NEURAL INFORMATION PROCESSING SYSTEMS

Infinite Recommendation Networks

A Data-Centric Approach

Question: Is **more data** what you need for **better recommendation**?

Noveen Sachdeva , Mehak Preet Dhaliwal , Carole-Jean Wu , Julian McAuley

@noveens97

UC San Diego & Meta AI

NeurIPS, 2022

∞-AE

Infinite-width Autoencoder for Recommendation

Premise: Does **stretching the hidden layers** of an autoencoder **till ∞** help in better recommendation?

∞-AE

Primer: Neural Tangent Kernel

- **Infinite-width Correspondence:** Performing Kernelized Ridge Regression with the Neural Tangent Kernel (NTK) emulates the training of an ∞-width NN for an ∞ number of SGD steps.

- For a given neural network architecture $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}$, its corresponding NTK, $\mathbb{K} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is given by:

$$\mathbb{K}(x, x') = \mathop{\mathbb{E}}_{\theta \sim W} \left[\left\langle \frac{\partial f_\theta(x)}{\partial \theta}, \frac{\partial f_\theta(x')}{\partial \theta} \right\rangle \right]$$

- Learning follows a **double-descent** phenomenon

- **Finite-width counterparts** empirically **outperform NTK** for standard image classification tasks

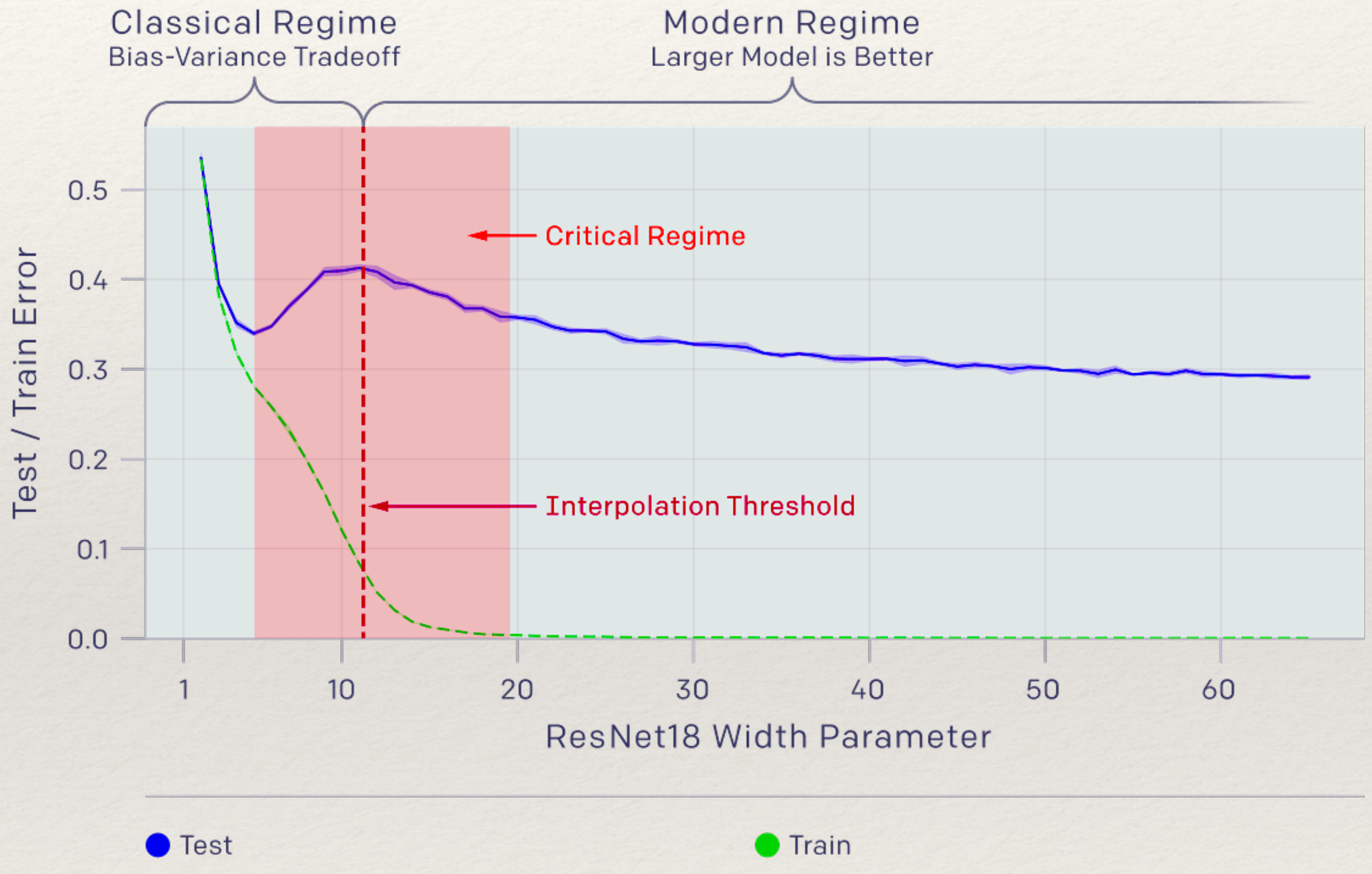


Figure 1: **Credit**: https://openai.com/blog/deep-double-descent/

∞-AE

Methodology

- X_u is the bag-of-items representation for user u i.e. all the items that u interacted with, and we aim to reconstruct it along with **missing user preferences**

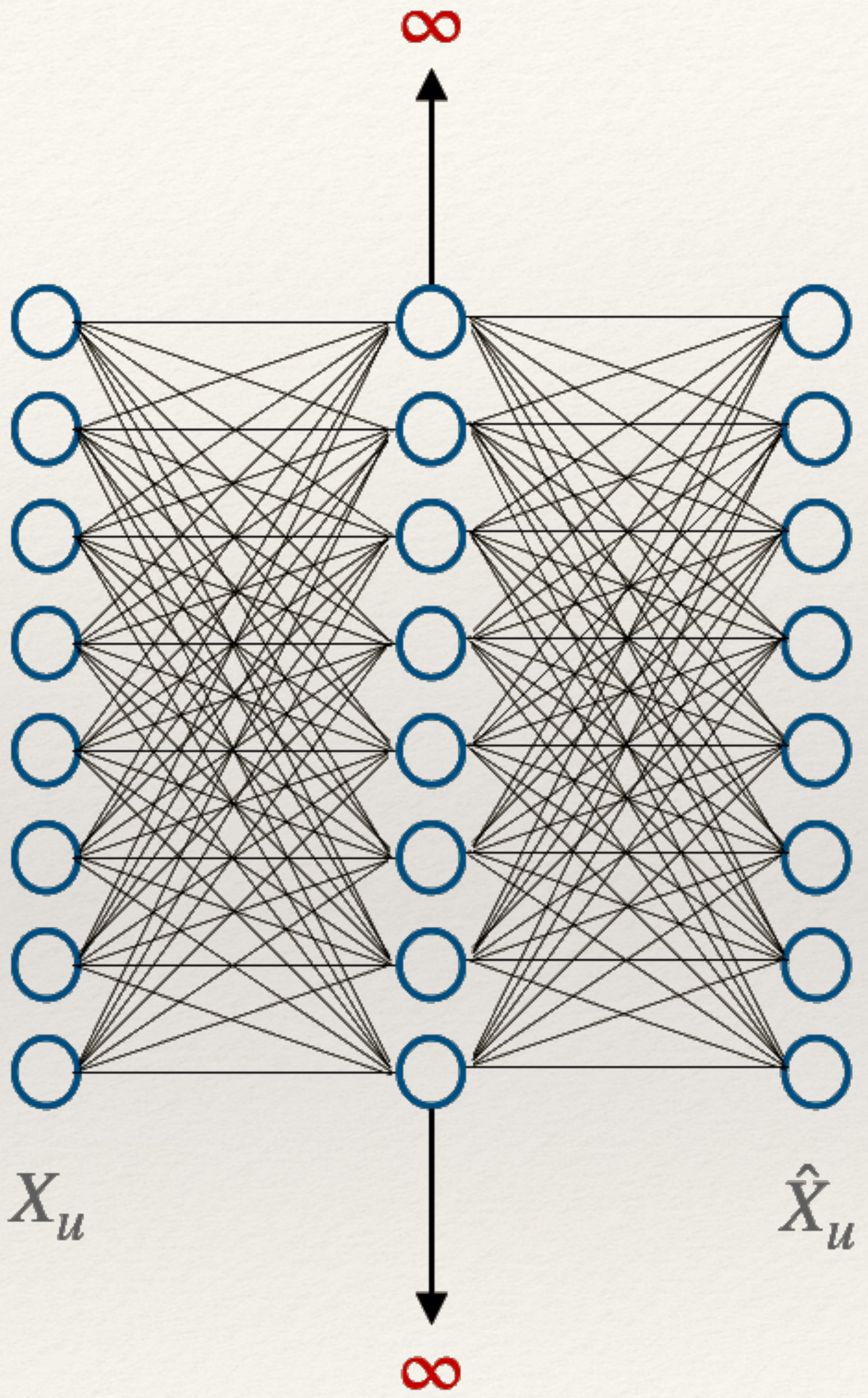
- Due to the infinite-width correspondence, ∞-AE **optimizes in closed-form**:

$$\hat{X} = K \cdot (K + \lambda I)^{-1} \cdot X \quad \text{s.t.} \quad K_{u,v} \triangleq \mathbb{K}(X_u, X_v) \quad \forall u, v$$

- The optimization has only a single **hyper-parameter** λ

- **Time complexity** Training: $\mathcal{O}(U^2 \cdot I + U^{2.376})$ Inference: $\mathcal{O}(U \cdot I)$

- **Memory complexity** Training: $\mathcal{O}(U \cdot I + U^2)$ Inference: $\mathcal{O}(U \cdot I)$



∞-AE

Experiments

Dataset	NeuMF	GCN	MVAE	EASE	∞-AE
Magazine	13.6	22.5	12.1	22.8	**23.0**
ML-1M	25.6	28.8	22.1	29.8	**32.8**
Douban	13.3	16.6	16.1	19.4	**24.9**
Netflix	12.0	—	20.8	26.8	**30.5***

Table 2: nDCG@10 performance (higher is better) of various recommendation algorithms.
* represents training on 5% random users.

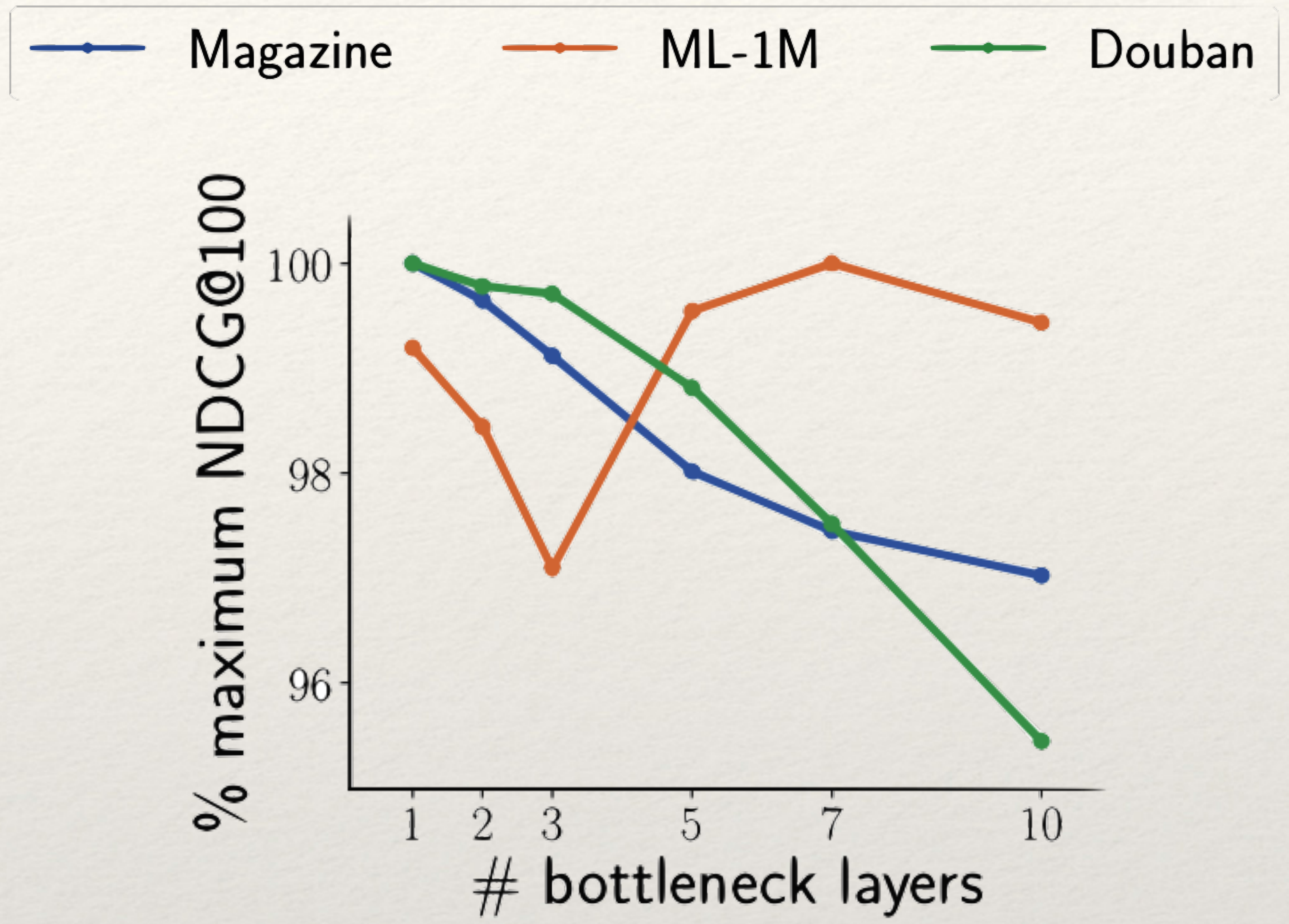


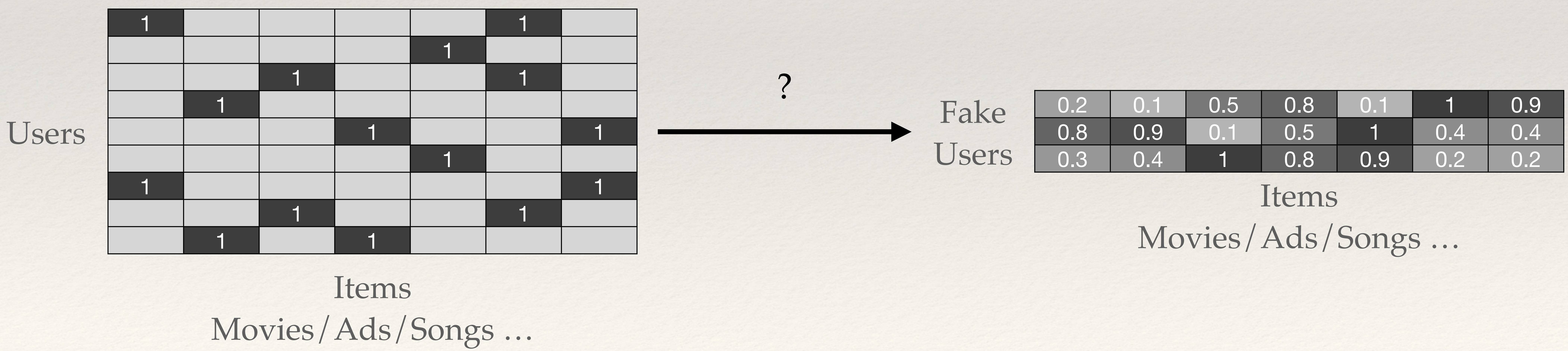
Figure 3: Performance of ∞-AE with varying depth.

- ∞-AE **outperforms** various **state-of-the-art** methods, even when trained on just 5% random users (Netflix)
- **1 layer** seems to be enough for optimal recommendation performance (common folk-knowledge)
- Even though the model is expensive; it is simplistic, easy to implement (thanks, `JAX`), and the performance is great! But, **how to scale it up?** 🤔

Distill-CF

Data Distillation for Collaborative Filtering Data

Premise: Can we **summarize** the massive & sparse user-item matrix into a **terse** data summary?

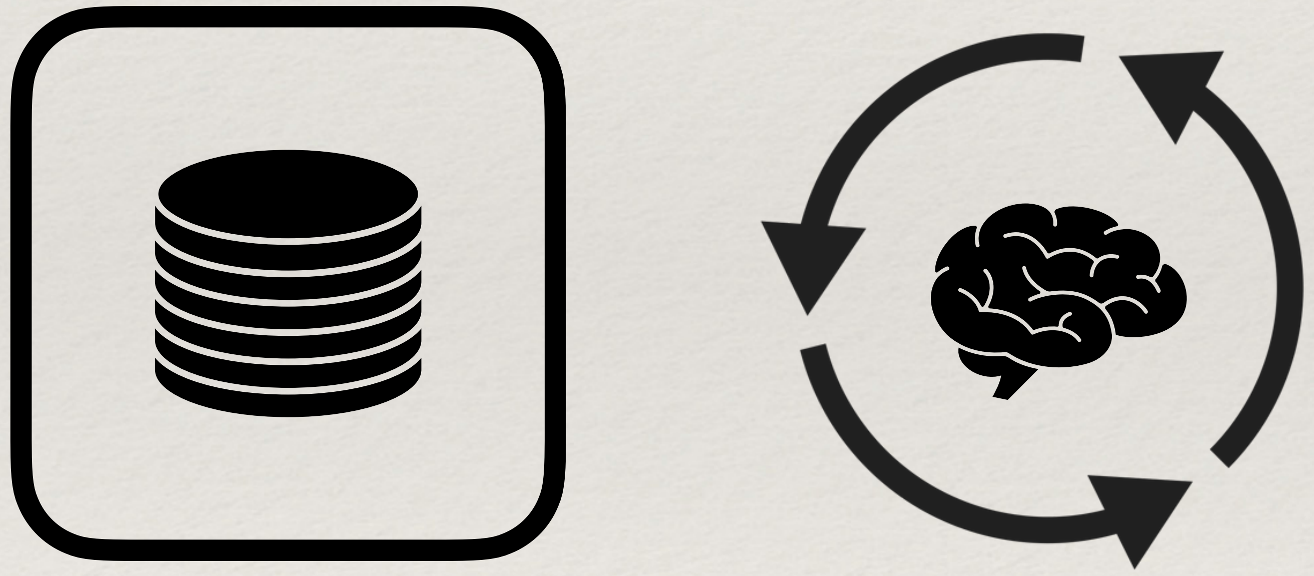


Premise

What is Data-Centric AI?

Model-Centric AI

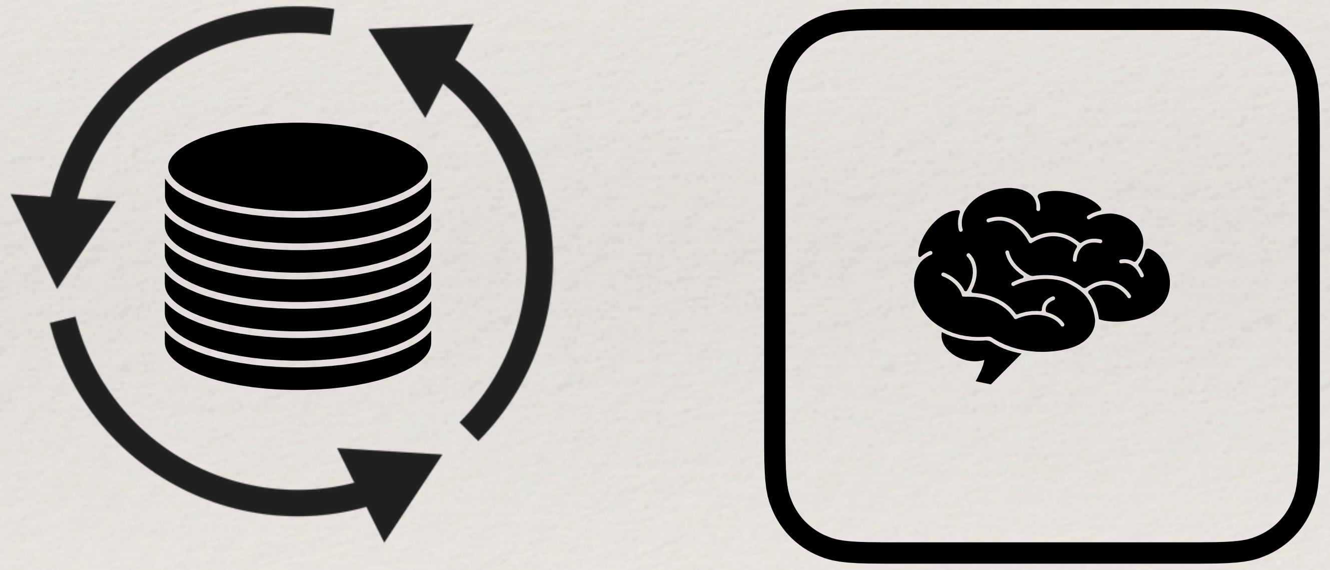
Data Model



Freeze *Improve*

Data-Centric AI

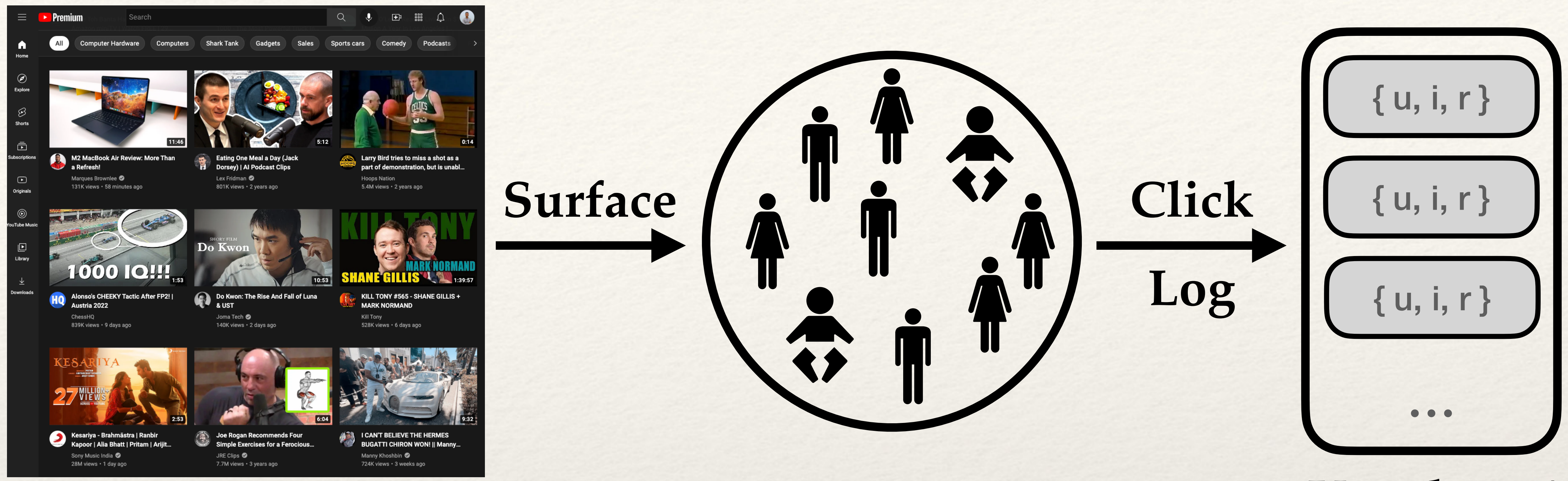
Data Model



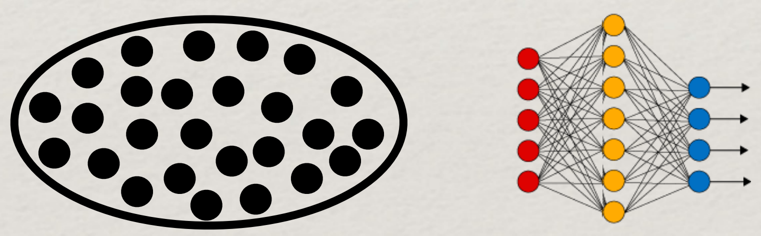
Improve *Freeze*

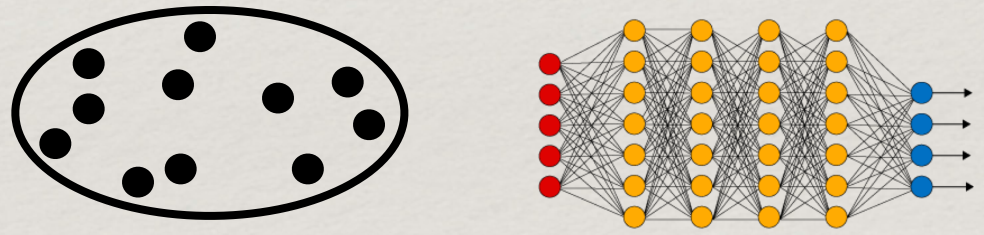
Premise

Why Data-Centric Recommender Systems?



- Unsupervised → large quantities of user-feedback
- Scaling-up systems by scaling-down data
  - Shift focus from data quantity → data "quality"
  - Savings in time, human-effort & environmental resources





Train **simpler** models on large data ← ? → Train **expressive** models on down-sampled data

E.g. Linear modeling, Matrix Factorization, Item-item CF, etc.

E.g. Higher-order modeling, User-user CF, etc.

Distill-CF

Overview & Challenges

<u>Idea</u>: Treat the to-be-synthesized data as **parameters**, and **learn** them through a bilevel optimization.

- Challenges:
  - Data consists of **discrete** (u, i, r) tuples
  - Data is extremely **sparse**
  - **Dynamic** users/item **popularity**
  - Expensive **bilevel optimization**
    - Use ∞-AE for closed-form computation of the inner loop
- **Optimizes** for data-quality rather than quantity

Outer loop — optimize the data summary for a fixed learning algorithm

$$\underset{\mathcal{D}^s}{\arg\min} \underset{(u,i,r)\sim \boxed{\mathcal{D}^{\text{val}}}}{\mathbb{E}} \left[l\left(\phi^*_{\mathcal{D}^s}(u,i), r\right)\right]$$

Original Dataset

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

Optimal recommendation algorithm trained on $\mathcal{D}^s$

Data Summary

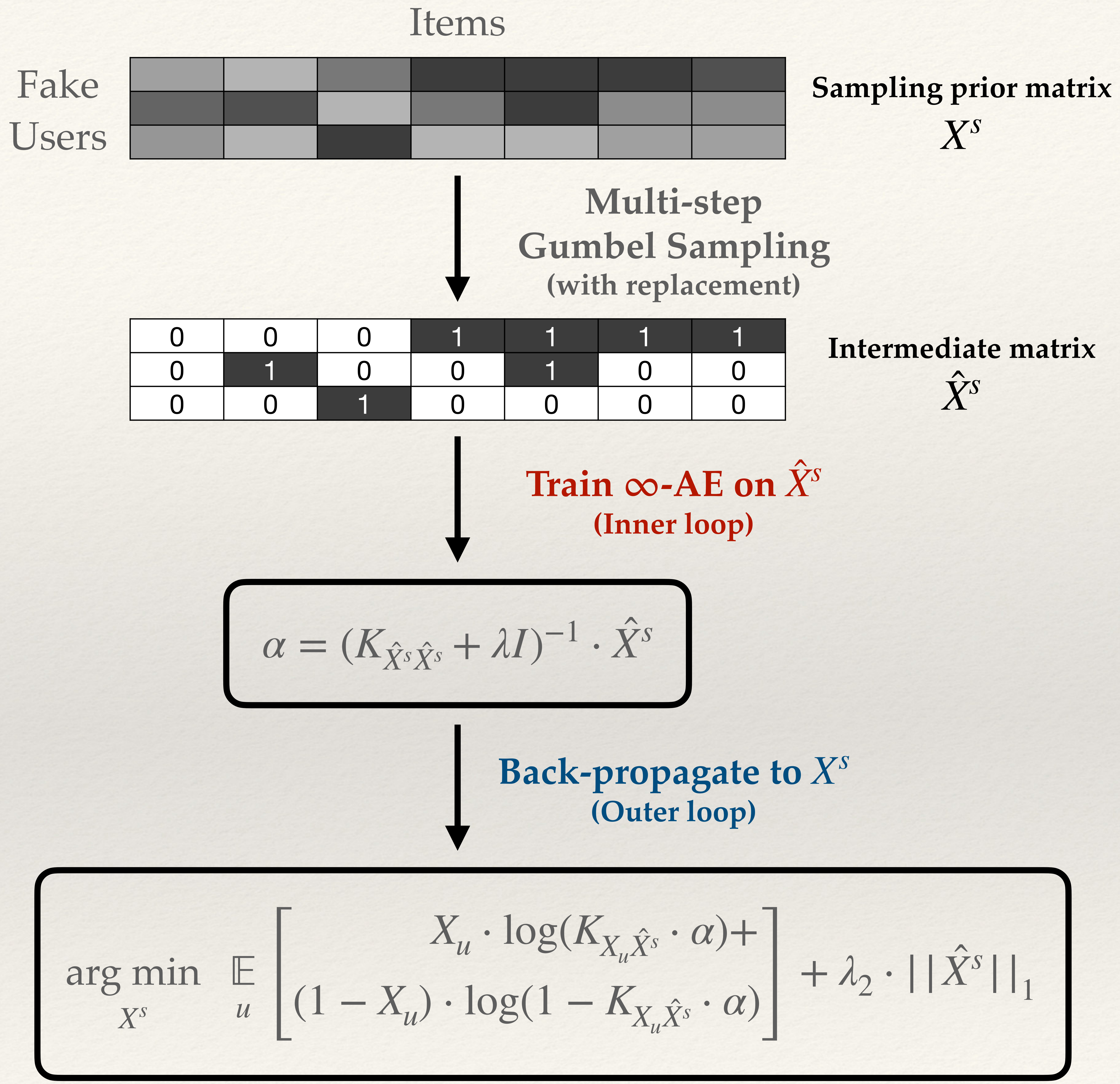
Differentiable cost-function

Inner loop — optimize the learning algorithm for a fixed data summary

Distill-CF

Methodology

- Uses Gumbel sampling on X^s to **mitigate the heterogeneity** of the problem
- Perform Gumbel sampling multiple times for each fake-user to handle **dynamic user/item popularity**
- Automatically **control sparsity** in $\hat{X}^s$ by controlling the **entropy** in X^s



Distill-CF

Experiments

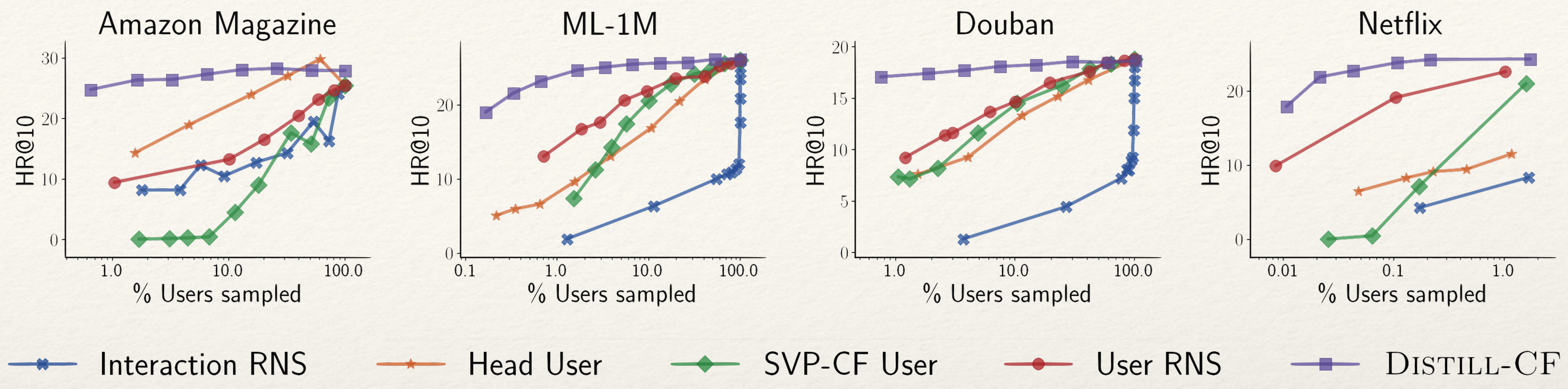


Figure 4: Does Distill-CF outperform other samplers? (Log-scale)

- Using Distill-CF, we can get **96-105%** of full-data performance on as small as **0.1%** data sub-samples, leading to as much as **~1000x** time speedup!
- Distill-CF works well even for the second-best model (EASE), even though the data isn't optimized for it

Dataset	NeuMF	GCN	MVAE	EASE	∞-AE	∞-AE (Distill-CF)
Magazine	13.6	22.5	12.1	22.8	23.0	**23.8**
ML-1M	25.6	28.8	22.1	29.8	**32.8**	32.5
Douban	13.3	16.6	16.1	19.4	**24.9**	24.2
Netflix	12.0	—	20.8	26.8	**30.5***	**30.5**

Table 5: nDCG@10 performance of various recommendation algorithms. * represents training on 5% random users. Distill-CF has a user budget of just 500 (0.1% for Netflix).

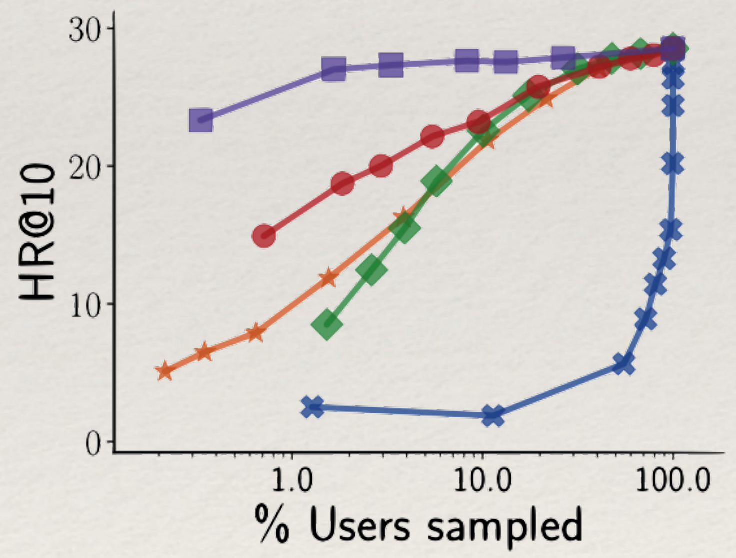


Figure 6: Distill-CF + EASE for the ML-1M dataset.

Distill-CF

Experiments (Contd.)

- Distill-CF is **robust to noise** (even though not optimized for it), and is able to offer significant performance even at high noise ratios and very small support datasets!

- **Less is more**: EASE is more accurate when trained on lesser amounts of data generated by Distill-CF, compared to training on the full-data

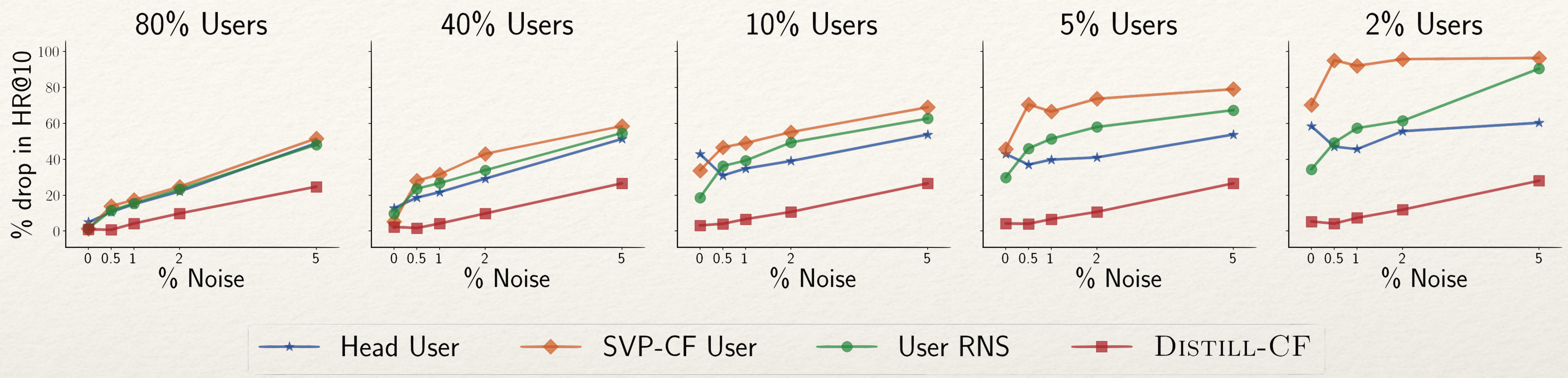


Figure 7: Performance of different samplers when there is noise in the original data.

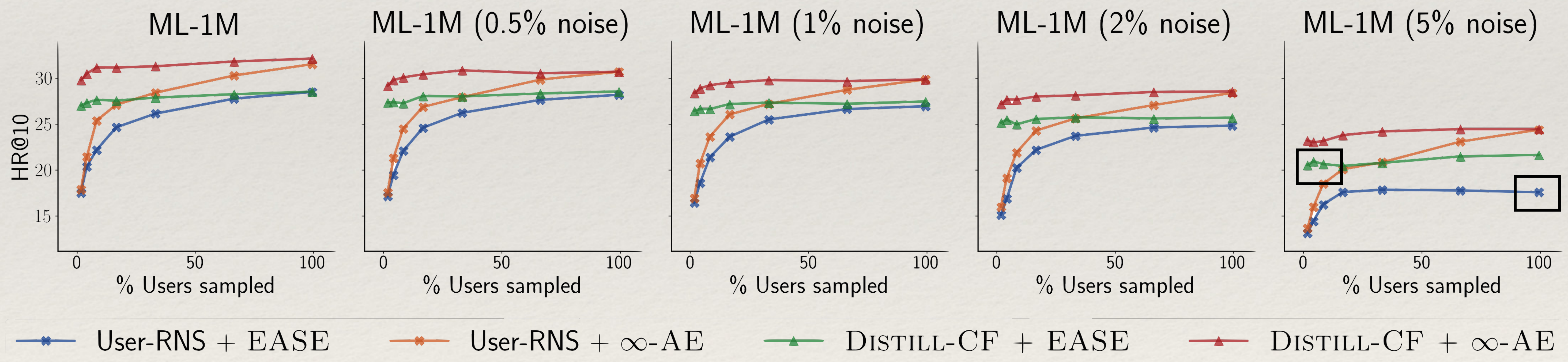


Figure 8: Performance comparison of ∞-AE *vs.* EASE when trained on down-sampled, noisy data.

Thank you!

@noveens97

For paper, code, and these slides:

noveens.com