UNIVERSITY OF MICHIGAN

# DiffusionBlend

## Learning 3D Image Prior through Position-aware Diffusion Score Blending for 3D Computed Tomography Reconstruction

Bowen Song*, Jason Hu*, Zhaoxu Luo, Jeff Fessler, Liyue Shen

https://eecs.umich.edu/

November 22, 2024

# Deep Learning for Inverse Problems



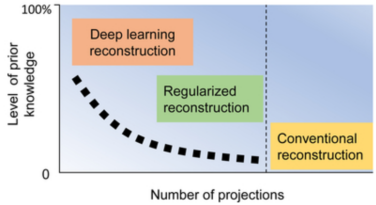



Figure: Learn the data-driven prior by deep learning when the measurement is sparse [1]

- Inverse problems arise from a wide range of applications across many domains, including computational imaging, remote sensing, and so on.
- The goal is to reconstruct an unknown signal $x_{true}$ given the observed measurements $y$ of the form $y = A(x_{true}) + \epsilon$, where $\epsilon$ can be an additive noise.
- Deep learning models that learn the data prior (distribution of $p(x_{clean,y})$) help reconstruct the clean images from very **sparse** measurements.

[1] Shen, Liyue et al., Patient-specific reconstruction of volumetric computed tomography images from a single projection view via deep learning, Nature biomedical enfgineering

# Deep Learning for Inverse Problems



- **Supervised** Approaches (assuming that the $x_{clean}$, $y$ pair is available during training, train a network that maps $y$ to $x_{clean}$)
  - Need to retrain for a different inverse problem
  - Generalization capabilities may be limited in the presence of noise/modality shift
  - Need paired data for training
- **Unsupervised** Approaches (assuming that only $x_{clean}$ is available during training)
  - Easily adapt to a new inverse problem in a zero-shot manner.
  - Do not need paired data for training.

Both approaches are widely reported in the literature [8].

UNIVERSITY OF MICHIGAN

# Diffusion (Score-based) Models



Denoising diffusion models consist of two processes

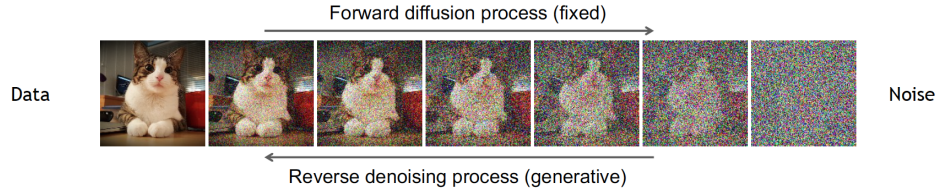



Figure: An illustration of the diffusion pipeline [9]

- A forward process in which gradually add noise to $x_{clean}$
- A reverse denoising process that remove noise from $x_t$ to recover $x_{clean}$

Specifically, the reverse process is governed by the score function $\nabla \log p(x_t)$. Training a neural network that approximates $\nabla \log p(x_t)$ would enable data generation capability.

# Mathematical Formulation of Diffusion Process



- As $x_t \approx x_{t-1} - \frac{\beta_t \Delta t}{2} x_{t-1} + \sqrt{\beta_t} \Delta t \omega$ where $\omega \in N(0, 1)$
- As $\Delta t \to 0$, then $dx_t = -\frac{1}{2}\beta_t x_t dt + \sqrt{\beta_t} d\omega_t$

**Forward Diffusion SDE:**

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t \, \mathrm{d}t + \sqrt{\beta(t)} \, \mathrm{d}\boldsymbol{\omega}_t$$

**Reverse Generative Diffusion SDE:**

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}} \mathrm{d}t + \overbrace{\sqrt{\beta(t)} \, \mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

Figure: The mathematical formulation of diffusion process [1]

- The solution of the stochastic differential equation can be utilized by the score function
- we can use a neural network to approximate it, such as $s_\theta(x_t) \approx \nabla_{x_t} \log p(x_t)$

# Solving Inverse Problem with Diffusion Models



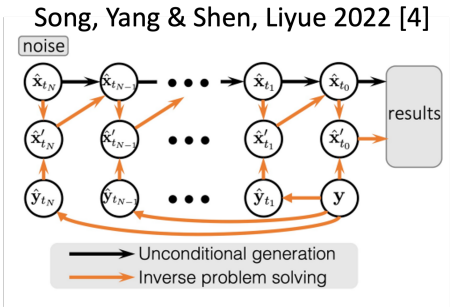



Figure: the flowchart of Score SDE [4]. Each $\hat{\mathbf{x}}_t$ is modified through optimization

To solve linear inverse problems with diffusion model priors, we can use

- **hard consistency**: modify $\mathbf{x}_t$ with optimization, such as with the objective $\text{argmin}_z \lambda||\hat{\mathbf{x}}_t - z||_2^2 + (1-\lambda)||\hat{\mathbf{y}}_t - Az||_2^2$ [4, 3]
- **soft consistency**: change $\nabla_{x_t} \log p(x_t)$ to $\nabla_{x_t} \log p(x_t|y)$ via Bayesian rule [1, 5]. We have $\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$, with $\nabla_{x_t} \log p(y|x_t)$ can be approximated through $\nabla_{x_t} \log p(y|\hat{x}_0(x_t))$

# Background: CT Reconstruction



By taking the log of the photon flux, we are able to encode the material physical properties.

**Beer's law:** Each material has a characteristic linear attenuation coefficient $\mu$ for rays emitted at a given frequency. More precisely,

$$\frac{dI}{ds} = -\mu(x(s))I(s),$$

where $s$ is the arc length along the trajectory described by the ray.

- The forward process is given by a linear transformation called "Radon Transform", which has the formula $Rf(\rho, \theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \delta(x \cos\theta + y \sin\theta - \rho)\, dx\, dy$.
- Denote the matrix of Radon Transform be $A$, the measurement (projections) is given by $Ax + n$, where n is usually a Poisson-Gaussian noise, and in a high-dose setting, we can just ignore this term.

# Background: CT Reconstruction



To reconstruct the original signal from the projections, we need to invert the Radon Transform process

- The backprojection formula is given by $f(x, y) = \int_0^{\pi} p_\theta(x \cos\theta + y \sin\theta)\, d\theta$. The vanilla backprojection without filtering gives blurry reconstruction.
- After adding a ramp filter to the backprojection, we are able to reconstruct high-quality images with full-view projections (like 180 projections). $f(x, y) = \int_0^{\pi} R^{-1}\left(F(\rho) * h(\rho)\right)(x \cos\theta + y \sin\theta)\, d\theta$
- Nevertheless, the filtered backprojection fails to reconstruct high-quality images when the forward operator is ill-posed (i.e. when projections are sparse).

# Learning cross slice scores



- Most existing diffusion-based methods for CT reconstruction use the scores of 2D slices [8]. There is no existing works on the usage of cross-slice 3D scores for CT reconstruction
- We observe that learning the distribution of the score of **a combination of 2D slices** can be effectively used for 3D CT reconstruction

UNIVERSITY OF MICHIGAN

# Learning cross slice scores



Specifically, let $\nabla \log p_x[i]$ be the score of the $i_{th}$ 2D CT slice. Existing works model $\nabla \log p_x \approx \sum_i \nabla \log p_x[i]$, and add external regularizations such as total variation to restrict the variability of inter-slice changes.

- We propose to learn the cross slice score $\nabla \log p_x[i_i, ..., i_k]$
- However, combining each cross slice score into the score of the entire volume is not a trivial task. i.e. how to write $\nabla \log p_x$ into a formula of $\nabla \log p_x[i_i, ..., i_k]$?

UNIVERSITY OF MICHIGAN

We propose a novel method do this by add conditional slice information to the original summation equation

- Instead of using $\nabla \log p_x \approx \sum_i \nabla \log p_x[i]$, we can rewrite this into $\nabla \log p_x \approx \sum_i \nabla \log p(x[i]|x[i-1], x[i+1])$ This equation is more accurate if we assume that $x[i]$ are not independent of each other (but dependent on neighboring slices).
- We then train a diffusion network, that takes a batch of 2D noisy slices as input and then denoise the central slice. We call this method **DiffusionBlend**

UNIVERSITY OF MICHIGAN

However, training a conditional diffusion model on noisy slices discard the joint slice distribution that we can learn.

- If we can learn $\nabla \log p(x[i], x[i-1], x[i+1])$, then we can compute $\nabla \log p(x[i]|x[i-1], x[i+1])$ through marginal probability.
- However, if we directly sum up $\nabla \log p(x[i], x[i-1], x[i+1])$ for each three blocks, we will lose the dependency information of each blocks. i.e. we learn no information about $p(x[i]|x[i+2])$
- We propose a novel method that **randomly partition** the 3D volumes into different joint slices blocks, and compute the score of the partition at each reverse sampling time step.

UNIVERSITY OF MICHIGAN

Why Random Partitioning work?

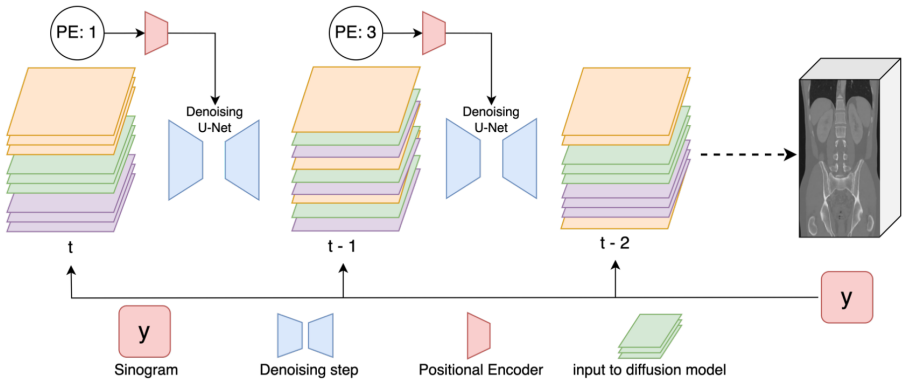
- Even though we may loss the cross-block conditional information at time t, with a different partition at t-1, the border of the original block incorporates the information of adjacency slices. So that it approximates the score of $\nabla \log p((x[i], x[i-1], x[i+1]) \cup (x[i+2], x[i+3]))$
- By doing random paritioning and score computing many times, we are able to approximate the score of the entire 3D volume.

UNIVERSITY OF MICHIGAN

# Overview of Our Method



We propose to randomly partition the volume at the reverse inference step and then compute the score respective of the partition while adding data consistency.



# DiffusionBlend++



One drop back of the naive random paritioning method is that it only learns the score of adjacency slices

- We can design a skip slice partitioning score that not only learns the score of adjacency slices, but the dependency of jumping slices (for example: dependency of slice 1, slice 4 and slice 7).
- The intuition of this approach is that the naive cross-slice score $\nabla \log p(x[i], x[i-1], x[i+1])$ with the partitioniong $(x[i], x[i-1], x[i+1]) \cup (x[i+2], x[i+3], x[i+4]...)$can have border artifacts.

UNIVERSITY OF MICHIGAN

# DiffusionBlend++



We propose a novel method to learn the score of the joint borders. For example, for a border of a partition $(x[i], x[i-1], x[i+1]) \cup (x[i+2], x[i+3], x[i+4]) \cup (x[i+5], x[i+6], x[i+7]))$, we can learn the joint border score $\nabla \log p(x[i+1], x[i+4], x[i+7])$, and then apply the previous algorithm to blend with this score during reverse sampling.

- We preserve more inter-slice information through this joint border score.
- We are able to remove artifacts on the border of the partition.

UNIVERSITY OF MICHIGAN

We also want to keep all different score computation done by one model. If we use different diffusion models, the same noise may not lead to the same output. To do this we propose to use a **relative encoding** method:

# DiffusionBlend++



- We propose to condition the diffusion model on the partition type, specifically, the distance between two slices in one block from the partition
- For instance, a partition of $(x[i], x[i-1], x[i+1]) \cup (x[i+2], x[i+3], x[i+4]) \cup (x[i+5], x[i+6], x[i+7])$ has the code to be 1, and a partition of $(x[i], x[i-3], x[i+3]) \cup (x[i+1], x[i-2], x[i+4]) \cup (x[i+2], x[i-1], x[i+5])$ has the code to be 3
- Fourier embedding is then computed for the code, and then feed into the diffusion model

UNIVERSITY OF MICHIGAN

## Algorithm PseudoCode:

**Algorithm 1** DiffusionBlend++

**Require:** $A$, $M$, $\zeta_i > 0$, $k$, $\boldsymbol{y}$
  Initialize $\boldsymbol{x}_T \sim \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$
  **for** $t = T : 1$ **do**
    Randomly select a partition $\mathcal{S} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r$ (the partition can either be of adjacency slices or jumping slices)
    Compute the relative positional encoding $PE_t$
    For each $i$ compute $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t[:, :, \mathcal{S}_i], PE_t)$
    Compute $\boldsymbol{s} = \nabla \log p(\boldsymbol{x}_t)$ using (9)
    Compute $\hat{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_0 | \boldsymbol{x}_t]$ using Tweedie's formula
    Set $\hat{\boldsymbol{x}}_t' = \mathrm{CG}(A^*A, A^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t)$
    Sample $\boldsymbol{x}_{t-1}$ using $\hat{\boldsymbol{x}}_t'$ and $\boldsymbol{s}$ via DDIM sampling
  **end for**
Return $\boldsymbol{x}$.

# Experimental Setting



- We perform experiments on two CT datasets: AAPM LDCT, and LIDC.
- For each dataset, we perform two tasks: Sparse-View CT reconstruction and Limited-Angle CT reconstruction.
- For Sparse-view CT reconstruction, we consider three experimental settings: 4 views, 6 views and 8 views.
- For LACT, we consider 90 degree limited angle CT reconstruction
- Results are evaluate on the entire of a patient in the validation set of both AAPM and LIDC

UNIVERSITY OF MICHIGAN

# Quantitative Results



**Algorithm 1** DiffusionBlend++

**Require:** $A$, $M$, $\zeta_i > 0$, $k$, $\boldsymbol{y}$
  Initialize $\boldsymbol{x}_T \sim \mathcal{N}(0, \sigma_T^2 \boldsymbol{I})$
  **for** $t = T : 1$ **do**
    Randomly select a partition $\mathcal{S} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_r$ (the partition can either be of adjacency slices or jumping slices)
    Compute the relative positional encoding $PE_t$
    For each $i$ compute $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t[:, :, \mathcal{S}_i], PE_t)$
    Compute $\boldsymbol{s} = \nabla \log p(\boldsymbol{x}_t)$ using (9)
    Compute $\hat{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_0 | \boldsymbol{x}_t]$ using Tweedie's formula
    Set $\hat{\boldsymbol{x}}_t' = \mathrm{CG}(A^*A, A^*\boldsymbol{y}, \hat{\boldsymbol{x}}_t)$
    Sample $\boldsymbol{x}_{t-1}$ using $\hat{\boldsymbol{x}}_t'$ and $\boldsymbol{s}$ via DDIM sampling
  **end for**
Return $\boldsymbol{x}$.

# Quantitative Results



| Method | AAPM and LIDC Datasets | | | | | | Sparse-View CT Reconstruction on LIDC | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AAPM Dataset | | LIDC Dataset | | Average | | 8 Views | | 6 Views | | 4 Views | |
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| FBP | 22.63 | 0.206 | 14.70 | 0.369 | 18.67 | 0.288 | 14.78 | 0.206 | 14.10 | 0.181 | 13.10 | 0.165 |
| FBP-UNet | 28.44 | 0.930 | 29.54 | 0.887 | 28.99 | 0.909 | 28.87 | 0.858 | 26.59 | 0.793 | 25.37 | 0.744 |
| DiffusionMBIR | 27.63 | 0.908 | - | - | - | - | 33.29 | 0.922 | 31.69 | 0.903 | 29.21 | 0.868 |
| TPDM | 28.12 | 0.833 | 25.78 | 0.804 | 22.29 | 0.735 | - | - | - | - | - | - |
| DDS 2D | 28.82 | 0.922 | 27.96 | 0.842 | 28.39 | 0.882 | 31.60 | 0.898 | 29.99 | 0.871 | 28.03 | 0.830 |
| DDS | 29.03 | 0.934 | 28.13 | 0.879 | 28.58 | 0.903 | 32.51 | 0.920 | 30.83 | 0.924 | 27.61 | 0.828 |
| DiffusionBlend (Ours) | 36.62 | 0.972 | 31.02 | 0.924 | 33.82 | 0.948 | 34.47 | 0.934 | 31.16 | 0.908 | 28.24 | 0.859 |
| DiffusionBlend++ (Ours) | 37.45 | 0.976 | 34.64 | 0.935 | 36.02 | 0.946 | 35.66 | 0.947 | 33.97 | 0.935 | 31.38 | 0.913 |

Table 2: Comprehensive comparison of quantitative results on Limited-Angle and Sparse-View CT Reconstruction on Coronal View for AAPM and LIDC datasets. Best results are in bold.

UNIVERSITY OF MICHIGAN

# References I



Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., Ye, J. C. (2022). Diffusion posterior sampling for general noisy inverse problems. arXiv preprint arXiv:2209.14687.

Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

Fabian, Zalan, Berk Tinaz, and Mahdi Soltanolkotabi. "DiracDiffusion: Denoising and Incremental Reconstruction with Assured Data-Consistency." arXiv preprint arXiv:2303.14353 (2023).

Song, Yang, et al. "Solving inverse problems in medical imaging with score-based generative models." International Conference on Learning Representations. 2022

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in Neural Information Processing Systems 34 (2021): 8780-8794.

Shen, Liyue, Wei Zhao, and Lei Xing. "Patient-specific reconstruction of volumetric computed tomography images from a single projection view via deep learning." Nature biomedical engineering 3.11 (2019): 880-888.

Song, Bowen, Liyue Shen, and Lei Xing. "PINER: Prior-informed Implicit Neural Representation Learning for Test-time Adaptation in Sparse-view CT Reconstruction." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023.

Ongie, Gregory, et al. "Deep learning techniques for inverse problems in imaging." IEEE Journal on Selected Areas in Information Theory 1.1 (2020): 39-56.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

Song, Bowen, et al. "Solving Inverse Problems with Latent Diffusion Models via Hard Data Consistency." International Conference on Learning Representations. 2024.

# Thank You