Generalization Bound for Diffusion Models using a Adaptive Random Features (DARF)

Esha Saha[1*], **Giang Tran**[1]

[1]University of Waterloo, Canada
esaha@uwaterloo.ca, giang.tran@uwaterloo.ca

Abstract

Diffusion probabilistic models have been successfully used to generate high resolution data from noise. However, limited interpretability and computationally expensive implementation renders most existing diffusion models unsuitable for applications where the goal is to achieve theoretically accurate results with a fair trade-off on the quality of generated samples. In this work, we propose an adaptive random feature architecture for training a diffusion model that can generate samples from the input distribution using limited parameters, and theoretically quantify the generalization bounds. Our results demonstrate the possibility of building interpretable diffusion model architectures. We validate the obtained theory using numerical results, which perform better (or comparably) to an U-Net architecture having same number of trainable parameters.

Code — https://github.com/esha-saha/darf

Introduction

Generative modeling has been successfully used to generate a wide variety of data. Some well-known models are Generative Adversarial Networks (Goodfellow et al. 2014; Dinh, Sohl-Dickstein, and Bengio 2017; Kingma and Welling 2014), flow-based models (Ho et al. 2019; Li et al. 2023a; Zhang and Chen 2021), energy-based modeling and score matching (Song and Ermon 2020; Jolicoeur-Martineau et al. 2021; Jolicoeur-Martineau et al. 2021), autoregressive models (Menick and Kalchbrenner 2019; Hoogeboom et al. 2022; Kingma et al. 2021), and variational autoencoders (Kingma and Welling 2014; Kingma, Welling et al. 2019; Pu et al. 2016; Xu et al. 2017). Language Modeling (LLMs) is an example of how generative models can be both powerful and computationally intensive at the same time (Chang et al. 2024; Zheng et al. 2025). Diffusion models are one such class of generative models that give exemplary performance in terms of data generation. A diffusion probabilistic model (or "diffusion model") is a parameterized model that is trained using variational inference to generate samples matching the data from input distribution after a finite number of timesteps. Although diffusion models can generate high quality samples, they are extremely complex and computationally intensive. Thus for applications where theoretical guarantee of the generated samples are more critical than the quality/accuracy (for example, generating maps of various climatic conditions in a region based on mesoscale climate models), existing models are an unfavorable choice. In order to get a deeper understanding of the generalization bounds of diffusion models, we propose a training process based on random features (DARF Model or Diffusion-Adaptive Random Feature Model) with adaptive weights for each timestep that can generate samples from the input distribution using less than half the number of trainable parameters in comparison to contemporary architectures. The proposed method is inspired by semi-random features (Kawaguchi, Xie, and Song 2018) and *Denoising Diffusion Probabilistic Models* (DDPMs) as formulated in (Ho, Jain, and Abbeel 2020). We propose a diffusion model-inspired adaptive random feature model, that uses adaptive random features to learn the reverse process in the diffusion models. Our main contributions are given below:

- By using a random feature model architecture for each diffusion timestep, we derive approximation bounds on samples generated by our framework and show that it is possible to stabilize model training with limited number of random features.
- We numerically validate the theoretical results by demonstrating the guaranteed convergence of the learned distribution to the true input distribution.

For demonstration purposes and rigor, results for numerically validating our proposed architecture and the theoretical bounds is given in the Appendix.

Background and Related Works

We first recall some useful notations and terminologies corresponding to diffusion models (Ho, Jain, and Abbeel 2020; Song and Ermon 2019). In this paper, we denote $\mathcal{N}(\mathbf{0}, \mathbf{I})$ the $d-$ dimensional Gaussian distribution with the zero mean vector and the identity covariance matrix. We say that a function $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ to mean that $p(\mathbf{x})$ is the p.d.f. of a random vector $\mathbf{x}$ following the multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. A diffusion model consists of two Markov chains: a forward process and a reverse process, summarized below.

[*]Current affiliation: Postdoctoral Researcher, University of Alberta, Canada.

The Forward and Reverse Processes

For K timesteps, the forward process degrades the input data such that $q(\mathbf{x}_K) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. More precisely, let $\mathbf{x}_0 \in \mathbb{R}^d$ be an input from an unknown distribution with p.d.f. $q(\mathbf{x}_0)$. Given a variance schedule $0 < \beta_1 < \beta_2 < ... < \beta_K < 1$, the forward process to obtain a degraded sample for a given timestep is defined as:

$$\mathbf{x}_{k+1} = \sqrt{1-\beta_{k+1}}\mathbf{x}_k + \sqrt{\beta_{k+1}}\boldsymbol{\epsilon}_k \tag{1}$$

where $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $k = 0, \ldots, K-1$. The forward process generates a sequence of random variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K$ with conditional distributions

$$q(\mathbf{x}_{k+1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{1-\beta_{k+1}}\mathbf{x}_k, \beta_{k+1}\mathbf{I}). \tag{2}$$

For $k = 1, \ldots, K$, let $\alpha_k = 1 - \beta_k$ and $\overline{\alpha}_k = \prod_{i=1}^{k} \alpha_i$. Then the conditional distribution is

$$q(\mathbf{x}_{k+1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{k+1}; \sqrt{\overline{\alpha}_{k+1}}\,\mathbf{x}_0, (1-\overline{\alpha}_{k+1})\mathbf{I}). \tag{3}$$

Hence, $q(\mathbf{x}_K) = \int q(\mathbf{x}_K|\mathbf{x}_0)q(\mathbf{x}_0)d\mathbf{x}_0 \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$, i.e., as the number of timesteps becomes very large, the distribution $q(\mathbf{x}_K)$ will approach the Gaussian distribution with mean $\mathbf{0}$ and covariance $\mathbf{I}$.

The reverse process aims to generate data from the input distribution by sampling from $q(\mathbf{x}_K)$ and gradually denoising for which one needs to know the reverse distribution $q(\mathbf{x}_{k-1}|\mathbf{x}_k)$. In general, computation of $q(\mathbf{x}_{k-1}|\mathbf{x}_k)$ is intractable without the knowledge of $\mathbf{x}_0$. Therefore, we condition the reverse distribution on $\mathbf{x}_0$ in order to obtain the mean and variance for the reverse process. The probability density function of the reverse distribution conditioned on $\mathbf{x}_0$ is also a Gaussian distribution with mean vector $\tilde{\boldsymbol{\mu}}_k$ and covariance matrix $\tilde{\beta}_k\mathbf{I}$ as defined in (Ho, Jain, and Abbeel 2020). Our aim is to learn the reverse distribution from the obtained conditional reverse distribution. From Markovian theory, if β_k's are small, the reverse process is also Gaussian (Sohl-Dickstein et al. 2015). Suppose $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k)$ be the learned reverse distribution, then Markovian theory tells us that $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}_k, k), \boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k))$, where $\boldsymbol{\mu}_\theta(\mathbf{x}_k, k)$ and $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)$ are the learned mean vector and covariance matrix respectively. Since the derived covariance matrix $\tilde{\beta}_k\mathbf{I}$ for conditional reverse distribution is constant, $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k)$ need not be learnt. In (Ho, Jain, and Abbeel 2020), the authors show that choosing $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, t)$ as $\beta_k\mathbf{I}$ or $\tilde{\beta}_k\mathbf{I}$ yield similar results and thus we fix $\boldsymbol{\Sigma}_\theta(\mathbf{x}_k, k) = \beta_k\mathbf{I}$ for simplicity. Since $\mathbf{x}_k$ is an input to the model, the loss function which minimizes the ℓ_2 norm between the true and learned mean of the reverse distribution (as derived in (Ho, Jain, and Abbeel 2020)) can be simplified to the ℓ_2 norm between the model output and noise added at each timestep. Thus, the trained model becomes a noise predictor whose outputs give the noise $\boldsymbol{\epsilon}_k$ added at each timestep. In DARF, we choose our model to be the noise predictor as it is one of the simplest model forms to train and understand. Furthermore, in order to derive generalization bound for our model, we use the Stochastic Differential Equations (SDEs) formulation of diffusion models which was shown to be equivalent to DDPMs through Denoising Score Matching (DSMs) in (Block, Mroueh, and Rakhlin 2020).

Random Feature Models

Let $f : \mathbb{R}^d \to \mathbb{C}$ be a given unknown function. Then the random feature approximation of f is

$$f(\mathbf{x}) \approx \mathbf{c}^T\xi(\mathbf{x}) = \mathbf{c}^T\phi(\mathbf{W}^T\mathbf{x} + \mathbf{b}), \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input data, ξ is the feature map defined through a random matrix $\mathbf{W} \in \mathbb{R}^{d\times N}$, bias vector $\mathbf{b} \in \mathbb{R}^N$ and a pre-defined non-linear function ϕ, and $\mathbf{c} \in \mathbb{C}^N$ is the final (learnt) weight layer. We assume that the entries of the matrix $\mathbf{W} = [\boldsymbol{\omega}_{j,k}]$ are independent and identically distributed (i.i.d.) random variables generated by the (user-defined) probability density function $\rho(\boldsymbol{\omega})$ i.e. $\boldsymbol{\omega}_{j,k} \sim \rho(\boldsymbol{\omega})$ for all $1 \le j \le d$ and $1 \le k \le N$. Similarly, the components of the bias vector $\mathbf{b}$ are also sampled i.i.d from a predefined distribution. The output layer $\mathbf{c}$ is trained, while the hidden layer $\mathbf{W}$ is fixed. The main idea of random feature methods is to transform the input with the map ϕ and apply linear methods to approximate the solution of a non-linear kernel machine. The random feature maps help in the quick evaluation of the machine.

Since distribution learning and data generation is a complex task, it is unsurprising that conventional diffusion models are computationally expensive. From previous works in (Ho, Jain, and Abbeel 2020; Yang et al. 2024; Zhang and Chen 2021), U-Net (or variations on U-Net combined with ResNet, CNNs, transformers, etc.) architecture is till date the most commonly used model for diffusion models. U-Nets not only preserve the dimension of input data, they also apply techniques such as downsampling using convolution which helps to learn the features if the input data. However, all these architectures have at least 100 million parameters making training (and sampling) cumbersome. While there are remarkable results for improving training and sampling for diffusion models, little has been explored in terms of theoretical guarantees of the model architectures. Some notable works that derive convergence bounds under multiple assumptions can be found in (Chen et al. 2024; Lyu, Chen, and Feng 2024; Conforti, Durmus, and Silveri 2025; Stéphanovitch, Aamari, and Levrard 2025; Li et al. 2023b). Although each of the works provides a unique way of proving convergence, they are all based on the major assumption that the network architecture converges. Although it is numerically evident, none of the existing models used for training can be studied for interpretability due to their complexity and limited understanding of how these models learn. An alternative approach to reduce the complexity of machine learning algorithms is to use cheaper function representation methods such as random feature models (RFM) (Rahimi and Recht 2007, 2008a) to approximate the kernels using a random basis. RFMs are derived from widely used kernel-based methods that utilize a predefined nonlinear function basis called kernel $K(\mathbf{x}, \mathbf{y})$ (Cervantes et al. 2020; Campbell 2002; Meyer, Leisch, and Hornik 2003). From the neural network point of view, an RFM is a two-layer network with a fixed single hidden layer randomly sampled (Rahimi and Recht 2007, 2008a). Not only do random feature-based

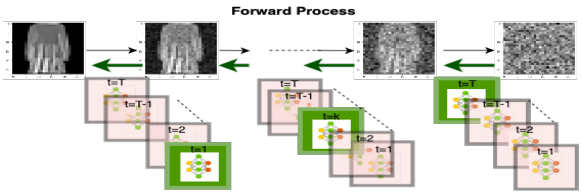


Figure 1: Representation of the proposed model. The green boxes denote the random feature layer which is active corresponding to the timestep selected, while the others remain fixed. The weights associated with each green RFM are also optimized during training.

methods give similar results to that of a shallow network, but the model in itself is also interpretable, which makes it a favorable method for various deep learning applications. However, random features can lack expressiblity due to their structure and thus we aim to propose an architecture that can be more flexible in learning yet retaining properties of random features. Our main goal is to build a diffusion model can be useful for applications where computational resources are limited and theoretical guarantees are critical, even if it comes at the cost of reduced accuracy.

DARF: Architecture and Algorithm

Our work is inspired by *denoising diffusion probabilistic model* (DDPM) proposed in (Ho, Jain, and Abbeel 2020) and the semi-random features proposed in (Kawaguchi, Xie, and Song 2018). Using the formulation from (Ho, Jain, and Abbeel 2020), we parameterize the added noise as a random feature-based network. The trainable parameters are optimized to approximate the noise added at each timestep such that during sample generation the parameterized model can be used to remove noise and generate samples from the input distribution. Let $\mathbf{x}_0 \in \mathbb{R}^d$ be the input data belonging to an unknown distribution $q(\mathbf{x}_0)$. Let K denote the total number of timesteps in which the forward process is applied. Suppose N is the number of features. For each timestep k, we build a noise predictor function p_θ of the form

$$p_\theta(\mathbf{x}_k, k) = (\sin(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}^T) \odot \cos(\boldsymbol{\tau}_k^T\boldsymbol{\theta}^{(1)}))\boldsymbol{\theta}^{(2)}, \quad (5)$$

where $\mathbf{x}_k \in \mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d\times N}$, $\mathbf{b} = [b_1 \quad \dots \quad b_N]^T \in \mathbb{R}^N$, $\boldsymbol{\theta}^{(1)} = \left[\theta_{ki}^{(1)}\right] \in \mathbb{R}^{K\times N}$, $\boldsymbol{\tau}_k \in \mathbb{R}^K$, $\boldsymbol{\theta}^{(2)} = \left[\theta_{ij}^{(2)}\right] \in \mathbb{R}^{N\times d}$, and $\odot$ denotes element-wise multiplication. The vector $\boldsymbol{\tau}_k\,(k \geq 1)$ is a one-hot vector with the position of one corresponding to the timestep k. The trainable weights $\boldsymbol{\theta}^{(1)}$ are what makes DARF an adaptive model as it learns to associate importance to the timestep (and hence level of noise) during loss optimization. The inspiration for using cosine as an activation for the adaptive layer comes from the idea of positional encoding used for similar tasks (Vaswani et al. 2017). Positional encodings generally remain fixed, but for our method, we wish to make the weights associated with each timestep adaptive i.e., trainable. This is done so that the model learns the noise level associated with the timestep. Our aim is to train the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}\}$ (while $\mathbf{W}$ and $\mathbf{b}$ are randomly sampled and fixed) by minimizing the simplified loss derived from KL Divergence between the learned and the true input distribution i.e., $L = \frac{1}{K}\sum_{k=1}^{K}\left\|\boldsymbol{\epsilon}_k - p_\theta(\mathbf{x}_k, k)\right\|_2^2$, where $\boldsymbol{\epsilon}_k$ denotes the noise added at each timestep. The training of diffusion models using adaptive random features offers multiple advantages. Firstly, the positional time encoding allows the model to learn the level of noise at each timestep allowing for effective denoising during sample generation. Secondly, the model can learn the input distributions using only half the number of trainable parameters (although more parameters may be required for very high resolution samples). And thirdly, the approximation bounds allow us to choose model parameters in a way to ensure stable training and distribution learning. The pseudocode is given in the Appendix.

In order to derive sampling bounds for our model, we first formulate our proposed model as an adaptive (time encoded) random features model. For a fixed timestep k, Eq. (5) can be written as:

$$\begin{aligned} p_\theta(\mathbf{x}_k, k) &= \left(\sin\left([y_1,\dots,y_d]\begin{bmatrix}\omega_{11} & \dots & \omega_{1N}\\ \vdots & \ddots & \vdots\\ \omega_{d1} & \dots & \omega_{dN}\end{bmatrix} + \begin{bmatrix}b_1\\ \vdots\\ b_N\end{bmatrix}\right)\right.\\ &\quad \left.\odot \begin{bmatrix}\cos(\theta_{k1}^{(1)}) & \dots & \cos(\theta_{kN}^{(1)})\end{bmatrix}\right)\boldsymbol{\theta}^{(2)}\\ &= \sin(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}^T)\begin{bmatrix}\cos(\theta_{k1}^{(1)})\theta_{11}^{(2)} & \dots & \cos(\theta_{k1}^{(1)})\theta_{1d}^{(2)}\\ \vdots & \dots & \vdots\\ \cos(\theta_{kN}^{(1)})\theta_{N1}^{(2)} & \dots & \cos(\theta_{kN}^{(1)})\theta_{Nd}^{(2)}\end{bmatrix}. \end{aligned}$$

For each $j = 1, ..., N$, let $a_j = \sin\left(\sum_{i=1}^{d} y_i\omega_{ij} + b_j\right)$. Note a_j's are fixed. Then Eq. (6) becomes,

$$p_\theta(\mathbf{y}, k) = \begin{bmatrix}a_1\cos(\theta_{k1}^{(1)}) & \dots & a_N\cos(\theta_{kN}^{(1)})\end{bmatrix}\begin{bmatrix}\theta_{11}^{(2)} & \dots & \theta_{1d}^{(2)}\\ \vdots & \dots & \vdots\\ \theta_{N1}^{(2)} & \dots & \theta_{Nd}^{(2)}\end{bmatrix} \quad (6)$$

$$= [a_1 \quad \dots \quad a_N]\begin{bmatrix}\cos(\theta_{k1}^{(1)})\theta_{11}^{(2)} & \dots & \cos(\theta_{k1}^{(1)})\theta_{1d}^{(2)}\\ \vdots & \dots & \vdots\\ \cos(\theta_{kN}^{(1)})\theta_{N1}^{(2)} & \dots & \cos(\theta_{kN}^{(1)})\theta_{Nd}^{(2)}\end{bmatrix} \quad (7)$$

Thus, for each fixed time k, our proposed architecture is a random feature model with a fixed dictionary having N features denoted by $\phi(\langle\mathbf{x}_k, \boldsymbol{\omega}_i\rangle) = \sin(\mathbf{x}_k^T\boldsymbol{\omega}_i + b_i)$, $\forall i = 1, \dots, N$ and learnt coeeficients $\mathbf{C} = (c_{ij}) \in \mathbb{R}^{N\times d}$ whose entries are

$$c_{ij} = \cos(\boldsymbol{\theta}_{ki}^{(1)})\boldsymbol{\theta}_{ij}^{(2)}, \ \forall i = 1, \dots, N; j = 1, \dots, d.$$

As depicted in Figure 1, DARF can be visualized as K random feature models stacked up in a column. Each random feature model has associated weights corresponding to its

timestep which also gets optimized implicitly while training. This reformulation shows that for each step, the model follows a weighted random feature architecture, with similar generalization guarantees. Using this reformulation of DARF, we prove two lemmas to show that the class of functions generated by our model is the same as the class of functions approximated by random feature models, thus quantifying the approximation error for vector-valued functions. The results are used directly in the main theorem which quantifies the error between the true and learned distribution using DARF.

Theoretical Results

In this section, we present two lemmas, followed by a theorem which comprises of the main theoretical result(s) for the proposed architecture. Note that while the notations $\phi(\mathbf{x}^T\boldsymbol{\omega}_j)$, $\phi(\langle x, \boldsymbol{\omega}_j\rangle)$ and $\phi(\mathbf{x};\boldsymbol{\omega}_j)$ might not be equivalent in general, in this paper they are used interchangeably depending on the context.

Lemma 0.1. *Let $\mathcal{G}_{k,\boldsymbol{\omega}}$ denote the set of functions that can be approximated by DARF at each timestep k defined as*

$$\mathcal{G}_{k,\boldsymbol{\omega}} = \left\{ \boldsymbol{g}(\mathbf{x}) = \sum_{j=1}^{N} \cos(\boldsymbol{\theta}_{kj}^{(1)})\boldsymbol{\theta}_j^{(2)}\phi(\mathbf{x}_k^T\boldsymbol{\omega}_j) \middle| \left\|\boldsymbol{\theta}_j^{(2)}\right\|_\infty \leq \frac{C}{N} \right\} \tag{8}$$

and

$$\mathcal{F}_{\boldsymbol{\omega}} = \left\{ \boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \boldsymbol{\alpha_j}\,\phi(\mathbf{x}^T\boldsymbol{\omega}_j) \middle| \|\boldsymbol{\alpha_j}\|_\infty \leq \frac{C}{N} \right\}. \tag{9}$$

Then for a fixed k, $\mathcal{G}_{k,\boldsymbol{\omega}} = \mathcal{F}_{\boldsymbol{\omega}}$.

Proof. The above equality can be proved easily.
Fix k. Consider $\boldsymbol{g}(\mathbf{x}) \in \mathcal{G}_{k,\boldsymbol{\omega}}$. Then $\boldsymbol{g}(\mathbf{x}) = \sum_{j=1}^{N} \cos(\boldsymbol{\theta}_{kj}^{(1)})\boldsymbol{\theta}_j^{(2)}\phi(\mathbf{x}_k^T\boldsymbol{\omega}_j)$.
Clearly, $\boldsymbol{g}(\mathbf{x}) \in \mathcal{F}_{\boldsymbol{\omega}}$ as $\left\|\cos(\boldsymbol{\theta}_{kj}^{(1)})\boldsymbol{\theta}_j^{(2)}\right\|_\infty \leq \left\|\boldsymbol{\theta}_j^{(2)}\right\|_\infty \leq \frac{C}{N}$. Thus $\mathcal{G}_{k,\boldsymbol{\omega}} \subseteq \mathcal{F}_{\boldsymbol{\omega}}$.
Conversely, let $\boldsymbol{f}(\mathbf{x}) \in \mathcal{F}_{\boldsymbol{\omega}}$, then $\boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \boldsymbol{\alpha_j}\,\phi(\mathbf{x}_k^T\boldsymbol{\omega}_j)$.
Choose $\boldsymbol{\theta}_j^{(2)} = \boldsymbol{\alpha}_j$ and $\boldsymbol{\theta}_{kj}^{(1)} = [0, \cdots, 0]$.
As $\cos(\boldsymbol{\theta}_{kj}^{(1)})\boldsymbol{\theta}_j^{(2)} = \boldsymbol{\alpha}_j$, thus $\boldsymbol{f}(\mathbf{x}) = \sum_{j=1}^{N} \cos(\boldsymbol{\theta}_{kj}^{(1)})\boldsymbol{\theta}_j^{(2)}\phi(\mathbf{x}^T\boldsymbol{\omega}_j)$ and $\|\boldsymbol{\theta}_j^{(2)}\|_\infty = \|\boldsymbol{\alpha}_j\|_\infty \leq \frac{C}{N}$.
Hence $\boldsymbol{f}(\mathbf{x}) \in \mathcal{G}(k, \boldsymbol{\omega})$ and $\mathcal{F}_{\boldsymbol{\omega}} \subseteq \mathcal{G}_{k,\boldsymbol{\omega}}$. □

Lemma 0.2. *Let $X \subset \mathbb{R}^d$ denote the training dataset and suppose q is a measure on X, and $\boldsymbol{f}^*$ a function in $\mathcal{F}_\rho$ where*

$$\mathcal{F}_\rho = \left\{ \boldsymbol{f}(\mathbf{x}) = \int_\Omega \boldsymbol{\alpha}(\boldsymbol{\omega})\phi(\mathbf{x};\boldsymbol{\omega})\, d\boldsymbol{\omega} \;\middle|\; \|\boldsymbol{\alpha}(\boldsymbol{\omega})\|_\infty \leq C\rho(\boldsymbol{\omega}) \right\}.$$

If $[\boldsymbol{\omega}_j]_{j\in[N]}$ are drawn iid from ρ, then for $\delta > 0$, with probability at least $1-\delta$ over $[\boldsymbol{\omega}_j]_{j\in[N]}$, there exists a function $\boldsymbol{f}^\sharp \in \mathcal{F}_{\boldsymbol{\omega}}$ such that

$$\|\boldsymbol{f}^\sharp - \boldsymbol{f}^*\|_2 \leq \frac{C\sqrt{d}}{\sqrt{N}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right), \tag{10}$$

where $\mathcal{F}_{\boldsymbol{\omega}}$ is defined in Eq. (9)*.*

Proof. We follow the proof technique described in (Rahimi and Recht 2008b). As $\boldsymbol{f}^* \in \mathcal{F}_\rho$, then $\boldsymbol{f}^*(\mathbf{x}) = \int_{\boldsymbol{\Omega}} \boldsymbol{\alpha}(\boldsymbol{\omega})\phi(\mathbf{x};\boldsymbol{\omega})d\boldsymbol{\omega}$. Construct $\boldsymbol{f}_k = \boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)$, $k = 1, \cdots, N$ such that $\boldsymbol{\beta}_k = \frac{\boldsymbol{\alpha}(\boldsymbol{\omega}_k)}{\rho(\boldsymbol{\omega}_k)} = \frac{1}{\rho(\boldsymbol{\omega}_k)}\begin{bmatrix}\alpha_1(\boldsymbol{\omega}_k)\\ \vdots \\ \alpha_d(\boldsymbol{\omega}_k)\end{bmatrix}$.
Note that $\mathbb{E}_{\boldsymbol{\omega}}(\boldsymbol{f}_k) = \int_{\boldsymbol{\omega}} \frac{\boldsymbol{\alpha}(\boldsymbol{\omega}_k)}{\rho(\boldsymbol{\omega}_k)}\phi(\mathbf{x};\boldsymbol{\omega})\rho(\boldsymbol{\omega}_k)d\boldsymbol{\omega} = \boldsymbol{f}^*$.
Define the sample average of these functions as $\boldsymbol{f}^\sharp(\mathbf{x}) = \sum_{k=1}^{N} \frac{\boldsymbol{\beta}_k}{N}\phi(\mathbf{x};\boldsymbol{\omega}_k)$.
As $\left\|\frac{\boldsymbol{\beta}_k}{N}\right\|_\infty \leq \frac{C}{N}$, thus $\boldsymbol{f}^\sharp \in \mathcal{F}_{\boldsymbol{\omega}}$. Also note that $\|\boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)\|_2 \leq \sqrt{d}\|\boldsymbol{\beta}_k\phi(\cdot;\boldsymbol{\omega}_k)\|_\infty \leq \sqrt{d}C$.
In order to get the desired result, we use McDiarmid's inequality. Define a scaler function on $F = \{\boldsymbol{f}_1, \cdots, \boldsymbol{f}_N\}$ as $g(F) = \|\boldsymbol{f}^\sharp - \mathbb{E}_F\boldsymbol{f}^\sharp\|_2$. We claim that the function g is stable under perturbation of its ith argument.
Define $\tilde{F} = \{\boldsymbol{f}_1, \cdots, \tilde{\boldsymbol{f}}_i, \cdots, \boldsymbol{f}_N\}$ i.e., $\tilde{F}$ differs from F only in its ith element. Then

$$\left|g(F) - g(\tilde{F})\right| = \left|\left\|\boldsymbol{f}^\sharp - \mathbb{E}_F\boldsymbol{f}^\sharp\right\|_2 - \left\|\tilde{\boldsymbol{f}}^\sharp - \mathbb{E}_{\tilde{F}}\boldsymbol{f}^\sharp\right\|_2\right| \leq \left\|\boldsymbol{f}^\sharp - \tilde{\boldsymbol{f}}^\sharp\right\|_2 \tag{11}$$

where the above inequality is obtained from triangle inequality. Further,

$$\left\|\boldsymbol{f}^\sharp - \tilde{\boldsymbol{f}}^\sharp\right\|_2 = \frac{1}{N}\left\|\boldsymbol{f}_i - \tilde{\boldsymbol{f}}_i\right\|_2 \leq \left\|(\boldsymbol{\beta}_i - \tilde{\boldsymbol{\beta}}_i)\phi(\cdot;\boldsymbol{\omega})\right\|_2 \leq \frac{\sqrt{d}C}{N}. \tag{12}$$

Thus $\mathbb{E}[g(F)^2] = \frac{1}{N}\left[\mathbb{E}\left[\left\|\sum_{k=1}^{N}\boldsymbol{f}_k\right\|_2^2\right] - \left\|\mathbb{E}\left[\sum_{k=1}^{N}\boldsymbol{f}_k\right]\right\|_2^2\right]$.
Since $\|\boldsymbol{f}_k\|_2 \leq \sqrt{d}C$, using Jensen's inequality and above result we get

$$\mathbb{E}[g(F)] \leq \sqrt{\mathbb{E}(g^2(F))} \leq \frac{\sqrt{d}C}{\sqrt{N}}. \tag{13}$$

Finally the required bounds can be obtained by combining above result and McDiarmid's inequality. □

We extend the results from (Chen et al. 2023) in order to get our main theorem given below.

Theorem 0.3. *For a given probability density $q(\mathbf{x}_0)$ on $\mathbb{R}^d$ suppose the following conditions hold:*

1. *For all $t \geq 0$, the score $\nabla \log q(\mathbf{x}(t))$ is $L-$Lipschitz.*

2. *The second moment of* $q(\mathbf{x}_0)$ *is finite i.e.,* $m_2^2 = E_{q(\mathbf{x}_0)}[\|.\|^2] < \infty$.

Let $p_\theta(\mathbf{x}_0, 0)$ *be the sample generated by DARF after* K *timesteps at terminal time* T*. Then for the SDE formulation of the DDPM algorithm, if the step size* $h := T/K$ *satisfies* $h \lesssim 1/L$*, where* $L \geq 1$*. Then for* $\delta > 0$*, with probability at least* $1 - \delta$ *over* $[\boldsymbol{\omega}_j]_{j\in[N]}$*,*

$$TV(p_\theta(\mathbf{x}_0, 0), q(\mathbf{x}_0)) \lesssim \sqrt{KL(q(\mathbf{x}_0)\|\gamma)}\exp(-T) + (L\sqrt{dh} + Lm_2h)\sqrt{T} + \frac{C_2\sqrt{TKd}}{\sqrt{N}}\left(1 + \sqrt{2\log\frac{1}{\delta}}\right)$$

where $C_2 \geq \max_{1\leq i\leq N, 1\leq j\leq d} \left|\boldsymbol{\theta}_{ij}^{(2)}\right|$ *and* γ *is the p.d.f. of the multivariate normal distribution with mean vector* $\mathbf{0}$ *and covariance matrix* $\mathbf{I}$*.*

Proof. Replace the third error term in Theorem 1 in (Chen et al. 2023) with bounds obtained for DARF using Lemma 0.1 and Lemma 0.2. □

The error bound given in Theorem 0.3 consists of three types of errors: (i) convergence error of the forward process; (ii) discretization error of the associated SDE with step size $h > 0$; and (iii) score estimation error, respectively. Note that the first two terms are independent of the model architecture. The third term in the error explicitly quantifies the bound for the model being used, which is in general hard to estimate for most architectures. From (Chen et al. 2023), we can choose appropriate T and h such that the first two terms become negligible. Furthermore, if we choose a very high number of features i.e., as $N \to \infty$ to be large enough, the third term goes to zero. Quantifying the error bound explicitly gives us a better understanding of the training process, and guides us on the number of trainable features N required for guaranteed convergence to the true distribution.

Discussion and Conclusion

In this paper, we proposed an interpretable training regime for diffusion models using adaptive random features and quantified upper bounds on the samples generated with respect to the input distribution. Since the theoretical results are core to the paper, details of numerical validation are given in the Appendix. To summarize, we trained DARF on both image (a subset of fMNIST dataset) and audio data for two tasks: (i) generating data from noise and; (ii) denoising data corrupted with Gaussian noise; and compared our method with a fully connected version of DARF (denoted by NN) where we trained $[\mathbf{W}, \mathbf{b}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}]$ while preserving the number of trainable parameters, a simple U-Net model and a classical random feature approach with only $\boldsymbol{\theta}^{(2)}$ being trainable (denoted by RF). As suggested by Theorem 0.3, we chose the number of features N to be greater than Kd (we omit T in this computation since the choice of this parameter is as suggested in (Chen et al. 2023)). Comparing the generated figures (and audio) with the true input dataset, we found DARF produce samples closer to the input distribution than other architecture. Moreover, when FID scores were calculated (for the sake of quantitative comparison only), DARF was found to have the lowest scores for very limited training data and scarce computational resources. It is important to note that while there may exist multiple methods that give improved sample quality or tighter approximation bounds, none of the previous works demonstrate any possibility of building a diffusion model that can quantify the bounds for the model architecture nor perform under data scarcity conditions. The DARF model however is limited to lower dimensional distributions due to its curse of dimensionality where the number of trainable features need to exceed the product of input dimension and the number of timesteps. An extension of DARF to generate higher resolution data using high dimensional distribution is left as a future work. Other future directions also involve extending DARF beyond its shallow nature into deeper architectures.

References

Block, A.; Mroueh, Y.; and Rakhlin, A. 2020. Generative Modeling with Denoising Auto-Encoders and Langevin Sampling. *CoRR*, abs/2002.00107.

Campbell, C. 2002. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4): 63–84.

Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; and Lopez, A. 2020. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408: 189–215.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3): 1–45.

Chen, M.; Mei, S.; Fan, J.; and Wang, M. 2024. An overview of diffusion models: Applications, guided generation, statistical rates and optimization. *arXiv preprint arXiv:2404.07771*.

Chen, S.; Chewi, S.; Li, J.; Li, Y.; Salim, A.; and Zhang, A. 2023. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Conforti, G.; Durmus, A.; and Silveri, M. G. 2025. KL convergence guarantees for score diffusion models under minimal data assumptions. *SIAM Journal on Mathematics of Data Science*, 7(1): 86–109.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Ho, J.; Chen, X.; Srinivas, A.; Duan, Y.; and Abbeel, P. 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design.

In *International Conference on Machine Learning*, 2722–2730. PMLR.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33: 6840–6851.

Hoogeboom, E.; Gritsenko, A. A.; Bastings, J.; Poole, B.; van den Berg, R.; and Salimans, T. 2022. Autoregressive Diffusion Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jolicoeur-Martineau, A.; Li, K.; Piché-Taillefer, R.; Kachman, T.; and Mitliagkas, I. 2021. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*.

Jolicoeur-Martineau, A.; Piché-Taillefer, R.; Mitliagkas, I.; and des Combes, R. T. 2021. Adversarial score matching and improved sampling for image generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Kawaguchi, K.; Xie, B.; and Song, L. 2018. Deep semi-random features for nonlinear function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Kingma, D.; Salimans, T.; Poole, B.; and Ho, J. 2021. Variational diffusion models. *Advances in neural information processing systems*, 34: 21696–21707.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kingma, D. P.; Welling, M.; et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4): 307–392.

Li, J.; Cao, H.; Wang, J.; Liu, F.; Dou, Q.; Chen, G.; and Heng, P.-A. 2023a. Fast Non-Markovian Diffusion Model for Weakly Supervised Anomaly Detection in Brain MR Images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 579–589. Springer.

Li, P.; Li, Z.; Zhang, H.; and Bian, J. 2023b. On the generalization properties of diffusion models. *Advances in Neural Information Processing Systems*, 36: 2097–2127.

Lyu, J.; Chen, Z.; and Feng, S. 2024. Sampling is as easy as keeping the consistency: convergence guarantee for Consistency Models. In *Forty-first International Conference on Machine Learning*.

Menick, J.; and Kalchbrenner, N. 2019. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *International Conference on Learning Representations*.

Meyer, D.; Leisch, F.; and Hornik, K. 2003. The support vector machine under test. *Neurocomputing*, 55(1-2): 169–186.

Pu, Y.; Gan, Z.; Henao, R.; Yuan, X.; Li, C.; Stevens, A.; and Carin, L. 2016. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29.

Rahimi, A.; and Recht, B. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.

Rahimi, A.; and Recht, B. 2008a. Uniform approximation of functions with random bases. In *2008 46th annual allerton conference on communication, control, and computing*, 555–561. IEEE.

Rahimi, A.; and Recht, B. 2008b. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2256–2265. PMLR.

Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y.; and Ermon, S. 2020. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33: 12438–12448.

Stéphanovitch, A.; Aamari, E.; and Levrard, C. 2025. Generalization bounds for score-based generative models: a synthetic proof. arXiv:2507.04794.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xu, W.; Sun, H.; Deng, C.; and Tan, Y. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Yang, L.; Zhang, Z.; Song, Y.; Hong, S.; Xu, R.; Zhao, Y.; Zhang, W.; Cui, B.; and Yang, M. 2024. Diffusion Models: A Comprehensive Survey of Methods and Applications. *ACM Comput. Surv.*, 56(4): 105:1–105:39.

Zhang, Q.; and Chen, Y. 2021. Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34: 16280–16291.

Zheng, Y.; Chen, Y.; Qian, B.; Shi, X.; Shu, Y.; and Chen, J. 2025. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 57(8): 1–35.

Appendix

Pseudocode for Training DARF

Numerically Validated Results

In order to validate our findings, we train our proposed model on both image and audio data. We evaluate the validity of our model on two tasks: (i) generating data from noise and; (ii) denoising data corrupted with Gaussian noise. Note that the main goal of the proposed work is to explore training of diffusion models under data scarcity conditions

Algorithm 1: Training and sampling using DARF

Input: Sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ where q is an unknown distribution, variance schedule $\beta = \{\beta_1, ..., \beta_K\}$ such that $0 < \beta_1 < \beta_2 < ... < \beta_K < 1$, random weight matrix $\mathbf{W} = [w_{ij}]$ and bias vector $\mathbf{b}$ sampled from a distribution ρ and total number of epochs `epoch`.

Algorithm:

Training

Choose random timestep $k \in \{1, 2, ..., K\}$ and build vector $\boldsymbol{\tau}_k = [0, ...0, 1, 0, ..., 0]^T$ where 1 is in k^{th} position.

Define the forward process for $k = 1, 2, ..., K$ as

$$\mathbf{x}_k = \sqrt{\alpha_k}\mathbf{x}_{k-1} + \sqrt{1-\alpha_k}\boldsymbol{\epsilon}_k$$

where $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

for j in `epochs` **do**

  $k \sim \mathcal{U}\{1, 2, ..., K\}$.

  $p_\theta(\mathbf{x}_k, k) \leftarrow (\sin(\mathbf{x}_k^T\mathbf{W} + \mathbf{b}) \odot \cos(\boldsymbol{\tau}_k^T\boldsymbol{\theta}^{(1)}))\boldsymbol{\theta}^{(2)}$

  Update $\boldsymbol{\theta} = [\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}]$ by minimizing the loss

$$L = \frac{1}{K}\sum_{k=1}^{K}\left\|\boldsymbol{\epsilon}_k - p_\theta(\mathbf{x}_k, k)\right\|_2^2.$$

end for

Sampling

Sample a point $\mathbf{x}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $k = K-1, ..., 1$ **do**

  Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

  Find $\tilde{\mathbf{x}}_{k-1}$ using

$$\frac{1}{\sqrt{\alpha_k}}\left(\mathbf{x}_k - \frac{\sqrt{1-\alpha_k}}{\sqrt{1-\overline{\alpha}_k}}p_\theta(\mathbf{x}_k, k))\right) + (1-\alpha_k)\boldsymbol{\epsilon}$$

end for

Output: Generated sample $\tilde{\mathbf{x}}_0$

and lack of computational resources. The models can be trained with a basic Google Colab T4 GPU. We compare our method with a fully connected version of DARF (denoted by NN) where we train $[\mathbf{W}, \mathbf{b}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}]$ while preserving the number of trainable parameters, a simple U-Net model and a classical random feature approach with only $\boldsymbol{\theta}^{(2)}$ being trainable (denoted by RF). We create the image dataset (training) for our experiments by considering 100 images of size 28×28 taken separately from two classes of fMNIST dataset, namely 'dress' and 'shoes'. DARF is trained with 80000 random features. Note that the number of features $N = 80000$ is greater than $Kd = 100 \times 28 \times 28$ (we omit T in this computation since the choice of this parameter is as suggested in (Chen et al. 2023)). For NN, U-Net, and RF we adjust the number of trainable parameters to match that of DARF. We use 100 equally spaced timesteps for the forward diffusion process between 10^{-4} and 0.02 and train for 30000 epochs by minimizing the mean squared error (MSE) loss using ADAM optimizer with a learning rate of $\eta = 0.001$. For the task of generating images, we generated fifteen samples from randomly sampled noise. In our second task we give fifteen noise corrupted images from the same class (but not in the training set) test if our model can denoise the image. For audio data, we train the model using only two tunes played using a guitar and flute. We then test our model on tune generation and denoising tasks. Results from Figure 2

Figure 2: Figures generated from random noise when trained on 100 "dress" images. **Top left block:** DARF. **Top right block:** NN. **Bottom left block:** U-Net. **Bottom right block:** RF.

demonstrate that our method learns the overall features of the input distribution. Although the model is trained with a small size of training data (only one hundred images) and timesteps (one hundred timesteps), we can see that the samples generated from pure noise have already started to resemble the input distribution of dresses. The samples generated by the U-Net model also simply learns to generate the overall features of the distribution. The overall sample quality is not significantly better than the ones generated by DARF. For NN model we note that most of the generated samples are the same with a dark shadow while for RF

model, the generated images are very noisy and barely recognizable.

We also test our models ability to remove noise from images not present in the training dataset. We corrupt 15 randomly sampled images with 20% noise. The trained model is then used for denoising. In Figure 3 we can see that the model can recover a denoised image which is in fact better than the results obtained when sampling from pure noise. The U-Net and NN models performs quite well for most of the images. However for few cases with the NN model, it fails to denoise anything and the final image remains noisy, indicating a lack of robustness in generating data. RF model fails to denoise and the resultant images still contain noise. Table 1 gives the Fréchet Inception Distance (FID) calcu-

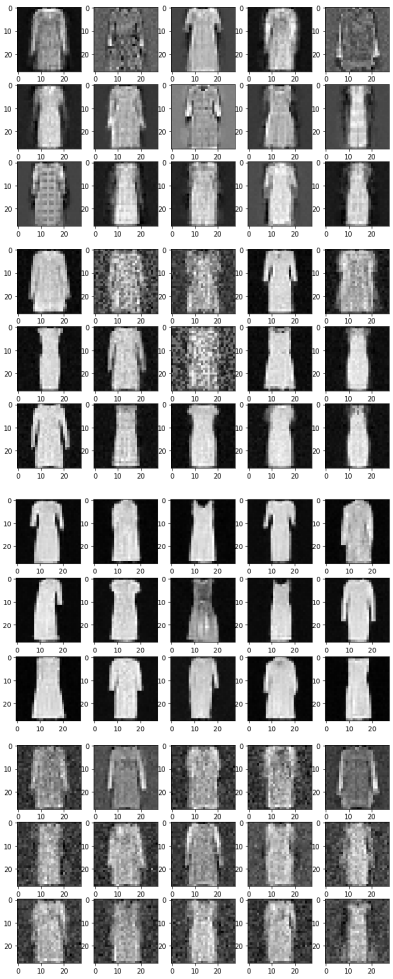


Figure 3: Figures denoised from images corrupted with 20 $\%$ noise.**Top left block:** DARF. **Top right block:** NN. **Bottom left block:** U-Net. **Bottom right block:** RF.

lated using a batch of 50 images from the training dataset and 15 of the generated images by each of the models. Note that the scores values given are for the sake of comparing the four models only. These values should not be taken as a measure of the models' best performance capability as training on more samples may be required to improve the FID scores (see results where we train our model using 1000 timesteps between 10^{-4} and 0.02 and see improved results). We see that for the generative task, the the proposed DARF architecture gives the lowest scores. The more commonly used U-Net model is at the third position after the NN model. Note that for U-Net, most images are denoised perfectly, some images are incomplete. The FID scores are also consistent with the qualitative assessment made in Figures 2.

Additional Results on Image Data

Table 1: FID scores for generative task for all the models trained on image data.

Model	Timesteps	FID Score
DARF	100	**453.87**
U-Net	100	463.28
NN	100	457.21
RF	100	470.12
DARF	1000	434.61
U-Net	1000	363.81

In order to check the effect of the number of timesteps on the sampling power of DARF, we also run our model using 1000 timesteps between 10^{-4} and 0.02. The images generated/denoised are given in Figure 4. The model performance seem to improve significantly only for the generative task and not the denoising task, which is expected as more reverse steps would be required to generate a point in the input distribution.

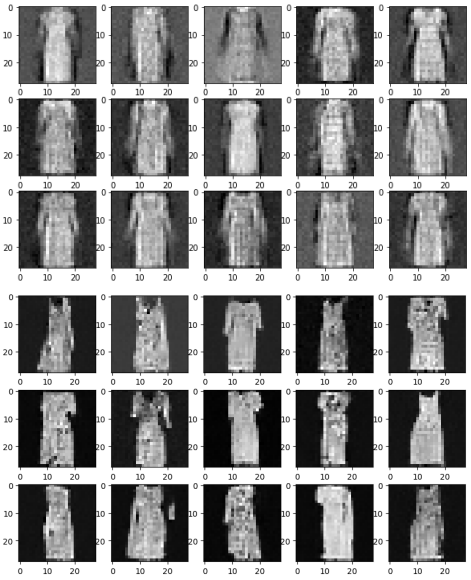


Figure 4: Generated images trained on 100 "dress" images with 1000 timesteps. **Left block:** DARF. **Right block:** U-Net.

We also include some additional experimental results for DARF trained with 1000 timesteps and on a different class of the fashion MNIST data as given below.

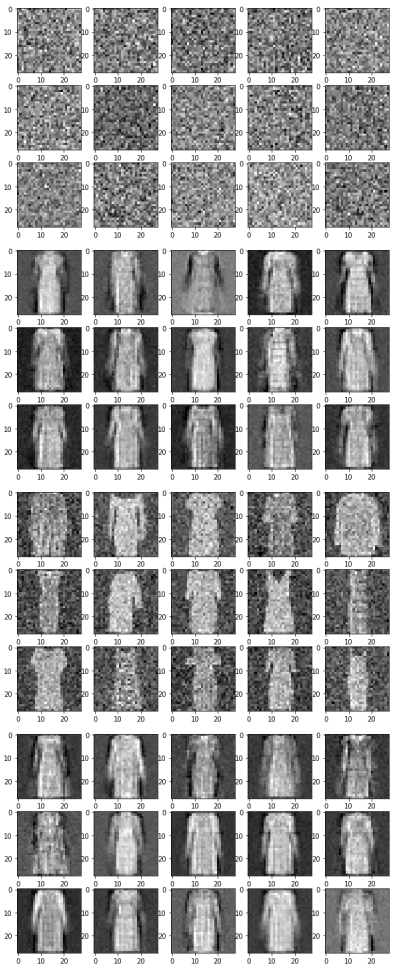


Figure 5: Generated and denoised images trained on 100 "dress" images with 1000 timesteps using DARF. Top row depicts samples generated from Gaussian noise. Bottom row depicts denoised images. Left column shows the inputs to the trained model. Right column shows the generated/denoised outputs using DARF.

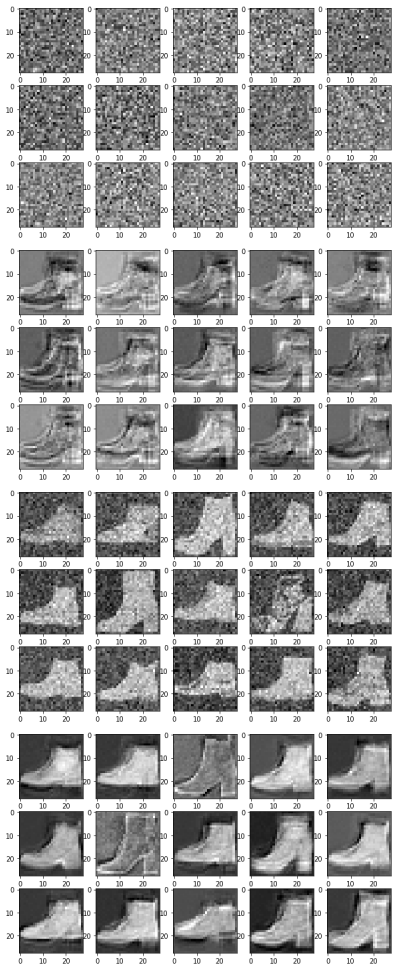


Figure 6: Samples generated from noise and noisy images when trained on 100 images of "shoes". Top row depicts samples generated from Gaussian noise. Bottom row depicts denoised images. Left column shows the inputs to the trained model. Right column shows the generated/denoised outputs using DARF.

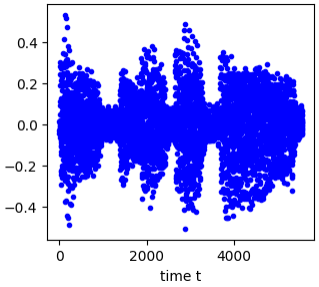
Additional Results on Audio Data

Figure 7 shows that when our trained model is applied to a noisy signal, our trained model can successfully recover and denoise the signal. This is more evident when the signal is played as an audio file. There are however some extra elements that get added while recovering due to the presence of noise which is a common effect of using diffusion models for denoising.

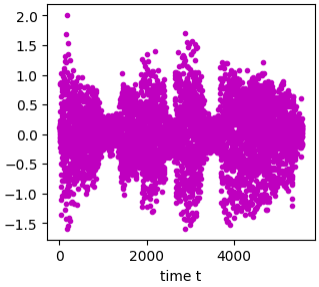
The table below shows the PSNR (peak signal-to-noise ratio) values for each method on denoising task. Our main aim is to explore if training all the parameters in a model actually benefits the learning process or not. Since the model is trained as a diffusion model, the denoised images might not be identical to the original one as can also be seen in (Sohl-Dickstein et al. 2015), leading to a low PSNR score. We calculate and report the scores simply to evaluate the three models on a simple quantitative metric. The table below shows that with the same trainable parameters, training all the layers barely benefits the model. Our proposed method is comparable to NN (slightly better even) in terms of PSNR values during denoising task. However, DARF is an interpretable model which makes it a more favourable model to use over NN. However, training only the last layer can also be disadvantageous since we lose the dependency on time which plays an important role in diffusion models.

Table 2: Table showing PSNR values for denoising

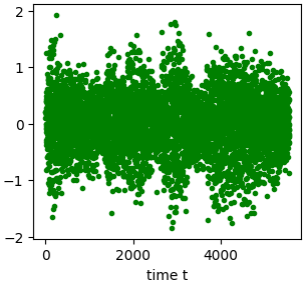
Model	Trainable parameters	PSNR
DARF	$\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$	11.8341
NN	$\mathbf{W}_1, \mathbf{b}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2$	11.6259
RF	$\boldsymbol{\theta}_2$	8.7679



(a) DARF



(b) NN



(c) RF

Figure 7: Denoised signals using different models.