RecFusion: A Binomial Diffusion Process for 1D Data for Recommendation

Bénédict, Gabriel; Jeunen, Olivier; Papa, Samuele; Bhargav, Samarth; Odijk, Daan; Rijke, Maarten de

DOI
10.48550/arXiv.2306.08947

Publication date
2023

Document Version
Submitted manuscript

RecFusion: A Binomial Diffusion Process for 1D Data for Recommendation

Gabriel Bénédict
University of Amsterdam and RTL Netherlands
g.benedict@uva.nl

Olivier Jeunen
Sharechat
jeunen@sharechat.co

Samuele Papa
University of Amsterdam
s.papa@uva.nl

Samarth Bhargav
University of Amsterdam
s.bhargav@uva.nl

Daan Odijk
RTL Netherlands
daan.odijk@rtl.nl

Maarten de Rijke
University of Amsterdam
m.derijke@uva.nl

arXiv:2306.08947v1 [cs.IR] 15 Jun 2023

Abstract

In this paper we propose RecFusion, which comprise a set of diffusion models for recommendation. Unlike image data which contain spatial correlations, a user-item interaction matrix, commonly utilized in recommendation, lacks spatial relationships between users and items. We formulate diffusion on a 1D vector and propose *binomial diffusion*, which explicitly models binary user-item interactions with a Bernoulli process. We show that RecFusion approaches the performance of complex VAE baselines on the core recommendation setting (top-n recommendation for binary non-sequential feedback) and the most common datasets (MovieLens and Netflix). Our proposed diffusion models that are specialized for 1D and/or binary setups have implications beyond recommendation systems, such as in the medical domain with MRI and CT scans.

1 Introduction

Diffusion models have been profusely used in the image domain [10]. Next to the 2D setup, an increasing amount of research is focused on the 3D domain [20], as well as diffusion on the embedding space [14]. Typical image diffusion models rely on a U-Net [48] architecture with attention layers and process entire images at once. However, image diffusion models rely on and exploit spatial correlations (i.e., between pixels in a localized regions). Settings where these assumptions do not hold are under-explored. In this paper, we consider the recommendation systems domain and, more specifically, 1D binary data in the classical recommendation setting. The recommendation setting is characterized by the following conditions: (i) a user's interaction history is organized like 1D binary data, where columns represent items; and (ii) organized as a matrix for multiple users, each entry in this *interaction matrix* corresponds to the type of interaction between a specific user and item. These interactions can either be *explicit* (ratings, 'likes' or dislikes), or *implicit* (dwell time, clicks, purchases, etc.).

Most modern recommender systems leverage the implicit feedback paradigm, which utilizes data that is not explicitly provided by the user, such as click data, purchase history, browsing behavior. Research in recommender systems employs simpler linear models [24, 31, 46, 47], or neural models, many of which employ the variational autoencoder [28] framework, e.g., cVAE [9], RecVAE [55]

Preprint. Under review.

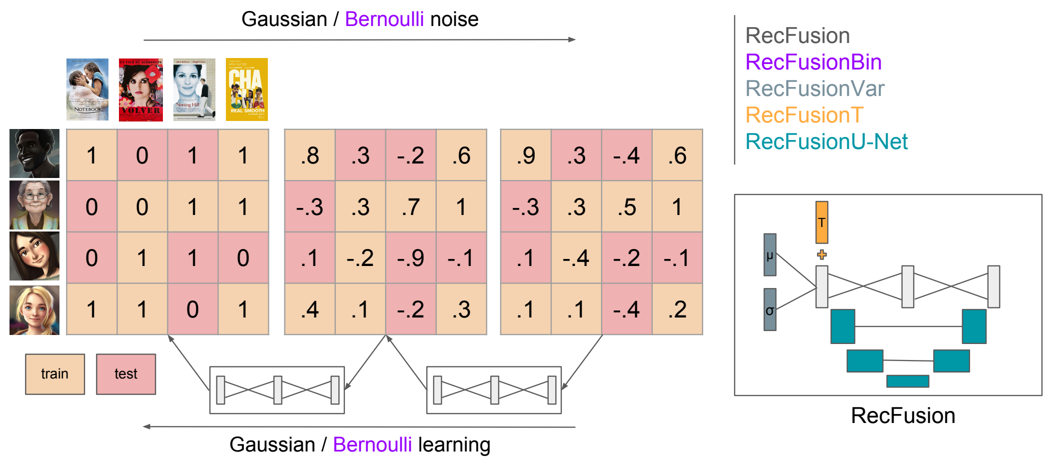


Figure 1: The RecFusion architecture and its variations (user images generated with DALL·E 2).

or the MultVAE [36]. Neural models have benefits beyond recommendation performance, e.g., in controllability / critiquing [33, 38, 64, 68], with some models utilizing disentanglement [2, 17] for this purpose [6, 39, 44]. Beyond controllability, neither non-neural nor VAE-based models can handle time information directly, making it hard to deal with, e.g., preference drift [23], where more recent items may be more relevant for future recommendations. In principle, diffusion models should be able to deal with these recommendation conditions. There have been some initial attempts at modeling recommendation problems using diffusion; CODIGEM [61] defines a diffusion model akin to early attempts in diffusion, with one neural network per diffusion step. We propose RecFusion, a diffusion model inspired by the DDPM [19] architecture adapted for 1D data. We also propose the Bernouilli diffusion process, specifically designed for binary data. We experiment with different common diffusion techniques, such as noise timestep embeddings, modelling the mean and variance and different noise schedules.

Setup. We assume a binary non-sequential top-n implicit feedback setting (see explicit assumptions in Section 2): we seek to predict only the immediate next best item(s) for each user and the time is unknown for any past user-item interaction. The reason for choosing this standard setup is two-fold: (i) by using binary data, we can study the use case of diffusion for binary data, and (ii) we remain comparable with the overwhelming majority of recommendation literature. Indeed, it is common for Assumptions 1–5 that we specify in Section 3.1 to be used.

Main results. As previously shown in the literature, VAE-based models and non-neural models outperform more complex methods in the standard recommendation setting. RecFusion outperforms existing diffusion models for recommendation and opens the way to use guidance and conditioning.

Key contributions of the paper include: (i) we demonstrate uses of diffusion where there is no spatial dependency, (ii) we offer a simple implementation of diffusion that can accomodate any binary and/or 1D data, (iii) we propose modern Variational Auto Encoder (VAE) architectures for recommendation (diffusion models are hierarchical VAEs [29]) (iv) we open the door to diffusion side-effects: guidance and inpainting, and (v) our code is open and available at `https://github.com/gabriben/recfusion`, implemented using a reproducible and well-tested framework to facilitate follow-up work.

2 Method

2.1 Diffusion models

Diffusion models [56] are physics inspired latent variable models that imitate the Brownian motion of molecules. Over time this Brownian motion converges towards a deterministic movement, visible to the human eye. This idea has been extended to the field of probabilistic models, where we are interested in mapping a complex distribution into a tractable one in a reversible way. The diffusion process does exactly this by slowly perturbing the data with a specific kernel. Then, a model is trained to reverse this process. Given the starting variable $\mathbf{X}^0$, the forward diffusion process can be defined using a Markov chain that is used to sample the latent variables $\mathbf{X}^1, \ldots, \mathbf{X}^t, \ldots, \mathbf{X}^T$. We explicitly use the notation $\mathbf{X}$ to differentiate the original 2D user-item matrix composed of user vectors $\mathbf{x}_u$, themselves composed of individual interactions x_{u_i}. In practicalapplications, $\mathbf{X}$ is oftentimes a 2-dimensional matrix representing an image and commonly denoted x [19, 56], of dimension $N \times N$, where N is the square image resolution.

Translated to a machine learning task, diffusion takes an original input $\mathbf{X}^0$ and learns successive disturbed latent representations of the input over discrete timestamps (a Markov chain) $\mathbf{X}^1, \ldots, \mathbf{X}^T$. $\mathbf{X}^1, \ldots, \mathbf{X}^T$ are of the same dimension as $\mathbf{X}^0$ and the final disturbed representation should look completely perturbed: $p\left(\mathbf{X}^T\right) = \mathcal{N}\left(\mathbf{X}^T; \mathbf{0}, \mathbf{I}\right)$. A diffusion model ought to approximate the original input, given that Markov chain process $p_\theta\left(\mathbf{X}^0\right) := \int p_\theta\left(\mathbf{X}^{0:T}\right) d\mathbf{X}^{1:T}$. $p_\theta\left(\mathbf{X}^{0:T}\right)$ corresponds to the *reverse process* starting at the final diffusion step:

$$p_\theta\left(\mathbf{X}^{0:T}\right) := p\left(\mathbf{X}^T\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{X}^{t-1} \mid \mathbf{X}^t\right),\; p_\theta\left(\mathbf{X}^{t-1} \mid \mathbf{X}^t\right) := \mathcal{N}\left(\mathbf{X}^{t-1}; \boldsymbol{\mu}_\theta\left(\mathbf{X}^t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{X}^t, t\right)\right). \quad (1)$$

In order to obtain latent representations at each step (x_t) in the first place, we first diffuse the original input via the *forward process*. We add Gaussian noise (in the next section we show our contribution with an alternative binomial noise) as a proxy to the Brownian motion observed in molecular physics:

$$q\left(\mathbf{X}^{1:T} \mid \mathbf{X}^0\right) := \prod_{t=1}^{T} q\left(\mathbf{X}^T \mid \mathbf{X}^{t-1}\right), \quad q\left(\mathbf{X}^T \mid \mathbf{X}^{t-1}\right) := \mathcal{N}\left(\mathbf{X}^T; \sqrt{1-\beta^t}\mathbf{X}^{t-1}, \beta^t \mathbf{I}\right). \quad (2)$$

The variance schedule $\beta_1, \ldots, \beta^t$ is either learned or follows a predetermined schedule (increasing, decreasing or constant). The optimization of the backwards diffusion process follows the classical Evidence Lower Bound (ELBO) formulation:

$$\mathbb{E}_q[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{X}^T \mid \mathbf{X}^0\right) \| p\left(\mathbf{X}^T\right)\right)}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{X}^{t-1} \mid \mathbf{X}^T, \mathbf{X}^0\right) \| p_\theta\left(\mathbf{X}^{t-1} \mid \mathbf{X}^T\right)\right)}_{L_{t-1}} - \underbrace{\log p_\theta\left(\mathbf{X}^0 \mid \mathbf{X}^1\right)}_{L_0}], \quad (3)$$

where D_{KL} is the KL divergence between each forward process step and its reconstructed representation in the backwards process. Both are assumed identity-Gaussian distributed and are thus estimated with a closed form KL divergence expression. Given that any diffusion step t is an accumulation of identity-normal noise, any $\mathbf{X}_t$ is directly tractable in closed form. It is thus possible to sample only some of the forward diffusion steps $\mathbf{X}_t$ to learn the backwards process.

2.2 A binomial instead of a Gaussian forward process

As one of our contributions, we formalize a binomial (Bernoulli) Markov diffusion process to fit the binomial input data. Intuitively, this corresponds to performing bit flips over diffusion time steps in the forward process and predicting these bit flips in the reverse process. The latter is defined with

$$p_\theta\left(\mathbf{X}^{t-1} \mid \mathbf{X}^t\right) := \mathcal{B}\left(\mathbf{X}^{t-1}; \pi_\theta\left(\mathbf{X}^t, t\right)\right), \quad (4)$$

where $\mathcal{B}(u; \pi)$ is the distribution for a single Bernoulli trial (bit flip), with $u = 1$ occurring with probability π, and $u = 0$ occurring with probability $1 - \pi$. The forward process is a flip of the original $\{0, 1\}$ bits with increasing chance, determined by the schedule β_t:

$$q\left(\mathbf{X}^t \mid \mathbf{X}^{t-1}\right) := \mathcal{B}\left(\mathbf{X}^t; \mathbf{X}^{t-1}\left(1-\beta_t\right) + 0.5\beta_t\right). \quad (5)$$

For the loss function we use a Binary Cross Entropy (BCE) ELBO formulation (Eq. 3) for L_0:

$$L_0 = -\log p_\theta\left(\mathbf{X}^0 \mid \mathbf{X}^1\right) = \mathbf{X}^1 \odot \log \mathbf{X}^0 + \left(1 - \mathbf{X}^1\right) \odot \log\left(1 - \mathbf{X}^0\right). \quad (6)$$

We use $\odot$ as the sign for element-wise multiplication. The traditional ELBO loss relies on the KL divergence. We demonstrate that the KL divergence is also suited for binary data, using Fano's inequality [52] (see Appendix A). Additionally, we derive the bernoulli Markov process and verify that a combination (multiplication) of Bernoulli distributions still a bernoulli distribution (see Appendix B). This guaranties that we can use the Gaussian Markov process properties.

2.3 RecFusion – Recommendation systems as diffusion models

In the image domain, $\mathbf{X}^0$ is of dimension $N \times N$, where N is the image resolution.[1] In the recommendation setup, $\mathbf{X}^0$ is the full user-item matrix.

Full-matrix. In our recommendation setting, we can consider the entire user-item matrix as $\mathbf{X}^0$ of dimension $U \times I$, where U is the number of users and I is the number of items. Each cell in that matrix is a binary representation of the feedback of a user on an item (e.g., for MovieLens [16], x_{ui} is 1 for ratings above 3 stars and 0 otherwise, following [36]). We are thus framing our setting as non-sequential recommendation with binary feedback (see Section 3)). With ever-growing user-item matrices, it quickly becomes infeasible to perform in-memory computations. The solution for image diffusion models is to use a first diffusion model for a say 32×32 images and then use several super-resolution models to upscale it [50]. We consider that a user-item matrix cannot be downscaled by blurring it, because it does not contain hierarchical features, unlike for an image (e.g. an image of a dog probably contains the dog's head, which contains eyes, etc.).

User-batch. Instead, we could think of batches of users $\mathbf{X}^0_{u \in b_j}$ $\forall b_j \in \mathbf{b}$.[2] In that case, the input matrix is still big. For example on MovieLens1M, a batch size 200 users leads to a 200×10000 matrix compared to a 32×32 image, but possible to fit in memory. There are two more advantages to feeding by batch. (I) We can now perform gradient descent over several examples of the data instead of just one matrix example, and (II) we can form batches of items of the same category and use that as a downstream task (a.k.a. diffusion guidance [18]. For example, we could batch by movie genre in the case of the MovieLens dataset [16]. This user-batch formulation is still similar to the original 2 dimensional image setting, but assumes relationships between users close together in the matrix if convolution-based models are used. This assumption is unrealistic and we thus think that this is not a viable approach theoretically. We nonetheless verify that assumption empirically with *RecFusionU-Net2D*.

User-by-user. Alternatively, we can use a 1D formulation (batch size of 1), with $\mathbf{x}^0_u$, the vector of all item feedbacks for user u. In that case, we assume no relationships between users. This corresponds to the formulation of MultVAE [36]. With this formulation, the advantages of the user-batch formulation are kept and spatial dependence between users does not need to be assumed. This setting applies to *RecFusion, RecFusionT, RecFusionVar, RecFusionBin, RecFusionU-Net1D*

We use the vector notation $\mathbf{x}$ for the rest of the paper, to refer to $\mathbf{x}_u$, a user vector. Below we look at two practicalities, conditional generation and the fully perturbed recommendation matrix.

2.3.1 Generate from $\mathbf{x}^1$, a simple alternative to conditional generation

In practice, a recommendation platform is interested in finding the top K next items for users (see Assumption 4 in Section 3.1). In a traditional diffusion inference setup, we would start with a completely random recommendation matrix $\mathbf{x}^T$ and generate $\mathbf{x}^0$ iteratively via the backward diffusion pass through the neural network $p_\theta\left(\mathbf{x}^{t-1} \mid \mathbf{x}^t\right)$ over each diffusion time step t. Without any conditioning / guidance / inpainting techniques, the generated matrix $\mathbf{x}^0$ remains the same. We propose a simpler approach: we feed the existing recommendation matrix at inference time as $\mathbf{x}^1$ and perform a single backward diffusion step to $\mathbf{x}^0$. One question remains: what is a completely perturbed matrix?

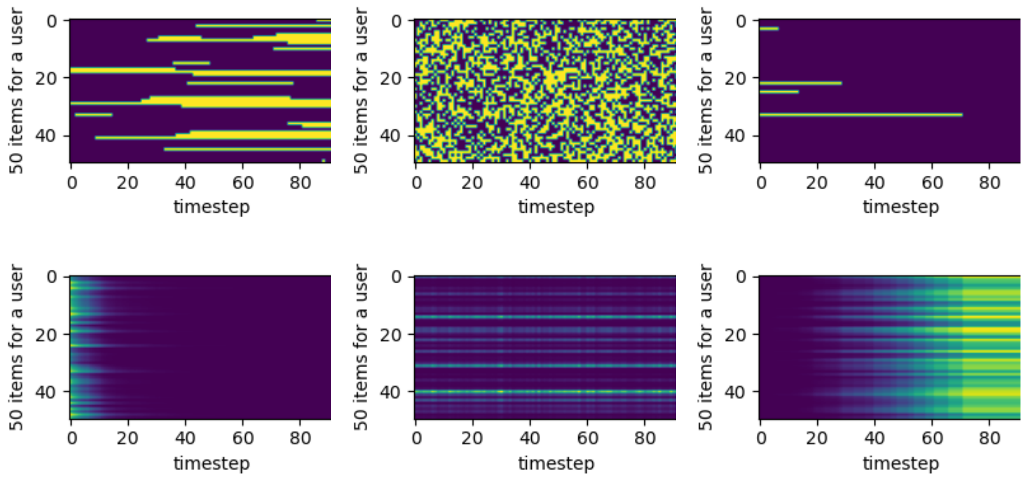
2.3.2 What is a completely perturbed matrix $\mathbf{X}^T$ in the recommendation setting?

What is a completely diffused matrix $\mathbf{X}^T$? Is it (a) a matrix with the same mean activity as the input data $p(\mathbf{X}^T = 1) = E(x^0) = \bar{\mathbf{X}}^0$ (as proposed by Sohl-Dickstein et al. [56]) (b) a matrix with a fair coin flip activity $p(\mathbf{X}^T = 1) = 0.5$ – in the binomial case $\mathcal{B}(\mathbf{X}^T; 0.5)$ – or (c) a matrix full of zero values $p(\mathbf{X}^T = 1) = 0$. We show these three alternatives in Figure 2 with a Bernouilli diffusion example.

With (a) and (b), we experiment with allowing bit flips from 0 to 1 and from 1 to 0, by formulating $p_\theta = \mathcal{B}\left(x^t; \beta_t\right)$ and $x^t = (1 - p_\theta) \cdot x^{t-1} + p_\theta \cdot (1 - x^{t-1})$. For (a), we use a fixed schedule of $\beta_t = 0.01$ $\forall t$. The reverse diffusion process is able to pick up a signal. For (b), we use a fixed

[1] See Section 2.1 our choice of notation.

[2] Batching by items is also possible, but would rather fit the domain of item-based collaborative filtering [51].



(a) a user vector with same mean activity as the input data $p(x^T = 1) = E(x^0) = \bar{\mathbf{X}}^0$

(b) a user vector with a coin flip activity $p(x^T = 1) = 0.5$

(c) a user vector full of zero values $p(x^T = 1) = 0$

Figure 2: Binomial diffusion on MovieLens100k after 20 epochs. Top row is the Bernouilli forward process and the bottom row is the learned reverse process. Blue is closer to 0; yellow is closer to 1.

schedule of $\beta_t = 0.5\ \forall t$. Right away, the user vector becomes chaotic and no real signal is picked up by the reverse diffusion process. With (c), we only allow bit flips from 1 to 0 and end up with with a matrix full of zeroes. For that we let $p_\theta = \mathcal{B}\left(\mathbf{X}^t; \mathbf{X}^{t-1}(1-\beta_t) + 0.5\beta_t\right)$ and $x^t = p_\theta \cdot x^t$. Again, the reverse diffusion process picks up a signal. We found (c) to work best in practice. We conjecture that this is because bit flips only go in one directions and that this information flows more smoothly in the gradient descent steps.

2.3.3 Architecture

We propose a few different architectures for RecFusion, in order of complexity. *RecFusion*, a three layer fully connected network with tanh activation. *RecFusionT* with a time step embedding: we first tried to use the time embedding as in the original attention paper [60], namely feeding the time embedding to the output of the MLP $f(x) + Z_t$. This was not very successful. Instead we fed the time embedding in a DDPM [19] manner (we are not sure if this practice emerged in DDPM or before): $f(x + Z_t)$. *RecFusionVar*, which predicts mean and variance of diffusion steps like in DDPM [19]. *RecFusionBin* our own 1D Bernoulli diffusion model: forward steps as described in Section 2.3.2 (c), reverse steps with a *RecFusion* architecture but a sigmoid final activation and our own BCE ELBO loss (Eqn. 3 and 6). *RecFusionU-Net1D* is the original DDPM [19] architecture simplified, with only one channel and flattened on one dimension to allow for user vector input $\mathbf{x}$. *RecFusionU-Net2D* is the original DDPM [19] architecture simplified and only one channel to allow for a user-batch matrix input $\mathbf{X}^0_{u \in b_j}\ \forall b_j \in \mathbf{b}$. Both U-Nets have a time embedding.

Our two Unet architectures are more as proof-of-concepts than theoretically grounded architectures. Some elements of the U-Net architecture make it rather impractical, such as the necessary spatial relationships in the matrix/vector and the necessity for an even-sized matrix/vector input for the up-downsizing steps in the Unet. For some datasets, we removed the least popular item from the data altogether, in order to be able to fit an even number of items as a vector / matrix.

3 Experimental Setup

Our experimental setup focuses on the classical recommendation task, where the task is to predict items which a user would enjoy / interact with, based on historical interactions [58]. For instance, prior models like the MultVAE [36] are fed the user history, and tasked to rank items, where each dimension of the input and output correspond to an item – in the case of the MultVAE, the predicted likelihood can be used to rank recommended items.

Given original binary input (feedback of whether or not someone liked / consumed an item), it is a bit harder to argue for a regular forward diffusion process with Gaussian noise. The forward process is either Gaussian or Binomial.

3.1 Assumptions

Our experiments make the following set of standard assumptions, following prior work: we assume a Top-K recommendation setup for binary implicit feedback, and evaluate using a strong generalization split. These, and other assumptions are explicitly described below:

Assumption 1 Top-K recommendation: We consider the Top-K recommendation problem, reflected primarily in the evaluation metrics we utilize: Recall@20, Recall@50 and NDCG@100.

Assumption 2 Binary feedback: If a rating is non-binary, we binarize it. We experiment with two datasets: for MovieLens [16] and Netflix [3], we convert ratings of 4 and higher to 1, and use 0 otherwise, following prior work [36, 39, 55].

Assumption 3 In contrast to sequential recommendation [62], we do not explicitly consider the order in which items are viewed, an assumption consistent with prior work [36, 39, 55], which RecFusion builds on. For evaluation (validation and test sets), we randomly sample items independently of item consumption time.

Assumption 4 We filter out users with fewer than five items, and items with fewer than five interactions, as is common practice [1, 36].

Assumption 5 Strong generalization [41]: Users are split into train/validation and test sets, with the training employing the entire history. For validation and test sets, a partial history is fed to the recommender, with a held-out set being used to evaluate the resulting recommendation.

3.2 Baselines

We benchmark our methods against the following *non-neural* baselines: (i) **Random**: Recommendations are generated by uniformly sampling without replacement from the set of items that have been interacted with. (ii) **Popularity**: The frequency of each item is calculated and subsequently normalized by dividing the individual count by the maximum count among all items. Consequently, every user receives identical recommendations with scores ranging from zero to one. (iii) **SLIM**: Linear model with a sparse item-to-item similarity matrix; solved using a constrained ℓ_1, ℓ_2 regularized optimization problem [45]. (iv) **EASE**: A variant of SLIM with a closed-form solution, obtained by dropping the non-negativity and ℓ_1 constraint, which simplifies to ridge regression [59].

We also consider the following *neural* baselines: (i) **MultVAE**: Variational autoencoder [28] with a multinomial likelihood [36]. (ii) **RecVAE**: Improves upon the MultVAE with a composite prior, newer architecture and a training schedule which alternates between training the encoder/decoder [55]. (iii) **CODIGEM**: We took the original CODIGEM code and had to to fix some bugs to make it run. Once it ran in the original bare repo, we transferred the modelling code to the RecPack framework [61].

3.3 Implementation and parameters

We provide a model card in the Appendix C. We use RecPack [43], a reproducibility framework for our experiments. We reproduce baselines ourselves, given the ambiguity over the aforementioned assumptions in existing literature. We promote a reproducible setup with the above assumptions.

We utilize the following datasets in our experiments (i) MovieLens [16] (we use 1M, 25M) (ii) Netflix [3]. Dataset statistics are reported in Appendix D. As mentioned before, we evaluate on the test set using the following metrics: Recall@20, Recall@50 and NDCG@100. We report *calibrated recall*, which adjusts for the number of true positive interactions and ensure that optimal recommendations map to a perfect recall value of 1, as is commonly done in previous work [36]. We train on single NVIDIA V100 GPUs.

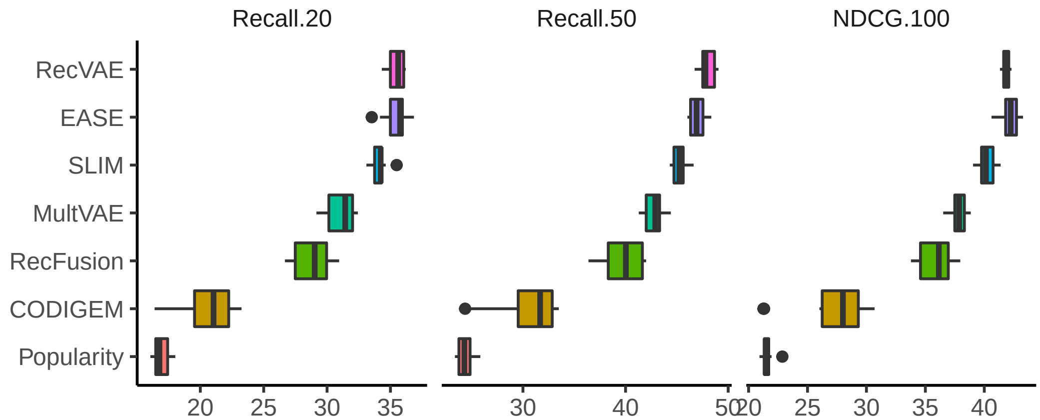


Figure 3: Experimental results on the MovieLens1M dataset. All results reproduced by us. Our method is RecFusion. Boxplots show median and IQR over 10 train/test splits

For hyperparameter tuning, we use Hyperopt [5] and its Tree of Parzen Estimators [4] algorithm and Sparktrial[3] to coordinate GPUs. We use the validation set NDCG@100 to navigate the hyperparameter space. Once the best hyperparameter combination is found, we run the model on the test split (train/val/test – $0.8/0.1/0.1$). For MovieLens1M, we bootstrap predictions and run on 10 different splits to obtain error bars on out-of-sample prediction (see Figure 3).

4 Results

Our results show that between the diffusion models, RecFusion outperforms CODIGEM on two of three datasets. However, on all datasets non-neural baselines, and EASE in particular, outperform both neural and diffusion-based models.

Non-neural baselines. Across datasets and metrics, the best performance is often obtained by EASE (see Table 3 and Table 1). EASE even outperforms MultVAE, a popular neural baseline, on most datasets and metrics. This is in line with prior research that demonstrates the efficacy of linear models for recommendation over some neural methods [11, 13, 59]. Despite this, neural methods make other tasks within recommendation viable (e.g., using user or item metadata), as highlighted in Section 5.

Comparing diffusion models and neural methods. From Figure 3 and Table 1, we observe a consistent trend: MultVAE outperforms both diffusion models, CODIGEM and RecFusion, across

[3] `http://hyperopt.github.io/hyperopt/scaleout/spark/`

Table 1: Experimental results on the MovieLens25M and Netflix datasets. All results are reproduced by us. Our method in bold.

Dataset	Model	Recall@20	Recall@50	NDCG@100
MovieLens25M	Random	0.13	0.30	0.24
	Popularity	16.63	24.43	19.69
	RecFusion	33.21	45.44	37.31
	CODIGEM	34.05	45.84	37.90
	MultVAE	35.12	48.09	39.12
	EASE	40.02	52.71	43.84
Netflix	Random	0.18	0.32	0.31
	Popularity	11.73	17.48	15.89
	CODIGEM	25.54	33.48	29.08
	RecFusion	29.68	37.63	32.87
	MultVAE	31.61	40.61	35.23
	EASE	36.19	44.49	39.35

Table 2: Experimental results for different RecFusion methods on MovieLens1M.

Model	Recall@20	Recall@50	NDCG@100
RecFusionU-Net1D	4.45	7.77	6.99
RecFusionU-Net2D	6.47	9.08	9.03
RecFusionT	14.03	17.80	16.59
RecFusionVar	16.71	24.73	21.63
RecFusionBin	17.59	23.53	21.94
RecFusion	30.91	41.76	37.44

all datasets and for all metrics. One reason might be the difference in the number of parameters employed by the two networks: MultVAE uses two three-layer networks, one each for the encoder and decoder, whereas RecFusion employs a single three-layer network, which is reflected in the number of parameters reported in Table 5. For MoveLens 1M, we observe that RecVAEoutperforms both MultVAE and both diffusion models. However, RecVAE uses a somewhat complicated (per-user) prior, along with a complicated training schedule where only the encoder (or decoder) is trained with the decoder (or encoder) is frozen. RecFusion employs none of these heuristics.

We highlight that RecFusion outperforms, or is on par with CODIGEM. We keep this time the most popular model in each of VAEs and non-neural classifications in Table 4.

Ablation study. In Table 2, we perform an ablation study: we start with models that integrate the most diffusion methods and remove elements, one-by-one. Perhaps unsurprisingly for recommendations where linear models dominate, we discover that the most bare-bone (close to linear) diffusion model works best. RecFusionU-Net1D and RecFusionU-Net2D drastically under-perform, even scoring below the Popularity baselines. For RecFusionUNet-2D, this is expected because of the lack of spatial correlations that the model was originally designed for.

Adding typical diffusion elements like time embeddings (RecFusionT), mean/variance (RecFusionVar) also underperforms compared to the base RecFusion model. We hypothesize that RecFusionBin underperforms due to the noise schedule employed: adding noise to images (256 colors) is more meaningful than adding noise to binary data. We exacerbate this problem by explicitly modelling it as a binomial Markov diffusion process.

Summary of results and discussion. Our results show that existing VAE (MultVAE, RecVAE) and non-neural baselines (EASE, SLIM) outperform more complicated architectures, like diffusion models. RecFusion, however, outperforms the diffusion baseline, CODIGEM, on two of three datasets across all metrics. RecFusion is the simplest form of our approach, with a VAE akin to MultVAE. We stress that our proposed method, RecFusion is a simpler and more elegant way to model the recommendation problem than CODIGEM. Our experiments, however, highlight the difficulty of utilizing generative models for real-world problems in which (close to) linear models dominate.

5 Related Work

This work should not be confused with diffusion models in social recommendation (e.g., [26, 65]), an orthogonal field. We briefly review the diffusion and the recommmendation literature.

Diffusion models. Diffusion probabilistic models were first introduced by [56], where the specific implementation and optimization objectives failed to surpass the state-of-the-art. A few years later, the denoising diffusion model (DDPM) was introduced by [19], where the loss function is simplified and the architecture proposed manages to achieve strong state-of-the-art performance. The rich literature that follows would be impossible to summarize in a single paragraph. The most relevant work is denoising diffusion implicit models which changes the parametrization of the sampling to make it deterministic instead of stochastic [57]. Diffusion is first often used in the 2D domain, it can also have a 3D interpretation [7, 20, 27, 67].

Diffusion for recommendation is already present in early work. CODIGEM is probably the first attempt at using diffusion models for recommendation [61]. They take inspiration from diffusion models and generate recommendations through iterative denoising. Although Diffusion models inspire CODIGEM, it is implemented effectively as a simple hierarchical variational autoencoder. The first reason is that the model does not share weights across timesteps. Also, diffusion models are based on the assumption that the forward process is performed in sufficiently small steps to guarantee that the reverse will have the same functional form [12, 56].

Recommender systems. Non-neural MF methods can solve minimization problems on single user-item rating matrices (see Figure 1). But (i) user-item metadata, (ii) time representation, (iii) and controllability / guidance (e.g., a movie recommendation set that must be action-comedy oriented) are harder to model in a closed form or iterative manner (e.g., Gibbs sampler for ALS [42]). This is where neural models can help. Within neural models, probabilistic models and especially Variational Auto Encoders (VAEs) are omnipresent, including MultVAE [36] and RecVAE [55].

Recommendation (together with, arguably, time series and tabular data) is one of only few areas where neural models do not seem to have gained supremacy yet. This has been shown in the settings of general recommendation [13, 24, 46, 47], sparse interactions [31], session-based [37] and next basket recommendation [22, 32, 34]. In these benchmarks, winning methods are variations of matrix factorization (MF) techniques (SVD++, (i)ALS, EASE [59], and SLIM [45]) or even the *most popular* benchmark. Neural models are a popular choice for recommender systems, with early models like AutoRec [54] or CDAE [66] employing auto-encoder architectures. Despite limited reproducibility of some neural models, [11, 13, 34] or the superior performance of non-neural methods in certain settings [45, 59], e.g., competitions [24], neural methods have comparable or better performance in several settings. Of these, probabilistic methods employing *variational* inference, i.e., variational auto-encoders (VAE) [28], like the MultVAE [36] or RecVAE [55] are notable, with the latter being the only neural model successfully reproduced in a large-scale reproducibility study [11, 13].

Contemporaneous work. In April 2023, while this paper was being finalized, three papers were published on diffusion for recommendation and one on Bernoulli diffusion (some peer-reviewed). *BSPM* [25] uses score-based models (the predecessor of diffusion models) as a testbed for generative models for the recommendation. *DiffuRec* [35] is the first attempt at diffusion for sequential recommendation. *DiffRec* is a similar paper to ours on smaller datasets [63]. They also report on ML1M with 3X lower results on all methods, compared to our computations or [49]. This highlights that it is hard to compare recommendation literature, due to underlying hidden data preprocessing assumptions. We make these assumptions explicit in our paper and code (see Section 3.1). Finally, *BerDiff* [8] is the first attempt we could find to explicitly model binary data with a Bernoulli Markov diffusion process.[4] *BerDiff* focuses on 2D CT scan and MRI data and thus relies heavily on the Unet architecture. In our paper, we show theoretically and empirically that we face more of a 1D problem and thus define our own 1D diffusion model for binary data.

6 Conclusion

We position this paper as a first attempt at designing diffusion models for binary 1D data in the context of recommendation and beyond. With RecFusion, our simple diffusion model (hierarchical VAE) is on par with popular VAE methods. We conjecture that extensions (techniques like composite priors, etc as in RecVAE [55]) can further improve performance. We first argue that we need to tackle limitations in our existing implementation and layout some proposals for future improvements. We can summarize our learnings as follows. First, we show theoretically and empirically that the lack of spatial relationship between users and items is a hindrance to using any image-inspired models, including even a 1D U-Net. We then implemented our binary (Bernoulli) Markov process, as a model adapted to the problem at hand.

Broader impact. The image domain sometimes still requires the simplicity of binary settings, like segmentation masks on MRI, CT scans [8, 40] or for conventional object detection techniques [30]. This is potentially fruitful ground for applying our proposed diffusion models for binary 1D data.

7 Limitations

Our setup relies on weak (even if common) recommendation setup assumptions. To these assumptions we have to add that the items list is fixed: our model can not account for new items in the catalogue after training. This is a limitation shared with CODIGEM, but also with VAE-based models.

[4]Sohl-Dickstein et al. [56] already hinted at diffusion as a 1-dimensional idea as a proof of concept. We could not find the full mathematical derivation or code for it, however.

We have yet to test how robust our diffusion models are to a relaxations of these assumptions. Diffusion can also be applied to further domains of recommendation like sequential recommendation with diffusion + RNN, or explicitly model count data input with star ratings instead of binarized feedback.

RecFusion does not yet use further diffusion methods, such as inpainting, guidance (e.g., to predict the user preference distribution or use a prior on movie genre a.k.a controllable recommendations), diffusion on the embedding space [15] (in particular, user-item matrix embeddings), or multinomial likelihood to model the dependencies of item feedbacks for a user [21], input masking. We believe these are fruitful areas for future work.

Acknowledgments

We are grateful to Christian Andersson Naesseth who helped in discussions about inpainting. This research was (partially) funded by Bertelsmann SE & Co. KGaA and by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, `https://hybrid-intelligence-centre.nl`.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Joeran Beel and Victor Brunel. Data pruning in recommender systems research: Best-practice or malpractice? In *13th ACM Conference on Recommender Systems (RecSys)*, 2019.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[3] James Bennett and Stan Lanning. The Netflix prize. In *Proceedings of KDD Cup Workshop 2007*, pages 3–6. ACM, 2007.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[5] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/bergstra13.html`.

[6] Samarth Bhargav and Evangelos Kanoulas. Controllable recommenders using deep generative models and disentanglement. *arXiv preprint arXiv:2110.05056*, 2021.

[7] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion NeRF: A unified approach to 3D generation and reconstruction. *arXiv preprint arXiv:2304.06714v2*, 2023.

[8] Tao Chen, Chenhui Wang, and Hongming Shan. Berdiff: Conditional bernoulli diffusion model for medical image segmentation. *arXiv preprint arXiv:2304.04429*, 2023.

[9] Yifan Chen and Maarten de Rijke. A collective variational autoencoder for top-n recommendation with side information. In *3rd Workshop on Deep Learning for Recommender Systems*. ACM, October 2018.

[10] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2023.

[11] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems*, 39(2):20:1–20:49, 2021.

[12] William Feller. On the theory of stochastic processes, with particular reference to applications. In *Proceedings of the First Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1949.

[13] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar 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*. ACM, 2019.

[14] Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Difformer: Empowering diffusion models on the embedding space for text generation. *arXiv preprint arXiv:2212.09412*, 2023.

[15] Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. Difformer: Empowering diffusion models on the embedding space for text generation, 2023.

[16] F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5:Article 19, 2015.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL `https://openreview.net/forum?id=qw8AKxfYbI`.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[20] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[21] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12454–12465. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/67d96d458abdef21792e6d8e590244e7-Paper.pdf`.

[22] Haoji Hu, Xiangnan He, Jinyang Gao, and Zhi-Li Zhang. Modeling personalized item frequency information for next-basket recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[23] Jin Huang, Harrie Oosterhuis, and Maarten de Rijke. It is different when items are older: Debiasing recommendations when selection bias and user preferences are dynamic. In *WSDM 2022: The Fifteenth International Conference on Web Search and Data Mining*, pages 381–289. ACM, February 2022.

[24] Dietmar Jannach, Gabriel de Souza P. Moreira, and Even Oldridge. Why are deep learning models not consistently winning recommender systems competitions yet? In *Proceedings of the Recommender Systems Challenge 2020*. ACM, 2020.

[25] Choi Jeongwhan, Hong Seoyoung, Noseong Park, and Sung-Bae Cho. Blurring-sharpening process models for collaborative filtering. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2023.

[26] Bo Jin, Ke Cheng, Liang Zhang, Yanjie Fu, Minghao Yin, and Lu Jiang. Partial relationship aware influence diffusion via a multi-channel encoding scheme for social recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, page 585–594. ACM, 2020.

[27] Firas Khader, Gustav Müller-Franzes, Soroosh Tayebi Arasteh, Tianyu Han, Christoph Haarburger, Maximilian Schulze-Hagen, Philipp Schad, Sandy Engelhardt, Bettina Baeßler, Sebastian Foersch, Johannes Stegmaier, Christiane Kuhl, Sven Nebelung, Jakob Nikolas Kather, and Daniel Truhn. Denoising diffusion probabilistic models for 3D medical image generation. *Scientific Reports*, 13(1), 2023.

[28] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[29] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. On density estimation with diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

[30] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[31] Yasamin Klingler, Claude Lehmann, João Pedro Monteiro, Carlo Saladin, Abraham Bernstein, and Kurt Stockinger. Evaluation of algorithms for interaction-sparse recommendations: Neural networks don't always win. In *25th International Conference on Extending Database Technology*, Proceedings of EDBT, OpenProceedings.org, March 2022. OpenProceedings.org. doi: 10.48786/edbt.2022.42.

[32] Sara Latifi, Noemi Mauro, and Dietmar Jannach. Session-aware recommendation: A surprising quest for the state-of-the-art. *Information Sciences*, 573:291–315, Sep 2021.

[33] Hanze Li, Scott Sanner, Kai Luo, and Ga Wu. A ranking optimization approach to latent linear critiquing for conversational recommender systems. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, page 13–22. ACM, 2020.

[34] Ming Li, Sami Jullien, Mozhdeh Ariannezhad, and Maarten de Rijke. A next basket recommendation reality check. *arXiv preprint arXiv:2109.14233*, 2021.

[35] Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *arXiv preprint arXiv:2304.00686*, 2023.

[36] 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*, page 689–698. ACM, 2018.

[37] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4–5):331–390, 2018.

[38] Kai Luo, Hojin Yang, Ga Wu, and Scott Sanner. Deep critiquing for vae-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 1269–1278. ACM, 2020.

[39] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[40] Jun Ma and Bo Wang. Segment anything in medical images, 2023.

[41] Benjamin Marlin. *Collaborative Filtering: A Machine Learning Perspective*. University of Toronto, 2004.

[42] Clara Menzen, Manon Kok, and Kim Batselier. Alternating linear scheme in a Bayesian framework for low-rank tensor approximation. *arXiv preprint arXiv:2012.11228*, 2021.

[43] Lien Michiels, Robin Verachtert, and Bart Goethals. RecPack: An (other) experimentation toolkit for top-n recommendation using implicit feedback data. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 648–651, 2022.

[44] Preksha Nema, Alexandros Karatzoglou, and Filip Radlinski. Disentangling preference representations for recommendation critiquing with ß-vae. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, page 1356–1365. ACM, 2021.

[45] Xia Ning and George Karypis. SLIM: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506, 2011.

[46] Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395v1*, 2019.

[47] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. Revisiting the performance of iALS on item recommendation benchmarks. In *Sixteenth ACM Conference on Recommender Systems*. ACM, 2022.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[49] Noveen Sachdeva, Mehak Preet Dhaliwal, Carole-Jean Wu, and Julian McAuley. Infinite recommendation networks: A data-centric approach. In *Advances in Neural Information Processing Systems*, 2022.

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*.

[51] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[52] Jonathan Scarlett and Volkan Cevher. An introductory guide to Fano's inequality with applications in statistical estimation. *arXiv preprint arXiv:1901.00555*, 2019.

[53] Jonathan Scarlett and Volkan Cevher. *An Introductory Guide to Fano's Inequality with Applications in Statistical Estimation*, page 487–528. Cambridge University Press, 2021. doi: 10.1017/9781108616799.017.

[54] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, page 111–112. ACM, 2015.

[55] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I. Nikolenko. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. ACM, 2020.

[56] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265. PMLR, 2015.

[57] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[58] Harald Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, page 213–220. ACM, 2013.

[59] Harald Steck. Embarrassingly shallow autoencoders for sparse data. *The World Wide Web Conference*, May 2019. doi: 10.1145/3308558.3313710. URL `http://dx.doi.org/10.1145/3308558.3313710`.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[61] Joojo Walker, Ting Zhong, Fengli Zhang, Qiang Gao, and Fan Zhou. Recommendation via collaborative diffusion generative model. In *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022, Singapore, August 6–8, 2022, Proceedings, Part III*, page 593–605. Springer-Verlag, 2022.

[62] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: Challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[63] Wang Wenjie, Xu Yiyan, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Jul 2023.

[64] Ga Wu, Kai Luo, Scott Sanner, and Harold Soh. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, page 137–145. ACM, 2019.

[65] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 235–244. ACM, 2019.

[66] 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*, page 153–162. ACM, 2016.

[67] Guandao Yang, Abhijit Kundu, Leonidas J. Guibas, Jonathan T. Barron, and Ben Poole. Learning a diffusion prior for NeRFs. *arXiv preprint arXiv:2304.14473v1*, 2023.

[68] Hojin Yang, Tianshu Shen, and Scott Sanner. Bayesian critiquing with keyphrase activation vectors for vae-based recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 2111–2115. ACM, 2021.

A The ELBO is also suited for bernoulli samples

According to the classic definition of the ELBO by [56] and [19], there are no assumptions regarding the distributions p_θ or q_θ. We reproduce here for completeness the derivation from [56] on why the ELBO satisfies any distribution given Jensen's inequality:

$$\begin{aligned} L &= \int d\mathbf{X}^0 q\left(\mathbf{X}^0\right) \log p\left(\mathbf{X}^0\right) \\ &= \int d\mathbf{X}^0 q\left(\mathbf{X}^0\right) \log \left[\begin{array}{c} \int d\mathbf{X}^{1:T} q\left(\mathbf{X}^{1:T} \mid \mathbf{X}^0\right) \\ p\left(\mathbf{X}^T\right) \prod_{t=1}^{T} \frac{p\left(\mathbf{X}^{t-1} \mid \mathbf{X}^T\right)}{q\left(\mathbf{X}^T \mid \mathbf{X}^{t-1}\right)} \end{array} \right] \end{aligned}$$

The latter has a lower bound given Jensen's inequality that also applies to the bernoulli distribution.

$$L \geq \int d\mathbf{X}^{0:T} q\left(\mathbf{X}^{0:T}\right) \log \left[p\left(\mathbf{X}^T\right) \prod_{t=1}^{T} \frac{p\left(\mathbf{X}^{t-1} \mid \mathbf{X}^T\right)}{q\left(\mathbf{X}^T \mid \mathbf{X}^{t-1}\right)} \right]$$

In practice, the product term is computed with a KL divergence. It can be shown with Fano's inequality [53] that our cross-entropy loss also aims for a lower bound like KL divergence.

For this assumption regarding p_θ or q_θ to be valid we make sure that the forward steps (i.e. β^t) are small enough, following [56].

B Proof of why bernoulli diffusion is multiplicative

Given our bernoulli diffusion formulation

$$q\left(\mathbf{X}^t \mid \mathbf{X}^{t-1}\right) := \mathcal{B}\left(\mathbf{X}^t; \mathbf{X}^{t-1}\left(1-\beta^t\right) + \tfrac{1}{2}\beta^t\right),$$

we would like to make sure that $q\left(\mathbf{X}^t \mid \mathbf{X}^{t-1}\right)$ can still be sampled at an arbitrary step t in closed form, as with traditional gaussian diffusion [56]. Without loss of generalization – since we sample independently from a bernoulli distribution – we show that this is true for a single user-item combination x^t

$$x^t = \left(1-\beta^t\right) x^{t-1} + \tfrac{1}{2}\beta^t$$

By substituting x^{t-1}, we get

$$\begin{aligned} x^t &= \left(1-\beta^t\right)\left[\left(1-\beta^{t-1}\right) x_{t-2} + \tfrac{1}{2}\beta^{t-1}\right] + \tfrac{1}{2}\beta^t \\ &= \left(1-\beta^t\right)\left(1-\beta^{t-1}\right) x_{t-2} + \tfrac{1}{2}\left(1-\beta^t\right)\beta^{t-1} + \tfrac{1}{2}\beta^t \end{aligned}$$

If we keep on substituting the previous diffusion step, we arrive at the original data x_0. By factorizing the above and by induction, it is trivial to show that

$$x^t = \prod_{i=1}^{t}\left(1-\beta^i\right) x_0 + \frac{1}{2}\sum_{j=1}^{t-1}\left[\prod_{i=j+1}^{t}\left(1-\beta^i\right)\right]\beta^j + \frac{1}{2}\beta^t \tag{7}$$

We can actually express the middle term with a common index by using $\beta^j = 1-(1-\beta^j)$[5]. We then obtain a telescoping sum:

$$\begin{aligned} \sum_{j=1}^{t-1}\left[\prod_{i=j+1}^{t}\left(1-\beta^i\right)\right]\beta^j &= \sum_{j=1}^{t-1}\left[\prod_{i=j+1}^{t}\left(1-\beta^i\right)\left(1-\left(1-\beta^j\right)\right)\right] \\ &= \sum_{j=1}^{t}\left[\prod_{i=j+1}^{t}\left(1-\beta^i\right) - \prod_{i=j}^{t}\left(1-\beta^i\right)\right] = 1 - \prod_{i=1}^{t}\left(1-\beta^i\right) \end{aligned}$$

Substituting this term back into Equation 7,

$$x^t = \prod_{i=1}^{t}\left(1-\beta^i\right) x_0 + \tfrac{1}{2}\left[1 - \prod_{i=1}^{t}\left(1-\beta^i\right)\right] + \tfrac{1}{2}\beta^t$$

[5] We borrow this trick from `https://math.stackexchange.com/questions/4467894/does-a-markov-chain-with-gaussian-transitions-px-tx-t-1-mathcal-n-sqrt1`

Finally, by defining $\alpha^t = 1 - \beta^t$ and $\bar{\alpha^t} = \prod_{\tau=1}^{t} \alpha^\tau$, we get

$$x^t = \bar{\alpha}^t x_0 + \frac{1}{2}\left(1 - \bar{\alpha}^t\right)\bar{\alpha}^t \tag{8}$$

We showed that $q\left(x^t \mid x^{t-1}\right)$ can be expressed in terms of x_0 as a single bernoulli sample. ■

C Model card

see `https://github.com/gabriben/recfusion/model_card.md`.

D Descriptive statistics

In Table 3, we show counts of users items and interactions on the train, val and test sets. We prvide this as an extra step for data preprocessing transparency.

Table 3: Descriptive statistics: Counts of active (non-zero) users, items and interactions after preprocessing and under train / val / test (0.8 / 0.1 / 0.1) splitting regime over users

	No. users			No. items			No. interactions		
Dataset	train	val	test	train	val	test	train	val	test
ML1M	4,832	604	604	3,416	3,158	3,282	798,608	76,513	84,772
ML25M	130,032	16,254	16,255	32,718	24,818	25,597	19,924,515	1,999,297	2,030,221
Netflix	378,389	47,299	47,299	17,769	17,761	17,759	80,418,808	8,011,940	8,060,214

E Results on MovieLens1M as a table

Table 4 is the pendant of Figure 3 for the MovieLens 1M dataset. We added the Random baseline here, but left it out of the figure for aesthetics.

Table 4: Experimental results on the MovieLens1M dataset. All results reproduced by us over 10 train/test splits, we report median results. Our method in bold.

type	model	Recall@20	Recall@50	NDCG@100
baselines	Random	0.87	1.71	1.73
	Popularity	16.77	24.30	21.44
diffusion	CODIGEM	21.04	31.67	28.00
	RecFusion	29.02	40.03	36.14
VAE	MultVAE	31.43	42.89	37.87
	RecVAE	35.61	47.79	41.81
non-neural	SLIM	34.23	45.25	40.14
	EASE	35.76	46.92	42.24

F Number of parameters

One of our arguments is that our model is more efficient than existing neural baselines (see Table 5).

Table 5: number of parameters for different neural architectures on the ML1M dataset

MultVAE	CODIGEM	RecFusion
446,421,6	560,388,6	141,021,8