Thompson Sampling with Diffusion Generative Prior

Yu-Guan Hsieh[*1] **Shiva Prasad Kasiviswanathan**[2] **Branislav Kveton**[3] **Patrick Blöbaum**[2]

Abstract

In this work, we initiate the idea of using denoising diffusion models to learn priors for online decision making problems. We specifically focus on bandit meta-learning, aiming to learn a policy 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 tasks at test time. Our posterior sampling algorithm carefully balances between the learned prior and the noisy observations that come from the learner's interaction with the environment. To capture realistic bandit scenarios, we propose a novel diffusion model training procedure that trains from incomplete and noisy data, which could be of independent interest. Finally, our extensive experiments clearly demonstrate the potential of the proposed 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 for exploitation (Puterman, 1994; Sutton and Barto, 1998). Among these, Bayesian methods quantify uncertainty using probability distributions, with the help of the powerful tools of Bayesian inference. Nonetheless, their performance is known to be sensitive to the choice of prior (Murphy, 2022).

For concreteness, let us consider the problem of stochastic multi-armed bandits (MABs) (Bubeck and Cesa-Bianchi, 2012; Lattimore and Szepesvári, 2020), 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_t is pulled at round t, the learner receives reward $r_t \in \mathbb{R}$ drawn from an arm-dependent distribution $\mathcal{P}^{a_t}$. The goal of the learner is either to *i*) accumulate the highest possible reward over time (a.k.a. regret-minimization, see Lai and Robbins, 1985; Auer et al., 2002) or to *ii*) find the arm with the highest expected reward (a.k.a. best-arm identification, see Even-Dar et al., 2006; Bubeck et al., 2009; Audibert et al., 2010).

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 pulling each arm a certain number of times, which becomes inefficient when K is large. While the no-free-lunch principle prevents us from improving upon this bottleneck in general situations, it is worth noticing that the bandit instances (referred to as tasks hereinafter) that we encounter in most practical problems are far from arbitrary. To name a few examples, in recommender 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 with 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 meta-learning paradigm, the prior itself can also be learned from data, which often leads to superior performance (Rothfuss et al., 2021; Hospedales et al., 2021). This has led to the idea of meta-learning a prior for bandits (Cella et al., 2020; Basu et al., 2021; Bastani et al., 2022).

On the other hand, we have recently witnessed the success of deep generative modeling in producing high-quality synthetic data across various modalities (Saharia et al., 2022; Wu et al., 2021; Brown et al., 2020). The impressive results of these methods show that they are a powerful tool for modeling complex distributions. While different models have their own strengths and weaknesses, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are particularly appealing for our use case because their iterative sampling scheme makes them more flexible to be applied on a downstream task. In this regard, this paper attempts to answer the following question:

Can diffusion models provide better priors to address the exploration-exploitation trade-off in bandits?

Our Contributions. In this work, we initiate the idea of

[*]Work done during internship at Amazon. [1]Université Grenoble Alpes [2]Amazon [3]AWS AI Labs. Correspondence to: Yu-Guan Hsieh <yu-guan.hsieh@univ-grenoble-alpes.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

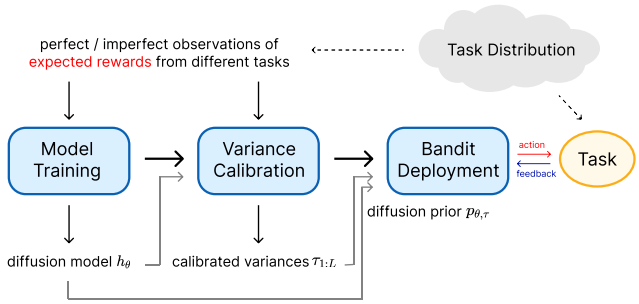


Figure 1. Overview of the meta-learning for bandits with diffusion prior framework.

using diffusion models to meta-learn a prior for bandit problems (cf. Figure 1). Our focus is on designing algorithms that have good empirical performance while being mathematically meaningful. Working towards this direction, we make the following contributions:

(a) We propose a new Thompson sampling scheme that incorporates a prior represented by a diffusion model. 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. In particular, the deployment of the diffusion model begins with a variance calibration step. Then, in each round of the interaction, we summarize the interaction history by a masked vector of dimension K, and perform posterior sampling with a modified iterative sampling process that makes use of this vector.

(b) Standard diffusion model training uses noise-free samples. Such data are however nearly impossible to obtain in most bandit applications. To overcome this limitation, we propose a novel diffusion model training procedure for incomplete and noisy data. Our method alternates between sampling from the posterior distribution and minimizing a tailored loss function. We believe that it could be of interest beyond its use in bandit problems.

(c) We experiment with various synthetic and real datasets to show benefits of our approach against several baselines, including Thompson sampling with Gaussian prior (Thompson, 1933), Thompson sampling with Gaussian mixture model (GMM) prior (Hong et al., 2022b), and UCB1 (Auer, 2002). We observe that the diffusion prior consistently improves performance. The improvement is especially significant when the underlying problem is complex.

Related Work. Prior to our work, the use of diffusion models in decision making has been explored by Janner et al. (2022); Ajay et al. (2023), who used conditional diffusion models to synthesize trajectories in offline decision making. Their approaches demonstrated good performance on various benchmarks. In contrast, our focus is on online decision making, where exploration is crucial to succeed. Additionally, we use diffusion models to learn a task prior, rather than a distribution specific to a single task.

More generally, diffusion models have been used as priors in various areas, primarily for inverse problem solving. At a high level, one common approach for diffusion model posterior sampling is to combine each unconditional sampling step with a step that ensures coherence with the observation. This approach was taken by Sohl-Dickstein et al. (2015); Song et al. (2022); Chung et al. (2023) and our posterior sampling algorithm can also be viewed in this way. Interested readers are referred to Appendix B for a detailed explanation of how our posterior sampling method differs from existing ones. In the same appendix, we also provide a brief overview of related multi-armed bandit works that discussed the influence of prior.

Notation. All the variables are multi-dimensional unless otherwise specified. For a vector x, x^a represents its a-th coordinate, x^2 represents its coordinate-wise square, and $\mathrm{diag}(x)$ represents the diagonal matrix with the elements of x on the diagonal. A sequence of vectors $(x_l)_{l\in\{l_1,\dots,l_2\}}$ is written as $x_{l_1:l_2}$. We use the symbol $\odot$ for element-wise multiplication between two vectors. To distinguish random variables from their realization, we represent the former with capital letters and the latter with the corresponding lowercase letters. Conditioning on $X = x$ is then abbreviated as $\cdot \mid x$. A Gaussian distribution centered at $\mu \in \mathbb{R}^d$ with covariance $\Sigma \in \mathbb{R}^{d\times d}$ is written as $\mathcal{N}(X; \mu, \Sigma)$, or simply $\mathcal{N}(\mu, \Sigma)$ if the random variable in question is clear from the context. $\mathbf{0}_d \in \mathbb{R}^d$ and $I_d \in \mathbb{R}^{d\times d}$ are respectively the zero vector of dimension d and the identity matrix of size $d \times d$. Finally, $[n]$ denotes the sequence of integers $\{1, ..., n\}$

2. Preliminaries and Problem Description

In this section, we briefly review denoising diffusion models and introduce our meta-learning for bandits framework.

2.1. Denoising Diffusion Probabilistic Model

First introduced by Sohl-Dickstein et al. (2015) and recently popularized by Ho et al. (2020) and Song and Ermon (2019), denoising diffusion models (or the closely related score-based models) have demonstrated state-of-the-art performance in various data generation tasks. A large number of variants of these models have been proposed since then. In this paper, we primarily follow the notation and formulation of Ho et al. (2020) with minor modifications.

Intuitively speaking, diffusion models learn to approximate a distribution $\mathcal{Q}_0$ over $\mathbb{R}^d$ 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

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

The first equality suggests that the forward process forms a Markov chain that starts at $x_0 \in \mathbb{R}^d$. The second one says that the transition kernel is Gaussian. Denoting the product of the scale factors by $\bar{\alpha}_\ell = \prod_{i=1}^{\ell} \alpha_i$, we get $q(X_\ell \,|\, x_0) = \mathcal{N}(X_\ell; \sqrt{\bar{\alpha}_\ell} x_0, (1-\bar{\alpha}_\ell) I_d)$.

A denoising diffusion model is an approximation of the reverse process

$$q(x_{0:L-1} \,|\, x_L) = \prod_{\ell=0}^{L-1} q(x_\ell \,|\, x_{\ell+1}).$$

The conditional distribution $q(X_\ell \,|\, x_{\ell+1})$ is in general not a Gaussian. However, if we additionally condition on x_0, we get a Gaussian distribution $q(X_\ell \,|\, x_{\ell+1}, x_0)$, which is easy to sample from (see Appendix C.1). Of course, since the goal of the reverse process is to sample x_0, it is unknown. Therefore, we approximate it as the output of a denoiser h_θ parameterized by θ. In particular, the denoised sample in step ℓ, $\hat{x}_0 = h_\theta(x_{\ell+1}, \ell+1)$, is estimated from an earlier sample $x_{\ell+1}$ by a function indexed by $\ell+1$.[1] Note that $\hat{x}_0$ depends on ℓ but we omit it in the notation to simplify it. The distribution of the reverse step $p_\theta(X_\ell \,|\, x_{\ell+1})$ is then modeled as

$$\begin{aligned} &p_\theta(X_\ell \,|\, x_{\ell+1}) = q(X_\ell \,|\, x_{\ell+1}, X_0 = \hat{x}_0) \\ &= \mathcal{N}\left(X_\ell; w_1 \hat{x}_0 + w_2 x_{\ell+1}, \frac{1-\bar{\alpha}_\ell}{1-\bar{\alpha}_{\ell+1}}(1-\alpha_{\ell+1}) I_d\right), \end{aligned} \tag{1}$$

where

$$w_1 = \frac{\sqrt{\bar{\alpha}_\ell}(1-\alpha_{\ell+1})}{1-\bar{\alpha}_{\ell+1}} \quad \text{and} \quad w_2 = \frac{\sqrt{\alpha_{\ell+1}}(1-\bar{\alpha}_\ell)}{1-\bar{\alpha}_{\ell+1}}.$$

The approximation $\hat{x}_0$ is refined through the reverse process and we end up with the sampled x_0 at $\ell = 0$. It is common to choose decreasing $(\alpha_\ell) \in (0,1)^L$ such that $\bar{\alpha}_L \approx 0$. In this case, $q(X_L) \approx \mathcal{N}(\mathbf{0}_d, I_d)$ and thus x_L can be initially sampled from $p_\theta(X_L) = \mathcal{N}(\mathbf{0}_d, I_d)$.

Finally, the denoiser h_θ is learned by optimizing a variational lower bound, which amounts to minimizing the following L_2 reconstruction loss,

$$\sum_{\ell=1}^{L} \mathbb{E}_{x_0 \sim \mathcal{Q}_0, x_\ell \sim X_\ell \,|\, x_0}[\|x_0 - h_\theta(x_\ell, \ell)\|^2]. \tag{2}$$

2.2. Meta-Learning of Bandit Tasks

Our work focuses on meta-learning problems in which the tasks are bandit instances drawn from an underlying distribution that we denote by $\mathcal{T}$. As in standard meta-learning, the goal is to learn an inductive bias from the meta training set that would improve the overall performance of an algorithm on new tasks drawn from the same distribution. In our work, the inductive bias is encoded by a prior distribution used by Thompson sampling when the learner interacts with new bandit instances.

For simplicity, we focus on multi-armed bandits presented in Section 1, with the additional assumption that the reward noise is Gaussian with known variance $\sigma^2_{\text{reward}} \in \mathbb{R}$.[2] The only unknown information is thus the mean reward vector $\mu = (\mu^a)_{a \in \mathcal{A}}$. In this case, Thompson sampling takes as input a prior distribution over $\mathbb{R}^K$, samples a guess $\tilde{\mu}_t$ of the mean reward vector from the posterior distribution at each round t, and pulls arm $a_t \in \arg\max_{a \in \mathcal{A}} \tilde{\mu}_t^a$. The posterior distribution is determined by both the prior and the interaction history, i.e., the sequence of the action-reward pairs $\mathcal{H}_{t-1} = (a_s, r_s)_{s \in [t-1]}$.

As for the meta-training phase, we consider two situations that are distinguished by whether the learner has access to *perfect* data or not. In the former case, the meta-training set is composed of the exact means $\mathcal{D}_{\text{tr}} = \{\mu_B\}_B$ of training tasks B drawn from the distribution $\mathcal{T}$. In the latter case, the training set is composed of incomplete and noisy observations of these vectors (see Section 4 for details). We use the term *imperfect* data to informally refer to the scenario where the data is incomplete and noisy. The entire algorithm flow is summarized in Figure 1 and Algorithm 5 (Appendix A). The model training and the variance calibration blocks together define the diffusion prior, which is then used by Thompson sampling in the deployment phase, as we show next in Section 3.

3. Diffusion Models in Thompson Sampling

In this section, we describe how a learned diffusion model can be incorporated as a prior in Thompson sampling. To begin, we first revisit the probability distribution defined by

[1] To obtain h_θ, we typically train a neural network with a U-Net architecture. In Ho et al. (2020), this network is trained to output the predicted noise $\bar{z}_\ell = (x_\ell - \sqrt{\bar{\alpha}_\ell} h_\theta(x_\ell, \ell))/\sqrt{1-\bar{\alpha}_\ell}$.

[2] We make this assumption as we use a diffusion prior. To our best knowledge, all existing diffusion model posterior sampling algorithms for Gaussian noise either rely on this assumption or circumvent it by adding some adjustable hyperparameter. How to extend these algorithms to cope with unknown noise variance properly is an interesting open question.

the diffusion model by introducing an additional variance calibration step. After that, we present a diffusion posterior sampling algorithm for noisy and partially observed sample vectors that uses this prior. Finally, we present Thompson sampling with a diffusion prior.

3.1. Variance Calibration

While Ho et al. (2020) fixed the variance of $p_\theta(X_\ell \mid x_{\ell+1})$ to that of $q(X_\ell \mid x_{\ell+1}, x_0)$ in (1), Bao et al. (2021) recently showed that this choice was sub-optimal. This is critical when we use diffusion model as a prior in online decision making, as it prevents us from quantifying the right level of uncertainty. To remedy this, we follow Bao et al. (2022) and calibrate the variances of the reverse process with a calibration set $\mathcal{D}_{\text{cal}} = \{x_{i,0}\}_{i\in[n_{\text{cal}}]}$. Here $x_{i,0} \in \mathbb{R}^K$ is a K-dimensional vector, n_{cal} is the number of calibration samples, and we use subscript i to denote data point i.

Our calibration step starts by quantifying the uncertainty of the denoiser output. Concretely, we model the distribution of $X_0 \mid x_\ell$ by a Gaussian distribution centered at the denoised sample $h_\theta(x_\ell, \ell)$. As for the covariance of the distribution, we take it as the diagonal matrix $\operatorname{diag}(\tau_\ell^2)$ with entries

$$\tau_\ell^a = \sqrt{\frac{1}{n_{\text{cal}}} \sum_{i=1}^{n_{\text{cal}}} \|x_{i,0}^a - h_\theta^a(x_{i,\ell}, \ell)\|^2}. \tag{3}$$

Here h_θ^a denotes the a-th entry of the output of h_θ. In words, for each sample $x_{i,0}$, we first diffuse it through the forward process to obtain $x_{i,\ell}$. Then we compute the coordinate-wise mean squared error between the predicted $\hat{x}_0$ and the actual x_0. Pseudo-code of the above procedure is provided in Algorithm 1.

Having introduced the above elements, we next define the calibrated reverse step as

$$\begin{aligned} p_{\theta,\tau}(X_\ell \mid x_{\ell+1}) &= \int q(X_\ell \mid x_{\ell+1}, x_0) p'_{\theta,\tau}(x_0 \mid x_{\ell+1})\, dx_0 \\ &= \mathcal{N}\Big(X_\ell; w_1 \hat{x}_0 + w_2 x_{\ell+1}, \\ &\qquad \frac{1-\bar{\alpha}_\ell}{1-\bar{\alpha}_{\ell+1}}(1-\alpha_{\ell+1}) I_d + w_1^2 \operatorname{diag}(\tau_{\ell+1}^2)\Big), \end{aligned} \tag{4}$$

where $\tau = \tau_{1:L}$ represents the estimated variance parameter and $p'_{\theta,\tau}(X_0 \mid x_{\ell+1}) = \mathcal{N}(h_\theta(x_{\ell+1}, \ell+1), \operatorname{diag}(\tau_{\ell+1}^2))$ is the aforementioned Gaussian approximation of $X_0 \mid x_{\ell+1}$. Compared to (1), the variance of the reverse step gets enlarged by an additive constant of $w_1^2 \operatorname{diag}(\tau_{\ell+1}^2)$ to reflect the uncertainty in the denoiser output. Note that we opt for a simple model where the covariance matrices are the same at all points, whereas Bao et al. (2022) predicted the mean squared residual at every x_ℓ using a neural network.

Algorithm 1 Diffusion Model Variance Calibration

Input: Diffusion model h_θ, calibration set $\mathcal{D}_{\text{cal}} = \{x_{i,0}\}_{i\in[n_{\text{cal}}]}$
Output: Variance parameters $\tau_{1:L}$
for $\ell = 1, \ldots, L$ **do**
  for all i, sample $x_{i,\ell}$ from $X_\ell \mid X_0 = x_{i,0}$
  for all a, set τ_ℓ^a following (3)

3.2. Diffusion Model Sampling with Partial Observation

Next we discuss how to sample from a diffusion model conditioned on an incomplete and noisy observation of x_0. This lays foundation for our Thompson sampling and training algorithms in Sections 3.3 and 4.

Formally, we represent the imperfect observation y of x_0 as $y = m \odot (x_0 + z)$, where $m \in \{0,1\}^K$ is a binary mask, $z \in \mathbb{R}^K$ is a noise vector sampled from $\mathcal{N}(\mathbf{0}_d, \operatorname{diag}(\sigma^2))$, and $\sigma \in \mathbb{R}^K$ is a vector of coordinate-wise standard error in the estimate y of x_0. In this notation, $m^a = 0$ if the a-th entry of the noisy y is unobserved and $m^a = 1$ otherwise. Our goal is to sample from the posterior

$$q(X_0 \mid y) \propto q(y \mid X_0) q(X_0).$$

This problem is closely related to inpainting in computer vision and imputation in statistics, and can be regarded as a special case of the noisy inverse problem. The solution of these with a diffusion prior are well studied in the literature. In Appendix B.2, we provide a detailed discussion on how our algorithm relates to these established methods.

Now we focus on our algorithm and the intuition behind it. Here we assume that both m and σ are given so that $q(y \mid x_0)$ is known. Then, similarly to how unconditional sampling of a diffusion model is designed, we approximate the exact posterior sampling by conditioning the reverse Markov process on $Y = y$ and forming an approximation for each conditional reverse step. This gradually guides our sample towards the observation, and makes it consistent with both the prior and observation.

The above procedure is detailed in Algorithm 2 (see also Figure 5 in Appendix B for an illustration and Appendix C.2 for the complete derivation). Concretely, we perform our initialization and recursive steps as below.

Sampling from $X_L \mid y$. For this part, we simply ignore y and sample from $\mathcal{N}(\mathbf{0}_d, I_d)$ as in Section 2.1.

Sampling from $X_\ell \mid x_{\ell+1}, y$. By Bayes' rule, we have

$$q(X_\ell \mid x_{\ell+1}, y) \propto q(X_\ell \mid x_{\ell+1}) q(y \mid X_\ell). \tag{5}$$

We approximate each term on the right hand side by a Gaussian distribution. For the first term, we use directly the

Algorithm 2 Posterior Sampling with Diffusion Prior

Input: Observation $y \in \mathbb{R}^K$, standard error $\sigma \in \mathbb{R}^K$, binary mask $m \in \{0,1\}^K$, diffusion model h_θ with variance parameters $\tau_{1:L}$
Output: Posterior sample x_0 (resp. $x_{0:L}$) approximately sampled from $X_0 \,|\, y$ (resp. $X_{0:L} \,|\, y$)
Sample initial state $x_L \sim \mathcal{N}(\mathbf{0}_d, I_d)$
for $\ell \in L-1, \dots, 0$ **do**
  Sample unconditional latent $x'_\ell \sim p_{\theta,\tau}(X_\ell \,|\, x_{\ell+1})$
  for $a \in \mathcal{A}$ such that $m^a = 0$ **do**
    Set $x^a_\ell \leftarrow x'^a_\ell$
  for $a \in \mathcal{A}$ such that $m^a = 1$ **do**
    Sample diffused observation $\tilde{y}^a_\ell$ using (8)
    Set x^a_ℓ following (10)

learned prior $p_{\theta,\tau}$. In Algorithm 2, this gives rise to the unconditionally sampled latent variable x'_ℓ.

The second term is approximated by incorporating x_0 as

$$q(y \,|\, x_\ell) = \int q(y \,|\, x_0) q(x_0 \,|\, x_\ell)\, dx_0.$$

As mentioned previously, $q(y \,|\, x_0)$ is known provided that both m and σ are given. Thus we focus on approximating $q(x_0 \,|\, x_\ell)$. A natural choice to consider is the output of the denoiser $\hat{x}_0 = h_\theta(x_\ell, \ell)$. Unfortunately, this would not lead to a closed-form expression of (5). To address this issue, we note that by defining the noise predicted from step ℓ as,

$$\bar{z}_\ell = (x_\ell - \sqrt{\bar{\alpha}_\ell} h_\theta(x_\ell, \ell))/\sqrt{1-\bar{\alpha}_\ell}, \tag{6}$$

this approximation can be rearranged as

$$\hat{x}_0 = (x_\ell - \sqrt{1-\bar{\alpha}_\ell}\bar{z}_\ell)/\sqrt{\bar{\alpha}_\ell}.$$

The above expression is linear in x_ℓ if we treat $\bar{z}_\ell$ as a separate variable. Under the assumption that the noise predicted in two consecutive steps is similar, we may approximate $\bar{z}_\ell$ by $\bar{z}_{\ell+1}$, the noise predicted from step $\ell+1$. This leads to

$$\hat{x}'_0 = (x_\ell - \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1})/\sqrt{\bar{\alpha}_\ell}.$$

With all these in mind, we approximate

$$q(y \,|\, x_\ell) \approx \sqrt{\rho_\ell} \prod_{\substack{a \in \mathcal{A} \\ m^a = 1}} \mathcal{N}\left(y^a; \hat{x}'^a_0, (\sigma^a)^2 + \rho_\ell (\tau^a_{\ell+1})^2\right), \tag{7}$$

where $\rho_\ell = \bar{\alpha}_{\ell+1}(1-\bar{\alpha}_\ell)/(\bar{\alpha}_\ell(1-\bar{\alpha}_{\ell+1}))$ arises because $\bar{z}_\ell$ is replaced by $\bar{z}_{\ell+1}$ in $q(x_0 \,|\, x_\ell)$ (see Appendix C.2 for details). The variance incorporates both the uncertainty σ in observation and the uncertainty $\tau_{\ell+1}$ of predicting the clean sample x_0 from $x_{\ell+1}$.

Having derived the two Gaussian approximations, we employ the perturbation sampling of Papandreou and Yuille (2010) to sample from the posterior of two Gaussians. This, in addition to the unconditionally sampled latent x'_ℓ, requires sampling a diffused observation for each observed entry (the expression below comes from scaling (7) to make it a distribution for X_ℓ)

$$\tilde{y}^a_\ell \sim \mathcal{N}(\sqrt{\bar{\alpha}_\ell} y^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}^a_{\ell+1}, (\zeta^a_{\ell,y})^2). \tag{8}$$

The standard deviation of the above Gaussian is

$$\zeta^a_{\ell,y} = \sqrt{\bar{\alpha}_\ell((\sigma^a)^2 + \rho_\ell(\tau^a_{\ell+1})^2)}, \tag{9}$$

and is just a scaled version of the standard deviation in (7). The mean is obtained using the forward process starting at y^a, but we use the predicted noise instead of a randomly sampled noise vector. This matches our definition of $\hat{x}'_0$.

At this point, for any observed entry a, we have two candidates for x^a_ℓ: x'^a_ℓ that incorporates the prior and $\tilde{y}^a_\ell$ that incorporates the observation. Writing $\zeta^a_{\ell,x}$ for the standard deviation of the a-th entry of $p_{\theta,\tau}(\cdot \,|\, x_{\ell+1})$, we mix them as

$$x^a_\ell = \frac{(\zeta^a_{\ell,x})^{-2} x'^a_\ell + (\zeta^a_{\ell,y})^{-2}\tilde{y}^a_\ell}{(\zeta^a_{\ell,x})^{-2} + (\zeta^a_{\ell,y})^{-2}}. \tag{10}$$

This takes into account both the prior and observation, and their weights reflect their respective uncertainties. For an unobserved entry a, we set $x^a_\ell = x'^a_\ell$. However, note that we do leverage information from those observed entries through the repeated application of the reverse step.

3.3. DiffTS: Thompson Sampling with Diffusion Prior

We finally return to our original goal: Thompson sampling with a diffusion prior. For that, we need to sample from the posterior $q(X_0 \,|\, \mathcal{H}_{t-1})$ at each round. Compared to Section 3.2, our observation is now the interaction history $\mathcal{H}_{t-1} = (a_s, r_s)_{s \in [t-1]}$ up to time $t-1$ and $x_0 = \mu \in \mathbb{R}^K$ is the mean reward vector of the task. Although this setting seems different from Section 3.2, we note that the conditional probability $q(\mathcal{H}_{t-1} \,|\, x_0)$ is proportional to a Gaussian when treated as a function of x_0,

$$q(\mathcal{H}_{t-1} \,|\, x_0) \propto \prod_{s=1}^{t-1} q(r_s \,|\, \mu, a_s) = \prod_{s=1}^{t-1} \mathcal{N}(r_s; \mu^{a_s}, \sigma^2_{\text{reward}}).$$

This holds since the learner's actions depend on the mean reward vector only through their interaction history with the environment, i.e., $q(a_s \,|\, a_1, r_1, \dots, a_{s-1}, r_{s-1}, \mu) = q(a_s \,|\, a_1, r_1, \dots, a_{s-1}, r_{s-1})$.

With this in mind, through a series of computations that we defer to Appendix C.2, we can show that Algorithm 2 almost directly applies: The mask m indicates whether an arm has been pulled up to time $t-1$ (included), with $m^a = 1$ if and only if arm a has been pulled. Then, for any pulled arm a,

Algorithm 3
DiffTS: Thompson Sampling with Diffusion Prior

Input: Diffusion model h_θ with variance parameters $\tau_{1:L}$, assumed noise level $\hat{\sigma} \in \mathbb{R}$
for $t = 1, \ldots$ **do**
  Sample $\tilde{x}_0$ using Algorithm 2 with $y \leftarrow \hat{\mu}_{t-1}$, $\sigma \leftarrow \sigma_{t-1}$, m defined by $m^a = \mathbb{1}\{N^a_{t-1} > 0\}$
  Pull arm $a_t \in \arg\max_{a \in \mathcal{A}} \tilde{x}^a_0$
  Update number of pulls N^a_t, standard error σ^a_t, and empirical mean reward $\hat{\mu}^a_t$ for $a \in \mathcal{A}$

we define N^a_{t-1} as the number of times that the arm was pulled, $\hat{\mu}^a_{t-1} = \sum_{s=1}^{t-1} r_s \, \mathbb{1}\{a_s = a\}/N^a_{t-1}$ as the empirical mean of that arm's reward, and $\sigma^a_{t-1} = \hat{\sigma}/\sqrt{N^a_{t-1}}$ as the corresponding standard error, where $\hat{\sigma}$ is an estimate of σ_{reward}. These quantities summarize the interaction history $\mathcal{H}_{t-1}$ and reduce the problem to that in Section 3.2. We can thus apply Algorithm 2 with $y = \hat{\mu}_{t-1}$ and $\sigma = \sigma_{t-1}$ to sample from $q(X_0 \,|\, \mathcal{H}_{t-1})$ (we set $y^a = \sigma^a = 0$ for arm a that has not been pulled). The rest of the algorithm is unchanged, as we outline in Algorithm 3.

4. Training Diff. Models from Imperfect Data

Standard training procedure of diffusion models requires access to a dataset of clean samples $\mathcal{D}_{\text{tr}} = \{x_{i,0}\}_{i \in [n_{\text{tr}}]}$. Nonetheless, in most bandit applications, it is nearly impossible to obtain such dataset as the exact mean reward vector μ of a task is never directly observed. Instead, one can collect imperfect observations of these vectors, either through previous bandit interactions or forced exploration. Taking this into account, we propose a systematic procedure to train diffusion models from imperfect data. Importantly, the application scope of our methodology goes beyond the bandit setup and covers any situation with imperfect data. As an example, we apply our approach to train from imperfect images (corrupted MNIST and Fashion-MNIST datasets, Xiao et al., 2017) and obtain promising results (details are provided in Appendix F.3).

We describe below how to train a diffusion model when only imperfect data are available and defer the explanation about how to calibrate the model's variance from imperfect data to Appendix C.5.

Setup. To ease exposition, we focus on homogeneous noise here, and study non-homogeneous noise in Appendix C.4. When the noise is homogeneous with variance $\sigma^2_{\text{data}} \in \mathbb{R}$, the samples of the imperfect dataset $\check{\mathcal{D}}_{\text{tr}} = \{y_i\}_{i \in [n_{\text{tr}}]}$ can be written as $y_i = m_i \odot (x_{i,0} + z_i)$ where, as in Section 3.2, $m_i \in \{0,1\}^K$ is a binary mask and z_i is a noise vector sampled from $\mathcal{N}(\mathbf{0}_d, \sigma^2_{\text{data}} I_d)$.[3] In our bandit problem, such dataset can be obtained by randomly pulling a subset of arms once for each arm. We also assume that the associated masks $\{m_i\}_{i \in [n_{\text{tr}}]}$ and the noise level σ_{data} are known. We can thus rewrite the dataset as $\check{\mathcal{D}}_{\text{tr}} = \{(y_i, m_i)\}_{i \in [n_{\text{tr}}]}$.

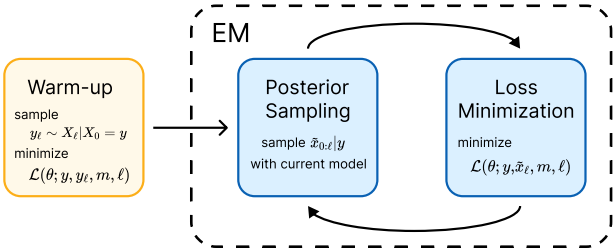


Figure 2. Overview of the proposed training procedure to deal with incomplete and noisy data.

Overall Training Procedure. In presence of perfect data, diffusion model training optimizes the denoising objective (2). However, neither x_0 nor x_ℓ are available when we only have an imperfect dataset $\check{\mathcal{D}}_{\text{tr}}$. To tackle these challenges, we propose an expectation-maximization (EM) procedure, which we illustrate in Figure 2 and summarize in Algorithm 4. The success of our method hinges on the following two observations.

1. The posterior sampling step allows us to progressively improve the quality of the data throughout training.
2. The loss minimization step, via the use of a modified loss function, enables us to learn the denoiser effectively even when we only have access to degraded data.

Our algorithm thus, after a warm-up phase, alternates between these two steps that play respectively the roles of expectation and maximization of a standard EM procedure.

Posterior Sampling. If we had x_0, we could sample x_ℓ via the forward process and optimize the standard objective (2). This is not the case. We thus propose to sample x_0 jointly with x_ℓ given observation y through posterior sampling with the current model parameter. Regarding diffusion model as a probability model over the random variables $X_{0:L}$, this would then correspond to the posterior sampling step done in several variants of stochastic EM (Fort and Moulines, 2003). Concretely, in our experiments, we use Algorithm 2 to construct a dataset of posterior samples $\widetilde{\mathcal{D}}_{\text{tr}} = \{\tilde{x}_{i,0:L}\}_{i \in [\tilde{n}_{\text{tr}}]}$ (note that that the algorithm allows us to sample jointly $\tilde{x}_{0:L}$ given y and that $\tilde{n}_{\text{tr}}$ can be different from n_{tr}).

Loss Minimization. Having obtained the posterior samples,

[3] As we discuss Appendix C.4, the masking of an entry can also be viewed as an observation with infinite variance.

Algorithm 4 Diffusion Model Training from Imperfect (incomplete and noisy) Data

Input: Training set $\check{\mathcal{D}}_{\text{tr}} = \{(y_i, m_i)\}_{i\in[n_{\text{tr}}]}$, calibration set $\check{\mathcal{D}}_{\text{cal}}$, noise standard deviation σ_{data}, number of warm-up, outer, and inner training steps S, J, and S'
Output: Diffusion model h_θ
`Warm-up`
for $s = 1, \ldots, S$ **do**
  Sample y, m from $\check{\mathcal{D}}_{\text{tr}}$
  Sample ℓ uniformly from $\{1, ..., L\}$
  Sample y_ℓ from $X_\ell \mid X_0 = y$
  Take gradient step to minimize $\mathcal{L}(\theta; y, y_\ell, m, \ell)$
`Main Training Procedure`
for $j = 1, \ldots, J$ **do**
  `Posterior Sampling`
  Compute reconstructions errors $\tau_{1:L}$ with Algorithm 6 (Appendix C.5) using $\check{\mathcal{D}}_{\text{cal}}$
  Construct a dataset of posterior samples $\widetilde{\mathcal{D}}'_{\text{tr}} = \{\tilde{x}_{i,0:L}, y_i, m_i\}_{i\in[\tilde{n}_{\text{tr}}]}$ with Algorithm 2
  `Loss Minimization`
  for $s = 1, \ldots, S'$ **do**
    Sample $\tilde{x}_{0:L}, y, m$ from $\widetilde{\mathcal{D}}'_{\text{tr}}$
    Sample ℓ uniformly from $\{1, ..., L\}$
    Take gradient step to minimize $\mathcal{L}(\theta; y, \tilde{x}_\ell, m, \ell)$

we have the option to either maximize the log-likelihood of $\widetilde{\mathcal{D}}_{\text{tr}}$ or minimize the denoising loss $\sum_{\tilde{x}_{0:L}\in\widetilde{\mathcal{D}}_{\text{tr}}} \sum_{\ell=1}^{L} \|\tilde{x}_0 - h_\theta(\tilde{x}_\ell, \ell)\|^2$. Both of these approaches rely heavily on the generated posterior samples, which can bias the model towards generating low-quality samples during early stages of training. To address this issue, we propose to replace the sampled x_0 with corresponding observation y and use a modified denoising loss that is suited to imperfect data. For a small value ε and a regularization parameter λ, the new loss function for a sample pair $(y, \tilde{x}_\ell)$, diffusion step ℓ, and associated mask m is

$$\mathcal{L}(\theta; y, \tilde{x}_\ell, m, \ell) = \|m \odot y - m \odot h_\theta(\tilde{x}_\ell, \ell)\|^2 + 2\lambda\sqrt{\bar{\alpha}_\ell}\sigma_{\text{data}}^2 \, \mathbb{E}_{b\sim\mathcal{N}(\mathbf{0}_d, I_d)} \, b^\top \left(\frac{h_\theta(\tilde{x}_\ell + \varepsilon b, \ell) - h_\theta(\tilde{x}_\ell, \ell)}{\varepsilon} \right). \tag{11}$$

Compared to (2), we have a slightly modified mean squared error term (first term) that handles incomplete data by only considering the observed entries as determined by the element-wise product with the mask. On the top of this, we include a regularization term (second term) that penalizes the denoiser from varying too much when the input changes to account for noisy observation. Our denoising loss finds its roots in works of Metzler et al. (2018); Zhussip et al. (2019), which train denoisers in the absence of clean ground-truth data. In particular, the expectation here is an approximation of the divergence $\text{div}_{\tilde{x}_\ell}(h_\theta(\tilde{x}_\ell, \ell))$ that appears in Stein's unbiased risk estimate (SURE) (Stein, 1981; Eldar, 2008), an unbiased estimator of the mean squared error whose computation only requires the use of noisy samples.[4]

From a practical viewpoint, the regularization term provides a trade-off between the bias and the variance of the learned model. When λ is set to 0, the model learns to generate noisy samples, which corresponds to a flatter prior that encourages exploration. When λ gets larger, the model tries to denoise from the observed noisy samples. This can however deviate the model from the correct prior and accordingly jeopardize the online learning procedure.

Warm-Up. In practice, we observe that posterior sampling with randomly initialized model produces poor training samples. Therefore, in the warm-up phase only, we sample y_ℓ from the forward distribution $\mathcal{N}(\sqrt{\bar{\alpha}_\ell}y, (1 - \bar{\alpha}_\ell)I_d)$ as in standard diffusion model training and minimize loss $\mathcal{L}$ evaluated at y_ℓ instead of $\tilde{x}_\ell$.

5. Numerical Experiments

In this section, we illustrate the benefit of using diffusion prior through numerical experiments on real and synthetic data. Missing experimental details, ablation studies, and additional experiments are presented in Appendices D to F.

Problem Construction. To show the wide applicability of our technique, we consider here three bandit problems inspired by recommender systems, online pricing, and online shortest path routing (Talebi et al., 2017). Detailed description of the task distributions and some visualization that help understand the problem structures are in Appendices D.1 and G. The first and the third problems listed below rely on synthetic data, and we obtain the rewards by perturbing the means with Gaussian noise $\sigma = 0.1$ (we will thus only specify the construction of the means). As for the second problem, we use the iPinYou dataset (Liao et al., 2014).

1. `Popular and Niche` Problem. We consider here the problem of choosing items to recommend to customers. Let $K = 200$. The arms (items) are separated into 40 groups, each of size 5. Among these, 20 groups of arms correspond to the popular items and tend to have high mean rewards. However, these arms are never the optimal ones. The other 20 groups of arms correspond to the niche items. Most of them have low mean rewards but a few of them (those that match the preferences of the customer) have higher mean rewards than those of all the other arms. A task corresponds to a customer so the partitions into groups of popular and niche items are

[4] When $\lambda = 1$, $x_\ell = \tilde{x}_\ell = \sqrt{\bar{\alpha}_\ell}y$, $m = \mathbf{1}$ (i.e., all the entries are observed), and the expectation is replaced by the divergence, we recover SURE up to additive constant $-K\sigma_{\text{data}}^2$. See Appendix C.3 for details.

fixed across tasks while the remaining elements vary.

2. `iPinYou Bidding` Problem. We consider here the problem of setting the bid price in auctions. Let $v = 300$ be the value of the item. Each arm corresponds to a bid price $b \in \{0, ..., 299\}$, and the reward is either $v - b$ when the learner wins the auction or 0 otherwise. The reward distribution of a task is then solely determined by the winning rates which are functions of the learner's bid and the distribution of the highest bid from competitors. For the latter, we use the corresponding empirical distributions of 1352 ad slots from the iPinYou bidding data set (each ad slot is a single bandit task).

3. `2D Maze` Problem. We consider here an online shortest path routing problem on grid graphs. We formalize it as a reward maximization combinatorial bandit (Chen et al., 2013) with semi-bandit feedback. As shown in Figure 3, the super arms are the simple paths between the source and the destination (fixed across all the tasks) whereas the base arms are the edges of the grid graph. At each round, the learner picks a super arm and observes the rewards of all the base arms (edges) that are contained in the super arm (path). Moreover, the edges' mean rewards in each task are derived from a 2D maze that we randomly generate. The mean reward is -1 when there is a wall on the associated case (marked by the black color) and -0.01 otherwise.

Training, Baselines, and Evaluation. To train the diffusion models, for each problem we construct a training set $\mathcal{D}_{\text{tr}}$ (of size 5000 or 1200) and a calibration set $\mathcal{D}_{\text{cal}}$ (of size 1000 or 100) that contain the expected means of the tasks. We then conduct experiments for the following two configurations:

1. *Learn from perfect data:* The priors are learned using $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{cal}}$ that contain the exact mean rewards. Standard training procedure is applied here.

2. *Learn from imperfect data:* The priors are learned using $\check{\mathcal{D}}_{\text{tr}}$ and $\check{\mathcal{D}}_{\text{cal}}$ that are obtained from $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{cal}}$ by perturbing the samples with noise of standard deviation $\sigma_{\text{data}} = 0.1$ and then dropping each feature of a sample with probability 0.5.[5] To tackle this challenging setting, we adopt the approach proposed in Section 4.

In terms of bandit algorithms, we compare our method, DiffTS, with UCB1 (Auer, 2002), with Thompson sampling with Gaussian prior using either diagonal or full covariance matrix (GTS-diag and GTS-full, Thompson, 1933), and with

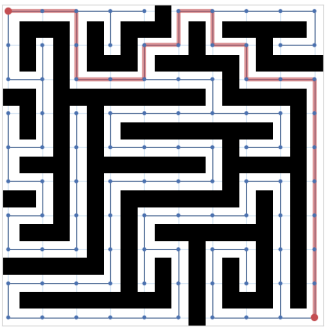


Figure 3. An example task of the `2D Maze` problem. The red path indicates the optimal (super-)arm.

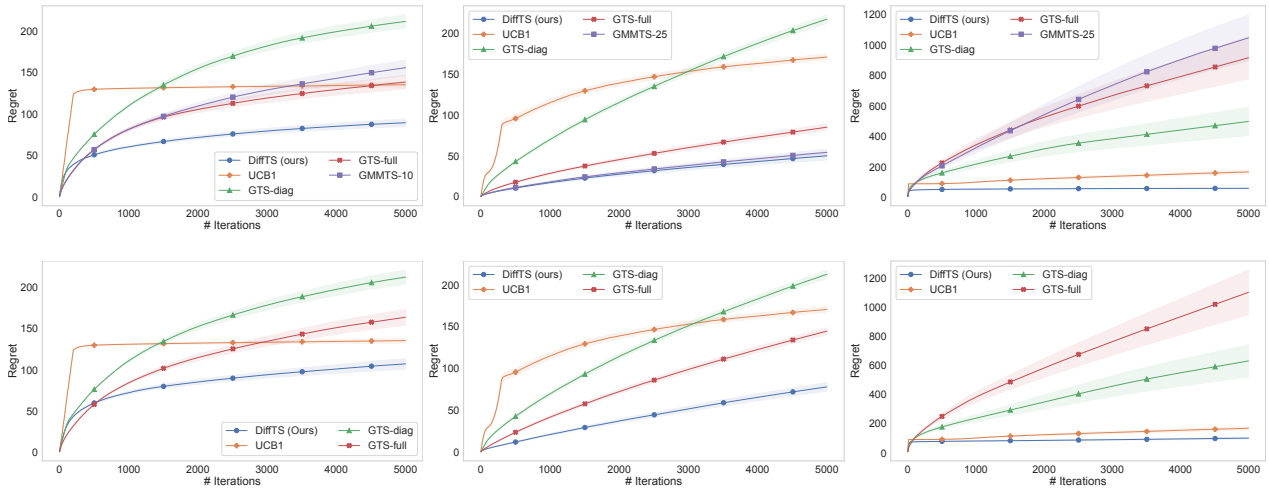
Thompson sampling with Gaussian mixture prior (Hong et al., 2022b).[6] The priors of the Thompson sampling algorithms are also learned with the same perfect / imperfect data that we use to train diffusion models. These thus form strong baselines against which we only improve in terms of the model used to learn the prior. As for the GMM baseline, we use full covariance matrices and consider the case of either 10 or 25 components (GMMTS-10 and GMMTS-25). We employ the standard EM algorithm to learn the GMM when perfect data are available but fail to find any existing algorithm that is able to learn a good GMM on the imperfect data that we consider. We thus skip the GMM baseline for the imperfect data setup. However, as we show, even with imperfect data, DiffTS performs better or comparably to GMMTS learned on perfect data.

To evaluate the performance of the algorithms, we measure their average regret on a standalone test set— for a sequence of arms $(a_t)_{t\in[T]}$ pulled by an algorithm in a bandit task, the induced regret is $\text{Reg}_T = T\mu^{a^\star} - \sum_{t=1}^{T} \mu^{a_t}$, where $a^\star \in \arg\max_{a\in\mathcal{A}} \mu^a$ is an optimal arm in this task. The assumed noise level $\hat{\sigma}$ is fixed to the same value across all the methods.

Results. Our results are presented in Figure 4. For ease of readability, among the two GMM priors (10 and 25 components), we only show the one that achieves smaller regret. We see clearly that throughout the three problems and the two setups considered here, the proposed DiffTS algorithm always has the best performance. The difference is particularly significant in the `Popular and Niche` and `2D Maze` problems, in which the regret achieved by DiffTS is around two times smaller than that achieved by the best performing baseline method. This confirms that using diffusion prior is more advantageous in problems with complex task

[5] Apparently, the performance would degrade when we increase noise level and dropping rate. The breakdown point beyond which the algorithm completely fails is problem-dependent and we do not try to identify it in our experiments.

[6] For the `2D Maze` problem we consider their combinatorial extensions in which the UCB index / sampled mean of a super arm is simply the sum of the corresponding quantities of the contained base arms (Chen et al., 2013; Wang and Chen, 2018).



(a) `Popular and Niche` **(b)** `iPinYou Bidding` **(c)** `2D Maze`

Figure 4. Regret performances on three different problems with priors fitted/trained on either exact expected rewards (top) or partially observed noisy rewards (bottom). The results are averaged over tasks of a test set and shaded areas represent standard errors.

distribution.

We also observe that the use of GMM prior in these two problems leads to worse performance than that of GTS-full, while it yields to competitive performance to DiffTS in the iPinYou Bidding problem. This is consistent with the visualizations in Appendix G, which show that the fitted GMM is only capable of generating good samples in the `iPinYou Bidding` problem. This, however, also suggests that the use of a more complex prior is a double-edged sword, and can lead to poor performance when the data distribution is not faithfully represented.

In Appendix E, we further present ablation studies to investigate the impacts of various components of our algorithm. In summary, we find that both the variance calibration step and the EM-like procedure for training with imperfect data are the most crucial to our algorithms, as dropping either of the two could lead to severe performance degradation. We also affirm that the use of SURE-based regularization does lead to smaller regret, but finding the optimal regularization parameter λ is a challenging problem.

Finally, while the good performance of DiffTS is itself an evidence of the effectiveness of our sampling and training algorithms, we provide additional experiments in Appendix F to show how these methods are relevant in other contexts.

6. Concluding Remarks

In this work, we argue that the flexibility of diffusion models makes them a promising choice for representing complex priors in real-world online decision making problems. Then we design a new algorithm for multi-armed bandits that uses a diffusion prior with Thompson sampling. Our experiments show that this can significantly reduce the regret when compared to existing bandit algorithms. Additionally, we propose a training procedure for diffusion models that can handle imperfect data, addressing a common issue in the bandit setting. This method is of independent interest.

Our work raises a number of exciting but challenging research questions. One potential extension is to apply our approach to meta-learning problems in contextual bandits or reinforcement learning. This would involve modeling a distribution of functions or even of Markov decision processes by diffusion models, which remains a largely unexplored area despite a few attempts that work toward these purposes (Dutordoir et al., 2022; Nava et al., 2022). Another factor not addressed in our work is the uncertainty of the learned model itself, in contrast to the uncertainty modeled by the model. When the diffusion model is trained on limited data, its uncertainty is high, and using it as a fixed prior may lead to poor results. Regarding theoretical guarantees, several recent works (Chen et al., 2023a; Lee et al., 2023) have shown that unconditional sampling of diffusion models can approximate any realistic distribution provided sufficiently accurate score estimate (the score-based interpretation of the predicted noise). Further extending the above results to cope with posterior sampling and deriving regret bounds would be a fruitful direction to work on.

Finally, the posterior sampling algorithm for the diffusion model is a key bottleneck in scaling up our method. There has been significant work on accelerating unconditional sampling of diffusion models (Salimans and Ho, 2021; Dockhorn et al., 2022; Zheng et al., 2022), but incorporating these into posterior sampling remains an open question.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023.

Jean-Yves Audibert, Sebastien Bubeck, and Remi Munos. Best arm identification in multi-armed bandits. In *Proceeding of the 23rd Annual Conference on Learning Theory*, pages 41–53, 2010.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2021.

Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. In *International Conference on Machine Learning*, pages 1555–1584. PMLR, 2022.

Hamsa Bastani, David Simchi-Levi, and Ruihao Zhu. Meta dynamic pricing: Transfer learning across experiments. *Management Science*, 68(3):1865–1881, 2022.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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.

Sebastien Bubeck, Remi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, pages 23–37, 2009.

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

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems 24*, pages 2249–2257, 2012.

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *International Conference on Learning Representations*, 2023a.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023b.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning*, pages 151–159. PMLR, 2013.

Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12413–12422, 2022a.

Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. In *Advances in Neural Information Processing Systems*, 2022b.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-Order Denoising Diffusion Solvers. In *Advances in Neural Information Processing Systems*, 2022.

Vincent Dutordoir, Alan Saul, Zoubin Ghahramani, and Fergus Simpson. Neural diffusion processes. *arXiv preprint arXiv:2206.03992*, 2022.

Yonina C Eldar. Generalized SURE for exponential families: Applications to regularization. *IEEE Transactions on Signal Processing*, 57(2):471–481, 2008.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

Gersende Fort and Eric Moulines. Convergence of the monte carlo expectation maximization for curved exponential families. *The Annals of Statistics*, 31(4):1220–1259, 2003.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. In *Advances in Neural Information Processing Systems*, 2022.

Samarth Gupta, Shreyas Chaudhari, Subhojyoti Mukherjee, Gauri Joshi, and Osman Yağan. A unified approach to translate classical bandit algorithms to the structured bandit setting. *IEEE Journal on Selected Areas in Information Theory*, 1(3):840–853, 2020.

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

Joey Hong, Branislav Kveton, Manzil Zaheer, Yinlam Chow, Amr Ahmed, and Craig Boutilier. Latent bandits revisited. In *Advances in Neural Information Processing Systems 33*, 2020.

Joey Hong, Branislav Kveton, Sumeet Katariya, Manzil Zaheer, and Mohammad Ghavamzadeh. Deep hierarchy in bandits. In *Proceedings of the 39th International Conference on Machine Learning*, 2022a.

Joey Hong, Branislav Kveton, Manzil Zaheer, Mohammad Ghavamzadeh, and Craig Boutilier. Thompson sampling with a mixture prior. In *International Conference on Artificial Intelligence and Statistics*, pages 7565–7586. PMLR, 2022b.

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.

Paul Kuo-Ming Huang, Si-An Chen, and Hsuan-Tien Lin. Improving conditional score-based generation with calibrated classification and joint training. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34: 14938–14954, 2021.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

Zahra Kadkhodaie and Eero Simoncelli. Stochastic solutions for linear inverse problems using the prior implicit in a denoiser. *Advances in Neural Information Processing Systems*, 34:13242–13254, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. SNIPS: Solving noisy inverse problems stochastically. In *Advances in Neural Information Processing Systems*, volume 34, pages 21757–21769, 2021a.

Bahjat Kawar, Gregory Vaksman, and Michael Elad. Stochastic image denoising by sampling from the posterior distribution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1866–1875, 2021b.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

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.

Branislav Kveton, Mikhail Konobeev, Manzil Zaheer, Chihwei Hsu, Martin Mladenov, Craig Boutilier, and Csaba Szepesvari. Meta-Thompson sampling. In *International Conference on Machine Learning*, pages 5884–5893. PMLR, 2021.

T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Tor Lattimore and Remi Munos. Bounded regret for finite-armed structured bandits. In *Advances in Neural Information Processing Systems 27*, pages 550–558, 2014.

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

Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence of score-based generative modeling for general data distributions. In *International Conference on Algorithmic Learning Theory*, pages 946–985. PMLR, 2023.

Hairen Liao, Lingxiao Peng, Zhenchuan Liu, and Xuehua Shen. ipinyou global rtb bidding algorithm competition dataset. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–6, 2014.

Xiuyuan Lu and Benjamin Van Roy. Information-theoretic confidence bounds for reinforcement learning. In *Advances in Neural Information Processing Systems 32*, 2019.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *Proceedings of the 31st International Conference on Machine Learning*, pages 136–144, 2014.

Christopher A Metzler, Ali Mousavi, Reinhard Heckel, and Richard G Baraniuk. Unsupervised learning with Stein's unbiased risk estimator. *arXiv preprint arXiv:1805.10531*, 2018.

Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

Elvis Nava, Seijin Kobayashi, Yifei Yin, Robert K Katzschmann, and Benjamin F Grewe. Meta-learning via classifier (-free) guidance. *arXiv preprint arXiv:2210.08942*, 2022.

George Papandreou and Alan L Yuille. Gaussian sampling by local perturbations. *Advances in Neural Information Processing Systems*, 23, 2010.

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.

Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, 1994.

Sathish Ramani, Thierry Blu, and Michael Unser. Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms. *IEEE Transactions on image processing*, 17(9):1540–1554, 2008.

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.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

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.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

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.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021.

Rajat Sen, Alexander Rakhlin, Lexing Ying, Rahul Kidambi, Dean Foster, Daniel Hill, and Inderjit Dhillon. Top-k extreme contextual bandits with arm hierarchy. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

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.

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.

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

Yang Song, Jascha Sohfl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

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*, 2022.

Charles M Stein. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151, 1981.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Mohammad Sadegh Talebi, Zhenhua Zou, Richard Combes, Alexandre Proutiere, and Mikael Johansson. Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 63(4):915–930, 2017.

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.

Michal Valko, Remi Munos, Branislav Kveton, and Tomas Kocak. Spectral bandits for smooth graph functions. In *Proceedings of the 31st International Conference on Machine Learning*, pages 46–54, 2014.

Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pages 5114–5122. PMLR, 2018.

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chin-Yun Yu, Sung-Lin Yeh, György Fazekas, and Hao Tang. Conditioning and sampling in variational diffusion models for speech super-resolution. *arXiv preprint arXiv:2210.15793*, 2022.

Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. *arXiv preprint arXiv:2211.13449*, 2022.

Magauiya Zhussip, Shakarim Soltanayev, and Se Young Chun. Extending stein's unbiased risk estimator to train deep denoisers with correlated pairs of noisy images. *Advances in neural information processing systems*, 32, 2019.

Appendix

A. Missing Pseudocode

Algorithm 5 Meta-learning Bandits with Diffusion Models

1: `Model Training`
2: **Input:** Training set containing reward observations from different tasks
3: Train a diffusion model h_θ to model the distribution of the mean rewards (in case of imperfect data use Algorithm 4)
4: `Variance Calibration`
5: **Input:** Diffusion model h_θ and calibration set containing reward observations from different tasks
6: Use Algorithm 1 to estimate the mean squared reconstruction errors $\tau_{1:L}$ of the model h_θ from different diffusion steps to calibrate the variance of each reverse step (in case of imperfect data use Algorithm 6)
7: `Bandit Deployment`
8: **Input:** Diffusion model h_θ, variance parameters $\tau_{1:L}$, and assumed noise level $\hat{\sigma}$
9: For any new task, run Thompson sampling with diffusion prior (Algorithm 3) with provided parameters

B. Additional Related Work

In this section we complement our related work with a discussion on the use of prior knowledge in multi-armed bandits and a comparison between our diffusion model posterior sampling algorithm and existing ones.

B.1. Prior Knowledge in Multi-Armed Bandits

Regarding the algorithmic framework, we build upon the well-known Thompson sampling idea introduced by Thompson (1933) nearly a century ago. It has reemerged as one of the most popular algorithms for bandit problems in the last decade due to its simplicity and generality (Chapelle and Li, 2012; Russo and Van Roy, 2014; Russo et al., 2018). Nonetheless, it was not until recently that a series of works (Lu and Van Roy, 2019; Simchowitz et al., 2021) provided a thorough investigation into the influence of the algorithm's prior, and confirmed the benefit of learning a meta-prior in bandits via empirical and theoretical evidences (Cella et al., 2020; Basu et al., 2021; Kveton et al., 2021; Peleg et al., 2022; Bastani et al., 2022). The main difference between our work and the above is the use of a more complex prior, which also goes beyond the previously studied mixture prior (Hong et al., 2022b) and multi-layered Gaussian prior (Hong et al., 2022a).

On a slightly different note, a large corpus of work have investigated other ways to encode prior knowledge, including the use of arm hierarchy (Sen et al., 2021), graphs (Valko et al., 2014), or more commonly a latent parameter shared by the arms (Lattimore and Munos, 2014; Maillard and Mannor, 2014; Hong et al., 2020; Gupta et al., 2020). The use of neural network for contextual bandits was specifically studied by Riquelme et al. (2018), where the authors compared a large number of methods that perform Thompson sampling of network models and found that measuring uncertainty with simple models (e.g., linear models) on top of learned representations often led to the best results. Instead, we focus on non-contextual multi-armed bandits and use neural networks to learn a prior rather than using it to parameterize actions.

B.2. Comparison of Diffusion Posterior Sampling Algorithms

While none of previous diffusion posterior sampling algorithms was designed specifically for the multi-armed bandit setup that we consider, it turns out that our Algorithm 2 shares the same general routine with many existing methods. In fact, a large family of algorithms proposed in the literature for posterior sampling with diffusion models (or equivalently, with trained denoisers or with learned score functions) goes through an iterative process that alternates between unconditional sampling and measurement consistency steps. The main difference thus lies in how the measurement consistency step is

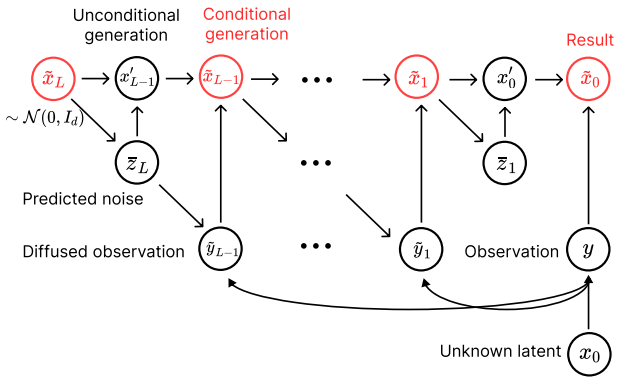


Figure 5. Illustration of the proposed posterior sampling with diffusion prior algorithm (Algorithm 2).

implemented. This can be roughly separated into the following three groups within the context of Algorithm 2.[7] We recall that unconditional sampled is represented by x'_ℓ and is drawn from $p_\theta(X_\ell \,|\, x_{\ell+1})$ [or $p_{\theta,\tau}(X_\ell \,|\, x_{\ell+1})$ in our case].

1. **Direct mix with the observation y.** The simplest solution is to mix directly the unconditional latent variable x'_ℓ with the observed features of y. That is, for a certain $\pi_\ell \in [0, 1]$, we take

$$x_\ell = (1 - m) \odot x'_\ell + m \odot (\pi_\ell x'_\ell + (1 - \pi_\ell) y). \tag{12}$$

   This is essentially the approach taken by Sohl-Dickstein et al. (2015); Jalal et al. (2021); Kawar et al. (2021b); Kadkhodaie and Simoncelli (2021).[8] However, the mismatch between the noise levels of y and x'_ℓ could be detrimental.

2. **Mix with a noisier version of the observation.** Alternatively, the most popular approach in the literature is probably to first pass the observation through the forward process by sampling y_ℓ from $\mathcal{N}(Y_\ell; \sqrt{\bar{\alpha}_\ell} y_\ell, (1 - \bar{\alpha}_\ell) I_d)$ and then perform a weighted average between the unconditional latent variable x'_ℓ and the diffused observation y_ℓ.

$$x_\ell = (1 - m) \odot x'_\ell + m \odot (\pi_\ell x'_\ell + (1 - \pi_\ell) y_\ell). \tag{13}$$

   This idea was introduced in (Song et al., 2021; 2022) and subsequently used by Chung et al. (2022a); Lugmayr et al. (2022) where the authors improved different aspects of the algorithm without modifying the implementation of the measurement consistency step.

3. **Gradient step with respect to denoiser input.** The most involved but also the most general solution is to take a gradient step to ensure that the denoised output from the latent variable is close to our observation after applying the measurement operator. In other words, for a certain stepsize η_ℓ, we set

$$x_\ell = x'_\ell - \eta_\ell \nabla_{x'_\ell} \|m \odot (y - h_\theta(x'_\ell, \ell))\|^2. \tag{14}$$

   This was the method used by Chung et al. (2023) and it was also jointly used with other measurement consistency strategy in (Chung et al., 2022b; Yu et al., 2022).

Provided the above overview, it is clear that our method (Algorithm 2/Figure 5) is similar but different from all the algorithms previously introduced in the literature. In fact, while we also use a diffused observation, it is sampled from $\mathcal{N}(\tilde{Y}_\ell; \sqrt{\bar{\alpha}_\ell} y_0 + \sqrt{1 - \bar{\alpha}_\ell} \bar{z}^a_{\ell+1}, \zeta^a_{\ell,y})$. The use of predicted noise $\bar{z}^a_{\ell+1}$ for the forward process improves the coherence of the output as we will demonstrate on a simple example in Appendix F.2. On the other hand, the third approach mentioned above

[7] In the literature this is often referred to as the problem of inpainting with noisy observation.

[8] Concretely, instead of the mixing step it could be a gradient step that minimizes $\|m \odot (y - x_{\ell+1})\|^2$. This becomes equivalent to (12) if we replace $x_{\ell+1}$ by x'_ℓ and the stepsize is smaller than $1/2$. Our presentation is intended to facilitate the comparison between different methods while keeping the essential ideas. We thus also make similar minor modifications in (13) and (14).

could potentially lead to even better results, but the need of computing the gradient with respect the denoiser makes it much less efficient. Our method can then be regarded as an approximation of (14) by using

$$h_\theta(x'_\ell, \ell) \approx \frac{x'_\ell - \sqrt{1-\bar{\alpha}_{\ell+1}}\bar{z}_{\ell+1}}{\sqrt{\bar{\alpha}_\ell}},$$

which eliminates the need for computing the gradient of the denoiser.

In additional to the aforementioned methods, other alternatives to perform posterior sampling with diffusion models include the use of a dedicated guidance network that learns directly $q(y \,|\, x_\ell)$ (Dhariwal and Nichol, 2021; Song et al., 2021; Huang et al., 2022), annealed Langevin dynamics (Song and Ermon, 2019), Gaussian approximation of posterior (Graikos et al., 2022), and finally, a closed-form expression for the conditional score function and the conditional reverse step can be derived if we assume that the observed noise is carved from the noise of the diffusion process (Kawar et al., 2021a; 2022).

C. Algorithm Design: Derivation, Special Case, and Extensions

In this appendix we complement the presentation of our algorithms by providing underlying mathematical derivations and extension of the training and variance calibration algorithms.

C.1. Reverse Step in Vanilla Diffusion Model

Ho et al. (2020) proposed to set the reverse step of diffusion model to be $q(X_\ell \,|\, x_{\ell+1}, X_0 = \hat{x}_0)$ as explained in (1). This is effectively a Gaussian distribution because

$$q(X_\ell \,|\, x_{\ell+1}, X_0 = \hat{x}_0) \propto q(x_{\ell+1} \,|\, X_\ell, X_0 = \hat{x}_0) q(X_\ell \,|\, X_0 = \hat{x}_0) = q(x_{\ell+1} \,|\, X_\ell) q(X_\ell \,|\, X_0 = \hat{x}_0).$$

By the definition of the forward process, we have $q(x_{\ell+1} \,|\, X_\ell) = \mathcal{N}(x_{\ell+1}; \sqrt{\alpha_{\ell+1}} X_\ell, (1-\alpha_{\ell+1}) I_d)$ and $q(X_\ell \,|\, X_0 = \hat{x}_0) = \mathcal{N}(X_\ell; \sqrt{\bar{\alpha}_\ell}\hat{x}_0, (1-\bar{\alpha}_\ell) I_d)$. The second equality of (1) then follows immediately.

C.2. Reverse Step in Posterior Sampling from Diffusion Prior

We next provide the derivation of the reverse step of our posterior sampling algorithm (variant of Algorithm 2 as described in Section 3.3) that samples from $X_L \,|\, x_{\ell+1}, y$. For this, we write

$$q(x_\ell \,|\, x_{\ell+1}, y) = \frac{q(x_\ell \,|\, x_{\ell+1}) q(y \,|\, x_\ell, x_{\ell+1})}{q(y \,|\, x_{\ell+1})} = \frac{q(x_\ell \,|\, x_{\ell+1}) \int q(y \,|\, x_0) q(x_0 \,|\, x_\ell, x_{\ell+1}) \, dx_0}{q(y \,|\, x_{\ell+1})}. \tag{15}$$

The term $q(x_\ell \,|\, x_{\ell+1})$ can be simply approximated with $p_{\theta,\tau}(x_\ell \,|\, x_{\ell+1})$. As for the integral, one natural solution is to use $q(x_0 \,|\, x_\ell, x_{\ell+1}) = q(x_0 \,|\, x_\ell) \approx p'_{\theta,\tau}(x_0 \,|\, x_\ell)$. Then, for example, if $q(y \,|\, x_0) = \mathcal{N}(y; x_0, \sigma^2 I_d)$, we can deduce

$$\int q(y \,|\, x_0) p'_\theta(x_0 \,|\, x_\ell) \, dx_0 = \mathcal{N}(y; h_\theta(x_\ell, \ell), \sigma^2 I_d + \mathrm{diag}(\tau_\ell^2)).$$

Nonetheless, as the denoiser h_θ can be arbitrarily complex, this does not lead to a close form expression to sample x_ℓ. Therefore, to avoid the use of involved sampling strategy in the recurrent step, we approximate $q(x_0 \,|\, x_\ell, x_{\ell+1})$ in a different way. We first recall that by definition of the diffusion model we may write

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

where both $\bar{Z}_\ell$ and $Z_{\ell+1}$ are random variable with distribution $\mathcal{N}(\mathbf{0}_d, I_d)$. This leads to

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

where

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

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 $p'_\theta(X_0 \,|\, x_{\ell+1})$. To summarize, we write

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

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,

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

In this way, we have approximated $q(x_0 \,|\, x_\ell, x_{\ell+1})$ by a Gaussian with diagonal covariance and with mean that depends only linearly on x_ℓ.

We next place ourselves in the multi-armed bandit setup. Suppose we are in round $t+1$, the observation is now $y = \mathcal{H}_t$ the interaction history up to round t (included) and $x_0 = \mu$ is the mean reward vector. The relation between $y = \mathcal{H}_t$ and $x_0 = \mu$ is as described in Section 3.3. Precisely,

$$q(\mathcal{H}_t \,|\, x_0) \propto \prod_{s=1}^{t} q(r_s \,|\, \mu, a_s) = \prod_{s=1}^{t} \mathcal{N}(r_s; \mu^{a_s}, \sigma^2_{\text{reward}})$$

There exists thus $C(\mathcal{H}_t)$ and $\widetilde{C}(\mathcal{H}_t)$ such that

$$\begin{aligned}
\underbrace{\int q(\mathcal{H}_t \,|\, x_0) q(x_0 \,|\, x_\ell, x_{\ell+1})\, dx_0}_{A} &= \int C(\mathcal{H}_t) \prod_{s=1}^{t} \mathcal{N}(r_s; \mu^{a_s}, \sigma^2) q(x_0 \,|\, x_\ell, x_{\ell+1})\, dx_0 \\
&= \int \widetilde{C}(\mathcal{H}_t) \prod_{\substack{a\in\mathcal{A} \\ N_t^a > 0}} \mathcal{N}(\hat{\mu}_t^a; \mu^a, (\sigma_t^a)^2) q(x_0 \,|\, x_\ell, x_{\ell+1})\, dx_0.
\end{aligned}$$

Using $x_0 = \mu$, the aforementioned approximation of $q(x_0 \,|\, x_\ell, x_{\ell+1})$, and ignoring the multiplicative constant that does not depend on x_ℓ, we get

$$\begin{aligned}
A &\propto \int \prod_{\substack{a\in\mathcal{A} \\ N_t^a > 0}} \mathcal{N}(\hat{\mu}_t^a; x_0^a, (\sigma_t^a)^2) q(x_0 \,|\, x_\ell, x_{\ell+1})\, dx_0 \\
&\approx \sqrt{\rho_\ell} \int \prod_{\substack{a\in\mathcal{A} \\ N_t^a > 0}} \mathcal{N}(\hat{\mu}_t^a; x_0^a, (\sigma_t^a)^2) \prod_{a\in\mathcal{A}} \mathcal{N}\left(x_0^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\,\bar{z}_{\ell+1}^a), \rho_\ell (\tau_{\ell+1}^a)^2\right) dx_0
\end{aligned}$$

$$= \sqrt{\rho_\ell} \prod_{\substack{a\in\mathcal{A}\\ N_t^a>0}} \int \mathcal{N}(\hat{\mu}_t^a; x_0^a, (\sigma_t^a)^2)\mathcal{N}\left(x_0^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a), \rho_\ell(\tau_{\ell+1}^a)^2\right) dx_0^a$$

$$= \sqrt{\rho_\ell} \prod_{\substack{a\in\mathcal{A}\\ N_t^a>0}} \mathcal{N}\left(\hat{\mu}_t^a; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(x_\ell^a - \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a), (\sigma_t^a)^2 + \rho_\ell(\tau_{\ell+1}^a)^2\right)$$

$$\propto \prod_{\substack{a\in\mathcal{A}\\ N_t^a>0}} \mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau_{\ell+1}^a)^2).\right)$$

Plugging the above into (15), we obtain $\tilde{q}(x_\ell \,|\, x_{\ell+1}, \mathcal{H}_t) = \prod_{a\in\mathcal{A}} \tilde{q}(x_\ell^a \,|\, x_{\ell+1}, \mathcal{H}_t)$ where $\tilde{q}(x_\ell^a \,|\, x_{\ell+1}, \mathcal{H}_t) = p_{\theta,\tau}(x_\ell^a \,|\, x_{\ell+1})$ if a is never pulled and otherwise it is the distribution satisfying

$$\tilde{q}(x_\ell^a \,|\, x_{\ell+1}, \mathcal{H}_t) \propto p_{\theta,\tau}(x_\ell^a \,|\, x_{\ell+1})\mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau_{\ell+1}^a)^2)\right). \tag{16}$$

To conclude, we resort to the following lemma (see Papandreou and Yuille, 2010 for more general results).

Lemma 1. *Let $\mu_1, \mu_2, \sigma_1, \sigma_2 \in \mathbb{R}$. The following two sampling algorithms are equivalent.*

1. *Sample x directly from the distribution whose density is proportional the product $\mathcal{N}(\mu_1, \sigma_1^2)\mathcal{N}(\mu_2, \sigma_2^2)$.*
2. *Sample x_1 from $\mathcal{N}(\mu_1, \sigma_1^2)$, x_2 from $\mathcal{N}(\mu_2, \sigma_2^2)$, and compute $x = \sigma_1^{-2}x_1 + \sigma_2^{-2}x_2/(\sigma_1^{-2} + \sigma_2^{-2})$.*

Proof. It is well known that the product of two Gaussian PDFs is itself proportional to a Gaussian PDF. Concretely, we have

$$\mathcal{N}(\mu_1, \sigma_1^2)\mathcal{N}(\mu_2, \sigma_2^2) \propto \mathcal{N}\left(\frac{\sigma_1^{-2}\mu_1 + \sigma_2^{-2}\mu_2}{\sigma_1^{-2} + \sigma_2^{-2}}, \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}}\right). \tag{17}$$

On the other hand, the linear combination of two independent Gaussian variables is also a Gaussian variable. For X_1, X_2 that follow $\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2)$ and $X = \sigma_1^{-2}X_1 + \sigma_2^{-2}X_2/(\sigma_1^{-2} + \sigma_2^{-2})$, we can compute

$$\mathbb{E}[X] = \frac{\sigma_1^{-2}\,\mathbb{E}[X_1] + \sigma_2^{-2}\,\mathbb{E}[X_2]}{\sigma_1^{-2} + \sigma_2^{-2}} = \frac{\sigma_1^{-2}\mu_1 + \sigma_2^{-2}\mu_2}{\sigma_1^{-2} + \sigma_2^{-2}},$$

$$\mathrm{Var}[X] = \frac{\sigma_1^{-4}\,\mathrm{Var}[X_1] + \sigma_2^{-4}\,\mathrm{Var}[X_2]}{(\sigma_1^{-2} + \sigma_2^{-2})^2} = \frac{\sigma_1^{-2} + \sigma_2^{-2}}{(\sigma_1^{-2} + \sigma_2^{-2})^2} = \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}}.$$

Therefore, X follows the distribution of (17) and computing the linear combination of x_1 and x_2 as suggested is equivalent to sampling directly from the resulting distribution. □

We obtain the algorithm presented in Section 3.3 by applying Lemma 1 to (16) with

$$\mathcal{N}(\mu_1, \sigma_1^2) \leftarrow p_{\theta,\tau}(x_\ell^a \,|\, x_{\ell+1})$$
$$\mathcal{N}(\mu_2, \sigma_2^2) \leftarrow \mathcal{N}\left(x_\ell^a; \sqrt{\bar{\alpha}_\ell}\hat{\mu}_t^a + \sqrt{1-\bar{\alpha}_\ell}\bar{z}_{\ell+1}^a, \bar{\alpha}_\ell((\sigma_t^a)^2 + \rho_\ell(\tau_{\ell+1}^a)^2)\right).$$

C.3. On SURE-based Regularization

In this part we show how the loss function (11) is related to Stein's unbiased risk estimate (SURE). We first note that by definition of the diffusion process, we have $x_\ell = \sqrt{\bar{\alpha}_\ell}x_0 + \sqrt{1-\bar{\alpha}_\ell}\bar{z}_\ell$ where $\bar{z}_\ell$ is a random variable following the distribution $\mathcal{N}(\mathbf{0}_d, I_d)$. Moreover, $\sqrt{\bar{\alpha}_\ell}h_\theta(x_\ell, \ell)$ is an estimator of $\sqrt{\bar{\alpha}_\ell}x_0$ from x_ℓ. The corresponding SURE thus writes

$$\mathrm{SURE}(\sqrt{\bar{\alpha}_\ell}h_\theta(\cdot, \ell)) = \|\sqrt{\bar{\alpha}_\ell}h_\theta(x_\ell, \ell) - x_\ell\|^2 - K(1-\bar{\alpha}_\ell) + 2(1-\bar{\alpha}_\ell)\,\mathrm{div}_{x_\ell}(\sqrt{\bar{\alpha}_\ell}h_\theta(x_\ell, \ell)).$$

If it holds $x_\ell = \sqrt{\bar{\alpha}_\ell}y$ while y follows the distribution $\mathcal{N}(x_0, \sigma^2 I_d)$, we get immediately $1 - \bar{\alpha}_\ell = \bar{\alpha}_\ell\sigma^2$. The above can thus be rewritten as

$$\mathrm{SURE}(\sqrt{\bar{\alpha}_\ell}h_\theta(\cdot, \ell)) = \|\sqrt{\bar{\alpha}_\ell}h_\theta(x_\ell, \ell) - \sqrt{\bar{\alpha}_\ell}y\|^2 - K\bar{\alpha}_\ell\sigma^2 + 2\bar{\alpha}_\ell^{\frac{3}{2}}\sigma^2\,\mathrm{div}_{x_\ell}(h_\theta(x_\ell, \ell)).$$

Dividing the above by $\bar{\alpha}_\ell$ we get an unbiased estimate of $\mathbb{E}[\|h_\theta(x_\ell, \ell) - x_0\|^2]$, i.e.,

$$\mathbb{E}[\|h_\theta(x_\ell, \ell) - x_0\|^2] = \mathbb{E}[\|h_\theta(x_\ell, \ell) - y\|^2 - K\sigma^2 + 2\sqrt{\bar{\alpha}_\ell}\sigma^2 \operatorname{div}_{x_\ell}(h_\theta(x_\ell, \ell))].$$

On the right hand side inside expectation we recover Eq. (11) with $m = \mathbf{1}$ and $\lambda = 1$ by replacing x_ℓ by $\tilde{x}_\ell$ and the divergence by its Monte-Carlo approximation (Ramani et al., 2008).

C.4. Training from Bandit Observations or Observations with Non-Homogeneous Noise

We discuss here how to extend Algorithm 4 to cope with bandit observations and observations with non-homogeneous noise. As suggested in Section 3.3, when the observations come from bandit interactions and each arm can be pulled more than once, we can first summarize the interaction history by the empirical mean and the vector of adjusted standard deviation. Therefore, it actually remains to address the case of non-homogeneous noise where the noise vector z_i is sampled from $\mathcal{N}(\mathbf{0}_d, \operatorname{diag}({\sigma_i}^2))$ for some vector $\sigma_i \in \mathbb{R}^K$. As the design of our posterior sampling algorithm already takes this into account, the posterior sampling steps of the algorithm remains unchanged. The only difference would thus lie in the definition of loss (11). Intuitively, we would like to give more weights to samples that are less uncertain. This can be achieved by weighting the loss by the inverse of the variances, that is, we set

$$\mathcal{L}'(\theta; y, \tilde{x}_\ell, m, \sigma, \ell) = \sum_{a=1}^{K} \frac{m^a |y^a - h_\theta^a(\tilde{x}_\ell, \ell)|}{(\sigma^a)^2} + 2\lambda\sqrt{\bar{\alpha}_\ell}\, \mathbb{E}_{b\sim\mathcal{N}(\mathbf{0}_d, I_d)}\, b^\top \left(\frac{h_\theta(\tilde{x}_\ell + \varepsilon b, \ell) - h_\theta(\tilde{x}_\ell, \ell)}{\varepsilon} \right). \tag{18}$$

To make sure the above loss is always well defined, we may further replace $(\sigma^a)^2$ by $(\sigma^a)^2 + \delta$ for some small $\delta > 0$. It is worth noticing that one way to interpret the absence of observation $m^a = 0$ is to set the corresponding variance to infinite, i.e., $\sigma^a = +\infty$. In this case we see there is even no need of m anymore as the coordinates with $\sigma^a = +\infty$ would already be given 0 weight. Finally, to understand why we choose to weight with the inverse of the variance, we consider a scalar x, and a set of noisy observations $y_1, \dots, y_n$ respectively drawn from $\mathcal{N}(x, \sigma_1^2), \dots, \mathcal{N}(x, \sigma_n^2)$. Then, the maximum likelihood estimate of x is $\sum_{i=1}^n \sigma_i^2 y_i / (\sum_{i=1}^n \sigma_i^2)$.

C.5. Variance Calibration with Imperfect Data

As mentioned in Section 3.1, a reliable variance estimate of the reverse process is essential for building a good diffusion prior. This holds true not only for the online learning process at test phase, but also for the posterior sampling step of our training procedure. The algorithm introduced in Section 3.1 calibrates the variance through perfect data. In this part, we extend it to operate with imperfect data.

Let $\check{\mathcal{D}}_{\text{cal}}$ be a set of imperfect data constructed in the same way as $\check{\mathcal{D}}_{\text{tr}}$. We write $\check{\mathcal{D}}_{\text{cal}}^a = \{(y, m) \in \check{\mathcal{D}}_{\text{cal}} : m^a = 1\}$ as the subset of $\check{\mathcal{D}}_{\text{cal}}$ for which a noisy observation of the feature at position a is available. Our algorithm (outlined in Algorithm 6) is inspired by the following two observations. First, if the entries are missing completely at random, observed y^a of $\check{\mathcal{D}}_{\text{cal}}^a$ and sampled $x_0^a + z^a$ with $x_0 \sim \mathcal{Q}_0$ and $z \sim \mathcal{N}(\mathbf{0}_d, \sigma_{\text{data}}^2 I_d)$ have the same distribution. Moreover, for any triple (x_0, y, x_ℓ) with $y = x_0 + z$, $x_\ell = \sqrt{\bar{\alpha}_\ell} x_0 + \sqrt{1 - \bar{\alpha}_\ell}\bar{z}_\ell$ and x_0, z, and $\bar{z}_\ell$ sampled independently from $\mathcal{Q}_0$, $\mathcal{N}(\mathbf{0}_d, \sigma_{\text{data}}^2 I_d)$, and $\mathcal{N}(\mathbf{0}_d, I_d)$, it holds that

$$\mathbb{E}[\|y^a - h_\theta^a(x_\ell, \ell)\|^2] = \mathbb{E}[\|x_0^a - h_\theta^a(x_\ell, \ell)\|^2] + \sigma_{\text{data}}^2.$$

We can thus estimate $\mathbb{E}[\|x_0^a - h_\theta^a(x_\ell, \ell)\|^2]$ if we manage to pair each $y^a \in \check{\mathcal{D}}_{\text{cal}}^a$ with a such x_ℓ.

We again resort to Algorithm 2 for the construction of x_ℓ (referred to as $\tilde{x}_\ell$ in Algorithm 6 and hereinafter). Unlike the training procedure, here we first construct $\tilde{x}_0$ and sample $\tilde{x}_\ell$ from $X_\ell \,|\, \tilde{x}_0$ to decrease the mutual information between $\tilde{x}_\ell$ and y. Nonetheless, the use of our posterior sampling algorithm itself requires a prior with calibrated variance. To resolve the chicken-and-egg dilemma, we add a warm-up step where we precompute the reconstruction errors with Algorithm 1 by treating $\check{\mathcal{D}}_{\text{cal}}$ as the perfect dataset. In our experiments, we observe this step yields estimates of the right order of magnitude but not good enough to be used with Thompson sampling, while the second step brings the relative error to as small as 5% compare to the estimate obtained with perfect validation data using Algorithm 1.

D. Missing Experimental Details

In this section, we provide missing experimental details mainly concerning the construction of the problem instances and the learning of priors. All the simulations are run on an Amazon p3.2xlarge instance equipped with 8 NVIDIA Tesla V100 GPUs.

Algorithm 6 Diffusion Model Variance Calibration from Imperfect (incomplete and noisy) Data

Input: Diffusion model h_θ, calibration set $\check{\mathcal{D}}_{\text{cal}} = \{y_i, m_i\}_{i \in [n_{\text{cal}}]}$, noise standard deviation σ_{data}
Output: Variance parameters $\tau_{1:L}$
`Data Set Preprocessing`
Precompute reconstructions errors $\tau_{1:L}$ with Algorithm 1 and $\mathcal{D}_{\text{cal}} \leftarrow \check{\mathcal{D}}_{\text{cal}}$ (masks ignored)
Construct $\widetilde{\mathcal{D}}_{\text{cal}} = \{\tilde{x}_{i,0}, y_i, m_i\}_i$ with Algorithm 2
`Variance Calibration`
for $\ell = 1 \dots L$ **do**
  Construct $\widetilde{\mathcal{D}}_{\text{cal},\ell} = \{\tilde{x}_{i,\ell}, y_i, m_i\}_i$ by sampling $\tilde{x}_{i,\ell}$ from $X_\ell \mid \tilde{x}_{i,0}$
  for $a = 1 \dots K$ **do**
    Let $\widetilde{\mathcal{D}}^a_{\text{cal},\ell} = \{\tilde{x}_\ell, y : (\tilde{x}_\ell, y, m) \in \widetilde{\mathcal{D}}_{\text{cal},\ell}, m^a = 1\}$
    Set $\tau^a_\ell \leftarrow \sqrt{\frac{1}{n_{\text{cal}}} \sum_{\tilde{x}_\ell, y \in \widetilde{\mathcal{D}}^a_{\text{cal},\ell}} \|x^a_0 - h^a_\theta(x_\ell, \ell)\|^2 - \sigma^2_{\text{data}}}$

D.1. Construction of Bandit Instances

We provide below more details on how the bandit instances are constructed in our problems. Besides the three problems described in Section 5, we consider an additional `Labeled Arms` problem that will be used for our ablation study. Some illustrations of the constructed instances and the vectors generated by learned priors are provided in Appendix G. As in `Popular and Niche` and `2D Maze` problems, in the `Labeled Arms` problem we simply add Gaussian noise of standard deviation 0.1 to the mean when sampling the reward. For these three problems we thus only explain how the means are constructed.

1. `Popular and Niche` ($K = 200$ arms). The arms are split into 40 groups of equal size. 20 of these groups represent the 'popular' items while the other 20 represent the 'niche' items. For each bandit task, we first construct a vector $\bar{\mu}$ whose coordinates' values default to 0. However, we randomly choose 1 to 3 groups of niche items and set the value of each of these items to 1 with probability 0.7 (independently across the selected items). Similarly, we randomly choose 15 to 17 groups of popular items and set their values to 0.8. Then, to construct the mean reward vector μ, we perturb the values of $\bar{\mu}$ by independent Gaussian noises with standard deviation of 0.1. After that, we clip the values of the popular items to make them smaller than 0.95 and clip the entire vector to the range $[0, 1]$.

2. `iPinYou bidding` ($K = 300$ arms). The set of tasks is constructed with the help of the iPinYou data set (Liao et al., 2014). This data set contains logs of ad biddings, impressions, clicks, and final conversions, and is separated into three different seasons. We only use the second season that contains the ads from 5 advertisers (as we are not able to find the data for the first and the third season). To form the tasks, we further group the bids according to the associated ad slots. By keeping only those ad slots with at least 1000 bids, we obtain a data set of 1352 ad slots. Then, the empirical distribution of the paying price (i.e., the highest bid from competitors) of each ad slot is used to computed the success rate of every potential bid $b \in \{0, \dots, 299\}$ set by the learner. The reward is either $300 - b$ when the learner wins the auction or 0 otherwise. Finally, we divide everything by the largest reward that the learner can ever get in all the tasks to scale the rewards to range $[0, 1]$.

3. `2D Maze` ($K = 180$ base arms). For this problem, we first use the code of the github repository `MattChanTK/gym-maze`[9] to generate random 2D mazes of size 19×19. Then, each bandit task can be derived from a generated 2D maze by associating the maze to a weighted 10×10 grid graph. As demonstrated by Figure 3, each case corresponds to either a node or an edge of the grid graph. Then, the weight (mean reward) of an edge (base arms) is either -1 or -0.01 depending on either there is a wall (in black color) or not (in white color) on the corresponding case. An optimal arm in this problem would be a path that goes from the source to the destination without bumping into any walls in the corresponding maze.

4. `Labeled Arms` ($K = 500$ arms). This problem is again inspired by applications in recommender systems. We are provided here a set of 50 labels $\mathcal{L} = \{1, ..., 50\}$. Each arm is associated to a subset $\mathcal{L}^a$ of these labels with size

[9] https://github.com/MattChanTK/gym-maze

$\text{card}(\mathcal{L}^a) = 7$. 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 $\bar{\mu}^a = 1 - 1/4^{\text{card}(\mathcal{L}^a \cap \mathcal{L}_B)}$ so that the more the two sets intersect the higher the value. Finally, to obtain the mean rewards μ, we perturb the coordinates of $\bar{\mu}$ by independent Gaussian noises of standard deviation 0.1 and scale the resulting vector to the range $[0, 1]$.

Training, Calibration, and Test Sets. Training, calibration, and test set are constructed for each of the considered problem. Their size are fixed at 5000, 1000, 100 for the `Popular and Niche`, `2D Maze`, and `Labeled Arms` Problems, and at 1200, 100, and 52 for the `iPinYou Bidding` problem.

D.2. Diffusion Models– Model Design

In all our experiments (including the ones described in Appendices E and F), we set the diffusion steps of the diffusion models to $L = 100$ and adopt a linear variance schedule that varies from $1 - \alpha_1 = 10^{-4}$ to $1 - \alpha_L = 0.1$. The remaining details are customized to each problem, taking into account the specificity of the underlying data distribution.

1. `Labeled Arms` and `Popular and Niche`. These two problems have the following two important features: (*i*) The expected means of the bandit instances do not exhibit any spatial correlations (see Figures 16a and 17a). (*ii*) The values of the expected means are nearly binary.

   The first point prevents us from using the standard U-Net architecture. Instead, we consider an architecture adapted from Kong et al. (2020); Rasul et al. (2021), with 5 residual blocks and each block containing 6 residual channels.[10] Then, to account for the lack of spatial correlations, we add a fully connected layer at the beginning to map the input to a vector of size 128×6, 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 that maps a vector of size 128×6 to a vector of size K. We find that these minimal modification already enable the model to perform well on these two problems.

   As for the latter point, we follow Chen et al. (2023b) and train the denoisers to predict the clean sample x_0 as it is reported in the said paper that this leads to better performance when the data are binary.

2. `iPinYou Bidding`. As shown in Figure 20, the pattern of this problem looks similar to that of natural images. We therefore adopt the standard U-Net architecture, with an adaption to the 1-dimensional case as described by (Janner et al., 2022). The model has three feature map resolutions (from 300 to 75) and the number of channels for each resolution is respectively 16, 32, and 64. No attention layer is used. The denoiser is trained to predict noise as in Ho et al. (2020); Song and Ermon (2019).

3. `2D Maze` As explained in Appendix D.1 and illustrated in Figure 3, the weighted grid graphs are themselves derived by the 2D mazes. We can accordingly establish a function that maps each 10×10 weighted grid graph to an image of size 19×19 and vice-versa— it suffices to match the value of each associated (edge, pixel) pair. For technical reason, we further pad the 19×19 images to size 20×20 by adding one line of -1 at the right and one row of -1 at the bottom (see Figure 21). We then train diffusion models to learn the distribution of the resulting images. For this, we use a 2-dimensional U-Net directly adapted from the ones used by Ho et al. (2020). The model has three feature map resolutions (from 20×20 to 5×5) and the number of channels for each resolution is respectively 32, 64, and 128. A self-attention block is used at every resolution. We again train the denoiser to predict the clean sample x_0 as we have binary expected rewards here (-0.01 or -1).

D.3. Diffusion Models– Training

Through out our experiments, we use Adam optimizer with learning rate 5×10^{-4} and exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The batch size and the epsilon constant in SURE-based regularization are respectively fixed at 128 and $\epsilon = 10^{-5}$. When the perfect data sets $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{cal}}$ are provided, we simply train the diffusion models for 15000 steps on the training set $\mathcal{D}_{\text{tr}}$ and apply Algorithm 1 on the calibration set $\mathcal{D}_{\text{cal}}$ to calibrate the variances. The training procedure is more complex when only imperfect data are available. We provide the details below.

[10] These numbers are rather arbitrary and do not seem to affect much our results.

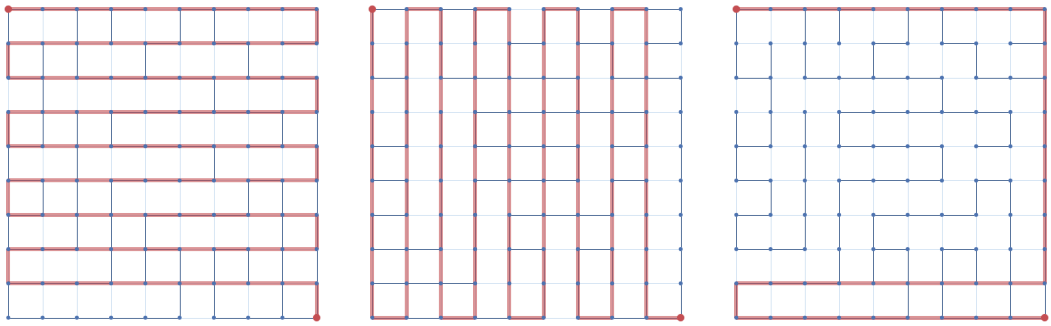


Figure 6. The three paths (super-arms) for UCB1 initialization in the `2D Maze` experiment.

Posterior Sampling. As explained in Section 4 and Algorithm 4, to train from imperfect data we sample the entire chain of diffused samples $\tilde{x}_{0:L}$ from the posterior. However, while Algorithm 2 performs sampling with predicted noise $\bar{z}_{\ell+1}$ and as we will show in Appendix F.2, this indeed leads to improved performance in a certain aspect, we observe that when used for training, it prevents the model from making further progress. We believe this is because in so doing we are only reinforcing the current model with their own predictions. Therefore, to make the method effective, in our experiments we slightly modify the posterior sampling algorithm that is used during training. While we still construct samples $x_{0:L}$ following Algorithm 2, the samples $\tilde{x}_{0:L}$ used for the loss minimization phase are obtained by replacing $\bar{z}_{\ell+1}$ (line 9) by $\tilde{z}_{\ell+1}$ sampled from $\mathcal{N}(\mathbf{0}_d, I_d)$ in the very last sampling step. That is, from $x_{\ell+1}$ we sample both x_ℓ for further iterations of the algorithm and $\tilde{x}_\ell$ to be used for loss minimization.

Training Procedure Specification. When training and validation data are incomplete and noisy, we follow the training procedure described in Algorithm 4 with default values $S = 15000$ warm-up steps, $J = 3$ repeats, and $S' = 3000$ steps within each repeat (thus 24000 steps in total). Moreover, during the warm-up phase we impute the missing value with constant 0.5 when constructing the diffused samples $\tilde{x}_\ell$. As for the regularization parameter λ, we fix it at 0.1 for the `Popular and Niche`, `2D Maze`, and `Labeled Arms` problems.

Nevertheless, training from imperfect data turns out to be difficult for the iPinYou Bidding problem. We conjecture this is both because the training set is small and because we train the denoiser to predict noise here. Two modifications are then brought to the above procedure to address the additional difficulty. First, as SURE-based regularization can prevent the model from learning any pattern from data when information is scarce, we drop it for the warm-up phase and the first two repeats (i.e., the first 21000 steps). We then get a model that has learned the noisy distribution. We then add back SURE-based regularization with $\lambda = 0.25$ in the third repeat. After the 24000 steps, the model is good enough at reconstructing the corrupted data set, but the unconditionally generated samples suffer from severe mode collapse. Provided that the reconstructed samples are already of good quality, we fix the latter issue simply by applying standard training on the reconstructed samples for another 3000 steps (thus 27000 training steps in total).

D.4. Other Details

In this part we provide further details about the evaluation phase and the baselines.

Assumed Noise Level. All the bandit algorithms considered in our work take as input a hyperparameter $\hat{\sigma}$ that should roughly be in the order of the scale of the noise. For the results presented in Section 5, we set $\hat{\sigma} = 0.1$ for the `Popular and Niche` and `2D Maze` problems and $\hat{\sigma} = 0.2$ for the `iPinYou Bidding` problem. The former is exactly the ground truth standard deviation of the underlying noise distribution. For the `iPinYou Bidding` problem the noise is however not Gaussian, and $\hat{\sigma} = 0.2$ is approximately the third quartile of the empirical distribution of the expected rewards' standard deviations (computed across tasks and arms). In Appendix F.1, we present additional results for algorithms run with different assumed noise levels $\hat{\sigma}$.

UCB1. The most standard implementation of the UCB1 algorithm sets the upper confidence bound to

$$U_t^a = \hat{\mu}_t^a + \hat{\sigma}\sqrt{\frac{2\log t}{N_t^a}}. \tag{19}$$

Instead, in our experiments we use $U_t^a = \hat{\mu}_t^a + \hat{\sigma}/\sqrt{N_t^a}$. Eq. (19) is more conservative than our implementation, and we thus do not expect it to yield smaller regret within the time horizon of our experiments.

UCB1 Initialization. In contrary to Thompson sampling-based methods, UCB1 typically requires an initialization phase. For vanilla multi-armed bandits (`Popular and Niche`, `iPinYou Bidding`, and `Labeled Arms`) this simply consists in pulling each arm once. For combinatorial bandits we need to pull a set of super arms that covers all the base arms. In the `2D Maze` experiment we choose the three paths shown in Figure 6.

Gaussian Prior with Imperfect Data. To fit a Gaussian on incomplete and noisy data, we proceed as follows: First, we compute the mean of arm a from those samples that have observation for a. Next, in a similar fashion, the covariance between any two arms are only computed with samples that have observations for both arms. Let the resulting matrix be $\hat{\Sigma}$. Since the covariance matrix of the sum of two independently distributed random vectors (in our case X_0 and noise) is the sum of the covariance matrices of the two random vectors, we further compute $\hat{\Sigma}' = \hat{\Sigma} - \sigma_{\text{data}}^2 I_d$ as an estimate of the covariance matrix of X_0. Finally, as $\hat{\Sigma}'$ is not necessarily positive semi-definite and can even have negative diagonal entries, for TS with diagonal covariance matrix we threshold the estimated variances to be at least 0 and for TS with full covariance matrix we threshold the eigenvalues of the estimated covariance matrix $\hat{\Sigma}'$ to be at least 10^{-4}.[11]

Arm Selection in `2D Maze` Problem. All the algorithms we use in the `2D Maze` problem first compute/sample some values for each base arm (edge) and then select the super arm (path) that maximizes the sum of its base arms' values (for DiffTS we first map the sampled 20×20 image back to a weighted graph and the remaining is the same). Concretely, we implement this via Dijkstra's shortest path algorithm applied to the weighted graphs with weights defined as the opposite of the computed/sampled values. However, these weights are not guaranteed to be non-negative, and we thus clip all the negative values to 0 before computing the shortest path.

E. Ablation Study

In this appendix, we perform ablation studies on the `Popular and Niche` and `Labeled Arms` problems to explore the impacts of various design choices of our algorithms.

E.1. Predicted versus Sampled Noise in Posterior Sampling

In the DiffTS scheme that we develop (Algorithms 2 and 3), we propose to use the predicted noise $\bar{z}_{\ell+1}$ in the construction of the diffused observation $\tilde{y}_\ell$. Alternatively, we can replace it by a sampled noise vector $\tilde{z}_{\ell+1}$ (the resulting algorithm then becomes very similar to the 'mix with a noisier version of the observation' approach presented in Appendix B.2). In Figure 9, we investigate how this decision affects the performance of DiffTS with diffusion priors trained on perfect data set $\mathcal{D}_{\text{tr}}$. It turns out that for the two problems considered here, there is not clear winner between the two options. However, it seems that using only sampled noise produces noisier samples, which leads to significant increase in regret in the `Labeled Arms` problem. We further confirm this intuition in Appendix F.2, where we show on a toy problem that the use of predicted noise often leads to samples that are more consistent with the learned prior. However, this does not always lead to performance improvement in bandit problems as the learned prior is never perfect.

E.2. Importance of Variance Calibration

Throughout our work, we have highlighted multiple times the importance of equipping the diffusion model with a suitable variance estimate. We demonstrate this in Figure 8. We consider diffusion priors trained on the perfect data set $\mathcal{D}_{\text{tr}}$ along with three different reverse variance schedules: (*i*) calibrated, i.e., Eq. (4); (*ii*) non-calibrated, i.e., Eq. (1); (*iii*) partially calibrated– precisely, only the variance of $X_0 \,|\, x_1$ is calibrated. We see clearly that a non-calibrated reverse variance schedule leads catastrophic regret performance. This is because the sampling process relies too much on the learned model; in particular, the variance of $p_\theta(X_0 \,|\, x_1)$ is fixed at zero. Instead, calibrating $X_0 \,|\, x_1$ itself already leads to significant decrease in regret, making it as competitive as (and sometimes even better than) the fully calibrated alternative. This suggests that the trade-off between the learned model and the observations mainly occurs at the last reverse step, whereas enlarging the variance of the remaining reverse steps has little to no effect. [Yet, it is also clear from the experiment on the `Popular and Niche` problem with presumed noise standard deviation 0.5 that calibrating the variance of all the reverse steps may

[11] Our implementation requires the prior covariance matrix to be positive definite.

still be beneficial in some situation.]

E.3. Ablation Study for Training from Imperfect Data

Our algorithm for training from imperfect data (Algorithm 4) makes two important modifications to the original training scheme: the Expectation Maximization-like procedure (abbreviated as EM hereinafter) and the use of SURE-based regularization. Below we discuss their effects for three types of data: noisy data, incomplete data, and noisy and incomplete data. We fix all the hyper-parameters to the ones used in the main experiment unless otherwise specified. In particular, we set the noise standard deviation to $\sigma_{\text{data}} = 0.1$ for noisy data and the missing rate to 0.5 for incomplete data.

For comparison, we also plot the regrets for the full covariance Gaussian prior baseline. The means and the covariance of the prior are fitted with the three types of imperfect data that are used to train and calibrate the diffusion models, following the procedure detailed in Appendix D.4.

Training from Noisy Data. To cope with noisy data, we add SURE-based regularization with weight λ to our training objective (11). In this part, we focus on how the choice of λ affects the regret when the data are noisy. For the sake of simplicity, we only complete the warm-up phase of the algorithm, that is, the models are only trained for 15000 steps with loss function $\mathcal{L}$ and x_ℓ sampled from $X_\ell \mid X_0 = y_0$. In our experiments we note this is generally good enough for noisy data without missing entries.

The results are shown in Figure 9. As we can see, the value of λ has a great influence on the regret achieved with the learned prior. However, finding the most appropriate λ for each problem is a challenging task. Using a larger value of λ helps greatly for the `Labeled Arms` problem when it is given the ground-truth standard deviation $\sigma_{\text{reward}} = 0.1$, but is otherwise harmful for the `Popular and Niche` problem. We believe that finding a way to determine the adequate value of λ will be an important step to make our method more practically relevant.

Training from Incomplete Data. The EM step is mainly designed to tackle missing data. In Figure 10 we show how the induced regrets differ when the models are trained with and without it and when the observations are missing at random but not noisy. To make a fair comparison, we also train the model for a total of 24000 (instead of 15000) steps when EM is not employed. As we can see, in all the setups the use of EM results in lower regret.

Training from Incomplete and Noisy Data. To conclude this section we investigate the effects of EM and SURE-based regularization when the data are both noisy and incomplete. We either drop totally the regularization term, i.e., set $\lambda = 0$, or skip the EM step (but again we train the models for 24000 steps with the configuration of the warm-up phase in this case). We plot the resulting regrets in Figure 11. For the models without EM, the variance calibration algorithm proposed in Appendix C.5 (Algorithm 6) does not work well so we calibrate it with a perfect calibration set $\mathcal{D}_{\text{cal}}$.[12] However, even with this the absence of EM consistently leads to the worst performance. On the other hand, dropping the regularization term only causes clear performance degradation for the `Labeled Arms` problem. This is in line with our results in Figure 9.

F. Additional Experiments

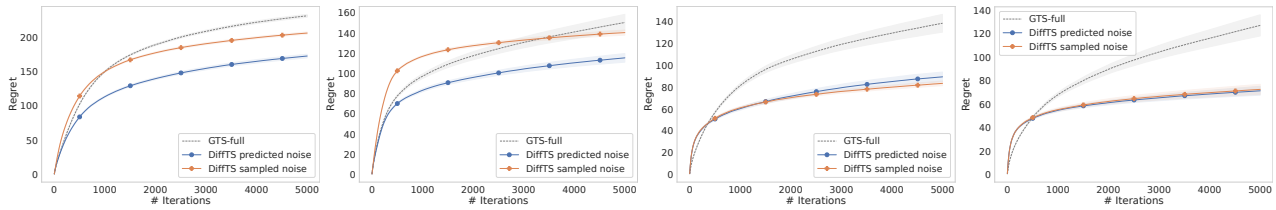
In this appendix, we first supplement our numerical section Section 5 with results obtained under different assumed noise levels. After that, we present additional experiments for the posterior sampling and the training algorithms.

F.1. Experimental Results with Different Assumed Noise Levels

To further validate the benefit of diffusion priors, we conduct experiments for the four problems introduced in Appendix D.1 under different assumed noise levels. The results are shown in Figure 12. We see that DiffTS achieves the smallest regret in 15 out of the 18 plots, confirming again the advantage of using diffusion priors. Moreover, although DiffTS performs worse than either GMMTS or GTS-full in `iPinYou Bidding` and `Labled Arms` for a certain assumed noise level, the smallest regret is still achieved by DiffTS when taking all the noise levels that we have experimented with into account.

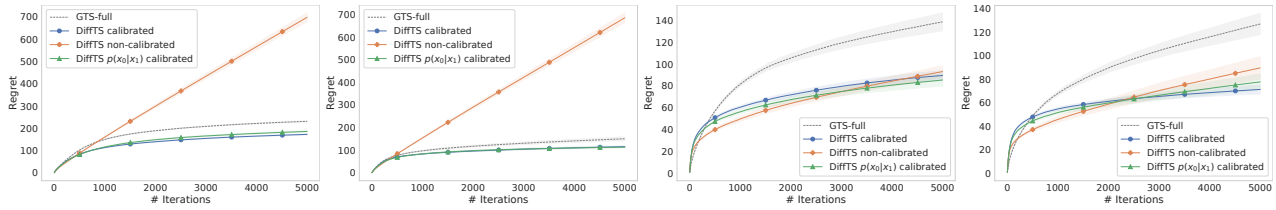
Finally, it is clear from Figure 12 that the choice of the assumed noise level $\hat{\sigma}$ also has a great influence on the induced regret. The problem of choosing an appropriate $\hat{\sigma}$ is however beyond the scope of our work.

[12] Indeed, by design Algorithm 6 only gives good result when the posterior sampling step provides a reasonable approximation of x_0. How to calibrate the variance of a poorly performed model from imperfect data is yet another difficult question to be addressed.



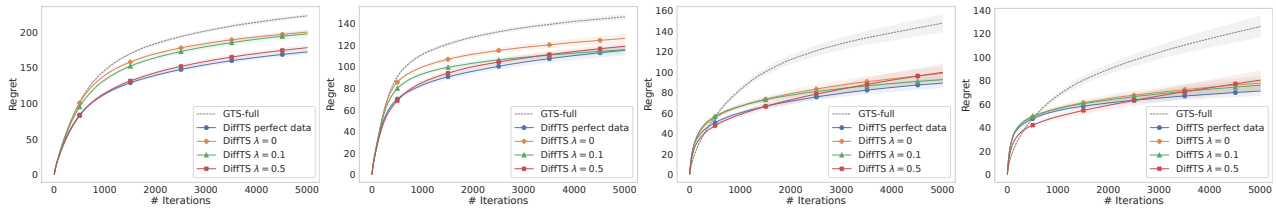
Labeled Arms $\hat{\sigma} = 0.1$ Labeled Arms $\hat{\sigma} = 0.05$ Popular & Niche $\hat{\sigma} = 0.1$ Popular & Niche $\hat{\sigma} = 0.05$

Figure 7. Regret comparison for DiffTS with predicted or independently sampled noise in the construction of diffused observation $\tilde{y}_\ell$.



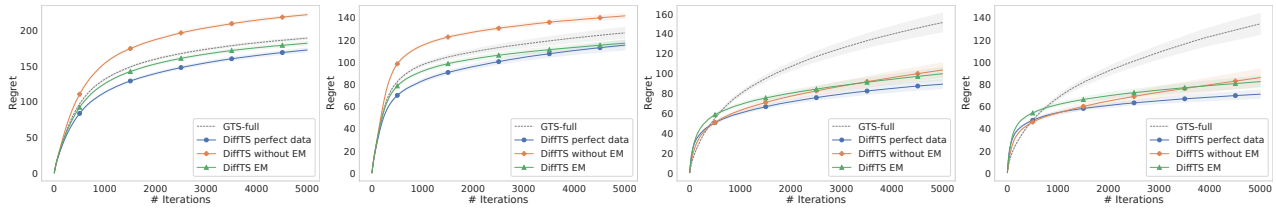
Labeled Arms $\hat{\sigma} = 0.1$ Labeled Arms $\hat{\sigma} = 0.05$ Popular & Niche $\hat{\sigma} = 0.1$ Popular & Niche $\hat{\sigma} = 0.05$

Figure 8. Regret comparison for DiffTS with three different types of reverse variance schedules.



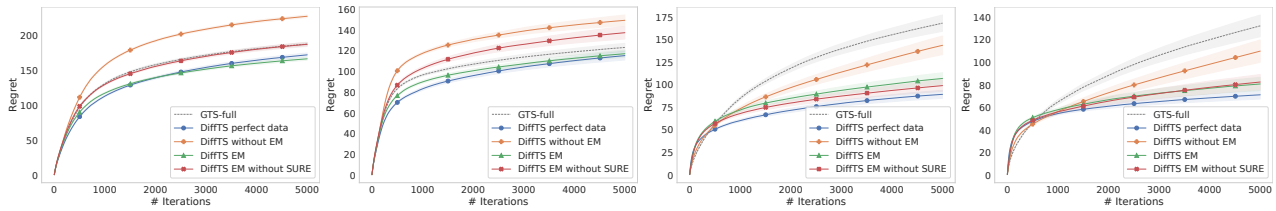
Labeled Arms $\hat{\sigma} = 0.1$ Labeled Arms $\hat{\sigma} = 0.05$ Popular & Niche $\hat{\sigma} = 0.1$ Popular & Niche $\hat{\sigma} = 0.05$

Figure 9. Regret comparison for DiffTS trained on noisy data with different regularization weight λ.



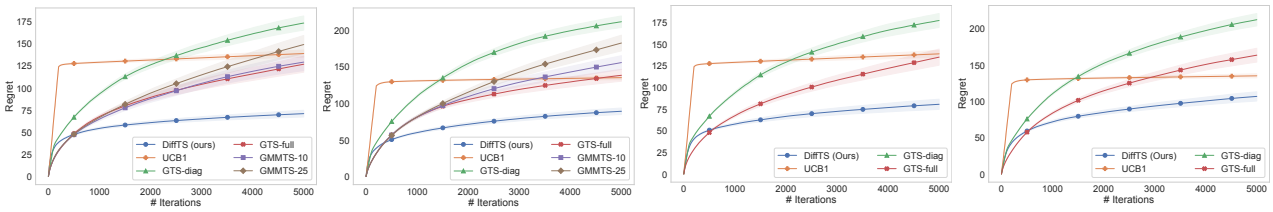
Labeled Arms $\hat{\sigma} = 0.1$ Labeled Arms $\hat{\sigma} = 0.05$ Popular & Niche $\hat{\sigma} = 0.1$ Popular & Niche $\hat{\sigma} = 0.05$

Figure 10. Regret comparison for DiffTS trained on incomplete data with or without EM.



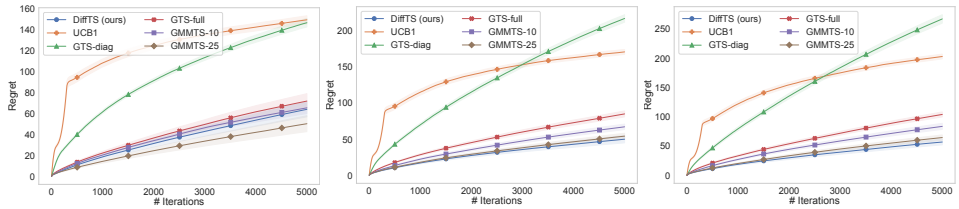
Labeled Arms $\hat{\sigma} = 0.1$ Labeled Arms $\hat{\sigma} = 0.05$ Popular & Niche $\hat{\sigma} = 0.1$ Popular & Niche $\hat{\sigma} = 0.05$

Figure 11. Regret comparison for DiffTS trained on noisy and incomplete data with or without EM / SURE-based regularization.

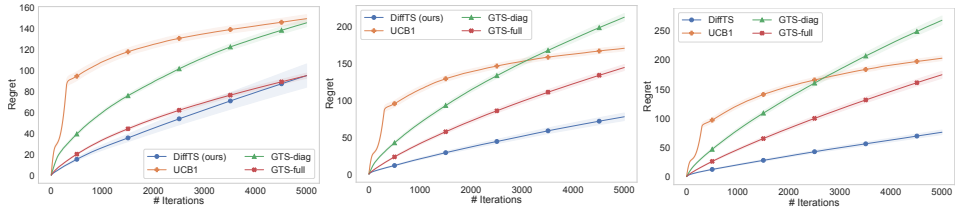


Perfect data $\hat{\sigma} = 0.05$ | Perfect data $\hat{\sigma} = 0.1$ | Imperfect data $\hat{\sigma} = 0.05$ | Imperfect data $\hat{\sigma} = 0.1$

(a) `Popular and Niche`

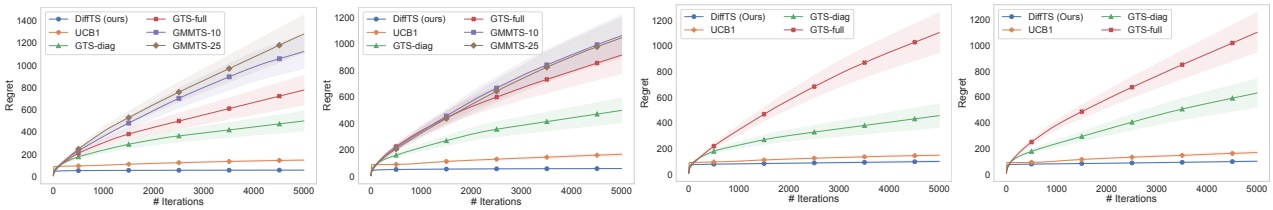


Perfect data $\hat{\sigma} = 0.1$ | Perfect data $\hat{\sigma} = 0.2$ | Perfect data $\hat{\sigma} = 0.3$



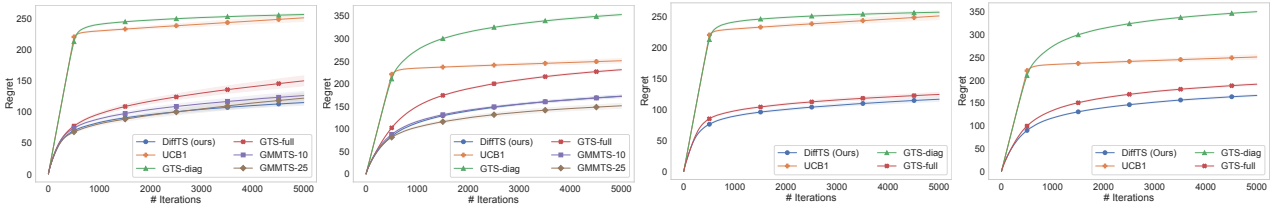
Imperfect data $\hat{\sigma} = 0.1$ | Imperfect data $\hat{\sigma} = 0.2$ | Imerfect data $\hat{\sigma} = 0.3$

(b) `iPinYou Bidding`



Perfect data $\hat{\sigma} = 0.05$ | Perfect data $\hat{\sigma} = 0.1$ | Imperfect data $\hat{\sigma} = 0.05$ | Imperfect data $\hat{\sigma} = 0.1$

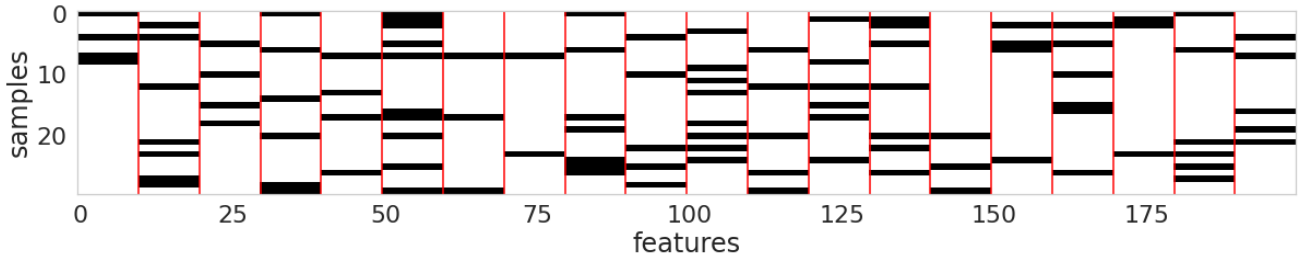
(c) `2D Maze`



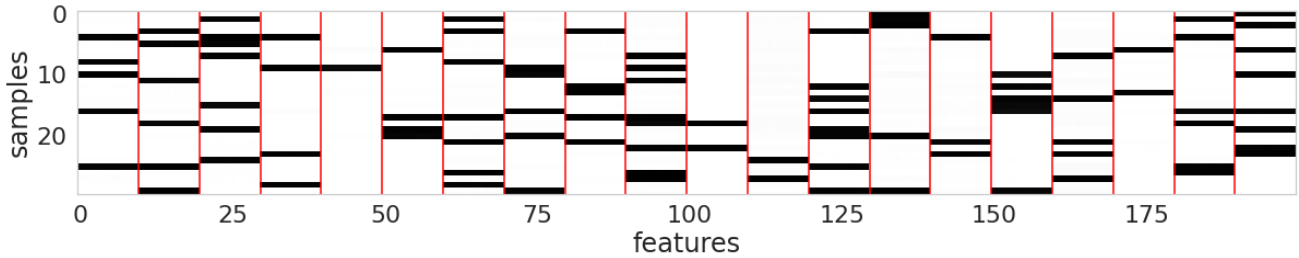
Perfect data $\hat{\sigma} = 0.05$ | Perfect data $\hat{\sigma} = 0.1$ | Imperfect data $\hat{\sigma} = 0.05$ | Imperfect data $\hat{\sigma} = 0.1$

(d) `Labeled Arms`

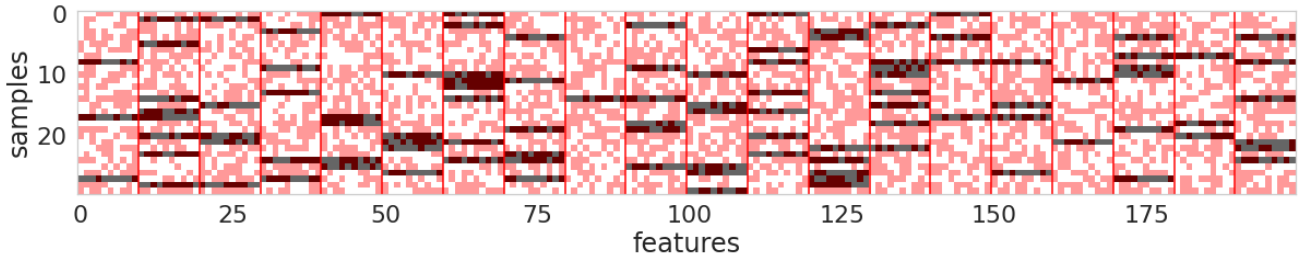
Figure 12. Regret performances on four different problems with priors fitted/trained on either exact expected rewards (perfect data) or partially observed noisy rewards (imperfect data) and with different assumed noise levels $\hat{\sigma}$. The results are averaged over tasks of a test set and shaded areas represent standard errors.



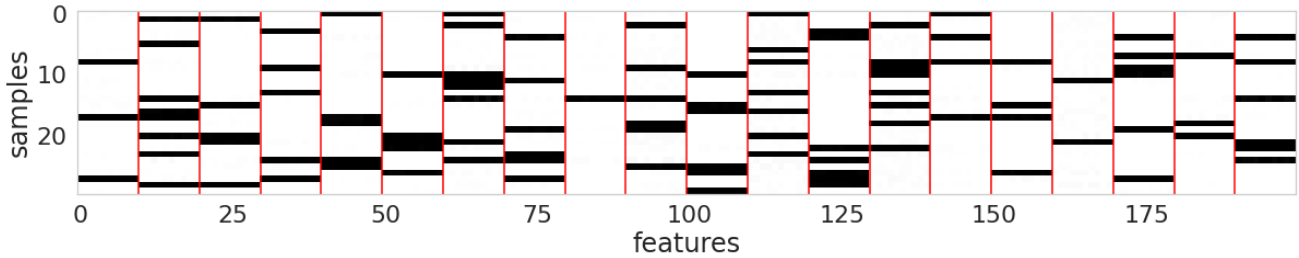
(a) 30 samples from the training set.



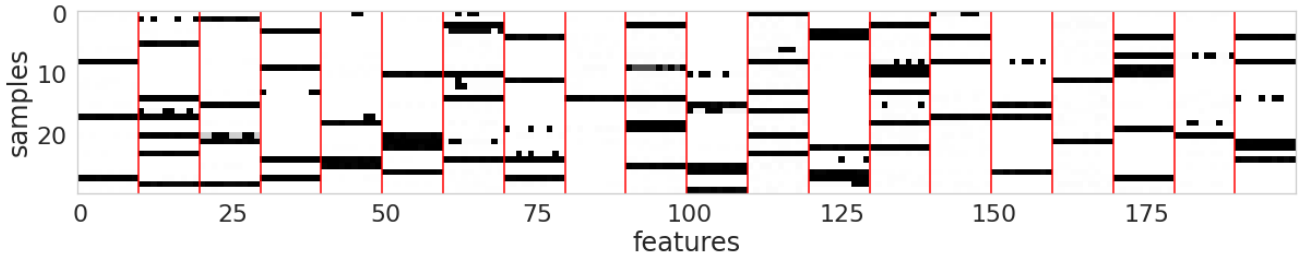
(b) 30 feature vectors generated by the learned diffusion model.



(c) 30 samples from the test set. Red squares indicate missing values.

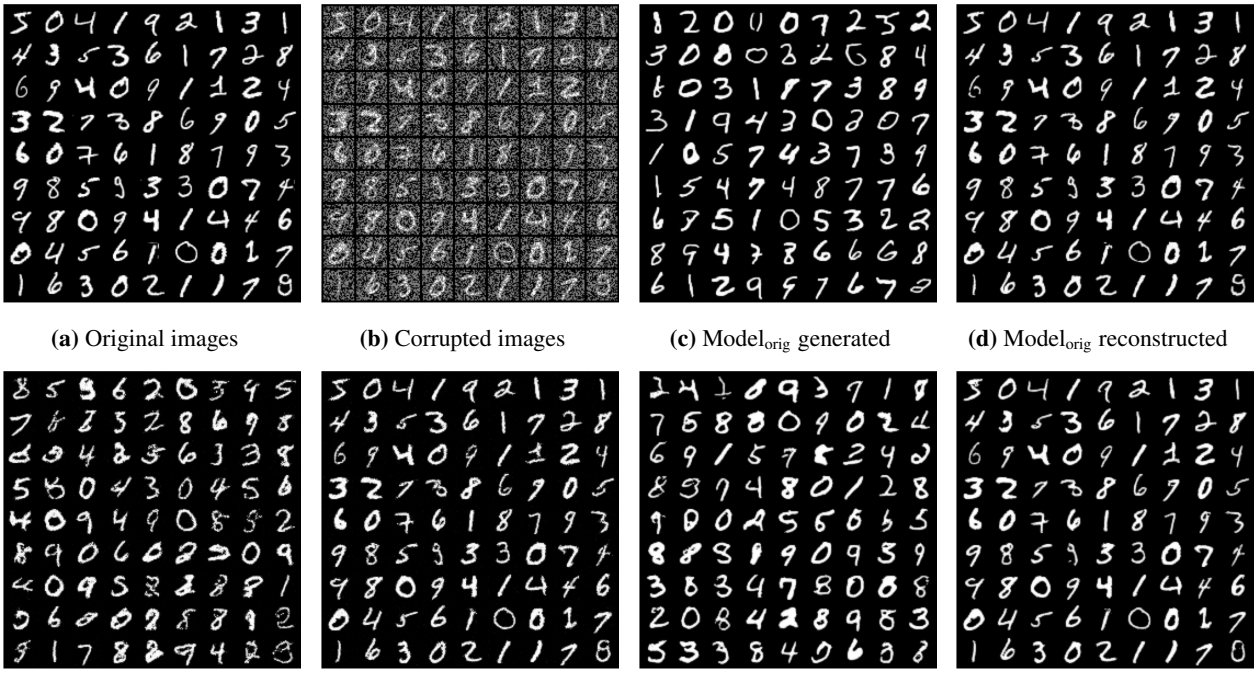


(d) Feature vectors reconstructed with learned diffusion model and Algorithm 2 using predicted noise vectors $\bar{z}_\ell$. The inputs are the ones shown in 13c.



(e) Feature vectors reconstructed with learned diffusion model and Algorithm 2 using independently sampled noise vectors $\tilde{z}_\ell$. The inputs are the ones shown in 13c.

Figure 13. Feature vectors of the toy problems presented in Appendix F.2. Rows and columns correspond respectively to features and samples. For visualization purpose, the features are ordered in a way that those of the same group are put together. The darker the color the higher the value, with white and black representing respectively 0 and 1.



(a) Original images **(b)** Corrupted images **(c)** $\text{Model}_{\text{orig}}$ generated **(d)** $\text{Model}_{\text{orig}}$ reconstructed

(e) $\text{Model}_{\text{cor14}}$ generated **(f)** $\text{Model}_{\text{cor14}}$ reconstructed **(g)** $\text{Model}_{\text{cor16}}$ generated **(h)** $\text{Model}_{\text{cor16}}$ reconstructed

Figure 14. Various images related to the MNIST data set. The three models $\text{Model}_{\text{orig}}$, $\text{Model}_{\text{cor14}}$, and $\text{Model}_{\text{cor16}}$ are respectively trained on the original data set, on the corrupted data set for 14000 steps, and on the corrupted data set for 16000 steps ($\text{Model}_{\text{cor16}}$ is trained on top of $\text{Model}_{\text{cor14}}$ for another 2000 steps; see the text for more details). 'Generated' means unconditional sampling while 'reconstructed' means posterior sampling with Algorithm 2 applied to the corrupted images shown in (b).

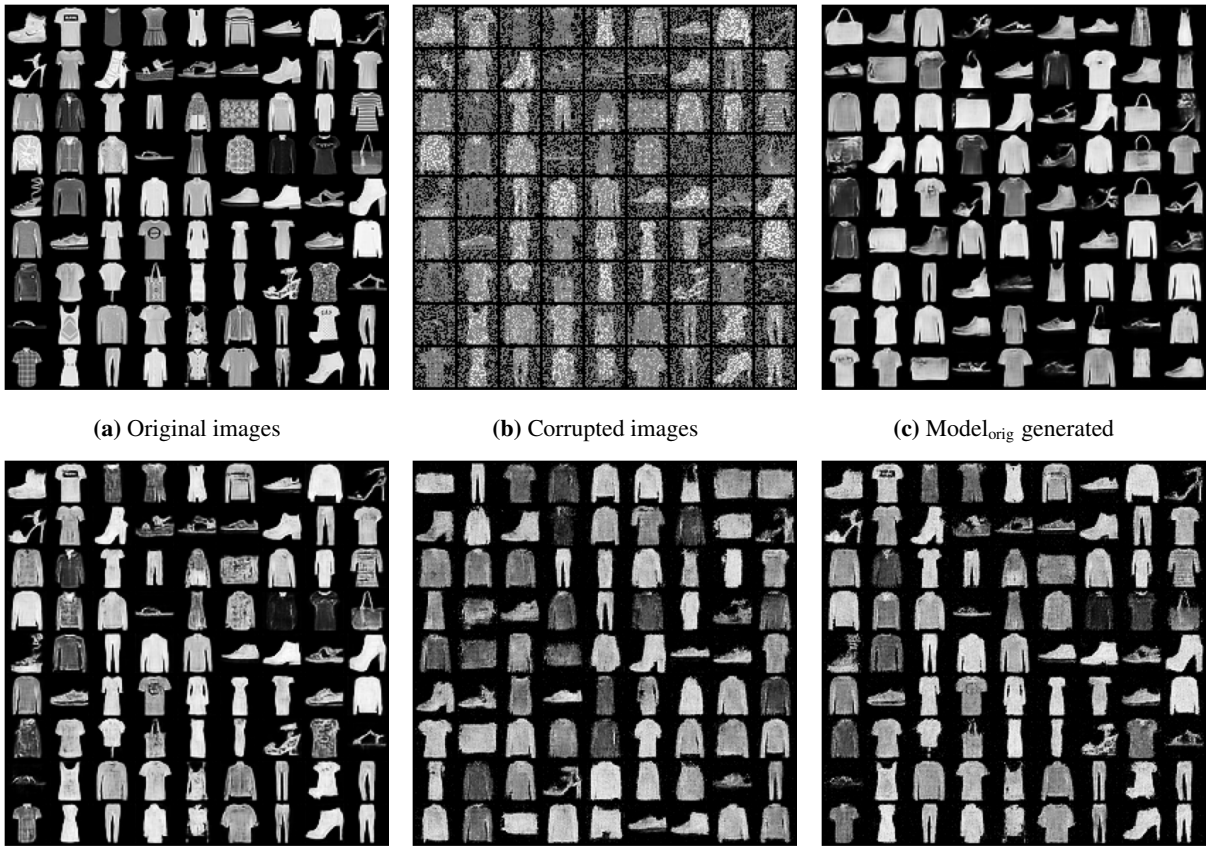
F.2. Comparison of Posterior Sampling Strategies on a Toy Problem

In this part, we demonstrate on a toy problem that using predicted noise $\bar{z}_{\ell+1}$ to construct the diffused observation $\tilde{y}_\ell$ leads to more consistent examples compared to using independently sampled noise vectors.

Data Set and Diffusion Model Training. We consider a simple data distribution over $\mathbb{R}^{200}$. The 200 features are grouped into 20 groups. For each sample, we randomly select up to 6 groups and set the values of the corresponding features to 1. The remaining features take the value 0. Some samples from this distribution are illustrated in Figure 13a. As for the diffusion model, the model architecture, hyper-parameters, and training procedure are taken to be the same as those for the `Popular and Niche` problem (Appendix D). In Figure 13b we see that the data distribution is perfectly learned.

Posterior Sampling. We proceed to investigate the performance of our posterior sampling algorithm on this example. For this, we form a test set of 100 samples drawn from the same distribution and drop each single feature with probability 0.5 as shown in Figure 13c. We then conduct posterior sampling with the learned model using Algorithm 2 (σ is set to 0). To define the diffused observation $\tilde{y}_\ell$, we either follow (8) or replace the predicted noise $\bar{z}_{\ell+1}$ by an independently sampled noise vector $\tilde{z}_{\ell+1} \sim \mathcal{N}(\mathbf{0}_d, I_d)$. The corresponding results are shown in Figures 13d and 13e. As we can see, using predicted noise clearly leads to samples that are more consistent with both the observations and the learned prior.

To provide a quantitative measure, in the constructed samples we define a group to be 'relevant' if the values of all its features are greater than 0.8. We then compute the recall and precision by comparing the ground-truth selected groups and the ones identified as relevant. When predicted noise is used, the average recall and precision are both at 100%. On the other hand, when independently sampled noise is used, the average recall falls to around 85% (this value varies due to the randomness of the sampling procedure but never exceeds 90%) while the average precision remains at around 98%.



(a) Original images **(b)** Corrupted images **(c)** $\text{Model}_{\text{orig}}$ generated

(d) $\text{Model}_{\text{orig}}$ reconstructed **(e)** $\text{Model}_{\text{cor}}$ generated **(f)** $\text{Model}_{\text{cor}}$ reconstructed

Figure 15. Various images related to the Fashion-MNIST data set. The two models $\text{Model}_{\text{orig}}$ and $\text{Model}_{\text{cor}}$ are respectively trained on the original data set and the corrupted data set. 'Generated' means unconditional sampling while 'reconstructed' means posterior sampling with Algorithm 2 applied to the corrupted images shown in (b).

F.3. Training from Imperfect Image Data

To illustrate the potential of the training procedure introduced in Section 4, we further conduct experiments on the MNIST and Fashion-MNIST (Xiao et al., 2017) data sets. Both data sets are composed of gray-scale images of size 28×28. MNIST contains hand-written digits whereas Fashion-MNIST contain fashion items taken from Zalando shopping catalog. Some images of the two data sets are shown in Figures 14a and 15a.

Data Corruption and Experimental Setup. For our experiments, we scale the images to range $[0, 1]$ and corrupt the resulting data with missing rate 0.5 (i.e., each pixel is dropped with 50%) and noise of standard deviation 0.1. As we only use training images, this results in 60000 corrupted images for each of the two data sets. We further separate 1000 images from the 60000 to form the calibration sets. We then train the diffusion models from these corrupted images following Algorithm 4, with $S = 5000$ warm-up steps, $J = 3$ repeats of the EM procedure, and $S' = 3000$ inner steps for each repeat (the total number of training steps is thus 14000). The learning rate and the batch size are respectively fixed at 10^{-4} and 128.

For the regularization term, we take $\lambda = 0.2$ for MNIST and $\lambda = 0.1$ for Fashion-MNIST. The constant ε is set to 10^{-5} as before. As in Ho et al. (2020); Song et al. (2021), we note that the use of exponential moving average (EMA) can lead to better performance. Therefore, we use the EMA model for the posterior sampling step. The EMA rate is 0.995 with an update every 10 training steps. For comparison, we also train diffusion models on the original data sets with the aforementioned learning rate and batch size for 10000 steps. Finally, to examine the influence of the regularization weight λ on the generated images, we consider a third model for MNIST trained on top of the 14000-step model with corrupted data. For this model, we perform an additional posterior sampling step and then train for another 2000 steps with $\lambda = 1$. The remaining details, including the model architecture, are the same as those for the `2D Maze` experiment.

Results. In Figures 14 and 15, we show images from the original data set, from the corrupted data set, and produced by the trained models either by unconditional sampling or data reconstruction with Algorithm 2. Overall, our models manage to generate images that resemble the ones from the original data set without overly sacrificing the diversity.

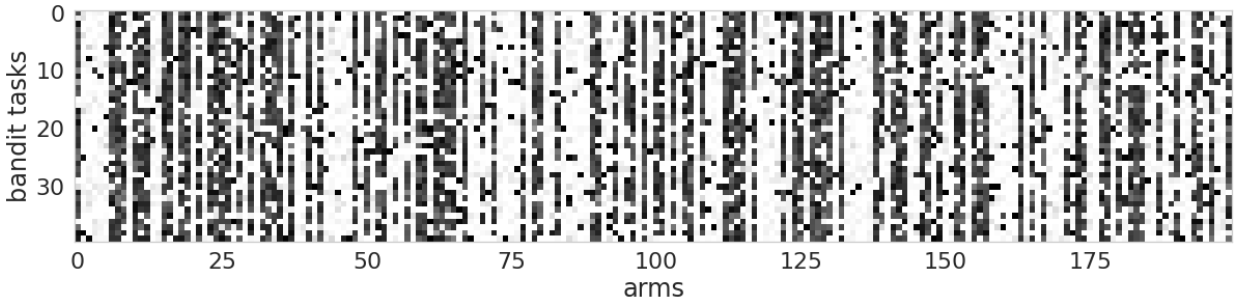
Nonetheless, looking at the samples for Fashion-MNIST we clearly see that a lot of details are lost in the images generated by or reconstructed with diffusion models. In the case of training from perfect data, this can clearly be improved with various modifications to the model including change in model architecture, number of diffusion steps, and/or sampling algorithms (Karras et al., 2022). This would become more challenging in the case of training from imperfect data as the image details can be heavily deteriorated by noise or missing pixels.

On the other hand, the effect of the regularization parameter λ can be clearly seen in the MNIST experiment from Figure 14. Larger λ enables the model to produce digits that are more 'connected' but could cause other artifacts. As in any data generation task, the definition of a good model, and accordingly the appropriate choice of λ, varies according to the context.

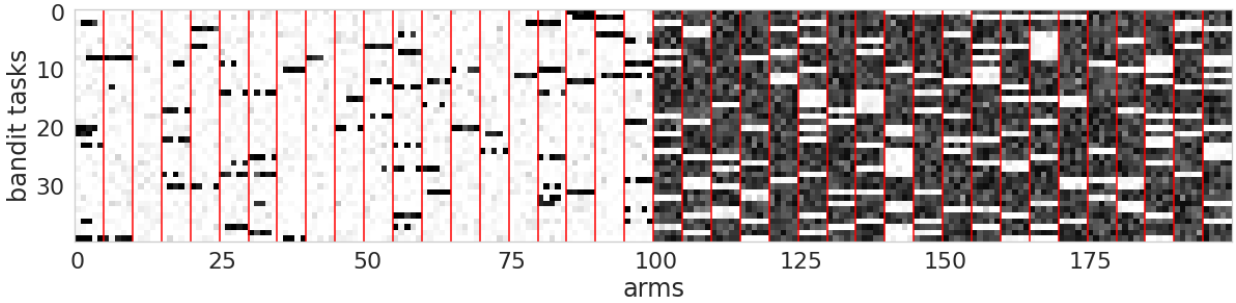
To summarize, we believe that the proposed training procedure has a great potential to be applied in various areas, including training from noisy and incomplete image data, as demonstrated in Figures 14 and 15. However, there is still some way to go in making the algorithm being capable of producing high-equality samples for complex data distribution.

G. Expected Reward Visualization

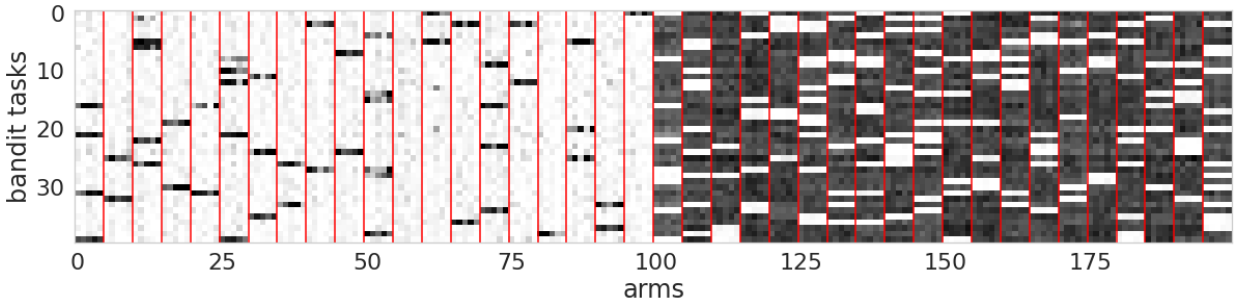
In Figures 16 to 21 we provide various visualizations of the bandit mean reward vectors either of the training sets or generated by the learned priors.



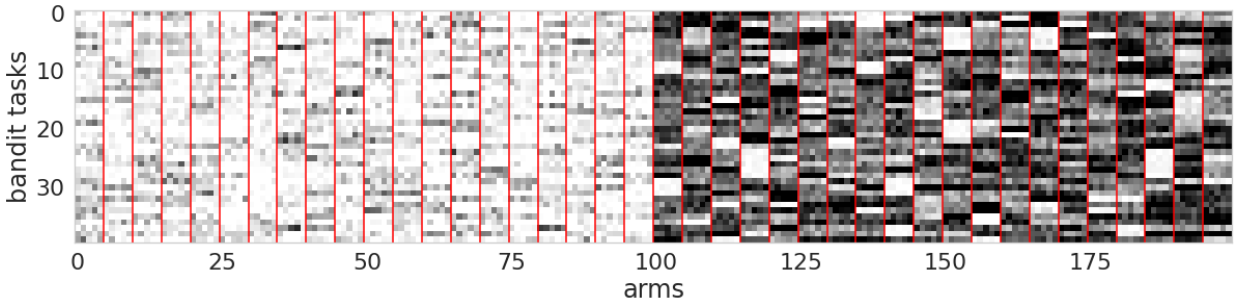
(a) 40 samples from the perfect training set $\mathcal{D}_{\text{tr}}$.



(b) 40 samples from the perfect training set $\mathcal{D}_{\text{tr}}$, reordered to put the arms of the same group together. The popular arms are on the right side of the figure.

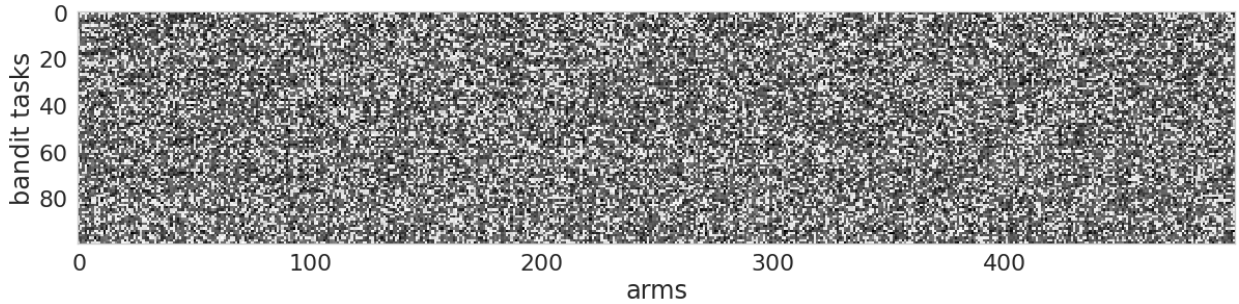


(c) 40 mean reward vectors generated the diffusion model trained on perfect data, reordered to put the arms of the same group together. The popular arms are on the right side of the figure.

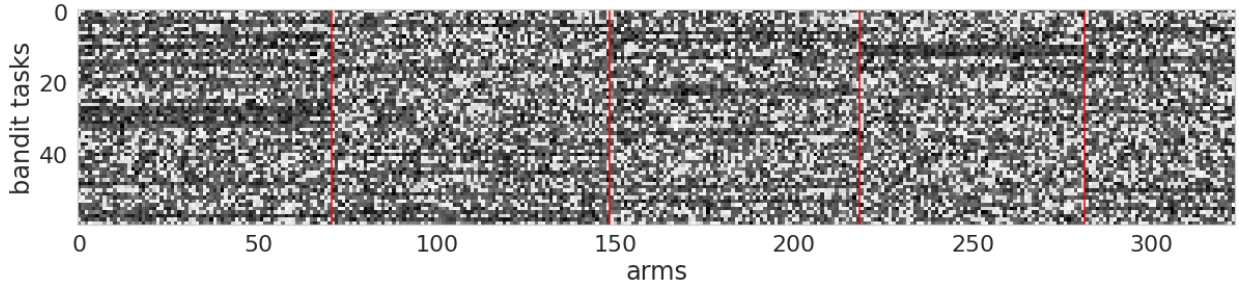


(d) 50 mean reward vectors generated by the 25-component GMM fitted on perfect data, reordered to put the arms of the same group together. The popular arms are on the right side of the figure.

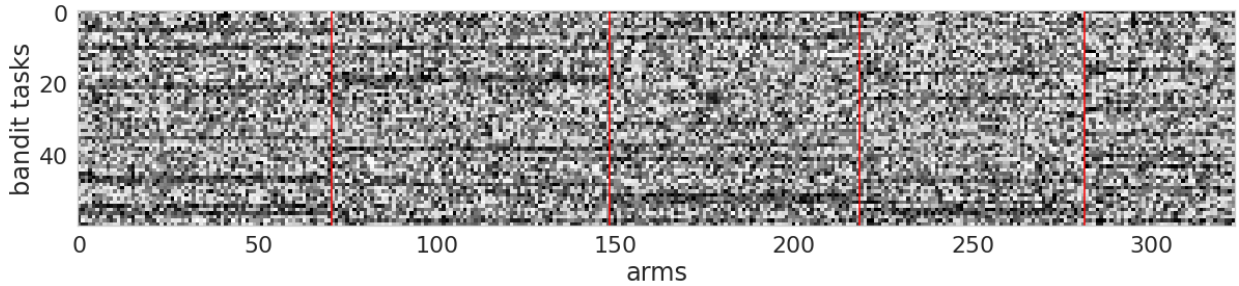
Figure 16. Visualization of the mean reward vectors of the `Popular and Niche` problem. Rows and columns correspond to tasks and arms. The darker the color the higher the value, with white and black representing respectively 0 and 1. Diffusion models manage to learn the underlying patterns that become recognizable by humans only when the arms are grouped in a specific way.



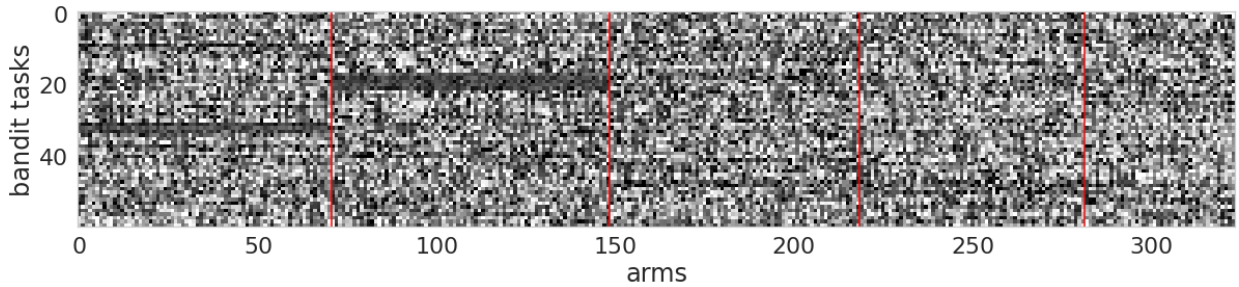
(a) 100 samples from the perfect training set $\mathcal{D}_{\text{tr}}$.



(b) 60 samples from the perfect training set $\mathcal{D}_{\text{tr}}$, grouped by labels and showing only 5 labels. Note that each arm has multiple labels and thus appears in multiple groups.

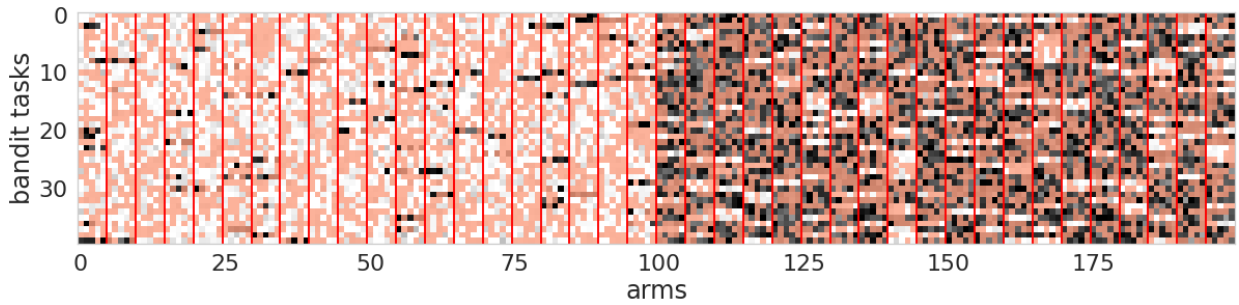


(c) 60 mean reward vectors generated by the diffusion model trained on perfect data, grouped by labels and showing only 5 labels. Note that each arm has multiple labels and thus appears in multiple groups.

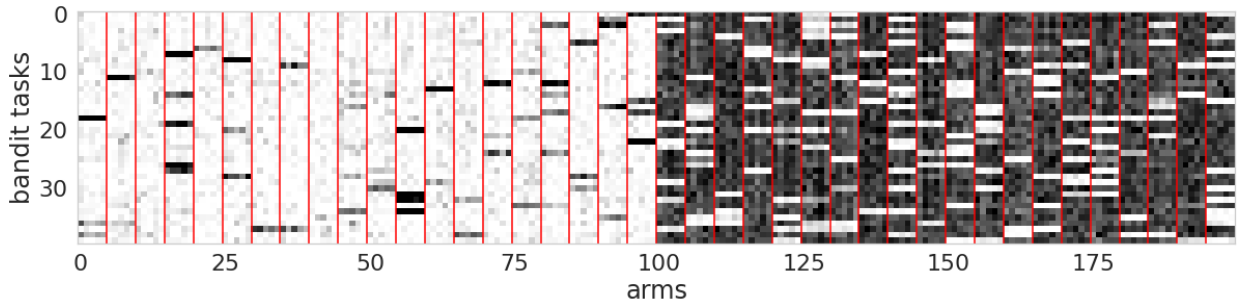


(d) 60 mean reward vectors generated by the 25-component GMM fitted on perfect data, grouped by labels and showing only 5 labels. Note that each arm has multiple labels and thus appears in multiple groups.

Figure 17. Visualization of the mean reward vectors of 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 0 and 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.

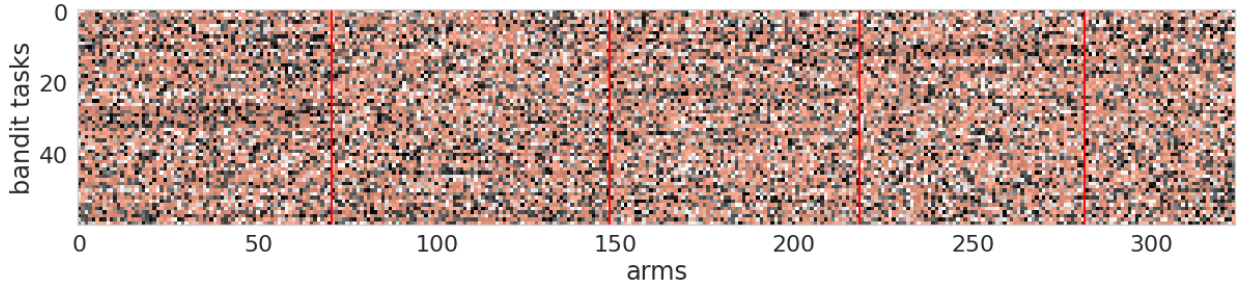


(a) 40 samples from the imperfect training set $\check{\mathcal{D}}_{\text{tr}}$. Red squares indicate missing values.

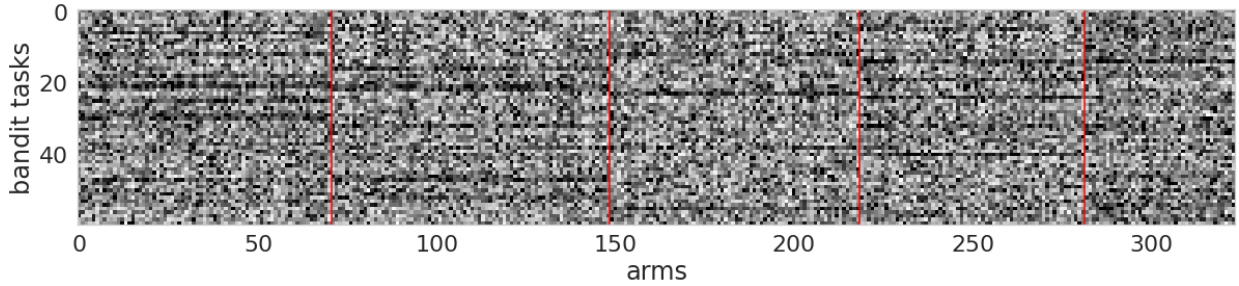


(b) 40 mean reward vectors generated by the diffusion model trained on imperfect data.

Figure 18. Mean reward vectors of the `Popular and Niche` problem. Rows and columns correspond to tasks and arms. For ease of visualization, the arms are reordered so that arms of the same group are put together and popular arms are on the right of the figures. The darker the color the higher the value, with white and black representing respectively 0 and 1.

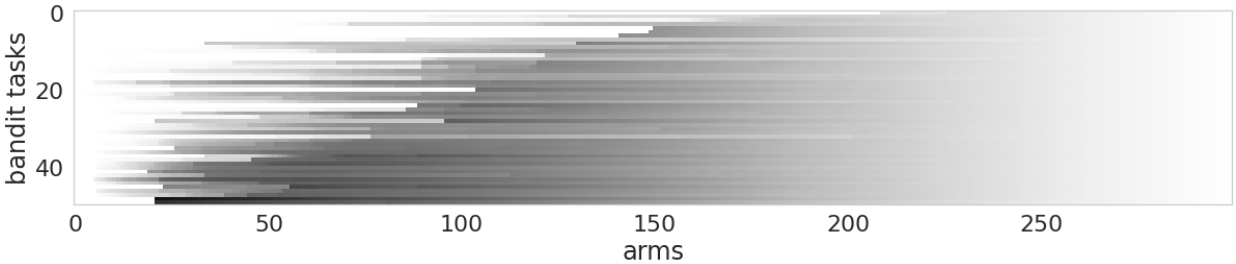


(a) 60 samples from the imperfect training set $\check{\mathcal{D}}_{\text{tr}}$. Red squares indicate missing values.

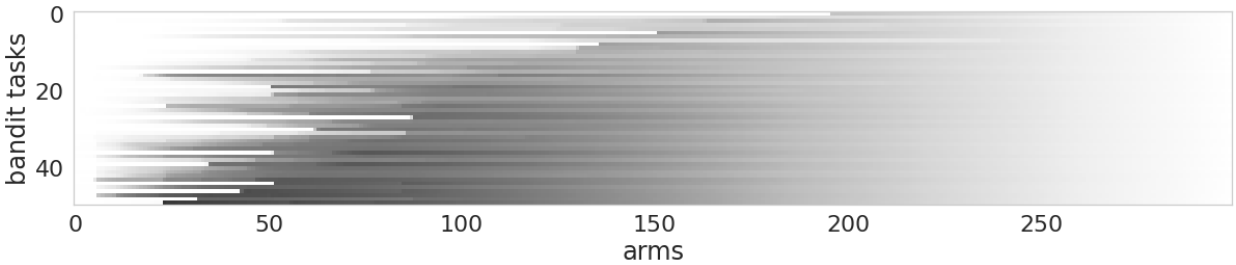


(b) 60 mean reward vectors generated by the diffusion model trained on imperfect data.

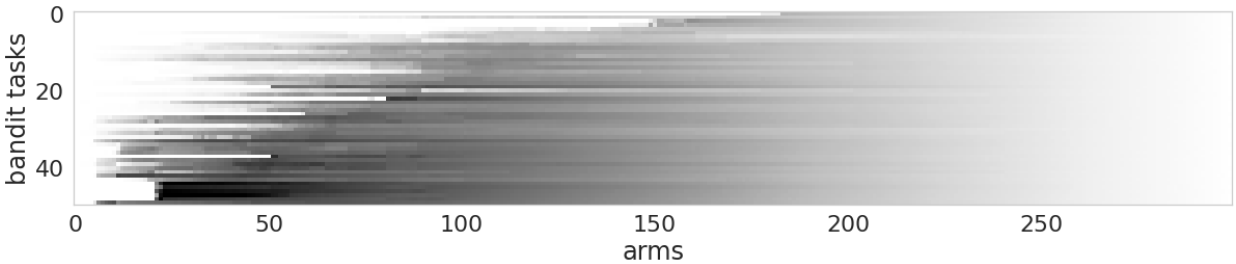
Figure 19. Mean reward vectors of the `Labeled Arms` problem. Rows and columns correspond to tasks and arms. For ease of visualization, the arms are grouped by labels and only arms that are associated to 5 labels are shown. The darker the color the higher the value, with white and black representing respectively 0 and 1.



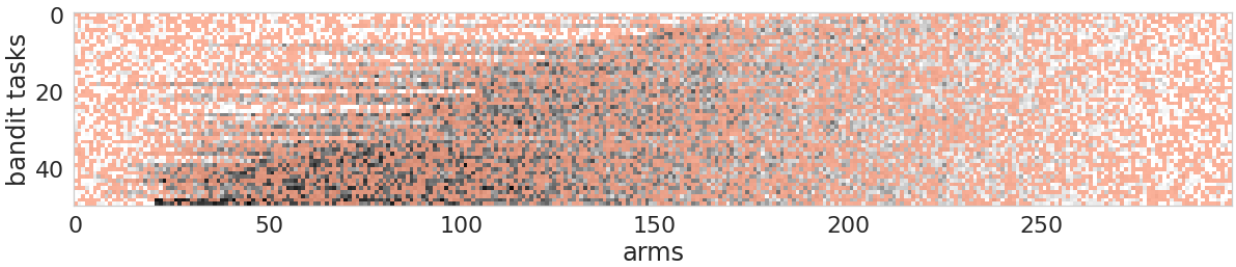
(a) 50 samples from the perfect training set $\mathcal{D}_{\text{tr}}$.



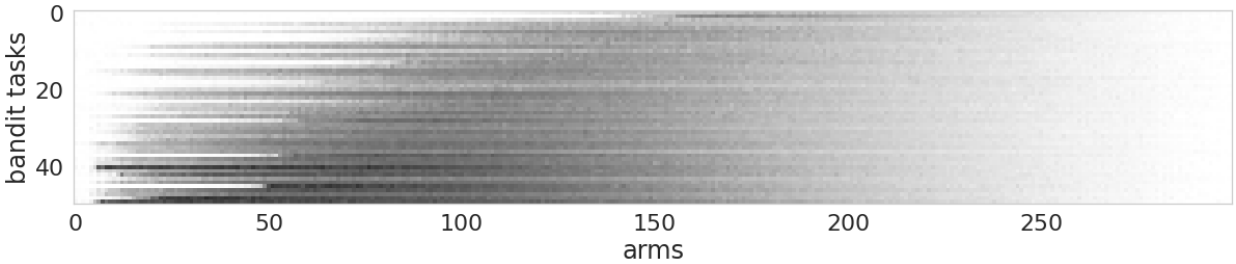
(b) 50 mean reward vectors generated by the diffusion model trained on perfect data.



(c) 50 mean reward vectors generated by the 25-component GMM fitted on perfect data.

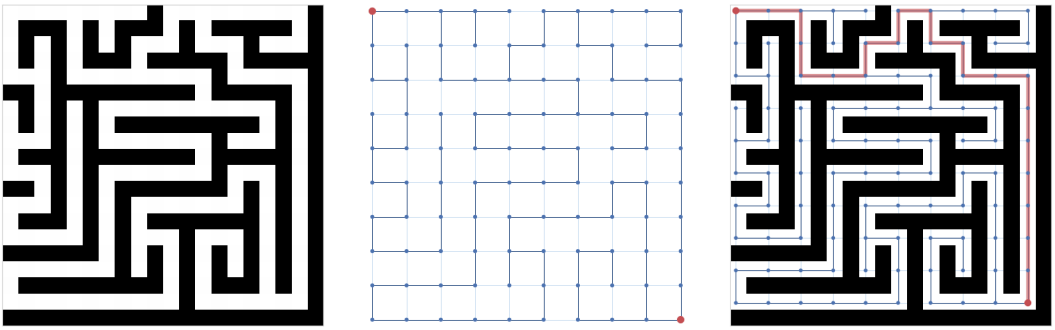


(d) 50 samples from the imperfect training set $\check{\mathcal{D}}_{\text{tr}}$. Red squares indicate missing values.

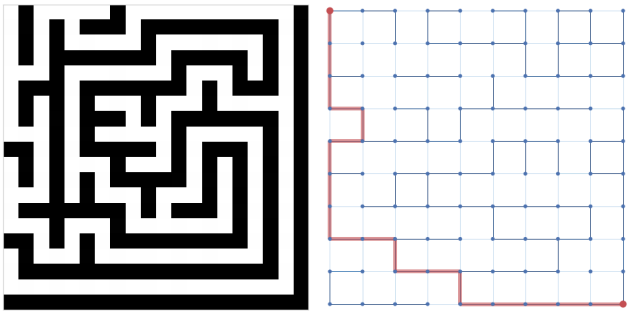


(e) 50 mean reward vectors generated by the diffusion model trained on imperfect data.

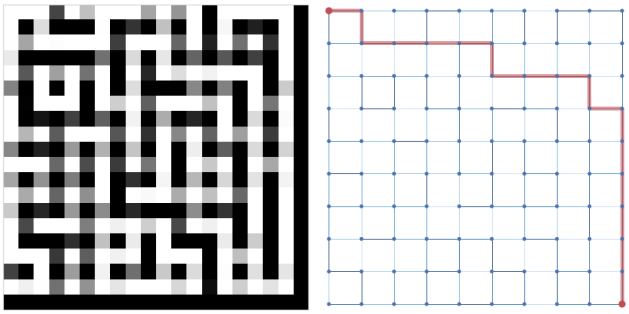
Figure 20. Mean reward vectors of the `iPinYou Bidding` problem. Rows and columns correspond respectively to tasks and arms. For visualization purpose, we order the tasks by the position of their optimal arm. The darker the color the higher the value, with white and black representing respectively 0 and 1.



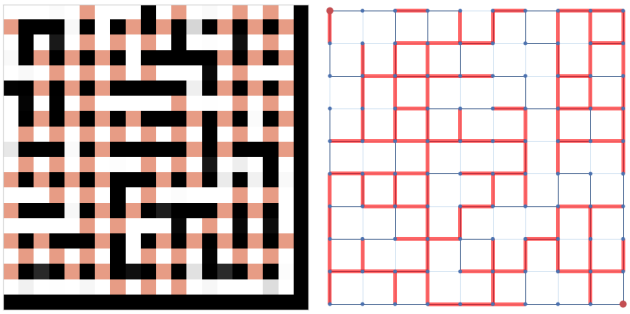
(a) Sample from the perfect training set $\mathcal{D}_{\text{tr}}$.



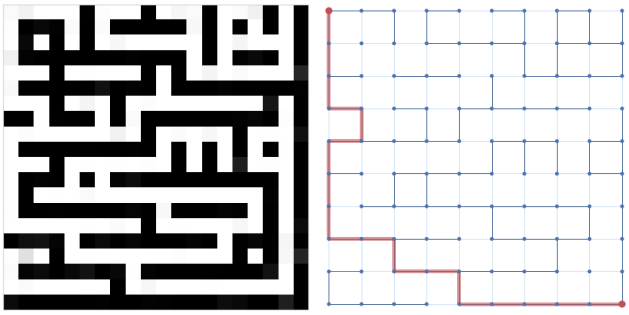
(b) Sample generated by the diffusion model trained on perfect data.



(c) Sample generated by the 25-component GMM fitted on prefect data.



(d) Sample from the imperfect training set $\check{\mathcal{D}}_{\text{tr}}$. Red squares and edges indicate missing values.



(e) Sample generated by the diffusion model trained on imperfect data.

Figure 21. The weighted grid graphs and the corresponding 2D maze representations of the `2D Maze` problem. For visualization, the weights (mean rewards) are first clipped to $[-1, 0]$. Then, for the grid graphs darker the color higher the mean reward (i.e., closer to 0) while for the maze representations it is the opposite. Also note that for the maze representations only a part of the pixels correspond the the edges of the grid graphs, while the remaining pixels are filled with default colors (black or white). The red paths indicate the optimal (super-)arms.