

STRUCTURED DIFFUSION MODELS WITH MIXTURE OF GAUSSIANS AS PRIOR DISTRIBUTION

Anonymous authors

Paper under double-blind review

ABSTRACT

We propose a class of structured diffusion models, in which the prior distribution is chosen as a mixture of Gaussians, rather than a standard Gaussian distribution. The specific mixed Gaussian distribution, as prior, can be chosen to incorporate certain structured information of the data. We develop a simple-to-implement training procedure that smoothly accommodates the use of mixed Gaussian as prior. Theory is provided to quantify the benefits of our proposed models, compared to the classical diffusion models. Numerical experiments with synthetic, image and operational data are conducted to show comparative advantages of our model. Our method is shown to be robust to mis-specifications and in particular suits situations where training resources are limited or faster training in real time is desired.

1 INTRODUCTION

Diffusion models, since Ho et al. (2020); Song et al. (2020b), have soon emerged as a powerful class of generative models to handle the training and generation for various forms of contents, such as image and audio. On top of the success, like many other models, training a diffusion model can require significant computational resources. Compared to more classical generative models such as generative adversarial networks (GAN), the inherent structure of diffusion models requires multiple steps to gradually corrupt structured data into noise and then reverse this process. This necessitates a large number of training steps to successfully denoise, further adding to the computational cost associated with the network and data size.

That said, not all scenarios where diffusion models are used enjoy access to extensive training resources. For example, small-sized or non-profit enterprises with limited budget of compute may desire to train diffusion models with their private data. In those cases with limited resources, the training of standard diffusion models may encounter budget challenges and cannot afford the training of adequate number of steps. In addition, there are scenarios one desires to train in real time with streaming data and aims at achieving certain training performance level as fast as possible given a fixed amount of resources. In such cases, it is preferable to further improve classical diffusion models to achieve faster training.

If training resources are limited, insufficient training can hinder the performance of the diffusion models and result in poorly generated samples. Below in Figure 1 is an illustrative example on gray-scale digits images, showing the performance of denoising diffusion probabilistic models given different training steps. When the model is trained for only 800 steps, it has limited exposure to the data, and as a result, the generated samples are likely to be blurry, incomplete, or show a lack of consistency in terms of digit shapes and structures. The model at this stage has not yet learned to fully reverse the noise process effectively. Our work was motivated by the considerations to improve training efficiency so that, if resources are limited, even with fewer training steps one can achieve certain satisfying level of performance.

In this work, we aim to provide one approach based on adjusting the prior distribution, to improve the performances of classical diffusion models when training resources are limited. Classical diffusion models use Gaussian distribution as the prior distribution, which was designed due to the manifold hypothesis and sampling in low-density areas Song & Ermon (2019). However, this approach does not use the potential structured information within the data and considerably adds to the training complexity. Admittedly, when training resources are not a constraint, or when the data structure is

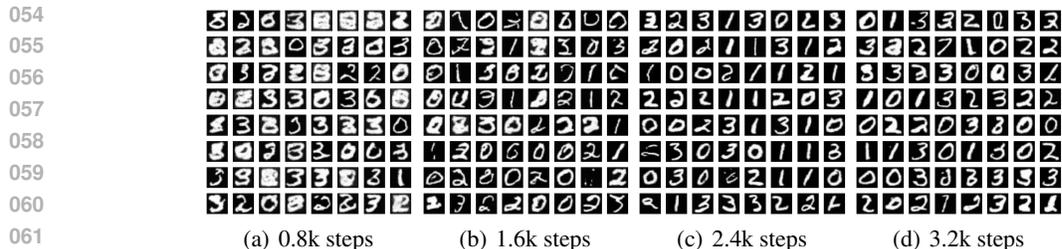


Figure 1: DDPM with varying training steps

difficult to interpret, the use of Gaussian distribution as a prior can be a safe and decent choice. That said, when users have certain structured domain knowledge about the data, say, there might be some clustered groups of data on some dimensions, it can be useful to integrate such information into the training of diffusion models. To increase the ability to incorporate such information, we propose the use of a Gaussian Mixture Model (GMM) as the prior distribution. We develop the associated training process, and examine the comparative performances. The main results of our work are summarized as follows.

1) We propose a class of mixed diffusion models, whose prior distribution is Gaussian mixture model instead of Gaussian noise in the previous diffusion literature. We detail the forward process and the reverse process of the mixed diffusion models, including both the Mixed Denoising Diffusion Probabilistic Models (mixDDPM) and the Mixed Score-based Generative Models (mixSGM) with an auxiliary dispatcher that assigns each data to their corresponding center.

2) We introduce a quantitative metric “Reverse Effort” in the reverse process, which measures the distance between the prior distribution and the finite-sample data distribution under appropriate coupling. With the ‘Reverse Effort’, we explain the benefits of the mixed diffusion models by quantifying the effort-reduction effect, which further substantiates the efficiency of this mixed model.

3) We conduct various numerical experiments among synthesized datasets, operational datasets and image datasets. All numerical results have advocated the efficiency of the mixed diffusion models, especially in the case when training resources are limited.

1.1 RELATED LITERATURE

Diffusion models and analysis. Diffusion models work by modeling the process of data degradation and subsequently reversing this process to generate new samples from noise. The success of diffusion models lies in their ability to generate high-quality, diverse outputs. Their application has expanded across fields such as image and audio synthesis tasks Kong et al. (2020); Dhariwal & Nichol (2021); Leng et al. (2022); Rombach et al. (2022); Yu et al. (2024); Skorokhodov et al. (2024), image editing Meng et al. (2021); Avrahami et al. (2022); Kavar et al. (2023); Mokady et al. (2023), text-to-image generation Saharia et al. (2022); Zhang et al. (2023); Kavar et al. (2023), and other downstream tasks including in medical image generation Khader et al. (2023); Kazerouni et al. (2023) and modeling molecular dynamics Wu & Li (2023); Arts et al. (2023), making them a pivotal innovation in the landscape of generative AI. Tang & Zhao (2024) provide further understanding of score-based diffusion models via stochastic differential equations.

Other methods for efficiency improvement. Various literature have contributed to improve the performance of the diffusion models by proposing more efficient noise schedules Kingma et al. (2021); Karras et al. (2022); Hang & Gu (2024), introducing latent structures Rombach et al. (2022); Kim et al. (2023); Podell et al. (2024); Pernias et al. (2024), improving training efficiency Wang et al. (2023); Haxholli & Lorenzi (2023) and applying faster samplers Song et al. (2020a); Lu et al. (2022a;b); Watson et al. (2022); Zhang & Chen (2023); Zheng et al. (2023); Pandey et al. (2024b); Xue et al. (2024); Zhao et al. (2024); Guo et al. (2024). In addition, Yang et al. (2024) employs a spectrum of neural networks whose sizes are adapted according to the importance of each generative step.

Use of non-Gaussian prior distribution. There exists a series of related but different work on using non-Gaussian noise distributions, to enhance the performance and efficiency of the diffusion models; see Nachmani et al. (2021); Zach et al. (2023); Yen et al. (2023); Bansal et al. (2024); Pandey et al. (2024a), among others. Our work instead emphasizes on the use of structured prior distribution (instead of noise distribution), with the purpose to focus on incorporating data information into the model.

The following of this paper are organized as follows. Section 2 reviews the background of diffusion models, including both Denoising Diffusion Probabilistic Model Ho et al. (2020) and Score-based Generative Models Song et al. (2020b). Section 3 starts from numerical experiments on 1D syntactic datasets to illustrate the motivation of this work. Section 4 details our new models and provide theoretical analysis. Section 5 includes numerical experiments and Section 6 concludes the paper with future directions.

2 BRIEF REVIEW ON DIFFUSION MODELS AND NOTATION

In this section, we briefly review the two classical class of diffusion models: Denoising Diffusion Probabilistic Model (DDPM) Ho et al. (2020) in Section 2.1, and Score-based Generative Models (SGM) Song et al. (2020b) in Section 2.2. Meanwhile, we specify the notation related to the diffusion models and prepare for the description of our proposed methods later in Section 4.

2.1 DENOISING DIFFUSION PROBABILISTIC MODEL

In DDPM, the forward process is modeled as a discrete-time Markov chain with Gaussian transition kernels. This Markov chain starts with the observed data \mathbf{x}_0 , which follows the data distribution p_{data} . The forward process gradually adds noises to \mathbf{x}_0 and forms a finite-time Markov process $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T\}$. The transition density of this Markov chain can be written as

$$\mathbf{x}_t | \mathbf{x}_{t-1} \sim \mathcal{N} \left(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I} \right), \quad t = 1, 2, \dots, T, \quad (1)$$

where β_1, \dots, β_T is called the noise schedule. Then, the marginal density of \mathbf{x}_t conditional on \mathbf{x}_0 can be written in closed-form: $\mathbf{x}_t | \mathbf{x}_0 \sim \mathcal{N} \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I} \right)$, where $\bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s)$ for $t = 1, 2, \dots, T$. The noise schedule is chosen so that $\bar{\alpha}_T$ is closet to 0.

During the training process, a neural network $\epsilon_\theta : \mathbb{R}^d \times \{1, 2, \dots, T\} \rightarrow \mathbb{R}^d$ parameterized by θ is trained to predict the random Gaussian noise ϵ given the time t and the value of the forward process \mathbf{x}_t . The DDPM training objective is proposed as

$$\mathcal{L}^{\text{DDPM}} := \sum_{t=1}^T \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\omega_t \cdot \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2 \right] \text{ with } \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + (1 - \bar{\alpha}_t) \epsilon \quad (2)$$

where $\omega_1, \omega_2, \dots, \omega_T$ is a sequence of weights and $\|\cdot\|_2$ is the l_2 metric.

Based on the trained neural network, the reverse sampling process is also modeled as a discrete-time process with Gaussian transition kernels. Here and throughout what follows, we use $\tilde{\mathbf{x}}$ to denote the reverse process. The reverse process starts from the prior distribution $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the transition density is given by

$$\tilde{\mathbf{x}}_{t-1} | \tilde{\mathbf{x}}_t \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_t, t), \beta_t \mathbf{I}) \text{ with } \boldsymbol{\mu}_\theta(\mathbf{x}, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(\mathbf{x} - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}, t) \right) \quad (3)$$

for $t = 1, 2, \dots, T$. The final result $\tilde{\mathbf{x}}_0$ is considered to be the output of the DDPM and its distribution is used to approximate the data distribution p_{data} .

2.2 SCORE-BASED GENERATIVE MODELS

Both of the forward process and the reverse process of the SGM are modeled by Stochastic Differential Equations (SDE). The forward SDE starts from $\mathbf{x}_0 \sim p_{\text{data}}$ and evolves according to

$$d\mathbf{x}_t = f_t \mathbf{x}_t dt + g_t d\mathbf{w}_t, \quad (4)$$

where f_t is the drift scalar function, g_t is the diffusion scalar function and \mathbf{w}_t is a d-dimensional standard Brownian motion. In particular, the forward SDE is an Ornstein–Uhlenbeck (OU) process and the marginal distribution has a closed-form Gaussian representation. Without loss of generality, we suppose $\mathbf{x}_t \sim \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$, where α_t and σ_t are solely determined by the scalar functions f_t, g_t .

According to Anderson (1982), the reverse of diffusion process (4) is also a diffusion process and can be represented as

$$d\tilde{\mathbf{x}}_t = (f_t \tilde{\mathbf{x}}_t - g_t^2 \nabla_{\mathbf{x}} \log p_t(\tilde{\mathbf{x}}_t)) dt + g_t d\tilde{\mathbf{w}}_t, \quad \tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I}), \quad (5)$$

where dt is an infinitesimal negative time step and $\tilde{d}w$ is the standard Brownian motion when time flows back from T to 0. Besides, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is called the score function and is approximated by a trained neural network s_θ . However, later researchers have substituted the score function by the noise model Lu et al. (2022a) and the prediction model Lu et al. (2022b) to improve the overall efficiency of the SGM. To keep align with the DDPM, we only introduce the noise model here.

Instead of learning the score function directly, the noise model utilizes a neural network $\epsilon_\theta(\mathbf{x}, t) : \mathbb{R}^d \times (0, T] \rightarrow \mathbb{R}^d$ to learn the scaled score function $-\sigma_t \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. According to Lu et al. (2022a), the training objective is elected to be

$$\mathcal{L}^{\text{SGM}} := \int_0^T \omega_t \cdot \mathbb{E}_{\mathbf{x}_0, \epsilon} [\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\|_2^2] dt \quad \text{with} \quad \mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon, \quad (6)$$

where ω_t is a weighting function. Having the trained noise model ϵ_θ , the previous reverse SDE (5) can be re-formalized as

$$d\tilde{\mathbf{x}}_t = \left(f_t \tilde{\mathbf{x}}_t + \frac{g_t^2}{\sigma_t} \epsilon_\theta(\tilde{\mathbf{x}}_t, t) \right) dt + g_t d\tilde{\mathbf{w}}_t, \quad \tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I}). \quad (7)$$

Various numerical SDE solvers can be applied on (7) to obtain the final output $\tilde{\mathbf{x}}_0$.

3 ILLUSTRATION WITH ONE-DIMENSIONAL EXAMPLES

In this section, we provide a brief numerical illustration to show the performance comparison between DDPM and mixDDPM (the method that will be formally introduced in the next section). We illustrate through two 1-dimensional experiments. The true data distribution for the first experiment is a standardized Gaussian mixture distribution with symmetric clusters $p_{\text{data}} = \frac{1}{2}(\mathcal{N}(-0.9, 0.19) + \mathcal{N}(0.9, 0.19))$. The second experiment chooses a Gamma mixture distribution as the data distribution that shares the same cluster mean and variance with the above-mentioned Gaussian mixture distribution.

Table 1: DDPM v.s. mixDDPM on 1D Gaussian Mixture Model

	DDPM	mixDDPM
\mathcal{W}_1 distance	0.222	0.113
K-S statistics	0.213	0.073

Table 2: DDPM v.s. mixDDPM on 1D Gamma Mixture Model

	DDPM	mixDDPM
\mathcal{W}_1 distance	0.206	0.136
K-S statistics	0.232	0.103

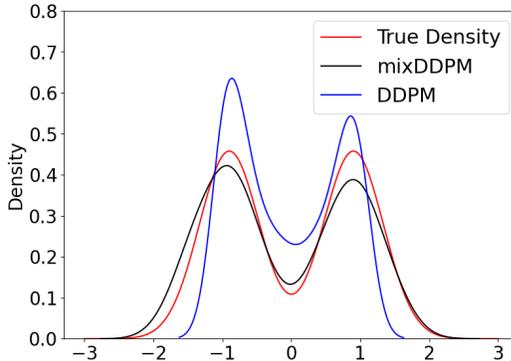


Figure 2: DDPM and mixDDPM on 1D Bimodal Gaussian Mixture Model

We present the results in Table 1 and Table 2. The experiment setting is as follows. The training dataset includes 256 samples from the data distribution. The classical DDPM involves 16k steps.

We also implement the mixDDPM, which is our new model and will be introduced in Section 4, on the training dataset with the same neural network architecture, the same size of network parameters, the same training steps and the same random seeds. We calculate the \mathcal{W}_1 distance and the Kolmogorov–Smirnov (K-S) statistics Massey Jr (1951); Fasano & Franceschini (1987); Berger & Zhou (2014) between the finite-sample distributions of the generated samples and the true data distribution. In addition, we draw the density of both the finite-sample distributions and the data distribution for the Gaussian mixture experiment in Figure 2.

This illustration of two 1-dimensional examples shows that when training steps are not adequate, the mixDDPM model with a non-Gaussian prior has the potential to achieve significant better performance than the classical DDPM model, by making sure all else is equal.

4 MIXED DIFFUSION MODELS

We propose in this section the mixed diffusion models. Instead of maintaining Gaussian prior distribution, our model chooses the Gaussian mixture model. While still keeping the benefits of Gaussian prior distributions, i.e., the manifold hypothesis and sampling in low density area, the additional parameters enable the model to incorporate more information about the data distribution and further reduce the overall loads of the reverse process. In what follows, we first introduce how to incorporate data information into the prior distribution and an additional dispatcher. We leave more details of the mixDDPM and the mixSGM in Subsection 4.1 and Subsection 4.2, respectively.

In general, the prior distributions in the mixed diffusion models belong to a class of Gaussian mixture distributions with centers $\mathbf{c}_1, \dots, \mathbf{c}_K$. The number of centers K , as well as the specific values of the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$, are predetermined before training and sampling in our model. These parameters can be flexibly chosen by users of the model, either through domain knowledge, or through various analysis methods. There is no need to concern whether the choices are optimal or not. Instead, such choices only need to contain some partial structure information of the data known by users. For instance, users may employ clustering techniques in some particular low-dimensional spaces of the data or use labels on some dimensions of the data. Additional discussions on possible center-selection methods are provided in the Appendix.

Now for each data sample $\mathbf{x}_0 \sim p_{\text{data}}$, a dispatcher $D : \mathbb{R}^d \rightarrow \{1, 2, \dots, K\}$ assigns \mathbf{x}_0 to one of the center. In the context of this work, the dispatcher is defined in the following way:

$$D(\mathbf{x}) = \arg \min_i \{d(\mathbf{x}, \mathbf{c}_i)\}, \quad (8)$$

where $d(\cdot, \cdot)$ is a distance metric. For example, we can set it to be l_2 distance. In other words, the dispatcher assigns the data \mathbf{x}_0 to the nearest center.

4.1 THE MIXDDPM

We first introduce the forward process. Given data sample \mathbf{x}_0 , we suppose the dispatcher assigns it to the j -th center, i.e., $D(\mathbf{x}_0) = j$. Similar to DDPM, the forward process is a discrete-time Markov chain. Conditional on \mathbf{x}_{t-1} and $D(\mathbf{x}_0) = j$, the distribution of \mathbf{x}_t is given by

$$\mathbf{x}_t | \mathbf{x}_{t-1} \sim \mathcal{N} \left(\sqrt{1 - \beta_t} (\mathbf{x}_{t-1} - \mathbf{c}_j) + \mathbf{c}_j, \beta_t \mathbf{I} \right), \quad t = 1, 2, \dots, T. \quad (9)$$

In other words, the process $\mathbf{x}_t - \mathbf{c}_j$ follows the same transition density as (1). As a result, the marginal distribution of \mathbf{x}_t , conditional on \mathbf{x}_0 assigned to the j -th center, is

$$\mathbf{x}_t | \mathbf{x}_0 \sim \mathcal{N} \left(\sqrt{\bar{\alpha}_t} (\mathbf{x}_0 - \mathbf{c}_j) + \mathbf{c}_j, (1 - \bar{\alpha}_t) \mathbf{I} \right), \quad t = 1, 2, \dots, T. \quad (10)$$

As t increases, the distribution of \mathbf{x}_t gradually converges to $\mathcal{N}(\mathbf{c}_j, \mathbf{I})$. That said, the prior distribution conditional on $D(\mathbf{x}_0) = j$ is a Gaussian distribution centered at \mathbf{c}_j with unit variance. Hence, the prior distribution is a Gaussian mixture model that can be represented as $\sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \mathbf{I})$, where p_i is the proportion of data that are assigned to the i -th center.

To learn the noise given the forward process, the mixed DDPM utilizes a neural network $\epsilon_\theta : \mathbb{R}^d \times \{1, 2, \dots, T\} \times \{1, 2, \dots, K\} \rightarrow \mathbb{R}^d$. The neural network takes three inputs: the state of the forward process $\mathbf{x}_t \in \mathbb{R}^d$, the time $t \in \{1, 2, \dots, T\}$ and the center number $D \in \{1, 2, \dots, K\}$.

Our method adopts the U-Net architecture, as suggested by Ho et al. (2020); Rombach et al. (2022); Ramesh et al. (2022). Similar to (2), the mixed DDPM adopts the following training objective:

$$\mathcal{L}_{\text{mix}}^{\text{DDPM}} := \sum_{t=1}^T \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\omega_t \cdot \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, j)\|_2^2 \right], \quad \text{with } \mathbf{x}_t = \sqrt{\bar{\alpha}_t}(\mathbf{x}_0 - \mathbf{c}_j) + \mathbf{c}_j + \sqrt{1 - \bar{\alpha}_t}\epsilon. \quad (11)$$

The training process can be viewed as solving the optimization problem $\min_{\theta} \mathcal{L}_{\text{mix}}^{\text{DDPM}}$ by the stochastic gradient descent method.

During the reverse sampling process, the mixed DDPM first samples $\tilde{\mathbf{x}}_T \sim \sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \mathbf{I})$. To do so, it first samples j from $\{1, 2, \dots, K\}$ such that $\mathbb{P}(j = i) = p_i$. Then, it proceeds to sample $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_j, \mathbf{I})$. The transition density for $\tilde{\mathbf{x}}_{t-1}$, conditional on $\tilde{\mathbf{x}}_t$, is given by

$$\tilde{\mathbf{x}}_{t-1} | \tilde{\mathbf{x}}_t \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_t, t), \beta_t \mathbf{I}), \quad t = 1, 2, \dots, T, \quad (12)$$

where

$$\boldsymbol{\mu}_\theta(\mathbf{x}, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(\mathbf{x} - \mathbf{c}_j - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}, t, j) \right) + \mathbf{c}_j. \quad (13)$$

We summarize the training process and the sampling process for the mixed DDPM in Algorithm 1 and Algorithm 2 below.

Algorithm 1 Training Process for the mixDDPM

Input: samples \mathbf{x}_0 from the data distribution, un-trained neural network ϵ_θ , time horizon T , noise schedule β_1, \dots, β_T , number of centers K and the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$

Output: Trained neural network ϵ_θ

repeat

 Get data \mathbf{x}_0

 Find center $j = D(\mathbf{x}_0)$

 Sample $t \sim U\{1, 2, \dots, T\}$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_t \leftarrow \sqrt{\bar{\alpha}_t}(\mathbf{x}_0 - \mathbf{c}_j) + \mathbf{c}_j + \sqrt{1 - \bar{\alpha}_t}\epsilon$

$\mathcal{L} \leftarrow \omega_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, j)\|_2^2$

 Take a gradient descent step on $\nabla_{\theta} \mathcal{L}$

until Converged or training resource/time limit is hit

Algorithm 2 Reverse Process for the mixDDPM

Input: Trained neural network ϵ_θ , center weights p_1, \dots, p_K , centers $\mathbf{c}_1, \dots, \mathbf{c}_K$

Sample $j \in \{1, \dots, K\}$ with $\mathbb{P}(j = i) = p_i$ for $i = 1, \dots, K$

Sample $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_j, \mathbf{I})$

for $t = T$ to 1 **do**

 Calculate $\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(\tilde{\mathbf{x}}_t - \mathbf{c}_j - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\tilde{\mathbf{x}}_t, t, j) \right) + \mathbf{c}_j$

 Sample $\tilde{\mathbf{x}}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_t, t), \beta_t \mathbf{I})$

end for

Return $\tilde{\mathbf{x}}_0$

Before ending this section, we illustrate why the mixDDPM improves the overall efficiency of the DDPM. We first define the reverse effort for the DDPM and the mixDDPM by

$$\text{ReEff}^{\text{DDPM}} := \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2], \quad (14)$$

$$\text{ReEff}_{\text{mix}}^{\text{DDPM}} := \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_{D(\mathbf{x}_0)}, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2], \quad (15)$$

where \bar{p}_{data} is the empirical distribution over the given data. We now explain the definition of the reverse effort. The forward process gradually adds noise to the initial data \mathbf{x}_0 , until it converges to the prior distribution. On the contrary, the reverse process aims to recover \mathbf{x}_0 given $\tilde{\mathbf{x}}_T$ as input. Hence, we evaluate the distance between \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ and define its expectation as the reverse effort. One noteworthy fact of the reverse effort for the mixDDPM is that \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ are not independent. This can be attributed to the dispatcher, which assigns \mathbf{x}_0 to the $D(\mathbf{x}_0)$ -th center. We present the relationship between the two reverse efforts defined by (14) and (15) in Proposition 1.

Proposition 1. Given the cluster number K and the cluster centers $\mathbf{c}_1, \dots, \mathbf{c}_K$, we define $X_i = \{\mathbf{x} : D(\mathbf{x}) = i\}$ and $p_i = \frac{|X_i|}{\sum_{j=1}^K |X_j|}$ for $i = 1, 2, \dots, K$. Under the assumption that \mathbf{c}_i is the arithmetic mean of X_i , we have

$$\text{ReEff}_{\text{mix}}^{\text{DDPM}} = \text{ReEff}^{\text{DDPM}} - \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2. \quad (16)$$

Proposition 1 shows that the mixDDPM requires less reverse effort compared to the classical DDPM. In addition, this reduction can be quantified as a weighted average of the l_2 -norm of the centers. This reduction can be understood in the following way. We have discussed in Section 3 that the prior distribution of the DDPM contains no information about the data distribution. In contrast, the prior distribution of the mixDDPM retains some data information through the choice of the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$. This retained data information, together with the dispatcher, helps reduce the reverse effort by providing guidance on where to initiate the reverse process. Although this reduction may not significantly affect sampling quality when the neural network is well-trained, it can lead to potential improvements when training is insufficient.

4.2 THE MIXSGM

Suppose a given data \mathbf{x}_0 is assigned to the j -th center by the dispatcher, i.e. $D(\mathbf{x}_0) = j$. The mixed SGM modifies the forward SDE from (4) to

$$d\mathbf{x}_t = f_t(\mathbf{x}_t - \mathbf{c}_j)dt + g_t d\mathbf{w}_t, \quad (17)$$

Equivalently, $\mathbf{x}_t - \mathbf{c}_j$ is the OU-process that follows (4). Then, the marginal distribution of \mathbf{x}_t , conditional on \mathbf{x}_0 and $D(\mathbf{x}_0) = j$, can be calculated as $\mathcal{N}(\alpha_t \mathbf{x}_0 + \mathbf{c}_j, \sigma_t^2 \mathbf{I})$. As the time horizon T increases, the prior distribution, conditional on $D(\mathbf{x}_0) = j$, is $\mathcal{N}(\alpha_T \mathbf{x}_0 + \mathbf{c}_j, \sigma_T^2 \mathbf{I})$, which can be approximated by $\mathcal{N}(\mathbf{c}_j, \sigma_T^2 \mathbf{I})$ if α_T is small enough. Hence, the unconditional prior distribution for the mixed SGM is chosen to be $\sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \sigma_T^2 \mathbf{I})$, where p_i is the proportion of data that are assigned to the i -th center. Below we draw a table to summarize and compare the prior distributions of both classical and mixed diffusion models.

Table 3: Prior distributions

Prior distribution	DDPM	SGM
Classical	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I})$
Mixed (our model)	$\sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \mathbf{I})$	$\sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \sigma_T^2 \mathbf{I})$

Again, we adopt the U-Net architecture to define the noise model $\epsilon_\theta : \mathbb{R}^d \times (0, T) \times \{1, 2, \dots, K\}$. Following (6), the training process can be modeled as solving the following optimization problem by stochastic gradient descent: $\min_{\theta} \mathcal{L}_{\text{mix}}^{\text{SGM}}$, where $\mathcal{L}_{\text{mix}}^{\text{SGM}} = \int_0^T \omega_t \cdot \mathbb{E}_{\mathbf{x}_0, \epsilon} [\|\epsilon_\theta(\mathbf{x}_t, t, j) - \epsilon\|_2^2] dt$ with $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \mathbf{c}_j + \sigma_t \epsilon$ and $j = D(\mathbf{x}_0)$.

Finally, the reverse sampling process can be modeled as both reverse SDE and probability ODE. Similar to what the mixed DDPM has done in Section 4.1, the mixed SGM first samples j from $\{1, 2, \dots, K\}$ according to the weights $\mathbb{P}(j = i) = p_i$ and then samples $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_j, \mathbf{I})$. The corresponding reverse SDE is given by

$$d\tilde{\mathbf{x}}_t = \left(f_t(\tilde{\mathbf{x}}_t - \mathbf{c}_j) + \frac{g_t^2}{\sigma_t} \epsilon_\theta(\tilde{\mathbf{x}}_t, t, j) \right) dt + g_t d\tilde{\mathbf{w}}_t, \quad (18)$$

For ease of exposition, we present the training process and the sampling process for the mixSGM in Appendix C.2. We also present the following Proposition to illustrate the effort-reduction effect of the mixSGM.

Proposition 2. Define

$$\text{ReEff}^{\text{SGM}} := \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2], \quad (19)$$

$$\text{ReEff}_{\text{mix}}^{\text{SGM}} := \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_{D(\mathbf{x}_0)}, \sigma_T^2 \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2], \quad (20)$$

where \bar{p}_{data} is the empirical distribution over the given data. Given the cluster number K and the cluster centers $\mathbf{c}_1, \dots, \mathbf{c}_K$, we define $X_i = \{\mathbf{x} : D(\mathbf{x}) = i\}$ and $p_i = \frac{|X_i|}{\sum_{j=1}^K |X_j|}$ for $i = 1, 2, \dots, K$. Under the assumption that \mathbf{c}_i is the arithmetic mean of X_i , we have

$$\text{ReEff}_{\text{mix}}^{\text{SGM}} = \text{ReEff}^{\text{SGM}} - \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2. \tag{21}$$

Proposition 2 provides a quantitative measurement of efforts reduction brought by the mixSGM, compared to the classical SGM. The amount of effort reduction reflects the amount of information provided by the structured prior distribution. One insight shown by Proposition 2 is that the effect of the effort reduction depends on σ_T , the standard deviation of the prior distribution in SGM. When σ_T is very large, the impact of the reduction term $\sum_{i=1}^K p_i \|\mathbf{c}_i\|^2$ is minimal because both of the reverse efforts for the SGM and the mixSGM become significantly large. On the contrary, when σ_T is moderate, the reduction effect becomes evident.

5 NUMERICAL EXPERIMENTS

5.1 OAKLAND CALL CENTER & PUBLIC WORK SERVICE REQUESTS DATASET

The Oakland Call Center & Public Works Service Requests Dataset is an open-source dataset containing service requests received by the Oakland Call Center. We preprocess the dataset to obtain the number of daily calls from July 1, 2009, to December 5, 2019. To learn the distribution of daily calls, we extract the number of daily calls from the 1,000th to the 2,279th day (a total of 1,280 days) since July 5, 2009, as the training data, and set the number of daily calls from the 2,280th to the 2,919th day (a total of 640 days) as the testing data. Since operational datasets often exhibit non-stationarity in terms of varying means, variances, and increasing (or decreasing) trends, we first conduct linear regression on the training data to eliminate potential trends and then normalize the data. We compare the effects of DDPM with mixDDPM. As the training data is one-dimensional, we utilize fully connected neural networks with the same architecture and an equal number of neurons. We train both models for 8k steps and independently generate 640 samples.

In Figure 3, we plot the density of the training data and the data generated by both DDPM and mixDDPM. We also calculate the \mathcal{W}_1 distances and the K-S statistics between the generated samples and the testing data, as shown in Table 4. The benchmark column is calculated by comparing the training data to the generated data, serving as a measurement of the distributional distances between the training and testing data. Relative errors are calculated as the difference between the metric values of the benchmark and the models, expressed as a fraction of the benchmark’s metric value.

Table 4: DDPM v.s. mixDDPM on Oakland Call Center Datasets

	Benchmark	DDPM	mixDDPM
\mathcal{W}_1 Distance	0.172	0.374	0.170
\mathcal{W}_1 relative error		1.174	-0.012
K-S statistics	0.112	0.277	0.105
K-S relative error		1.473	-0.063

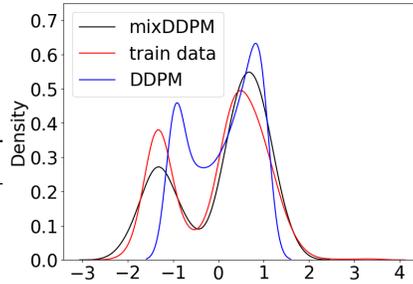


Figure 3: DDPM and mixDDPM on Oakland Call Center Dataset

5.2 EXPERIMENTS ON EMNIST

In this section, we compare mixDDPM with DDPM using the EMNIST dataset Cohen et al. (2017), an extended version of MNIST that includes handwritten digits and characters in the format of $1 \times 28 \times 28$. We extract the first N images of digits 0, 1, 2, and 3 to form the training dataset,

with N values set to 64, 128, and 256. We select U-Net as the model architecture to learn the noise during training. As an illustrative example, we present the generated samples for $N = 128$ in Figure 4 below.

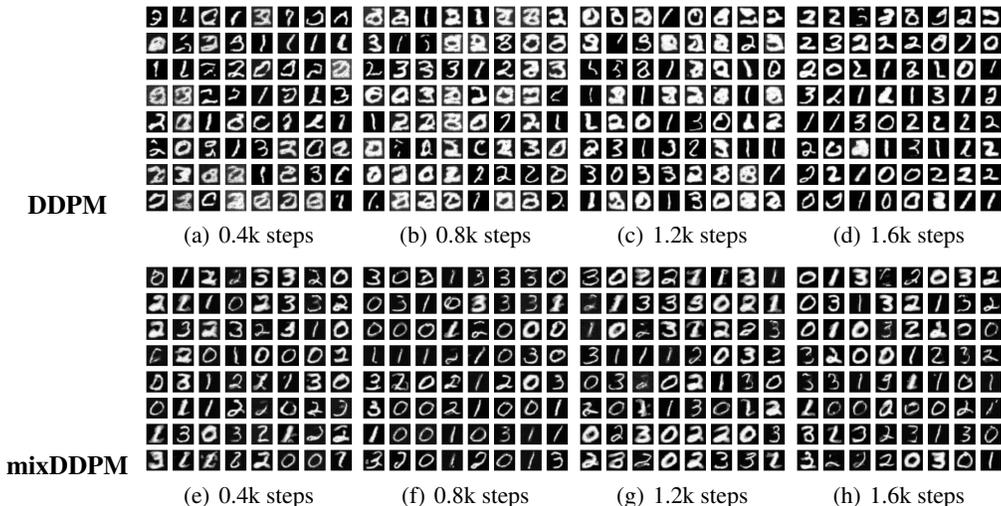


Figure 4: EMNIST Experiments with N=128

When training resources are limited, i.e., the number of training steps is relatively small, mixDDPM performs better than DDPM. Specifically, when the training step count is 0.4k, approximately one-quarter of the images generated by DDPM are difficult to identify visually, whereas only 10% of the images generated by mixDDPM are hard to identify. As the training step count increases to 1.6k, the sample quality of both DDPM and mixDDPM becomes visually comparable. This observation suggests that mixDDPM significantly improves the visual quality of the samples compared to DDPM when training resources are constrained. More experimental results, including variations in the size of the training data and the number of training steps, can be found in Appendix E.3. In addition, experiments for SGM and mixSGM can be found in Appendix E.3.

5.3 EXPERIMENTS ON CIFAR10

We test our model on CIFAR10, a dataset consisting of images with dimensions of $3 \times 32 \times 32$. We extract the first 2,560 images from three categories: dog, cat, and truck. These 7,680 images are fixed as the training data. During training, we use the same model architecture and noise schedule for both DDPM vs. mixDDPM and SGM vs. mixSGM to minimize the influence of other variables. We present the Fréchet Inception Distance (FID) Heusel et al. (2017) for the generated samples and the improvement ratio (Impr. Ratio) in Table 5 and Table 6. The improvement ratio is calculated as the difference between the FID for DDPM/SGM and the FID for mixDDPM/mixSGM, expressed as a fraction of the FID for DDPM/SGM. Additionally, we provide a comparison of generated samples in Figure 5.

Table 5: DDPM v.s. mixDDPM on CIFAR10

Model \ Training Steps	180k	240k	300k	360k	420k	480k	540k	600k
DDPM	71.97	49.11	44.52	38.30	41.34	34.83	28.61	33.04
mixDDPM	35.84	23.43	20.78	18.15	16.43	13.82	14.80	12.88
Impr. Ratio	0.50	0.52	0.53	0.47	0.60	0.60	0.48	0.61

Table 6: SGM v.s. mixSGM on CIFAR10

Model \ Training Steps	180k	240k	300k	360k	420k	480k	540k	600k
SGM	65.82	45.55	49.94	35.22	34.88	24.58	28.42	20.46
mixSGM	62.41	40.52	36.38	22.66	24.25	16.93	17.81	21.89
Impr. Ratio	0.05	0.11	0.27	0.36	0.30	0.31	0.37	-0.07

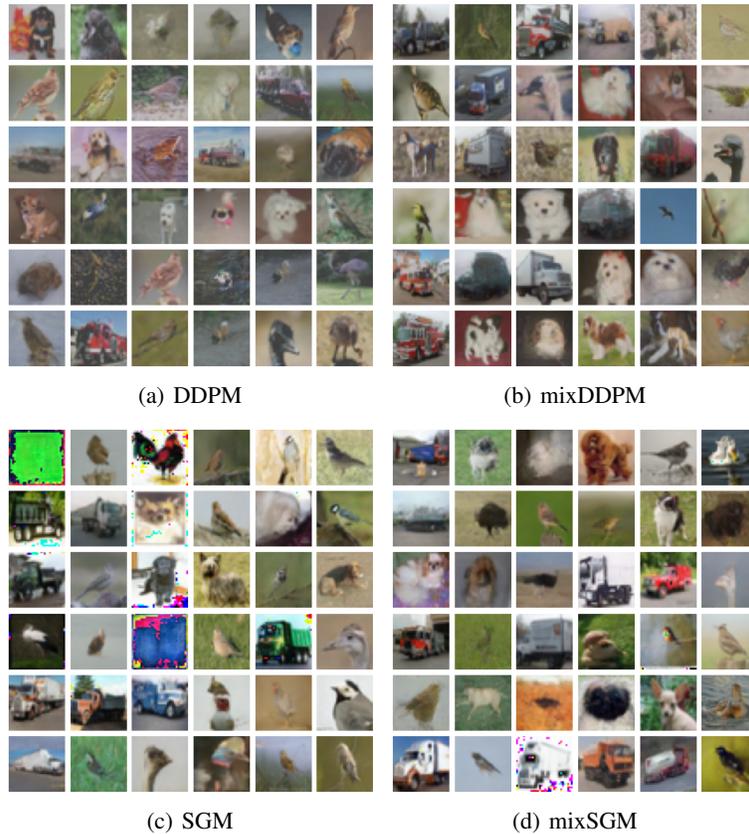


Figure 5: Experiments on CIFAR10 with 480k Training Steps

The results on CIFAR10 demonstrate that mixed diffusion models with Gaussian mixture priors generally achieve smaller FID scores (approximately 60% lower for mixDDPM and 3% lower for mixSGM) and better sample quality. The reduced FID and improved sample quality can be attributed to the utilization of the data distribution. By identifying suitable centers for the data distribution, the reverse process can begin from these centers instead of the zero point, thereby reducing the effort required during the reverse process. This leads to the improvements observed in the numerical results. Further implementation details and additional experimental results are provided in Appendix C.4 and E.4, respectively.

6 CONCLUSION

In this work, we propose and theoretically analyze a class of mixed diffusion models, where the prior distribution is chosen as mixed Gaussian distribution. The goal is to allow users to flexibly incorporate structured information or domain knowledge of the data into the prior distribution. The proposed model is shown to have advantageous comparative performance particularly when the training resources are limited. For future work, we plan to further the theoretical analysis and examine the performance of mixed diffusion models with data of different modalities.

REFERENCES

- 540
541
542 Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Ap-*
543 *plications*, 12(3):313–326, 1982.
- 544 Marloes Arts, Victor Garcia Satorras, Chin-Wei Huang, Daniel Zugner, Marco Federici, Cecilia
545 Clementi, Frank Noé, Robert Pinsler, and Rianne van den Berg. Two for one: Diffusion mod-
546 els and force fields for coarse-grained molecular dynamics. *Journal of Chemical Theory and*
547 *Computation*, 19(18):6151–6159, 2023.
- 548 Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of
549 natural images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*
550 *recognition*, pp. 18208–18218, 2022.
- 551 Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie Li, Hamid Kazemi, Furong Huang, Micah Gold-
552 blum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms
553 without noise. *Advances in Neural Information Processing Systems*, 36, 2024.
- 554 Vance W Berger and YanYan Zhou. Kolmogorov–smirnov test: Overview. *Wiley statsref: Statistics*
555 *reference online*, 2014.
- 556 Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist
557 to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp.
558 2921–2926. IEEE, 2017.
- 559 Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data
560 via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):
561 1–22, 1977.
- 562 Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances*
563 *in neural information processing systems*, 34:8780–8794, 2021.
- 564 Giovanni Fasano and Alberto Franceschini. A multidimensional version of the kolmogorov–smirnov
565 test. *Monthly Notices of the Royal Astronomical Society*, 225(1):155–170, 1987.
- 566 Hanzhong Guo, Cheng Lu, Fan Bao, Tianyu Pang, Shuicheng Yan, Chao Du, and Chongxuan Li.
567 Gaussian mixture solvers for diffusion models. *Advances in Neural Information Processing Sys-*
568 *tems*, 36, 2024.
- 569 Tiankai Hang and Shuyang Gu. Improved noise schedule for diffusion training. *arXiv preprint*
570 *arXiv:2407.03297*, 2024.
- 571 Etrit Haxholli and Marco Lorenzi. Faster training of diffusion models and improved density estima-
572 tion via parallel score matching, 2023. URL <https://arxiv.org/abs/2306.02658>.
- 573 Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter.
574 Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in*
575 *neural information processing systems*, 30, 2017.
- 576 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In
577 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neu-*
578 *ral Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc.,
579 2020. URL [https://proceedings.neurips.cc/paper_files/paper/2020/](https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf)
580 [file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf).
- 581 Tero Karras. Progressive growing of gans for improved quality, stability, and variation. *arXiv*
582 *preprint arXiv:1710.10196*, 2017.
- 583 Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-
584 based generative models. *Advances in neural information processing systems*, 35:26565–26577,
585 2022.
- 586 Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster anal-*
587 *ysis*. John Wiley & Sons, 2009.

- 594 Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and
595 Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the*
596 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6007–6017, 2023.
597
- 598 Amirhossein Kazerooni, Ehsan Khodapanah Aghdam, Moein Heidari, Reza Azad, Mohsen Fayyaz,
599 Ilker Hacihaliloglu, and Dorit Merhof. Diffusion models in medical imaging: A comprehensive
600 survey. *Medical Image Analysis*, 88:102846, 2023.
- 601 Firas Khader, Gustav Müller-Franzes, Soroosh Tayebi Arasteh, Tianyu Han, Christoph Haarbuerger,
602 Maximilian Schulze-Hagen, Philipp Schad, Sandy Engelhardt, Bettina Baeßler, Sebastian Foer-
603 sch, et al. Denoising diffusion probabilistic models for 3d medical image generation. *Scientific*
604 *Reports*, 13(1):7303, 2023.
- 605
- 606 Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architec-
607 turally compressed stable diffusion for efficient text-to-image generation. In *Workshop on Effi-*
608 *cient Systems for Foundation Models@ ICML2023*, 2023.
- 609 Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Ad-*
610 *vances in neural information processing systems*, 34:21696–21707, 2021.
611
- 612 Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile
613 diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.
- 614 Yichong Leng, Zehua Chen, Junliang Guo, Haohe Liu, Jiawei Chen, Xu Tan, Danilo Mandic, Lei
615 He, Xiangyang Li, Tao Qin, et al. Binauralgrad: A two-stage conditional diffusion probabilistic
616 model for binaural audio synthesis. *Advances in Neural Information Processing Systems*, 35:
617 23689–23700, 2022.
- 618
- 619 Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast
620 ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural*
621 *Information Processing Systems*, 35:5775–5787, 2022a.
- 622 Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast
623 solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*,
624 2022b.
- 625
- 626 Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American*
627 *statistical Association*, 46(253):68–78, 1951.
- 628 Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon.
629 Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint*
630 *arXiv:2108.01073*, 2021.
631
- 632 Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for
633 editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference*
634 *on Computer Vision and Pattern Recognition*, pp. 6038–6047, 2023.
- 635 Eliya Nachmani, Robin San Roman, and Lior Wolf. Non gaussian denoising diffusion models. *arXiv*
636 *preprint arXiv:2106.07582*, 2021.
637
- 638 Kushagra Pandey, Jaideep Pathak, Yilun Xu, Stephan Mandt, Michael Pritchard, Arash Vahdat, and
639 Morteza Mardani. Heavy-tailed diffusion models, 2024a. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2410.14171)
640 [2410.14171](https://arxiv.org/abs/2410.14171).
- 641 Kushagra Pandey, Maja Rudolph, and Stephan Mandt. Efficient integrators for diffusion generative
642 models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL
643 <https://openreview.net/forum?id=qA4FoxO5GE>.
644
- 645 Pablo Pernias, Dominic Rampas, Mats Leon Richter, Christopher Pal, and Marc Aubreville.
646 Würstchen: An efficient architecture for large-scale text-to-image diffusion models. In *The*
647 *Twelfth International Conference on Learning Representations*, 2024. URL [https://](https://openreview.net/forum?id=gU58d5QeGv)
openreview.net/forum?id=gU58d5QeGv.

- 648 Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe
649 Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image
650 synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL
651 <https://openreview.net/forum?id=di52zR8xgf>.
- 652
- 653 Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-
654 conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.
- 655
- 656 Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-
657 resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF confer-
658 ence on computer vision and pattern recognition*, pp. 10684–10695, 2022.
- 659
- 660 Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analy-
661 sis. *Journal of computational and applied mathematics*, 20:53–65, 1987.
- 662
- 663 Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar
664 Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic
665 text-to-image diffusion models with deep language understanding. *Advances in neural informa-
666 tion processing systems*, 35:36479–36494, 2022.
- 667
- 668 Ivan Skorokhodov, Willi Menapace, Aliaksandr Siarohin, and Sergey Tulyakov. Hierarchical patch
669 diffusion models for high-resolution video generation. In *Proceedings of the IEEE/CVF Confer-
670 ence on Computer Vision and Pattern Recognition*, pp. 7569–7579, 2024.
- 671
- 672 Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv
673 preprint arXiv:2010.02502*, 2020a.
- 674
- 675 Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution.
676 *Advances in neural information processing systems*, 32, 2019.
- 677
- 678 Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben
679 Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint
680 arXiv:2011.13456*, 2020b.
- 681
- 682 Wenpin Tang and Hanyang Zhao. Score-based diffusion models via stochastic differential
683 equations—a technical tutorial. *arXiv preprint arXiv:2402.07487*, 2024.
- 684
- 685 Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang,
686 Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of
687 diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
688 URL <https://openreview.net/forum?id=iv2sTQtbst>.
- 689
- 690 Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers
691 for diffusion models by differentiating through sample quality. In *International Conference on
692 Learning Representations*, 2022.
- 693
- 694 Fang Wu and Stan Z Li. Diffmd: a geometric diffusion model for molecular dynamics simulations.
695 In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5321–5329,
696 2023.
- 697
- 698 Shuchen Xue, Mingyang Yi, Weijian Luo, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhi-Ming
699 Ma. Sa-solver: Stochastic adams solver for fast sampling of diffusion models. *Advances in Neural
700 Information Processing Systems*, 36, 2024.
- 701
- 702 Shuai Yang, Yukang Chen, Luozhou Wang, Shu Liu, and Ying-Cong Chen. Denoising diffusion
703 step-aware models. In *The Twelfth International Conference on Learning Representations*, 2024.
704 URL <https://openreview.net/forum?id=c43FGk8Pcg>.
- 705
- 706 Hao Yen, François G Germain, Gordon Wichern, and Jonathan Le Roux. Cold diffusion for speech
707 enhancement. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and
708 Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

702 Sihyun Yu, Weili Nie, De-An Huang, Boyi Li, Jinwoo Shin, and Anima Anandkumar. Efficient video
703 diffusion models via content-frame motion-latent decomposition. In *The Twelfth International*
704 *Conference on Learning Representations*, 2024. URL [https://openreview.net/forum?](https://openreview.net/forum?id=dQVtTdsVZH)
705 [id=dQVtTdsVZH](https://openreview.net/forum?id=dQVtTdsVZH).
706

707 Martin Zach, Thomas Pock, Erich Kobler, and Antonin Chambolle. Explicit diffusion of gaussian
708 mixture model based image priors. In *International Conference on Scale Space and Variational*
709 *Methods in Computer Vision*, pp. 3–15. Springer, 2023.

710 Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image
711 diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*,
712 pp. 3836–3847, 2023.

713 Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator.
714 In *The Eleventh International Conference on Learning Representations*, 2023. URL [https:](https://openreview.net/forum?id=Loek7hfb46P)
715 [//openreview.net/forum?id=Loek7hfb46P](https://openreview.net/forum?id=Loek7hfb46P).
716

717 Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-
718 corrector framework for fast sampling of diffusion models. *Advances in Neural Information*
719 *Processing Systems*, 36, 2024.

720 Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast
721 sampling of diffusion models via operator learning. In *International conference on machine learn-*
722 *ing*, pp. 42390–42402. PMLR, 2023.
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

A MIXSGM WITH VARIANCE ESTIMATION

As discussed in Section 4.2, the variance of each Gaussian component in the prior distribution of the mixSGM can be any arbitrary positive value, denoted by σ_T^2 , and is not necessarily constrained to 1. In this section, we incorporate data-driven variance estimation for each component and provide numerical results to demonstrate the improvements achieved through variance estimation.

Given the number of components K , the parametric estimation of the prior distribution can be formalized as

$$\min_{\mathbf{c}_i, \sigma_i} \text{ReEff}_{\text{mix+var}}^{\text{SGM}} := \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_{D(\mathbf{x}_0)}, \sigma_{D(\mathbf{x}_0)}^2 \mathbf{I})} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2]. \quad (22)$$

Classical methods including Expectation Maximization algorithm Dempster et al. (1977) can be applied to solve the optimization problem (22). In addition, we provide a simpler method to estimate the variances based on the dispatcher D . To be more specific, we define

$$\hat{\sigma}_i^2 = \frac{1}{|\{\mathbf{x} : D(\mathbf{x}) = i\}|} \sum_{D(\mathbf{x})=i} \frac{1}{d} \|\mathbf{x} - \mathbf{c}_i\|_2^2 \text{ for } i = 1, 2, \dots, d, \quad (23)$$

where d is the dimension of the state space.

With the given variance estimations σ_i , the forward SDE for the model starting from data samples \mathbf{x}_0 is given by

$$d\mathbf{x}_t = f_t(\mathbf{x}_t - \mathbf{c}_j)dt + \sigma_j g_t d\mathbf{w}_t, \quad j = D(\mathbf{x}_0). \quad (24)$$

Following the notations in Section 4.2, the training procedure is to solve the optimization problem:

$$\min_{\theta} \mathcal{L}_{\text{mix+var}}^{\text{SGM}} = \int_0^T \omega_t \cdot \mathbb{E}_{\mathbf{x}_0, \epsilon} [\|\epsilon_{\theta}(\mathbf{x}_t, t, j) - \epsilon\|_2^2] dt, \quad (25)$$

where $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \mathbf{c}_{D(\mathbf{x}_0)} + \sigma_t \sigma_{D(\mathbf{x}_0)} \epsilon$. The current prior distribution can be written as $\sum_{i=1}^K p_i \mathcal{N}(\mathbf{c}_i, \sigma_i^2 \mathbf{I})$, where p_i is the proportion of data that are assigned to the i -th center, as defined in Section 4. Moreover, the reverse SDE is given by

$$d\tilde{\mathbf{x}}_t = \left(f_t(\tilde{\mathbf{x}}_t - \mathbf{c}_j) + \frac{g_t^2 \sigma_j^2}{\sigma_t} \epsilon_{\theta}(\tilde{\mathbf{x}}_t, t, j) \right) dt + g_t \sigma_j d\tilde{\mathbf{w}}_t \quad (26)$$

given $\tilde{\mathbf{x}}_T$ comes from the j -th component, i.e. $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_j, \sigma_j^2 \mathbf{I})$. For ease of exposition, we abbreviate the above mixSGM with variance estimation to mixSGM+var. Following the same experiment settings in Section 5.3, we compare FID score among the SGM, the mixSGM and the mixSGM+var in Table 7 below. All the Improvement Ratio (Impr. Ratio) are calculated with respect to the SGM.

Table 7: Experiment Result for mixSGM+var on CIFAR10

Model \ Training Steps	180k	240k	300k	360k	420k	480k	540k	600k
SGM	65.82	45.55	49.94	35.22	34.88	24.58	28.42	20.46
mixSGM	62.41	40.52	36.38	22.66	24.25	16.93	17.81	21.89
Impr. Ratio	0.05	0.11	0.27	0.36	0.30	0.31	0.37	-0.07
mixSGM+var	51.22	36.17	29.58	22.17	18.05	16.65	15.73	13.09
Impr. Ratio	0.22	0.21	0.41	0.37	0.48	0.32	0.45	0.36

The results in Table 7 indicate that mixSGM+var consistently achieves lower FID scores compared to mixSGM. This finding further demonstrates the efficacy of the mixed diffusion model for image generation tasks, as the variance estimation method proposed in (23) requires minimal computation even in high-dimensional state spaces.

B VISUALIZATION OF 2D EXAMPLE

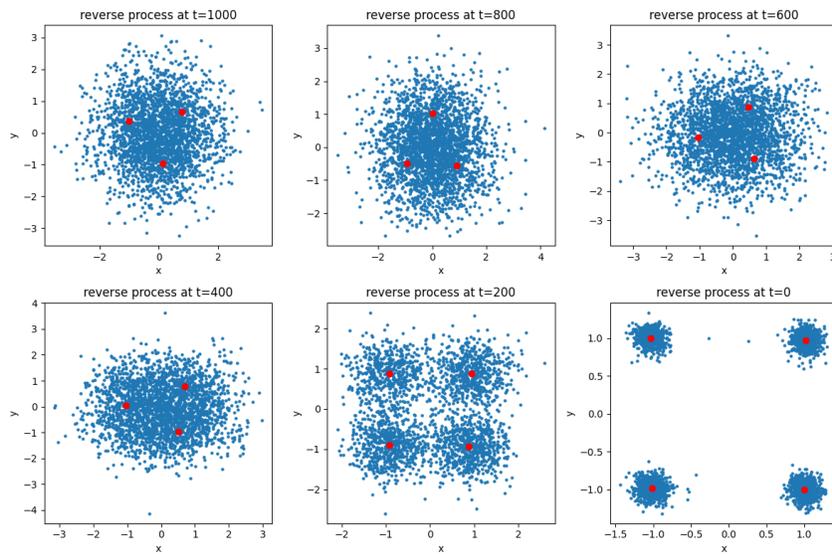
In this section, we take 2D diffusion model as an illustrative example to show how smaller reverse effort help improve the sample quality. Throughout the section, we choose the data distribution to

810 be

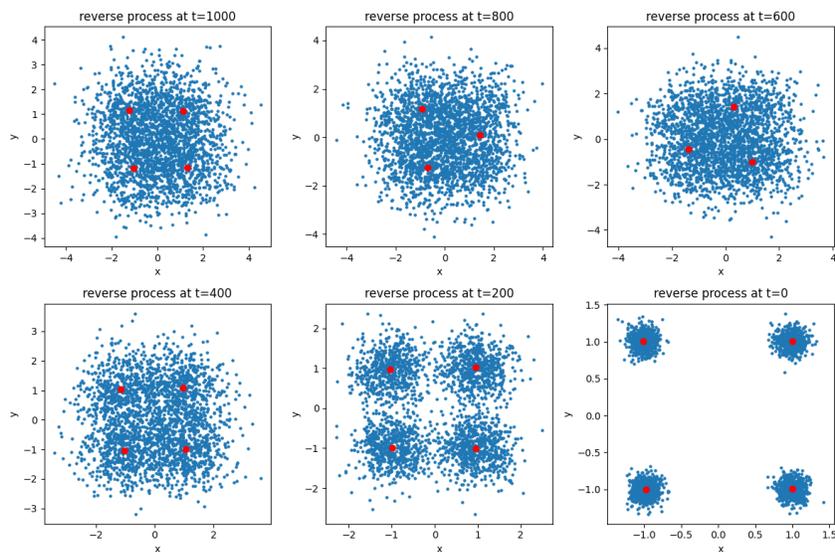
$$811 \frac{1}{4} \left(\mathcal{N}((1, 1), 0.01\mathbf{I}) + \mathcal{N}((-1, 1), 0.01\mathbf{I}) + \mathcal{N}((1, -1), 0.01\mathbf{I}) + \mathcal{N}((-1, -1), 0.01\mathbf{I}) \right), \quad (27)$$

813 which is a 2D Gaussian mixture distribution.

814 We draw 5,120 samples from this distribution and use the same data samples to independently train
 815 both the DDPM and the mixDDPM, using the same noise schedule and time horizon. Additionally,
 816 the depth of the two neural networks and the width of each layer are kept the same. During the
 817 reverse sampling process, we draw 2560 samples from each model. We plot the sample paths of
 818 the reverse process for both models in Figure 6 below. The blue points represent samples from the
 819 reverse process, drawn at six equidistant time points: 1000, 800, 600, 400, 200 and 0. We also
 820 apply the Silhouette method Rousseeuw (1987) to determine the optimal number of clusters from
 821 the reverse process samples and plot the cluster centers with red points.
 822 the reverse process samples and plot the cluster centers with red points.



831 (a) DDPM



839 (b) mixDDPM

840 Figure 6: The reverse process

The visualizations of the reverse process reveal that mixDDPM forms four clusters at $t = 400$, while DDPM samples still fail to exhibit clear clustering at this stage. By $t = 200$, mixDDPM produces well-defined clusters, whereas DDPM exhibits more points distributed between nearby clusters. Ultimately, mixDDPM demonstrates better sampling quality since the variance of each cluster is closer to the data distribution, compared to samples generated by DDPM. These findings indicate that mixDDPM, by reducing the reverse effort, alleviates the model’s burden and facilitates more efficient sampling by producing clusters earlier and more distinctly than DDPM.

C ADDITIONAL IMPLEMENTATION DETAILS

C.1 CENTER SELECTION METHODS

In this section, we provide several center selection methods based on the analysis of the training data. As mentioned in Section 4, the specific method can be determined by users and does not need to be optimal.

1. **Data-driven clustering method.** The data-driven clustering method first applies a traditional data-clustering technique Rousseeuw (1987); Kaufman & Rousseeuw (2009) to the samples from the data distribution. To be more specific, the method first calculates the average Silhouette’s coefficient for samples from the data distribution under different number of clusters. By maximizing the average Silhouette’s coefficient over different values of K , the method determines an optimal value of K . Subsequently, the method applies the k-means algorithm to find the K cluster centers for the data distribution and we denote them by $\mathbf{c}_1, \dots, \mathbf{c}_K \in \mathbb{R}^d$. We summarize the implementation of this method in Algorithm 3 below.

Algorithm 3 Data-driven Clustering Method

Input: Datasets \mathcal{S} , maximum K value K_{\max}
Output: the number of clusters K and the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$
 $SC \leftarrow [0] * (K_{\max} - 1)$
for $k = 2$ to K_{\max} **do**
 Apply the k-means algorithm to find k cluster centers $\mathbf{c}_1^k, \dots, \mathbf{c}_k^k$
 for \mathbf{x}_0 in \mathcal{S} **do**
 $SC[k - 2]_+ = \text{Silhouette's coefficient of } \mathbf{x}_0$
 end for
 $K' = \arg \min_k SC[k]$
end for
Return $K = K' + 2$ and $\mathbf{c}_1^K, \dots, \mathbf{c}_K^K$

2. **Data Labeling.** When samples from the data distribution have either pre-given labels or can be labeled through pre-trained classifier, the labels naturally separate the samples into several groups. Hence, the mixed diffusion models can follow the number of different labels and the centers among samples with the same label to determine the value of K and $\mathbf{c}_1, \dots, \mathbf{c}_K$.
3. **Alternative methods.** Alternatively, the number of clusters K and the centers of the clusters $\mathbf{c}_1, \dots, \mathbf{c}_K \in \mathbb{R}^d$ can be seen as pre-given hyperparameters that are possibly specified by domain knowledge or other preliminary data analysis.

C.2 ALGORITHMS FOR THE MIXSGM

Algorithms for the training and sampling process of the mixSGM are shown in Algorithm 4 and 5 below.

Algorithm 4 Training Process for the Mixed SGM

Input: samples \mathbf{x}_0 from the data distribution, un-trained neural network ϵ_θ , time horizon T , scalar function f, g , number of centers K and the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$
Output: Trained neural network ϵ_θ
 Calculate α_t and σ_t in closed-form
repeat
 Get data \mathbf{x}_0
 Find center $j = D(\mathbf{x}_0)$
 Sample $t \sim U[0, T]$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 $\mathbf{x}_t \leftarrow \alpha_t \mathbf{x}_0 + \mathbf{c}_j + \sigma_t \epsilon$
 $\mathcal{L} \leftarrow \omega_t \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, j)\|_2^2$
 Take a gradient descent step on $\nabla_\theta \mathcal{L}$
until Converged or training resource/time limit is hit

Algorithm 5 Reverse Process for the Mixed SGM

Input: Trained neural network ϵ_θ , center weights p_1, \dots, p_K , centers $\mathbf{c}_1, \dots, \mathbf{c}_K$
 Sample $j \in \{1, \dots, K\}$ with $\mathbb{P}(j = i) = p_i$ for $i = 1, \dots, K$
 Sample $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_j, \sigma_T^2 \mathbf{I})$
 Apply numerical solvers to the reverse SDE (18).
Return $\tilde{\mathbf{x}}_0$

C.3 IMPLEMENTATION DETAILS ON EMNIST

We apply the U-Net architecture to learn the noise during the training of both the original diffusion models and the mixed diffusion models. The down-sampling path consists of three blocks with progressively increasing output channels. The specific number of output channels are 32, 64, and 128. The third block incorporates attention mechanisms to capture global context. Similarly, the up-sampling path mirrors the down-sampling structure, with the first block replaced by an attention block to refine spatial details. For the mixed diffusion models, we use class embeddings to incorporate the assignments from the dispatcher and employ a data-driven clustering method, as described in Algorithm 3. The data are preprocessed with a batch size of 16.

For both DDPM and mixDDPM, we set the time step to $T = 1000$ and choose the noise schedule β_t as a linear function of t , with $\beta_1 = 0.001$ and $\beta_{1000} = 0.02$. For the SGM, we select the following forward SDE:

$$d\mathbf{x}_t = -\frac{1}{2}\beta_t \mathbf{x}_t dt + \sqrt{\beta_t} d\mathbf{w}_t. \quad (28)$$

For mixSGM, the forward SDE is defined as:

$$d\mathbf{x}_t = -\frac{1}{2}\beta_t (\mathbf{x}_t - \mathbf{c}_j) dt + \sqrt{\beta_t} d\mathbf{w}_t, \quad \text{where } j = D(\mathbf{x}_0). \quad (29)$$

Here, β_t is chosen to be a linear function with $\beta_0 = 0.1$ and $\beta_1 = 40$ for both SGM and mixSGM. We use the DPM solver Lu et al. (2022a) for efficient sampling.

C.4 IMPLEMENTATION DETAILS ON CIFAR10

For both the original and the mixed diffusion models, we apply the U-Net architecture, which consists of a series of down-sampling and up-sampling blocks, with each block containing two layers. The down-sampling path has five blocks with progressively increasing output channels. The specific number of output channels are 32, 64, 128, 256, and 512. Among these, the fourth block integrates attention mechanisms to capture global context. Similarly, the up-sampling path mirrors the down-sampling structure, with the second block replaced by an attention block to refine spatial details. A dropout rate of 0.1 is applied to regularize the model. Specifically for the mixed diffusion models, the model utilizes class embeddings to incorporate the assignment provided by the dispatcher. Before training the neural networks, we first scale the training data to the range of $[-2, 2]$ with a batch size of 128. We choose the weighting function ω_t in (11) to be 1, regardless of the time step.

Since the images in CIFAR10 are already labeled, we adopt a data labeling method to determine the number of centers K and the centers $\mathbf{c}_1, \dots, \mathbf{c}_K$.

For DDPM and mixDDPM, the noise schedules are set with $\beta_1 = 0.001$ and $\beta_{1000} = 0.02$, following a linear schedule over 1000 steps. For SGM, we choose the forward SDE as:

$$d\mathbf{x}_t = -\frac{1}{2}\beta_t\mathbf{x}_tdt + \sqrt{\beta_t}d\mathbf{w}_t. \quad (30)$$

For mixSGM, we choose the forward SDE as:

$$d\mathbf{x}_t = -\frac{1}{2}\beta_t(\mathbf{x}_t - \mathbf{c}_j)dt + \sqrt{\beta_t}d\mathbf{w}_t, \text{ where } j = D(\mathbf{x}_0). \quad (31)$$

Here, β_t is chosen as a linear function with $\beta_0 = 0.1$ and $\beta_1 = 40$ for both SGM and mixSGM. We set the batch size to 128 and apply the DPM solver Lu et al. (2022a) for efficient sampling.

D PROOF

Proof of Proposition 1. We first calculate Eff^{DDPM} . Since \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ are independent, we have

$$\begin{aligned} \text{Eff}^{\text{DDPM}} &= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} [\|\tilde{\mathbf{x}}_T\|^2] - 2\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] \\ &= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + d, \end{aligned} \quad (32)$$

where d is the dimension of the state space. On the contrary, we calculate $\text{Eff}_{\text{mix}}^{\text{DDPM}}$ by first conditioning on the centers:

$$\begin{aligned} \text{Eff}_{\text{mix}}^{\text{DDPM}} &= \mathbb{E} \left[\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_D(\mathbf{x}_0), \mathbf{I})} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2] \mid D(\mathbf{x}_0) = i \right] \\ &= \sum_{i=1}^K p_i \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2] \\ &= \sum_{i=1}^K p_i \left(\mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\|\mathbf{x}_0\|^2] + \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I})} [\|\tilde{\mathbf{x}}_T\|^2] \right. \\ &\quad \left. - 2\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] \right) \\ &= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + d + \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2 - 2 \sum_{i=1}^K p_i \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T]. \end{aligned} \quad (33)$$

Since \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ are independent conditioned on the subspace X_i , we obtain

$$\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] = \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0]^T \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I})} [\tilde{\mathbf{x}}_T] = \mathbf{c}_i^T \mathbf{c}_i = \|\mathbf{c}_i\|^2. \quad (34)$$

Hence, the effort of the mixDDPM is given by

$$\text{Eff}_{\text{mix}}^{\text{DDPM}} = \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + d - \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2. \quad (35)$$

Combining (32) and (35), we finish the proof for (16). \square

Proof of Proposition 2. To prove Proposition 2, we first calculate Eff^{SGM} . Since \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ are independent, we have

$$\begin{aligned} \text{Eff}^{\text{SGM}} &= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I})} [\|\tilde{\mathbf{x}}_T\|^2] - 2\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, \sigma_T^2 \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] \\ &= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + \sigma_T^2 d, \end{aligned} \quad (36)$$

where d is the dimension of the state space. On the contrary, we calculate $\text{Eff}_{\text{mix}}^{\text{SGM}}$ by first conditioning on the centers:

$$\begin{aligned}
\text{Eff}_{\text{mix}}^{\text{SGM}} &= \mathbb{E} \left[\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_{D(\mathbf{x}_0)}, \sigma_T^2 \mathbf{I})} \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2] \mid D(\mathbf{x}_0) = i \right] \\
&= \sum_{i=1}^K p_i \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \sigma_T^2 \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\|\mathbf{x}_0 - \tilde{\mathbf{x}}_T\|^2] \\
&= \sum_{i=1}^K p_i \left(\mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\|\mathbf{x}_0\|^2] + \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \sigma_T^2 \mathbf{I})} [\|\tilde{\mathbf{x}}_T\|^2] \right. \\
&\quad \left. - 2 \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \sigma_T^2 \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] \right) \\
&= \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + \sigma_T^2 d + \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2 - 2 \sum_{i=1}^K p_i \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T].
\end{aligned} \tag{37}$$

Since \mathbf{x}_0 and $\tilde{\mathbf{x}}_T$ are independent conditioned on the subspace X_i , we obtain

$$\mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I}), \mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0^T \tilde{\mathbf{x}}_T] = \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}} | X_i} [\mathbf{x}_0]^T \mathbb{E}_{\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{c}_i, \mathbf{I})} [\tilde{\mathbf{x}}_T] = \mathbf{c}_i^T \mathbf{c}_i = \|\mathbf{c}_i\|^2. \tag{38}$$

Hence, the effort of the mixSGM is given by

$$\text{Eff}_{\text{mix}}^{\text{SGM}} = \mathbb{E}_{\mathbf{x}_0 \sim \bar{p}_{\text{data}}} [\|\mathbf{x}_0\|^2] + \sigma_T^2 d - \sum_{i=1}^K p_i \|\mathbf{c}_i\|^2. \tag{39}$$

Combining (36) and (39), we finish the proof for Proposition 2. \square

E ADDITIONAL NUMERICAL RESULTS

E.1 EXPERIMENT RESULTS ON CELEBA-HQ

In this section, we compare mixDDPM with DDPM on CelebA-HQ Karras (2017), a high-resolution facial image dataset. We extract a total of 12,800 images, evenly split between male and female subjects. We re-size the images to 128×128 resolution and choose $K = 2$ for mixDDPM.

For both the classic and the mixed diffusion models, we apply the U-Net architecture, which consists of a series of down-sampling and up-sampling blocks, with each block containing two layers. The down-sampling path has five blocks with progressively increasing output channels. The specific number of output channels are 64, 128, 128, 256 and 256. Among these, the fourth block integrates attention mechanisms to capture global context. Similarly, the up-sampling path mirrors the down-sampling structure, with the second block replaced by an attention block to refine spatial details. A dropout rate of 0.1 is applied to regularize the model.

Below are the generated samples from DDPM and mixDDPM with same experimental settings.

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

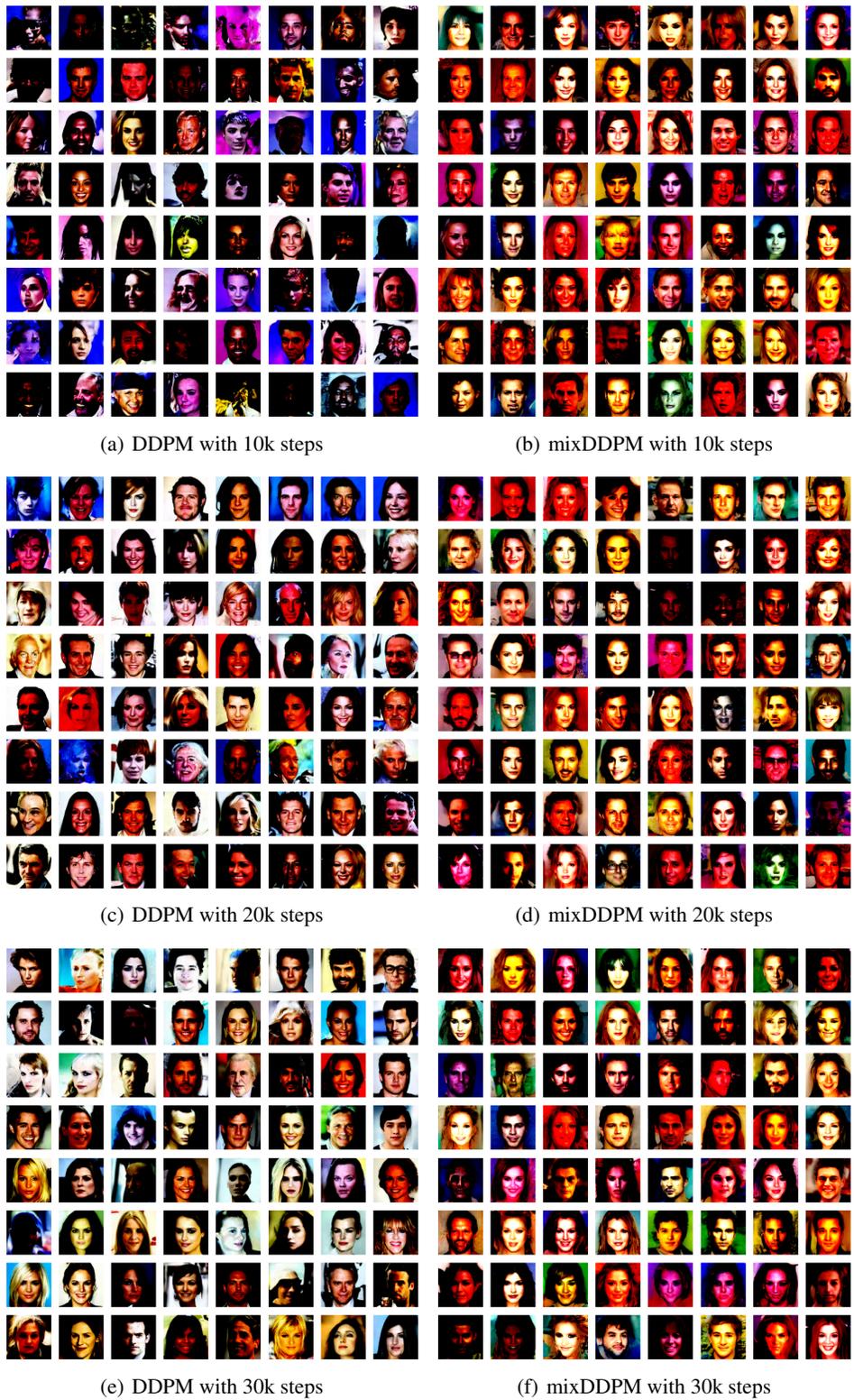


Figure 7: Experiment results on CelebA-HQ

E.2 ADDITIONAL EXPERIMENT RESULTS ON OAKLAND CALL CENTER DATASET

We present in this section the numerical results for SGM and mixSGM on the Oakland Call Center experiment. For this experiment, the training steps is 4k.

Table 8: SGM v.s. mixSGM on Oakland Call Center Datasets

	Benchmark	SGM	mixSGM
\mathcal{W}_1 Distance	0.172	0.400	0.228
\mathcal{W}_1 relative error		1.325	0.326
K-S statistics	0.112	0.189	0.144
K-S relative error		0.688	0.286

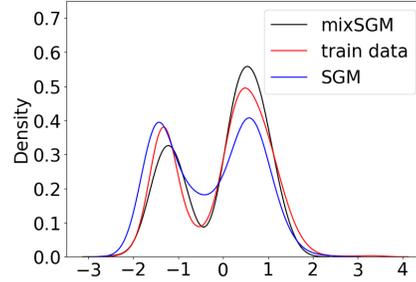


Figure 8: SGM and mixSGM on Oakland Call Center Dataset

E.3 ADDITIONAL EXPERIMENT RESULTS ON EMNIST

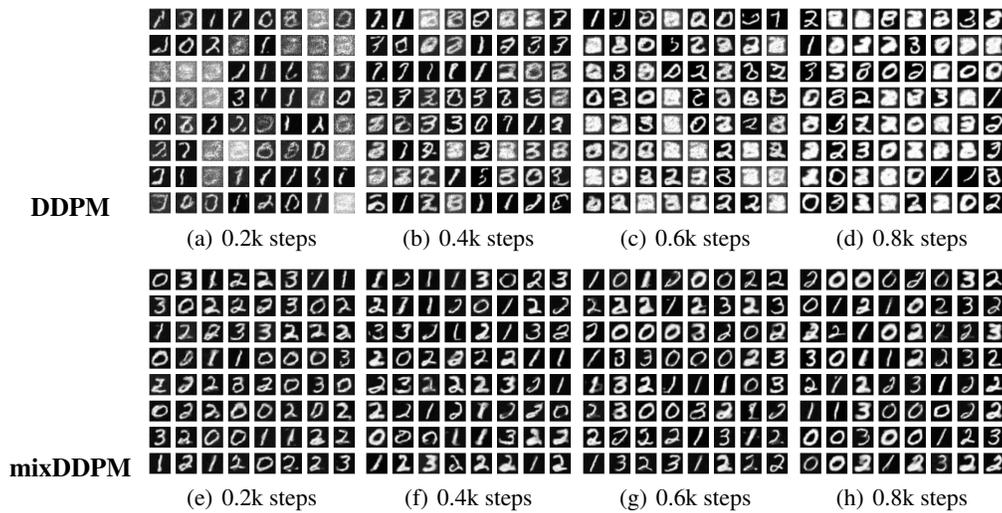


Figure 9: EMNIST Experiments with $N = 64$

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

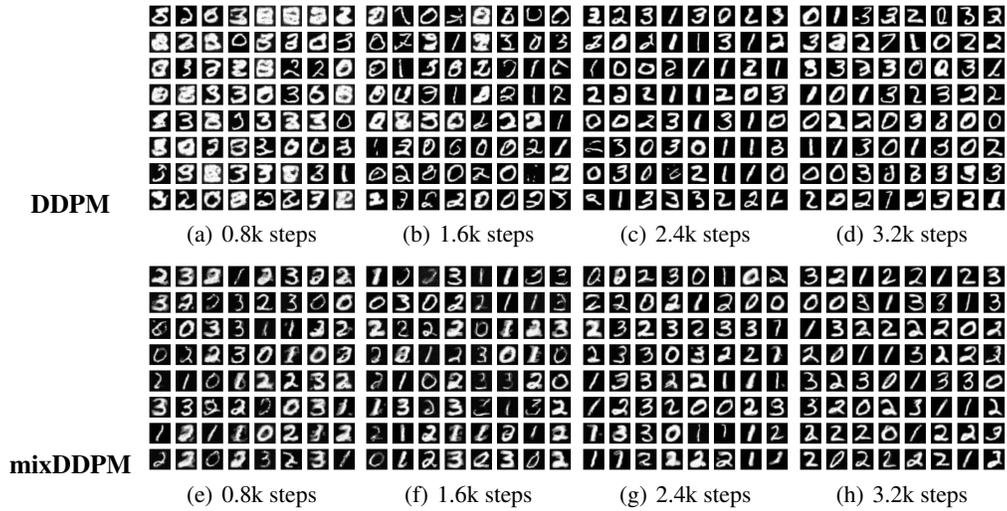


Figure 10: EMNIST Experiments with $N = 256$

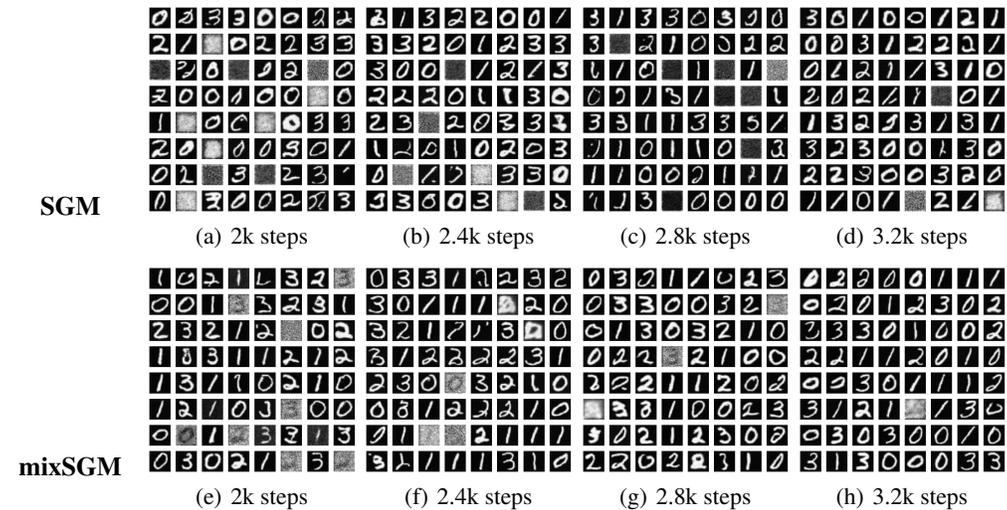


Figure 11: EMNIST Experiments with $N = 128$

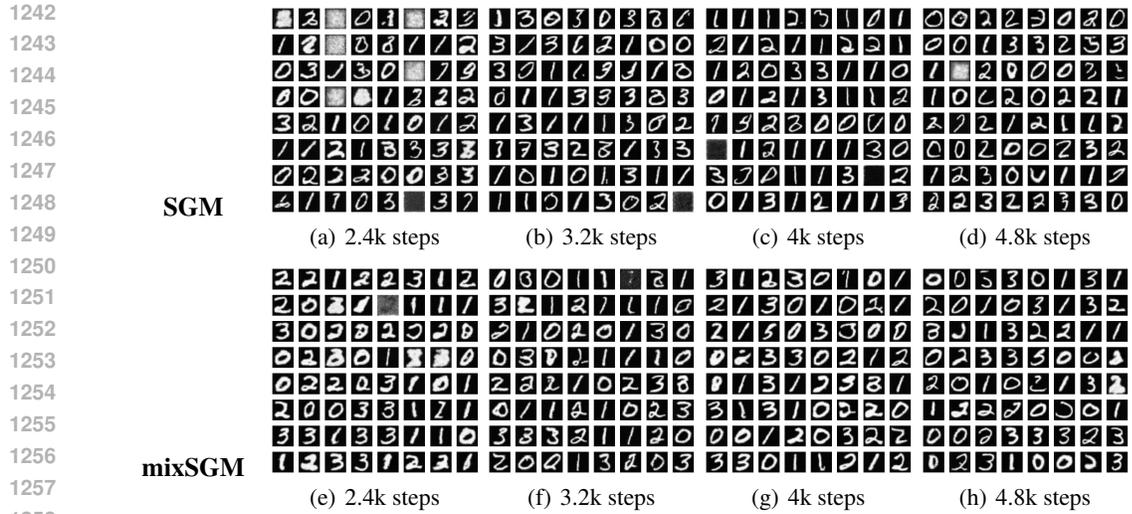


Figure 12: EMNIST Experiments with $N = 256$

E.4 ADDITIONAL EXPERIMENT RESULTS ON CIFAR10

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

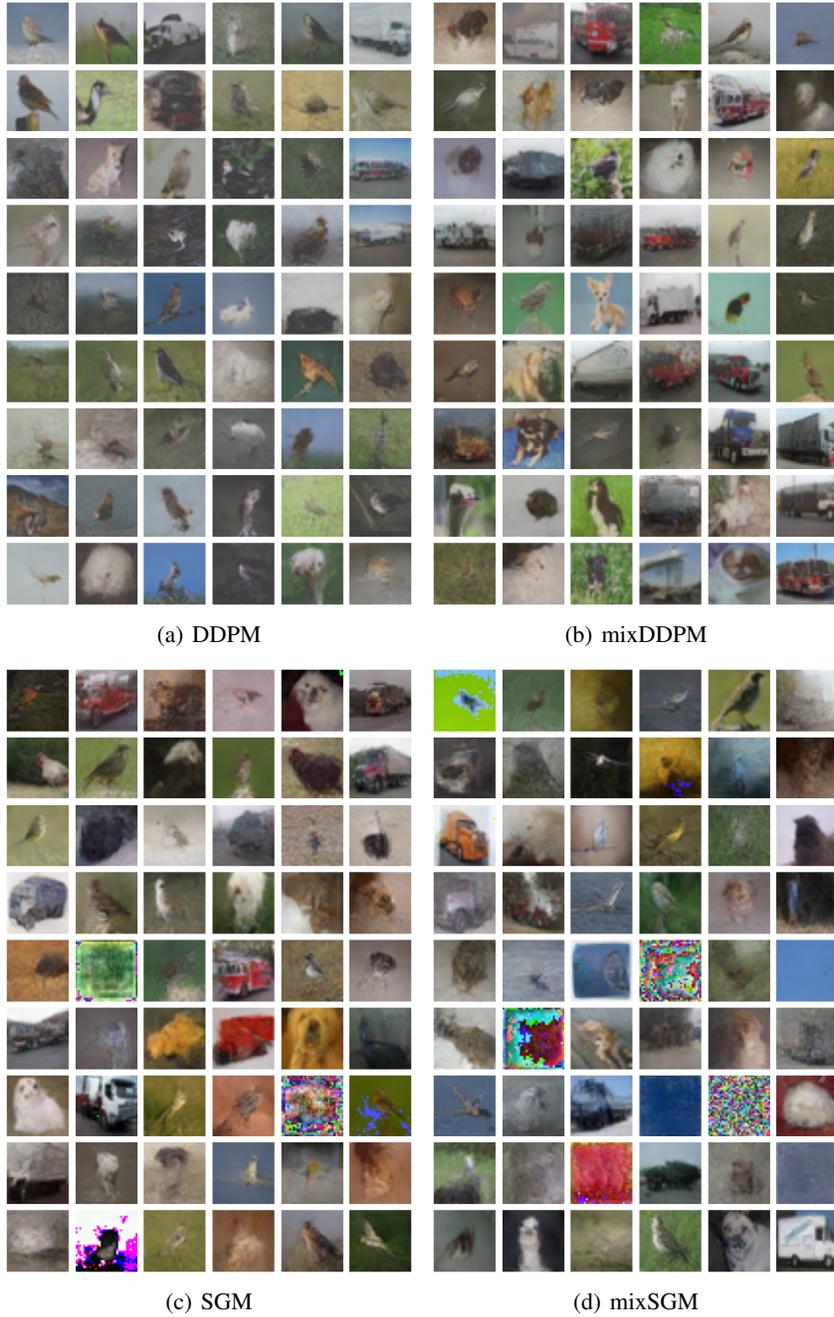
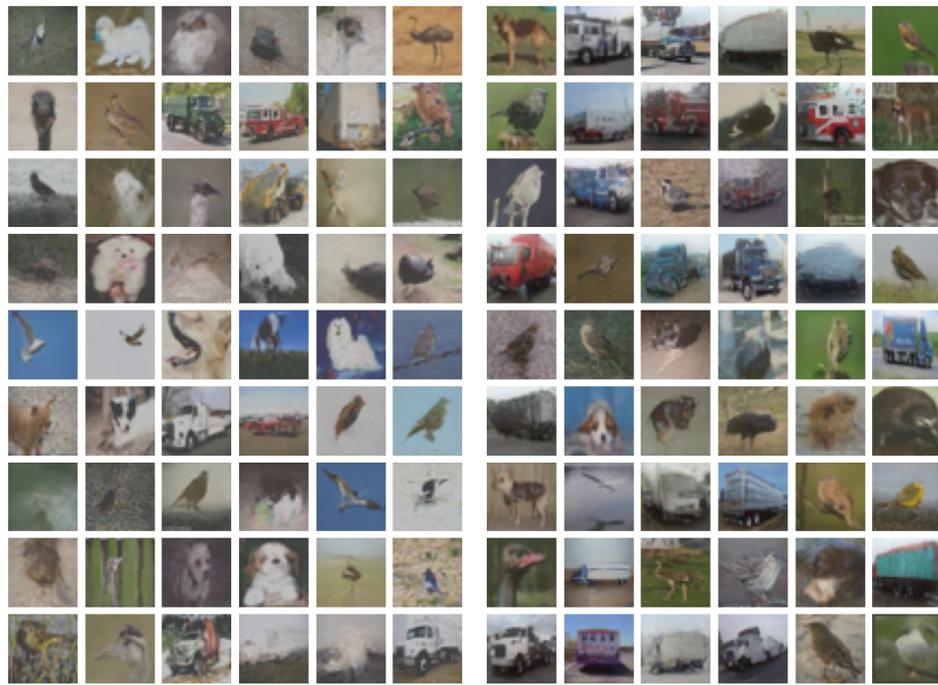


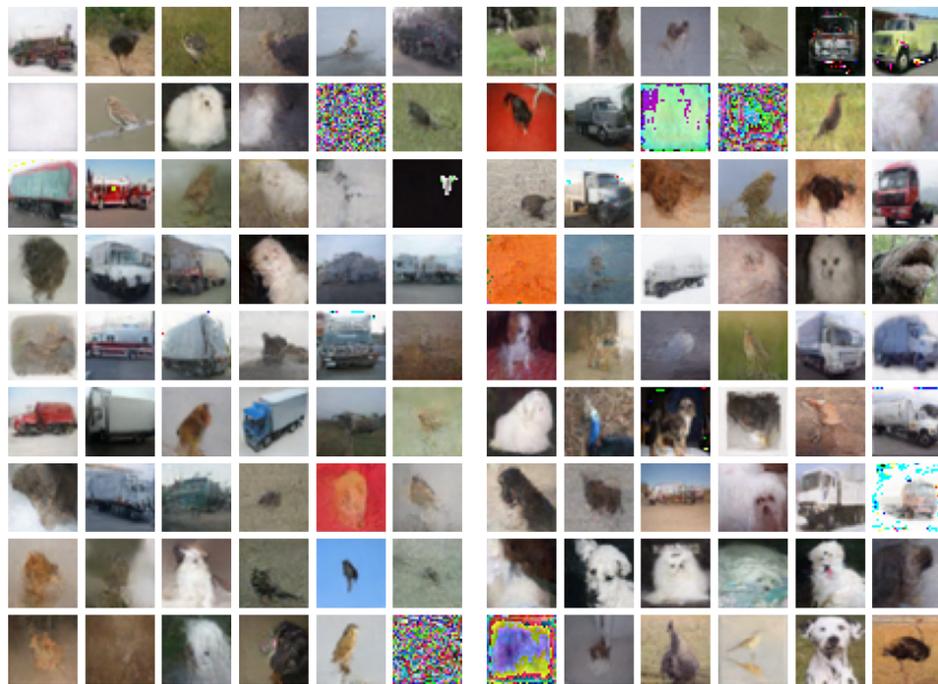
Figure 13: Experiments on CIFAR10 with 180k training steps

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403



(a) DDPM

(b) mixDDPM

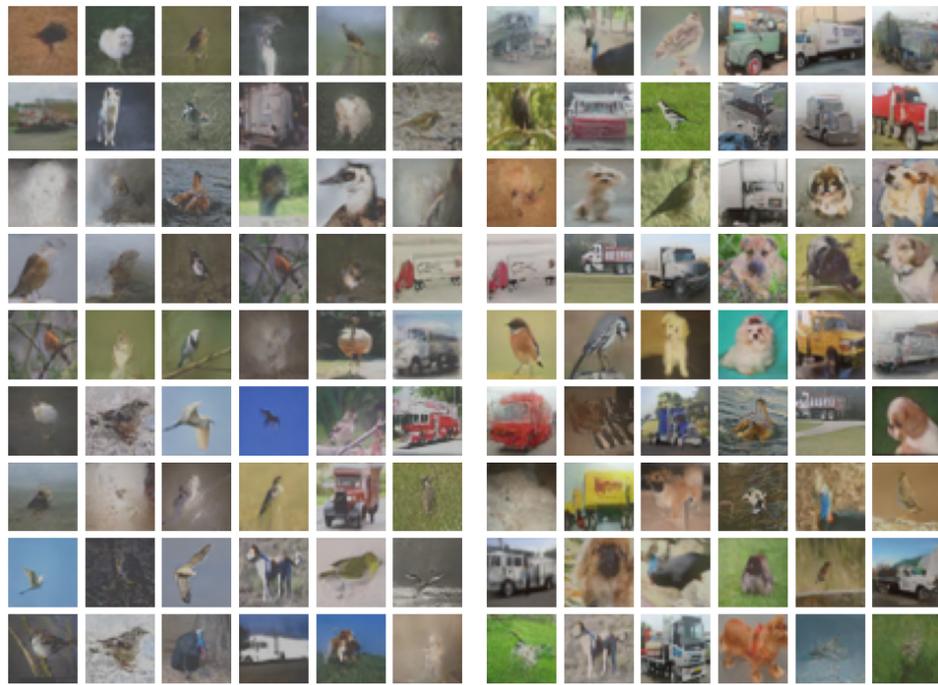


(c) SGM

(d) mixSGM

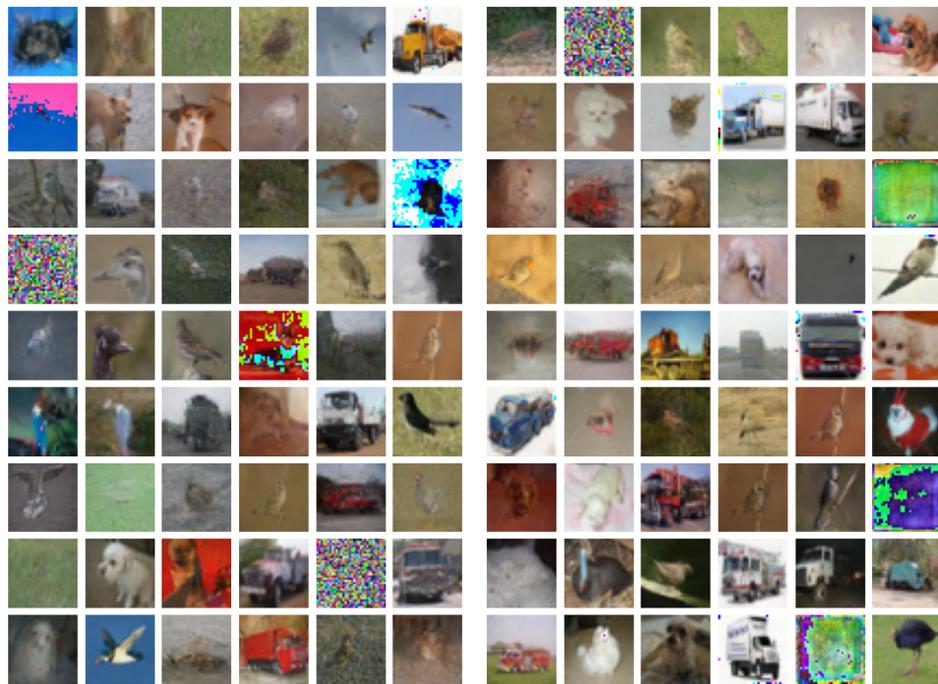
Figure 14: Experiments on CIFAR10 with 240k training steps

1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457



(a) DDPM

(b) mixDDPM

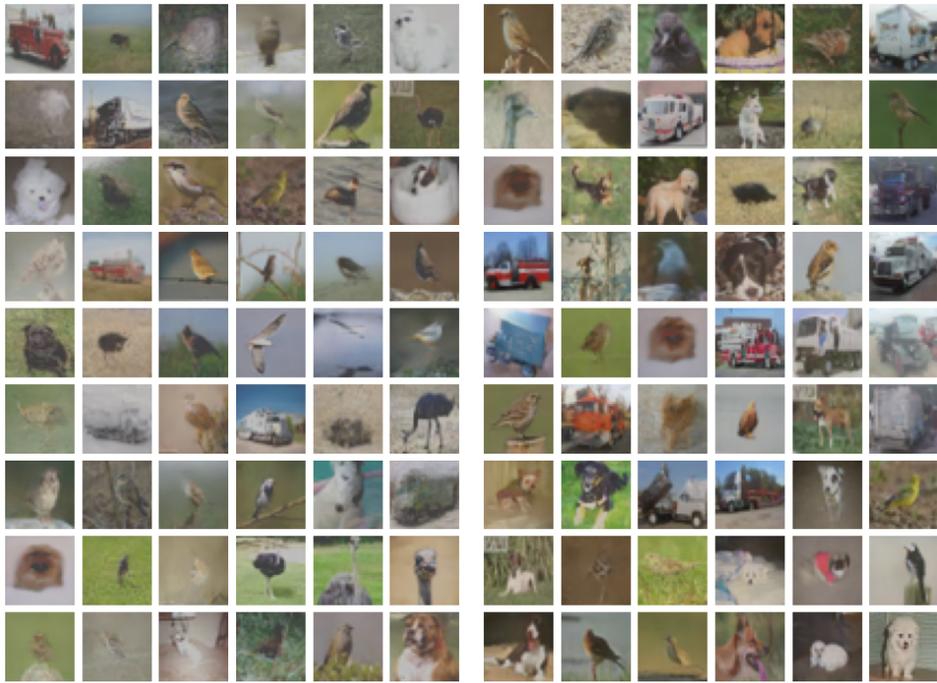


(c) SGM

(d) mixSGM

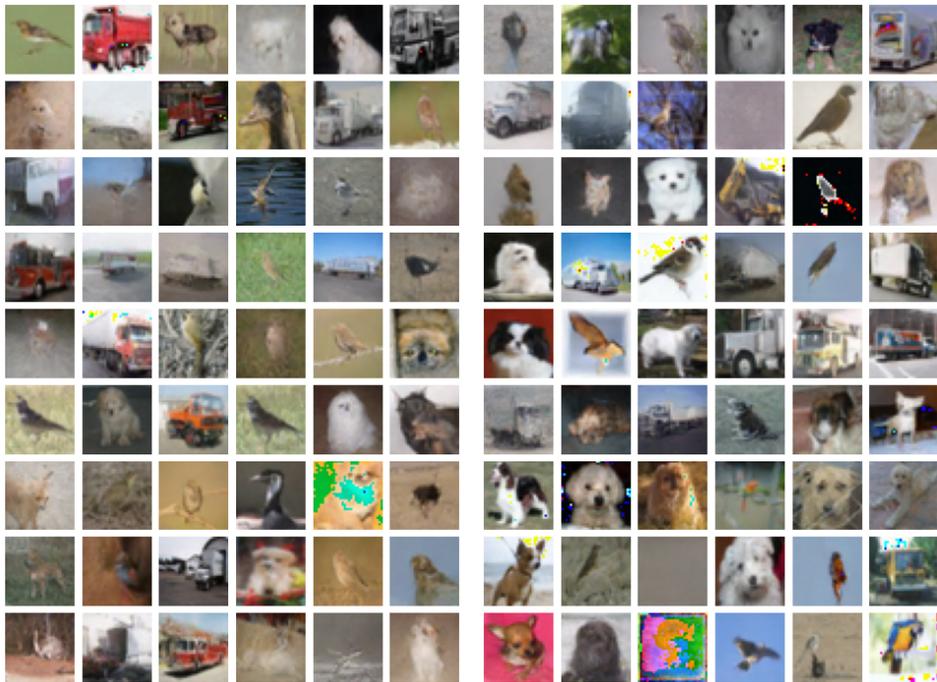
Figure 15: Experiments on CIFAR10 with 300k training steps

1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511



(a) DDPM

(b) mixDDPM

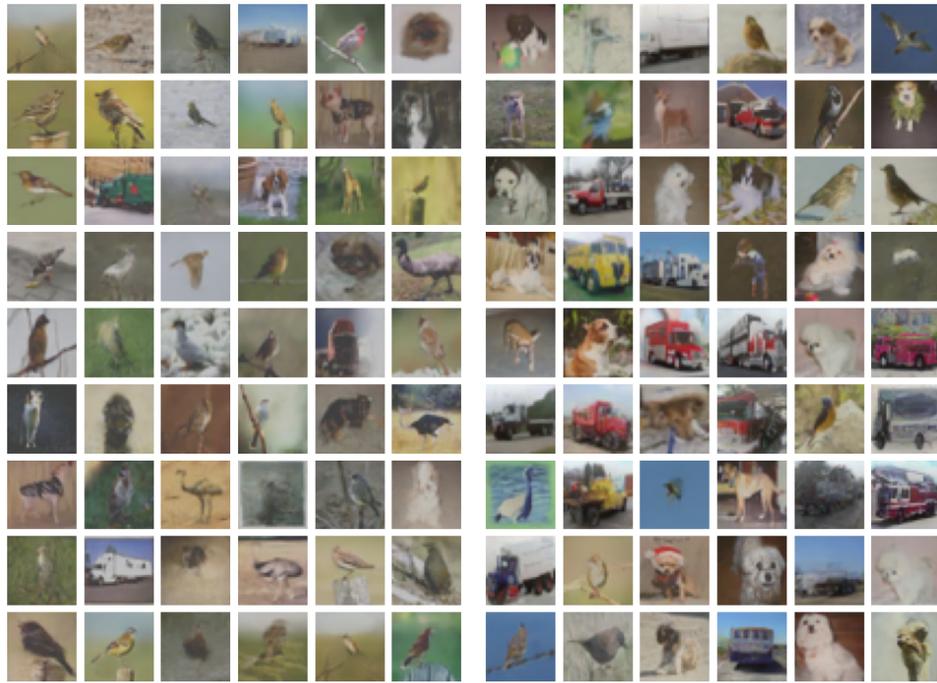


(c) SGM

(d) mixSGM

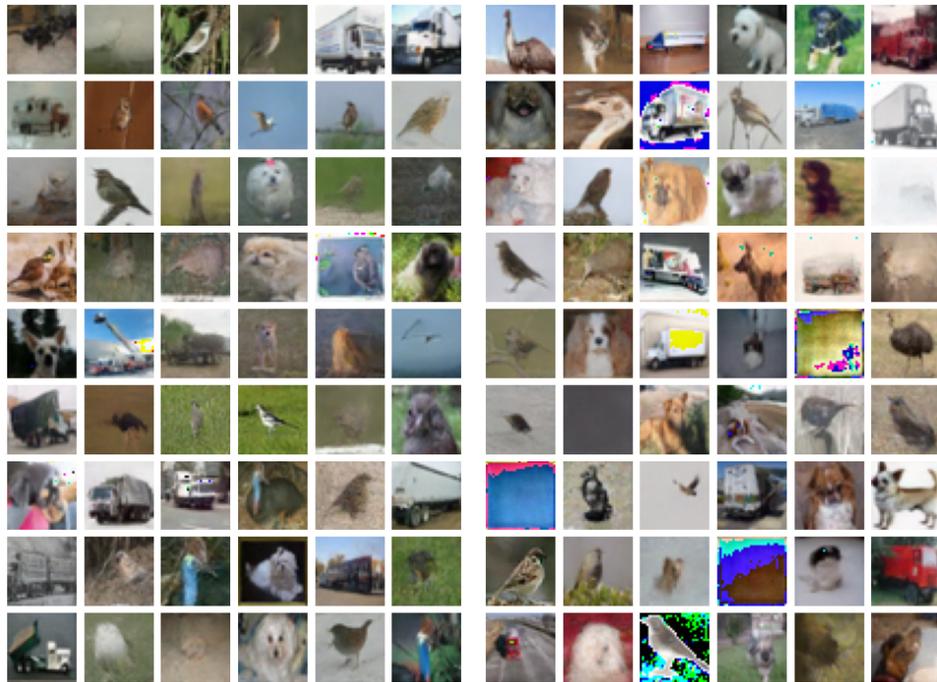
Figure 16: Experiments on CIFAR10 with 360k training steps

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565



(a) DDPM

(b) mixDDPM

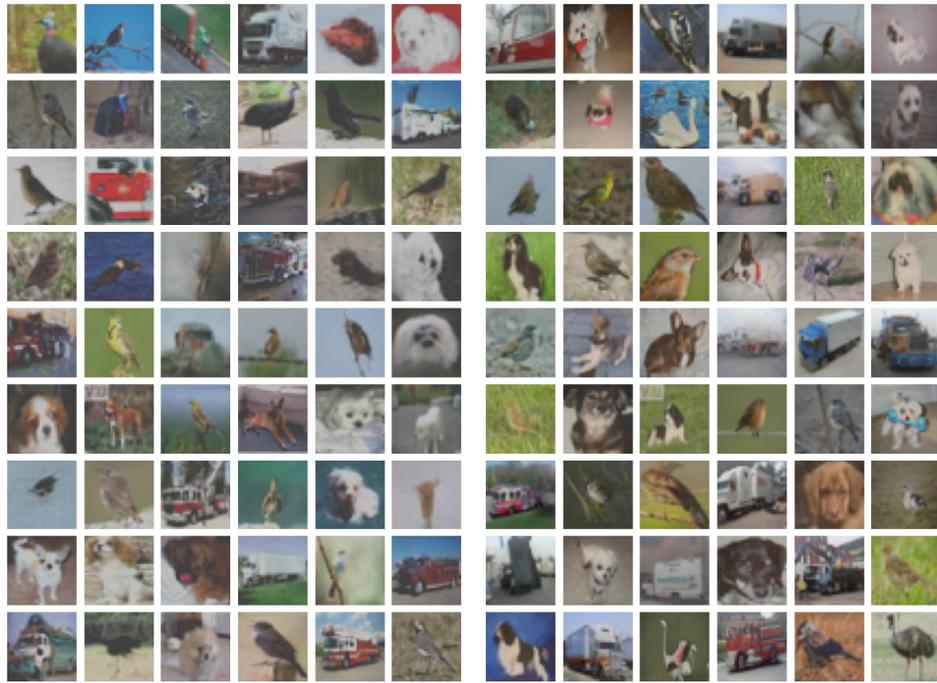


(c) SGM

(d) mixSGM

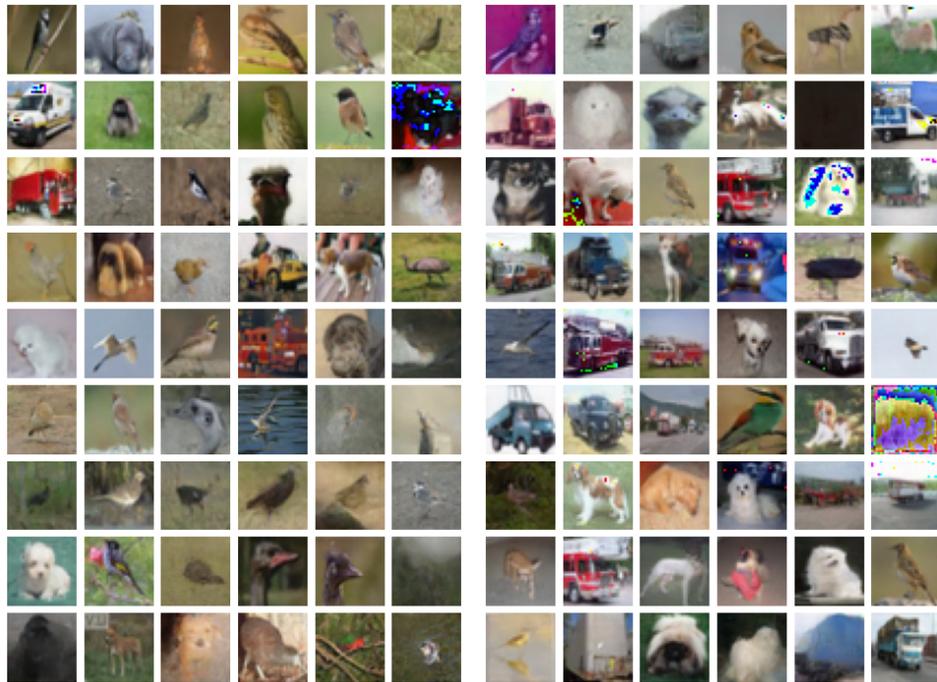
Figure 17: Experiments on CIFAR10 with 420k training steps

1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619



(a) DDPM

(b) mixDDPM

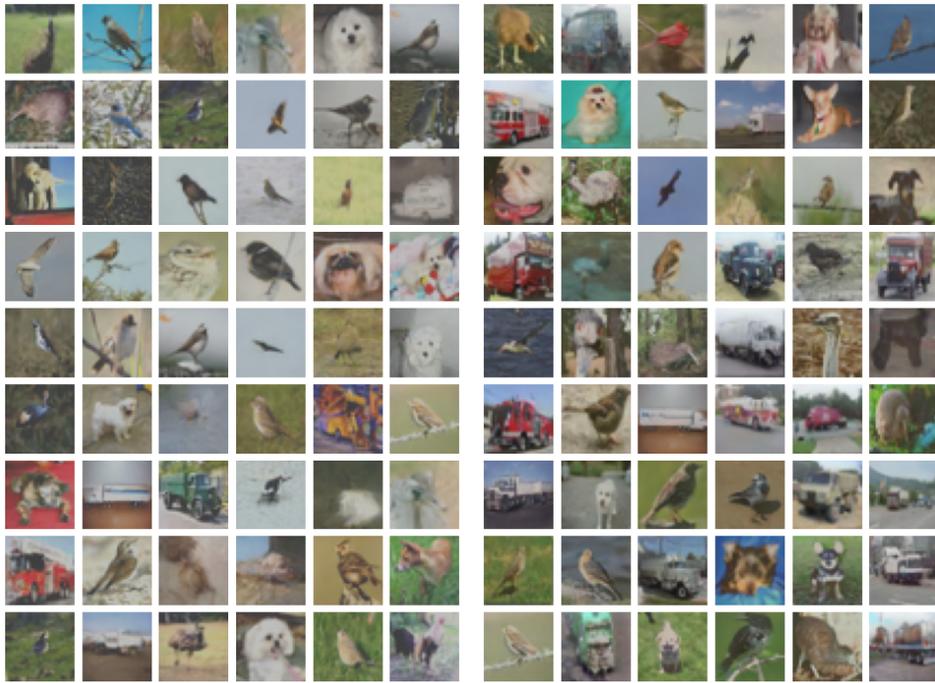


(c) SGM

(d) mixSGM

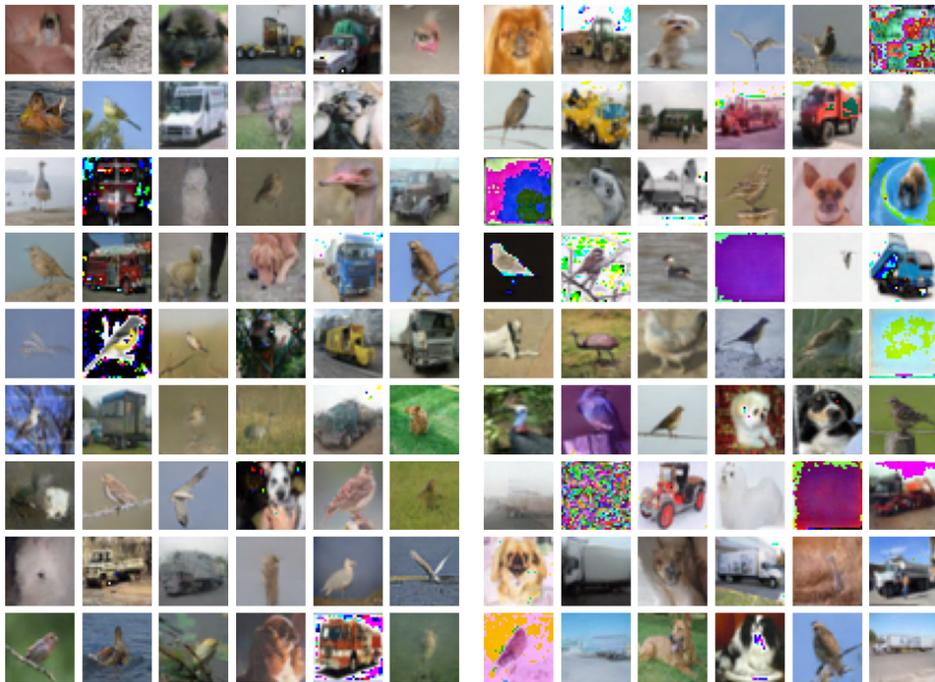
Figure 18: Experiments on CIFAR10 with 540k training steps

1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673



(a) DDPM

(b) mixDDPM



(c) SGM

(d) mixSGM

Figure 19: Experiments on CIFAR10 with 600k training steps