NF-ULA

Cai, Ziruo; Tang, Junqi; Mukherjee, Subhadip; Li, Jinglai; Schönlieb, Carola-Bibiane; Zhang, Xiaoqun

DOI:
10.1137/23M1581807

NF-ULA: Normalizing flow-based unadjusted Langevin algorithm for imaging inverse problems

Ziruo Cai*, Junqi Tang†, Subhadip Mukherjee‡, Jinglai Li§, Carola-Bibiane Schönlieb¶, and Xiaoqun Zhang‖

Abstract. Bayesian methods for solving inverse problems are a powerful alternative to classical methods since the Bayesian approach offers the ability to quantify the uncertainty in the solution. In recent years, data-driven techniques for solving inverse problems have also been remarkably successful, due to their superior representation ability. In this work, we incorporate data-based models into a class of Langevin-based sampling algorithms for Bayesian inference in imaging inverse problems. In particular, we introduce NF-ULA (Normalizing Flow-based Unadjusted Langevin algorithm), which involves learning a *normalizing flow* (NF) as the image prior. We use NF to learn the prior because a tractable closed-form expression for the log prior enables the differentiation of it using *autograd* libraries. Our algorithm only requires a normalizing flow-based generative network, which can be pre-trained independently of the considered inverse problem and the forward operator. We perform theoretical analysis by investigating the well-posedness and non-asymptotic convergence of the resulting NF-ULA algorithm. The efficacy of the proposed NF-ULA algorithm is demonstrated in various image restoration problems such as image deblurring, image inpainting, and limited-angle X-ray computed tomography (CT) reconstruction. NF-ULA is found to perform better than competing methods for severely ill-posed inverse problems.

Key words. Bayesian inference, Langevin algorithms, normalizing flows, inverse problems.

MSC codes. 62F15, 49N45, 92C55

1. Introduction.

Imaging inverse problems can be formulated as $y = Ax + n$, where $y \in \mathbb{R}^m$ is the indirect noisy observation, $A : \mathbb{R}^d \to \mathbb{R}^m$ is the observation operator, n is the measurement noise, and $x \in \mathbb{R}^d$ represents the unknown image that one aims to recover. In the classical variational framework, the reconstruction problem is formulated as the minimization of an energy functional $J(x) = L(y, Ax) + \alpha\, g(x)$, where L measures data-consistency and g is a regularizer that penalizes undesirable images. Following the surge of deep learning, data-driven regularization methods have become ubiquitous in imaging inverse problems [7, 10, 72], leading to state-of-the-art results which significantly outperform classical hand-crafted regularization schemes such as the total-variation [13] or sparsity-based regularizers (see [10] and references therein). Starting from the plug-and-play methods [96] which combine proximal-splitting optimization algorithms [17] with learned denoisers [45, 83, 103], researchers have made considerable progress in this direction. Current popular trends in this

*School of Mathematical Sciences, Shanghai Jiao Tong University, China (sjtu_caiziruo@sjtu.edu.cn).
†School of Mathematics, University of Birmingham, UK (j.tang.2@bham.ac.uk).
‡Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology (IIT) Kharagpur, India. (smukherjee@ece.iitkgp.ac.in).
§School of Mathematics, University of Birmingham, UK (j.li.10@bham.ac.uk).
¶Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK (cbs31@cam.ac.uk).
‖School of Mathematical Sciences, MOELSC and Institute of Natural Sciences, Shanghai Jiao Tong University, China(xqzhang@sjtu.edu.cn).

line of research include the studies in improving practical performances and theoretical guarantees [33,38,47,81,90,94], the development of deep unrolling networks [1,67], deep equilibrium models [34], the studies on the image prior by specific networks structures [59], the extension of generative models in imaging applications [8,73,87,99], operator regularization methods [77], learning explicitly the regularization functional such as a gradient-step denoiser [42], total deep variation [53], adversarial regularizers [63,69,78] and the learned convex regularizer [70] with input-convex neural networks [5].

While the previously mentioned approaches treat x deterministically, another alternative framework for solving inverse problems is to do it within a Bayesian setting [46,93,95]. Different from the functional-analytic methods, Bayesian methods model the image x as a random variable and usually seek to approximate the posterior distribution $p(x|y)$ based on Bayes' formula. The methods based on Bayesian inference can not only give a point estimator (e.g., the maximum a posteriori probability (MAP) estimator) but also describe the uncertainty in the solution in a probabilistic way in terms of variance and credible intervals. The capability of uncertainty quantification is particularly helpful for decision-making and reliability assessment. Typical examples of Bayesian imaging schemes include the classical approach using the total variation prior [62,76], the works on Markov random fields [11], and more recently the patch-based models [2,41,102,105].

In Bayesian inference, one explores the posterior distribution to generate samples from it, typically using the Markov Chain Monte Carlo (MCMC) methods [32]. Among these sampling algorithms, the Langevin Monte Carlo (LMC) algorithms [71,80], also referred to as the *Unadjusted Langevin Algorithms* (ULA), stand out as an increasingly popular tool, since they bridge the gap between theoretical guarantees of nonasymptotic convergence analysis [20,22,28] and practical performance [29,56]. Note that ULA is subject to bias related to the stepsize, ULA can also be modified into Metropolis-adjusted Langevin algorithm (MALA) [80], a non-biased version, by adding a Metropolis-Hastings (MH) accept-reject step. Apart from the MCMC-based methods, there are also other kinds of sampling methods worth mentioning: methods based on variational inference [12,40,61] posit a family of densities and then attempt to find a member of that family which is close to the target density. Variational auto-encoders (VAEs) [52] approximate the posterior by learning deep encoders and decoders. Generative adversarial networks (GAN) [19,35] learn the generator to sample from the training distribution through adversarial learning. More recently, diffusion models [39,88,101] have been shown to be a powerful tool for image generation. They learn the target distribution by transforming an image into a Gaussian noise and then by reversing the noising process.

In recent years, the theoretical analysis and nonasymptotic convergence of ULA [20,28] have opened a new direction of research. Besides convex and smooth potentials [20,21,27,28], ULA for non-convex or non-smooth potentials has also seen great progress. While ULA requires evaluating the score, ULA for non-smooth distributions [29,58,64,68,76] draw samples from a smoothed proxy by borrowing the tools such as proximity operators from non-smooth optimization literature, or consider potential splitting [85]. For non-convex potentials, ULA also has convergence guarantees [14,22,31,65] if some conditions, (e.g., contractivity condition on the drift) are satisfied.

Incorporating data-based approaches into classical algorithms is a trending topic in ULA and Bayesian methods for solving inverse problems. More specifically, one aims to utilize an over-parameterized model learned on given data, such as a neural network, instead of handcrafted prior. Recently, Langevin Monte Carlo using Plug and Play Prior (PnP-ULA) [56] was shown to yield promising results for Bayesian imaging problems. PnP-ULA leverages an implicit image prior learned via a Lipschitz-continuous image denoiser [84]. Since the true image prior is not assumed to be convex or smooth, PnP-ULA convergence was established for non-convex potentials.

Besides PnP priors [96], normalizing flow (NF)-based approaches [25, 74, 79] also lead to impressive performance on imaging problems [25, 50] and have the potential of learning the prior in the Bayesian imaging framework. In this work, we attempt to integrate an image prior that is learned by NF into the Langevin algorithms. Notably, the resulting negative log posterior in our case is non-convex. To make the model well-defined in the Bayesian setting and to ensure that the algorithm is numerically stable, we make minor changes to the standard ULA to add a regularization on the posterior, akin to PnP-ULA [56]. As some studies of normalizing flows have shown [25, 50, 74, 79], training a normalizing flow prior for natural images generally requires utilizing larger networks, larger training dataset, more computational resources and more time than training a PnP denoiser, our proposed method is more efficient if the normalizing flow prior is pre-trained and available.

The idea of interlacing NF with MCMC algorithms has been considered previously in the literature, but these methods had significant conceptual differences from our approach. For instance, [100] proposed stochastic NF, an arbitrary sequence of deterministic invertible functions and stochastic sampling blocks, to sample from target density. The authors of [36,91] considered stochastic NF from a Markov chain point of view and replaced the transition densities with general Markov kernels. [15] utilized NF to sample from the target distribution in the latent domain before transporting it back to the target domain relying on MALA. There are some studies combining other generative models with non-Langevin Monte Carlo algorithms, e.g., [16] introduced a stochastic PnP sampling algorithm leveraging variable splitting to efficiently sample from a posterior distribution using diffusion-based generative models [23]. To summarize, all the above mentioned approaches are different from ours, mainly because they do not directly utilize the log gradient density of NF in Langevin algorithms.

1.1. Our contributions.

The main contributions of this work are:

1. We propose NF-ULA, a novel framework of sampling by Langevin Monte Carlo-based algorithms while leveraging a pre-trained normalizing flow induced prior. Since both the density and the log gradient of the density of normalizing flows can be evaluated, NF-ULA can potentially be extended to a Metropolis-adjusted version.
2. We give a sufficient condition to ensure the Lipschitz gradient of the log density of the normalizing flows since the Lipschitz gradient is one of the most essential conditions to guarantee the convergence of ULA. This might also be useful in the future when an NF-based prior is used in methods other than Langevin algorithms.
3. We show that the Bayesian solution of NF-ULA is well-defined and well-posed and establish that NF-ULA admits an unique invariant distribution. We also give a non-asymptotic bound on the bias.
4. We demonstrate that NF-ULA yields high-quality results in applications such as image deblurring, image inpainting, and limited-angle X-ray computed tomography (CT) re-

construction. For more ill-posed problems, NF-ULA demonstrates stronger regularization than competing methods. We also provide experimental evidence that enhanced training of the NF prior results in improved sampling and reconstruction, especially for severely ill-posed problems (such as limited-angle CT).

The rest of the paper is organized as follows: Sec. 2 gives a brief review of both Langevin Monte Carlo and normalizing flow, leading to the proposed NF-ULA method. Sec. 3 presents a theoretical analysis of the Bayesian solution obtained using NF-ULA. In Sec. 4, we evaluate NF-ULA on image deblurring, image inpainting, and limited-angle CT reconstruction. Final conclusions are summarized in Sec. 5. The proofs and extra experiments are in the Appendix.

2. Mathematical background and the proposed method.

We begin by giving some background on Langevin Monte Carlo (LMC) algorithms and normalizing flow. Subsequently, we propose NF-ULA, an LMC algorithm that utilizes a pre-trained normalizing flow network.

2.1. LMC for Non-smooth Potentials.

In Bayesian inference, there is a broad class of problems where we seek to draw samples $\{X_k\}_{k=1}^{K}, X_k \in \mathbb{R}^d$, from a target posterior distribution $p(x|y)$, given the observation $y \in \mathbb{R}^m$. Using Bayes' formula, we have that

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|\tilde{x})p(\tilde{x})\mathrm{d}\tilde{x}}. \tag{2.1}$$

Under some assumptions on the likelihood $p(y|x)$ and the prior $p(x)$, the posterior distribution $p(x|y)$ is well-posed; meaning that it is well-defined ($\int p(y|\tilde{x})p(\tilde{x})\mathrm{d}\tilde{x}$ is finite), unique, and varies continuously in y with respect to appropriate distance metrics for probability distributions [55, 89]. The well-known LMC approach [71, 80], also referred to as the *unadjusted Langevin algorithm* (ULA), can efficiently sample from $p(x|y)$ using the following Markov chain:

$$\begin{aligned} X_{k+1} &= X_k + \delta\nabla \log p\left(X_k|y\right) + \sqrt{2\delta}Z_{k+1} \\ &= X_k + \delta\nabla \log p\left(y|X_k\right) + \delta\nabla \log p\left(X_k\right) + \sqrt{2\delta}Z_{k+1}, \end{aligned} \tag{2.2}$$

where $\{Z_k\}_k \sim \mathcal{N}(0, I^d)$ is a family of i.i.d. standard Gaussian random variables. The ULA approach in (2.2) is based on the Euler-Maruyama (EM) discretization with step-size δ of the over-damped Langevin stochastic differential equation (SDE) given by

$$\mathrm{d}X_t = \nabla \log p\left(X_t|y\right)\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t, \tag{2.3}$$

where B_t is a Brownian motion. It has been shown in [20, 28] that when $-\log p(x|y)$ is continuously differentiable and has Lipschitz gradient, the convergence of ULA can be guaranteed if the convexity of $-\log p(x|y)$ [20] or contractivity in the tails [28] is satisfied. The convergence is subject to a bias related to the step-size δ. In general, smaller δ leads to a smaller bias and larger δ leads to faster convergence of the Markov Chain. The non-asymptotic bias and convergence analysis of ULA have remained relatively under-explored until the last few years [20, 21, 27, 28]. Notably, the bias of ULA in (2.2) can be removed by adding a Metropolis-Hastings (MH) accept-reject step, leading to the so-called Metropolis-adjusted Langevin algorithm (MALA) [80]. In this paper, we will focus on ULA without any MH adjustments.

When the potential $-\log p(x)$ is convex but non-smooth, [29] uses a smooth proxy utilizing the Moreau envelope $U^{(\lambda)}(x)$ of $U(x) = -\log p(x)$ in (2.2). The Moreau envelope $U^{(\lambda)}(x)$ and the proximity operator $\text{prox}_{\lambda,U}$ of $U(x)$ are defined as

$$U^{(\lambda)}(x) := \inf_{z\in\mathbb{R}^d} \left(U(z) + \frac{1}{2\lambda}\|x - z\|_2^2 \right), \text{ and } \quad \text{prox}_{\lambda,U}(x) := \arg\min_{z\in\mathbb{R}^d} \left(U(z) + \frac{1}{2\lambda}\|x - z\|_2^2 \right).$$

For a convex function U, $\text{prox}_{\lambda,U}(x)$ is unique and well-defined.

Since the Moreau envelope $U^{(\lambda)}(x)$ is always continuously differentiable [9, 18] even if $U(x)$ is not, the authors of [29] replace $\nabla U(x)$ by $\nabla U^{(\lambda)}(x) = \left(x - \text{prox}_{\lambda,U}(x)\right)/\lambda$, resulting in Moreau-Yoshida regularized ULA (referred to as MYULA), which requires the proximal operator of $U(x)$ in each iteration of (2.2).

In a more general case where the prior $p(x)$ is not available in closed form, the authors of [56] propose a plug-and-play (PnP) denoising-based approach for learning the prior [84, 96]. This is achieved by training a Lipschitz-continuous Gaussian denoiser $D_\varepsilon(x)$. More precisely, $D_\varepsilon(x)$ is trained on a given dataset $\{x_n\}_{n=1}^N$ by learning to remove Gaussian noise of zero-mean and ε variance added to the clean images x_n, which are i.i.d. samples of $p(x)$. The ideal minimum mean-squared-error (MMSE) denoiser takes the form

$$D_\varepsilon(x) = (2\pi\varepsilon)^{-d/2} \int_{\mathbb{R}^d} \tilde{x} \exp\left[-\|x - \tilde{x}\|^2/(2\varepsilon)\right] p(\tilde{x})\mathrm{d}\tilde{x}. \tag{2.4}$$

The noisy data follows the Gaussian-smoothed prior

$$p_\varepsilon(x) = (2\pi\varepsilon)^{-d/2} \int_{\mathbb{R}^d} \exp\left[-\|x - \tilde{x}\|_2^2/(2\varepsilon)\right] p(\tilde{x})\mathrm{d}\tilde{x},$$

which is the convolution of the non-explicit prior $p(x)$ with a Gaussian smoothing kernel. Similar to the Moreau envelope [9, 18], p_ε is always differentiable and satisfies Tweedie's identity [30]: $\varepsilon\nabla \log p_\varepsilon(x) = D_\varepsilon(x) - x$. While computing $\nabla \log p(x)$ could be intractable, one can use $\nabla \log p_\varepsilon(x)$ as a surrogate in (2.2), leading to the PnP-ULA approach [56]:

$$\begin{aligned}\text{(PnP-ULA)}: \quad X_{k+1} &= X_k + \delta\nabla \log p\left(y|X_k\right) \\ &\quad + \frac{\delta\alpha}{\varepsilon}\left(D_\varepsilon\left(X_k\right) - X_k\right) + \frac{\delta}{\lambda}\left(\Pi_C\left(X_k\right) - X_k\right) + \sqrt{2\delta}Z_{k+1},\end{aligned} \tag{2.5}$$

where $\alpha > 0$ is a regularization parameter associated with the PnP prior and $\{Z_k\}_k$ are i.i.d. drawn from $\mathcal{N}(0, I^d)$. A projection $\Pi_C\left(X_k\right)$ onto a convex and compact set C is added in each iteration to enable the theoretical analysis for PnP-ULA. $\lambda > 0$ is a parameter associated with the operator $\Pi_C - \text{Id}$. Moreover, the Lipschitz continuity of the denoiser $D_\varepsilon(x)$ is required for convergence. A detailed convergence analysis of (2.5) is available in [56].

2.2. Normalizing Flow. Similar to a PnP prior, a flow-based model can also serve as a prior. A flow-based model seeks to express $x \in \mathbb{R}^d$ as

$$x = T(z), \tag{2.6}$$

where $T : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible transformation applied to $z \in \mathbb{R}^d$, where $z \sim q_z(z)$. Here, $q_z(z)$ is the input (or, latent) distribution of the flow-based model and is generally chosen to be a distribution that can be sampled easily, such as a multivariate Gaussian [51, 54, 74, 79]. Apart from $T : \mathbb{R}^d \to \mathbb{R}^d$ being invertible, both T and T^{-1} must be differentiable [74, 79]. The flow-based model is also called *normalizing flow* since T^{-1} implicitly transforms $q(x)$, the distribution of x, into a normal distribution. In practice, T is typically implemented with an invertible neural network [25, 50]. By a change of variables in (2.6), the distribution of x can be written as

$$q(x) = q_z(z) \left|\det J_T(z)\right|^{-1} = q_z\left(T^{-1}(x)\right) \left|\det J_{T^{-1}}(x)\right|, \tag{2.7}$$

where $z = T^{-1}(x)$ and $J_T(z)$ is the $d \times d$ Jacobian matrix of T. Many normalizing flows [50,51,74,75,79] use specific network architectures such that T^{-1} is a triangular mapping, that is, the Jacobian $J_{T^{-1}}(x)$ is a triangular matrix, which simplifies the calculation of $|\det J_{T^{-1}}(x)|$. Note that T is used to generate x from z, and T^{-1} is needed for evaluating the density $q(x)$.

Some works on normalizing flow use coupling layers in the network to make T^{-1} a triangular mapping [24, 25, 50, 51, 75]. Denote $G(x) = T^{-1}(x)$, $G : \mathbb{R}^d \to \mathbb{R}^d$. Let x_j be the j-th element of x and $x_{<j}$ be the elements before x_j, i.e. $x_1, \cdots, x_{j-1}$. Then, for one-layer network, [44] summarizes the coupling layer-based flows as $G_j(x_j, x_{<j}) = \varphi_j(x_{<j})x_j + \eta_j(x_{<j})$, where G_j is the j-th element of the vector $G(x)$ and the functions φ_j and η_j map $x_{<j}$ to a real number. The Jacobian $J_G(x)$ is triangular since G_j only depends on x_j and $x_{<j}$.

Assume that the unknown prior distribution that we aim to learn is $p(x)$. Then, the forward KL divergence between the target distribution $p(x)$ and the output distribution $q(x)$ of the NF model [54, 74, 79] can be written as

$$\begin{aligned} D_{\mathrm{KL}}(p, q) &= -\mathbb{E}_{p(x)}\left[\log q(x)\right] + \text{const.} \\ &= -\mathbb{E}_{p(x)}\left[\log q_z\left(T^{-1}(x)\right) + \log\left|\det J_{T^{-1}}(x)\right|\right] + \text{const.} \end{aligned} \tag{2.8}$$

When the transformation T is parameterized by an invertible neural network T_θ with parameters $\theta \in \Theta$, we denote the parameterized density of x as $q_\theta(x)$ and the optimization problem of learning T_θ reads:

$$\min_{\theta \in \Theta} D_{\mathrm{KL}}(p, q_\theta). \tag{2.9}$$

Given samples $\{x_n\}_{n=1}^N$ drawn i.i.d. from $p(x)$, we can estimate the expectation in (2.8) by Monte Carlo averaging over the training samples $\{x_n\}_{n=1}^N$. Correspondingly, the loss function for training the NF model becomes

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left(\log q_z\left(T_\theta^{-1}(x_i)\right) + \log\left|\det J_{T_\theta^{-1}}(x_i)\right|\right) + \text{const.} \tag{2.10}$$

Generally, it is reasonable to assume that the data samples $\{x_i\}_i^N$ lie within a compact set $C_R \subset \mathbb{R}^d$. In particular, when the flow-based model is learned on imaging data, it is common to set $C_R = [0, 1]^d$. Knowing the set where the data samples lie will give us the intuition to select some parameters in the next section. From the numerical observations, the networks also partially know C_R while trained from the data - the knowledge of C_R is implicitly encapsulated in a well-trained flow model, meaning that most generated samples using a well-trained NF model fall within C_R.

2.3. ULA with NF-prior . In this section, we propose a framework for sampling using the LMC algorithm based on a pre-trained normalizing flow network. Given data samples $\{x_n\}_{n=1}^N$ drawn i.i.d. from $p(x)$, one can approximate $p(x)$ by learning a flow-based model $x = T_\theta(z)$, with output distribution $q_\theta(x) = q_z\left(T_\theta^{-1}(x)\right)\left|\det J_{T_\theta^{-1}}(x)\right|$. Once $q_\theta(x)$ is learned, $\log q_\theta(x)$ is always differentiable since T_θ and T_θ^{-1} are differentiable. By replacing $p(x)$ with $q_\theta(x)$ in (2.2), the ULA scheme boils down to

$$X_{k+1} = X_k + \delta\nabla \log p\left(y|X_k\right) + \delta\nabla \log q_\theta(X_k) + \sqrt{2\delta}Z_{k+1}.$$

Since convexity of $-\log q_\theta(x)$ and the Lipschitz continuity of its gradient are not guaranteed to be satisfied, one does not yet have the sufficient conditions to infer convergence and numerical stability similar to the cases in [20, 28]. In this work, we follow [56] to impose a projection $\Pi_C\left(X_k\right)$ onto a convex and compact set C to ensure that the posterior distribution is well-defined and propose the resulting NF-ULA algorithm (c.f. Algorithm 2.1). The parameter

Algorithm 2.1 Normalizing Flow-based Unadjusted Langevin algorithm (NF-ULA)

Input: $y \in \mathbb{R}^m$, $X_0 \in \mathbb{R}^d$, $\alpha > 0$, $\lambda > 0$, $K \in \mathbb{N}$, $C \subset \mathbb{R}^d$
L_y: Lipschitz constant of $\nabla \log p(y|x)$.
L: Lipschitz constant of $\nabla \log q_\theta(x)$.
Output: $\{X_k\}_{k=1}^K$

Set: $k = 0$, $\delta < (1/6)\left(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda\right)^{-1}$.
Initialize X_0 according to the considered problems.
while $k < K$ **do**
  $Z_{k+1} \sim \mathcal{N}(0, I^d)$
  $X_{k+1} = X_k + \delta\nabla \log p\left(y|X_k\right) + \delta\alpha\nabla \log q_\theta(X_k) + \dfrac{\delta}{\lambda}\left(\Pi_C\left(X_k\right) - X_k\right) + \sqrt{2\delta}Z_{k+1}$
  $k = k + 1$
end while

$\alpha > 0$ controls how strongly the regularization of q_θ is imposed and λ controls the amount of the projection $(\Pi_C - \mathrm{Id})$ enforced. Theoretical analysis of NF-ULA is presented in Sec. 3, while in Sec. 4, we provide some general guidelines for selecting the hyper-parameters involved in NF-ULA. One can efficiently compute $\nabla \log q_\theta(x)$ using the automatic differentiation libraries in the standard deep learning frameworks (such as `PyTorch`).

Remark: Algorithm 2.1 only requires evaluating the $\nabla \log q_\theta(x)$ and its Lipschitz constant. Our theoretical analysis in Sec. 3 depends on the properties of $q_\theta(x)$ and holds even when q_θ does not arise from a normalizing flow. This is essential since in our CT experiments in Sec. 4.3, we utilize *patchNR* [3], a normalizing flow-based regularizer which cannot generate x by (2.6) but is able to evaluate the log gradient $\nabla \log q_\theta(x)$. Moreover, since $q_\theta(x)$ can also be evaluated given x, Algorithm 2.1 can be extended to a Metropolis-adjusted version by adding an accept-reject step. We leave this as a possible future work.

It is imperative to understand why the projection $(\Pi_C - \mathrm{Id})$ is necessary for the convergence of NF-ULA. Let $\iota_C^{(\lambda)}(x)$ be the λ-Moreau envelope [9] of the indicator function

$$\iota_C(x) = \begin{cases} 0, & x \in C, \\ +\infty, & x \notin C. \end{cases}$$

Then, we have that

$$\iota_C^{(\lambda)}(x) := \inf_{u \in \mathbb{R}^d} \left(\iota_C(u) + \frac{1}{2\lambda} \|x - u\|_2^2 \right) = \frac{1}{2\lambda} \|x - \Pi_C(x)\|_2^2 ,$$
$$\text{and } \nabla \iota_C^{(\lambda)}(x) = \frac{x - \mathrm{Prox}_{\iota_C}(x)}{\lambda} = \frac{x - \Pi_C(x)}{\lambda},$$

where Π_C is the projection operator on the convex and compact (i.e., closed and bounded) set $C \subset \mathbb{R}^d$. Define $p_\lambda(x|y)$ as

$$p_\lambda(x|y) = \frac{p(y|x) q_\theta^\alpha(x) \exp(-\iota_C^{(\lambda)}(x))}{\int_{\mathbb{R}^d} p(y|\tilde{x}) q_\theta^\alpha(\tilde{x}) \exp(-\iota_C^{(\lambda)}(\tilde{x})) \mathrm{d}\tilde{x}}, \tag{2.11}$$

where the exponent $\alpha > 0$. The subscript λ in p_λ underlines the distinction from the posterior $p(x|y) = p(y|x)p(x)/p(y)$. Since θ is fixed if the NF is pre-trained and α is adjusted in the experiments section, they are not in the notation of p_λ for brevity. We show in Sec. 3.2 that $p_\lambda(x|y)$ is well-defined and therefore the projection term is necessary for NF-ULA, without which, $p(y|x)q_\theta^\alpha(x)/\int_{\mathbb{R}^d} p(y|\tilde{x})q_\theta^\alpha(\tilde{x})\mathrm{d}\tilde{x}$ is not guaranteed to be well-defined in our settings. Denote by $\pi_{\lambda,y}$ (which we will write as π_λ for brevity) the probability measure whose density is $p_\lambda(x|y)$ in (2.11), i.e.,

$$\frac{\mathrm{d}\pi_\lambda}{\mathrm{d}\pi_{\mathrm{leb}}}(x) = p_\lambda(x|y), \tag{2.12}$$

where π_{leb} denotes the Lebesgue measure. Then, NF-ULA in Algorithm 2.1 is essentially equivalent to

$$X_{k+1} = X_k + \delta \nabla \log p_\lambda(X_k|y) + \sqrt{2\delta} Z_{k+1}. \tag{2.13}$$

For standard ULA (2.2), the tail-decay condition ($-\log p(x|y)/\|x\|^2$ converges to a positive constant when $x \to \infty$) was first studied in [80, 92] and was shown to imply the convergence of ULA. For NF-ULA (2.13), we want to emphasize that in most of our experiments, NF-ULA is convergent while using a well-pre-trained normalizing flow, even without the projection term. This is presumably because the density q_θ of a well-trained normalizing flow already satisfies the tail-decay condition [80, 92] and most of the probability mass lies within C. For the cases where the normalizing flow is poorly trained, one should select a smaller C, without which the samples generated by NF-ULA will go far beyond our expected region (for imaging it is $C_R = [0, 1]^d$).

3. Theoretical Analysis.

We define some useful notations for our analysis in Sec. 3.1 and present a theoretical analysis (well-definedness and well-posedness) of the Bayesian posterior $p_\lambda(x|y)$ in Sec. 3.2. Subsequently, we prove the convergence and non-asymptotic bias of NF-ULA in Sec. 3.3.

3.1. Notations.

Denote by $\mathcal{B}\left(\mathbb{R}^d\right)$ the Borel σ-field of $\mathbb{R}^d$. Let μ be a probability measure on $\left(\mathbb{R}^d, \mathcal{B}\left(\mathbb{R}^d\right)\right)$ and f be a μ-integrable function. Denote by $\mu(f)$ the integral of f w.r.t. μ. For measurable $f: \mathbb{R}^d \to \mathbb{R}$ and measurable $V: \mathbb{R}^d \to [1, \infty)$, the V-norm of f is defined as $\|f\|_V = \sup_{\tilde{x} \in \mathbb{R}^d} |f(\tilde{x})|/V(\tilde{x})$. Let ξ be a finite signed measure on $\left(\mathbb{R}^d, \mathcal{B}\left(\mathbb{R}^d\right)\right)$. Then the V-total variation norm of ξ is defined as

$$\|\xi\|_V = \sup_{\|f\|_V \leqslant 1} \left| \int_{\mathbb{R}^d} f(\tilde{x}) \mathrm{d}\xi(\tilde{x}) \right|. \tag{3.1}$$

Note that if $V = 1$, then $\|\cdot\|_V$ is the total variation $\|\cdot\|_{\mathrm{TV}}$. $\|\cdot\|_V$ is weaker than $\|\cdot\|_{\mathrm{TV}}$ and from the definitions one has $\|\xi\|_{\mathrm{TV}} \leqslant \|\xi\|_V$. $\|\cdot\|_V$ has been used a lot in the studies of ULA [22, 28, 56].

We denote by $\mathscr{P}\left(\mathbb{R}^d\right)$ the set of probability measures over $\left(\mathbb{R}^d, \mathcal{B}\left(\mathbb{R}^d\right)\right)$ and for any $m \in \mathbb{N}, \mathscr{P}_m\left(\mathbb{R}^d\right) = \left\{\nu \in \mathscr{P}\left(\mathbb{R}^d\right) : \int_{\mathbb{R}^d} \|\tilde{x}\|^m \, \mathrm{d}\nu(\tilde{x}) < +\infty\right\}$. Denote by $\mathbf{W}_p$ as Wasserstein-p metric:

$$\mathbf{W}_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \mathbf{E}_{(x,y)\sim\gamma} \|x - y\|^p \right)^{1/p}, \quad p \geqslant 1, \tag{3.2}$$

where $\Gamma(\mu, \nu)$ is the set of all joint probability whose marginal distributions are μ and ν respectively.

Let $b \in \mathrm{C}\left(\mathbb{R}^d, \mathbb{R}^d\right)$ where $C\left(\mathbb{R}^d, \mathbb{R}^d\right)$ stands for the set of all continuous functions from $\mathbb{R}^d$ to $\mathbb{R}^d$. We consider the Markov chain $(X_k)_{k\in\mathbb{N}}$ given by the following recursion for any $k \in \mathbb{N}$ and $x \in \mathbb{R}^d$, initialized at $X_0 = x$:

$$X_{k+1} = X_k + \gamma b\left(X_k\right) + \sqrt{2\gamma} Z_k,$$

where $\gamma > 0$ and $\{Z_k : k \in \mathbb{N}\}$ a family of i.i.d. Gaussian random variables with zero mean and identity covariance matrix. We define its associated Markov kernel $\mathrm{R}_\gamma : \mathbb{R}^d \times \mathcal{B}\left(\mathbb{R}^d\right) \to [0, 1]$ as follows for any $x \in \mathbb{R}^d$ and $\mathrm{A} \in \mathcal{B}\left(\mathbb{R}^d\right)$:

$$\mathrm{R}_\gamma(x,\ \mathrm{A}) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} \mathbf{1}_{\mathrm{A}}(x + \gamma b(x) + \sqrt{2\gamma} z) \exp\left[-\|z\|^2/2\right] \mathrm{d}z,$$

where $\mathbf{1}_{\mathrm{A}}(x)$ is the function taking the value 1 if $x \in \mathrm{A}$ or 0 if $x \notin \mathrm{A}$. We say that R_γ satisfies a discrete drift condition $\mathbf{D}_{\mathrm{d}}(W, \zeta_{\mathrm{d}}, c)$ if there exist $\zeta_{\mathrm{d}} \in [0, 1), c \geqslant 0$ and a measurable function $W : \mathbb{R}^d \to [1, +\infty)$ such that for all $x \in \mathbb{R}^d$

$$\mathrm{R}_\gamma W(x) \leqslant \zeta_{\mathrm{d}} W(x) + c,$$

where $\mathrm{R}_\gamma W(x) := \int_{\mathbb{R}^d} \mathrm{R}_\gamma(x, \mathrm{d}\tilde{x}) W(\tilde{x})$. Note that this drift condition implies the existence of an invariant probability measure if R_γ is a Feller kernel and the level sets of W are compact, see [22] and Theorem 12.3.3 in [26].

Similarly, let $b \in \mathrm{C}\left(\mathbb{R}^d, \mathbb{R}^d\right)$ such that for any $x \in \mathbb{R}^d$, the following SDE admits a unique strong solution

$$
\begin{aligned}
&\mathrm{d}\mathbf{X}_t = b\left(\mathbf{X}_t\right)\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{B}_t,\\
&\mathbf{X}_0 = x,
\end{aligned} \tag{3.3}
$$

where $(\mathbf{B}_t)_{t\geqslant 0}$ is a d-dimensional Brownian motion. For any $x \in \mathbb{R}^d$ and $\mathrm{A} \in \mathcal{B}\left(\mathbb{R}^d\right)$, equation (3.3) defines a Markov semi-group $(\mathrm{P}_t)_{t\geqslant 0}$ by $\mathrm{P}_t(x,\ \mathrm{A}) = \mathbb{P}\left(\mathbf{X}_t \in \mathrm{A}\right)$ where $(\mathbf{X}_t)_{t\geqslant 0}$ is the solution of (3.3) with $\mathbf{X}_0 = x$. For any $f \in \mathrm{C}^2\left(\mathbb{R}^d, \mathbb{R}\right)$, define the generator $\mathcal{A}$ of $(\mathrm{P}_t)_{t\geqslant 0}$ by $\mathcal{A}f = \langle \nabla f, b(x)\rangle + \Delta f$, where Δ is the Laplace operator. We say that $(\mathrm{P}_t)_{t\geqslant 0}$ on $\mathbb{R}^d \times \mathcal{B}\left(\mathbb{R}^d\right)$ with extended infinitesimal generator $(\mathcal{A}, \mathrm{D}(\mathcal{A}))$ (see e.g. [66] for the definition of $(\mathcal{A}, \mathrm{D}(\mathcal{A}))$) satisfies a continuous drift condition $\mathbf{D}_{\mathrm{c}}(W, \zeta, \beta)$ if there exist $\zeta > 0, \beta \geqslant 0$ and a measurable function $W : \mathbb{R}^d \to [1, +\infty)$ with $W \in \mathrm{D}(\mathcal{A})$ such that for all $x \in \mathbb{R}^d$,

$$
\mathcal{A}W(x) \leqslant -\zeta W(x) + \beta.
$$

This assumption is the continuous counterpart of the discrete drift condition $\mathbf{D}_{\mathrm{d}}(W, \zeta_{\mathrm{d}}, c)$, which will be used in Appendix A.7.

3.2. Well-posedness of the Bayesian solution.

In this section, we first prove that the posterior distribution (2.11) is well-defined. Secondly, we prove the well-posedness for the Bayesian solution, i.e., the Lipschitz continuity of the posterior measure (2.12) with respect to changes in y. To start with, we give a lemma that will be used later.

Lemma 3.1. *Let $\lambda > 0$. For any convex and compact subset C of $\mathbb{R}^d$ and for all $k \in \mathbb{N}$, it holds that*

$$
\int_{\mathbb{R}^d} \|x\|^k \exp\left(-\frac{\|x - \Pi_C(x)\|_2^2}{2\lambda}\right) \mathrm{d}x < +\infty.
$$

Proof. See Appendix A.1. ∎

Lemma 3.1 implies that the integral of any polynomials multiplied by $\exp\left(-\iota_C^{(\lambda)}\right)$, where $\iota_C^{(\lambda)} = \dfrac{\|x - \Pi_C(x)\|_2^2}{2\lambda}$, is finite. To prove that $p_\lambda(x|y)$ and π_λ are well-defined, besides Lemma 3.1, we need an assumption about the boundedness of the prior and the likelihood.

Assumption 3.2. The distribution learned by NF is bounded, i.e., $\sup_{x\in\mathbb{R}^d} q_\theta(x) < +\infty$. Moreover, for any $y \in \mathbb{R}^m$, $\sup_{x\in\mathbb{R}^d} p(y|x) < +\infty$ and $p(y|\cdot) \in \mathrm{C}^1\left(\mathbb{R}^d, (0, +\infty)\right)$.

Since $q_\theta(x)$ is a distribution induced by normalizing flow and $q_\theta(x)$ is continuous on $\mathbb{R}^d$, intuitively $\sup_{x\in\mathbb{R}^d} q_\theta(x)$ is bounded and Assumption 3.2 is easily satisfied. To give rigorous proof, we state the following proposition which assumes a similar triangular network architecture as mentioned in Sec. 2.2.

Proposition 3.3. *Assume that the input distribution* $q_z(z)$ *to the normalizing flow network is the standard normal distribution. Assume that* $T^{-1}(x) = G^{(k)} \circ \cdots \circ G^{(1)}(x)$ *is a composition of* k *coupling layers and each of the layer* $G^{(i)} : \mathbb{R}^d \to \mathbb{R}^d, x^{(i)} \mapsto x^{(i+1)}$ *is given by*

$$G_j^{(i)}(x_j^{(i)}, x_{<j}^{(i)}) = \varphi_j^{(i)}(x_{<j}^{(i)})x_j^{(i)} + \eta_j^{(i)}(x_{<j}^{(i)}),\ j = 1, \cdots, d. \tag{3.4}$$

Denote $x^{(1)} = x$ *and* $x^{(k+1)} = z$*. If* $\varphi_j^{(i)}$*s are bounded, then* $\log q_\theta(x)$ *is upper bounded on* $\mathbb{R}^d$*.*

Proof. See Appendix A.2. ■

Using Lemma 3.1, we can then prove that the normalizing constant in the expression for $p_\lambda(x|y)$ in (2.11) is finite.

Corollary 3.4. *Suppose Assumption 3.2 holds. Let* $\lambda > 0$*. Then, for any convex and compact set* C *and* $\alpha > 0$*, we have*

$$\int_{\mathbb{R}^d} p(y|x) q_\theta^\alpha(x) \exp\left(-\frac{\|x - \Pi_C(x)\|_2^2}{2\lambda}\right) \mathrm{d}x < +\infty.$$

Hence, $p_\lambda(x|y)$ *in* (2.11) *is well-defined.*

Proof. Letting $k = 0$ in Lemma 3.1 and using Assumption 3.2, we conclude the proof. ■

Remark: Although $\int_{\mathbb{R}^d} q_\theta(x)\mathrm{d}x = 1$, $\int_{\mathbb{R}^d} q_\theta^\alpha(x)\mathrm{d}x$ may not be finite in rare cases. This depends on how heavy the tail of $q_\theta(x)$ is. Corollary 3.4 shows that multiplying $q_\theta^\alpha(x)$ with $\exp\left(-\iota_C^{(\lambda)}(x)\right)$ always leads to a finite integral, regardless of the tail behavior of $q_\theta(x)$.

Now, we establish the well-posedness of the posterior measure π_λ in the following proposition. Note that the local Lipschitz stability of posterior distribution in the observation has been studied in [55, 89] and applied to posterior sampling with PnP prior [56] and generative models in [4]. Apart from the considered $\iota_C^{(\lambda)}(\tilde{x})$, Proposition 3.5 and Proposition 3 in [56] are based on similar ideas.

Proposition 3.5. *Suppose Assumption 3.2 holds and that there exist continuous functions* $\Phi_1 : \mathbb{R}^d \to [0, +\infty)$ *and* $\Phi_2 : \mathbb{R}^m \to [0, +\infty)$ *such that for any* $x \in \mathbb{R}^d$ *and* $y_1, y_2 \in \mathbb{R}^m$*, the following are satisfied:*

$$\begin{aligned} &\big|\log\left(p\left(y_1|x\right)\right) - \log\left(p\left(y_2|x\right)\right)\big| \leqslant \left(\Phi_1(x) + \Phi_2\left(y_1\right) + \Phi_2\left(y_2\right)\right)\|y_1 - y_2\|, \\ &\textit{and} \int_{\mathbb{R}^d} \left(1 + \Phi_1(\tilde{x})\right) \exp\left[c_0\, \Phi_1(\tilde{x}) - \iota_C^{(\lambda)}(\tilde{x})\right] \mathrm{d}\tilde{x} < +\infty, \end{aligned}$$

for all $c_0 > 0$*. Then,* $y \mapsto \pi_{\lambda,y}$ *defined in* (2.12)*) is locally Lipschitz w.r.t. the total-variation (TV) norm* $\|\cdot\|_{\mathrm{TV}}$*, i.e., for any compact set* K*, there exists* $M_K \geqslant 0$ *such that for any* $y_1, y_2 \in K$, $\|\pi_{\lambda,y_1} - \pi_{\lambda,y_2}\|_{\mathrm{TV}} \leqslant M_K \|y_1 - y_2\|$.

Proof. See Appendix A.3. ■

For Gaussian likelihood $p(y|x)$, the conditions in Proposition 3.5 are satisfied when $\Phi_1(x) = c_1\|x\|_2$ and $\Phi_2(y) = c_2\|y\|_2$ with positive constants c_1 and c_2.

3.3. Convergence of NF-ULA. Most of the existing works on ULA for non-convex potentials [14, 28, 31, 56, 65] assume Lipschitz-continuity of the score. If the drift term $\nabla \log p_\lambda(x|y)$ is not Lipschitz, from [43, 48], it cannot generally be guaranteed that the SDE (2.3) will have a unique strong solution. This is why one must investigate the Lipschitz continuity of $\nabla \log p_\lambda(x|y)$ before studying the convergence of NF-ULA. First, we make an assumption about the Lipschitz-continuity of $\nabla \log(p(y|\cdot))$:

Assumption 3.6. $\nabla \log(p(y|x))$ is L_y-Lipschitz continuous in x, where $\mathrm{L}_y > 0$ is a constant.

Note that Assumption 3.6 is generally satisfied for common imaging inverse problems. One example is the popular Gaussian likelihood where $p(y|x) \propto \exp\left(-\|y - Ax\|_2^2/(2\sigma^2)\right)$, for which $\mathrm{L}_y = \|A^\top A\|/\sigma^2$.

Lemma 3.7. *Under Assumption 3.6, $\nabla \log p_\lambda(x|y)$ is Lipschitz continuous if and only if $\nabla \log q_\theta(x)$ is Lipschitz continuous.*

Proof. See Appendix A.4. ■

For convenience, we explicitly define the Lipschitz condition on the log gradient of $q_\theta(x)$ in the following assumption:

Assumption 3.8. There exist $\mathrm{L} \geqslant 0$ such that for any $x_1, x_2 \in \mathbb{R}^d$,

$$\|\nabla \log q_\theta(x_1) - \nabla \log q_\theta(x_2)\| \leqslant \mathrm{L}\,\|x_1 - x_2\|.$$

It is therefore natural to ask how to enforce Assumption 3.8 on the NF-based image prior $q_\theta(x)$ during training or by the network architecture. There have been some studies about the Lipschitz continuity of the invertible transform T_θ [54, 74, 97], the Lipschitz constants of invertible neural networks by changing the latent distribution from a standard normal one to a Gaussian mixture model [37], the Lipschitz constants of other "push-forward" generative models [86]. However, to the best of our knowledge, there is no study about the Lipschitz continuity of $\nabla \log q_\theta(x)$ until now.

While the equivalent conditions on T_θ for Assumption 3.8 remain unknown, a sufficient condition on T_θ for Assumption 3.8 can be obtained easily. For instance, when T_θ is a linear transform mapping a Gaussian distribution $q_z(z)$ to another Gaussian distribution $q_\theta(x)$, Assumption 3.8 holds. However, this may not be true if T_θ is nonlinear.

As we have mentioned that Assumption 3.8 is necessary for the convergence of NF-ULA, we derive a sufficient condition on T_θ for Assumption 3.8 to hold. Intuitively, distributions with similar tail behaviors as Gaussian may have similar log gradients as Gaussian, if more conditions are satisfied. We thus refer to some studies on the tails of normalizing flow priors [44]. Theorem 4 in [44] shows that affine coupling layer-based flows (e.g., NICE [24], Real-NVP [25], MAF [75], IAF [51], and Glow [50]) can only map the base normal distribution $q_z(z)$ to a light-tailed distribution $q_\theta(x)$. To be more specific, denote $G(x) = T^{-1}(x)$, where $G(x)$ is a triangular mapping and the Jacobian $J_G(x)$ is a triangular matrix function. From [44], generally one can assume that for affine coupling layer-based flows, $G_j(x_j, x_{<j}) = \varphi_j(x_{<j})x_j + \eta_j(x_{<j})$, where G_j is the j-th element of the vector $G(x)$ and $x_{<j}$ indicate $x_1, \cdots, x_{j-1}$. The condition they assume is heuristic: if φ_j is bounded above and η_j is Lipschitz, then $q_\theta(x)$ is light-tailed. In Glow, [50] these conditions on φ and η are satisfied and even stricter.

Therefore, we are able to prove the Lipschitz continuity of $\nabla \log q_\theta(x)$ in the proposition below, by enforcing a stricter condition on φ and η.

Proposition 3.9. *Assume that the input distribution $q_z(z)$ to the normalizing flow network is the standard normal distribution, and that $T^{-1}(x) = G^{(k)} \circ \cdots \circ G^{(1)}(x)$ is a composition of k coupling layers, where each of the layers $G^{(i)} : \mathbb{R}^d \to \mathbb{R}^d, x^{(i)} \mapsto x^{(i+1)}$ is given by*

$$G_j^{(i)}(x_j^{(i)}, x_{<j}^{(i)}) = \varphi_j^{(i)}(x_{<j}^{(i)})x_j^{(i)} + \eta_j^{(i)}(x_{<j}^{(i)}),\ j = 1, \cdots, d. \tag{3.5}$$

Denote $x^{(1)} = x$ and $x^{(k+1)} = z$. If $\varphi_j^{(i)}$ is a constant function, $\eta_j^{(i)}$ is Lipschitz and for all $r < j$, $\dfrac{\partial \eta_j^{(i)}}{\partial x_r}$ is well-defined almost everywhere and piecewise constant on $\mathbb{R}$, then $\nabla \log q_\theta(x)$ is Lipschitz continuous on $\mathbb{R}^d$.

Proof. See Appendix A.5. ∎

The conditions on φ, η in Proposition 3.9 are satisfied in *Glow* [50] with *additive coupling layers* where each η is a five-layer sequential network with 2D convolutional layers (denoted as `Conv2d`) and `ReLU` activations:

$$\eta(x) = \texttt{Conv2d}(\texttt{ReLu}(\texttt{Conv2d}(\texttt{ReLu}(\texttt{Conv2d}(x))))),$$

where $\texttt{ReLu}(x) := \max(0, x)$ (applied in an element-wise manner) and $\texttt{Conv2d}(x) := K_{\mathrm{NF}} * x$ denotes a 2D convolution layer acting on x with a kernel K_{NF}. Further, $\varphi = 1$ is used in the additive coupling layer. Note that in Glow, there is an option of using an *affine coupling layer* where φ is the sigmoid function $\varphi(x) = 1/(1 + e^{-x})$ element-wise. This leads to a more powerful network and can generate better human face images [50], but $\nabla \log q_\theta(x)$ is not guaranteed to be Lipschitz anymore. This theoretical observation is corroborated by our experiments in Sec. 4.1, as we found that NF-ULA with affine coupling layer did not converge. The conditions on φ and η might be relaxed if $q_z(z)$ is not Gaussian, but this requires re-training the network since most of the popular normalizing flows accept standard Gaussian base distribution as input. We leave these studies on the Lipschitz-continuity of $\nabla \log q_\theta(x)$ for future work.

In order to prove the convergence of NF-ULA, we need one final assumption.

Assumption 3.10. There exists $\mathrm{m}_y \in \mathbb{R}$ such that for all $x_1, x_2 \in \mathbb{R}^d$, we have

$$\langle \nabla \log p\,(y|x_2) - \nabla \log p\,(y|x_1)\,, x_2 - x_1 \rangle \leqslant -\mathrm{m}_y \left\| x_2 - x_1 \right\|_2^2.$$

This condition is called the *contractivity condition* of $\nabla \log p(y|x)$ and is used to prove the contractivity of the drift term $\nabla \log p_\lambda(x|y)$ at infinity (see proofs of Theorem 3.11 in Appendix A.6). Note that the influence of the drift's contractivity condition has been studied in ULA for non-convex potentials [14, 22, 65].

If Assumption 3.10 is satisfied with $\mathrm{m}_y > 0$, then $x \mapsto -\log p(y|x)$ is m_y-strongly convex. If Assumption 3.6 is satisfied, then Assumption 3.10 holds for $\mathrm{m}_y = -\mathrm{L}_y$. However, we are interested to find $\mathrm{m}_y > -\mathrm{L}_y$ while Assumption 3.6 holds, since we will see in the proofs of Theorem 3.11 and Theorem 3.12 in Appendix A.6 and A.7 that a larger m_y is beneficial to the convergence of NF-ULA.

In what follows, we introduce the associated stochastic kernel $\mathrm{R}_\delta : \mathbb{R}^d \times \mathcal{B}(\mathbb{R}^d) \to [0,1]$ of the NF-ULA (2.13) and the drift $b_\lambda \in \mathrm{C}\left(\mathbb{R}^d, \mathbb{R}^d\right)$:

$$
\begin{aligned}
\mathrm{R}_\delta(x,\ \mathrm{A}) &= (2\pi)^{-d/2} \int_{\mathbb{R}^d} \mathbf{1}_{\mathrm{A}}\left(x + \delta b_\lambda(x) + \sqrt{2\delta} z\right) \exp\left[-\|z\|^2/2\right] \mathrm{d}z, \\
b_\lambda(x) &= \nabla \log p_\lambda(x|y) = \nabla \log p\,(y|x) + \alpha \nabla \log q_\theta(x) + \frac{\Pi_C\,(x) - x}{\lambda},
\end{aligned} \tag{3.6}
$$

where $x \in \mathbb{R}^d$ and $\mathrm{A} \in \mathcal{B}(\mathbb{R}^d)$. Here b_λ has the subscript λ and is different from the b defined in Sec. 3.1 because of $(\Pi_C\,(x) - x)/\lambda$. Given X_k in NF-ULA (2.13), $\mathrm{R}_\delta(X_k, \cdot)$ is actually a probability measure which defines the transition probability $p(X_{k+1}|X_k)$.

With all the previous four assumptions A 3.2, A 3.6, A 3.8, and A 3.10 holding, we can prove that NF-ULA (Algorithm 2.1) is convergent, or more precisely, the stochastic kernel R_δ admits an unique invariant distribution $\pi_{\delta,\lambda}$. We follow the proof in SM6.2 from [57] but our theorem and proof are slightly different, as we do not include the parameter ε of PnP denoisers in the condition. The first thing to prove is that R_δ defines a contractive mapping.

Theorem 3.11. *Assume A 3.2, A 3.6, A 3.8, and A 3.10. Assume* $V(x) = 1 + \|x\|^2, x \in \mathbb{R}^d$. *Let* $\lambda, \alpha, C, \mathrm{L}_y, \mathrm{L}$ *be the ones in NF-ULA (Algorithm 2.1). Let* m_y *be the parameter in A 3.10. Let* $\lambda > 0$, *such that* $2\lambda\left(\mathrm{L}_y + \alpha \mathrm{L} - \min(\mathrm{m}_y, 0)\right) \leqslant 1$ *and let* $\bar{\delta} = (1/6)\left(\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda\right)^{-1}$. *Then for any convex and compact* C *with* $0 \in C$, *there exist* $A_1 \geqslant 0$ *and* $\rho_1 \in [0,1)$ *such that for any* $\delta \in (0, \bar{\delta}]$, $x_1, x_2 \in \mathbb{R}^d$, *and* $k \in \mathbb{N}$ *we have*

$$
\begin{aligned}
\left\|\boldsymbol{\delta}_{x_1}\mathrm{R}_\delta^k - \boldsymbol{\delta}_{x_2}\mathrm{R}_\delta^k\right\|_V &\leqslant A_1 \rho_1^{k\delta}\left(V^2\,(x_1) + V^2\,(x_2)\right), \text{ and} \\
\mathbf{W}_1\left(\boldsymbol{\delta}_{x_1}\mathrm{R}_\delta^k, \boldsymbol{\delta}_{x_2}\mathrm{R}_\delta^k\right) &\leqslant A_1 \rho_1^{k\delta}\, \|x_1 - x_2\|_2\,.
\end{aligned}
$$

Proof. See Appendix A.6. ∎

In the above theorem the Dirac measures $\boldsymbol{\delta}_{x_1}, \boldsymbol{\delta}_{x_2}$ can be extended to any measures $\nu_1, \nu_2 \in \mathscr{P}_1\left(\mathbb{R}^d\right)$:

$$
\begin{aligned}
\left\|\nu_1\mathrm{R}_\delta^k - \nu_2\mathrm{R}_\delta^k\right\|_V &\leqslant A_1\rho_1^{k\delta}\left(\int_{\mathbb{R}^d} V^2(\tilde{x})\mathrm{d}\nu_1(\tilde{x}) + \int_{\mathbb{R}^d} V^2(\tilde{x})\mathrm{d}\nu_2(\tilde{x})\right), \\
\mathbf{W}_1\left(\nu_1\mathrm{R}_\delta^k, \nu_2\mathrm{R}_\delta^k\right) &\leqslant A_1\rho_1^{k\delta}\left(\int_{\mathbb{R}^d} \|\tilde{x}\|\mathrm{d}\nu_1(\tilde{x}) + \int_{\mathbb{R}^d} \|\tilde{x}\|\mathrm{d}\nu_2(\tilde{x})\right).
\end{aligned} \tag{3.7}
$$

From Theorem 6.18 in [98], $\left(\mathscr{P}_1\left(\mathbb{R}^d\right), \mathbf{W}_1\right)$ is a complete metric space. For any measure $\nu \in \mathscr{P}_1\left(\mathbb{R}^d\right)$, define $\mathrm{f} : \mathscr{P}_1\left(\mathbb{R}^d\right) \to \mathscr{P}_1\left(\mathbb{R}^d\right)$ as $\mathrm{f}(\nu) = \nu\mathrm{R}_{\varepsilon,\delta}$. Then for any $\delta \in (0, \bar{\delta}]$, there exists large enough $\mathrm{m}_\delta \in \mathbb{N}^*$ such that $\mathrm{f}^{\mathrm{m}_\delta}$ is a contractive mapping. Therefore we can apply the Picard fixed point theorem and we obtain that R_δ admits an unique invariant probability measure $\pi_{\delta,\lambda} \in \mathscr{P}_1\left(\mathbb{R}^d\right)$. Since $\pi_{\delta,\lambda}$ is subject to bias comparing with the solution of the SDE $\mathrm{d}X_t = b_\lambda(X_t)\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t$, in the Theorem below, we follow the proof in SM6.3 from [57] and give a nonasymptotic bias analysis:

Theorem 3.12. *Assume A 3.2, A 3.6, A 3.8, A 3.10. Assume* $V(x) = 1 + \|x\|^2, x \in \mathbb{R}^d$. *Let* $\lambda, \alpha, C, \mathrm{L}_y, \mathrm{L}$ *be the ones in NF-ULA (Algorithm 2.1). Let* m_y *be the parameter in A 3.10.*

Let $\lambda > 0$ such that $2\lambda(\mathrm{L}_y + \alpha\mathrm{L} - \min(\mathrm{m}_y, 0)) \leqslant 1$ and let $\bar{\delta} = (1/6)(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda)^{-1}$. Then for any $\delta \in (0, \bar{\delta}]$ and C convex and compact, R_δ admits an unique invariant probability measure $\pi_{\delta,\lambda}$. In addition, there exists $B_1, B_2, B_3 \geqslant 0$, $\tilde{\rho}_1 \in [0,1)$ such that for any $\delta \in (0, \bar{\delta}]$, $k \in \mathbb{N}^*$,

$$\left\| \boldsymbol{\delta}_x \mathrm{R}_\delta^k - \pi_\lambda \right\|_V \leqslant B_1 \tilde{\rho}_1^{k\delta} V^2(x) + B_2 V(x) \sqrt{\delta^2 k \left(d + \frac{B_3 \delta}{3} \right)}.$$

Proof. See Appendix A.7. ∎

Remark: Note that there is a trade-off of selecting the step-size δ. In order to achieve a small bias, one needs to set a large time interval $t = k\delta$, keep t fixed and use a small step size δ. However, larger k means drawing more samples, resulting in longer computation time. In practice, the burn-in period is incorporated in t, in which the Markov Chain is dramatically exploring the state space.

4. Experiments in Bayesian Imaging.

We apply NF-ULA and PnP-ULA on three inverse problems: image motion deblurring, image inpainting, and limited-angle computed tomography (CT) reconstruction. We compare with PnP-ULA since, to the best of our knowledge, it is the state-of-the-art Langevin algorithm with data-driven non-convex regularizers.

Choice of α: For both NF-ULA and PnP-ULA on different problems, we fine-tune α such that the peak signal-to-noise ratio (PSNR) of the sample mean gets maximized. While in most cases $\alpha \in (0, 5]$ works well, for NF-ULA it is also related to the architecture of the normalizing flow. For CT reconstruction, we use the pre-trained patchNR, a NF-based regularizer learned on medical images, from the code provided in [3] and choose $\alpha = 5000$. Notably, in the original implementation, the maximum a posteriori estimator was considered, and $\alpha = 700$ was the best choice.

Choices of C and λ: We only perform the study of choosing different C and λ in the deblurring experiments. From [56], a projection term $(\mathrm{Id} - \Pi_C)$ is introduced to PnP-ULA to make sure that the posterior satisfies the tail-decay condition. Therefore, for posterior distributions with a slower tail-decay, a smaller C is recommended. We found experimentally that NF-ULA was numerically stable when the NF prior was trained for more than 20 epochs, even with a large C. In this case, C is chosen to be large enough such that Π_C is never activated, since we do not expect to choose a small C to change the behaviors of NF-ULA if it already converges. For a normalizing flow that is not well trained (less than 5 epochs), it is recommended that C should be the same as the range C_R of the dataset. In the imaging problems, we have that $C_R = [0,1]^d$. See Table 1 for details on the algorithm behaviors of NF-ULA with different choices of C and normalizing flow architectures. For well-trained normalizing flows in NF-ULA and denoiser in PnP-ULA, we set $C = [-100, 100]^d$. Actually all the samples generated in Tables 2, 3, and 4 never escaped $[-0.2, 1.2]^d$, indicating that the projection $\Pi_C(x)$ was never activated. We keep $\lambda = 5 \times 10^{-5}$, even though different λ makes no difference in most of our experiments.

Choice of δ: From the convergence analysis in Theorem 3.11 and Theorem 3.12, any $\delta < (1/6)(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda)^{-1}$ should work. However, this upper bound is not a strict bound and in practice, it is not easy to know the Lipschitz constant L of $\nabla \log q_\theta(x)$. To give an upper bound of L, we calculate the spectral norm of $\nabla^2 \log q_\theta(x)$ through power iteration when x is randomly choosen in C_R and the spectral norm are smaller than 2×10^5. This upper bound for L is still too loose since we find that NF-ULA converges for many δ larger than the corresponding upper bound. Moreover, as different λ makes no difference in most of our experiments, we fine tune δ instead of precisely calculating the upperbound given by L and λ. In most of our experiments, δ is chosen to be smaller than $(1/10)\mathrm{L}_y^{-1}$, to ensure convergence of different algorithms. Our choice of δ is slightly different from PnP-ULA because the Lipschitz parameter L of the PnP prior can be set to 1 during training.

Implementations: We implement all the experiments in `Python` and utilize `PyTorch` for implementing the ULA Markov chains. The numerical experiments are run on Intel(R) Xeon(R) Platinum 8358P CPU with four Nvidia Tesla A100 GPUs. Codes for NF-ULA are available at Github[1].

4.1. Image Deblurring.

We first consider a non-blind motion deblurring problem on human face images. The corresponding forward operator A applies a convolution on the image x with a 9×9 motion-blurring kernel of horizontal blurring direction, with all the elements in the fifth row of the kernel being 1/9 and the other rows being 0. Both $x, y \in \mathbb{R}^d$, where $d = 3 \times 128 \times 128$ and the forward operator $A : \mathbb{R}^d \mapsto \mathbb{R}^d$ is linear. To describe the forward model (likelihood), we add Gaussian noise $n \sim \mathcal{N}(0, \sigma^2 I^d)$, leading to the following measurement equation and likelihood:

$$y = Ax + n, \quad p(y|x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|y - Ax\|^2}{2\sigma^2}\right).$$

4.1.1. Networks and Parameters.

To realize NF-ULA, we train the well-known flow-based model, Glow [50], on the human face dataset FFHQ [49] without the first 20 images, which amounts to 69980 images in total. All the images are 3-channel images normalized to $C_R = [0,1]^{3\times128\times128}$. We train Glow from scratch using the publicly available `PyTorch` implementation[2], however, NF-ULA can also use an appropriate pre-trained model. The architecture of Glow has five blocks with 32 flows in each block.

For PnP-ULA [56], we use the real spectral normalization DnCNN (realSN-DnCNN), which is a Lipschitz-continuous denoiser proposed in [84]. In order to see the behavior of the denoiser without the Lipschitz constraint, we train both the standard DnCNN [104] and realSN-DnCNN [84] on the image patches of a 980-image subset of FFHQ. To train the denoiser, we follow the same procedure reported in [56], i.e., we add Gaussian noise with the variance $\varepsilon = (5/255)^2$ on the training data batches. In fact, we also tested $\varepsilon = (15/255)^2$ or $(25/255)^2$ but the generated samples get lower PSNR. To train the standard DnCNN, we directly use the code in the Image Restoration Toolbox[3]. We keep the default parameter settings to train a 17-layer DnCNN on image patches of size 40×40. For realSN-DnCNN, the original implementation[4] in [84] only supports training on grayscale images, therefore we

[1]https://github.com/caiziruo/NF-ULA
[2]https://github.com/rosinality/glow-pytorch
[3]https://github.com/cszn/KAIR
[4]https://github.com/uclaopt/Provable_Plug_and_Play/

Table 1

The behavior of NF-ULA by different Glow and different choices of C. The algorithm does not converge For Glow with affine coupling layers. For Glow with additive coupling layers, the algorithm converges better when Glow is trained for more epochs.

Deblurring	network: Glow. $\alpha = 1.5$			
	coupling layers	epochs	C	PSNR
face1				
NF-ULA	affine	100	$[0, 1]^d$	divergent
NF-ULA	additive	5	$[-100, 100]^d$	divergent
NF-ULA	additive	5	$[0, 1]^d$	26.58
NF-ULA	additive	20	$[-100, 100]^d$	29.84
NF-ULA	additive	100	$[-100, 100]^d$	30.42

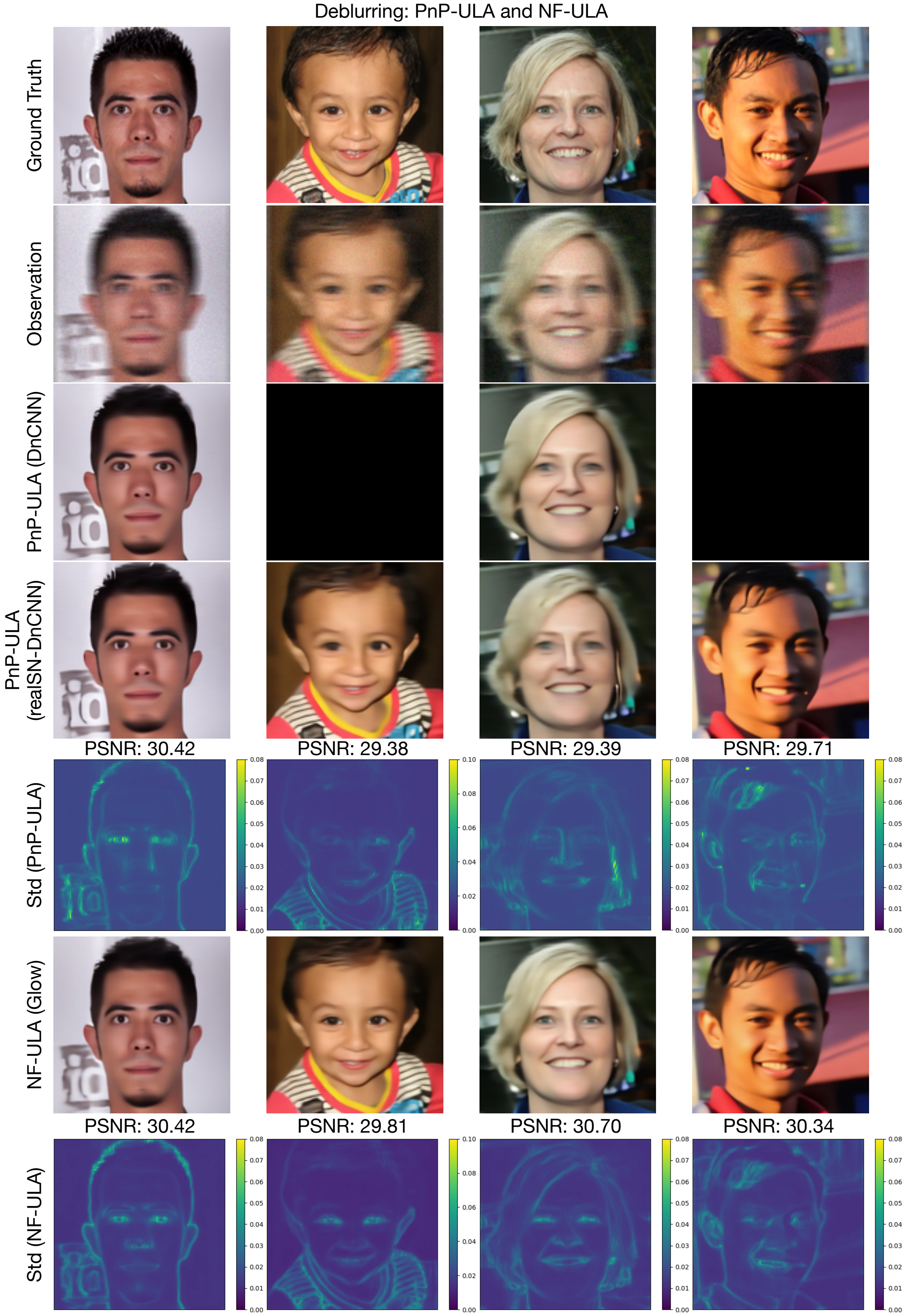
modified the code to make it applicable to color images. We also set up the number of network layers as 17 and preprocess the data to patches of size 40×40, while setting the Lipschitz parameter to 1. Although DnCNN and realSN-DnCNN are trained on such a small dataset, they can still obtain a peak signal-to-noise ratio (PSNR) of more than 40 dB on the validation set. In fact, the original implementation in [84] trains the denoiser on a dataset consisting of only 400 images, and increasing the size of the dataset does not necessarily lead to a higher PSNR on the validation set.

The Glow network that we used for NF-ULA has 100870544 parameters in total, while DnCNN has 559363 parameters and realSN-DnCNN has 558336 parameters. To train 100 epochs, Glow spent up to 100 hours, while DnCNN and realSN-DnCNN spent less than 5 hours. The heavier network and the longer training time for Glow pay off when it comes to reconstruction performance and image quality.

ULA parameters settings: We set the standard deviation of the Gaussian noise n to $\sigma = 0.02$. To ensure that both PnP-ULA and NF-ULA are numerically stable, we select the step size $\delta = 5 \times 10^{-5}$. For Glow, DnCNN and realSN-DnCNN, $\alpha = 1.5$ leads to the highest PSNR. We initialize $X_0 = y$, the noisy blurred observation for both NF-ULA and PnP-ULA.

4.1.2. Performance of the Algorithms. To explore the state space thoroughly, all the experiments have burn-in iterations less than 5000. Since the first sample X_0 is initialized as the observation y, the PSNR of the samples X_n starts from around 22.78 dB and then keeps going up and finally stays in an interval, e.g. $[29.0, 31.0]$. After the burn-in time, we calculate the posterior mean by obtaining 10000 samples and compute the PSNR of the sample mean. To draw 10000 samples, NF-ULA spends around 3100 seconds, while PnP-ULA spends 30 seconds. For both algorithms, calculating the posterior mean by more samples, e.g. 10^6 samples, does not improve the PSNR. When generating equal samples, NF-ULA spends more time mainly because of the large network Glow uses - the Glow we use has approximately 100 times more parameters than realSN-DnCNN. In fact, we found that computing and forwarding the auto-gradient function of $q_\theta(x)$ takes 10% longer time than forwarding $q_\theta(x)$ itself. However, we believe that NF-ULA has great potential to leverage smaller and more advanced normalizing flows to reduce computational time. In Sec. 4.3, we use a lightweight NF-based regularizer and the resulting NF-ULA requires significantly less time.

Figure 1. *Deblurring by PnP-ULA and NF-ULA. What each row represents is written on left of the rows. PSNR values corresponding to the sample mean are provided in Table 2. PnP-ULA with standard DnCNN does not converge on face2 and face4. On all four faces, NF-ULA (Glow) yields a higher PSNR (for the sample mean estimator) than PnP-ULA (realSN-DnCNN). The sample mean images also have a better visual quality for NF-ULA.*



To examine the Lipschitz continuity of $\nabla \log q_\theta(x)$ for different kinds of coupling layers, we train two different Glow networks for 100 epochs each, with affine and additive coupling layers, respectively. Also, to verify our hypothesis that better training of the normalizing flow prior will imply better samples from NF-ULA, we trained Glow (additive coupling layers) for 5, 20, and 100 epochs, and compared their performance when used in the NF-ULA framework. The PSNR values of the sample mean images corresponding to these variants of NF-ULA with different NF-based priors are reported in Table 1. With affine coupling layers in Glow, NF-ULA fails to converge because $\nabla \log q_\theta(x)$ is not Lipschitz continuous, which is consistent with Proposition 3.9. For Glow with additive coupling layers and also for the case where the Glow model is well-trained (more than 20 epochs), NF-ULA works well and the generated samples do not blow up, even in the case where $C = [-100, 100]^d$ is much bigger than C_R. This suggests that a well-trained prior $q_\theta(x)$ already satisfies the tail decay conditions, without imposing the projection $\mathrm{Id} - \Pi_C$. However, it is still essential for the theoretical study. For poorly trained Glow (less than 5 epochs) and large C, NF-ULA does not work well - most of the samples go far beyond C_R and the PSNR of them are below 10 dB. If C is set to be a much smaller set, e.g., $C = C_R$, then the PSNR can be up to 26 dB, which is still considerably lower than what one can achieve with a well-trained Glow.

Intuitively, $q_\theta(x)$ is more *diffusive* when Glow is trained for only a few epochs. After training for some epochs, the normalizing flow is more suitable to serve as an image prior, and the density $q_\theta(x)$ is more concentrated. Moreover, the tail decay condition of $p(x|y)$ is also satisfied with a well-trained prior, even without the projection term.

To compare the performance of ULA with both PnP- and normalizing flow-induced priors, we run NF-ULA using Glow, PnP-ULA using DnCNN, and PnP-ULA with realSN-DnCNN on four human face images randomly selected from the first 20 images of FFHQ [49], which are the ones not used during training. In the following experiments, we use Glow with additive coupling layers. Glow, DnCNN, and realSN-DnCNN are all trained for 100 epochs for a fair comparison. The results are shown in Figure 1 and Table 2. From Table 2, we note that NF-ULA with Glow generates samples with the highest PSNR. We also present the standard deviation of the samples on the same channel in Fig 1. NF-ULA has richer details for the posterior mean and more variations for standard deviation, particularly on the eyes, mouths, and hair. This is probably due to a more accurate prior learned by the generative model. It is worth noting that PnP-ULA with DnCNN shows great performance on Face-1 and Face-3, but is divergent on Face-2 and Face-4. However, PnP-ULA with realSN-DnCNN converges on all images, albeit with lower PSNR than NF-ULA. Moreover, we also performed the simulations of PnP-ULA using DRUnet [103], a newer denoiser than DnCNN, but the results are very comparable to the ones obtained with DnCNN - the algorithm is not convergent on Face-2 and Face-4 due to DRUnet not being Lipschitz.

We record the PSNR of the samples and the minimum mean square error (MMSE) estimator in Figure 2. It's about the deblurring experiments of face1 and the evolutions for face2, face3, and face4 are similar. In the left figure, we start from the burn-in period until 15000 samples. The MMSE estimator is approximated by the last 10000 samples. For both algorithms, the burn-in periods are less than 5000 samples. Regardless of the sampling time, NF-ULA shows a faster increase of PSNR, which means the convergence speed of the first-order moment for NF-ULA mildly outperforms PnP-ULA. However, in the right figure,

Table 2

Deblurring: Comparison of ULA with different priors for image deblurring. PnP-ULA with a standard DnCNN does not converge on face2 and face4. NF-ULA (Glow) generates samples with slightly higher PSNR than PnP-ULA.

Deblurring	net_epochs = 100, $C = [-100, 100]^d$		
	network	parameters	PSNR
face1			
NF-ULA	Glow	$\alpha = 1.5$	30.42
PnP-ULA	DnCNN	$\alpha = 1.5$	30.40
PnP-ULA	realSN-DnCNN	$\alpha = 1.5$	30.42
face2			
NF-ULA	Glow	$\alpha = 1.5$	29.81
PnP-ULA	DnCNN	$\alpha = 1.5$	divergent
PnP-ULA	realSN-DnCNN	$\alpha = 1.5$	29.38
face3			
NF-ULA	Glow	$\alpha = 1.5$	30.70
PnP-ULA	DnCNN	$\alpha = 1.5$	29.61
PnP-ULA	realSN-DnCNN	$\alpha = 1.5$	29.39
face4			
NF-ULA	Glow	$\alpha = 1.5$	30.34
PnP-ULA	DnCNN	$\alpha = 1.5$	divergent
PnP-ULA	realSN-DnCNN	$\alpha = 1.5$	29.71

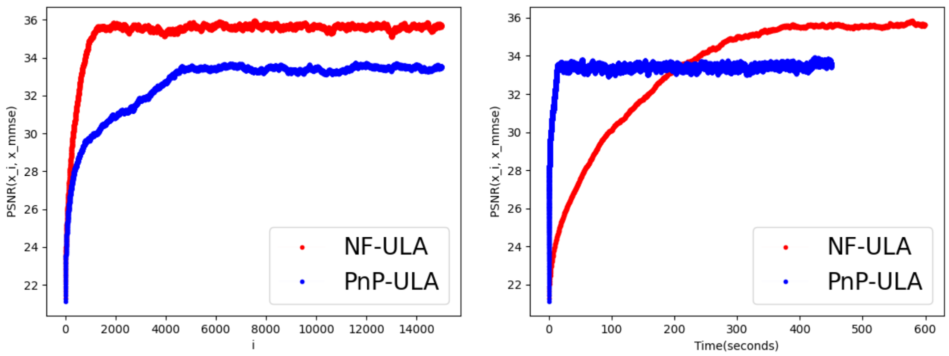


Figure 2. *The evolution of PSNR(x_i, x_{mmse}) of deblurring (face1). The left figure is according to the number of the samples and the right one is according to elapsed time. A faster increase means a faster convergence speed.*

we consider evolution w.r.t. the sampling time and NF-ULA has a slower increase of PSNR. NF-ULA has a burn-in time of about 400 seconds while PnP-ULA is less than 40 seconds.

One common approach to studying the convergence speed of a Markov chain is to calculate the d-dimensional auto-correlation function (ACF) of it. For samples $\{Y_i\}_{i=1}^N$ from a one-

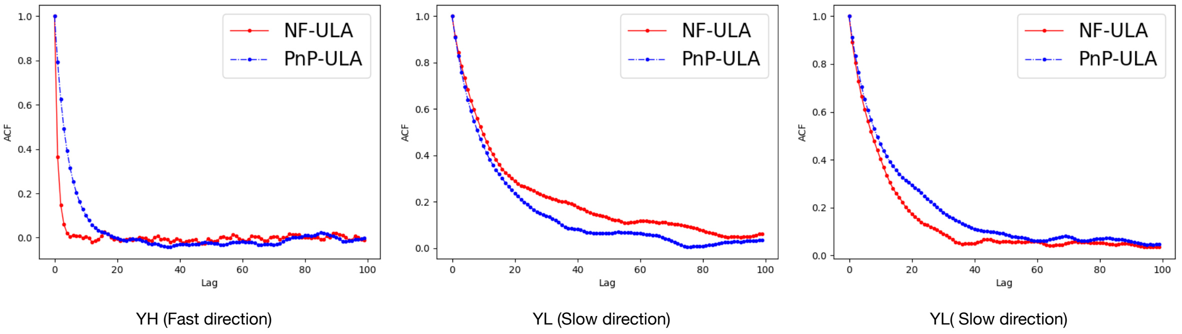


Figure 3. *The autocorrelation function (ACF) of the samples (deblurring on face1). The definition of the ACF is given in* (4.1)*. ACF is calculated by wavelet basis using the band-pass coefficients (YH) and the low-pass coefficients (YL). Faster decreasing ACF implies faster convergence of the Markov chain.*

dimensional Markov chain, the sample auto-correlation function is given by

$$\omega(l) = \frac{\sum_{t=1}^{n-l}(Y_{t+l} - \bar{Y})(Y_t - \bar{Y})}{\sum_{t=1}^{n}(Y_t - \bar{Y})^2}, \quad \bar{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t, \tag{4.1}$$

where $l = 0, 1, \cdots, n-1$, is the lag between the samples. Since the samples generated by ULA are not strictly uncorrelated, faster decreasing ACF means that the samples are less correlated and generally implies faster convergence of the Markov chain to some extent. Notably, the calculation of ACF is not easy in high-dimensional problems. Therefore, we firstly transform the image samples using wavelet basis and obtain the band-pass coefficients (YH) and the low-pass coefficients (YL). YH contains the image details while YL captures the overall image structure. We consider the finest scale coefficients in YH. To characterize the Markov chain generated by NF-ULA (Glow) and PnP-ULA (realSN-DnCNN), we randomly select 100 dimensions respectively from YH and YL, and calculate the ACF on those dimensions. It should be noted that the ACF can have different rates of decay in different directions, therefore it is time-consuming to analyze the ACF of all the image dimensions and calculate the fastest and slowest decreasing direction. However, ACF in YH mostly have faster decrease and ACF in YL will have slower decrease. In Fig 3, we show the convergence of ACF (face1), along one *fast direction* in YH and two *slow directions* in YL. In the fast direction, the ACF of PnP-ULA decreases from 1 to 0 within about 20 lags, while for NF-ULA it converges even faster (within approx. 10 lags). For slow directions, both NF-ULA and PnP-ULA hold a non-zero ACF until more than 40 lags, and it is not immediately clear which of these two methods has a faster decay of the ACF. ACF of face2, face3 and face4 are similar as face1 and hence omitted here.

4.2. Image Inpainting. In this section, we present the experimental results on image inpainting. We still consider human face images and use the Glow and realSN-DnCNN networks trained as explained in Sec. 4.1. For inpainting, the forward operator A applies masking on x so that 80% of the pixels in x are missing. We choose different α to ensure both NF-ULA and PnP-ULA have the best performance: $\alpha = 2.0$ works well for NF-ULA, while for

Figure 4. *Comparison of image inpainting performance of PnP-ULA and NF-ULA. The PSNR values of the sample mean images are reported in Table 3. NF-ULA (Glow) yields a higher PSNR (by approximately 2.5-3.0 dB) of the sample mean images than PnP-ULA with a realSN-DnCNN denoiser. This experiment underscores the importance of stronger regularization (which the Glow-based prior can achieve) when the forward operator is severely ill-posed.*

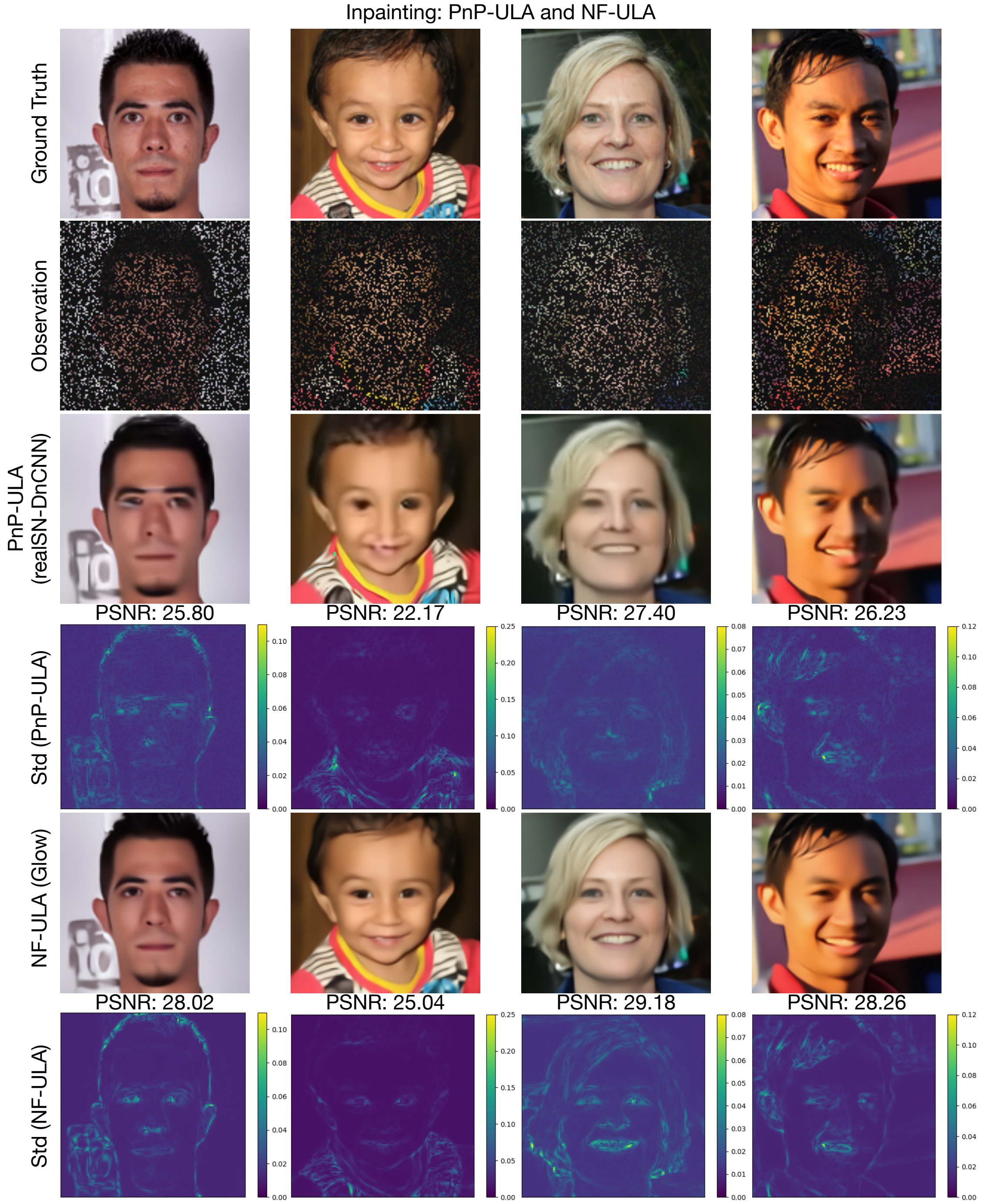


Table 3

Inpainting: Comparison of the ULA with different priors. The parameter α is fine-tuned to maximize the PSNR for both algorithms. Since inpainting relies more on the prior, NF-ULA has a higher PSNR for the sample mean as compared with PnP-ULA.

Inpainting	net_epochs $= 100$, $C = [-100, 100]^d$		
	network	parameters	PSNR
face1			
NF-ULA	Glow	$\alpha = 2$	28.02
PnP-ULA	realSN-DnCNN	$\alpha = 2.5$	25.80
face2			
NF-ULA	Glow	$\alpha = 2$	25.04
PnP-ULA	realSN-DnCNN	$\alpha = 2.5$	22.17
face3			
NF-ULA	Glow	$\alpha = 2$	29.18
PnP-ULA	realSN-DnCNN	$\alpha = 2.5$	27.40
face4			
NF-ULA	Glow	$\alpha = 2$	28.26
PnP-ULA	realSN-DnCNN	$\alpha = 2.5$	26.23

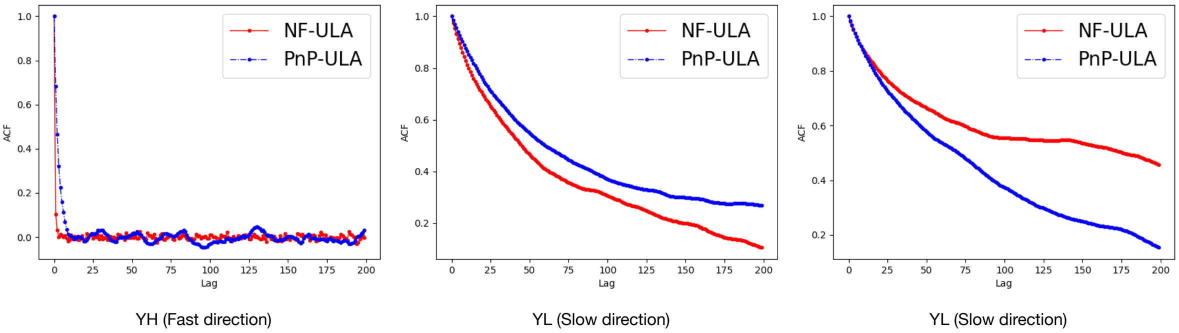


Figure 5. *The auto-correlation function (ACF) of the samples (inpainting on face1). The definition of ACF is given in* (4.1)*. ACF is calculated by wavelet basis using the band-pass coefficients (YH) and the low-pass coefficients (YL). Faster decreasing ACF implies faster convergence of the Markov chain.*

PnP-ULA $\alpha = 2.5$ works the best. We maintain the same setting for the other important hyper-parameters of the experiment, such as the noise standard deviation $\sigma = 0.02$, the dimension of image and observation $x, y \in \mathbb{R}^d = \mathbb{R}^{3\times128\times128}$, the step-size of both algorithms $\delta = 5 \times 10^{-5}$, the convex set $C = [-100, 100]^d$, and the initialization $X_0 = y$.

Performance of the algorithms: In contrast with deblurring, we found that both NF-ULA and PnP-ULA have much longer burn-in times. We initialize X_0 with the measurement y, whose PSNR is only 5.46 dB. NF-ULA has a burn-in iteration of 10000 until the PSNR of X_n grows more than 25 dB and becomes stable, while PnP-ULA takes about 80000-iterations (eight times larger than NF-ULA) for burn-in. The reason might be that Glow's powerful

Table 4

Limited-angle CT reconstruction from Gaussian noise-corrupted limited-angle projection data. α is chosen to maximize the PSNR for both PnP-ULA and NF-ULA to make a fair comparison. NF-ULA leads to a higher sample mean PSNR than PnP-ULA.

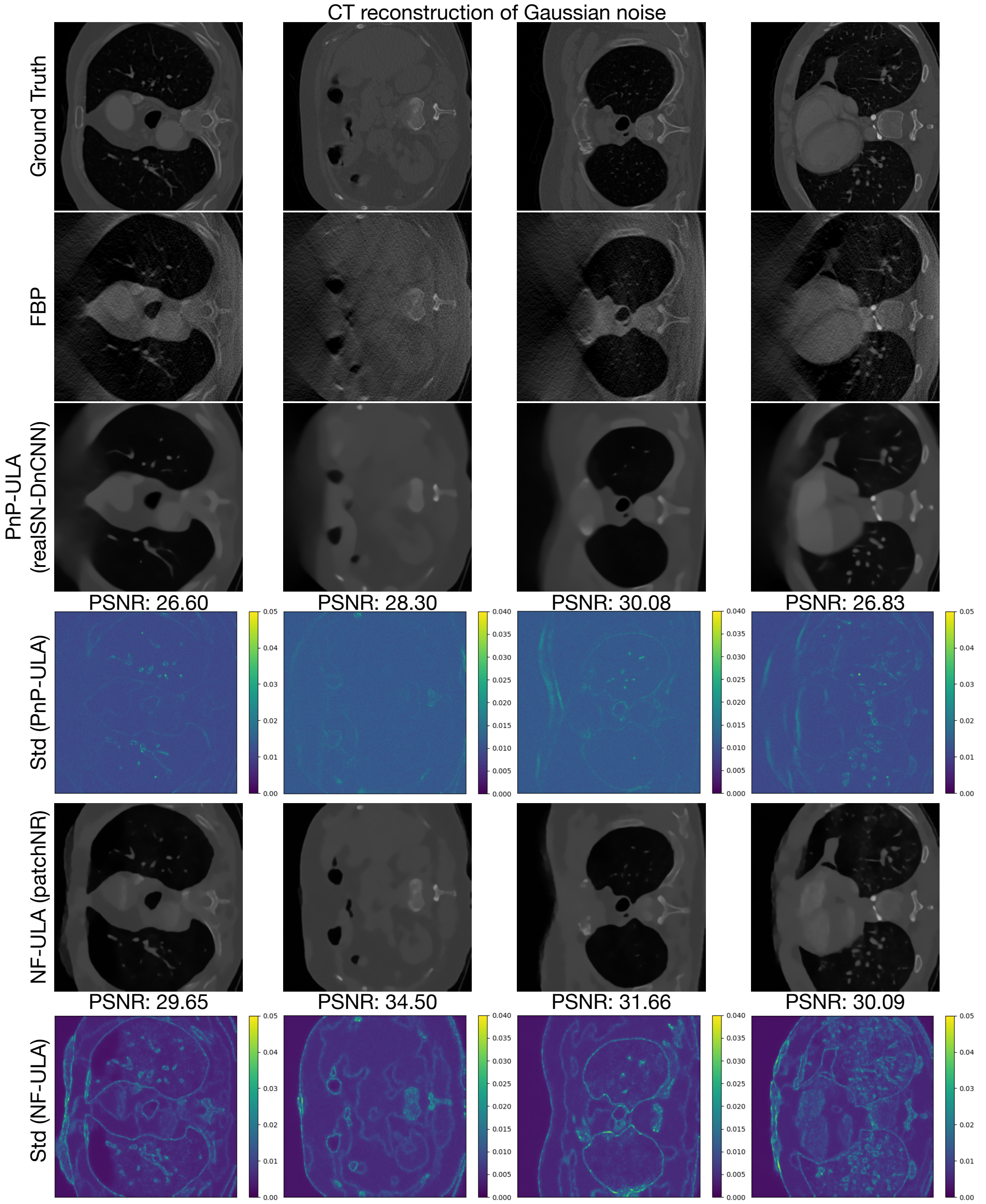
CT	$C = [-100, 100]^d$		
	network	parameters	PSNR
Image-1			
NF-ULA	PatchNR	$\alpha = 5000$	29.65
PnP-ULA	realSN-DnCNN	$\alpha = 3$	26.60
Image-2			
NF-ULA	PatchNR	$\alpha = 5000$	34.50
PnP-ULA	realSN-DnCNN	$\alpha = 3$	28.30
Image-3			
NF-ULA	PatchNR	$\alpha = 5000$	31.66
PnP-ULA	realSN-DnCNN	$\alpha = 3$	30.08
Image-4			
NF-ULA	PatchNR	$\alpha = 5000$	30.09
PnP-ULA	realSN-DnCNN	$\alpha = 3$	26.83

prior information accelerates the burn-in process, particularly on the pixels missing in the observation. After the burn-in time, we draw 10000 samples and compute the PSNR of the samples' mean. Drawing 10000 samples takes approximately the same time as in the deblurring experiment.

The sample mean images and the standard deviations are shown in Fig. 4. As compared with PnP-ULA, NF-ULA recovers more areas of the face and shows higher uncertainties on eyes, hairs, noses, and teeth. Those areas are easily distinguishable between different human faces and should have higher uncertainties than other areas, e.g., foreheads and cheeks. From Table 3, we observe that NF-ULA achieves a higher PSNR than PnP-ULA. For both NF-ULA and PnP-ULA, the PSNR of the posterior mean is lower than that of the deblurring experiment - the forward operator of masking 80% pixels is not invertible and the observation y in inpainting is ill-conditioned, which means that in the Bayesian setting, the samples rely on the prior than the likelihood. In such cases, NF-ULA provides a stronger and more informative prior as compared to PnP-ULA.

To calculate the ACF in this inpainting results, we use the same strategy as in deblurring: calculating the ACF respectively on 100 randomly selected dimensions of YH and YL. In Fig. 5, we show the ACF including one fast direction in YH and two slow directions in YL. Similar to Fig. 3, among those fast decreasing directions, the ACF of NF-ULA is slightly faster than PnP-ULA and they both decrease from 1 to 0 within 20 lags. For slow directions, both algorithms have slower decreasing ACF than the deblurring experiments and we cannot conclude for which method, the ACF decreases faster. ACF of face2, face3 and face4 are similar as face1 and omitted.

Figure 6. *CT reconstruction of Gaussian noise (limited angles). What each column represents is written on top of the columns. PSNR of the samples mean are provided in Table 4. NF-ULA (patchNR) yields higher PSNR of samples mean and better samples Std than PnP-ULA (realSN-DnCNN).*



4.3. CT Reconstruction from limited-angle measurements. We consider the classical ill-posed inverse problem of X-ray CT reconstruction from limited-angle projection data. We use the `torch_radon` library [82] to model the forward operator A that computes projections using a fan-beam acquisition geometry. Instead of considering the full angular range $[0, 2\pi]$, we only have projection data corresponding to an angular sweep over the range $[0.1\,\pi, 0.9\,\pi]$ of angles. We set the number of detector elements to 144, and test the algorithms for both Gaussian noise and Poisson noise (see Appendix B). The noisy projection data is given by

$$y = Ax + n \text{ or } y \sim P(Ax), \tag{4.2}$$

where n is used to denote additive Gaussian noise and $P(Ax)$ denotes adding a non-additive noise on Ax such as Poisson noise. The image to be recovered is $x \in \mathbb{R}^{362\times 362}$ and the sinogram is $y \in \mathbb{R}^{144\times 512}$. We calculate the norm of A and obtain that $\|A\| = \sup_{x:\|x\|=1} \|Ax\| \approx 100$.

Network architecture: The features and textures of medical images are more difficult to learn as compared with those in natural images. Hence, normalizing flows do not have comparable performance in generating semantically meaningful images for medical imaging applications, unlike applications involving natural images. Therefore, we utilize *patchNR* [3], which is analogous to normalizing flow, to apply NF-ULA for CT reconstruction. PatchNR is a powerful regularizer that involves Glow coupling layers learned on small patches extracted from very few images (only six images), which has shown promising results for CT reconstruction [3]. PatchNR uses five GlowCoupling blocks and permutations in an alternating manner, where the coupling blocks are from the FrEIA package [6]. The three-layer subnetworks are fully connected with ReLU activation functions and 512 nodes, which overall result in a much smaller network than Glow. It should be noted that extracting the patches from an image is not a reversible process, therefore patchNR actually learns the prior over the image patches and cannot do unconditional sampling using $x = T(z)$. Even so, the log gradient is still computable and Lipschitz continuous, since its GlowCoupling blocks satisfy Proposition 3.9.

The patchNR we used is given by the pre-trained model[5] trained on six images from the LoDoPaB dataset [60]. For PnP-ULA, we train the denoiser realSN-DnCNN on a 128-image subset of LoDoPaB, by adding Gaussian noise with the variance $\varepsilon = (5/255)^2$ on the training data batches. We train a 17-layer realSN-DnCNN on the preprocessed image patches with size 40×40. The Lipschitz parameter of the realSN-DnCNN is set to 1. The patchNR has 2908880 parameters in total and the realSN-DnCNN has 556032 parameters.

ULA parameters settings: While in [3] $\alpha = 700$ is the default setting of the considered maximum a posteriori estimator, $\alpha = 5000$ (Gaussian noise) works fine for NF-ULA. For PnP-ULA we set $\alpha = 3$. We use a smaller step size for both algorithms, namely $\delta = 10^{-6}$, to ensure convergence, since in CT reconstruction the forward operator A has a larger norm (approximately 100) than deblurring and inpainting. The convex set is set to be $C = [-100, 100]^d$. We initialize X_0 using the filtered back-projection (FBP) reconstruction.

Gaussian noise-corrupted measurement: We first test the case with additive Gaussian noise. To be more specific, we add Gaussian noise $n \sim \mathcal{N}(0, \sigma^2 I^m)$ in (4.2) to the clean projection data. Since $\|A\| \approx 100$, we select $\sigma = 1.0$ to simulate the noisy sinogram y. The

[5] https://github.com/FabianAltekrueger/patchNR

likelihood can be expressed as

$$p(y|x) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\frac{\|y - Ax\|^2}{2\sigma^2}\right). \tag{4.3}$$

Since the gradient of the log-likelihood is not globally Lipschitz for Poisson likelihood, the additional experiments with Poisson noise are moved to Appendix B. Note that NF-ULA with Poisson likelihood still converges although the assumptions needed for the theoretical guarantees do not hold, which warrants further investigations and we leave it for future work.

Performance of the algorithms: We test PnP-ULA and NF-ULA on another four images from LoDoPaB [60] which were not used for training the patchNR network utilized by NF-ULA and the realSN-DnCNN denoiser used in PnP-ULA. They are different from the six images trained by patchNR and 128 images trained by realSN-DnCNN. The four ground-truth images used for evaluating the performance of NF-ULA and PnP-ULA for limited-angle CT are shown in the first column of Fig. 6.

Both PnP-ULA and NF-ULA have more than 20000 burn-in iterations. Since we initialize by setting X_0 equal to the FBP reconstruction, the PSNR of X_n starts from around 21.90 dB, then slowly increases, and finally stabilizes. Note that for different test images, the burn-in time varies. For Image-2 in Table 4, PnP-ULA has 30000 burn-in iterations, and the PSNR of the samples never exceeds 29 dB. In contrast, the PSNR of the samples increases until 33 dB for NF-ULA and finally the burn-in time for NF-ULA on Image-2 is around 70000 iterations.

After the burn-in time, we calculate the posterior mean and the standard deviation around it by obtaining 10000 samples and computing the PSNR of the samples' mean. For Gaussian noise, drawing 10000 samples by NF-ULA takes around 500 seconds, whereas, for PnP-ULA, it takes about 70 seconds. Thanks to the smaller network size of patchNR compared to Glow, it saves a large proportion of time in computation.

Fig. 6 shows the ground-truth images (1st column), the FBP (2nd column), the posterior mean and standard deviation of PnP-ULA (in Columns 3 and 4, respectively), and those corresponding to NF-ULA (in Columns 5 and 6, respectively). The posterior mean images indicate that NF-ULA has a significantly better sample quality than PnP-ULA, which exhibits poor reconstruction in the left area, due to the missing angles and the extremely ill-posed problem. NF-ULA can recover the details well, which is consistent with the results in [3] that patchNR works well in the limited-angle CT experiments. For standard deviation in the case of Gaussian noise, NF-ULA shows more realistic uncertainties than PnP-ULA in most areas but still has relatively large uncertainties in the left area (where no projection is available). Table 4 shows the PSNR of the posterior mean. NF-ULA achieves a considerably higher PSNR than PnP-ULA.

We also compare the ACF (Image-1) in Fig. 7 to study the convergence speed. The ACF is calculated by randomly selecting 100 dimensions respectively from YH and YL. The ACF on the fast direction is different from deblurring and inpainting: On fastest directions NF-ULA decreases from 1 to 0 within 100 lags and the independence is achieved, while the independence of PnP-ULA is not achieved (as shown in the first sub-figure). On some fast directions, the independence of NF-ULA and PnP-ULA is both not well achieved, as demonstrated in the second sub-figure. For slow directions, both two algorithms decrease slowly and independence

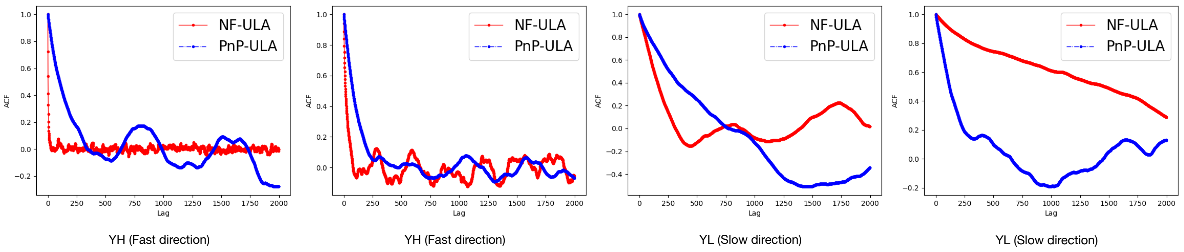


Figure 7. *The autocorrelation function (ACF) of the samples (Gaussian noise CT on Image-1). The definition of ACF is given in* (4.1)*. ACF is calculated by wavelet basis using the band-pass coefficients (YH) and the low-pass coefficients (YL). Faster decreasing ACF implies faster convergence of the Markov chain. On slow directions, the independence are not achieved for both algorithms.*

is not achieved. ACF of Image-2, Image-3 and Image-4 are similar and omitted.

5. Conclusion and Outlook.

We introduced NF-ULA, a Langevin diffusion-based Monte Carlo algorithm, which takes advantage of a normalizing flow for prior density estimation. The normalizing flow can be pre-trained agnostic to the forward operator of the inverse problem that one seeks to solve. Since NF-ULA only requires the log gradient of the prior, our algorithm still works in cases where the normalizing flow can only evaluate the density but cannot do unconditional sampling. To guarantee that the posterior distribution is well-defined, we follow [56] to add a projection operator onto a convex and compact subset of the image space, although in most cases the projection is not activated, for instance, if the prior is well-trained. Since the density of normalizing flow itself can be evaluated, NF-ULA can be extended to a Metropolis-adjusted version, which is left for future studies. For the theoretical analysis of NF-ULA, we first prove the well-posedness of the posterior distribution that we aim to draw samples from. To prove the convergence of NF-ULA, the most essential condition is the Lipschitz drift, and we, therefore, derive a sufficient condition for having a Lipschitz-continuous gradient of the log density of the normalizing flow. Moreover, we show that NF-ULA admits an unique invariant distribution, and we give a non-asymptotic bound on the bias. We demonstrate our method through several Bayesian imaging experiments, namely image deblurring, image inpainting, and limited-angle CT reconstruction. We show that better training of the normalizing flows leads to better samples and convergence of NF-ULA. Although currently, NF-ULA has a longer sampling time because of the large network of normalizing flows, it has the potential to use a better and smaller network to reduce the computation in the future.

There are still some unanswered questions about NF-ULA. Although we give a sufficient condition for the gradient of the log density of normalizing flow to be Lipschitz, the condition might be relaxed, or it might even be possible to derive a condition that is both necessary and sufficient. Moreover, given different curvature conditions [22, 65] on the drift other than Lipschitz, the studies of ULA on non-convex potentials have shown different convergence results and they can also be applied to NF-ULA. However, this might require re-training the normalizing flows to enforce such conditions and necessitates further research. Meanwhile when the Lipschitz assumption does not hold, the results of our Poisson noise experiments lack an explanation, which also requires a more detailed study.

Acknowledgments. CBS acknowledges support from the Philip Leverhulme Prize, the Royal Society Wolfson Fellowship, the EPSRC advanced career fellowship EP/V029428/1, EPSRC grants EP/S026045/1 and EP/T003553/1, EP/N014588/1, EP/T017961/1, the Wellcome Innovator Awards 215733/Z/19/Z and 221633/Z/20/Z, the European Union Horizon 2020 research and innovation program under the Marie Skodowska-Curie grant agreement No. 777826 NoMADS, the Cantab Capital Institute for the Mathematics of Information and the Alan Turing Institute. ZC and XZ were partially supported by NSFC (No. 12090024) and Sino-German center grant (No.M-0187) .

REFERENCES

[1] J. Adler and O. Öktem, *Learned primal-dual reconstruction*, IEEE transactions on medical imaging, 37 (2018), pp. 1322–1332.

[2] C. Aguerrebere, A. Almansa, J. Delon, Y. Gousseau, and P. Musé, *A bayesian hyperprior approach for joint image denoising and interpolation, with an application to hdr imaging*, IEEE Transactions on Computational Imaging, 3 (2017), pp. 633–646.

[3] F. Altekrüger, A. Denker, P. Hagemann, J. Hertrich, P. Maass, and G. Steidl, *Patchnr: Learning from very few images by patch normalizing flow regularization*, arXiv preprint arXiv:2205.12021, (2022).

[4] F. Altekrüger, P. Hagemann, and G. Steidl, *Conditional generative models are provably robust: Pointwise guarantees for bayesian inverse problems*, arXiv preprint arXiv:2303.15845, (2023).

[5] B. Amos, L. Xu, and J. Z. Kolter, *Input convex neural networks*, in International Conference on Machine Learning, PMLR, 2017, pp. 146–155.

[6] L. Ardizzone, T. Bungert, F. Draxler, U. Köthe, J. Kruse, R. Schmier, and P. Sorrenson, *Framework for Easily Invertible Architectures (FrEIA)*, 2018-2022, https://github.com/vislearn/FrEIA.

[7] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, *Solving inverse problems using data-driven models*, Acta Numerica, 28 (2019), pp. 1–174.

[8] M. Asim, M. Daniels, O. Leong, A. Ahmed, and P. Hand, *Invertible generative models for inverse problems: mitigating representation error and dataset bias*, in International Conference on Machine Learning, PMLR, 2020, pp. 399–409.

[9] H. H. Bauschke, P. L. Combettes, et al., *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408, Springer, 2011.

[10] M. Benning and M. Burger, *Modern regularization methods for inverse problems*, Acta Numerica, 27 (2018), pp. 1–111.

[11] A. Blake, P. Kohli, and C. Rother, *Markov random fields for vision and image processing*, MIT press, 2011.

[12] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational inference: A review for statisticians*, Journal of the American statistical Association, 112 (2017), pp. 859–877.

[13] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock, *An introduction to total variation for image analysis*, Theoretical foundations and numerical methods for sparse recovery, 9 (2010), p. 227.

[14] X. Cheng, N. S. Chatterji, Y. Abbasi-Yadkori, P. L. Bartlett, and M. I. Jordan, *Sharp convergence rates for langevin dynamics in the nonconvex setting*, arXiv preprint arXiv:1805.01648, (2018).

[15] F. Coeurdoux, N. Dobigeon, and P. Chainais, *Normalizing flow sampling with langevin dynamics in the latent space*, arXiv preprint arXiv:2305.12149, (2023).

[16] F. Coeurdoux, N. Dobigeon, and P. Chainais, *Plug-and-play split gibbs sampler: embedding deep generative priors in bayesian inference*, arXiv preprint arXiv:2304.11134, (2023).

[17] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, Fixed-point algorithms for inverse problems in science and engineering, (2011), pp. 185–212.
[18] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Modeling & Simulation, 4 (2005), pp. 1168–1200.
[19] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, *Generative adversarial networks: An overview*, IEEE signal processing magazine, 35 (2018), pp. 53–65.
[20] A. S. Dalalyan, *Theoretical guarantees for approximate sampling from smooth and log-concave densities*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79 (2017), pp. 651–676.
[21] A. S. Dalalyan and A. Karagulyan, *User-friendly guarantees for the langevin monte carlo with inaccurate gradient*, Stochastic Processes and their Applications, 129 (2019), pp. 5278–5311.
[22] V. De Bortoli and A. Durmus, *Convergence of diffusions and their discretizations: from continuous to discrete processes and back*, arXiv preprint arXiv:1904.09808, (2019).
[23] P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, Advances in neural information processing systems, 34 (2021), pp. 8780–8794.
[24] L. Dinh, D. Krueger, and Y. Bengio, *Nice: Non-linear independent components estimation*, arXiv preprint arXiv:1410.8516, (2014).
[25] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, arXiv preprint arXiv:1605.08803, (2016).
[26] R. Douc, E. Moulines, P. Priouret, and P. Soulier, *Markov chains*, Springer, 2018.
[27] A. Durmus, S. Majewski, and B. Miasojedow, *Analysis of langevin monte carlo via convex optimization*, The Journal of Machine Learning Research, 20 (2019), pp. 2666–2711.
[28] A. Durmus and E. Moulines, *Nonasymptotic convergence analysis for the unadjusted langevin algorithm*, The Annals of Applied Probability, 27 (2017), pp. 1551–1587.
[29] A. Durmus, E. Moulines, and M. Pereyra, *Efficient bayesian computation by proximal markov chain monte carlo: when langevin meets moreau*, SIAM Journal on Imaging Sciences, 11 (2018), pp. 473–506.
[30] B. Efron, *Tweedie's formula and selection bias*, Journal of the American Statistical Association, 106 (2011), pp. 1602–1614.
[31] M. A. Erdogdu and R. Hosseinzadeh, *On the convergence of langevin monte carlo: The interplay between tail growth and smoothness*, in Conference on Learning Theory, PMLR, 2021, pp. 1776–1822.
[32] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*, CRC press, 1995.
[33] D. Gilton, G. Ongie, and R. Willett, *Learned patch-based regularization for inverse problems in imaging*, in 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), IEEE, 2019, pp. 211–215.
[34] D. Gilton, G. Ongie, and R. Willett, *Deep equilibrium architectures for inverse problems in imaging*, IEEE Transactions on Computational Imaging, 7 (2021), pp. 1123–1133.
[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 27 (2014).
[36] P. Hagemann, J. Hertrich, and G. Steidl, *Stochastic normalizing flows for inverse problems: a markov chains viewpoint*, SIAM/ASA Journal on Uncertainty Quantification, 10 (2022), pp. 1162–1190.
[37] P. Hagemann and S. Neumayer, *Stabilizing invertible neural networks using mixture models*, Inverse Problems, 37 (2021), p. 085002.
[38] J. Hertrich, S. Neumayer, and G. Steidl, *Convolutional proximal neural networks and plug-and-play algorithms*, Linear Algebra and its Applications, 631 (2021), pp. 203–234.
[39] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, Advances in neural information processing systems, 33 (2020), pp. 6840–6851.
[40] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, *Stochastic variational inference*, Journal of Machine Learning Research, (2013).
[41] A. Houdard, C. Bouveyron, and J. Delon, *High-dimensional mixture models for unsupervised image*

denoising (hdmi), SIAM Journal on Imaging Sciences, 11 (2018), pp. 2815–2846.

[42] S. Hurault, A. Leclaire, and N. Papadakis, *Gradient step denoiser for convergent plug-and-play*, in International Conference on Learning Representations, 2022.

[43] N. Ikeda and S. Watanabe, *Stochastic differential equations and diffusion processes*, Elsevier, 2014.

[44] P. Jaini, I. Kobyzev, Y. Yu, and M. Brubaker, *Tails of lipschitz triangular flows*, in International Conference on Machine Learning, PMLR, 2020, pp. 4673–4681.

[45] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, *Deep convolutional neural network for inverse problems in imaging*, IEEE Transactions on Image Processing, 26 (2017), pp. 4509–4522.

[46] J. Kaipio and E. Somersalo, *Statistical and computational inverse problems*, vol. 160, Springer Science & Business Media, 2006.

[47] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, *Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications*, IEEE Signal Processing Magazine, 40 (2023), pp. 85–97.

[48] I. Karatzas, I. Karatzas, S. Shreve, and S. E. Shreve, *Brownian motion and stochastic calculus*, vol. 113, Springer Science & Business Media, 1991.

[49] T. Karras, S. Laine, and T. Aila, *A style-based generator architecture for generative adversarial networks*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4401–4410.

[50] D. P. Kingma and P. Dhariwal, *Glow: Generative flow with invertible 1x1 convolutions*, Advances in neural information processing systems, 31 (2018).

[51] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, *Improved variational inference with inverse autoregressive flow*, Advances in neural information processing systems, 29 (2016).

[52] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114, (2013).

[53] E. Kobler, A. Effland, K. Kunisch, and T. Pock, *Total deep variation: A stable regularization method for inverse problems*, IEEE transactions on pattern analysis and machine intelligence, 44 (2021), pp. 9163–9180.

[54] I. Kobyzev, S. J. Prince, and M. A. Brubaker, *Normalizing flows: An introduction and review of current methods*, IEEE transactions on pattern analysis and machine intelligence, 43 (2020), pp. 3964–3979.

[55] J. Latz, *On the well-posedness of bayesian inverse problems*, SIAM/ASA Journal on Uncertainty Quantification, 8 (2020), pp. 451–482.

[56] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, *Bayesian imaging using plug & play priors: when langevin meets tweedie*, SIAM Journal on Imaging Sciences, 15 (2022), pp. 701–737.

[57] R. Laumont, V. D. Bortoli, A. Almansa, J. Delon, A. Durmus, and M. Pereyra, *Supplementary materials: Bayesian imaging using plug & play priors: when langevin meets tweedie*. https://epubs.siam.org/doi/suppl/10.1137/21M1406349/suppl_file/M140634_01.pdf, 2022.

[58] J. Lehec, *The langevin monte carlo algorithm in the non-smooth log-concave case*, arXiv preprint arXiv:2101.10695, (2021).

[59] V. Lempitsky, A. Vedaldi, and D. Ulyanov, *Deep image prior*, in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 9446–9454.

[60] J. Leuschner, M. Schmidt, D. O. Baguer, and P. Maass, *Lodopab-ct, a benchmark dataset for low-dose computed tomography reconstruction*, Scientific Data, 8 (2021), p. 109.

[61] Q. Liu and D. Wang, *Stein variational gradient descent: A general purpose bayesian inference algorithm*, Advances in neural information processing systems, 29 (2016).

[62] C. Louchet and L. Moisan, *Posterior expectation of the total variation model: properties and experiments*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 2640–2684.

[63] S. Lunz, O. Öktem, and C.-B. Schönlieb, *Adversarial regularizers in inverse problems*, Advances in neural information processing systems, 31 (2018).

[64] T. D. Luu, J. Fadili, and C. Chesneau, *Sampling from non-smooth distributions through langevin diffusion*, Methodology and Computing in Applied Probability, 23 (2021), pp. 1173–1201.

[65] M. B. Majka, A. Mijatović, and Łukasz Szpruch, *Nonasymptotic bounds for sampling algorithms*

without log-concavity, The Annals of Applied Probability, 30 (2020), pp. 1534 – 1581, https://doi.org/10.1214/19-AAP1535.

[66] S. P. Meyn and R. L. Tweedie, *Stability of markovian processes iii: Foster–lyapunov criteria for continuous-time processes*, Advances in Applied Probability, 25 (1993), pp. 518–548.

[67] V. Monga, Y. Li, and Y. C. Eldar, *Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing*, IEEE Signal Processing Magazine, 38 (2021), pp. 18–44.

[68] W. Mou, N. Flammarion, M. J. Wainwright, and P. L. Bartlett, *An efficient sampling algorithm for non-smooth composite potentials*, Journal of Machine Learning Research, 23 (2022), pp. 1–50.

[69] S. Mukherjee, M. Carioni, O. Öktem, and C.-B. Schönlieb, *End-to-end reconstruction meets data-driven regularization for inverse problems*, Advances in Neural Information Processing Systems, 34 (2021), pp. 21413–21425.

[70] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb, *Learned convex regularizers for inverse problems*, arXiv preprint arXiv:2008.02839, (2020).

[71] R. Neal, *Bayesian learning via stochastic dynamics*, Advances in neural information processing systems, 5 (1992).

[72] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, *Deep learning techniques for inverse problems in imaging*, IEEE Journal on Selected Areas in Information Theory, 1 (2020), pp. 39–56.

[73] X. Pan, X. Zhan, B. Dai, D. Lin, C. C. Loy, and P. Luo, *Exploiting deep generative prior for versatile image restoration and manipulation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2021), pp. 7474–7489.

[74] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference.*, J. Mach. Learn. Res., 22 (2021), pp. 1–64.

[75] G. Papamakarios, T. Pavlakou, and I. Murray, *Masked autoregressive flow for density estimation*, Advances in neural information processing systems, 30 (2017).

[76] M. Pereyra, *Proximal markov chain monte carlo algorithms*, Statistics and Computing, 26 (2016), pp. 745–760.

[77] J.-C. Pesquet, A. Repetti, M. Terris, and Y. Wiaux, *Learning maximally monotone operators for image recovery*, SIAM Journal on Imaging Sciences, 14 (2021), pp. 1206–1237.

[78] J. Prost, A. Houdard, A. Almansa, and N. Papadakis, *Learning local regularization for variational image restoration*, in Scale Space and Variational Methods in Computer Vision: 8th International Conference, SSVM 2021, Virtual Event, May 16–20, 2021, Proceedings, Springer, 2021, pp. 358–370.

[79] D. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in International conference on machine learning, PMLR, 2015, pp. 1530–1538.

[80] G. O. Roberts and R. L. Tweedie, *Exponential convergence of langevin distributions and their discrete approximations*, Bernoulli, (1996), pp. 341–363.

[81] Y. Romano, M. Elad, and P. Milanfar, *The little engine that could: Regularization by denoising (red)*, SIAM Journal on Imaging Sciences, 10 (2017), pp. 1804–1844.

[82] M. Ronchetti, *Torchradon: Fast differentiable routines for computed tomography*, arXiv preprint arXiv:2009.14788, (2020), https://arxiv.org/abs/arXiv:2009.14788.

[83] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, Springer, 2015, pp. 234–241.

[84] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, *Plug-and-play methods provably converge with properly trained denoisers*, in International Conference on Machine Learning, PMLR, 2019, pp. 5546–5557.

[85] A. Salim, D. Kovalev, and P. Richtárik, *Stochastic proximal langevin algorithm: Potential splitting and nonasymptotic rates*, Advances in Neural Information Processing Systems, 32 (2019).

[86] A. Salmona, V. De Bortoli, J. Delon, and A. Desolneux, *Can push-forward generative models fit multimodal distributions?*, Advances in Neural Information Processing Systems, 35 (2022), pp. 10766–10779.

[87] Y. Song, L. Shen, L. Xing, and S. Ermon, *Solving inverse problems in medical imaging with score-*

based generative models, arXiv preprint arXiv:2111.08005, (2021).

[88] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, arXiv preprint arXiv:2011.13456, (2020).

[89] B. Sprungk, *On the local lipschitz stability of bayesian inverse problems*, Inverse Problems, 36 (2020), p. 055015.

[90] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, *Plug-and-play priors for bright field electron tomography and sparse interpolation*, IEEE Transactions on Computational Imaging, 2 (2016), pp. 408–423.

[91] G. Steidl, P. L. Hagemann, and J. Hertrich, *Generalized normalizing flows via markov chains*, Elements in Non-local Data Interactions: Foundations and Applications, (2022).

[92] O. Stramer and R. Tweedie, *Langevin-type models i: Diffusions with given stationary distributions and their discretizations*, Methodology and Computing in Applied Probability, 1 (1999), pp. 283–306.

[93] A. M. Stuart, *Inverse problems: a bayesian perspective*, Acta numerica, 19 (2010), pp. 451–559.

[94] H. Y. Tan, S. Mukherjee, J. Tang, and C.-B. Schönlieb, *Provably convergent plug-and-play quasi-newton methods*, arXiv preprint arXiv:2303.07271, (2023).

[95] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*, SIAM, 2005.

[96] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, *Plug-and-play priors for model based reconstruction*, in 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 945–948.

[97] A. Verine, B. Negrevergne, F. Rossi, and Y. Chevaleyre, *On the expressivity of bi-lipschitz normalizing flows*, arXiv preprint arXiv:2107.07232, (2021).

[98] C. Villani et al., *Optimal transport: old and new*, vol. 338, Springer, 2009.

[99] J. Whang, Q. Lei, and A. Dimakis, *Solving inverse problems with a flow-based noise model*, in International Conference on Machine Learning, PMLR, 2021, pp. 11146–11157.

[100] H. Wu, J. Köhler, and F. Noé, *Stochastic normalizing flows*, Advances in Neural Information Processing Systems, 33 (2020), pp. 5933–5944.

[101] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, Y. Shao, W. Zhang, B. Cui, and M.-H. Yang, *Diffusion models: A comprehensive survey of methods and applications*, arXiv preprint arXiv:2209.00796, (2022).

[102] G. Yu, G. Sapiro, and S. Mallat, *Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity*, IEEE Transactions on Image Processing, 21 (2011), pp. 2481–2499.

[103] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, *Plug-and-play image restoration with deep denoiser prior*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44 (2021), pp. 6360–6376.

[104] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising*, IEEE transactions on image processing, 26 (2017), pp. 3142–3155.

[105] D. Zoran and Y. Weiss, *From learning models of natural image patches to whole image restoration*, in 2011 international conference on computer vision, IEEE, 2011, pp. 479–486.

Appendix A. Proofs.

A.1. Proof of Lemma 3.1.

Proof. For a constant $R_0 > 0$, let

$$B(0, R_0) := \left\{ z \in \mathbb{R}^d : \|z\|_2 \leq R_0 \right\}$$

be the closed ball of radius R_0 centered at the origin. Since $C \subset \mathbb{R}^d$ is compact, there exists $R_0 > 0$ such that $C \subset B(0, R_0)$. Therefore, for all $x \notin B(0, R_0)$, it follows that

$$\|x - \Pi_C(x)\|_2 \overset{(a)}{\geqslant} \left\|x - \Pi_{B(0,R_0)}(x)\right\|_2 \overset{(b)}{\geqslant} \|x\|_2 - R_0 \geq 0,$$

where (a) is true since $C \subset B(0, R_0)$ and (b) follows from the triangle inequality. Then, for all $k \in \mathbb{N}$, the following holds:

$$
\begin{aligned}
&\int_{\mathbb{R}^d \setminus B(0,R_0)} \|x\|^k \exp\left(-\frac{\|x - \Pi_C(x)\|_2^2}{2\lambda}\right) \mathrm{d}x \\
&\leqslant \int_{\mathbb{R}^d \setminus B(0,R_0)} \|x\|^k \exp\left(-\frac{(\|x\|_2 - R_0)^2}{2\lambda}\right) \mathrm{d}x \\
&\leqslant \int_{\mathbb{R}^d \setminus B(0,R_0)} \|x\|^k \exp\left(-\frac{\|x\|_2^2 - 2R_0^2}{4\lambda}\right) \mathrm{d}x \\
&< +\infty,
\end{aligned}
$$

where the last inequality follows from the fact that k-order moments of Gaussian distribution are finite for any k. ■

A.2. Proof of Proposition 3.3.

Proof. Without loss of generality, we only need to consider the cases when the total number of layers is $k = 1, 2$.

(1) We firstly consider the case that $k = 1$ and $T^{-1} = G$ is a composition of only a one-layer coupling network. Then (3.4) can be simplified as:

$$
G_j(x_j, x_{<j}) = \varphi_j(x_{<j}) x_j + \eta_j(x_{<j}), \ j = 1, \cdots, d. \tag{A.1}
$$

Since $\forall r < j$, G_r is independent of x_j and the diagonal of the Jacobian is $(J_G(x))_{j,j} = \varphi_j(x_{<j})$, from the change of variables

$$
\begin{aligned}
q(x) &= q_z(z) \left|\det J_T(z)\right|^{-1} \\
&= q_z\left(T^{-1}(x)\right) \left|\det J_{T^{-1}}(x)\right|,
\end{aligned} \tag{A.2}
$$

we have that

$$
\begin{aligned}
\log q_\theta(x) &= \log q_z\left(G(x)\right) + \log\left|\det J_G(x)\right| \\
&= -\frac{1}{2}\|G(x)\|_2^2 + \log\left|\det J_G(x)\right| + \text{const.} \\
&= -\frac{1}{2}\|G(x)\|_2^2 + \sum_{j=1}^{d} \log\left|\varphi_j(x_{<j})\right| + \text{const.} \\
&\leqslant \sum_{j=1}^{d} \log\left|\varphi_j(x_{<j})\right| + \text{const.}
\end{aligned}
$$

Since φ_j is a bounded function $\forall j$, it follows that $\log|\varphi_j(x_{<j})|$ is upper bounded for all j and $\log q_\theta(x)$ is upper bounded on $\mathbb{R}^d$.

(2) Secondly, assume that $k = 2$ and $T^{-1} = G \circ H(x)$, where $H : x \mapsto \omega$ and $G : \omega \mapsto z$. Similarly, we have that

$$\begin{aligned}\log q_\theta(x) &= \log q_z\left(G \circ H(x)\right) + \log\left|\det J_{G\circ H}(x)\right| \\ &= -\frac{1}{2}\left\|G \circ H(x)\right\|_2^2 + \log\left|\det J_G(\omega)\right| + \log\left|\det J_H(x)\right| + \text{const.} \\ &= -\frac{1}{2}\left\|G \circ H(x)\right\|_2^2 + \sum_{j=1}^{d}\left(\log\left|\varphi_j^{(2)}(\omega_{<j})\right| + \log\left|\varphi_j^{(1)}(x_{<j})\right|\right) + \text{const.} \\ &\leqslant \sum_{j=1}^{d}\left(\log\left|\varphi_j^{(2)}(\omega_{<j})\right| + \log\left|\varphi_j^{(1)}(x_{<j})\right|\right) + \text{const.}\end{aligned}$$

Since $\varphi_j^{(1)}$ and $\varphi_j^{(2)}$ are bounded functions $\forall j$, it follows that $\log\left|\varphi_j^{(2)}(\omega_{<j})\right| + \log\left|\varphi_j^{(1)}(x_{<j})\right|$ is upper bounded for all j and $\log q_\theta(x)$ is upper bounded on $\mathbb{R}^d$. ∎

A.3. Proof of Proposition 3.5.

Proof. By Assumption 3.2, we have that ∎

$$\int_{\mathbb{R}^d}\left(1 + \Phi_1(\tilde{x})\right)\exp\left[c_0\Phi_1(\tilde{x}) - \iota_C^{(\lambda)}(\tilde{x})\right] q_\theta^\alpha(\tilde{x})\mathrm{d}\tilde{x} < +\infty,$$

and we conclude the proof from Proposition 2.3 of [56].

A.4. Proof of Lemma 3.7.

Lemma A.1. *Let Assumption 3.6 be true. Then, $\nabla \log p_\lambda(x|y)$ is Lipschitz continuous if and only if $\nabla \log q_\theta(x)$ is Lipschitz continuous.*

Proof. Since Assumption 3.6 is satisfied, from Algorithm 2.1 and (2.13) we have that $\nabla \log p_\lambda(x|y)$ is Lipschitz continuous if and only if $\alpha\nabla \log q_\theta(x) + (\Pi_C(x) - x)/\lambda$ is Lipschitz continuous.

From Proposition 12.28 in [9], the operator $(\mathrm{Id} - \mathrm{Prox}_{\iota_C})$ is firmly non-expansive, i.e., for all $x, y \in \mathbb{R}^d$,

$$\begin{aligned}\left\|(\Pi_C(x) - x) - (\Pi_C(y) - y)\right\|_2^2 &\leqslant \langle(\Pi_C(x) - x) - (\Pi_C(y) - y), x - y\rangle \\ &\leqslant \left\|(\Pi_C(x) - x) - (\Pi_C(y) - y)\right\|_2 \left\|x - y\right\|_2.\end{aligned}$$

Therefore, $(\Pi_C(x) - x)/\lambda$ is $1/\lambda$-Lipschitz. Hence, for any $\alpha > 0$, $\nabla \log p_\lambda(x|y)$ is Lipschitz-continuous if and only if $\nabla \log q_\theta(x)$ is Lipschitz-continuous. ∎

A.5. Proof of Proposition 3.9.

Proof. Without loss of generality, we only need to consider the cases when the total number of layers is $k = 1, 2$.

(1) We firstly consider the case that $k = 1$ and $T^{-1} = G$ is a composition of only a one-layer coupling network. Then (3.5) can be simplified as:

$$G_j(x_j, x_{<j}) = \varphi_j(x_{<j})x_j + \eta_j(x_{<j}),\ j = 1, \cdots, d. \tag{A.3}$$

Since $\forall r < j$, G_r is independent of x_j and the diagonal of the Jacobian is $(J_G(x))_{j,j} = \varphi_j(x_{<j})$, from the change of variables

$$q(x) = q_z(z) \left|\det J_T(z)\right|^{-1} = q_z\left(T^{-1}(x)\right) \left|\det J_{T^{-1}}(x)\right|, \tag{A.4}$$

we have that

$$\begin{aligned}\log q_\theta(x) &= \log q_z\left(G(x)\right) + \log\left|\det J_G(x)\right| \\ &= -\frac{1}{2}\left\|G(x)\right\|_2^2 + \log\left|\det J_G(x)\right| + \text{const.} \\ &= -\frac{1}{2}\left\|G(x)\right\|_2^2 + \sum_{j=1}^{d} \log\left|\varphi_j(x_{<j})\right| + \text{const.}\end{aligned}$$

Taking the gradient of both sides w.r.t. x, we get

$$\nabla \log q_\theta(x) = -\left(J_G(x)\right)^T G(x) + \sum_{j=1}^{d} \nabla \log \varphi_j(x_{<j}). \tag{A.5}$$

Since φ_j is a constant function, we have that $\nabla \log \varphi_j = 0$. Furthermore as η_j is Lipschitz and $\forall r < j$, $\frac{\partial \eta_j}{\partial x_r}$ is piecewise constant on $\mathbb{R}$, $\frac{\partial \eta_j}{\partial x_r}$ is hence bounded. Meanwhile, $(J_G(x))_{j,r} = \frac{\partial \eta_j}{\partial x_r}$, therefore every element of $J_G(x)$ is a bounded piecewise constant function of x. Then both $G(x)$ and $(J_G(x))^\top G(x)$ are Lipschitz, therefore $\nabla \log q_\theta(x)$ is Lipschitz.

(2) Secondly, assume that $k = 2$ and $T^{-1} = G \circ H(x)$, where $H : x \mapsto \omega$ and $G : \omega \mapsto z$. Similarly, we have that

$$\begin{aligned}\log q_\theta(x) &= \log q_z\left(G \circ H(x)\right) + \log\left|\det J_{G\circ H}(x)\right| \\ &= -\frac{1}{2}\left\|G \circ H(x)\right\|_2^2 + \log\left|\det J_G(\omega)\right| + \log\left|\det J_H(x)\right| + \text{const.} \\ &= -\frac{1}{2}\left\|G \circ H(x)\right\|_2^2 + \sum_{j=1}^{d}\left(\log\left|\varphi_j^{(2)}(\omega_{<j})\right| + \log\left|\varphi_j^{(1)}(x_{<j})\right|\right) + \text{const.}\end{aligned}$$

and

$$\begin{aligned}\nabla \log q_\theta(x) &= -\left(J_{G\circ H}(x)\right)^T G \circ H(x) + 0 \\ &= -\left(J_G(H(x)) J_H(x)\right)^T G \circ H(x).\end{aligned} \tag{A.6}$$

Since every element of $J_H(x)$ is a bounded piecewise constant function of x, every element of $J_G(w)$ is a bounded piecewise constant function of w, and meanwhile $w = H(x)$ is continuous w.r.t. x, then every element of $J_{G\circ H}(x)$ is a bounded piecewise constant function of x. Then both $G \circ H(x)$ and $(J_{G\circ H}(x))^\top G \circ H(x)$ are Lipschitz, therefore $\nabla \log q_\theta(x)$ is Lipschitz. ■

A.6. Proof of theorem 3.11.

Proof. Denote $R_{\mathrm{C}} = \sup\{\|x_1 - x_2\| : x_1, x_2 \in \mathrm{C}\}$. Since we have $2\lambda(\alpha\mathrm{L} - \mathrm{m}_y) \leqslant 1$, from A 3.8, A 3.10, $b_\lambda(x)$ in (3.6) and the Cauchy-Schwarz inequality we have that for any $x_1, x_2 \in \mathbb{R}^d$,

$$\begin{aligned}\langle b_\lambda(x_1) - b_\lambda(x_2), x_1 - x_2\rangle &\leqslant (-\mathrm{m}_y + \alpha\mathrm{L})\|x_1 - x_2\|^2 - \frac{\|x_1 - x_2\|^2}{\lambda} + \frac{R_{\mathrm{C}}\|x_1 - x_2\|}{\lambda} \\ &\leqslant -\frac{\|x_1 - x_2\|^2}{2\lambda} + \frac{R_{\mathrm{C}}\|x_1 - x_2\|}{\lambda}.\end{aligned} \tag{A.7}$$

For any $x_1, x_2 \in \mathbb{R}^d$ satisfying $\|x_1 - x_2\| \geqslant 4R_{\mathrm{C}}$, we obtain the contractivity at infinity condition on the drift b_λ

$$\langle b_\lambda(x_1) - b_\lambda(x_2), x_1 - x_2\rangle \leqslant -\frac{\|x_1 - x_2\|^2}{4\lambda}, \tag{A.8}$$

which indicates the strongly convexity at infinity.

After simple computation by letting $x_2 = 0$ in (A.7), we also have that for any $x \in \mathbb{R}^d$,

$$\langle b_\lambda(x), x\rangle \leqslant -\|x\|^2/(4\lambda) + \sup_{\tilde{x}\in\mathbb{R}^d}\left\{(R_{\mathrm{C}}/\lambda + \|b_\lambda(0)\|)\|\tilde{x}\| - \|\tilde{x}\|^2/(4\lambda)\right\}. \tag{A.9}$$

From A 3.6, A 3.8, $b_\lambda(x)$ in (3.6) and that $(\mathrm{Id} - \Pi_C)/\lambda$ is $1/\lambda$-Lipschitz, we have that for any $x_1, x_2 \in \mathbb{R}^d$,

$$\|b_\lambda(x_1) - b_\lambda(x_2)\|_2 \leqslant (\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda)\|x_1 - x_2\|_2. \tag{A.10}$$

Let $\bar{\gamma} = (4\lambda)^{-1}(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda)^{-2}$. From (A.9) and (A.10), using Lemma SM5.1 in [57] and we get that there exist $\lambda_V \in (0,1]$, $c \geqslant 0$ such that for any $\delta \in (0, \bar{\gamma}]$, R_δ satisfies the discrete drift condition $\mathbf{D}_{\mathrm{d}}\left(V, \lambda_V^\delta, c\delta\right)$.

For any probability measure ν_1, ν_2, from the definition (3.1) and Hölder's inequality we have that

$$\|\nu_1 - \nu_2\|_V \leqslant \|\nu_1 - \nu_2\|_{\mathrm{TV}}^{1/2}\left(\nu_1\left[V^2\right] + \nu_2\left[V^2\right]\right)^{1/2}. \tag{A.11}$$

Since $\bar{\delta} \leqslant \bar{\gamma}$, the contractivity condition (A.8) holds, (A.11) holds, then from Theorem 8 and Corollary 2 in [22], we can find $A_2 \geqslant 0$ and $\rho_2 \in [0,1)$ such that for any $\delta \in (0, \bar{\delta}], x_1, x_2 \in \mathbb{R}^d$, and $k \in \mathbb{N}$,

$$\begin{aligned}\left\|\boldsymbol{\delta}_{x_1}\mathrm{R}_\delta^k - \boldsymbol{\delta}_{x_2}\mathrm{R}_\delta^k\right\|_{\mathrm{TV}} &\leqslant A_2\rho_2^{k\delta}\left(V(x_1) + V(x_2)\right) \\ &\leqslant A_2\rho_2^{k\delta}\left(V^2(x_1) + V^2(x_2)\right), \\ \mathbf{W}_1\left(\boldsymbol{\delta}_{x_1}\mathrm{R}_\delta^k, \boldsymbol{\delta}_{x_2}\mathrm{R}_\delta^k\right) &\leqslant A_2\rho_2^{k\delta}\|x_1 - x_2\|_2.\end{aligned} \tag{A.12}$$

Then we conclude the proof from (A.11). ■

A.7. Proof of theorem 3.12.

Proof. Most of our proof is based on [57] and [22].

Recall that

$$\mathrm{R}_\delta(x,\ \mathrm{A}) = (2\pi)^{-d/2} \int_{\mathbb{R}^d} \mathbf{1}_{\mathrm{A}} \left(x + \delta b_\lambda(x) + \sqrt{2\delta} z \right) \exp\left[-\|z\|^2/2 \right] \mathrm{d}z. \tag{A.13}$$

We introduce the stochastic process $\left(\overline{\mathbf{X}}_t\right)_{t\geqslant 0}$, which is exactly the solution of the following SDE:

$$\begin{cases} \mathrm{d}\overline{\mathbf{X}}_t = b_\lambda\left(\overline{\mathbf{X}}_t\right) \mathrm{d}t + \sqrt{2}\, \mathrm{d}\mathbf{B}_t \\ b_\lambda(x) = \nabla \log(p(y|x)) + \alpha \nabla \log q_\theta(x) + \dfrac{\Pi_C(x) - x}{\lambda} \\ \overline{\mathbf{X}}_0 = X_0, \end{cases} \tag{A.14}$$

where $(\mathbf{B}_t)_{t\geqslant 0}$ is a d-dimensional Brownian motion.

From Lemma 3.7, b_λ is $(\mathrm{L}_y+\alpha\mathrm{L}+1/\lambda)$-Lipschitz continuous. From Chapter 5, Theorem 2.9 of [48] we have that the SDE (A.14) admits a unique strong solution for any initial condition $\overline{\mathbf{X}}_0$ with $\mathbb{E}\left[\left\|\overline{\mathbf{X}}_0\right\|^2\right] < +\infty$. We denote by $(\mathrm{P}_t)_{t\geqslant 0}$ the semigroup associated with the strong solutions of SDE (A.14). Similarly to the proof of Theorem 3.11, replacing Corollary 2 in [22] by Theorem 21 and Corollary 22 in [22], there exist $\tilde{A}_1 \geqslant 0$ and $\tilde{\rho}_1 \in [0, 1)$ such that that for any $x_1, x_2 \in \mathbb{R}^d$ and $t \geqslant 0$,

$$\begin{aligned} \|\boldsymbol{\delta}_{x_1}\mathrm{P}_t - \boldsymbol{\delta}_{x_2}\mathrm{P}_t\|_V &\leqslant \tilde{A}_1 \tilde{\rho}_1^t \left(V^2(x_1) + V^2(x_2) \right), \\ \mathbf{W}_1\left(\boldsymbol{\delta}_{x_1}\mathrm{P}_t, \boldsymbol{\delta}_{x_2}\mathrm{P}_t\right) &\leqslant \tilde{A}_1 \tilde{\rho}_1^t \|x_1 - x_2\|_2. \end{aligned} \tag{A.15}$$

Combining (A.15), Theorem 3.11, the fact that $\left(\mathscr{P}_1\left(\mathbb{R}^d\right), \mathbf{W}_1\right)$ is a complete metric space and the Picard fixed point theorem, we can obtain that for any $\delta \in (0, \bar{\delta}]$ there exist unique $\pi_{\delta,\lambda}, \tilde{\pi}_\lambda \in \mathscr{P}_1\left(\mathbb{R}^d\right)$ such that $\pi_{\delta,\lambda}\mathrm{R}_\delta = \pi_{\delta,\lambda}$ and for any $t \geqslant 0, \tilde{\pi}_\lambda \mathrm{P}_t = \tilde{\pi}_\lambda$. By Theorem 2.1 in [80] we have that for any $x \in \mathbb{R}^d$,

$$\left(\mathrm{d}\tilde{\pi}_\lambda/\mathrm{dLeb}\right)(x) \propto \exp\left[-\iota_C^{(\lambda)}(x) \right] p(y|x) p_\lambda^\alpha(x), \tag{A.16}$$

Therefore from (2.12) π_λ and $\tilde{\pi}_\lambda$ are exactly the same.

Similar to (3.7), from (A.15) we have that for any $t \geqslant 0$ and $x \in \mathbb{R}^d$,

$$\|\boldsymbol{\delta}_x \mathrm{P}_t - \pi_\lambda\|_V \leqslant \tilde{A}_1 \tilde{\rho}_1^t \left(V^2(x) + \int_{\mathbb{R}^d} V^2(\tilde{x}) \mathrm{d}\pi_\lambda(\tilde{x}) \right). \tag{A.17}$$

Since we already proved that $\int_{\mathbb{R}^d} V^2(\tilde{x})\mathrm{d}\pi_\lambda(\tilde{x}) < +\infty$ in Lemma 3.1, we can find $B_1 \geqslant 0$ such that for any $x \in \mathbb{R}^d$ we have

$$\|\boldsymbol{\delta}_x \mathrm{P}_t - \pi_\lambda\|_V \leqslant B_1 \tilde{\rho}_1^t V^2(x). \tag{A.18}$$

Select a large $m_1 \in \mathbb{N}^*$ such that $m_1 \geqslant \bar{\delta}^{-1}$. Let's now consider the interval $[0, l]$, $l \in \mathbb{N}^*$. To compare $\pi_{\delta,\lambda}$ with π_δ, we first construct a continuous time Markov process $X_t^{(1)}$ such that $X^{(1)}_{j/m_1}$ has the same distribution as the j-th sample X_j by NF-ULA (2.13). Define $b_1\left(t,(w_t)_{t\in[0,l]}\right) = \sum_{j=0}^{m_1 l-1} \mathbf{1}_{[j/m_1,(j+1)/m_1)}(t) b_\lambda\left(w_{j/m_1}\right)$ and $b_2\left(t,(w_t)_{t\in[0,l]}\right) = b_\lambda(w_t)$. Let $\mathbf{X}^{(1)}_t$ and $\mathbf{X}^{(2)}_t$ be the unique strong solution of SDE $\mathrm{d}\mathbf{X}_t = b\left(t,(\mathbf{X}_t)_{t\in[0,l]}\right)\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t$ with $\mathbf{X}_0 = x \in \mathbb{R}^d$ and $b = b_1$, respectively $b = b_2$. Note that $\left(\mathbf{X}^{(1)}_{k/m_1}\right) = (X_k)_{k\in\mathbb{N}}$ and $\left(\mathbf{X}^{(2)}_t\right)_{t\geqslant 0} = \left(\overline{\mathbf{X}}_t\right)_{t\geqslant 0}$. Denote $P^{(1)}_t$ and $P^{(2)}_t$ the Markov semigroup associated with $\mathbf{X}^{(1)}_t$ and $\mathbf{X}^{(2)}_t$. Then for any $x \in \mathbb{R}^d$, $k \in \mathbb{N}^*$ we have

$$\boldsymbol{\delta}_x \mathrm{R}^{km_1}_{1/m_1} = \boldsymbol{\delta}_x \mathrm{P}^{(1)}_k, \quad \boldsymbol{\delta}_x \mathrm{P}_k = \boldsymbol{\delta}_x \mathrm{P}^{(2)}_k. \tag{A.19}$$

From Lemma 3.7 and A 3.8, for any $t \in [j/m_1,(j+1)/m_1), j \in \{0,\dots,m_1 l-1\}$ and $(w_t)_{t\in[0,l]} \in \mathrm{C}\left([0,l],\mathbb{R}^d\right)$ we have that

$$\begin{aligned}\left\|b_1\left(t,(w_t)_{t\in[0,l]}\right) - b_2\left(t,(w_t)_{t\in[0,l]}\right)\right\|^2 &= \left\|b_\lambda\left(w_{j/m_1}\right) - b_\lambda(w_t)\right\|^2 \\ &\leqslant (\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda)^2 \left\|w_{j/m_1} - w_t\right\|^2.\end{aligned} \tag{A.20}$$

Using Cauchy-Schwarz inequality, Hölder's inequality and Itô's isometry we have for any $t \in [j/m_1,(j+1)/m_1)$,

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbf{X}^{(2)}_t - \mathbf{X}^{(2)}_{j/m_1}\right\|^2\right] &= \mathbb{E}\left[\left\|\int_{j/m_1}^t \left(b_\lambda\left(\mathbf{X}^{(2)}_\tau\right)\mathrm{d}\tau + \sqrt{2}\,\mathrm{d}\mathbf{B}_\tau\right)\right\|^2\right] \\ &\leqslant \mathbb{E}\left[2\left\|\int_{j/m_1}^t b_\lambda\left(\mathbf{X}^{(2)}_\tau\right)\mathrm{d}\tau\right\|^2 + 2\left\|\sqrt{2}\left(\mathbf{B}_t - \mathbf{B}_{j/m_1}\right)\right\|^2\right] \\ &\leqslant 2\left(t - \frac{j}{m_1}\right)\mathbb{E}\left[\int_{j/m_1}^t \left\|b_\lambda\left(\mathbf{X}^{(2)}_\tau\right)\right\|^2 \mathrm{d}\tau\right] + 4d\left(t - \frac{j}{m_1}\right) \\ &\leqslant 2\left(t - \frac{j}{m_1}\right)^2 \sup_{\tau\leqslant(j+1)/m_1} \mathbb{E}\left\|b_\lambda\left(\overline{\mathbf{X}}_\tau\right)\right\|^2 + 4d\left(t - \frac{j}{m_1}\right).\end{aligned} \tag{A.21}$$

Since we have proved (A.8), (A.9), (A.10) in Appendix A.6, from Lemma 2.11 and Lemma 2.12 in [65], for any $\tau > 0$ we have

$$\mathbb{E}\left\|\overline{\mathbf{X}}_\tau\right\|^2 \leqslant B_{0,0}, \tag{A.22}$$

where $B_{0,0}$ is an upper bound formed by $\lambda, C, b_\lambda(0), d, x$. Then from (A.10) we have that

$$\mathbb{E}\left\|b_\lambda\left(\overline{\mathbf{X}}_\tau\right)\right\|^2 \leqslant 2\left(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda\right)^2 \mathbb{E}\left\|\overline{\mathbf{X}}_\tau\right\|^2 + 2\left\|b_\lambda(0)\right\|^2 \leqslant B_3, \quad \forall\tau > 0, \tag{A.23}$$

where $B_3 = 2\left(\mathrm{L}_y + \alpha\mathrm{L} + 1/\lambda\right)^2 B_{0,0} + 2\left\|b_\lambda(0)\right\|^2 \geqslant 0$.

Then from (A.20), (A.21), (A.23), for $i \in \{0, \cdots l-1\}$ we have that

$$
\begin{aligned}
&\int_i^{i+1} \mathbb{E}\left[\left\|b_1\left(t, \mathbf{X}_t^{(2)}\right) - b_2\left(t, \mathbf{X}_t^{(2)}\right)\right\|^2\right] \mathrm{d}t \\
&\leqslant \sum_{j=im_1}^{(i+1)m_1-1} \int_{j/m_1}^{(j+1)/m_1} \mathbb{E}\left[\left\|b_1\left(t, \mathbf{X}_t^{(2)}\right) - b_2\left(t, \mathbf{X}_t^{(2)}\right)\right\|^2\right] \mathrm{d}t \\
&\leqslant (\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda)^2 \sum_{j=im_1}^{(i+1)m_1-1} \int_{j/m_1}^{(j+1)/m_1} \mathbb{E}\left[\left\|\mathbf{X}_t^{(2)} - \mathbf{X}_{j/m_1}^{(2)}\right\|^2\right] \mathrm{d}t \\
&\leqslant (\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda)^2 \left(\frac{2B_3}{3m_1^2} + \frac{2d}{m_1}\right).
\end{aligned} \tag{A.24}
$$

From (A.19) and Lemma SM6.1 in [57], we obtain that there exists $B_b \geqslant 0$ such that for any $x \in \mathbb{R}^d$,

(A.25)

$$
\begin{aligned}
&\left\|\boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{lm_1} - \boldsymbol{\delta}_x \mathrm{P}_l\right\|_V = \left\|\boldsymbol{\delta}_x \mathrm{P}_l^{(1)} - \boldsymbol{\delta}_x \mathrm{P}_l^{(2)}\right\|_V = \left\|\boldsymbol{\delta}_x \mathrm{P}_l^{(2)} - \boldsymbol{\delta}_x \mathrm{P}_l^{(1)}\right\|_V \\
&\leqslant \left(\boldsymbol{\delta}_x \mathrm{P}_l^{(1)}\left[V^2\right] + \boldsymbol{\delta}_x \mathrm{P}_l^{(2)}\left[V^2\right]\right)^{1/2} \times \left(\sum_{i=0}^{l-1} \int_i^{i+1} \mathbb{E}\left[\left\|b_1\left(t, \mathbf{X}_t^{(2)}\right) - b_2\left(t, \mathbf{X}_t^{(2)}\right)\right\|^2\right] \mathrm{d}t\right)^{1/2} \\
&\leqslant (\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda) \sqrt{l\left(\frac{2B_3}{3m_1^2} + \frac{2d}{m_1}\right)} \left(\boldsymbol{\delta}_x \mathrm{P}_t^{(1)}\left[V^2\right] + \boldsymbol{\delta}_x \mathrm{P}_t^{(2)}\left[V^2\right]\right)^{1/2}.
\end{aligned}
$$

Assume that there is a function $W \in \mathrm{C}^2\left(\mathbb{R}^d, [1, +\infty)\right)$ such that $\lim_{\|x\| \to +\infty} W(x) = +\infty$. Recall that from (A.9), using Lemma SM5.1 in [57] and we get that there exist $\lambda_W \in (0, 1]$, $c, \beta \geqslant 0$ and $\zeta > 0$ such that for any $\delta \in \left(0, (4\lambda)^{-1}\left(\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda\right)^{-2}\right]$, R_δ satisfies the discrete drift condition $\mathbf{D}_{\mathrm{d}}\left(W, \lambda_W^\delta, c\delta\right)$ and $(\mathrm{P}_t)_{t \geqslant 0}$ satisfies the continuous drift condition $\mathbf{D}_{\mathrm{c}}(W, \zeta, \beta)$. From Lemma SM5.2 in [57], there exists $B_c \geqslant 0$ such that for any $x \in \mathbb{R}^d, t \geqslant 0$ and $k \in \mathbb{N}^*$ we have

$$
\mathrm{R}_\delta^k W(x) + \mathrm{P}_t W(x) \leqslant B_c^2 W(x). \tag{A.26}
$$

Let $W(x) = V^2(x)$ and $k = m_1 l$, $\delta = 1/m_1$, $t = l$, then $\forall x \in \mathbb{R}^d$,

$$
\boldsymbol{\delta}_x \mathrm{P}_l^{(1)}\left[V^2\right] + \boldsymbol{\delta}_x \mathrm{P}_l^{(2)}\left[V^2\right] \leqslant B_c^2 V^2(x). \tag{A.27}
$$

Combined with (A.25), we have that

$$
\left\|\boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{m_1 l} - \boldsymbol{\delta}_x \mathrm{P}_l\right\|_V \leqslant B_c V(x)(\mathrm{L}_y + \alpha \mathrm{L} + 1/\lambda) \sqrt{l\left(\frac{2B_3}{3m_1^2} + \frac{2d}{m_1}\right)}. \tag{A.28}
$$

To give a bound on $\left\|\boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{m_1 l} - \pi_\lambda\right\|_V$, we use triangular inequality to split it into two terms:

$$
\left\|\boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{m_1 l} - \pi_\lambda\right\|_V \leqslant \left\|\boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{m_1 l} - \boldsymbol{\delta}_x \mathrm{P}_l\right\|_V + \left\|\boldsymbol{\delta}_x \mathrm{P}_l - \pi_\lambda\right\|_V. \tag{A.29}
$$

Using this result and (A.18), we obtain that there exists $B_1, B_2 \geqslant 0$ such that for any $m_1 \in \mathbb{N}^*$ with $1/m_1 \leqslant \bar{\delta}$,

$$\left\| \boldsymbol{\delta}_x \mathrm{R}_{1/m_1}^{m_1 l} - \pi_\lambda \right\|_V \leqslant B_1 \tilde{\rho}_1^l V^2(x) + B_2 V(x) \sqrt{l \left(\frac{B_3}{3m_1^2} + \frac{d}{m_1} \right)}. \tag{A.30}$$

The proof in the general case where $\delta \in (0, \bar{\delta}]$ is similar when the interval $[0, l]$ is changed to $[0, l m_1 \delta]$.

Then we obtain that there exists $B_1, B_2, B_3 \geqslant 0$, $\tilde{\rho}_1 \in [0, 1)$ such that for any $\delta \in (0, \bar{\delta}]$, $k \in \mathbb{N}^*$,

$$\left\| \boldsymbol{\delta}_x \mathrm{R}_\delta^k - \pi_\lambda \right\|_V \leqslant B_1 \tilde{\rho}_1^{k\delta} V^2(x) + B_2 V(x) \sqrt{\delta^2 k \left(d + \frac{B_3 \delta}{3} \right)}. \tag{A.31}$$

■

Appendix B. Additional experiments.

The second limited-angle computed tomography reconstruction experiment we test is using the Poisson noise, where the model can be formulated as $y \sim P(Ax)$ and $P(Ax)$ denotes adding a Poisson noise on Ax. We simulate the noisy sinogram as

$$y = -\frac{1}{\mu} \log \left(\frac{N_1}{N_0} \right), \quad N_1 \sim \text{Poisson} \left(N_0 \exp(-A(x)\mu) \right).$$

Here $N_0 = 4096$ is the mean photon count per detector bin without attenuation. $\mu = 0.05$ is a constant. Since Poisson noise implies a different likelihood

$$\begin{aligned} p(y|x) &= \frac{1}{K_0} \exp(-J(x, y)), \\ J(x, y) &= \sum_{i=1}^{m} e^{-A(x)_i \mu} N_0 + e^{-y_i \mu} N_0 \left(A(x)_i \mu - \log \left(N_0 \right) \right), \end{aligned}$$

we calculate $\nabla \log p(y|x) = -\nabla J(x, y)$ by using the auto-gradient library.

We select a different $\alpha = 4000$ for NF-ULA while keeping all the other settings the same as in the main paper.

Both PnP-ULA and NF-ULA have burn-in iterations of more than 20000. After the burn-in time, we calculate the posterior mean and the standard deviation by obtaining 10000 samples and computing the PSNR of the samples' mean. For Poisson noise, the likelihood is more complicated than Gaussian, and NF-ULA spends 510s.

Fig 8 includes the original image, the FBP, the posterior mean and the standard deviation of PnP-ULA (realSN-DnCNN) and NF-ULA (patchNR). Table 5 provides the PSNR of the posterior mean. All the samples generated in Table 5 never escape $[-0.2, 1.2]^d$, indicating that the projection $\Pi_C(x)$ is never activated. Note that the huge uncertainties of standard deviation on the left area in the Gaussian-noise case in the main paper are slightly alleviated in the Poison noise experiments. The ACF test results are similar to the CT experiment with Gaussian noise, therefore here we do not repeat them again.

Figure 8. *Limited-view CT reconstruction with Poisson noise. Column 1: Original image. Column 2: Filtered back projection (FBP). Columns 3, and 4: Posterior mean and the standard deviation of the samples generated by PnP-ULA (realSN-DnCNN). Columns 5, and 6: Posterior mean and the standard deviation of the samples generated by NF-ULA (patchNR). PSNR values of the sample mean images are provided in Table 5.*

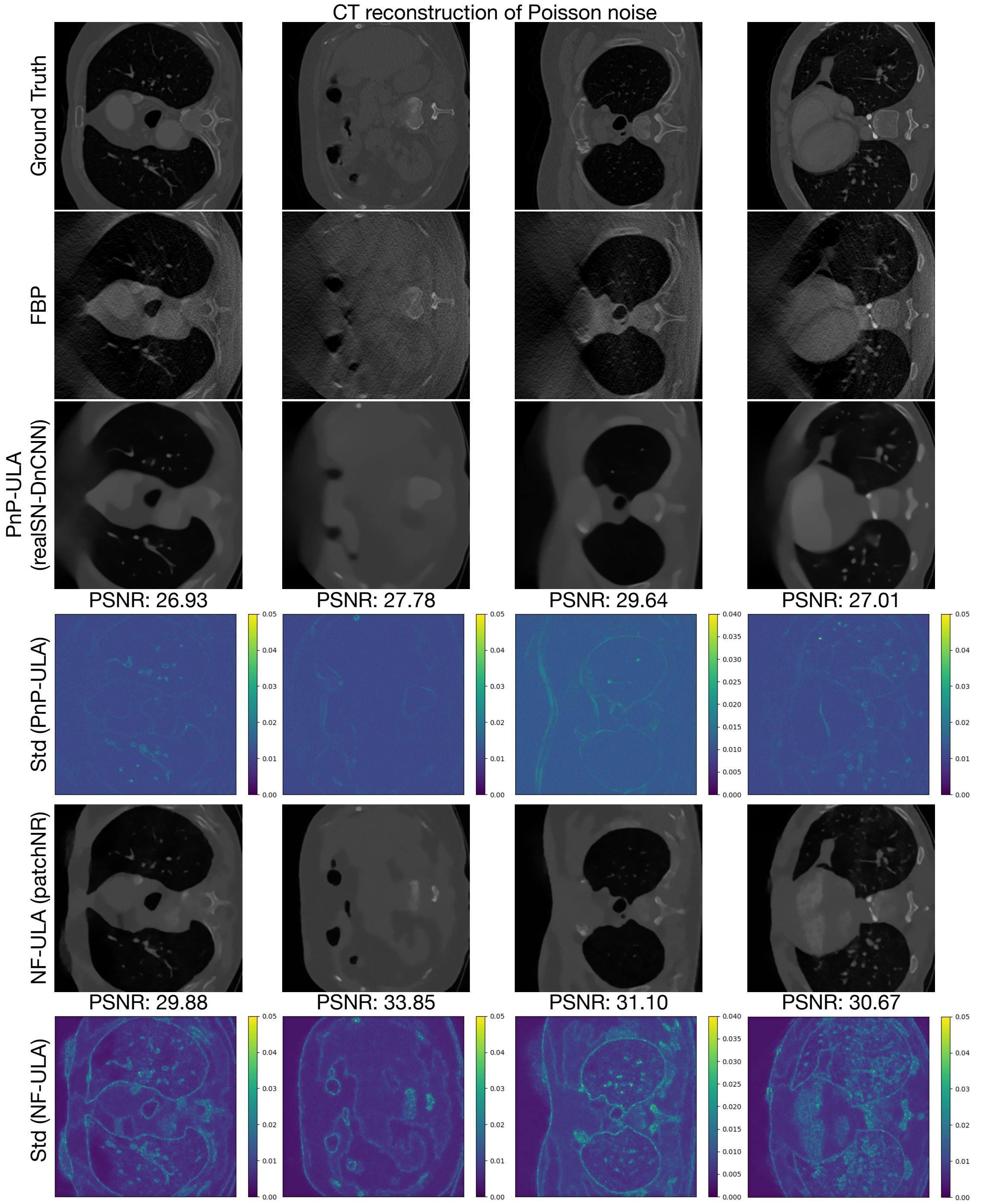


Table 5
CT reconstruction of Poisson noise, limited angles.

CT	$C = [-100, 100]^d$		
	network	parameters	PSNR
figure1			
NF-ULA	PatchNR	$\alpha = 4000$	29.88
PnP-ULA	realSN-DnCNN	$\alpha = 3$	26.93
figure2			
NF-ULA	PatchNR	$\alpha = 4000$	33.85
PnP-ULA	realSN-DnCNN	$\alpha = 3$	27.78
figure3			
NF-ULA	PatchNR	$\alpha = 4000$	31.10
PnP-ULA	realSN-DnCNN	$\alpha = 3$	29.64
figure4			
NF-ULA	PatchNR	$\alpha = 4000$	30.67
PnP-ULA	realSN-DnCNN	$\alpha = 3$	27.01