# Diffusion Prior for Online Decision Making: A Case Study of Thompson Sampling

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

In this work, we investigate the possibility of using denoising diffusion models to learn priors for online decision making problems. Our special focus is on the meta-learning for bandit framework, with the goal of learning a strategy that performs well across bandit tasks of a same class. To this end, we train a diffusion model that learns the underlying task distribution and combine Thompson sampling with the learned prior to deal with new task at test time. Our posterior sampling algorithm is designed to carefully balance between the learned prior and the noisy observations that come from the learner's interaction with the environment. Preliminary experiments clearly demonstrate the potential of the considered approach.

## 1 Introduction

Uncertainty quantification is an integral part of online decision making and forms the basis of various online algorithms that trade-off exploration against exploitation. Among these methods, Bayesian approaches allow us to quantify the uncertainty using probability distributions, with the help of the powerful tools of Bayesian inference. Nonetheless, their performance are known to be sensitive to the choice of prior.

For concreteness, let us consider the problem of stochastic multi-armed bandits (MABs) [2, 9], in which a learner repeatedly pulls one of the $K$ arms from a given set $\mathcal{A} = \{1, ..., K\}$ and receives rewards that depend on the learner's choices. More precisely, when arm $a$ is pulled at round $t$, the learner receives reward $r_t$ drawn from an arm-dependent distribution $\mathcal{P}^a$. The goal of the learner is either to *i*) accumulate the highest possible reward over time (a.k.a. regret-minimization) or to *ii*) find the arm with the highest expected reward within a prescribed number of rounds. (a.k.a. best-arm identification).

For both purposes, we need to have a reasonable estimate of the arms' mean rewards $\mu^a = \mathbb{E}_{r^a \sim \mathcal{P}^a}[r^a]$. In general, this would require us to pull each arm a certain number of times, which becomes inefficient when $K$ is large. Moreover, unfortunately, the no-free-lunch principle precludes the possibility of improving upon this bottleneck, since an algorithm that performs better on one instance necessarily sacrifices its performance on the other. In spite of the said limitation, the bandit instances (referred as tasks hereinafter) that we encounter in most practical situations are far from arbitrary. To name a few examples, in recommendation systems, each task corresponds to a user with certain underlying preferences that affect how much they like each item; in online shortest path routing we operate in real-world networks that feature specific characteristics.

In this regard, introducing such *inductive bias* to the learning algorithm would be beneficial. In Bayesian models, this can be expressed through the choice of the prior distribution. Moreover, as suggested by the mete-learning paradigm, the prior itself can also be learned from data, which often leads to superior performance [6, 12].

Under review at the NeurIPS 2022 Workshop on Score-Based Methods. Do not distribute.

**Our contributions.** This work tackles the problem of meta-learning the priors for bandits [1, 3, 10], using Thompson sampling as the base algorithm in each task. In order to approximate the complex prior that arise in practice, we build upon the powerful tools of deep generative modeling, whose recent progress have enabled impressive results in various areas ranging from image generation [14] to protein design [20]. Specifically, we propose to meta-learn the prior using denoising diffusion models [5, 16], and develop an algorithm to perform Thompson sampling under the learned prior. The designed algorithm strikes a delicate balance between the learned prior and bandit observations, bearing in mind the importance of having an accurate uncertainty estimate. Through synthetic experiments, we demonstrate the benefit of the considered approach against several baseline methods.

## 2 Preliminaries and Problem Description

In this section, we briefly review denoising diffusion models and introduce our meta-learning framework over a class of bandit tasks.

### 2.1 Denoising Diffusion Probabilistic Model

First introduced by Sohl-Dickstein et al. [16] and recently popularized by Ho et al. [5] and Song and Ermon [17], denoising diffusion models (or the closely related score-based models) have been shown to achieve state-of-the-art performance in various data generation tasks. Numerous variants of these models have been proposed. Below, we mainly adopt the notations and formulation of Ho et al. [5], with minimal modifications to adapt it to our purpose.

Intuitively speaking, diffusion models learn to approximate a distribution $\mathcal{Q}_0$ by training a series of denoisers with samples drawn from this distribution. Writing $q$ for the probability density function (assume everything is Lebesgue measurable for simplicity) and $X_0$ for the associated random variable, we define the forward diffusion process with respect to a sequence of scale factors $(\alpha_\ell) \in (0,1)^L$ by

$$q(x_{1:L} \,|\, x_0) = \prod_{\ell=0}^{L-1} q(x_{\ell+1} \,|\, x_\ell), \qquad q(X_{\ell+1} \,|\, x_\ell) = \mathcal{N}(X_{\ell+1}; \sqrt{\alpha_{\ell+1}} x_\ell, (1-\alpha_{\ell+1})I).$$

The first equality suggests that the forward process is Markovian, while the second equality implies that the transition kernel is Gaussian. Further denoting the product of the scale factors by $\bar{\alpha}_\ell = \prod_{i=1}^{\ell} \alpha_i$, we then have $q(X_\ell \,|\, x_0) = \mathcal{N}(X_\ell; \sqrt{\bar{\alpha}_\ell} x_0, (1-\bar{\alpha}_\ell)I)$.

The sequence $(\alpha_\ell) \in (0,1)^L$ is chosen to be decreasing and such that $\bar{\alpha}_L \approx 0$. We thus expect $q(X_\ell) \approx \mathcal{N}(0,1)$. A denosing diffusion model learns to reverse the diffusion process with a distribution $\mathcal{P}_\theta$ over random variables $X'_{0:L}$, in the hope that the marginal distribution $\mathcal{P}_\theta(X'_0)$ is a good approximation of $\mathcal{Q}_0$. This is achieved by setting $p_\theta(X_\ell) = \mathcal{N}(0,1)$, enforcing the learned reverse process to be Markovian, and modeling $p_\theta(X_\ell \,|\, x_{\ell+1})$ as a Gaussian parameterized by

$$p_\theta(X_\ell \,|\, x_{\ell+1}) = q(X_\ell \,|\, x_{\ell+1}, X_0 = \underbrace{h_\theta(x_{\ell+1}, \ell+1)}_{\hat{x}_0}) \propto \underbrace{q(x_{\ell+1} \,|\, X_\ell) q(X_\ell \,|\, X_0 = \hat{x}_0)}_{\text{both are Gaussian by construction}}. \tag{1}$$

In the above $h_\theta$ is the learned denoiser and $h_\theta(x_{\ell+1}, \ell+1)$ is the predicted clean sample.[1]

### 2.2 Meta-Learning of Bandit Tasks

We consider a meta-learning for bandits framework in which the bandit tasks are drawn from an underlying distribution $\mathcal{T}$. Unlike [3, 10], we focus on the a more standard scenario which features a meta-train and a meta-test phase. The goal is to learn a good prior using the training set such that we can perform better once the algorithm is deployed with the learned prior in the test phase. Mathematically, we can characterize the performance of the algorithm with the so-called *transfer regret* [3], defined by $\mathrm{Reg}_T(\pi) = \mathbb{E}_{B\sim\mathcal{T}} \mathrm{Reg}_T(\pi, B)$ where $\pi$ is the algorithm that uses the learned prior and $\mathrm{Reg}_T(\pi, B)$ is the regret incurred by the algorithm within task $B$ (see Appendix B).

Throughout the work, we assume that the noise in the rewards are Gaussian with known variance $\sigma^2$. The learner thus only needs to learn the vector of the mean rewards $\mu = (\mu^a)_{a\in\mathcal{A}}$. Accordingly, the prior is defined as a distribution in $\mathbb{R}^K$.

[1] To obtain $h_\theta$ we typically train a neural network with a U-Net architecture. In [5], this network is trained to output the predicted noise $\bar{z}_\ell = (x_\ell - \sqrt{\bar{\alpha}_\ell} h_\theta(x_\ell, \ell))/\sqrt{1-\bar{\alpha}_\ell}$.

# 3 Algorithm

In this section, we describe our backbone algorithm for Thompson sampling with diffusion prior.

## 3.1 Diffusion Model as Prior

Our training data for the diffusion prior can have different forms. For example, these data may be composed of the interaction history with tasks using some bandit algorithms. How to leverage such confounded and noisy data to learn a diffusion model is a challenging problem. In this short paper, we simply assume that the mean vectors of these tasks are known, from which we can apply the standard diffusion model training process.

Once the diffusion model is learned, we may use it directly as a prior in any downstream tasks. Nonetheless, the original sampling process of the diffusion model does not take the quality of the learned model into account, and overly relying on a bad prior does more harm than good. To address this problem, we propose to inject more noise in the sampling process when the model is of poorer quality. Precisely, we write

$$p_\theta(X_\ell \mid x_{\ell+1}) = \int q(X_\ell \mid x_{\ell+1}, x_0) p'_\theta(x_0 \mid x_{\ell+1})\, dx_0. \tag{2}$$

In the above, $p'_\theta(X_0 \mid x_{\ell+1})$ is a Gaussian distribution centered at $\hat{x}_0 = h_\theta(x_{\ell+1}, \ell+1)$. This is different from (1) where by analogy we may interpret $p'_\theta(x_0 \mid x_{\ell+1})\, dx_0$ as a Dirac. As learning the covariance of the reverse process has been shown to result in poor performance in [5], the covariance of $p'_\theta(X_0 \mid x_{\ell+1})$ is simply taken as a diagonal matrix $\mathrm{diag}(\tau^2_{\ell+1})$ whose diagonal elements are the (coordinate-wise) mean squared reconstruction errors of the model on a calibration set $\mathcal{V}$. That is, we construct $\tilde{\mathcal{V}}_\ell$ containing pairs $(x_0, x_\ell)$ with $x_0 \in \mathcal{V}$ and $x_\ell$ sampled from $X_\ell \mid x_0$, and set $\tau_\ell(i)^2 = \sum_{x_0, x_\ell \in \tilde{\mathcal{V}}_\ell} \|x_0(i) - h^i_\theta(x_\ell, \ell)\|^2 / \mathrm{card}(\tilde{\mathcal{V}}_\ell)$. Intuitively, this automatically adjusts how much we rely on the prior in the upcoming tasks by taking the reconstruction error as a proxy for the quality of the learned model.

## 3.2 Thompson Sampling with Diffusion Prior

Thompson sampling [13, 19] is one of the most popular strategies for tackling stochastic bandits due to its simplicity and generality. This strategy takes as input a prior distribution for the parameter of interest, and samples a guess of the parameter from the posterior distribution at each round to determine which arm to pull. In the MAB setting in Section 1, the parameter of interest is the vector of mean rewards $\mu$. At each round $t$, a vector $\tilde{\mu}_t$ is then sampled from the posterior distribution determined by the prior and the interaction history $(a_s, r_s)_{s \in \{1,\ldots,t-1\}}$. Subsequently, the learner pulls an arm $a_t \in \arg\max_{a \in \mathcal{A}} \tilde{\mu}^a_t$.

Our algorithm takes a prior learned by a diffusion model as input. We thus need to design an algorithm that samples from the posterior distribution when the prior is described in this way. While an exact solution does not exist in general, we aim to find a good approximation that balances well between the meta-learned prior and noisy observations. In contrast, most existing approaches for posterior sampling with diffusion models focus on the solution of inverse problems [4, 7, 16, 18], and do not take the underlying uncertainty into account.

To proceed, the general idea of our algorithm is to guide the sample towards the observation during the sampling process. Formally, let us consider $Y_0$ as a random observation of $X_0$ with known $q(Y_0 \mid X_0)$. For given $y_0$, we are interested in sampling from $X_0 \mid y_0$. We achieve this by going through the reverse Markovian process, conditioning on $Y_0 = y_0$. In fact, to sample from $X_\ell \mid y_0$, we only need to first sample $x_{\ell+1}$ from $X_{\ell+1} \mid y_0$ and then sample $x_\ell$ from $X_\ell \mid x_{\ell+1}, y_0$. Repeating the process for $\ell = L-1, \ldots, 0$ then gives the desired result. Below we briefly explain the initialization and the recursive steps of our method. Detailed derivation of the algorithm along with alternative approaches are discussed in Appendix A.

**Sampling from $X_L \mid y_0$.** For this part, we simply ignore $y_0$ and sample from $\mathcal{N}(0, 1)$ as before.

**Sampling from $X_\ell \mid x_{\ell+1}, y_0$** For simplicity, we restrict our attention to the case of multi-armed bandit. Here, $y_0$ is the interaction history and $x_0 = \mu$ is the vector of mean rewards, leading to

$$q(y_0 \mid x_0) \propto \prod_{s=1}^{t} q(r_s \mid \mu, a_s) = \prod_{s=1}^{t} \mathcal{N}(r_s; \mu^{a_s}, \sigma^2) \quad \text{[as a function of } x_0\text{]}. \tag{3}$$

We distinguish between two situations.

- Arm $a$ has never been pulled in the first $t$ rounds: We sample the corresponding coordinate $x_\ell^a$ from the Gaussian distribution $\tilde{q}(x_\ell^a \mid x_{\ell+1}, y_0) = p_\theta(x_\ell \mid x_{\ell+1})$ introduced in (2).
- Arm $a$ has been pulled in the first $t$ rounds: We denote by $\hat{\mu}_t^a = \sum_{s=1}^t r_s \mathbb{1}\{a_s = a\}/N_t^a$ as the empirical mean and $\sigma_t^a = \sigma/\sqrt{N_t^a}$ as the adjusted standard deviation, where $N_t^a$ is the number of times that arm $a$ has been pulled up to time $t$ (included). We also write $\bar{z}_{\ell+1}$ for the noise predicted by the denoiser from $x_{\ell+1}$.[2] Then, with $\rho_\ell = \bar{\alpha}_{\ell+1}(1-\bar{\alpha}_\ell)/(\bar{\alpha}_\ell(1-\bar{\alpha}_{\ell+1}))$, we sample from the Gaussian distribution $\tilde{q}(x_\ell^a \mid x_{\ell+1}, y_0)$ satisfying that
$$\tilde{q}(x_\ell^a \mid x_{\ell+1}, y_0) \propto p_\theta(x_\ell^a \mid x_{\ell+1}) \mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau_{\ell+1}^a)^2)\right).$$
In words, we sample $x_\ell$ from a Gaussian whose mean is a weighted average of the mean of $p_\theta(x_\ell^a \mid x_{\ell+1})$ and a noisy version of the empirical mean $\hat{\mu}_t$ (only defined for those arms that have been pulled). Importantly, the noisy version of the observation is computed with the predicted noise, which distinguishes our work from existing ones such as [18].

# 4 Numerical Experiments

In this section, we illustrate the benefit of using diffusion prior through two synthetic experiments. Missing details can be found in Appendix B.

**Problem construction.** For the two problems, we fix $\sigma = 0.1$ and construct the means as follows

1. `Labeled Arms` Problem. Let $K = 500$. In this problem, each arm is associated to 7 of the 50 labels. For each bandit task, we randomly pick 7 labels; the expected reward of an arm is positively correlated to the number of labels that fall into the intersection of the arm's labels and the task's labels. This is a simplest model for modeling user preferences over a list of items.
2. `Popular and Niche` Problem. Let $K = 200$. In this problem, the arms are separated into 20 groups, each of size 10. Arms of the same groups have close expected rewards. Moreover, 10 groups of arms tend to have high expected rewards but these arms are never the optimal ones. The other 10 groups of arms have lower expected rewards in general but contain a group of arms that yield much higher rewards. This represents a situation where Gaussian prior can lead to poor performance due to its inability to model multi-modal distributions.

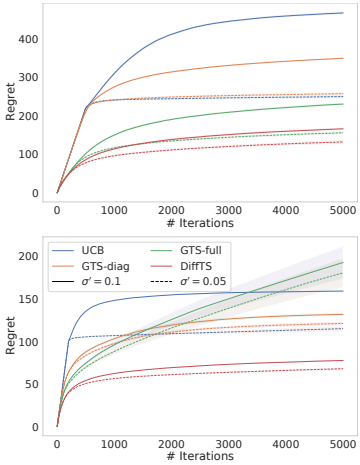



**Fig. 1:** Average regret over 100 tasks of different algorithms in the two problems. Shaded areas represent standard errors.

**Training and baselines.** In terms of algorithms, we compare our method, DiffTS, with UCB and Thompson sampling with Gaussian prior using either diagonal or full covariance matrix (GTS-diag and GTS-full). For each algorithm, we test two configurations: either the algorithm is given the true standard deviation $\sigma' = 0.1$ or an underspecified standard deviation $\sigma' = 0.05$.

To train the diffusion model, we use a training set of 5000 samples and a calibration set of 1000 samples. The 6000 samples put together are also used to compute the covariance matrices used by the Gaussian Thompson sampling algorithms. To test the performance of the algorithms, we then sample another 100 bandit tasks and run the aforementioned algorithms on these tasks.

**Results.** We report the regrets of the algorithms averaged over the 100 test tasks in Figure 1 (the top and the bottom figures correspond respectively to the `Labeled Arms` and the `Popular and Niche` problem). We verify that using a diffusion prior which describes better the task distribution indeed helps achieve smaller regret. However, the amount of improvement varies. In the `Labeled Arms` problem, learning the correlation between arms already lower the regret significantly, and the assumed noise standard deviation $\sigma'$ seems to have a greater impact. In the `Popular and Niche` problem, using a Gaussian prior with full covariance matrix however leads to poor performance, and the ability of denoising diffusion models to model complex distribution helps greatly here.

[2] It holds that $x_{\ell+1} = \sqrt{\bar{\alpha}_{\ell+1}} h_\theta(x_{\ell+1}, \ell+1) + \sqrt{1-\alpha_{\ell+1}}\bar{z}_{\ell+1}$.

# References


[1] Soumya Basu, Branislav Kveton, Manzil Zaheer, and Csaba Szepesvári. No regrets for learning the prior in bandits. *Advances in Neural Information Processing Systems*, 34:28029–28041, 2021.

[2] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[3] Leonardo Cella, Alessandro Lazaric, and Massimiliano Pontil. Meta-learning with stochastic linear bandits. In *International Conference on Machine Learning*, pages 1360–1370. PMLR, 2020.

[4] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *arXiv preprint arXiv:2206.09012*, 2022.

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[6] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44 (9):5149–5169, 2021.

[7] Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769, 2021.

[8] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2020.

[9] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[10] Amit Peleg, Naama Pearl, and Ron Meir. Metalearning linear bandits by prior update. In *International Conference on Artificial Intelligence and Statistics*, pages 2885–2926. PMLR, 2022.

[11] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pages 8857–8868. PMLR, 2021.

[12] Jonas Rothfuss, Dominique Heyn, Andreas Krause, et al. Meta-learning reliable priors in the function space. *Advances in Neural Information Processing Systems*, 34:280–293, 2021.

[13] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[14] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[15] Max Simchowitz, Christopher Tosh, Akshay Krishnamurthy, Daniel J Hsu, Thodoris Lykouris, Miro Dudik, and Robert E Schapire. Bayesian decision-making under misspecified priors with applications to meta-learning. *Advances in Neural Information Processing Systems*, 34: 26382–26394, 2021.

[16] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[17] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2021.

[19] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[20] Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Current opinion in chemical biology*, 65:18–27, 2021.

# Appendix

## A Posterior Sampling: Algorithms and Derivation

Below we provide two approximations to sample from $X_L \mid x_{\ell+1}, y_0$. The first one is easier to derive but the second one consistently achieves better performance in our experiments.[3] We thus utilize the second approach in Sections 3 and 4.

1. **Approach 1, acting on $x_0$.** We write

$$q(x_\ell \mid x_{\ell+1}, y_0) = \frac{\int q(x_\ell, y_0, x_0 \mid x_{\ell+1})\, dx_0}{q(y_0 \mid x_{\ell+1})} = \frac{\int q(y_0 \mid x_0) q(x_\ell \mid x_0, x_{\ell+1}) q(x_0 \mid x_{\ell+1})\, dx_0}{q(y_0 \mid x_{\ell+1})}. \tag{4}$$

   As in Section 3.1, we approximate $q(x_0 \mid x_{\ell+1})$ by

$$p'_\theta(x_0 \mid x_{\ell+1}) = \mathcal{N}(x_0; h_\theta(x_{\ell+1}, \ell+1), \mathrm{diag}(\tau^2_{\ell+1})).$$

   Then, if $q(Y_0 \mid x_0)$ is also Gaussian, the integral in (4) with $q(x_0 \mid x_{\ell+1})$ replaced by $p'_\theta(x_0 \mid x_{\ell+1})$ can be computed in close form.

   For further illustration, we focus on the case of multi-armed bandits where the relation between $y_0$ the interaction history and $x_0 = \mu$ the mean reward vector is given by (3). The approximate distribution of $X_\ell \mid x_{\ell+1}, y_0$ is thus independent across coordinates (arms) and we can compute the density individually for each arm, which we denote by $\tilde{q}(x^a_\ell \mid x_{\ell+1}, y_0)$. It is computed using (4) with $q(x_0 \mid x_{\ell+1}) \approx p'_\theta(x_0 \mid x_{\ell+1})$ followed by a normalization.

   When arm $a$ has never been pulled, $\tilde{q}(x^a_\ell \mid x_{\ell+1}, y_0)$ is nothing but $p_\theta(x^a_\ell \mid x_{\ell+1})$. Otherwise, $\tilde{q}(x^a_\ell \mid x_{\ell+1}, y_0)$ is computed first using the fact that

$$q(y^a_0 \mid x^a_0) p'_\theta(x^a_0 \mid x_{\ell+1}) \propto \mathcal{N}\left( x^a_0; \underbrace{\frac{\hat{\mu}^a_t/(\sigma^a_t)^2 + h^a_\theta(x_{\ell+1}, \ell+1)/(\tau^a_{\ell+1})^2}{1/(\sigma^a_t)^2 + 1/(\tau^a_{\ell+1})^2}}_{\hat{x}^a_0}, \underbrace{\frac{1}{1/(\sigma^a_t)^2 + 1/(\tau^a_{\ell+1})^2}}_{(\hat{\sigma}^a)^2} \right).$$

   Next, with

$$q(x^a_\ell \mid x_0, x_{\ell+1}) = \mathcal{N}\left( x^a_\ell; \underbrace{\frac{\sqrt{\bar{\alpha}_\ell}\beta_{\ell+1}}{1-\bar{\alpha}_{\ell+1}}}_{\pi_1} x^a_0 + \underbrace{\frac{\sqrt{\alpha_{\ell+1}}(1-\bar{\alpha}_\ell)}{1-\bar{\alpha}_{\ell+1}}}_{\pi_2} x^a_{\ell+1}, \frac{(1-\bar{\alpha}_\ell)\beta_{\ell+1}}{1-\bar{\alpha}_{\ell+1}} \right),$$

   we see that we just need to replace $x^a_0$ by $\hat{x}^a_0$ and augment the variance by $\pi_1^2(\hat{\sigma}^a)^2$ when sampling $x^a_\ell$. Compared to the case where $y_0$ is not given, the main difference lies in that we first change the estimated $\hat{x}_0$ before sampling $x_\ell$.

2. **Approach 2, acting on $x_\ell$.** Alternatively, we may write

$$q(x_\ell \mid x_{\ell+1}, y_0) = \frac{q(x_\ell \mid x_{\ell+1}) q(y_0 \mid x_\ell, x_{\ell+1})}{q(y_0 \mid x_{\ell+1})} = \frac{q(x_\ell \mid x_{\ell+1}) \int q(y_0 \mid x_0) q(x_0 \mid x_\ell, x_{\ell+1})\, dx_0}{q(y_0 \mid x_{\ell+1})}. \tag{5}$$

   To begin, we use $p_\theta(x_\ell \mid x_{\ell+1})$ to approximate $q(x_\ell \mid x_{\ell+1})$. Next, one natural way to tackle the integral is to use $q(x_0 \mid x_\ell, x_{\ell+1}) = q(x_0 \mid x_\ell) \approx p'_\theta(x_0 \mid x_\ell)$. In the simplest case $q(y_0 \mid x_0) = \mathcal{N}(y_0; x_0, \sigma^2 I)$, and we deduce

$$\int q(y_0 \mid x_0) p'_\theta(x_0 \mid x_\ell)\, dx_0 = \mathcal{N}(y_0; h_\theta(x_\ell, \ell), \sigma^2 I + \mathrm{diag}(\tau^2_\ell)).$$

[3] We conjecture this is because the first approach leads to results that are less consistent with the observations.

Nonetheless, as the denoiser $h_\theta$ can be arbitrarily complex, this does not lead to a close form expression to sample $x_\ell$. We may resort to Langevin dynamics sampling, i.e., taking gradient steps on $x_\ell$ to make sure the denoised result is coherent with $y_0$. This is a general approach that may be of interest for arbitrary conditional distribution $Y_0 \mid X_0$, but we have simpler solution when the noise is Gaussian. In fact, it holds that

$$X_\ell = \sqrt{\bar{\alpha}_\ell} X_0 + \sqrt{1-\bar{\alpha}_\ell}\bar{Z}_\ell \quad \text{and} \quad X_{\ell+1} = \sqrt{\alpha_{\ell+1}} X_\ell + \sqrt{1-\alpha_\ell} Z_{\ell+1},$$

where both $\bar{Z}_\ell$ and $Z_{\ell+1}$ are random variable with distribution $\mathcal{N}(0, I)$. This results in

$$X_{\ell+1} = \sqrt{\bar{\alpha}_{\ell+1}} X_0 + \sqrt{1-\bar{\alpha}_{\ell+1}}\bar{Z}_{\ell+1}$$

for

$$\bar{Z}_{\ell+1} = \sqrt{\frac{\alpha_{\ell+1}(1-\bar{\alpha}_\ell)}{1-\bar{\alpha}_{\ell+1}}}\bar{Z}_\ell + \sqrt{\frac{1-\alpha_{\ell+1}}{1-\bar{\alpha}_{\ell+1}}} Z_{\ell+1}.$$

Therefore, we may take $\bar{Z}_{\ell+1}$ as a reasonable approximation of $\bar{Z}_\ell$, while sampling $\bar{Z}_{\ell+1}$ is basically the same as sampling from $p'_\theta(X_0 \mid x_{\ell+1})$. To summarize, we write

$$\begin{aligned}
q(x_0 \mid x_\ell, x_{\ell+1}) &= q\left(\bar{Z}_\ell = \frac{x_\ell - \sqrt{\bar{\alpha}_\ell} x_0}{\sqrt{1-\bar{\alpha}_\ell}} \,\middle|\, x_\ell, x_{\ell+1}\right) \\
&\approx q\left(\bar{Z}_{\ell+1} = \frac{x_\ell - \sqrt{\bar{\alpha}_\ell} x_0}{\sqrt{1-\bar{\alpha}_\ell}} \,\middle|\, x_\ell, x_{\ell+1}\right) \\
&= q\left(X_0 = \frac{1}{\sqrt{\bar{\alpha}_{\ell+1}}}\left(x_{\ell+1} - \left(x_\ell - \sqrt{\bar{\alpha}_\ell} x_0\right)\sqrt{\frac{1-\bar{\alpha}_{\ell+1}}{1-\bar{\alpha}_\ell}}\right) \,\middle|\, x_\ell, x_{\ell+1}\right) \\
&\approx p'_\theta\left(X_0 = \frac{1}{\sqrt{\bar{\alpha}_{\ell+1}}}\left(x_{\ell+1} - \left(x_\ell - \sqrt{\bar{\alpha}_\ell} x_0\right)\sqrt{\frac{1-\bar{\alpha}_{\ell+1}}{1-\bar{\alpha}_\ell}}\right) \,\middle|\, x_{\ell+1}\right) \\
&= \mathcal{N}\Bigg(\sqrt{\frac{\bar{\alpha}_\ell(1-\bar{\alpha}_{\ell+1})}{\bar{\alpha}_{\ell+1}(1-\bar{\alpha}_\ell)}} x_0 + \frac{x_{\ell+1}}{\sqrt{\bar{\alpha}_{\ell+1}}} - \sqrt{\frac{(1-\bar{\alpha}_{\ell+1})}{\bar{\alpha}_{\ell+1}(1-\bar{\alpha}_\ell)}} x_\ell \,; \\
&\qquad h_\theta(x_{\ell+1}, \ell+1), \operatorname{diag}(\tau^2_{\ell+1})\Bigg) \\
&= \sqrt{\rho_\ell}\,\mathcal{N}\left(x_0 \,;\, \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell - \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}), \rho_\ell \operatorname{diag}(\tau^2_{\ell+1})\right),
\end{aligned}$$

where $\rho_\ell = \bar{\alpha}_{\ell+1}(1-\bar{\alpha}_\ell)/(\bar{\alpha}_\ell(1-\bar{\alpha}_{\ell+1}))$ and $\bar{z}_{\ell+1}$ represents the noise predicted by the denoiser from $x_{\ell+1}$, that is,

$$\bar{z}_{\ell+1} = \frac{x_{\ell+1} - \sqrt{\bar{\alpha}_{\ell+1}} h_\theta(x_{\ell+1}, \ell+1)}{\sqrt{1-\bar{\alpha}_{\ell+1}}}.$$

Then, focusing on the arm that has been pulled at least once and ignoring the multiplicative constant that does not depend on $x_\ell$, we get

$$\begin{aligned}
q(y_0^a \mid x_\ell) &= \int q(y_0^a \mid x_0^a) q(x_0^a \mid x_\ell, x_{\ell+1})\, dx_0 \\
&\approx \sqrt{\rho_\ell} \int q(y_0^a \mid x_0^a)\mathcal{N}\left(x_0^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}), \rho_\ell(\tau^a_{\ell+1})^2\right)\, dx_0 \\
&\propto \int \mathcal{N}\left(x_0^a; \hat{\mu}_t^a, (\sigma_t^a)^2\right)\mathcal{N}\left(x_0^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}), \rho_\ell(\tau^a_{\ell+1})^2\right)\, dx_0 \\
&= \mathcal{N}\left(\hat{\mu}_t^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}), (\sigma_t^a)^2 + \rho_\ell(\tau^a_{\ell+1})^2\right) \\
&\propto \mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau^a_{\ell+1})^2).\right)
\end{aligned}$$

Subsequently, we can approximate the posterior distribution of $x_\ell$ using (5)

$$\tilde{q}(x_\ell^a \mid x_{\ell+1}, y_0) \propto p_\theta(x_\ell^a \mid x_{\ell+1})\mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau^a_{\ell+1})^2)\right).$$

This results in the algorithm that we present in Section 3.2.

## B Missing Experimental Details

In this section, we provide missing experimental details mainly concerning the used diffusion models and the construction of the problem instances. All the simulations are run on an Amazon p3.2xlarge instance equipped with an NVIDIA Tesla V100 GPU.

**Diffusion models.** In our experiments, we set the diffusion steps of the diffusion models to $L = 100$ and adopt a linear variance schedule from $1 - \alpha_1 = 10^{-4}$ to $1 - \alpha_L = 0.1$. The denoiser itself is a 1-dimensional U-Net adapted from [8, 11]. We use 5 residual blocks, each block containing 6 residual channels. These numbers are rather arbitrary and do not seem to affect much our results. More importantly, since the patterns that we want to learn do not have the spatial correlation that convolutional layers are designed for, we add a fully connected layer at the beginning to map the input to a vector of size $128 \times 6$, before reshaping these vectors into 6 channels and feeding them to the convolutional layers. In a similar fashion, we also replace the last layer of the architecture by a fully connected layer. We find out that these minimal modification already enable the model to perform well on our problems, but believe a thorough investigation into the architecture design would further benefit our approach.

These diffusion models are then trained with Adam for 600 epochs, with a learning rate of $1 \times 10^{-3}$, $5 \times 10^{-4}$, and $2 \times 10^{-4}$, for the first, middle, and last 200 epochs. The batch size is fixed at 128. Again, we believe our synthetic problems are simple enough for diffusion models so that these hyperparameters should have minimal effect on the results.

**Construction of bandit instances.** We now give more details on how the mean reward vectors are constructed in the two problems. Some illustrations of the constructed instances and the vectors generated by the trained diffusion models are provided in Figures 2 and 3.

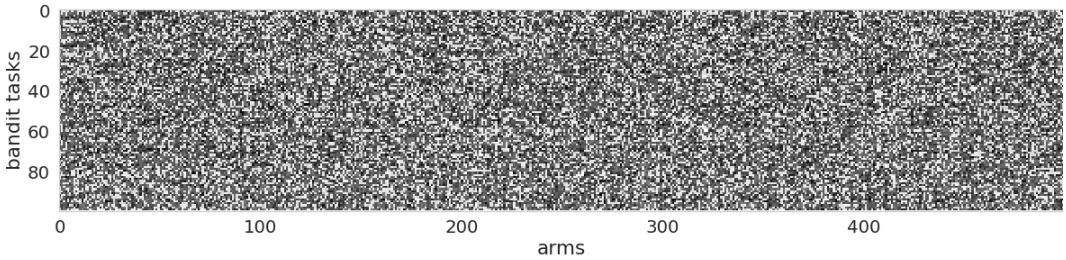
1. `Labeled Arms` ($K = 500$). As described in Section 4, in our first problem we are provided a set of 50 labels $\mathcal{L} = \{1, ..., 50\}$. Each arm is associated to a subset $\mathcal{L}^a$ of these labels with size $\mathrm{card}(\mathcal{L}^a) = 7$. To sample a new bandit task $B$, we randomly draw a set $\mathcal{L}_B \subseteq \mathcal{L}$ again with size 7. Then for each arm $a$, we set $\bar{\mu}^a = 1 - 1/4^{\mathrm{card}(\mathcal{L}^a \cap \mathcal{L}_B)}$. Finally, to obtain the mean rewards $\mu$, we perturb the coordinates of $\bar{\mu}$ by independent Gaussian noises of standard deviation 0.1 and scale the resulting vector to the range $[0, 1]$.
2. `Popular and Niche` ($K = 200$). The arms are split into 20 groups of equal size. Conceptually, 10 of these groups represent the 'popular' items while the other 10 represent the 'niche' items. For each bandit task, we first construct a vector $\bar{\mu}$ whose coordinates' values default to 0.5. However, we choose one group of niche items and set their values to 1. For each group of popular items, with probability 0.8 their values are set to 0.8. Then, to construct the mean reward vector $\mu$, we respectively perturb the values of the niche items and of the popular items by independent Gaussian noises with standard deviation of 0.1 and of 0.2. After that, we clip the values of the popular items to make them smaller than 0.95 and finally scale the vector to the range $[0, 1]$.

**UCB.** The most standard implementation of the UCB algorithm sets the upper confidence bound to

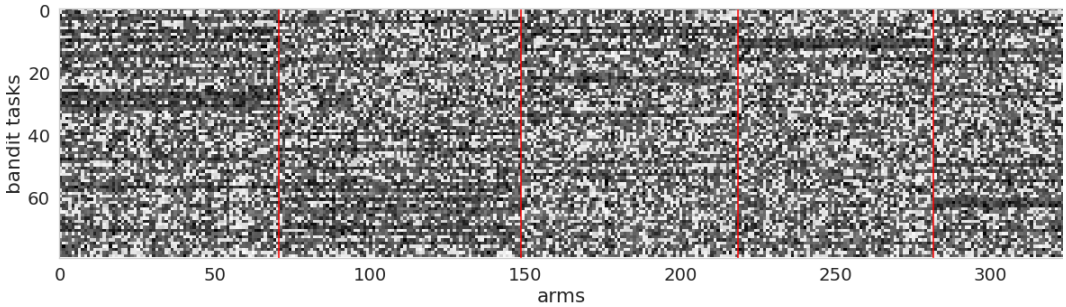
$$U_t^a = \hat{\mu}_t^a + \sigma' \sqrt{\frac{\log t}{N_t^a}}. \tag{6}$$

Instead, in our experiments we use $U_t^a = \hat{\mu}_t^a + \sigma'/\sqrt{N_t^a}$. Eq. (6) is more conservative than our implementation. However, looking at Figure 1, we figure out that UCB without the $\log t$ factor is already the most conservative algorithm in our experiments. Therefore, while in the long term UCB with (6) may achieve lower regret, with limited budget Thompson sampling with a wrong but good enough prior would generally be preferable [15].

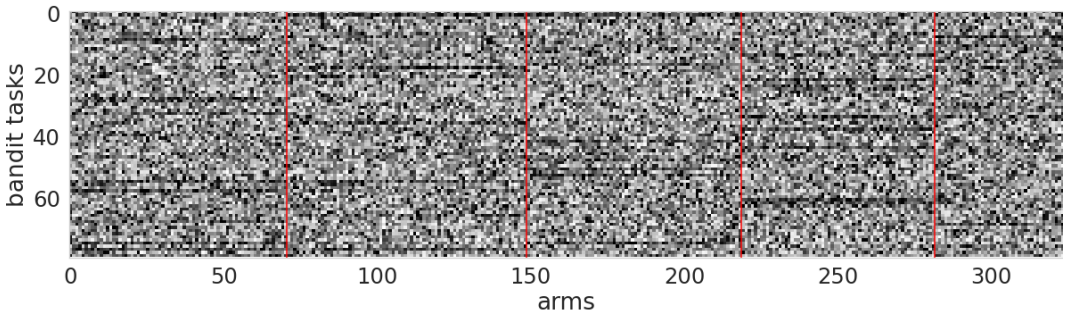
**Regret.** For sake of completeness, we state here the definition of the (pseudo-)regret with respect to a sequence of arms $(a_t)_{t \in \{1,...,T\}}$ in a single bandit task. Let $a^\star \in \arg\max_{a \in \mathcal{A}} \mu^a$ be an optimal arm. It is defined as $\mathrm{Reg}_T = T\mu^{a^\star} - \sum_{t=1}^T \mu^{a_t}$.




**(a)** The mean reward vectors of 100 constructed bandit tasks.

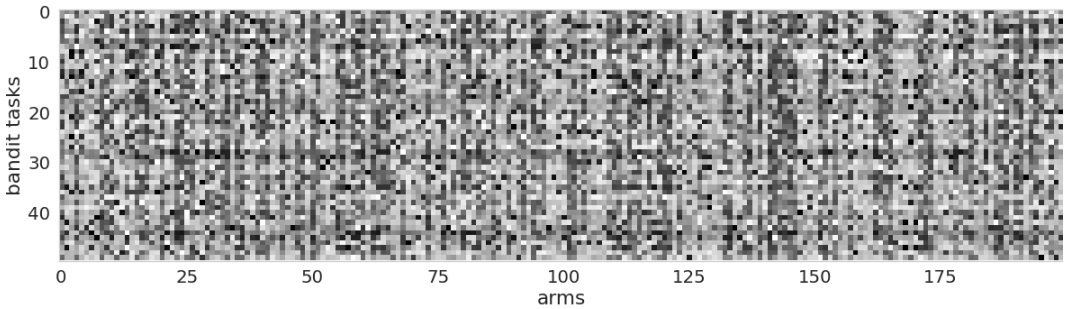



**(b)** The mean reward vectors of 80 constructed bandit tasks, grouped by labels and showing only 5 labels. Note that each arm has multiple labels and thus appears in multiple groups.

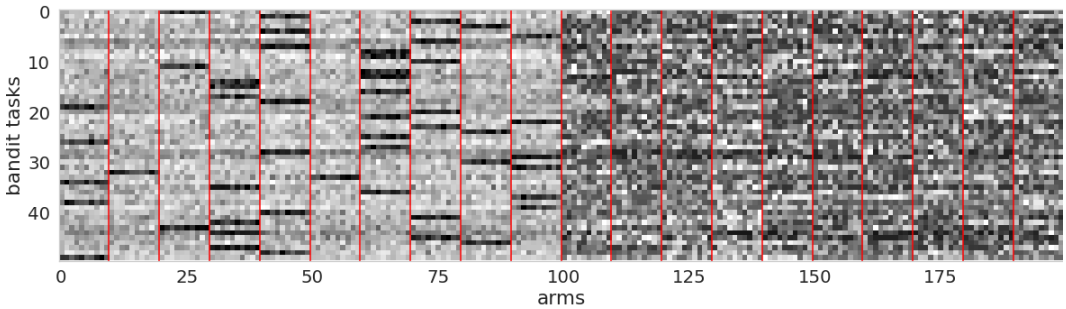



**(c)** 80 mean reward vectors generated by the trained diffusion model, grouped by labels and showing only 5 labels. Note that each arm has multiple labels and thus appears in multiple groups.

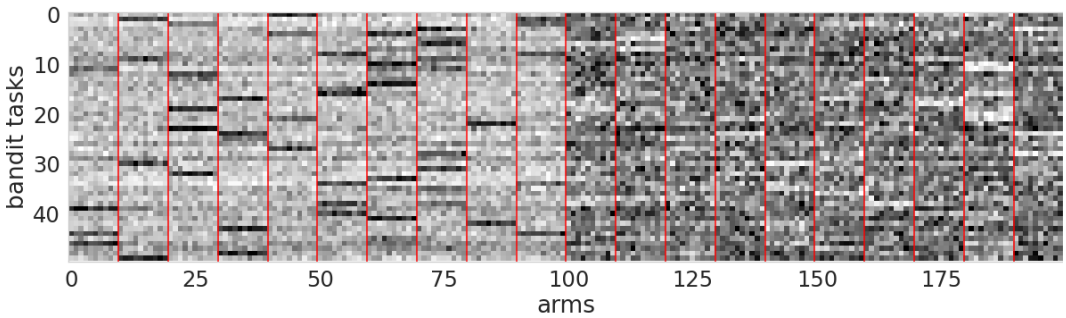
**Fig. 2:** Visualization of the mean reward vectors constructed in the `Labeled Arms` problem. Rows and columns correspond to tasks and arms. The darker the color the higher the value, with white and black representing respectively $\mu^a = 0$ and $\mu^a = 1$. While human eyes can barely recognize any pattern in the constructed vectors. Diffusion models manage to learn the underlying patterns that become recognizable by humans only when the arms are grouped in a specific way.




**(a)** The mean reward vectors of 50 constructed bandit tasks.




**(b)** The mean reward vectors of 50 constructed bandit tasks. Reordered to put the arms of the same group together. The popular arms are on the right side of the figure.




**(c)** 50 mean reward vectors generated by the trained diffusion model. Reordered to put the arms of the same group together. The popular arms are on the right side of the figure.

**Fig. 3:** Visualization of the mean reward vectors constructed in the `Popular and Niche` problem. Rows and columns correspond to tasks and arms. The darker the color the higher the value, with white and black representing respectively $\mu^a = 0$ and $\mu^a = 1$. While human eyes can barely recognize any pattern in the constructed vectors. Diffusion models manage to learn the underlying patterns that become recognizable by humans only when the arms are grouped in a specific way.