Exploring the Capabilities of Classifier-Free Guidance in Recommendation Tasks

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

by

Noah Buchanan
University of Arkansas - Fort Smith
Bachelor of Science in Computer Science, 2022

December 2024
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

Susan E. Gauch, Ph.D.
Thesis Director

Brajendra Nath Panda, Ph.D.
Committee Member

Lu Zhang, Ph.D.
Committee Member

Abstract

This thesis addresses the problem of recommending items to users based on their ratings of items through a diffusion recommender system. Regular recommender systems are already capable of efficient recommendation through conventional methods such as collaborative or content-based filtering. Diffusion is a new type of generative AI that aims to improve our previous AI's shortcomings in the generative domain, like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). We use diffusion to create a recommender system that mirrors the sequence users take when browsing and rating items. The current methods of recommendation with diffusion do not use the new innovation in diffusion models as a whole, known as classifier-free guidance. In this thesis, we aim to improve the previous approaches of diffusion recommender systems by implementing classifier-free guidance and augmenting the underlying model that powers diffusion recommender systems. We implement several different architectures for the underlying model and many different methods, including classifier-free guidance. Our findings show improvements to a previous implementation for most metrics on most datasets and an enhanced ability to create relevant recommendations for users from sparse and not information-dense datasets, where there are far fewer items than users.

Acknowledgments

I would like to thank my advisor, Dr. S. Gauch, for her invaluable help with countless problems throughout the duration of this thesis. Last but certainly not least, all that helped support or contribute in one way or another, all of your efforts pushed me forward, and I thank you personally for that.

Table of contents

1. Introduction

The age-old saying of machines surpassing us in intelligence but not in imagination may be coming to a head with the most recent advancements in generative AI. The introduction of generative models marked a cornerstone of the capabilities of AI and gave us a glimpse into the distances we can go utilizing such tools. Generative AI is generally considered to have started with the production of the Markov Chain; a simple statistical algorithm that can generate new data sequences based on input, a relatively rudimentary generation method and likely not the first thought when considering generative AI. Markov chains will be useful in understanding some of the methods described in this paper and will be discussed more in-depth in later sections. Fast forward to the introduction of Autoencoders for learning a compressed format of information passed into the model. It can be shown that these models have generative capabilities, however, it is limited. Part of the problem with these models is that in the latent space created from the encoder's compression, there are parts that don't correspond to any data point in the original input. Using those as inputs to the decoder will result in something that does not look like anything from any of the original data points. Thus, only parts of the latent space have generative capabilities and are limited at that, only generating data we have seen before. This is what it means for the latent spaces in Autoencoders not being regularized. During inference, and when the training data is gone, we have no way of knowing whether the output generated is garbage or not; this is where VAEs make their appearance as one of our first modern generative AI models (Kingma, Welling, 13).

VAEs are designed to address the non-regularized latent space and provide them with robust generative capability for the entire latent space. Instead of a compressed vector output in

the latent space, VAEs give us the parameters for a distribution for every input as well as impose a constraint on this distribution, forcing it to be normal so that the space is regularized and all inputs to the decoder are meaningful. This process of learning a distribution of the inputs is another topic similar to something we will be discussing and will be touched on more later. GANs came not long after the introduction of VAEs and boast their generative capabilities all the same. GANs are relatively simple conceptually, with two models locked in competition. One is a generator of data, and another is a discriminator attempting to determine if the data being fed to it is real or created by the generator. The adversarial process helps the generator to improve its capabilities of generation, and eventually, if we were to use this model to simply generate data we would have no more need for the discriminator as it is only used to push the generator to improve.

Both VAEs and GANs have been shown to be capable of a range of types of generation (Goodfellow et al., 14) (Kingma, Welling, 2013) (Bond-Taylor, et al, 22). Specifically, text-to-image synthesis is one such capability and, coincidentally, a very popular area of research. The idea of a pipeline capable of transforming our human language into another form of media that has none of the ease of creation that language does is an extremely appealing concept; as a result, there has been an explosion of research in this area in recent years. GANs and VAEs were the main generative models used for a broad scope of tasks for some time. Gans were our most accomplished text-to-image model only a couple of years ago, but the rate at which advancements are being made appears to be dwarfing the capabilities of these once giants in the generative domain. They remain somewhat competitive with less recent methods, but the quality of generations and capability for imagination in recent techniques heavily outshines

GANs. VAEs have also remained somewhat competitive in the generative domain; however, just as GANs have been surpassed, so have VAEs. You may be curious about what technique is surpassing the concrete methods used for the last decade, and the answer is *diffusion* of course!

Not only do diffusion-based techniques heavily outperform any other recent technique in visual fidelity of images and their understanding of textual prompts, but also the range of generations that diffusion creates and their ability to create variations of the same image while still accurately following the guidelines provided by the text prompt. Diffusion models are based on the idea of diffusion in the natural sciences and are heavily inspired by non-equilibrium thermodynamics. The process of diffusion in natural science is the movement of solutes and molecules from a high concentration to a lower concentration. As the name suggests, diffusion models attempt to mimic this process using Gaussian noise instead as our measure of concentration, a fully noised image being the parallel of a perfectly diffused liquid. The idea is that in actual diffusion, no matter where you drop or how you insert dye in a container of water, given enough time, the dye will diffuse evenly into the entire container, reproducing the same result every time; diffusion models attempt to recreate this but instead of recreating the same equally diffused dye every time we recreate a distribution of the original data from some random Gaussian noise. Using this idea, diffusion-based techniques have achieved unparalleled results seen anywhere else in text-to-image synthesis and general image generation. While text-to-image is likely the most popular use case, they are capable of other tasks, such as text-to-text (sequence-to-sequence) or text-to-video. One such unique use case is personalized recommendation.

Generative recommender systems infer the user interaction probabilities of items that have not been interacted with using users' interactions with said items. The problem with generative recommender models currently is that we lack variety, given that we only have GANs and VAEs for this task. GANs suffer from instability (Saad, et al., 23), and VAEs suffer from a trade-off between tractability and representation (Bond-Taylor et al., 22). Diffusion models suffer from neither of these problems thus, they seem an apt solution to the predicament of generative recommender systems. The current problem with the diffusion recommender system is that the method of implementation lacks the capability of adding a recent innovation in diffusion models known as classifier-free guidance. We will go in-depth into what classifier and classifier-free guidance are in the related work section, but classifier-free guidance is essentially a form of guidance that requires no external classifier, as its name suggests, and improves the generative capability of diffusion models. In this thesis, one of our main contributions is to modify the implementation of a diffusion-based recommender system so that we can properly utilize classifier-free guidance and prove that the benefits it provides do transfer to this new task of recommendation. We also contribute to the discussion of the architecture of the underlying model used in these diffusion recommender systems which we will also discuss more in depth in a later section. Lastly, we contribute to a common problem that all recommender systems are often afflicted by: few-shot and zero-shot scenarios.

Section 2 will give a summary of related work on Recommender Systems, Machine Learning and its relevancy to this work, Diffusion models, Diffusion-based recommender models, and Classifier-Free Guidance. In section 3, we shall detail our process of implementing Classifier-Free Guidance into our inspiration paper's implementation for a diffusion

recommender system and explain what exactly is happening in our recommender system, specifically, the diffusion process. Section 4 will detail my results and how they compare to results lacking the modifications we have made and how these modifications also give us new tools to tackle old problems recommender systems deal with. Finally, in Section 5, we present conclusions and discuss future work in this area that could push the capabilities of diffusion recommenders even further.

2. Related Work

In this chapter, we discuss Recommender Systems, Machine Learning, and Diffusion Models and how these fields combine. We also give some background on a technique used in diffusion models that have seen significant success in recent literature, known as classifier-free guidance, which is the basis of the improvement for this paper.

2.1 Recommender Systems

Recommender systems are a crucial part of how businesses operate today, with users providing information in return for recommendations on products, whether items from a store or shows/movies. The foundation for recommender systems was generally considered to be first introduced by Elaine Rich in 1979 through a system she called Grundy (Rich, 79). She wanted a way to recommend users' books by asking them specific questions, in turn classifying them into classes of preferences depending on their answers. Jussi Karlgren improved upon Elaine Rich's system by introducing a measure of "closeness" to attempt to imitate the occurrences in bookcases where interesting documents often happen to be found adjacent to each other regardless of a relatively unordered bookcase (Karlgren, 90). This idea of closeness was adopted

and used in a variety of models after its inception. In recommender systems, there are two main techniques used: collaborative-based filtering and content-based filtering, as well as, of course, a hybrid approach of these two. For collaborative-based filtering, we can use the theoretical measure of closeness introduced by Jussi Karlgren for measuring a user's closeness to another user, and depending on what we have already seen that this other user likes, then perhaps the original user will have a similar preference if they have a closeness or likeness. Content-based filtering can also make use of closeness by simply measuring a user's closeness to an item perhaps, and sorting in descending order; however, there are multiple different ways to use this closeness for both methods, but the important thing to note is that collaborate-based filtering is based on other users' preferences that we have already observed and content-based filtering looks at the item features to recommend.

The first information filtering system based on collaborative filtering through human evaluation was introduced by (Goldberg et al., 92), where they would make recommendations based on similarities between the interest profile of a user in question and other users; the concept was coined "weaving an information tapestry." Inspired by the results of the study, researchers from MIT and UMN created a new recommendation service, *GroupLens,* whose key component is a user-to-user collaborative filtering model (Resnick et al., 94). Also named GroupLens was a research lab at UMN founded after the previous paper, which pioneered recommender system studies. The emergence of e-commerce sparked the realization of the business value of recommendation, and as a result, companies began offering recommender engines for customers, such as Amazon, Best Buy, etc. The GroupLens research lab launched the *MovieLens* project and created the first version of the recommender model (Harper, Konstan,

15). As a result of this project and its success, several MovieLens datasets were released from then to now and became one of, if not the most, popular datasets for recommendation studies. An iteration of the MovieLens dataset is even used in this thesis as part of our experiments. While content-based filtering is another valid technique, collaborative-based filtering has dominated the field for quite some time, and as of now, that has not changed.

There is another popular tool at our disposal that, depending on the implementation, could be either collaborative or content-based recommendations. AI-based methods have been very popular, with the explosion of AI research in the last 20 years being an obvious indicator of its soon-to-be application to the recommendation domain. The application in this field did see success as early as 2006 and 2009, inspired by the Netflix Prize, matrix factorization models that would learn a user embedding matrix and an item embedding matrix; using these embedding matrices, we can find similarities using the measure of closeness we talked about earlier, in this case typically it would be the dot product (Koren, 08), (Koren, 10). Other techniques were created during this period, and linear models for this task gained some popularity. (Richardson et al., 07) presented a logistic regression model that achieved tangible improvements on new user-centric evaluation methods, such as errors in click-through rate estimation. Since the implementation of these AI techniques for recommendation, a variety of improvements have come but none so impactful as the introduction of deep learning for recommendation systems. (Liang et al., 18) introduced a VAE for collaborative filtering with success on the MovieLens and Netflix Prize datasets. (Gao et al., 20) showed us that GANs are also capable of creating a recommender model. Models for video recommendation were also created following the introduction of deep learning for recommender systems (Covington et al., 16), showing that deep

models have a surprising ability to recommend videos and the capability to perform both candidate generation and ranking.

2.2 Machine Learning

Machine Learning (ML) approaches have been used in a myriad of tasks over the last decade, including classification, language processing, and image recognition. Recent advances in neural networks have provided big improvements in performance. Neural networks are a machine learning algorithm that similarly makes decisions to the human brain, processing information in a way designed to mimic the biological neurons in our own brains. Neural networks consist of layers of artificial neurons, an input layer, one or more hidden layers, and an output layer. Each neuron decides whether data should be passed along to the next layer based on a threshold value. The connections between layers have what we call weights, which, as their name suggests, weigh the connections from layer to layer. These weights are how we customize a neural network to a specific task. Once trained, the weights will be in a specific configuration for a task for which we have trained it. We can use this concept of neural networks for all kinds of tasks and create new architectures that can solve the ones we cannot. The most basic of neural networks are known as feedforward neural networks, which can perform a wide variety of tasks depending on how they are implemented. These models are the origin of all more recent models, and they can perform tasks such as pattern recognition, classification, non-linear regression, and function approximation.

Generative neural networks have not been around as long as regular feedforward networks but have made waves with their capabilities. We have already briefly discussed GANs

and VAEs, which provide breakthroughs in generative AI. While there are other methods of generation, such as text generation through large language models, these are not as relevant to this thesis' purpose and will not be discussed in depth. Diffusion is an emerging technique in generative AI that is garnering attention. Diffusion is typically used for image generation purposes but, as we will show in this thesis, it has uses in other generative models.

2.3 Diffusion

The first paper that broaches the topic of diffusion models coined "Deep Unsupervised Learning using Nonequilibrium Thermodynamics", aimed to attack the tradeoff between *tractability* and *flexibility* that probabilistic models historically are afflicted by (Sohl-Dickstein et al., 15). A *tractable* model is evaluated analytically and easily fits to data. Unfortunately, these models do not perform well with rich datasets. *Flexible* models can be fit to structure in arbitrary data. However, evaluating, training, or drawing samples from flexible models requires a very expensive Monte Carlo method of estimation for determining the normalizing constant Z, which describes the yielding flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where $\phi(\mathbf{x})$ is the function that describes a flexible model. Their method of combating this uses Markov chains to convert the Gaussian distribution into a target distribution using a diffusion process. Learning using this framework, as they defined it, involves estimating the small perturbations that are added iteratively in a diffusion process. Estimating small changes is more tractable than describing a full change of distribution in one step.

The next major breakthrough came from a team at UC Berkeley which was the first major adoption of this style of model when it was shown to be capable of generating high-quality

images (Ho et al., 20). Not only did they show that these models are indeed capable of high-quality image generation, being highly competitive with the state-of-the-art on the Cifar-10 dataset, but also even the class-conditional models that have a classifier specifically for guiding generation to be higher fidelity on the same task. They also introduced a new parameterization of diffusion models where, instead of training the reverse process to estimate the forward process posterior mean at each time step, they instead attempted to estimate the noise from the timesteps. This new parameterization led to a new simplified loss function, which is essentially just the mean squared error of the noise added to the forward process and the noise predicted by the model. As we mentioned earlier this model is competitive with class-conditional models but is not itself a conditional model. The authors noted that this process of creating a conditional diffusion model would be straightforward and left it to future work.

The previous paper sparked a major adoption of these models and as a result, lots of improvements followed swiftly. We already know that diffusion models are capable of high-quality image generation in unconditional scenarios, and it will be shown that they excel even further in other areas. Diffusion models became the de-facto start-of-the-art in class-conditional generation, with things such as classifier guidance being adopted (Ho et al., 21) (Dhariwal, Alex, 21). Super-resolution also became a domain that diffusion models would begin to dominate (Saharia et al., 21). The use cases and ability of diffusion models were not the only areas that improved after this major adoption. The performance of these models began to be optimized with techniques such as *latent diffusion models* being introduced in the famous *Stable Diffusion* paper (Rombach et al., 22). Latent diffusion models take from the idea of the latent space in autoencoders; before we perform the forward process, the data is compressed into the

latent space, and we perform both the forward and reverse Markov transitions in this compressed dimensionality and upscale the result back to the original size before output. Not only did this paper make waves in the research community but also among non-researchers as this was the first time the capabilities of these models were known to all, as well as providing us open-resource access to code and model weights so that anyone could use it if following a tutorial. The use of the latent space was not the only technique taken from VAEs to improve diffusion models (Gu et al., 22) showed that we can also utilize Vector Quantization techniques in the latent space to further improve the efficiency of both training and sampling speeds. Another technique by a team at Snapchat used step-distillation to decrease the number of steps required in the denoising process to create the *SnapFusion* model which boasts image generation within two seconds on mobile devices (Y. Li et al., 23).

Large language models from NLP were a major breakthrough in AI research as a whole, excelling in chatbot capabilities, content generation, language translation, text summarization, question-answering systems, and even personalized recommendations. As a result, the timing could not be more perfect to combine their capabilities of textual understanding with the prompt-based conditional diffusion models that are currently exploding in popularity. A team at Google explored such a combination of techniques and found that large language models were surprisingly effective at encoding text for image synthesis and, as a result developed a model named *Imagen* (Saharia et al., 22). This combination of techniques was one of the first and was extremely effective. *DALL-E 2,* a model by a team at OPEN AI, also used large language models with diffusion models by creating a CLIP model that could generate images using its improved language-image understanding (Ramesh et al., 22). The third iteration of DALL-E also utilized

this technique but found that the captions for images were causing flaws in the generations as they were imperfect, further emphasizing the importance of the language component of these models (Betker et al., 23).

2.4 Diffusion-Based Recommender Systems

As a result of the success of diffusion models and the expansion of their use cases, diffusion-based recommender systems began to gain traction. As we already know, GANs and VAEs are widely utilized to model the generative process of user interactions, but they suffer from the same instability as GANs and the limited representation ability of VAEs. Diffusion models were to be the hope of solving both of these issues in the image generation domain, so it leads one to believe that they are capable of also overcoming these problems in generative recommender systems where GANs and VAEs once again fall short. (Wang Wenjie, 23) gave us the answer to this question by providing us with one of the first diffusion-based recommender models in recent research. They argued that the objectives of recommender models align well with diffusion models since recommender models essentially infer the future interaction probabilities based on corrupted historical interactions. Their approach did not involve any direct conditioning, instead, the amount of noise added to the data was limited so that the data was corrupted but not completely gone.

While diffusion-based recommendation seems to be a relatively new area of research, it appears to be gaining traction quickly. Within the same year, we saw other diffusion recommender models that attempted to perform the same task with different approaches. (Wang Yu, 23) Introduced a conditional diffusion-based recommender system using a cross-attentive

denoising decoder for removing noise. Their method instead used direct conditioning, opting away from the guidance via partially corrupted data from the previous approach. Their results claim to beat multiple recommender models in every category, ranging from Precision @ K, Recall @ K, and Mean Reciprocal Rank on multiple Amazon datasets (Amazon Office, Beauty, Tools, Toys). This inspired other papers, such as (Z. Li et al., 23), to create their own methods of sequential recommendation using diffusion models. Papers such as (Lin, 23) even went so far as to use diffusion for reranking purposes in a recommendation model, which also found great success. All of these methods being produced in the same year that diffusion-based recommendation was introduced assumingly leads one to believe that this research will continue to pick up and improve, and hopefully, my research will compound and generate something interesting that others can use.

2.5 Classifier-Free Guidance

The previous section details diffusion recommender systems, some of which were conditional. All conditional models mentioned do not make use of a technique that has seen some recent success for diffusion models in other domains, such as their main use case, text-to-image synthesis. Firstly, we need to talk about classifier guidance introduced by the previously mentioned (Dhariwal, Alex, 21) in the diffusion related work section. Classifier guidance essentially modifies the diffusion score, the gradient of the log probability density function, which is much easier to learn than directly modeling the data distribution. The classifier, which classifies a type of data we are trying to generate using the diffusion model, up-weights the probability of data for which the classifier assigns a high likelihood to the correct label. This, put in simple terms, guides the diffusion process towards something the classifier

would classify with a high likelihood as what it is designed to classify. (Salimans et al., 2016) has already shown that data that can be classified well scores high on perceptual quality, which all lends to a higher-quality image generation model.

Classifier guidance does appear to effectively trade off precision and recall, providing us with images closer to our description but with a smaller variation of images, as requested. However, the models that use this method are reliant on the gradients of an image classifier, and depending on how many different classes of images we want to generate, creating a classifier for every single one could be a tedious exercise. What if we were capable of eliminating the classifier as well as providing even better capabilities of guidance than classifier guidance? Introduce *Classifier-Free Guidance* (Ho et al., 22). Classifier-free guidance essentially attempts to have the same effect as classifier guidance but without a dedicated classifier. Instead of training a separate classifier model, they chose to train an unconditional denoising diffusion model parameterized through a score estimator (as we mentioned earlier, estimating a score function is easier than the actual distribution) together with the conditional denoising diffusion model parameterized through a conditional score estimator. However, these are not actually two separate models; they used a single neural network to parameterize both models. Where the unconditional model is used, they simply input a null token for the class identifier. The models are jointly trained by simply randomly setting the conditioning to a null token based on the probability of a hyperparameter *p-uncond*.

3. Approach

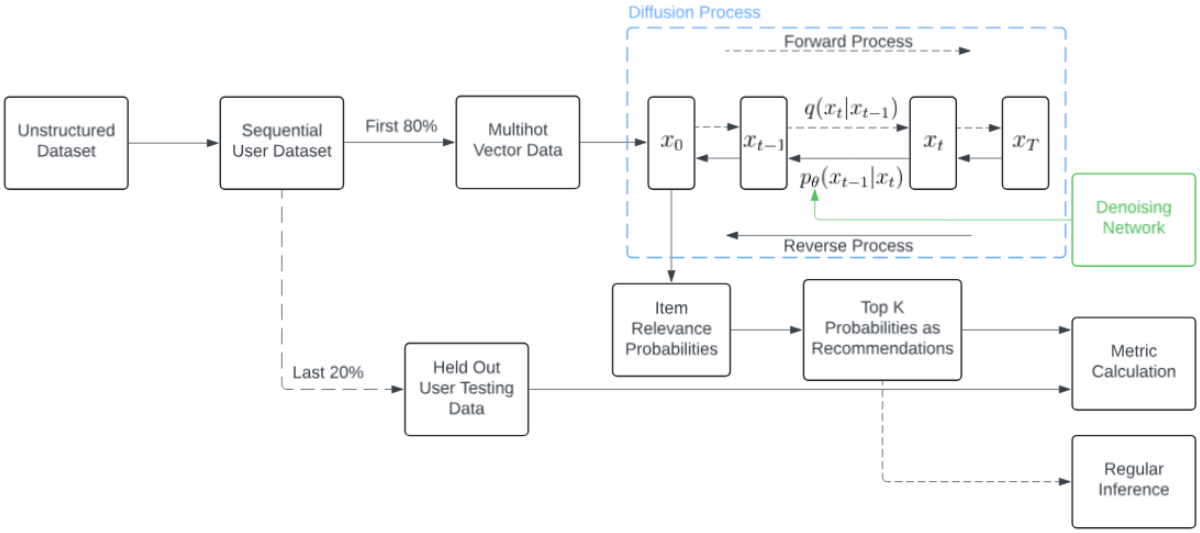

Figure 1: Diagram of Our Recommender System

The approach taken has 2 main components that we will talk about in this section, detailed in green and blue in Figure 1 above. The first component is the diffusion process, which is responsible for the forward process, the reverse process, and the sampling procedure. The reverse process makes use of the denoising model utilized within the reverse process but will be addressed as a separate component. This work is based on the work of (Wang Wenjie, 23) and uses their code as a baseline to make modifications. The original codebase had multiple iterations of the diffusion model used, such as the baseline DiffRec, L-DiffRec with the forward and reverse processes happening in the latent space borrowing from autoencoders, T-DiffRec making use of timestep weighting in the denoising model, and LT-DiffRec, a combination of the previous two. For the purposes of this thesis, only the baseline configuration will be used for simplicity. The original work does not perform their guidance, or as we prefer to refer to it, pseudo-guidance, with any external guidance, only the partially destroyed original data that can

give clues to the denoising model in the reverse process. Thus, how guidance is administered in the reverse process must be altered to implement classifier-free guidance. In addition to detailing how we implemented this external guidance in this section, we will also touch on the options available for this type of task, the modifications made to the model architecture on their own and also as a result of the form of guidance, and how the diffusion process works.

Modifications included implementing classifier-free guidance, guidance through concatenation, guidance through cross-attention, denoising model architecture, diffusion process, etc. As you may recall, their work utilizes only partially noised data as their guidance, essentially corrupting data but not rendering it completely meaningless. This was their proposed guidance, and as a result of the method used, the guidance is constant throughout training and cannot be removed to implement classifier-free guidance. Thus, some modifications were required to be made to their method. Firstly, we can no longer use partially noised data as the guidance as we require a new method of guidance that is separate. My solution was to simply use the methods of previous papers and take guidance from the data pre-noised. The exact form of this guidance will be detailed later. With this new form of guidance, we can now implement classifier-free guidance into the training.

3.1 Diffusion Process

The diffusion process is split into a forward and reverse process. Since the modifications we made are heavily weighted on the reverse process and the denoising model, we will detail the reverse process in depth in the next section and briefly mention how it integrates as a whole in this section. If we want to destroy a distribution that is not random, like, for instance, an image,

we can destroy it by adding noise to it, pushing the image towards pure noise. The forward and reverse processes will have different notations; the forward process can be described as $q(x_t|x_{t-1}) = \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, q, x_t is the output of the forward process at step t, since this is similar to a Markov chain and the output depends on the previous input, naturally x_{t-1} will be the input at time step t , $\mathcal{N}$ is to denote a normal distribution. p will be the reverse process since, in the reverse, we are predicting the noise and trying to recreate q whereas q is just a function we determine ahead of time. Since our formula for q at timestep t is a normal distribution, we have the two parameters of a normal distribution, mean and variance, respectively. β_t in both mean and variance refers to a scheduler we have.

The Stable Diffusion paper uses a linear schedule, and it has become a popular one; although there are other viable schedules, our program also uses a linear schedule to keep things simple. The linear schedule is, as its name suggests, a linear schedule for adding noise to an image or, in our case, a sequence of interactions. This linear schedule of adding noise can be described as a Markov chain. A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the previous event. Similarly, this linear schedule is iterative like a Markov chain and each timestep depends on the previous to add the correct amount of noise, but this process would be slow during training if we did not have some workaround. Luckily, we don't have to do this, thanks to the Stable Diffusion paper by (Ho et al., 20). Instead of iterating through t steps to generate every training sample, let's do it in one fell swoop. We can define the entire noise to be added by the scheduler at the final timestep by $q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|q_{t-1})$; we will also need the reparameterization trick here defined as follows: $\mathcal{N}(\mu, \sigma^2) = \mu + \sigma * \epsilon$. Referring again to the single forward-step

definition provided in the previous paragraph, we can rework it to $\sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$ using the reparameterization trick, ϵ is sample noise. The authors of the Stable Diffusion paper introduce some notation to help ease the next transformation: $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. Using this new notation our formula for the forward process at timestep t looks like this $q(x_t|x_{t-1}) = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon$. We can extend this formula to previous timesteps now, allowing us to compress all the α values to one formula starting at x_0: $q(x_t|x_0) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon = \mathcal{N}(x_t, \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$. With this new formula, we can calculate the noise to be added at any arbitrary step without going through all the previous steps, thus expediting the process during training.

The reverse process is designed to attempt to actually recreate the forward process of noising but, in reverse, to create some meaningful data from the original distribution. This is where the magic happens, and of course, where there is magic in the modern day, there is a neural network powering it. This neural network is known as a denoising network and was originally a U-Net architecture after some very smart people discovered it was good at removing noise from images. Our architecture differs since we are not working with images but sequences, however, the goal of the use of a neural network for denoising is the same. We won't be going into depth on the reverse process here since the denoising network is core to the idea of the reverse process, and the next section is specifically for the denoising network.

3.2 Denoising Model / Reverse Process

The main modifications made were to the denoising model, and in this section, we will go into depth about what those modifications were and how we executed them. Firstly, we will

detail what exactly the purpose of the denoising model is in diffusion models and how, if any it differs for recommender systems purposes. We will detail the shortcomings of the original denoising model for the new functionality we want to add (classifier-free guidance), and finally, how we fixed those shortcomings and what experimental configurations we performed that ultimately led to the final settled-upon model architecture. Experimental tests on architecture that did not result in the desired effect of improved performance and were not included in the final implementation will still be detailed briefly in section 4.2.

As detailed in the previous section, the denoising model is core to the reversal of noise added in the forward process. In image diffusion models, some form of U-Net is typically used as it's been shown to perform well on noise removal tasks, but we can use something simpler for our data. Since our data is not extremely vast like image data, we can opt for something unlike U-Net, like for instance, a simple feedforward network. The purpose of this model is to predict all of the noise to be removed at a timestep to get to timestep 0 (fully reversed), and once predicted, we take only a portion of that predicted noise depending on the timestep and remove it. Beginning from the equation for the reverse process given by the original Stable Diffusion paper (Ho et al., 20), $p_\theta(x_0:T) := p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$ where we can define $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}, \mu_\theta(x_t,t), \sum_\theta(x_t,t))$. This says that $p_\theta(x_0:T)$, otherwise known as the diffusion process, is simply a chain of Gaussian transitions starting at $p(x_T)$ that iterates to T using $p_\theta(x_{t-1}|x_t)$ equation for each singular step. We need to discuss now the components of the formula of $p_\theta(x_{t-1}|x_t)$. The formula has 2 parts: $\mu_\theta(x_t,t)$ the mean and $\sum_\theta(x_t,t))$ the variance at a given timestep. The variance is time-dependent; as you can see, it depends on the timestep but is not trainable. Rather than being set to a constant, it is equal to $\beta_T I$ from the

schedule described in the forward process. Now we only need to worry about the mean of each step; luckily, the authors of the stable diffusion paper have already done the heavy lifting for us, and all we need to know is this formula: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t))$ which gives us $x_{t-1} = \mathcal{N}(x_{t-1}, \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_T}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \sqrt{\beta_t \epsilon}$ and from here we can use this to calculate the output for a given timestep t: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}}}\epsilon_\theta(x_t, t)) + \sqrt{\beta_t \epsilon}$ where $\epsilon_\theta(x_t, t)$ is our model's output or in this case the predicted noise. We simply use this to reverse the diffusion process iteratively using the predicted noise from our denoising model.

Now, we will discuss the model architecture, what modifications were made, and how. The original paper model architecture was a simple input to an embedding layer back to the output with embedding dimensions of 1000, essentially mimicking the autoencoder structure except without any convolutions or max-pooling layers, obviously, since we are not dealing with image data but sequence data. There's no inherent problem with this architecture for their implementation, but we decided to make some modifications that improved performance alongside the addition of classifier-free guidance, which saw performance improvements as well. Firstly, what we provided the model needed to be modified to include an un-noised iteration of the batch of data we are working with to act as our guidance for conditioning. Beforehand, the data being partially noised acted as the pseudo-guidance, but now we require actual external guidance to implement classifier-free guidance. The forward method takes our noisy batch of data and the same unnoised batch of data, which we will refer to as guidance from now on. It also takes a parameter pertinent to classifier-free guidance known as p_uncond; this acts as the ratio of the data where the gradients from the guidance will be zero to act as a non-conditional prediction, essentially removing the guidance for a ratio of the data which is the core idea behind

classifier-free guidance, this ratio usually sits around .2 which is what we use as well. 20% of the guidance becomes zero gradients; we then normalize the data and concatenate it with the noised data (x). Thus, during training, we are training our denoising model to make recommendations without any guidance and with guidance at the same time, improving the model's capability as proven by classifier-free guidance. As mentioned before, there are other ways to condition the denoising model, such as attention but the best-performing method we found also ended up being the simplest, thus, concatenation of the data and guidance instead. We keep the embedding size of the original paper so the concatenated data is transformed into the dimensions of batch size * embedding size and then run through a hyperbolic tangent activation function and then one more linear layer to project the data back to the dimensions of the number of the items in the dataset providing us with essentially a heatmap of outputs for each item in the dataset. The model is relatively lightweight, and we will detail the reasons behind this in section 4.2.

4. Results

In this section, we will discuss the specifics of the datasets used and how they pertain to our training and testing. We will also discuss the configuration of the experiments we ran when discovering the best method for personalizing recommendations through conditioning and finding the best denoising model architecture. Lastly, the actual results of our chosen architecture and method of guidance in comparison to the original implementation that we based ours upon, as well as the special use cases we have identified that our modified system excels in.

4.1 Datasets

There were multiple datasets used for experiments, training, and testing during the development phase of this thesis. The main dataset used was the MovieLens 1 Million dataset, which consists of 1 million movie ratings, ranging from 1 to 5, from 6000 users on 4000 movies. There are two formats of training specified in the inspiration paper: clean and noisy training. We did some light experimentation into the capabilities of our modifications in both formats, however, it is not utilized outside of the datasets provided by the original paper, and all outside datasets that we personally created will only consist of the "clean" format. We found nothing noteworthy specifically in regard to our modified implementation on either noisy or clean datasets when compared to the original method. Both the MovieLens and Yelp datasets provided by the original implementation have a noisy and clean setting; however, only the MovieLens noisy configuration will be used as the Yelp noisy dataset required too much computing power from the machine used to train on all the datasets. Clean training is described as only utilizing ratings where the user rated a 4 or 5. In this situation, we can consider a rating of 4 or higher as a hit or an item the user would likely recommend to someone else and/or rate highly. Obviously, any item rated 3 or below is considered a miss in this scenario. The configuration of the model does not take into account non-positive ratings (3 or below), so anything considered a hit will be represented as a 1 in a vector of length N, where N is the number of unique items in the dataset, and anything 3 or below, including unrated items by a user, are grouped into the same category and labeled 0. The two original datasets provided by the research included in the experimentation, MovieLens 1 Million and Yelp have numeric ratings that we can use when creating our datasets so there isn't a situation where we have to guess the process others took when truncating the reviews to hits and misses. Unfortunately, we use a range of other Amazon

datasets, but the actual numeric ratings are not provided so all provided reviews are considered hits. We can assume the training data to be slightly noisy but in our experiments, it does not seem to negatively impact the perceived differences between the original method and our modifications, therefore it is not something to worry about and we can proceed without fear. The training testing split is an 80:20 split when only training and testing are used and 70:20:10 when training validation and testing are all used. The last 20% of each sequence, or 30% in the case of training validation and testing, is held out for each input sequence corresponding to a user used when training the model.

Dataset	Users	Items	Total Reviews
ML-1M Clean	5949	2810	403,277
ML-1M Noisy	5949	3494	429,993
Yelp Clean	54,574	34,395	1,014,486
Amazon Kitchen	11,566	7,722	100,464
Amazon Beauty	3,782	2,658	38,867
Amazon Office	1,719	901	21,466
Amazon Toys	2,676	2,474	26,850
Amazon VG	5,435	4,295	60,497

Figure 2: Dataset Details

When creating the datasets, most consist of a raw format of user ID, item ID, some rating 1-5 (for all datasets used), time of review, and some extraneous data that is not important to us and is cut from each entry. Firstly, since we are training in a “clean” format, all entries with ratings 3 and below are instantly discarded; as we mentioned before, our method does not take into account non-positive reviews, non-positive being a miss, and a miss being a rating 3 and

below. Once these reviews are gone, we have no more need for the rating column, which is also dropped. For datasets that do not have a numeric rating to work with, we simply begin from here, assuming all the reviews provided have a positive correlation between the user and the item they reviewed, with some allowance for small amounts of noise. The data is then sorted by user ID and then by review time so that we have the format of index 1-k being an arbitrary user's sequential list of interactions, starting with the oldest and only ratings rated 4 or higher (otherwise known as hits). The user and item IDs are then renumbered to begin at 0 so that we can easily obtain the user and item count later since the item and user count would no longer be accurate after entries have been removed. Once all this processing has been done, all that remains is to split the data into training, validation, and testing, and we simply do this by taking the first 80% or 70%, depending on how we decide to split and separate them into their respective lists. At this point, all further processing to the dataset will be done at runtime, and before then, we are left with a dataset of time-sequential user data that we have split so that all recommendations can be compared with the newest user data we have based on their old ratings.

The datasets are prepared for training by taking the previous format we created for them before runtime. If you recall, this consists of a simple list of user-item interactions (confirmed hits) that are sorted by user and then chronological time, with only user ID and item ID as relevant data at this point. This format is inadequate for training since we need the users to be differentiable by entry and not a continuous flow into the next user from the current. Therefore, we transform the data into a sparse format, known as multi-hot vectors; unlike one-hot vectors where we have just 1 value to indicate where our hits are among the items for a specific user, we instead have multiple 1’s to indicate all the hits or interactions for specific users. Obviously, if

we are using a method of this kind, the vector length must be the length of the number of items in the dataset to represent the items themselves. At this point, the model is not making recommendations based on an exact sequence, but we do, however have past data to make recommendations and future data to compare our recommendations to; it simply wouldn't make sense to do it any other way.

4.2 Experimental Configurations

In this section, we will detail some of the experiments we performed to find the best-performing recommender system, what worked, what didn't, and why we think they did or didn't. Most of the experiments for this work involved modifications to either the guidance going to the denoising model or the denoising model itself. Regardless, some small experiments were performed to determine the capability of certain functionalities. Among those was making the recommendation model aware of the actual numeric ratings as opposed to the current implementation of 1's and 0's representing hits and misses. The implementation was relatively simple to create this, where currently there are 0's that represent both items that a user did not enjoy and or did not rate; it would only represent the prior in this scenario, we would have negative signals (1-3) for items as well as the positive signals we were working with before (4-5). The resulting recommendations were subpar and not comparable to any other results obtained and, therefore, will not be compared. It is unclear why this was the case, but the idea was scrapped. Our hypothesis is that due to there being far more negative ratings than positive ratings, the signal for what constitutes what a user likes is lost in the sea of recommendations.

For most experiments, we found a general rule that seemed to apply regardless of how it was implemented: that any significantly larger parameter count greatly reduced effectiveness. We believe this to be the result of overfitting in our testing; due to the iterative nature of the denoising process, a large model was not required to remove small chunks of noise from the data, and because we had far more parameters than were needed it would begin to attempt to essentially memorize instead of finding a general rule for reversing the forward process. As a result of this, all configurations with more layers did not perform well, and more neurons per layer had no positive effect. We also experimented with attention, both self-attention for the input sequence and cross-attentive conditioning from the guidance. In general, the model did not benefit as much as expected from having attention and we eventually opted for concatenation for incorporating the guidance with the input data. This could be in part because of the prior rule we found to be true in that having more parameters is not necessarily a good thing and including attention adds parameters. Different embeddings for the guidance were used with varying success, mainly prevailing as beneficial when an exceptionally large dataset was used. Still, embeddings had a slight disadvantage to incorporating the raw data in terms of effectiveness but did show improved efficency. However, if this system were to be scaled for use on large datasets, an embedding would definitely be recommended, but the largest of our datasets is the Yelp dataset and it performs well without embeddings.

4.3 Training Results

For all datasets, we save the highest-performing model based on nDCG @ 10. For all metrics described, we calculate based on K @ [1,5,10,20]; we originally had higher values but, after deliberation, settled on the importance, specifically for recommender systems, to create

valuable recommendations within this range because the average user is not going to search through 50 or 100 potential recommendations. All configurations were trained for varying times depending on the size of the dataset but all were well beyond the point of finding the best-performing results on the validation/testing dataset. Some of the results you will see yield interesting insights for a particular scenario that our model may excel in, in comparison to other methods, and especially in comparison to the original method. The hyperparameter configuration from the original work to our modified implementation does not change aside from the new hyperparameter that we require for classifier-free guidance: p_uncond @ 0.2. In the following figures, any underlined value is a score that beats the other implementation or is tied with the other.

Dataset	ML-1M_Clean		ML-1m_Noisy		Yelp_Clean		Amazon Kitchen	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
Precision @ [1,5,10,20]	0.0864	0.0925	0.0299	0.0463	0.028	0.0261	0.0035	0.0052
	0.0792	0.0861	0.0337	0.036	0.0222	0.208	0.0021	0.0024
	0.0716	0.0783	0.0346	0.0357	0.019	0.0187	0.0014	0.0016
	0.0654	0.0682	0.0345	0.0355	0.0161	0.0161	0.001	0.001
Recall @ [1,5,10,20]	0.0069	0.0092	0.0029	0.0061	0.0048	0.0048	0.0013	0.0039
	0.0318	0.0396	0.0151	0.0222	0.0185	0.0177	0.0042	0.0056
	0.0549	0.066	0.0304	0.037	0.0312	0.0311	0.0062	0.0068
	0.0972	0.1166	0.0555	0.0657	0.0514	0.0519	0.0089	0.0079
nDCG @ [1,5,10,20]	0.0864	0.0925	0.0299	0.0463	0.028	0.0261	0.0035	0.0052
	0.0833	0.09	0.0349	0.0418	0.0267	0.0253	0.0035	0.0055
	0.0843	0.0932	0.0397	0.0458	0.0294	0.0288	0.0041	0.0059
	0.0944	0.1056	0.0483	0.0557	0.0356	0.0353	0.005	0.0064
MRR	0.0864	0.0925	0.0299	0.0463	0.028	0.0261	0.0035	0.0052

@ [1,5,10,20]	0.1541	0.1639	0.0678	0.0827	0.0529	0.0494	0.0053	0.007
	0.1706	0.1827	0.0834	0.0973	0.0604	0.0577	0.0057	0.0074
	0.1815	0.1933	0.0938	0.109	0.0662	0.0635	0.0061	0.0077

Figure 3: Dataset Results

Dataset	Amazon Beauty		Amazon Toys		Amazon Office		Amazon VG	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
Precision @ [1,5,10,20]	0.0093	0.0106	0.0112	0.0075	0.0029	0.0058	0.0147	0.0129
	0.0087	0.009	0.0045	0.003	0.0035	0.0052	0.009	0.0109
	0.0083	0.0081	0.0032	0.0026	0.0026	0.0061	0.009	0.0084
	0.0062	0.0067	0.002	0.0023	0.0017	0.0044	0.007	0.0071
Recall @ [1,5,10,20]	0.0038	0.0042	0.0042	0.0047	0.0004	0.001	0.0082	0.0072
	0.0164	0.0171	0.0097	0.0064	0.0074	0.0065	0.0232	0.0263
	0.0344	0.0318	0.0115	0.0136	0.0108	0.0177	0.0463	0.0412
	0.0519	0.0528	0.0137	0.022	0.0134	0.0257	0.068	0.0699
nDCG @ [1,5,10,20]	0.0093	0.0106	0.0112	0.0075	0.0029	0.0058	0.0147	0.0129
	0.0137	0.0142	0.0094	0.0065	0.0061	0.0061	0.0194	0.0203
	0.0205	0.0192	0.0098	0.0089	0.0072	0.0106	0.0275	0.0254
	0.0258	0.0257	0.0104	0.0115	0.0082	0.0135	0.034	0.0349
MRR	0.0093	0.0106	0.0112	0.0075	0.0029	0.0058	0.0147	0.0129

@ [1,5,10,20]	0.0194	0.0176	0.0154	0.0087	0.0095	0.0109	0.0252	0.0261
	0.0222	0.0214	0.0159	0.0102	0.0105	0.0142	0.0307	0.0294
	0.0236	0.0232	0.0161	0.0114	0.011	0.0154	0.0337	0.0327

Figure 4: Dataset Results

Our system performs on par with, if not better, in most scenarios. You will notice when looking at the results that when our method is beaten, it is not by a sizeable margin, but when ours outperforms the previous method, it is by a larger margin than we are beaten. This is true for almost all scenarios, aside from the occasional outlier. We outperform the original implementation in 84 of the 128 categories for all datasets, showing a clear improvement upon the original method. We can also see on datasets where the item selection is more sparse, and our method beats the previous by a factor of 2, if not more, in some situations. This is true for any of the datasets we selected that are of this form. This is clearly a result of the improved guidance, as on sparse datasets, there is not enough guidance for personalizing recommendations with the previous method. Although we did not test this specifically, this lends to the idea of a capable, zero-shot, or few-shot recommendation system. When training, the model is trained with and without guidance simultaneously, essentially a user with little to no history for us to draw upon; we can utilize its capabilities for making recommendations when there is little to no guidance. We already perform this during training regularly, we are just not sure of its capabilities as this was not something we tested for specifically.

5. Conclusions

This thesis successfully implements a new form of conditioning known as classifier-free guidance as a form of personalization for recommendations and optimizes the architecture of the denoising model in diffusion-based recommender systems. Our process involved modifying the structure of the guidance by removing partial noise as pseudo-guidance and including true guidance through conditioning of our denoising model on pre-noise data. The effectiveness of our method is tested through precision, recall, nDCG, and mean reciprocal rank @ K on the held-out data of each user. It is clear to see the benefits of our method when compared with the previous, and we also demonstrate an increased capability in new scenarios where the previous model fell short, such as few-shot or potentially zero-shot recommendations. This scenario of few-shot recommendations could be the result of either a sparse selection of items when compared to the users or a particular user did not make many reviews; either scenario we are better equipped to handle with our new method. This was shown on any dataset we tested that had a sparse item review count per user.

Our major contributions include: the implementation and interpretation of classifier-free guidance in diffusion recommendation tasks, a discussion on the parameterization and architecture of the denoising model used in diffusion models that are utilized for tasks like recommendation systems, and lastly, insight into the capabilities of diffusion models into tasks outside of the visual domain as they are so commonly used for now. To summarize, we learned that the classifier-free guidance also boasts performance improvements in recommendation tasks as it does in visual tasks. There is room for improvement in diffusion recommendation systems, and we provide a framework forward through these difficulties, and in addition to the

performance improvements in a general use-case, classifier-free guidance provides us with new capabilities in few-shot and zero-shot scenarios.

Future Work

The potential for future work and improvements in this area is high. Our belief is that a system that is aware of actual numerical ratings could potentially perform better than our implementation if done properly. We did not perform a thorough exploration in this area, but an implementation that utilizes triplet loss to account for the present but negative reviews as opposed to the "hits" that we would normally have to make recommendations for could solve the issues we had. The hope is that in this scenario, the performance would improve, because not only would we know what to avoid but also what to recommend improving general performance across the board. The cross-attentive conditioning is another feature we were surprised to have failed in terms of performance in relation to its counterpart, but we believe there is an architecture that would have greater performance using cross-attentive methods for guidance; we just did not find it in our own exploration. The use of transformers in the diffusion process is another innovation made recently with respect to image generation diffusion models where instead of a U-Net backbone architecture for the denoising model, a Vision Transformer (ViT) is used to perform the denoising, and this could have applications in this area of work. We would have liked to experiment with this possibility given more time, however, for the scope of this work, it was not included. We would, however, encourage future research in this direction for this work specifically. Lastly, an area that we heavily think should be explored further is the few-shot and zero-shot capabilities of a diffusion recommender system trained with classifier-free guidance. The implication of its potential is obvious when you think about what

exactly classifier-free guidance is doing. This area of work is also heavily desired, as this is a very common scenario in e-commerce that can be costly to providers and overall undesirable while also being simultaneously unavoidable as new users joining with no history will always be a point of trouble.

References

(Rich, 79) E. Rich, “User modeling via stereotypes,” *Cognitive Science*, Volume 3, Issue 4, 1979.

(Karlgren, 90) J. Karlgren, "An Algebra for Recommendations," *Syslab Working Paper* 179, 1990.

(Goldberg et al., 92) D. Goldberg, D. Nichols, B. M. Oki, D. Terry, “Using collaborative filtering to weave an information tapestry,” *Communications of the ACM*, Volume 35, Issue 12, pp 61-70, 1992.

(Sohl-Dickstein et al., 15) J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, S. Ganguli, “Deep Unsupervised Learning using Nonequilibrium Thermodynamics,” *CoRR*, abs/1503.03585, 2015.

(Ho et al., 20) J. Ho, A. Jain, P. Abbeel. “Denoising Diffusion Probabilistic Models,” *CoRR*, abs/2006.11239, 2020.

(Dhariwal, Alex, 21) P. Dhariwal, A. Nichol. “Diffusion Models Beat GANs on Image Synthesis,” *CoRR*, abs/2105.05233, 2021.

(Ho et al., 21) J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, T. Salimans, “Cascaded Diffusion Models for High Fidelity Image Generation,” *CoRR*, abs/2106.15282, 2021.

(Saharia et al., 21) C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, “Image Super-Resolution via Iterative Refinement,” 2021.

(Rombach et al., 22) R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *CoRR*, abs/2112.10752, 2022.

(Saharia et al., 22) C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. Kamyar, S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, M. Norouzi, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," 2022.

(Ramesh et al., 22) A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, "Hierarchical Text-Conditional Image Generation with CLIP Latents," 2022.

(Betker et al., 23) J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, W. Manassra, P. Dhariwal, C. Chu, Y. Jiao, "Improving Image Generation with Better Captions," 2023.

(Gu et al., 22) S. Gu, D. Chen, J. Bao, F. Wen, B. Zhwang, D. Chen, L. Yuan, B. Guo, "Vector Quantized Diffusion Model for Text-to-Image Synthesis," *CoRR*, abs/2111.14822, 2022.

(Y. Li et al., 23) Y. Li, H. Wang, Q. Jin, J. Hu, P. Chemerys, Y. Fu, Y. Wang, S. Tulyakov, J. Ren, "SnapFusion: Text-to-Image Diffusion Model on Mobile Devices with Two Seconds," 2023.

(Wang Wenjie et al., 23) W. Wang, Y. Xu, F. Feng, X. Lin, X. He, T. Chua, "Diffusion Recommender Model," *SIG IR*, 2023.

(Wang et al., 23) Y. Wang, Z. Liu, L. Yang, P. S. Yu, "Conditional Denoising Diffusion for Sequential Recommendation," 2023.

(Lin et al., 23) X. Lin, X. Chen, C. Wang, H. Shu, L. Song, B. Li, P. Jiang, "Discrete Conditional Diffusion for Reranking in Recommendation," 2023.

(Z. Li et al., 23) Z. Li, A. Sun, C. Li, "DiffuRec: A Diffusion Model for Sequential Recommendation," 2023.

(Salimans et al., 16) T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, “Improved Techniques for Training GANs,” *CoRR*, abs/1606.03498, 2016.

(Resnick et al., 94) P. Resnick, N. Lacovou, M. Suchak, P. Bergstrom, J. Riedl, “GroupLens: An Open Architecture for Collaborative Filtering of Netnews,” *ACM*, pp 175-186, 1994.

(Harper, Konstan, 15) F. M. Harper, J. A. Konstan, “The MovieLens Datasets: History and Context,” *ACM*, Volume 5, Issue 4, Article No.: 19, pp 1-19, 2015.

(Koren, 08) Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” *ACM*, pp 426-434, 2008.

(Koren, 10) Y. Koren, “Collaborative filtering with temporal dynamics,” *ACM*, pp 447-456, 2010.

(Richardson et al., 07) M. Richardson, E. Dominowska, R. Ragno, “Predicting clicks: estimating the click-through rate for new ads,” *ACM*, pp 521-530, 2007.

(Liang et al., 18) D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, “Variational Autoencoders for Collaborative Filtering,” 2018.

(Gao et al., 20) M. Gao, J. Zhang, J. Yu, J. Li, J. Wen, Q. Xiong, “Recommender Systems Based on Generative Adversarial Networks: A Problem-Driven Perspective,” 2020.

(Covington et al., 16) P. Covington, J. Adams, E. Sargin, “Deep Neural Networks for YouTube Recommendations,” *ACM*, pp 191-198, 2016.

(Kingma, Welling, 2013) D. P. Kingma, M. Welling, “Auto-Encoding Variational Bayes,” 2013.

(Saad et al., 23) M. M. Saad, R. O’Reilly, M. H. Rehmani, “A Survey on Training Challenges in Generative Adversarial Networks for Biomedical Image Analysis,” 2023.

(Bond-Taylor et al., 22) S. Bond-Taylor, A. Leach, Y. Long, C. G. Willcocks, “Deep Generative Modeling: A Comparative Review of VAEs, GANs, Normalizing FLows, Energy0Based and Autoregressive Models,” *CoRR*, abs/2103.04922, 2022.

(Goodfellow et al., 14) I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative Adversarial Nets,” 2014.