subsequent model training [7, 27, 53, 26]. These methods typically assume the availability of a submodular set-function $f : \mathbf{V} \mapsto \mathbb{R}_+ \; \forall \; \mathbf{V} \subseteq \mathbf{X}$ for a given dataset $\mathbf{X}$ (see [6] for a systematic review on submodularity), and use this set-function $f$ as a proxy to guide the search for the most informative subset; and (2) dataset distillation: instead of picking the most informative data-points, dataset distillation techniques aim to *synthesize* data-points that can accurately distill the knowledge from the entire dataset into a small, synthetic data summary. Originally proposed in [62], the authors built upon the notions of gradient-based hyper-parameter optimization [34] to synthesize representative images for model training. Subsequently, a series of works [67, 66, 40, 41] propose various subtle modifications to the original framework, for improving the sample-complexities of models trained on data synthesized using their algorithms. Note that such distillation techniques focused on continuous data like images, which are trivial to optimize in the original data-distillation framework. More recently, [24] proposed a distillation technique for synthesizing fake graphs, but also assumed to have innate node representations available beforehand, prohibiting their method's application for recommendation data.

## 3 $\infty$-AE: Infinite-width Autoencoders for Recommendation

**Notation.** Given a recommendation dataset $\mathcal{D} \coloneqq \{(\text{user}_i, \text{item}_i, \text{relevance}_i)\}_{i=1}^{n}$ consisting of $n$ user-item interactions defined over a set of users $\mathcal{U}$, set of items $\mathcal{I}$, and operating over a binary relevance score (implicit feedback); we aim to best model user preferences for item recommendation. The given dataset $\mathcal{D}$ can also be viewed as an interaction matrix, $X \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ where each entry $X_{u,i}$ either represents the observed relevance for item $i$ by user $u$ or is missing. Note that $X$ is typically extremely sparse, *i.e.*, $n \ll |\mathcal{U}| \times |\mathcal{I}|$. More formally, we define the problem of recommendation as accurate likelihood modeling of $\text{P}(i \mid u, \mathcal{D}) \; \forall u \in \mathcal{U}, \; \forall i \in \mathcal{I}$.

**Model.** $\infty$-AE takes an autoencoder approach to recommendation, where the all of the bottleneck layers are infinitely-wide. Firstly, to make the original bi-variate problem of which *item* to recommend to which *user* more amenable for autoencoders, we make a simplifying assumption that a user can be represented only by their historic interactions with items, *i.e.*, the much larger set of users lie in the linear span of items. This gives us two kinds of modeling advantages: (1) we no longer have to find a unique latent representation of users; and (2) allows $\infty$-AE to be trivially applicable for any user not in $\mathcal{U}$. More specifically, for a given user $u$, we represent it by the sparse, bag-of-words vector of historical interactions with items $X_u \in \mathbb{R}^{|\mathcal{I}|}$, which is simply the $u^{\text{th}}$ row in $X$. We then employ the Neural Tangent Kernel (NTK) [20] of an autoencoder that takes the bag-of-items representation of users as input and aims to reconstruct it. Due to the infinite-width correspondence [20, 2], performing Kernelized Ridge Regression (KRR) with the estimated NTK is equivalent to training an infinitely-wide autoencoder for an infinite number of SGD steps. More formally, given

the NTK, $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$ over an RKHS $\mathcal{H}$ of a single-layer autoencoder with an activation function $\sigma$ (see [47] for the NTK derivation of a fully-connected neural network), we reduce the recommendation problem to KRR as follows:

$$\begin{aligned} &\underset{[\alpha_j]_{j=1}^{|\mathcal{U}|}}{\arg\min} \quad \sum_{u \in \mathcal{U}} \|f(X_u \mid \alpha) - X_u\|_2^2 + \lambda \cdot \|f\|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & f(X_u \mid \alpha) = \sum_{j=1}^{|\mathcal{U}|} \alpha_j \cdot \mathbb{K}(X_u, X_{u_j}) \;\; ; \;\; \mathbb{K}(X_u, X_v) = \tilde{\sigma}(X_u^T X_v) + \tilde{\sigma'}(X_u^T X_v) \cdot X_u^T X_v \end{aligned} \tag{1}$$

Where $\lambda$ is a fixed regularization hyper-parameter, $\alpha \coloneqq [\alpha_1; \alpha_2 \cdots ; \alpha_{|\mathcal{U}|}] \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ are the set of dual parameters we are interested in estimating, $\tilde{\sigma}$ represents the dual activation of $\sigma$ [14], and $\tilde{\sigma'}$ represents its derivative. Defining a gramian matrix $K \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$ where each element can intuitively be seen as the *similarity* of two users, *i.e.*, $K_{i,j} \coloneqq \mathbb{K}(X_{u_i}, X_{u_j})$, the optimization problem defined in Equation (1) has a closed-form solution given by $\hat{\alpha} = (K + \lambda I)^{-1} \cdot X$. We can subsequently perform inference for any novel user as follows: $\hat{\mathrm{P}}(\cdot \mid u, \mathcal{D}) = \text{softmax}(f(X_u \mid \hat{\alpha}))$. We also provide $\infty$-AE's training and inference pseudo-codes in Appendix A, Algorithms 1 and 2.

**Scalability.** We carefully examine the computational cost of $\infty$-AE's training and inference. Starting with training, $\infty$-AE has the following computationally-expensive steps: (1) computing the gramian matrix $K$; and (2) performing its inversion. The overall training time complexity thus comes out to be $\mathcal{O}(|\mathcal{U}|^2 \cdot |\mathcal{I}| + |\mathcal{U}|^{2.376})$ if we use the Coppersmith-Winograd algorithm [12] for matrix inversion, whereas the memory complexity is $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{I}| + |\mathcal{U}|^2)$ for storing the data matrix $X$ and the gramian matrix $K$. As for inference for a single user, both the time and memory requirements are $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{I}|)$. Observing these computational complexities, we note that $\infty$-AE can be difficult to scale-up to larger datasets naïvely. To this effect, we address $\infty$-AE's scalability challenges in DISTILL-CF (Section 4), by leveraging a simple observation from support vector machines: not all datapoints (users in our case) are important for model learning. Additionally, in practice, we can perform all of these matrix operations on accelerators like GPU/TPUs and achieve a much higher throughput.

## 4 DISTILL-CF

**Motivation.** Representative sampling of large datasets has numerous downstream applications. Consequently, in this section we develop DISTILL-CF: a data distillation framework to *synthesize* small, high-fidelity data summaries for collaborative filtering (CF) data. We design DISTILL-CF with the following rationales: (1) mitigating the scalability challenges in $\infty$-AE by training it only on the terse data summaries generated by DISTILL-CF; (2) improving the sample complexity of existing, finite-width recommendation models; (3) addressing the privacy risks of releasing user feedback data by releasing their synthetic data summaries instead; and (4) abating the downstream $CO_2$ emissions of frequent, large-scale recommendation model training by estimating their parameters only on much smaller data summaries synthesized by DISTILL-CF.

**Challenges.** Existing work in data distillation has focused on continuous domain data such as images [40, 41, 67, 66], and are not directly applicable to heterogeneous and semi-structured domains such as recommender systems and graphs. This problem is further exacerbated since the data for these tasks is severely sparse. For example, in recommendation scenarios, a vast majority of users interact with very few items [22]. Likewise, the nodes in a number of graph-based datasets tend to have connections with very small set of nodes [68]. We will later show how our DISTILL-CF framework is elegantly able to circumvent both these issues while being accurate, and scalable to large datasets.

**Methodology.** Given a recommendation dataset $\mathcal{D}$, we aim to synthesize a smaller, support dataset $\mathcal{D}^s$ that can match the performance of recommendation models $\phi : \mathcal{U} \times \mathcal{I} \mapsto \mathbb{R}$ when trained on $\mathcal{D}$ versus $\mathcal{D}^s$. We take inspiration from [40], which is also a data distillation technique albeit for images. Formally, given a recommendation model $\phi$, a held-out validation set $\mathcal{D}^{\text{val}}$, and a differentiable loss function $l : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ that measures the correctness of a prediction with the actual user-item relevance, the data distillation task can be viewed as the following bilevel optimization problem:

$$\underset{\mathcal{D}^s}{\arg\min} \quad \underset{(u,i,r)\sim\mathcal{D}^{\text{val}}}{\mathbb{E}} \left[ l\left(\phi^*_{\mathcal{D}^s}(u,i), r\right) \right] \quad ; \quad \text{s.t.} \quad \phi^*_{\mathcal{D}^s} \coloneqq \underset{\phi}{\arg\min} \quad \underset{(u,i,r)\sim\mathcal{D}^s}{\mathbb{E}} \left[ l(\phi(u,i), r) \right] \tag{2}$$

This optimization problem has an *outer loop* which searches for the most informative support dataset $\mathcal{D}^s$ given a fixed recommendation model $\phi^*$, whereas the *inner loop* aims to find the optimal recommendation model for a fixed support set. In DISTILL-CF, we use $\infty$-AE as the model of choice at each step of the inner loop for two reasons: (1) as outlined in Section 3, $\infty$-AE has a closed-form solution with a single hyper-parameter $\lambda$, making the inner-loop extremely efficient; and (2) due to the infinite-width correspondence [20, 2], $\infty$-AE is equivalent to training an infinitely-wide autoencoder on $\mathcal{D}^s$, thereby not compromising on performance.

For the outer loop, we focus only on synthesizing *fake users* (given a fixed user budget $\mu$) through a learnable matrix $X^s \in \mathbb{R}^{\mu \times |\mathcal{I}|}$ which stores the interactions for each fake user in the support dataset. However, to handle the discrete nature of the recommendation problem, instead of directly optimizing for $X^s$, DISTILL-CF instead learns a *continuous prior* for each user-item pair, denoting the importance of sampling that interaction in our support set (similar to the notion of propensity [56, 58]). We then sample $\hat{X}^s \sim X^s$ to get our final, discrete support set.

Instead of keeping this sampling operation post-hoc, *i.e.*, after solving for the optimization problem in Equation (2); we perform differentiable Gumbel-sampling [23] on each row (user) in $X^s$ at every optimization step, thereby ensuring search only over sparse and discrete support sets. A notable property of recommendation data is that each user can interact with a variable number of items (this distribution is typically long-tailed due to Zipf's law [63]). We circumvent this dynamic user sparsity issue by taking multiple Gumbel-samples for each user, with replacement. This implicitly gives DISTILL-CF the freedom to control the user and item popularity distributions in the generated data summary $\hat{X}^s$ by adjusting the entropy in the prior-matrix $X^s$.

Having controlled for the discrete and dynamic nature of recommendation data by the multi-step Gumbel-sampling trick, we further focus on maintaining the sparsity of the synthesized data. To do so, in addition to the outer-loop reconstruction loss, we add an L1-penalty over $\hat{X}^s$ to promote and explicitly control its sparsity [16, Chapter 3]. Furthermore, tuning the number of Gumbel samples we take for each fake user, gives us more control over the sparsity in our generated data summary. More formally, the final optimization objective in DISTILL-CF can be written as:

$$\begin{aligned} \underset{X^s}{\arg\min} \quad & \underset{u\sim\mathcal{U}}{\mathbb{E}} \left[ X_u \cdot \log(K_{X_u \hat{X}^s} \cdot \alpha) + (1 - X_u) \cdot \log(1 - K_{X_u \hat{X}^s} \cdot \alpha) \right] + \lambda_2 \cdot ||\hat{X}^s||_1 \\ \text{s.t.} \quad & \alpha = (K_{\hat{X}^s \hat{X}^s} + \lambda I)^{-1} \cdot \hat{X}^s \quad ; \quad \hat{X}^s = \sigma\left( \sum_{i=1}^{\gamma} \text{Gumbel}_\tau(\text{softmax}(X^s)) \right) \end{aligned} \tag{3}$$

Where, $\lambda_2$ represents an appropriate regularization hyper-parameter for minimizing the L1-norm of the sampled support set $\hat{X}^s$, $K_{XY}$ represents the gramian matrix for the NTK of $\infty$-AE over $X$ and $Y$ as inputs, $\tau$ represents the temperature hyper-parameter for Gumbel-sampling, $\gamma$ denotes the number of samples to take for each fake user in $X^s$, and $\sigma$ represents an appropriate activation function which clips all values over 1 to be exactly 1, thereby keeping $\hat{X}^s$ binary. We use hard-tanh in our experiments. We also provide DISTILL-CF's pseudo-code in Appendix A, Algorithm 3.

**Scalability.** We now analyze the time and memory requirements for optimizing Equation (3). The inner loop's major component clearly shares the same complexity as $\infty$-AE. However, since the parameters of $\infty$-AE ($\alpha$ in Equation (1)) are now being estimated over the much smaller support set $\hat{X}^s$, the time complexity reduces to $\mathcal{O}(\mu^2 \cdot |\mathcal{I}|)$ and memory to $\mathcal{O}(\mu \cdot |\mathcal{I}|)$, where $\mu$ typically only lies in the range of hundreds for competitive performance. On the other hand, for performing multi-step Gumbel-sampling for each synthetic user, the memory complexity of a naïve implementation would be $\mathcal{O}(\gamma \cdot \mu \cdot |\mathcal{I}|)$ since AutoGrad stores all intermediary variables for backward computation. However, since the gradient of each Gumbel-sampling step is independent of other sampling steps and can be computed individually, using `jax.custom_vjp`, we reduced its memory complexity to $\mathcal{O}(\mu \cdot |\mathcal{I}|)$, adding nothing to the overall inner-loop complexity.

Table 1: Comparison of $\infty$-AE with different methods on various datasets. All metrics are better when higher. Brief set of data statistics can be found in Appendix B.3, Table 2. **Bold** values represent the best in a given row, and underlined represent the second-best. Results for $\infty$-AE on the Netflix dataset (marked with a *) consist of random user-sampling with a max budget of $25\text{K} \equiv 5.4\%$ users, and results for DISTILL-CF + $\infty$-AE have a user-budget of 500 for all datasets.

| Dataset | Metric | PopRec | Bias only | MF | NeuMF | Light GCN | EASE | MVAE | $\infty$-AE | DISTILL-CF + $\infty$-AE |
|---|---|---|---|---|---|---|---|---|---|---|
| **Amazon Magazine** | AUC | 0.8436 | 0.8445 | 0.8475 | 0.8525 | 0.8141 | 0.6673 | 0.8507 | 0.8539 | **0.8584** |
| | HR@10 | 14.35 | 14.36 | 18.36 | 18.35 | 27.12 | 26.31 | 17.94 | 27.09 | **28.27** |
| | HR@100 | 59.5 | 59.35 | 58.94 | 59.3 | 58.00 | 48.36 | 57.3 | 60.86 | **61.78** |
| | NDCG@10 | 8.42 | 8.33 | 13.1 | 13.6 | 22.57 | 22.84 | 12.18 | 23.06 | **23.81** |
| | NDCG@100 | 19.38 | 19.31 | 21.76 | 21.13 | 29.92 | 28.27 | 19.46 | 30.75 | **31.75** |
| | PSP@10 | 6.85 | 6.73 | 9.24 | 9.00 | 13.2 | 12.96 | 8.81 | 13.22 | **13.76** |
| **ML-1M** | AUC | 0.8332 | 0.8330 | 0.9065 | 0.9045 | 0.9289 | 0.9069 | 0.8832 | **0.9457** | 0.9415 |
| | HR@10 | 13.07 | 12.93 | 24.63 | 23.25 | 27.43 | 28.54 | 21.7 | **31.51** | 31.16 |
| | HR@100 | 30.38 | 29.63 | 53.26 | 51.42 | 55.61 | 57.28 | 52.29 | **60.05** | 58.28 |
| | NDCG@10 | 13.84 | 13.74 | 25.65 | 24.44 | 28.85 | 29.88 | 22.14 | **32.82** | 32.52 |
| | NDCG@100 | 19.49 | 19.13 | 35.62 | 33.93 | 38.29 | 40.16 | 33.82 | **42.53** | 41.29 |
| | PSP@10 | 1.10 | 1.07 | 2.41 | 2.26 | 2.72 | 3.06 | 2.42 | **3.22** | 3.15 |
| **Douban** | AUC | 0.8945 | 0.8932 | 0.9288 | 0.9258 | 0.9391 | 0.8570 | 0.9129 | **0.9523** | 0.9510 |
| | HR@10 | 11.06 | 10.71 | 12.69 | 12.79 | 15.98 | 17.93 | 15.36 | **23.56** | 22.98 |
| | HR@100 | 17.07 | 16.63 | 20.29 | 19.69 | 22.38 | 25.41 | 22.82 | **28.37** | 27.20 |
| | NDCG@10 | 11.63 | 11.24 | 13.21 | 13.33 | 16.68 | 19.48 | 16.17 | **24.94** | 24.20 |
| | NDCG@100 | 12.63 | 12.27 | 14.96 | 14.39 | 17.20 | 19.55 | 17.32 | **23.26** | 22.21 |
| | PSP@10 | 0.52 | 0.50 | 0.63 | 0.63 | 0.86 | 1.06 | 0.87 | **1.28** | 1.24 |
| **Netflix** | AUC | 0.9249 | 0.9234 | 0.9234 | 0.9244 | | 0.9418 | 0.9495 | 0.9663* | **0.9728** |
| | HR@10 | 12.14 | 11.49 | 11.69 | 11.06 | | 26.03 | 20.6 | **29.69*** | 29.57 |
| | HR@100 | 28.47 | 27.66 | 27.72 | 26.76 | Timed | 50.35 | 44.53 | **50.88*** | 49.24 |
| | NDCG@10 | 12.34 | 11.72 | 12.04 | 11.48 | Out | 26.83 | 20.85 | **30.59*** | 30.54 |
| | NDCG@100 | 17.79 | 16.95 | 17.17 | 16.40 | | 35.09 | 29.22 | **36.59*** | 35.58 |
| | PSP@10 | 1.45 | 1.28 | 1.31 | 1.21 | | 3.59 | 2.77 | **3.75*** | 3.62 |

For the outer loop, we optimize the logistic reconstruction loss using SGD where we randomly sample a batch of $b$ users from $\mathcal{U}$ and update $X^s$ directly. In totality, for an $\xi$ number of outer-loop iterations, the time complexity to run DISTILL-CF is $\mathcal{O}(\xi \cdot (\mu^2 + b + b \cdot \mu) \cdot |\mathcal{I}|)$, and $\mathcal{O}(b \cdot \mu + (\mu + b) \cdot |\mathcal{I}|)$ for memory. In our experiments, we note convergence in only hundreds of outer-loop iterations, making DISTILL-CF scalable even for the largest of the publicly available datasets and practically useful.

## 5 Experiments

**Setup.** We use four recommendation datasets with varying sizes and sparsity characteristics. A brief set of data statistics can be found in Appendix B.3, Table 2. For each user in the dataset, we randomly split their interaction history into $80/10/10\%$ train-test-validation splits. Following recent warnings against unrealistic dense preprocessing of recommender system datasets [55, 57], we only prune users that have fewer than 3 interactions to enforce at least one interaction per user in the train, test, and validation sets. No such preprocessing is followed for items.

**Competitor methods & evaluation metrics.** We compare $\infty$-AE with various baseline and SoTA competitors as surveyed in recent comparative analyses [1, 13]. More details on their architectures can be found in Appendix B.1. We evaluate all models on a variety of pertinent ranking metrics, namely AUC, HitRate (HR@k), Normalized Discounted Cumulative Gain (nDCG@k), and Propensity-scored Precision (PSP@k), each focusing on different components of the algorithm performance. A notable addition to our list of metrics compared to the literature is the PSP metric [22], which we adapt to the recommendation use case as an indicator of performance on tail items. The exact definition of all of these metrics can be found in Appendix B.5.

**Training details.** We implement both $\infty$-AE and DISTILL-CF using `JAX` [8] along with the Neural-Tangents package [44] for the relevant NTK computations.[1,2] We re-use the official implementation

[1]Our implementation for $\infty$-AE is available at https://github.com/noveens/infinite_ae_cf
[2]Our implementation for DISTILL-CF is available at https://github.com/noveens/distill_cf

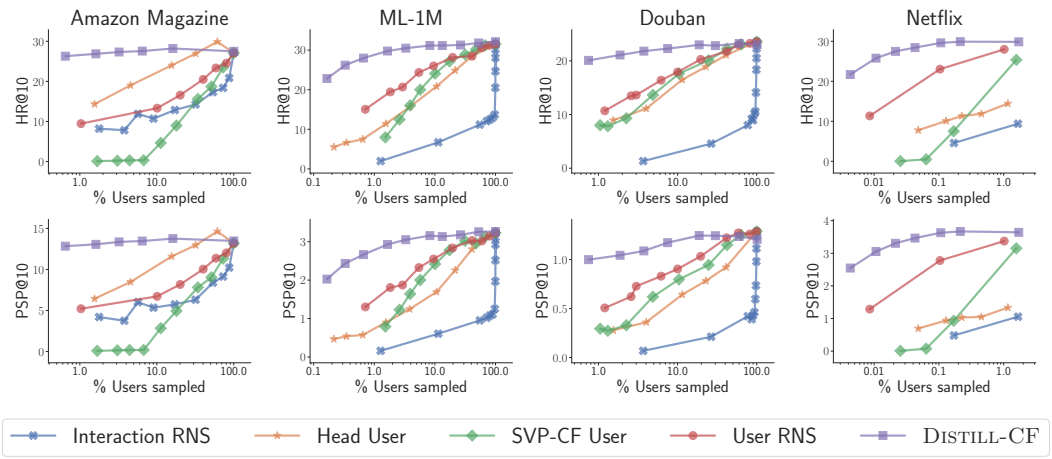


Figure 1: Performance of $\infty$-AE with the amount of users (log-scale) sampled according to different sampling strategies over the HR@10 and PSP@10 metrics. Results for the Netflix dataset have been clipped due to memory constraints. Results for the remaining metrics can be found in Appendix B.6, Figure 13.

of LightGCN, and implement the remaining competitors ourselves. To ensure a fair comparison, we conduct a hyper-parameter search for all competitors on the validation set. More details on the hyper-parameters for $\infty$-AE, DISTILL-CF, and all competitors can be found in Appendix B.3, Table 3. All of our experiments are performed on a single RTX 3090 GPU, with a random-seed initialization of 42. Additional training details about DISTILL-CF can be found in Appendix B.4.

**Does $\infty$-AE outperform existing methods?** We compare the performance of $\infty$-AE with various baseline and competitor methods in Table 1. We also include the results of training $\infty$-AE on data synthesized by DISTILL-CF with an additional constraint of having a budget of only $\mu = 500$ synthetic users. For the sake of reference, for our largest dataset (Netflix), this equates to a mere $0.1\%$ of the total users. There are a few prominent observations from the results in Table 1. First, $\infty$-AE significantly outperforms SoTA recommendation algorithms despite having only a single fully-connected layer, and also being much simpler to train and implement. Second, we note that $\infty$-AE trained on just $500$ users generated by DISTILL-CF is able to attain $96 - 105\%$ of $\infty$-AE's performance on the full dataset while also outperforming all competitors trained on the *full* dataset.

**How sample efficient is $\infty$-AE?** Having noted from Table 1 that $\infty$-AE is able to outperform all SoTA competitors with as little as $0.1\%$ of the original users, we now aim to better understand $\infty$-AE's sample complexity, *i.e.*, the amount of training data $\infty$-AE needs in order to perform accurate recommendation. In addition to DISTILL-CF, we use the following popular heuristics for down-sampling (more details in Appendix B.2): interaction random negative sampling (RNS); user RNS; head user sampling; and a coreset construction technique, SVP-CF user [57]. We then train $\infty$-AE on sampled data for different sampling budgets, while evaluating on the original test-set. We plot the performance for all datasets computed over the HR@10 and PSP@10 metrics in Figure 1. We observe that while all heuristic sampling strategies tend to be closely bound to the identity line with a slight preference to user RNS, $\infty$-AE when trained on data synthesized by DISTILL-CF tends to quickly saturate in terms of performance when the user budget is increased, even on the log-scale. This indicates DISTILL-CF's superiority in generating terse data summaries for $\infty$-AE, thereby allowing it to get SoTA performance on the largest datasets with as little as $500$ users.

**How transferable are the data summaries synthesized by DISTILL-CF?** In order to best evaluate the quality and universality of data summaries synthesized by DISTILL-CF, we train and evaluate EASE [59] on data synthesized by DISTILL-CF. Note that the inner loop of DISTILL-CF still consists of $\infty$-AE, but we nevertheless train and evaluate EASE to test the synthesized data's universality. We re-use the heuristic sampling strategies from the previous experiment for comparison with DISTILL-CF. From the results in Figure 2, we observe similar scaling laws as $\infty$-AE's for the heuristic

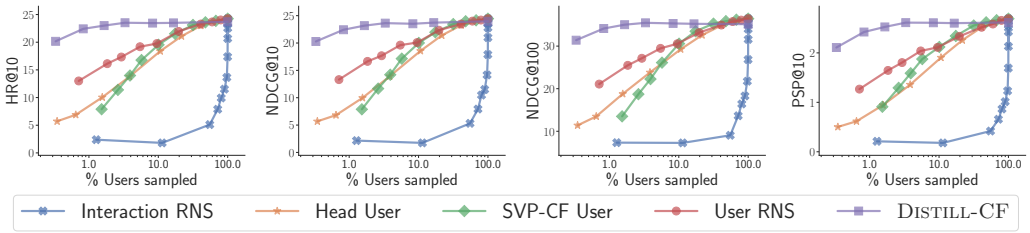


Figure 2: Performance of the EASE model trained on different amounts of users (log-scale) sampled by different samplers on the ML-1M dataset.

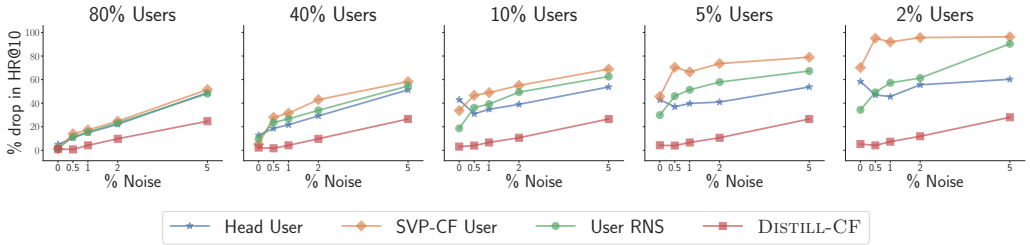


Figure 3: Performance drop of the EASE model trained on data sampled by different sampling strategies when there is varying levels of noise in the data. Performance drop is relative to training on the full, noise-free ML-1M dataset. Results for the remaining metrics can be found in Appendix B.6, Figure 14.

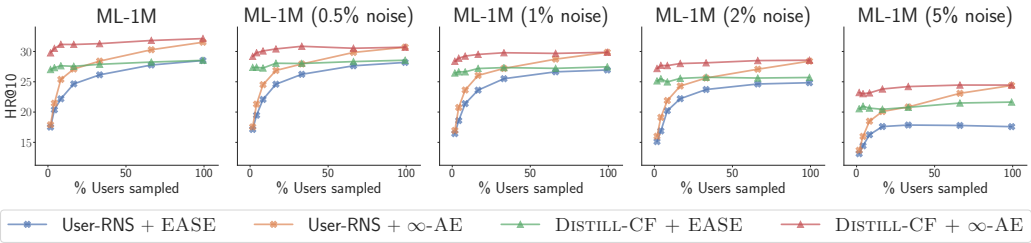


Figure 4: Performance of $\infty$-AE on data sampled by DISTILL-CF and User-RNS when there is noise in the data. Results for EASE have been added for reference. All results are on the ML-1M dataset. Results for the remaining metrics can be found in Appendix B.6, Figure 15.

samplers as well as DISTILL-CF. The semantically similar results for MVAE [32] are presented in Appendix B.6, Figure 10 for completeness. This behaviour validates the re-usability of data summaries generated by DISTILL-CF, because they transfer well to SoTA finite-width models, which were not involved in DISTILL-CF's user synthesis optimization.

**How robust are DISTILL-CF and $\infty$-AE to noise?** User feedback data is often noisy due to various biases (see [10] for a detailed review). Furthermore, due to the significant number of logged interactions in these datasets, recommender systems are often trained on down-sampled data in practice. Despite this, to the best of our knowledge, there is no prior work that explicitly studies the interplay between noise in the data and how sampling it affects downstream model performance. Consequently, we simulate a simple experiment: we add $x\%$ noise in the original train-set $\rightarrow$ sample the noisy training data $\rightarrow$ train recommendation models on the sampled data $\rightarrow$ evaluate their performance on the original, noise-free test-set. For the noise model, we randomly flip $x\%$ of the total number of items in the corpus for each user. In Figure 3, we compare the drop in HR@10 the EASE model suffers for different sampling strategies when different levels of noise are added to the MovieLens-1M dataset [15]. We make a few main observations: (1) unsurprisingly, sampling noisy data compounds the performance losses of learning algorithms; (2) DISTILL-CF has the best

noise:sampling:performance trade-off compared to other sampling strategies, with an increasing performance gap relative to other samplers as we inject more noise into the original data; and (3) as we down-sample noisy data more aggressively, head user sampling improves relative to other samplers, simply because these head users are the least affected by our noise injection procedure.

Furthermore, to better understand $\infty$-AE's denoising capabilities, we repeat the aforementioned noise-injection experiment but now train $\infty$-AE on down-sampled, noisy data. In Figure 4, we track the change in $\infty$-AE's performance as a function of the number of users sampled, and the amount of noise injected before sampling. We also add the semantically equivalent results for the EASE model for reference. Firstly, we note that the full-data performance-gap between $\infty$-AE and EASE significantly increases when there is more noise in the data, demonstrating $\infty$-AE's robustness to noise, even when its not specifically optimized for it. Furthermore, looking at the 5% noise injection scenario, we notice two counter-intuitive observations: (1) training EASE on tiny data summaries synthesized by DISTILL-CF is better than training it on the full data; and (2) solely looking at data synthesized by DISTILL-CF for EASE, we notice the best performance when we have a lower user sampling budget. Both of these observations call for more investigation of a data-centric viewpoint to recommendation, *i.e.*, focusing more on the quality of collected data rather than its quantity.

**Applications to continual learning.** Continual learning (see [45] for a detailed review) is an important area for recommender systems, because these systems are typically updated at regular intervals. A continual learning scenario involves data that is split into multiple periods, with the predictive task being: given data until the $i^{\text{th}}$ period, maximize algorithm performance for prediction on the $(i+1)^{\text{th}}$ period. ADER [35] is a SoTA continual learning model for recommender systems, that injects the most informative user sequences from the last period to combat the catastrophic forgetting problem [52]. An intuitive application for DISTILL-CF is to synthesize succinct data summaries of the last period and inject these instead. To compare these approaches, we simulate a continual learning scenario by splitting the MovieLens-1M dataset into 17 equal sized epochs, and perform experiments with MVAE [32] for each period. Note that in DISTILL-CF, we still use $\infty$-AE to synthesize data summaries (inner loop). We also compare with two baselines: (1) Joint: concatenate the data from all periods before the current; and (2) Individual: use the data only from the current period. As we can see from Figure 5, DISTILL-CF consistently outperforms ADER and the baselines, again demonstrating its ability to generate high-fidelity data summaries.

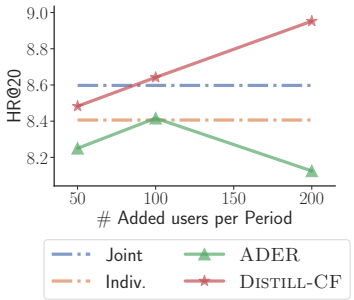


Figure 5: DISTILL-CF for continual learning.

## 6 Conclusion & Future Work

In this work, we proposed two complementary ideas: $\infty$-AE, an infinite-width autoencoder for modeling recommendation data, and DISTILL-CF for creating tiny, high-fidelity data summaries of massive datasets for subsequent model training. To our knowledge, our work is the first to employ and demonstrate that infinite-width neural networks can beat complicated SoTA models on recommendation tasks. Further, the data summaries synthesized through DISTILL-CF outperform generic samplers and demonstrate further performance gains for $\infty$-AE as well as finite-width SoTA models despite being trained on orders of magnitude less data.

Both our proposed methods are closely linked with one another: $\infty$-AE's closed-loop formulation is especially crucial in the practicality of DISTILL-CF, whereas DISTILL-CF's ability to distill the entire dataset's knowledge into small summaries helps $\infty$-AE to scale to large datasets. Moreover, the Gumbel sampling trick enables us to adapt data distillation techniques designed for continuous, real-valued, dense domains to heterogeneous, semi-structured, and sparse domains like recommender systems and graphs. We additionally explore the strong denoising effect observed with DISTILL-CF, noting that in the case of noisy data, models trained on considerably less data synthesized by DISTILL-CF perform better than the same model trained on the entire original dataset. These observations lead us to contemplate a much larger, looming question: *Is more data what you need for recommendation?* Our results call for further investigation on the data-centric viewpoint of recommendation.

The findings of our paper open up numerous promising research directions. First, building such closed-form, easy-to-implement infinite networks is beneficial for various downstream practical applications like search, sequential recommendation, or CTR prediction. Further, the anonymization achieved by synthesizing fake data summaries is crucial for mitigating the privacy risks associated with confidential or PII datasets. Another direction is analyzing the environmental impact and reduction in carbon footprint as our experiments show that models can achieve similar performance gains when trained on much less data.

## References


[1] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, Dietmar Jannach, and Claudio Pomo. Top-n recommendation algorithms: A quest for the state-of-the-art. *arXiv preprint arXiv:2203.01155*, 2022.

[2] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020.

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

[5] James Bennett and Stan Lanning. The netflix prize. 2007.

[6] Jeff Bilmes. Submodularity in machine learning and artificial intelligence. *arXiv preprint arXiv:2202.00132*, 2022.

[7] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[9] Dong-Kyu Chae, Jihoo Kim, Duen Horng Chau, and Sang-Wook Kim. Ar-cf: Augmenting virtual users and items in collaborative filtering for addressing cold-start problems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1251–1260, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *CoRR*, abs/2010.03240, 2020.

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

[12] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.

[13] M. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, 2019.

[14] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *Advances in neural information processing systems*, 29, 2016.

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

[16] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 2017.

[19] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

[20] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[21] H. Jain, V. Balasubramanian, B. Chunduri, and M. Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the 12th ACM Conference on Web Search and Data Mining*, 2019.

[22] H. Jain, Y. Prabhu, and M. Varma. Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016.

[23] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[24] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022.

[25] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020.

[26] Ehsan Kazemi, Shervin Minaee, Moran Feldman, and Amin Karbasi. Regularized submodular maximization at scale. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5356–5366. PMLR, 18–24 Jul 2021.

[27] Krishnateja Killamsetty, Durga S, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR, 18–24 Jul 2021.

[28] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[29] Taeyong Kong, Taeri Kim, Jinsung Jeon, Jeongwhan Choi, Yeon-Chang Lee, Noseong Park, and Sang-Wook Kim. Linear, or non-linear, that is the question! In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 517–525, 2022.

[30] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 2020.

[31] Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

[32] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, 2018.

[33] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[34] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[35] Fei Mi, Xiaoyu Lin, and Boi Faltings. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Fourteenth ACM Conference on Recommender Systems*, page 408–413, New York, NY, USA, 2020. Association for Computing Machinery.

[36] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[37] A. Mittal, N. Sachdeva, S. Agrawal, S. Agarwal, P. Kar, and M. Varma. Eclare: Extreme classification with label graph correlations. In *Proceedings of The ACM International World Wide Web Conference*, 2021.

[38] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[39] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019.

[40] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021.

[41] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[42] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[43] Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

[44] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020.

[45] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[46] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the International World Wide Web Conference*, 2018.

[47] Adityanarayanan Radhakrishnan. Lecture 5: Ntk origin and derivation. 2022.

[48] Adityanarayanan Radhakrishnan, George Stefanakis, Mikhail Belkin, and Caroline Uhler. Simple, fast, and flexible framework for matrix completion with infinite width neural networks. *Proceedings of the National Academy of Sciences*, 119(16):e2115064119, 2022.

[49] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 2009.

[50] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Fourteenth ACM conference on recommender systems*, pages 240–248, 2020.

[51] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

[52] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[53] Durga S, Rishabh Iyer, Ganesh Ramakrishnan, and Abir De. Training data subset selection for regression with controlled generalization error. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9202–9212. PMLR, 18–24 Jul 2021.

[54] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, WSDM '19, 2019.

[55] Noveen Sachdeva and Julian McAuley. How useful are reviews for recommendation? a critical review and potential improvements. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 2020.

[56] Noveen Sachdeva, Yi Su, and Thorsten Joachims. Off-policy bandits with deficient support. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, 2020.

[57] Noveen Sachdeva, Carole-Jean Wu, and Julian McAuley. On sampling collaborative filtering datasets. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 842–850, New York, NY, USA, 2022. Association for Computing Machinery.

[58] T. Schnabel, A. Swaminathan, A. Singh, N. Chandak, and T. Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[59] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257, 2019.

[60] Eliza Strickland. Andrew ng: Farewell, big data. IEEE Spectrum, Mar 2022.

[61] M. Toneva, A. Sordoni, R. Combes, A. Trischler, Y. Bengio, and G. Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.

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

[63] Carole-Jean Wu, Robin Burke, Ed H. Chi, Joseph Konstan, Julian McAuley, Yves Raimond, and Hao Zhang. Developing a recommendation benchmark for mlperf training and inference. *CoRR*, abs/2003.07336, 2020.

[64] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162, 2016.

[65] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Rethinking neural vs. matrix-factorization collaborative filtering: the theoretical perspectives. In *International Conference on Machine Learning*, pages 11514–11524. PMLR, 2021.

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

[67] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *International Conference on Learning Representations*, 2021.

[68] Wenqing Zheng, Edward W Huang, Nikhil Rao, Sumeet Katariya, Zhangyang Wang, and Karthik Subbian. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In *International Conference on Learning Representations*, 2022.

[69] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1533–1542, 2019.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We discuss both $\infty$-AE and DISTILL-CF in detail in Section 3 and Section 4, along with validating their performance through experiments in Section 5
   (b) Did you describe the limitations of your work? [Yes] We talked about the lack of scalability of using $\infty$-AE naïvely in Section 3
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Discussed throughout the setup sub-section in Section 5, and provided detailed instructions in Appendix B
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Discussed in the setup sub-section in Section 5, and provided hyper-parameters in Appendix B.3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Noted the seed we used in our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Discussed in the training details subsection in Section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Released our anonymous code as mentioned in Section 5
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Appendix: Pseudo-codes

**Algorithm 1** $\infty$-AE model training

**Input:** User set $\mathcal{U}$ ; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$ ; regularization const. $\lambda \in \mathbb{R}$
**Output:** Dual parameters $\alpha \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$

**procedure** FIT($\mathcal{U}, \mathbf{X}, \mathbb{K}$)
  $\mathbf{K} \leftarrow [0]_{|\mathcal{U}|\times|\mathcal{U}|}$ ▷ Zero Initialization
  $\mathbf{K}_{u,v} \leftarrow \mathbb{K}(\mathbf{X}_u, \mathbf{X}_v) \quad \forall u \in \mathcal{U}, v \in \mathcal{U}$
  $\alpha \leftarrow (\mathbf{K} + \lambda I)^{-1} \cdot \mathbf{X}$
  **return** $\alpha$

**Algorithm 2** $\infty$-AE inference

**Input:** User set $\mathcal{U}$ ; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$; dual params. $\alpha \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ ; inference user history $\hat{\mathbf{X}}_u \in \mathbb{R}^{|\mathcal{I}|}$
**Output:** Prediction $\hat{y} \in \mathbb{R}^{|\mathcal{I}|}$

**procedure** PREDICT($\mathcal{U}, \mathbf{X}, \hat{\mathbf{X}}_u, \mathbb{K}, \alpha$)
  $\mathbf{K} \leftarrow [0]_{|\mathcal{U}|}$ ▷ Zero Initialization
  $\mathbf{K}_v \leftarrow \mathbb{K}(\hat{\mathbf{X}}_u, \mathbf{X}_v) \quad \forall v \in \mathcal{U}$
  $\hat{y} \leftarrow softmax(\mathbf{K} \cdot \alpha)$
  **return** $\hat{y}$

**Algorithm 3** Data synthesis using DISTILL-CF

**Input:** User set $\mathcal{U}$ ; dataset $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$; NTK $\mathbb{K} : \mathbb{R}^{|\mathcal{I}|} \times \mathbb{R}^{|\mathcal{I}|} \mapsto \mathbb{R}$ ; support user size $\mu \in \mathbb{R}$ ; gumbel softmax temperature $\tau \in \mathbb{R}$ ; reg. const. $\lambda_2 \in \mathbb{R}$ ; SGD batch-size $b$, step-size $\eta \in \mathbb{R}$
**Output:** Synthesized data $\mathbf{X^s} \in \mathbb{R}^{\mu\times|\mathcal{I}|}$

**procedure** SAMPLE($n, \mathcal{U}, \mathbf{X}$)
  $\mathcal{U}' \sim \mathcal{U}$ ▷ Randomly sample $n$ users from $\mathcal{U}$
  $\mathbf{X}' \leftarrow \mathbf{X_u} \quad \forall u \in \mathcal{U}'$ ▷ Retrieve corresponding rows from $\mathbf{X}$
  **return** $\mathcal{U}', \mathbf{X}'$
**procedure** SYNTHESIZE($\mathcal{U}, \mathbf{X}, \mathbb{K}$)
  $\mathcal{U}^s, \mathbf{X^s} \leftarrow$ SAMPLE($\mu, \mathcal{U}, \mathbf{X}$) ▷ Sample support data
  **for** $steps = 0 \ldots \xi$ **do**
    $\hat{\mathbf{X}}^\mathbf{s} \leftarrow \sigma\left[\sum_{i=1}^{\gamma} gumbel_\tau(softmax(\mathbf{X^s}))\right]$
    $\alpha^s \leftarrow$ FIT($\mathcal{U}^s, \hat{\mathbf{X}}^\mathbf{s}, \mathbb{K}$) ▷ Fit $\infty$-AE on support data
    $\mathcal{U}^b, \mathbf{X^b} \leftarrow$ SAMPLE($b, \mathcal{U}, \mathbf{X}$)
    $\tilde{\mathbf{X}} \leftarrow [0]_{b\times|\mathcal{I}|}$
    $\tilde{\mathbf{X}}_\mathbf{u} \leftarrow$ PREDICT($\mathcal{U}^s, \hat{\mathbf{X}}^\mathbf{s}, \mathbf{X^b_u}, \mathbb{K}, \alpha^s) \quad \forall u \sim \mathcal{U}^b$ ▷ Predict for all sampled users
    $L \leftarrow \mathbf{X^b} \cdot log(\tilde{\mathbf{X}}) + (1 - \mathbf{X^b}) \cdot log(1 - \tilde{\mathbf{X}}) + \lambda_2 \cdot ||\hat{\mathbf{X}}^\mathbf{s}||_1$ ▷ Re-construction loss
    $\mathbf{X^s} \leftarrow \mathbf{X^s} - \eta \cdot \dfrac{\partial L}{\partial \mathbf{X^s}}$ ▷ SGD on $\mathbf{X^s}$
  **return** $\mathbf{X^s}$

# B Appendix: Experiments

## B.1 Baselines & Competitor Methods

We provide a high-level overview of the competitor models used in our experiments:

- **PopRec:** This implicit-feedback baseline simply recommends the most *popular* items in the catalog irrespective of the user. Popularity of an item is estimated by their empirical frequency in the logged train-set.
- **Bias-only:** This baseline learns scalar user and item biases for all users and item in the train-set, optimized by solving a least-squares regression problem between the predicted and observed relevance. More formally, given a user $u$ and an item $i$, the relevance is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i$, where $\alpha \in \mathbb{R}$ is a global offset bias, and $\beta_u, \beta_i \in \mathbb{R}$ are the user and item specific biases respectively. This model doesn't consider any cross user-item interactions, and hence lacks expressivity.
- **MF:** Building on top of the bias-only baseline, the Matrix Factorization algorithm tries to represent the users and items in a shared latent space, modeling their relevance by the dot-product of their representations. More formally, given a user $u$ and an item $i$, the relevance

Table 2: Brief set of statistics of the datasets used in this paper.

| Dataset | # Users | # Items | # Interactions | Sparsity |
|---|---|---|---|---|
| **Amazon Magazine [42]** | 3k | 1.3k | 12k | 99.7% |
| **ML-1M [15]** | 6k | 3.7k | 1M | 95.6% |
| **Douban [69]** | 2.6k | 34k | 1.2M | 98.7% |
| **Netflix [5]** | 476k | 17k | 100M | 98.9% |

is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + (\gamma_u \cdot \gamma_i)$, where $\alpha, \beta_u, \beta_i$ are global, user, and item biases respectively, and $\gamma_u, \gamma_i \in \mathbb{R}^d$ represent the learned user and item representations. The biases and latent representations in this model are estimated by optimizing for the Bayesian Personalized Ranking (BPR) loss [49].

- **NeuMF [18]:** As a neural extension to MF, Neural Matrix Factorization aims to replace the linear cross-interaction between the user and item representations with an arbitrarily complex, non-linear neural network. More formally, given a user $u$ and an item $i$, the relevance is predicted as $\hat{r}_{u,i} = \alpha + \beta_u + \beta_i + \phi(\gamma_u, \gamma_i)$, where $\alpha, \beta_u, \beta_i$ are global, user, and item biases respectively, $\gamma_u, \gamma_i \in \mathbb{R}^d$ represent the learned user and item representations, and $\phi : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is a neural network. The parameters for this model are again optimized using the BPR loss.
- **MVAE [32]:** This method proposed using variational auto-encoders for the task of collaborative filtering. Their main contribution was to provide a principled approach to perform variational inference for the task of collaborative filtering.
- **LightGCN [17]:** This simplistic Graph Convolution Network (GCN) framework removes all the steps in a typical GCN [28], only keeping a linear neighbourhood aggregation step. This *light* model demonstrated promising results for the collaborative filtering scenario, despite its simple architecture. We use the official public implementation[3] for our experiments.
- **EASE [59]:** This linear model proposed doing ordinary least squares regression by estimating an item-item similarity matrix, that can be viewed as a zero-depth auto-encoder. Performing regression gives them the benefit of having a closed-form solution. Despite its simplicity, EASE has been shown to out-perform most of the deep non-linear neural networks for the task of collaborative filtering.

### B.2 Sampling strategies

Given a recommendation dataset $\mathcal{D} := \{(\text{user}_i, \text{item}_i, \text{relevance}_i)\}_{i=1}^n$ consisting of $n$ user-item interactions defined over a set of users $\mathcal{U}$, set of items $\mathcal{I}$, and operating over a binary relevance score (implicit feedback); we aim to make a $p\%$ sub-sample of $\mathcal{D}$, defined in terms of number of interactions. Below are the different sampling strategies we used in comparison with DISTILL-CF:

- **Interaction-RNS:** Randomly sample $p\%$ interactions from $\mathcal{D}$.
- **User-RNS:** Randomly sample a user $u \sim \mathcal{U}$, and add all of its interactions into the sampled set. Keep repeating this procedure until the size of sampled set is less than $\frac{p \times n}{100}$.
- **Head user:** Sample the user $u$ from $\mathcal{U}$ with the most number of interactions, and add all of its interactions into the sampled set. Remove $u$ from $\mathcal{U}$. Keep repeating this procedure until the size of sampled set is less than $\frac{p \times n}{100}$.
- **SVP-CF user [57]:** This coreset mining technique proceeds by first training a proxy model on $\mathcal{D}$ for $e$ epochs. SVP-CF then modifies the forgetting events approach [61], and counts the inverse AUC for each user in $\mathcal{U}$, averaged over all $e$ epochs. Just like head-user sampling, we now iterate over users in the order of their forgetting count, and keep sampling users until we exceed our sampling budget of $\frac{p \times n}{100}$ interactions. We use the bias-only model as the proxy.

[3] https://github.com/gusye1234/LightGCN-PyTorch

Table 3: List of all the hyper-parameters grid-searched for $\infty$-AE, DISTILL-CF, and baselines.

| Hyper-Parameter | Model | Amz. Magazine | ML-1M | Douban | Netflix |
|---|---|---|---|---|---|
| Latent size | MF<br>NeuMF<br>LightGCN<br>MVAE | {4, 8, 16, 32, 50, 64, 128} | | | |
| # Layers | MF<br>NeuMF<br>LightGCN<br>MVAE | {1, 2, 3} | | | |
| | $\infty$-AE | {1} | | | |
| Learning rate | MF<br>NeuMF<br>LightGCN<br>MVAE | {0.001, 0.006, 0.01} | | | {0.006} |
| | DISTILL-CF | {0.04} | | | |
| Dropout | MF<br>NeuMF<br>LightGCN<br>MVAE | {0.0, 0.3, 0.5} | | | |
| $\lambda$ | EASE | {1, 10, 100, 1K, 10K} | | | |
| | $\infty$-AE | {0.0, 1.0, 5.0, 20.0 50.0, 100.0} | | | |
| | DISTILL-CF | {1e-5, 1e-3, 0.1, 1.0, 5.0, 50.0} | | | |
| $\lambda_2$ | DISTILL-CF | $\frac{\{1e\text{-}3,\ 10.0\}}{\text{avg. \# of interactions per user}}$ | | | |
| $\tau$ | DISTILL-CF | {0.3, 0.5, 0.7, 5.0} | | | |
| $\gamma$ | DISTILL-CF | {50, 100, 200} | {200, 500, 700} | {500, 1K, 2K} | {500, 700} |

### B.3 Data statistics & hyper-parameter configurations

We present a brief summary of statistics of the datasets used in our experiments in Table 2, and list all the hyper-parameter configurations tried for $\infty$-AE, DISTILL-CF, and other baselines in Table 3.

### B.4 Additional training details

We now discuss additional training details about DISTILL-CF that could not be presented in the main text due to space constraints. Firstly, we make use of a validation-set, and evaluate the performance of the $\infty$-AE model trained in DISTILL-CF's inner-loop to perform hyper-parameter search, as well as early exit. Note that we don't validate the inner-loop's $\lambda$ at every outer-loop iteration, but keep changing it on-the-fly at each validation cycle. We notice this trick gives us a much faster convergence compared to keeping $\lambda$ fixed for the entire distillation procedure, and validating for it like other hyper-parameters.

We also discuss the Gumbel sampling procedure described in Equation (3), in more detail. Given the sampling prior matrix $X^s$, that intuitively denotes the importance of sampling a specific user-item interaction, we intend to sample $\hat{X}^s$ which will finally be used for downstream model applications. Note that for each row (user) in $X^s$, we need multiple, but variable number of samples to conform to the Zipfian law for user and item popularity. This requirement in itself rejects the possibility to use top-K sampling which will sample the same number of items for each row. Furthermore, to keep $\hat{X}^s \sim X^s$ sampling part of the optimization procedure, we need to compute the gradients of the logistic objective in Equation (3) with respect to $X^s$, and hence need the entire process to be differentiable. This requirement prohibits the usage of other popular strategies like Nucleus sampling [19], which is non-differentiable. To workaround all the requirements, we devise a multi-step Gumbel sampling strategy where for each row (user) we take a fixed number of Gumbel samples ($\gamma$), with replacement, followed by taking a union of all the sampled user-item interactions. Note that the union

operation ensures that due to sampling with replacement, if a user-item interaction is sampled multiple times, we sample it only once. Hence, the number of sampled interactions is strictly upper-bounded by $\gamma \times |\mathcal{I}|$. To be precise, the sampling procedure is formalized below:

$$\hat{X}^s_{u,i} = \sigma\left[\sum^{\gamma} \frac{exp(\frac{\log(X^s_{u,i})+g_{u,i}}{\tau})}{\sum_{j\in\mathcal{I}} exp(\frac{\log(X^s_{u,j})+g_{u,j}}{\tau})}\right] \quad \text{s.t.} \quad g_{u,i} \sim \text{Gumbel}(\mu=0,\beta=1) \quad \forall u \in \mathcal{U}, i \in \mathcal{I}$$

Where $\sigma$ represents an appropriate function which clamps all values between $0$ and $1$. In our experiments, we use hard-tanh.

### B.5 Evaluation metrics

We now present formal definitions of all the ranking metrics used in this study:

- **AUC:** Intuitively defined as a threshold independent classification performance measure, AUC can also be interpreted as the expected probability of a recommender system ranking a positive item over a negative item for any given user. More formally, given a user $u$ from the user set $\mathcal{U}$ with its set of positive interactions $\mathcal{I}^+_u \subseteq \mathcal{I}$ with a similarly defined set of negative interactions $\mathcal{I}^-_u = \mathcal{I} \backslash \mathcal{I}^+_u$, the AUC for a relevance predictor $\phi(u,i)$ is defined as:

$$\text{AUC}(\phi) \coloneqq \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\mathop{\mathbb{E}}_{i^+\sim\mathcal{I}^+_u}\left[\mathop{\mathbb{E}}_{i^-\sim\mathcal{I}^-_u}\left[\phi(u,i^+) > \phi(u,i^-)\right]\right]\right]$$

- **HitRate (HR@k):** Another name for the recall metric, this metric estimates how many positive items are predicted in a top-k recommendation list. More formally, given recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$ and the set of positive interactions $\mathcal{I}^+_u \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$:

$$\text{HR@k} \coloneqq \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\frac{|\hat{Y}_u \cap \mathcal{I}^+_u|}{|\mathcal{I}^+_u|}\right]$$

- **Normalized Discounted Cumulative Gain (nDCG@k):** Unlike HR@k which gives equal importance to all items in the recommendation list, the nDCG@k metric instead gives a higher importance to items predicted higher in the recommendation list and performs logarithmic discounting further down. More formally, given *sorted* recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$ and the set of positive interactions $\mathcal{I}^+_u \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$:

$$\text{nDCG@k} \coloneqq \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\frac{\text{DCG}_u}{\text{IDCG}_u}\right] \;;\; \text{DCG}_u \coloneqq \sum_{i=1}^{k} \frac{\hat{Y}^i_u \in \mathcal{I}^+_u}{log_2(i+1)} \;;\; \text{IDCG}_u \coloneqq \sum_{i=1}^{|\mathcal{I}^+_u|} \frac{1}{log_2(i+1)}$$

- **Propensity-scored Precision (PSP@k):** Originally introduced in [22] for extreme classification scenarios [46, 21, 37], the PSP@k metric intuitively accounts for missing labels (items in the case of recommendation) by dividing the true relevance of an item (binary) with a propensity correction term. More formally, given recommendation lists $\hat{Y}_u \subseteq \mathcal{I}^K \quad \forall u \in \mathcal{U}$, the set of positive interactions $\mathcal{I}^+_u \subseteq \mathcal{I} \quad \forall u \in \mathcal{U}$, and a propensity model $\phi : \mathcal{I} \mapsto \mathbb{R}$:

$$\text{PSP@k} \coloneqq \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\frac{\text{uPSP}_u}{\text{mPSP}_u}\right] \;;\; \text{uPSP}_u \coloneqq \frac{1}{k}\cdot\sum_{i=1}^{k} \frac{\hat{Y}^i_u \in \mathcal{I}^+_u}{\phi(\hat{Y}^i_u)} \;;\; \text{mPSP}_u \coloneqq \sum_{i\in\mathcal{I}^+_u} \frac{1}{\phi(i)}$$

  For $\phi$, we adapt the propensity model proposed in [22] for the case of recommendation as:

$$\phi(i) \equiv \mathop{\mathbb{E}}_{u\sim\mathcal{U}}\left[\text{P}(r_{u,i}=1|r^*_{u,i}=1)\right] = \frac{1}{1+C\cdot e^{-A\cdot log(n_i+B)}} \;;\; C = (logN-1)\cdot(B+1)^A$$

  Where, $N$ represents the total number of interactions in the dataset, and $n_i$ represents the empirical frequency of item $i$ in the dataset. We use $A = 0.55$ and $B = 1.5$ for our experiments.

### B.6 Additional experiments

**How does depth affect $\infty$-AE?** To better understand the effect of depth on an infinitely-wide auto-encoder's performance for recommendation, we extend $\infty$-AE to multiple layers and note its downstream performance change in Figure 6. The prominent observation is that models tend to get *worse* as they get deeper, with generally good performance in the range of $1-2$ layers, which also has been common practical knowledge even for finite-width recommender systems.

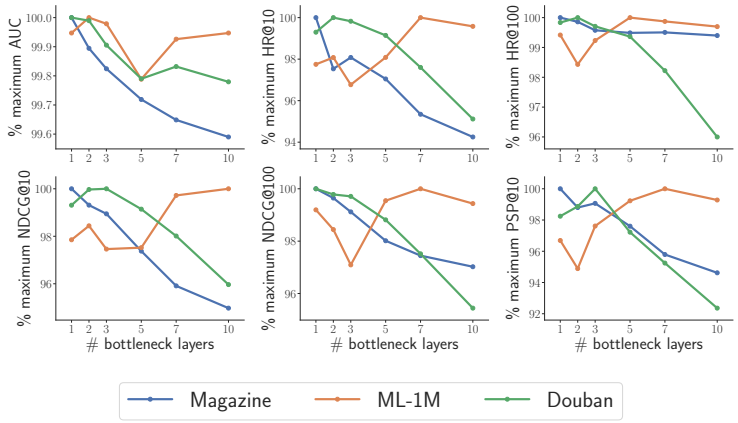


Figure 6: Performance of $\infty$-AE with varying depths. The y-axis represents the normalized metric *i.e.* performance relative to the best depth for a given metric.

**How does $\infty$-AE perform on cold users & cold items?** Cold-start has been one of the hardest problems in recommender systems — how to best model users or items that have very little training data available? Even though $\infty$-AE doesn't have any extra modeling for these scenarios, we try to better understand the performance of $\infty$-AE over users' and items' coldness spectrum. In Figure 7, we quantize different users and items based on their coldness (computed by their empirical occurrence in the train-set) into equisized buckets and measure different models' performance only on the binned users or items. We note $\infty$-AE's dominance over other competitors especially over the tail, head-users; and torso, head-items.

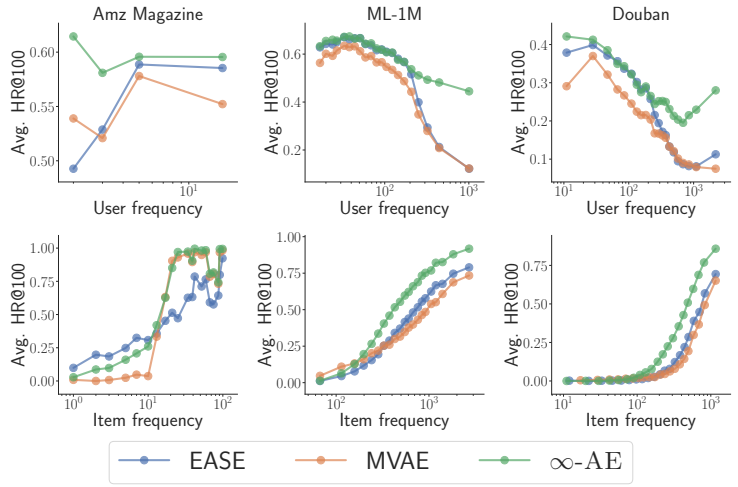


Figure 7: Performance comparison of $\infty$-AE with SoTA finite-width models stratified over the coldness of users and items. The y-axis represents the average HR@100 for users/items in a particular quanta. All user/item bins are equisized.

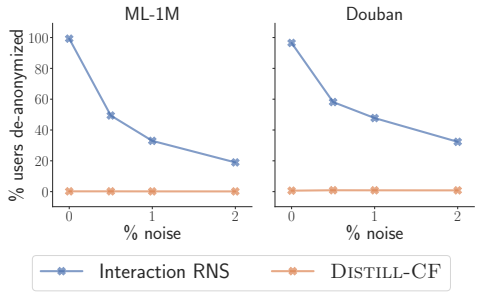


Figure 8: Amount of noise added in $\mathcal{D}'$ *vs.* % of users de-anonymized.

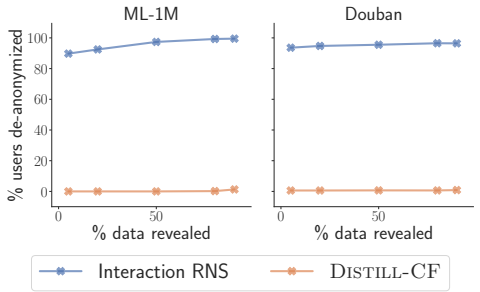


Figure 9: Amount of data revealed in $\mathcal{D}'$ *vs.* % of users de-anonymized.

**How anonymized is the data synthesized by DISTILL-CF?** Having evaluated the fidelity of distills generated using DISTILL-CF, we now focus on understanding its anonymity and syntheticity. For the generic data down-sampling case, the algorithm presented in [38] works well to de-anonymize the Netflix prize dataset. The algorithm assumes a complete, non-PII dataset $\mathcal{D}$ along with an incomplete, noisy version of the same dataset $\mathcal{D}'$, but also has the sensitive PII available. We simulate a similar setup but extend to datasets other than Netflix, by following a simple down-sampling and noise addition procedure: given a sampling strategy $s$, down-sample $\mathcal{D}$ and add $x\%$ noise by randomly flipping $x\%$ of the total items for each user to generate $\mathcal{D}'$. We then use our implementation of the algorithm proposed in [38] to *match* the corresponding users in the original, noise-free dataset $\mathcal{D}$.

However, if instead of a down-sampled dataset $\mathcal{D}'$, a data distill of $\mathcal{D}$ (using DISTILL-CF), let's say $\tilde{\mathcal{D}}$, is made publicly available. The task of de-anonymization can no longer be carried out by simply *matching* user histories from $\mathcal{D}'$ to $\mathcal{D}$, since $\mathcal{D}$ is no longer available. The only solution now is to *predict* the missing items in $\mathcal{D}'$. Note that this this task is easier than the usual recommendation problem, as the user histories to complete in $\mathcal{D}'$ do exist in some incoherent way in the data distill $\tilde{\mathcal{D}}$, and is more similar to train-set prediction. To test this out, we formulate a simple experiment: given a data distill $\tilde{\mathcal{D}}$, an incomplete, noisy subset $\mathcal{D}'$ with PII information, and also hypothetically the number of missing items for each user in $\mathcal{D}'$ — how accurately can we predict the *exact* set of missing items in $\mathcal{D}'$ using an $\infty$-AE model trained on $\tilde{\mathcal{D}}$.

We perform experiments for both the cases of data-sampling and data-distillation. In Figure 8, we measure the % of users de-anonymized using the aforementioned procedures. We interestingly note no level of de-anonymization with the data-distill, even if there's no noise in $\mathcal{D}'$. We also note the expected observation for the data-sampling case: less users are de-anonymized when there's more noise in $\mathcal{D}'$. In Figure 9, we now control the amount of data revealed in $\mathcal{D}'$. We again note the same observation: even with $90\%$ of the correct data from $\mathcal{D}$ revealed in $\mathcal{D}'$ with $0\%$ of noise, we still note a very tiny $0.86\%$ of user de-anonymization with data-distillation, whereas $96.43\%$ with data-sampling for the Douban dataset.

**How does DISTILL-CF compare to data augmentation approaches?** We compare the quality of data synthesized by DISTILL-CF with generative models proposed for data augmentation. One such SoTA method is AR-CF [9], which leverages two conditional GANs [36] to generate fake users and fake items. For our experiment, we focus only on AR-CF's user generation sub-network and consequently train the EASE [59] model *only* on these synthesized users, while testing on the original test-set for the MovieLens-1M dataset. We plot the results in Figure 11, comparing the amount of users synthesized according to different strategies and plot the HR@10 of the correspondingly trained model. The results signify that training models only on data synthesized by data augmentation models is impractical, as these users have only been optimized for being *realistic*, whereas the users synthesized by DISTILL-CF are optimized to be *informative for model training*. The same observation tends to hold true for the case of images as well [67].

**Additional experiments on the generality of data summaries synthesized by DISTILL-CF.** Continuing the results in Figure 2, we now train and evaluate MVAE [32] on data synthesized by DISTILL-CF. Note that the inner loop of DISTILL-CF still consists of $\infty$-AE, but we nevertheless

train and evaluate MVAE to test the synthesized data's universality. We re-use the heuristic sampling strategies from Figure 2 for comparison with DISTILL-CF. From the results in Figure 10, we observe similar scaling laws as EASE for the heuristic samplers as well as DISTILL-CF. A notable exception is interaction RNS, that scales like $\infty$-AE. Another interesting observation to note is that training MVAE on the full data performs slightly worse than training MVAE on the same amount of data synthesized by DISTILL-CF. This behaviour validates the re-usability of data summaries generated by DISTILL-CF, because they transfer well to SoTA finite-width models, which were not involved in DISTILL-CF's user synthesis procedure.

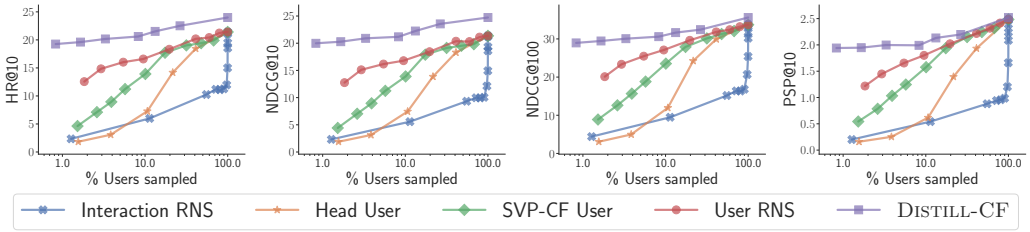


Figure 10: Performance of the MVAE model trained on different amounts of users (log-scale) sampled by different samplers on the ML-1M dataset.

**Additional experiments on Continual learning.** Continuing the results in Figure 5, we plot the results stratified per period for the MovieLens-1M dataset in Figure 12. The results are a little noisy, but we can observe that exemplar data distilled with DISTILL-CF has better performance for a majority of the data periods. Note that we use the official public implementation[4] of ADER.

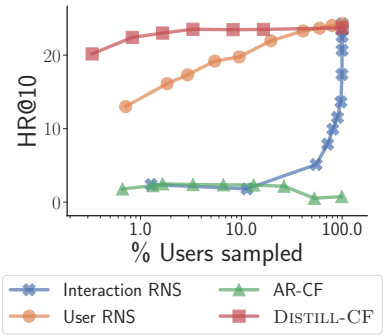


Figure 11: Performance of EASE on varying amounts of data sampled/synthesized using various strategies for the MovieLens-1M dataset.

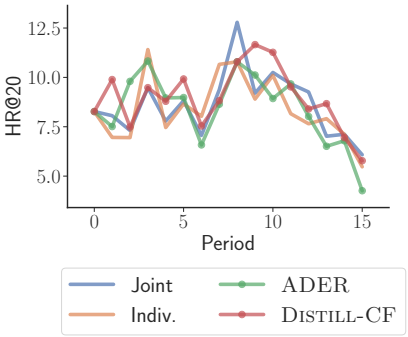


Figure 12: Per-period evaluation of the MVAE model on various continual learning strategies as discussed in Section 5.

**Additional plots for the sample complexity of $\infty$-AE.** In addition to Figure 1 in the main paper, we visualize the sample complexity of $\infty$-AE for all datasets and all metrics in Figure 13. We notice similar trends for all metrics across datasets.

**Additional plots on the robustness of DISTILL-CF & $\infty$-AE to noise.** In addition to Figure 3 and Figure 4 in the main paper, we plot results for the EASE model trained on data sampled by different sampling strategies, when there's varying levels of noise in the original data. We plot this for the MovieLens-1M dataset and all metrics in Figure 13. We notice similar trends for all metrics across datasets. We also plot the sample complexity results for EASE and $\infty$-AE over the MovieLens-1M dataset and all metrics in Figure 15. We observe similar trends across metrics.

[4] https://github.com/doublemul/ADER available with the MIT license.

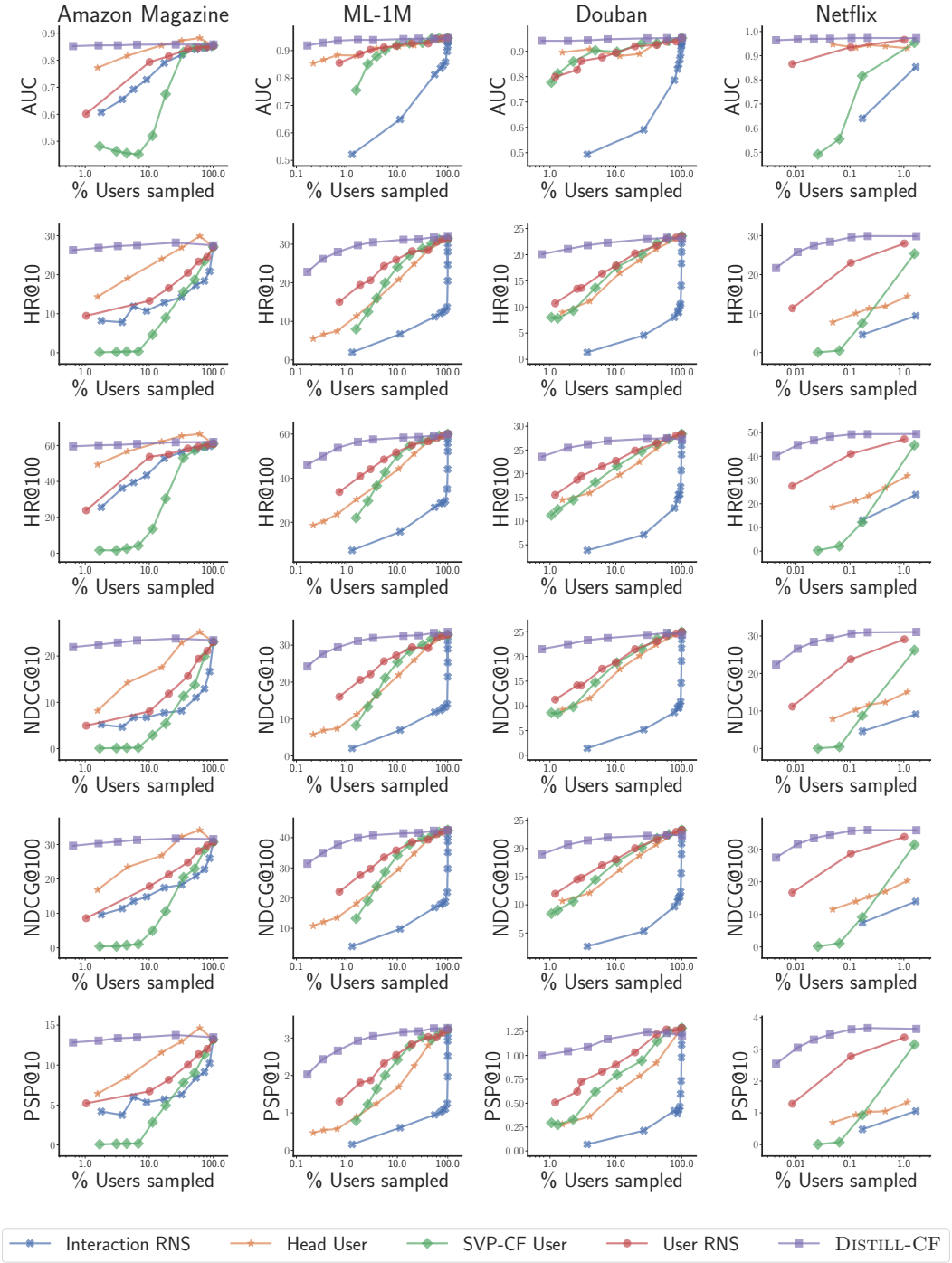


Figure 13: Performance of $\infty$-AE with the amount of users sampled according to different sampling strategies over different metrics. Each column represents a single dataset, and each row represents an evaluation metric.

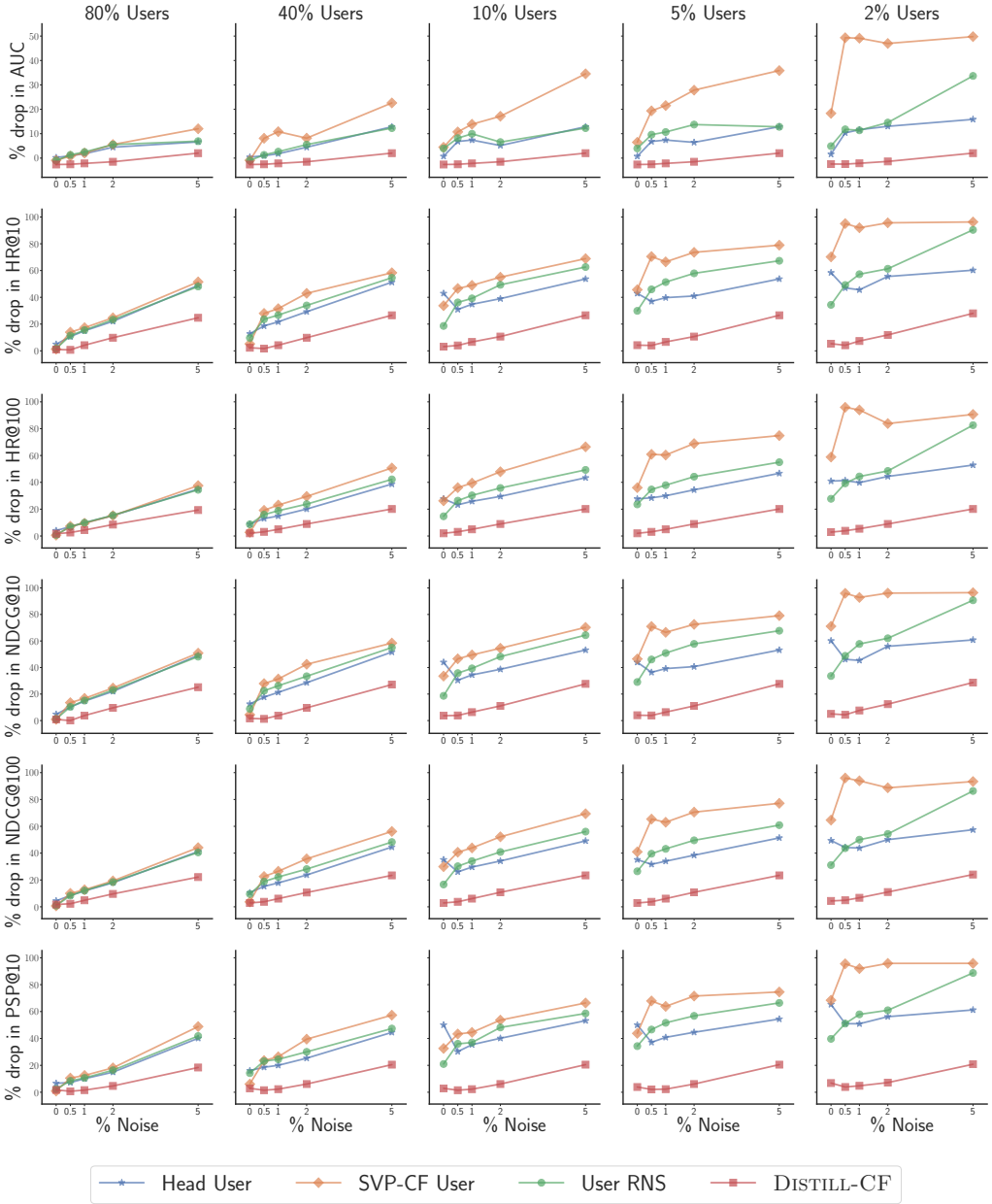


Figure 14: Performance of the EASE model trained on data sampled by different sampling strategies when there's varying levels of noise in the data. Each column represents a user sampling budget, and each row represents the % drop w.r.t a single evaluation metric. All results are on the MovieLens-1M dataset.

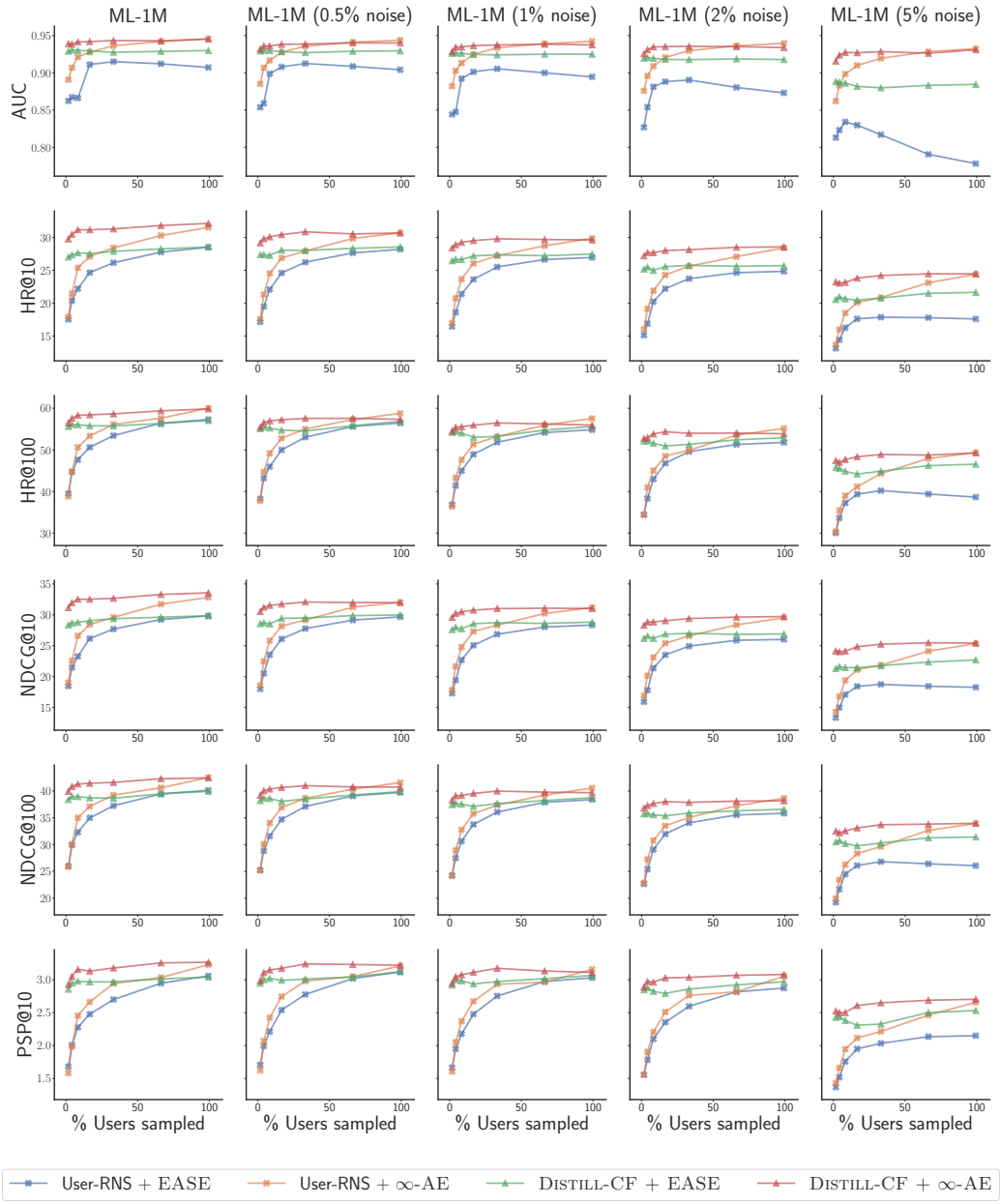


Figure 15: Performance of ∞-AE on data sampled by DISTILL-CF and User-RNS when there's noise in the data. Results for EASE have been added for reference. Each column represents a specific level of noise in the original data, and each row represents an evaluation metric. All results are on the MovieLens-1M dataset.