


SCHOOL OF
COMPUTING &
DATA SCIENCE
The University of Hong Kong

HKU Musketeers Foundation
Institute of Data Science
香港大學同心基金數據科學研究院

On the Sampling Theory for Auto-Regressive Diffusion Inference

Difan Zou

May. 28-29, 2025

HKU IDS Interdisciplinary Workshop – Exploring the Foundations:
Fundamental AI and Theoretical Machine Learning

Joint work with Xunpeng Huang, Yujin Han, Hanze Dong, Yi Zhang, Yian Ma, and Tong Zhang

Diffusion Model

Sohl-Dickstein et al, ICML 2015

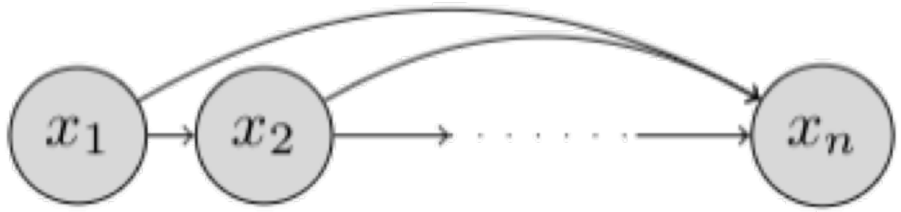
Adding noise via OU process

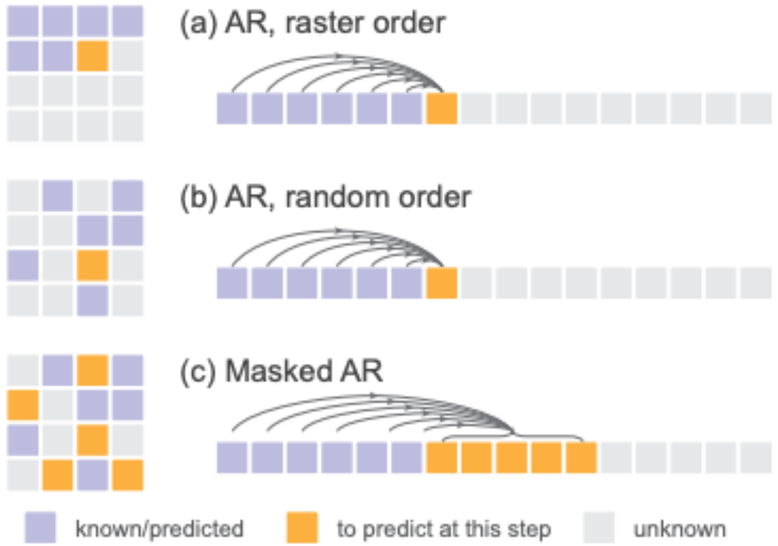
$$p_0(x) \longrightarrow p_T(x) \approx N(0,\mathbf{I})$$

$$p_T^{\leftarrow}(x) \approx p_0(x) \longleftarrow p_0^{\leftarrow}(x) = N(0,\mathbf{I})$$

Denoising via reversing OU process

Auto-regressive Generative Models

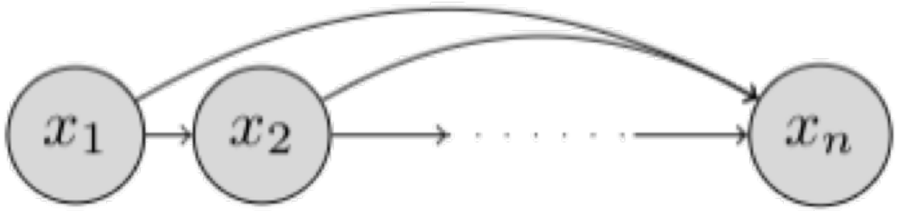


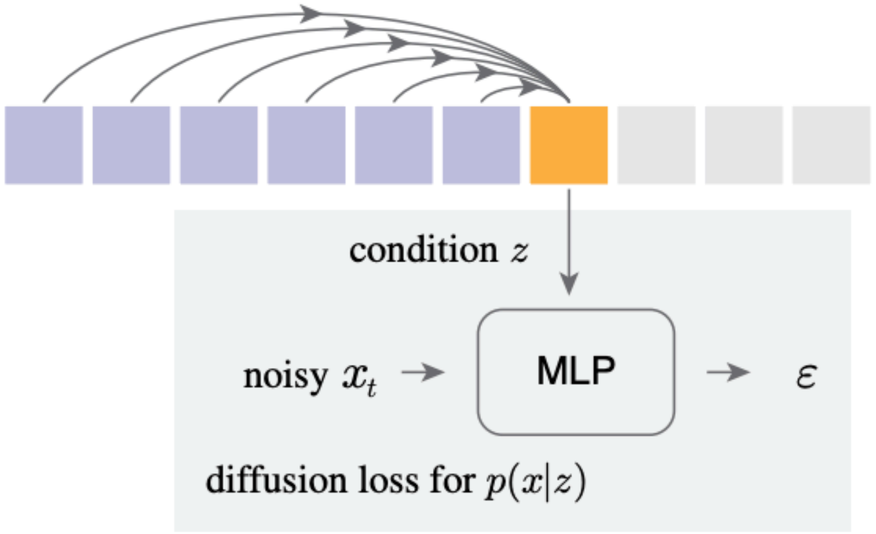


- The data is decomposed into $x = [x_1, x_2, \ldots, x_n]$, each token is **encoded in the discrete space**.
- The tokens are generated in a sequential manner: $x_k \sim p(x \,|\, x_1, \ldots, x_{k-1})$
- In the standard AR generative model, $p(\,\cdot\,|\, x_1, \ldots, x_{k-1})$ is implemented as **next-token prediction** using transformer models.

https://deepgenerativemodels.github.io/notes/autoregressive/

AR Diffusion Inference





- When modeling the conditional distribution $p(\cdot \,|\, x_1, \ldots, x_{k-1})$, using **discrete encoding** may lead to information loss.
- AR diffusion inference leverages **diffusion model** to model $p(\cdot \,|\, x_1, \ldots, x_{k-1})$, which enables conditional generation in continuous space.

Li et al., Autoregressive Image Generation without Vector Quantization, NeurIPS 2024

AR Diffusion Inference

- Procedure of AR diffusion model:
  - Decompose the data x into K parts.
  - Given the previous tokens $x_1, \ldots, x_k$ (in continuous space), we first leverage a encoder g_{θ_e} to get the condition embedding $z = g_{\theta_e}(x_{[1:k]})$.
  - Then, we leverage the conditional diffusion model s_θ to generate the next token $x_{k+1} = s_\theta(\xi, k, z)$.

Li et al., Autoregressive Image Generation without Vector Quantization, NeurIPS 2024

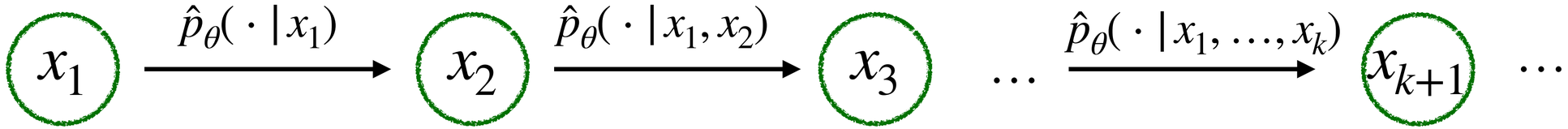
AR Diffusion Model VS. Standard Diffusion Model

- Some facts:
  - When $K = 1$, AR DM reduces to the standard DM.
  - AR DM learn many conditional probabilities, while standard DM directly learns the joint distribution.
  - AR DM needs to perform inference in a sequential manner (token-by-token), while standard DM generates data in parallel.
  - Each step of AR DM operates on the space with low dimension, while DM works on full dimension space.

Goal of our work: build the theoretical foundation of AR DM to explain the pros and cons of the auto-regressive paradigm.

Sampling Error of AR Diffusion Model

- Inference Process



- Sampling error analysis:
  - In each step, the DM sampling process $\hat{p}_\theta(\cdot \mid x_1, \ldots, x_k)$ could lead to sampling error.
  - The single-step error will accumulate during the sequential sampling process.

Our analysis needs to (1) study single-step sampling error; and

(2) study the error propagation during AR process.

DM Sampling Algorithm: DDPM

- Forward process: adding noise

$$p_{t|0}(\mathbf{x}_t \mid \mathbf{x}_0) = N(\alpha_t \mathbf{x}_0, \beta_t \mathbf{I})$$

Gaussian Transition Kernel

- Reverse process: generating via denoising

$$p_{t|t-1}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \approx N(\mu(\mathbf{x}_t, t), \mathbf{\Sigma}(\mathbf{x}_t, t))$$

Use a gaussian kernel to approximate the single reverse step.

Data distribution

Gaussian noise

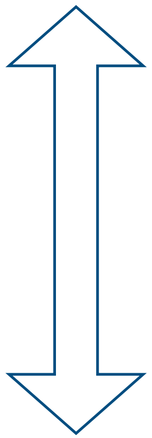
$\mathbf{x}_0$ $\mathbf{x}_{t-1}$ $\mathbf{x}_t$

$$\mathrm{d}\hat{\mathbf{x}}_t = (\hat{\mathbf{x}}_t + 2\nabla \ln p_{T-t}(\hat{\mathbf{x}}_t))\,\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \quad \hat{\mathbf{x}}_0 \sim p_T.$$

Theoretical Analysis for DDPM

- Error Decomposition of DDPM

$$\mathrm{d}\mathbf{x}_t^{\leftarrow} = \left(\mathbf{x}_t^{\leftarrow} + \mathbf{v}_\theta(\mathbf{x}_{k\eta}^{\leftarrow}, k)\right)\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \quad t \in [k\eta, (k+1)\eta]$$



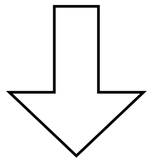
- Truncation error: $p(\mathbf{x}_T) \approx N(0, \mathbf{I})$
- Discretization error: $\nabla \ln p(\mathbf{x}_{k\eta}) \approx \nabla p_t(\mathbf{x})$
- Score estimation error: $\mathbf{v}_\theta(\mathbf{x}) \approx \nabla p_{T-t}(\mathbf{x})$
- Error propagation

$$\mathrm{d}\hat{\mathbf{x}}_t = (\hat{\mathbf{x}}_t + 2\nabla \ln p_{T-t}(\hat{\mathbf{x}}_t))\,\mathrm{d}t + \sqrt{2}\mathrm{d}B_t, \quad \hat{\mathbf{x}}_0 \sim p_T.$$

Theoretical Analysis for DDPM

- To guarantee small sampling error

  Discretization error: $\|\nabla \ln p(\mathbf{x}_{k\eta}) - \nabla p_t(\mathbf{x})\| \sim \mathrm{poly}(\eta)$

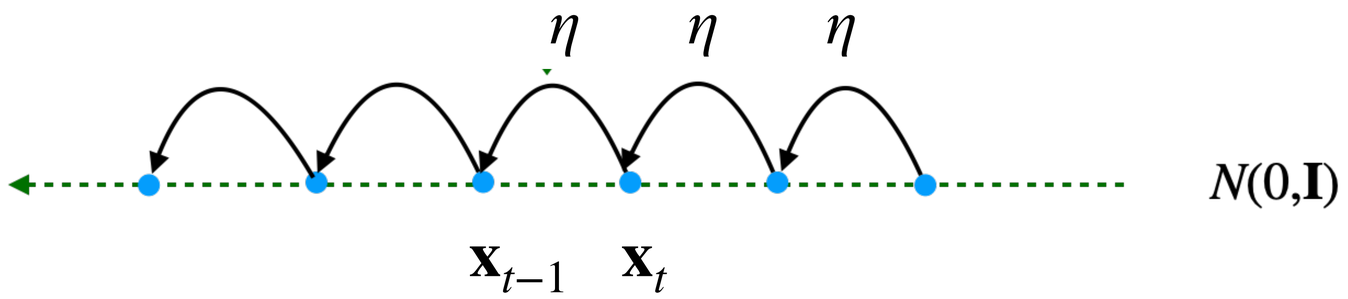
  

  - We need to use small stepsize η to ensure small discretization error.
  - Consequently, we need to use a large iteration number $K = T/\eta \sim \log(1/\epsilon)/\eta$.

Chen, S., Chewi, S., Li, J., Li, Y., Salim, A., & Zhang, A. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*.

Lee, H., Lu, J., & Tan, Y. (2022). Convergence for score-based generative modeling with polynomial complexity. *Advances in Neural Information Processing Systems*, *35*, 22870-22882.

Revisit DDPM Algorithm

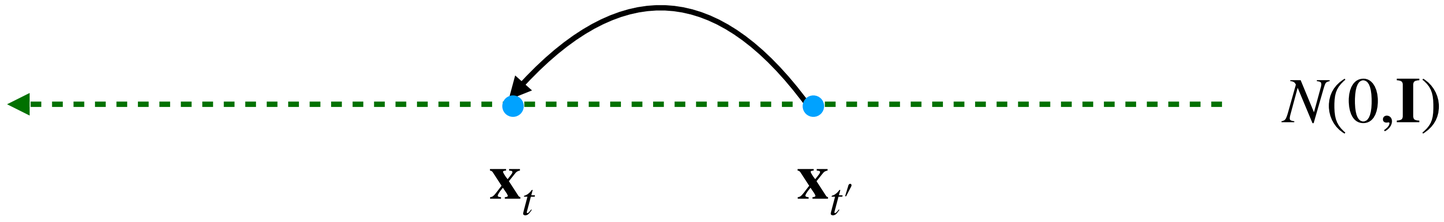


DDPM algorithm aims to sample $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$ using a Gaussian kernel.

$$p_{t-1|t}(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t) \approx N(\mu(\mathbf{x}_t, t), \mathbf{\Sigma}(\mathbf{x}_t, t))$$

- Gaussian kernel is easy to implement.
- Gaussian approximation can only be good when η is small .
- Then we must require many reverse sampling steps (T/η).
- What if we consider larger η?

Reverse Transitional Kernel (RTK)



We can get the exact formula of the **reverse transition kernel** $p_{t|t'}(\mathbf{x}_t \mid \mathbf{x}_{t'})$

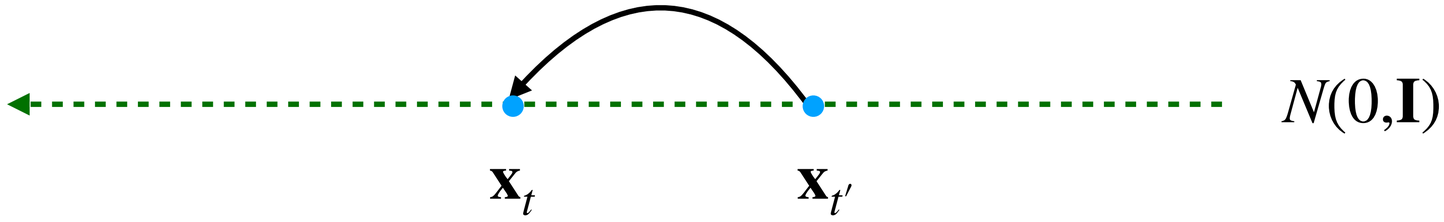
$$p_{t|t'}(\mathbf{x}_t \mid \mathbf{x}_{t'}) \propto p_{t|t'}(\mathbf{x}_t \mid \mathbf{x}_{t'}) \cdot p_{t'}(\mathbf{x}_{t'}) = \boxed{p_{t'|t}(\mathbf{x}_{t'} \mid \mathbf{x}_t)} \cdot \boxed{p_t(\mathbf{x}_t)}$$

Forward transition kernel, which is a Gaussian

Marginal distribution at time t, $\propto e^{-f_t(\mathbf{x}_t)}$

$$\Rightarrow \quad p_{t|t'}(\mathbf{x} \mid \mathbf{x}') = \exp\left(-f_t(\mathbf{x}) - \frac{\|\mathbf{x}' - \mathbf{x} \cdot e^{-2(t'-t)}\|^2}{2(1 - e^{-2(t'-t)})} \right)$$

Reverse Transitional Kernel (RTK)

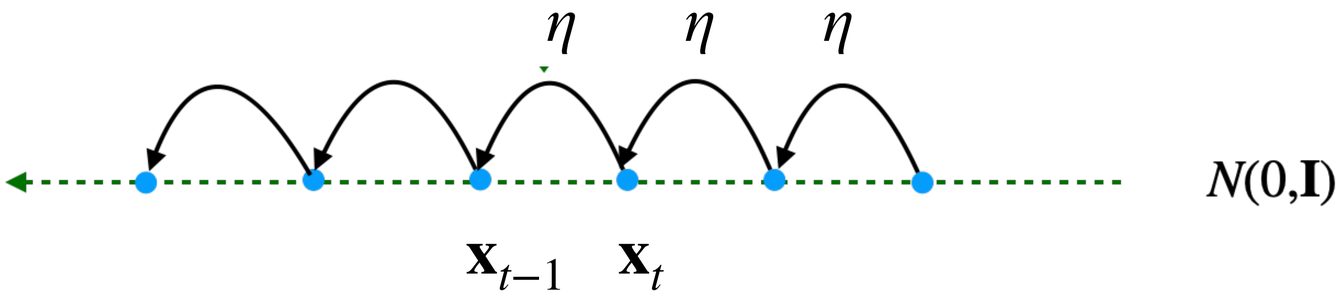


Reverse transitional kernel

$$p_{t|t'}(\mathbf{x}\,|\,\mathbf{x}') = \exp\left(-f_t(\mathbf{x}) - \frac{\|\mathbf{x}' - \mathbf{x}\cdot e^{-2(t'-t)}\|^2}{2(1-e^{-2(t'-t)})} \right)$$

- When $t' - t$ is very small, the quadratic term dominates, the kernel can be well approximated by Gaussian, i.e., DDPM.
- When $t' - t$ is large, we need to use more complicated sampler to achieve high-accuracy sampling.

Trade-off of RTK



Consider fixed segment

$$p_{(k-1)\eta|k\eta}(\mathbf{x}\,|\,\mathbf{x}') = \exp\left(-f_{(k-1)\eta}(\mathbf{x}) - \frac{\|\mathbf{x}' - \mathbf{x}\cdot e^{-2\eta}\|^2}{2(1-e^{-2\eta})} \right)$$

- Smaller η implies easier sampling subproblems, but the number of subproblems increases (e.g., DDPM).
- Larger η implies smaller number of subproblems, but the problem becomes easier.
- How to pick a proper η?

Hardness of RTK

RTK target $$p_{(k-1)\eta|k\eta}(\mathbf{x}\,|\,\mathbf{x}') = \exp\left(-f_{(k-1)\eta}(\mathbf{x}) - \frac{\|\mathbf{x}' - \mathbf{x}\cdot e^{-2\eta}\|^2}{2(1-e^{-2\eta})} \right)$$

We can verify the **log-concavity**, let's check the Hessian of $\log p_{(k-1)\eta|k\eta}(\mathbf{x}\,|\,\mathbf{x}')$

$$-\nabla^2_{\mathbf{x}} \log p_{(k-1)\eta|k\eta}(\mathbf{x}\,|\,\mathbf{x}') = -\nabla^2_{\mathbf{x}} f_{(k-1)\eta}(\mathbf{x}) + \frac{e^{-2\eta}}{1-e^{2\eta}}$$

- Assume that $f_t(\mathbf{x})$ is L-smooth.
- Then we can set $\eta \le 1/2 \cdot \log(1 + 1/(2L))$ to ensure the log-concavity.
- We can just set $\eta = 1/2 \cdot \log(1 + 1/(2L))$ to ensure a small number of subproblems.

Diffusion Inference with RTK

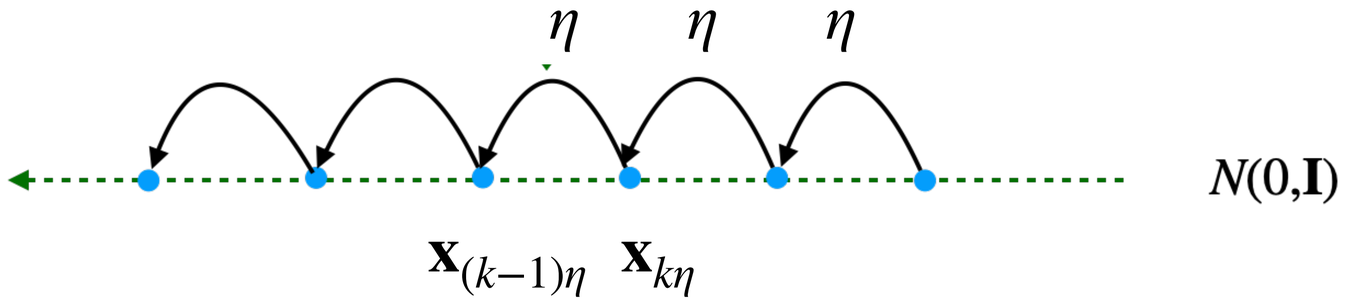
Algorithm INFERENCE WITH REVERSE TRANSITION KERNEL (RTK)

1: **Input:** Initial particle $\hat{\mathbf{x}}_0$ sampled from the standard Gaussian distribution, Iteration number K, Step size η, required convergence accuracy ϵ;
2: **for** $k = 0$ to $K - 1$ **do**
3: Draw sample $\hat{\mathbf{x}}_{(k+1)\eta}$ with MCMCs from $\hat{p}_{(k+1)\eta|k\eta}(\cdot|\hat{\mathbf{x}}_{k\eta})$ which is closed to the ground-truth reverse transition kernel, i.e.,

$$p^{\leftarrow}_{(k+1)\eta|k\eta}(\boldsymbol{z}|\hat{\mathbf{x}}_{k\eta}) \propto \exp\left(-g(\boldsymbol{z})\right) := \exp\left(-f_{(K-k-1)\eta}(\boldsymbol{z}) - \frac{\|\hat{\mathbf{x}}_{k\eta} - \boldsymbol{z}\cdot e^{-\eta}\|^2}{2(1-e^{-2\eta})}\right). \tag{23}$$

4: **end for**
5: **return** $\hat{\mathbf{x}}_K$.

Efficiently solving RTK is crucial.



Solving RTK

Sampling from the distribution $p_{(k-1)\eta|k\eta}(\mathbf{x}\,|\,\mathbf{x}') = \exp\left(-f_{(k-1)\eta}(\mathbf{x}) - \frac{\|\mathbf{x}' - \mathbf{x}\cdot e^{-2\eta}\|^2}{2(1-e^{-2\eta})} \right)$

Multiple sampling algorithms can be applied:

- Unadjusted Langevin Algorithms (ULA)
- Underdamped Langevin Dynamics (ULD)
  - provide fast sampling and requires mild assumptions
- Metropolis Adjusted Langevin Algorithms (MALA)
  - Provide even faster sampling but need stricter assumptions
- Hamiltonian Monte Carlo (HMC)
- …

RTK with ULD

Algorithm 3 ULD FOR RTK INFERENCE

Input: Returned particle of the previous iteration $\boldsymbol{x}_0$, current iteration number k, inner iteration number S, inner step size τ, velocity diffusion coefficient γ, required convergence accuracy ϵ;

Initialize the particle and velocity pair, i.e., $(\hat{\boldsymbol{z}}_0, \hat{\boldsymbol{v}}_0)$ with a Gaussian type product measure, i.e., $\mathcal{N}(\mathbf{0}, e^{2\eta} - 1) \otimes \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;

for $t = s$ to $S - 1$ **do**

  Draw noise sample pair (ξ_s^z, ξ_s^v) from a Gaussian type distribution.

  $\hat{\boldsymbol{z}}_{s+1} = \hat{\boldsymbol{z}}_s + \gamma^{-1}(1 - e^{-\gamma\tau})\hat{\boldsymbol{v}}_s - \gamma^{-1}(\tau - \gamma^{-1}(1 - e^{-\gamma\tau}))s_\theta(\hat{\boldsymbol{z}}_s) + \xi_s^z$

  $\hat{\boldsymbol{v}}_{s+1} = e^{-\gamma\tau}\hat{\boldsymbol{v}}_s - \gamma^{-1}(1 - e^{-\gamma\tau})s_\theta(\hat{\boldsymbol{z}}_s) + \xi_t^v$

return $\boldsymbol{z}_S$;

Neural network approximation of the score function $\nabla f_s(\hat{\mathbf{z}}_s)$

Here the random Gaussian vectors are generated via:

$$(\xi_s^z, \xi_s^v) \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \frac{2}{\gamma}\left(\tau - \frac{2}{\gamma}\left(1 - e^{-\gamma\tau}\right)\right) + \frac{1}{2\gamma}\left(1 - e^{-2\gamma\tau}\right) & \frac{1}{\gamma}\left(1 - 2e^{-\gamma\tau} + e^{-2\gamma\tau}\right) \\ \frac{1}{\gamma}\left(1 - 2e^{-\gamma\tau} + e^{-2\gamma\tau}\right) & 1 - e^{-2\gamma\tau} \end{bmatrix}\right)$$

RTK with ULD: Rates

Theorem: *Under the assumptions on the score estimation*

$$\eta = 1/2 \cdot \log[(2L+1)/2L] \quad \text{and} \quad K = 4L \cdot \log[((1+L^2)d + \|\nabla f_*(\mathbf{0})\|^2)^2 \cdot \epsilon^{-2}]$$

and implement Step 3 of Alg. 2 with ULD. For the k-th run of ULD, we require Gaussian-type initialization and high-accurate score estimation, i.e.,

$$\hat{\pi}_0 = \mathcal{N}(\mathbf{0}, e^{2\eta} - 1) \otimes \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \quad \text{and} \quad \epsilon_{\text{score}} = \tilde{\mathcal{O}}(\epsilon/\sqrt{L}).$$

If we set the hyperparameters of inner loops as follows. the step size and the iteration number as

$$\tau = \tilde{\Theta}\left(\epsilon d^{-1/2} L^{-1/2} \cdot \left(\log\left[\frac{L(d + m_2^2 + \|\boldsymbol{x}_0\|^2)}{\epsilon^2}\right]\right)^{-1/2}\right)$$

$$S = \tilde{\Theta}\left(\epsilon^{-1} d^{1/2} \cdot \left(\log\left[\frac{L(d + m_2^2 + \|\boldsymbol{x}_0\|^2)}{\epsilon^2}\right]\right)^{1/2}\right).$$

It can achieve $\|\hat{p}_{K\eta} - p_*\|_{\text{TV}} \lesssim \epsilon$ *with an* $\tilde{\mathcal{O}}\left(L^2 d^{1/2} \epsilon^{-1}\right)$ *gradient complexity.*

RTK with MALA

Algorithm 2 MALA/Projected MALA for RTK Inference

Input: Returned particle of the previous iteration $\boldsymbol{x}_0$, current iteration number k, inner iteration number S, inner step size τ, required convergence accuracy ϵ;

Draw the initial particle $\mathbf{z}_0$ from

$$\frac{\mu_0(\mathrm{d}\boldsymbol{z})}{\mathrm{d}\boldsymbol{z}} \propto \exp\left(-L\|\boldsymbol{z}\|^2 - \frac{\|\boldsymbol{x}_0 - e^{-\eta}\boldsymbol{z}\|^2}{2(1-e^{-2\eta})}\right).$$

for $s = 0$ to $S-1$ **do**

  Draw a sample $\tilde{\boldsymbol{z}}_s$ from the Gaussian distribution $\mathcal{N}(\boldsymbol{z}_s - \tau \cdot s_\theta(\boldsymbol{z}_s), 2\tau \boldsymbol{I})$;

  if $\boldsymbol{z}_{s+1} \notin \mathcal{B}(\boldsymbol{z}_s, r) \cap \mathcal{B}(\mathbf{0}, R)$ **then**

    $\boldsymbol{z}_{s+1} = \boldsymbol{z}_s$; ▷ This condition step is only activated for Projected MALA.

    continue;

  Calculate the accept rate as

$$a(\tilde{\boldsymbol{z}}_s - (\boldsymbol{z}_s - \tau \cdot s_\theta(\boldsymbol{z}_s)), \boldsymbol{z}_s) = \min\left\{1, \exp\left(r_g(\boldsymbol{z}_s, \tilde{\boldsymbol{z}}_s) + \frac{\|\tilde{\boldsymbol{z}}_s - \boldsymbol{z}_s + \tau \cdot s_\theta(\boldsymbol{z}_s)\|^2 - \|\boldsymbol{z}_s - \tilde{\boldsymbol{z}}_s + \tau \cdot s_\theta(\tilde{\boldsymbol{z}}_s)\|^2}{4\tau}\right)\right\};$$

MH Acceptance probability for mitigating the bias

  Update the particle $\boldsymbol{z}_{s+1} = \tilde{\boldsymbol{z}}_s$ with probability a, otherwise $\boldsymbol{z}_{s+1} = \boldsymbol{z}_s$.

return $\mathbf{z}_S$;

$r_g(\mathbf{z}', \mathbf{z}) \approx \log p_t(\mathbf{z}') - \log p_t(\mathbf{z})$ denotes the approximation of the energy difference.

RTK with MALA: Assumptions

MALA algorithm needs both **score**, i.e., $\nabla \log p$ and the **energy difference** $\log p(\mathbf{z}) - \log p(\mathbf{z}')$.

Assumptions:

- For all $t \geq 1$, the score ∇p_t is L-Lipschitz.
- The second moment of the data distribution p_* is upper bounded.
- The score and energy difference estimation errors satisfy: for all $\mathbf{z}$ and $\mathbf{z}'$

$$\|s_{\theta,t}(\mathbf{z}) - \nabla \log p_t(\mathbf{z})\| \leq \epsilon_{\text{score}}$$

$$|\, r_t(\mathbf{z}', \mathbf{z}) + \log p_t(\mathbf{z}') - \log p_t(\mathbf{z}) \,| \leq \epsilon_{\text{energy}}$$

RTK with MALA: Rates

Theorem: *Under the assumptions on the score and energy difference estimation*

Suppose the estimation errors of score and energy difference satisfy

$$\epsilon_{\text{score}} \le \frac{\rho\epsilon}{Ld^{1/2}} \quad \text{and} \quad \epsilon_{\text{energy}} \le \frac{\rho\epsilon}{L^2 \cdot (d^{1/2} + m_2 + Z)},$$

If we implement Alg. 2 with the projected version of MALA with the following hyperparameter settings

$$\eta = \frac{1}{2}\log\frac{2L+1}{2L}, \quad K = 4L \cdot \log\frac{(1+L^2)d + \|\nabla f_*(\mathbf{0})\|^2}{\epsilon^2}$$

and

$$\frac{\mu_0(\mathrm{d}\boldsymbol{z})}{\mathrm{d}\boldsymbol{z}} \propto \exp\left(-L\|\boldsymbol{z}\|^2 - \frac{\|\hat{\boldsymbol{x}}_k - e^{-\eta}\boldsymbol{z}\|^2}{2(1-e^{-2\eta})}\right),$$

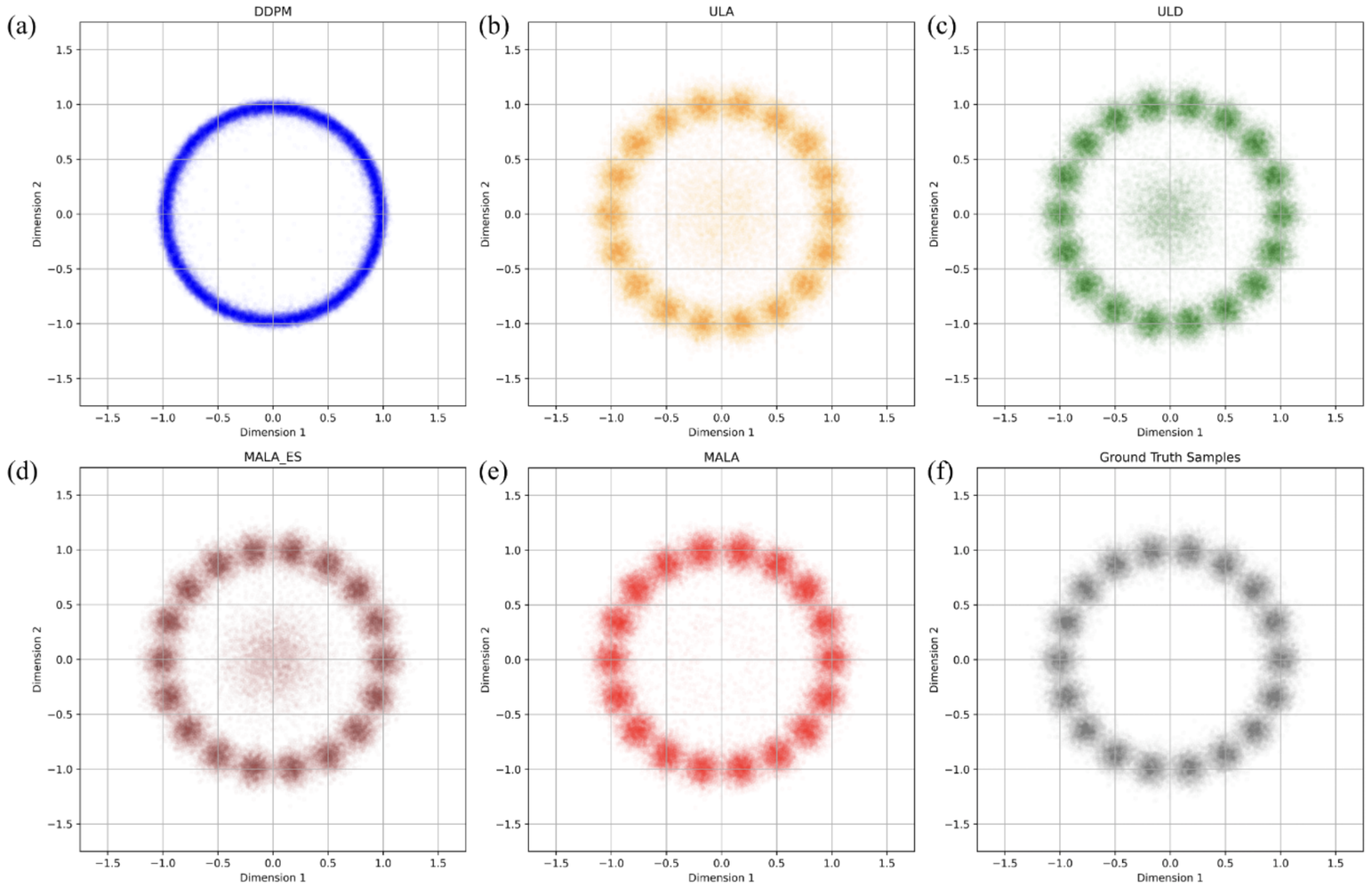
it has $\|\hat{p}_{K\eta} - p_*\|_{\text{TV}} \le \tilde{\mathcal{O}}(\epsilon)$ *with an* $\mathcal{O}(L^4\rho^{-2} \cdot (d + m_2^2)^2 Z^2 \cdot \log(d/\epsilon))$ *complexity.*

Comparison with Other Methods

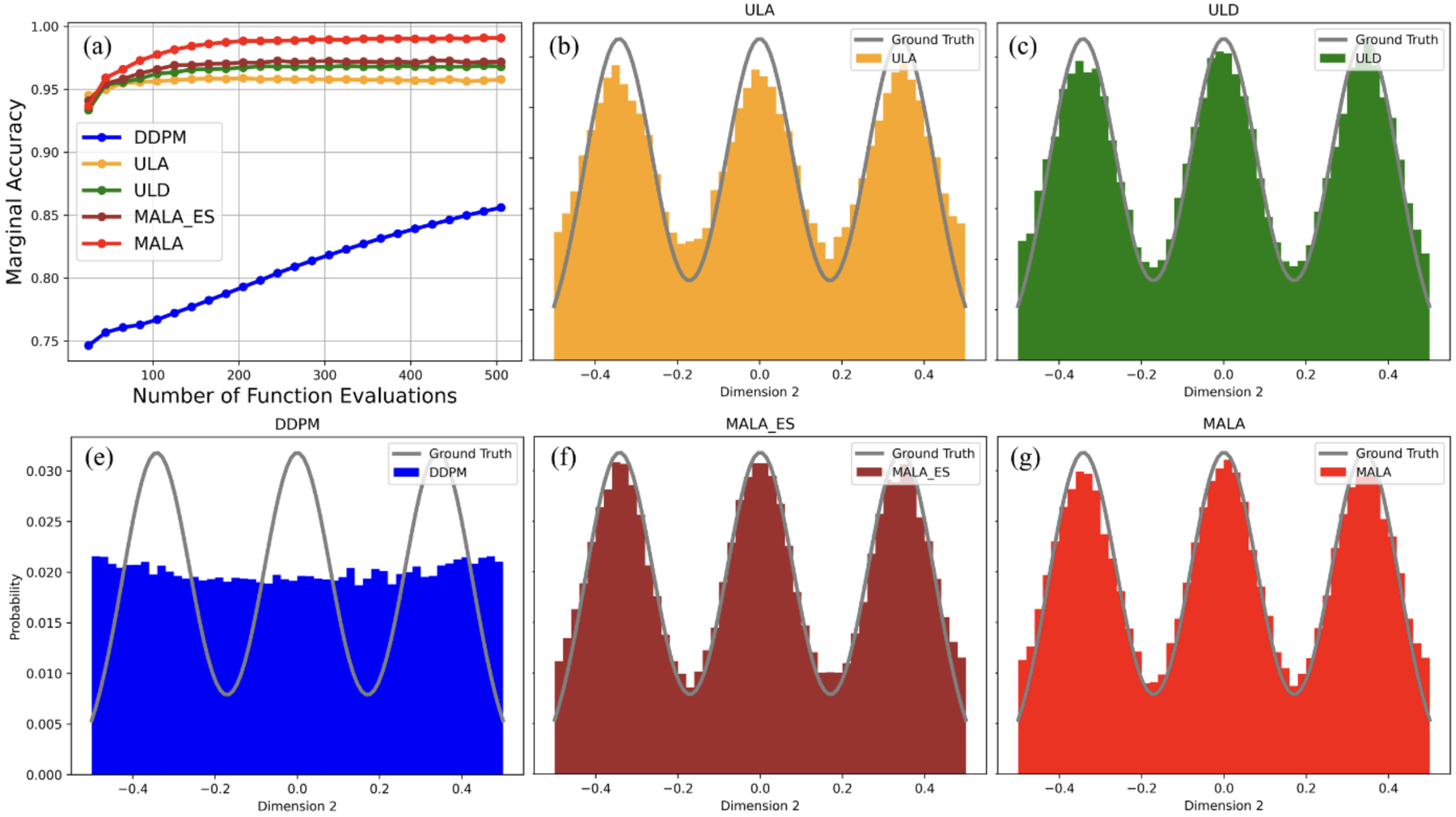
Results	Algorithm	Complexity
Chen et al. (2022)	DDPM (SDE-based)	$\tilde{\mathcal{O}}(L^2 d\epsilon^{-2})$
Chen et al. (2024)	DPOM (ODE-based)	$\tilde{\mathcal{O}}(L^3 d\epsilon^{-2})$
Chen et al. (2024)	DPUM (ODE-based)	$\tilde{\mathcal{O}}(L^2 d^{1/2}\epsilon^{-1})$
Li et al. (2023)	ODE-based sampler	$\tilde{\mathcal{O}}(d^3\epsilon^{-1})$
Ours	RTK-MALA	$\mathcal{O}(L^4 d^2 \log(d/\epsilon))$
Ours	RTK-ULD	$\tilde{\mathcal{O}}(L^2 d^{1/2}\epsilon^{-1})$

Table. Comparison with prior works for RTK-based methods. The complexity denotes the number of calls for the score estimation to achieve $\|\hat{p}_{K\eta} - p_*\|_{\mathrm{TV}} \leq \tilde{\mathcal{O}}(\epsilon)$. d and ϵ mean the dimension and error tolerance. Compared with the state-of-the-art result, RTK-ULD achieves the best dependence for both d and ϵ. Though RTK-MALA requires slightly stricter assumptions and worse dimension dependence, a linear convergence w.r.t. ϵ makes it suit high-accuracy sampling tasks.

Numerical Experiments: Sampling Gaussian Mixture



Numerical Experiments: Sampling Gaussian Mixture



Autoregressive Diffusion Model

For autoregressive diffusion model, we need to perform K diffusion inference steps. Then, if direc
adapt the result of single diffusion model, we need:

- Each conditional score $\nabla \log p_t(x \,|\, x_1, \ldots, x_k)$ should be Lipschitz
- Each target conditional distribution $p_*(\,\cdot\,|\, x_1, \ldots, x_k)$ need to have bounded second moment.
- Each score estimation need to be small.

$$\mathbb{E}_{p_t(\cdot|x_1,\ldots,x_k)}\left[\|s_{\theta,t}(\mathbf{z}; x_1, \ldots, x_k) - \nabla \log p_t(\mathbf{z} \,|\, x_1, \ldots, x_k)\|^2\right] \leq \epsilon^2_{\text{score}} \quad \forall k = 1,\ldots,K$$

Do we really need these assumptions for autoregressive diffusion model?

Autoregressive Diffusion Model: Assumptions

In fact, we do not need to make assumptions on all conditional targets, but only require

- The target distribution p_* satisfies $\|\nabla^2 \log p_*\|_2 \leq L,\ \|\nabla \log p_*\|_2 \leq \sqrt{L}$
- The target distribution p_* has bounded moment.
- The average of score estimation errors should be small:

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{p_t(\cdot|x_1,\ldots,x_k)}\left[\|s_{\theta,t}(\mathbf{z}; x_1, \ldots, x_k) - \nabla \log p_t(\mathbf{z}\,|\,x_1, \ldots, x_k)\|^2\right] \leq \epsilon_{\text{score}}^2$$

The assumption is almost as mild as that for standard diffusion model!

Autoregressive Diffusion Model: General Theory

Theorem: *Under the aforementioned assumptions and apply appropriate learning rate for diffusion sampling, let η be the base stepsize and R be the iteration number in each DM, then considering DDPM solver for diffusion inference, we have*

$$\mathrm{KL}\left(p_* \middle\| \hat{p}_*\right) \lesssim \underbrace{2e^{-2T} L \cdot (m_0 + d)}_{\text{Truncation error}} + \underbrace{(L^2 R\eta^2 + T\eta) \cdot d + \eta m_0}_{\text{Accumulated discretization error}} + \underbrace{\eta K R \cdot \epsilon_{\text{score}}^2}_{\text{Accumulated score estimation error}}.$$

- Discretization error is accumulated in dimension: $d = \sum_{i=1}^{K} d_i$
- The score estimation error is accumulated with an additional K factor (but ϵ_{score} is different).

Huang et al., Capturing Conditional Dependence via Auto-regressive Diffusion Models, ArXiv 2025

Autoregressive Diffusion Model: General Theory

Theorem: *Under the aforementioned assumptions and apply appropriate learning rate for diffusion sampling, let η be the base learning rate and R be the iteration number in each DM, then considering DDPM solver for diffusion inference, we have*

$$\mathrm{KL}\left(p_* \middle\| \hat{p}_*\right) \lesssim 2e^{-2T} L \cdot (m_0 + d) + \boxed{(L^2 R\eta^2 + T\eta) \cdot d + \eta m_0} + \eta K R \cdot \epsilon_{\mathrm{score}}^2.$$

Accumulated discretization error

- The discretization error can be improved using RTK-ULD or RTK-MALA (need stronger assumptions.)
  - For DDPM, the total complexity will be $\tilde{O}(Kd\epsilon^{-2})$.
  - For RTK-ULD, the total complexity will be $\tilde{O}(Kd^{1/2}\epsilon^{-1})$.

Autoregressive diffusion model requires longer inference time!

Autoregressive Diffusion Model: Conditional Dependency

Lemma: *Under the aforementioned assumptions and apply appropriate learning rate for diffusion sampling, considering DDPM solver, we have*

$$\begin{aligned}
&\mathrm{KL}\left(p_{*,k+1|[1:k]}(\cdot|\boldsymbol{x}_{[1:k]})\middle\|\hat{p}_{*,k+1|[1:k]}(\cdot|\boldsymbol{x}_{[1:k]})\right) \\
&\lesssim e^{-2T}\cdot\left(2Ld_{k+1}+\mathbb{E}_{p_{*,k+1|[1:k]}(\cdot|\boldsymbol{x}_{[1:k]})}\left[\|\mathbf{y}\|^2\right]\right)+\eta\cdot\sum_{r=0}^{R-1}\tilde{L}_{k+1,r}(\boldsymbol{\theta}|\boldsymbol{x}_{[1:k]})+d_{k+1}L^2R\eta^2 \\
&\quad+d_{k+1}T\eta+\eta\mathbb{E}_{p_{*,k+1|[1:k]}(\cdot|\boldsymbol{x}_{[1:k]})}\left[\|\mathbf{y}\|^2\right].
\end{aligned}$$

- For any conditional distribution, AR DM can well capture it with appropriate choice of step size.
- This implies that the dependency between tokens x_i and x_j can be well reflected in the generated data $x=[x_1,\ldots,x_K]$

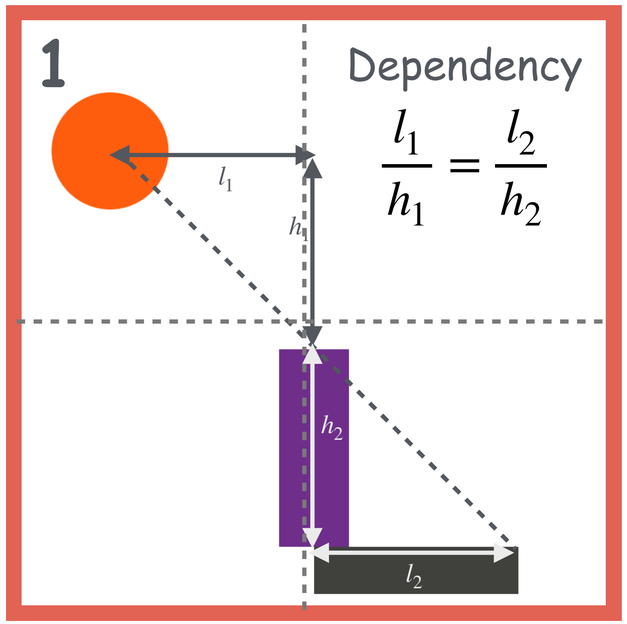
Single Diffusion Model: Possible Failure

Lemma 4.4. *Consider random vectors* $\mathbf{y} \in \mathbb{R}^{d_{k+1}}$ *and* $\boldsymbol{x} \in \mathbb{R}^{d_1+d_2+\cdots+d_k}$. *For any error threshold* $\varepsilon \in (0, 1/2]$ *and for any* $M \in \mathbb{R}$, *there exists a pair of Gaussian probability densities* $(p_*(\mathbf{y}, \boldsymbol{x}), \hat{p}_*(\mathbf{y}, \boldsymbol{x}))$, *such that* $\mathrm{KL}\left(p_*(\mathbf{y}, \boldsymbol{x}) \middle\| \hat{p}_*(\mathbf{y}, \boldsymbol{x})\right) \leq \varepsilon$, *while* $\mathrm{KL}\left(p_*(\mathbf{y}|\boldsymbol{x}) \middle\| \hat{p}_*(\mathbf{y}|\boldsymbol{x})\right) > M^2 \cdot \|\boldsymbol{x}_{(1:d_{k+1})}\|^2$.

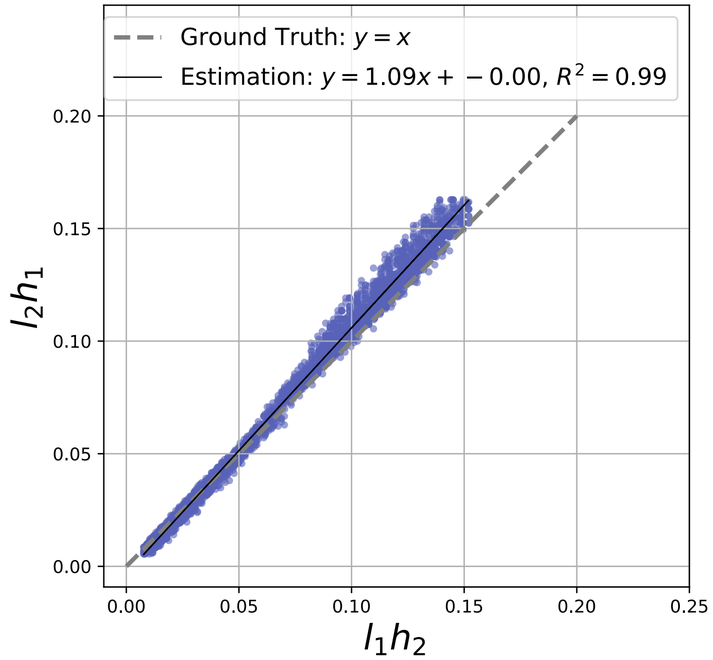
- Single diffusion model can well capture the joint distribution, while may be failed in learning the conditional dependency between different parts in the image.
- A simple case is $x = [x_1, x_2]$, $x_1 \sim N(0,1)$, $x_2 = x_1 + N(0,\epsilon^2)$
  - The joint distribution x has very large condition number (covariance matrix is kind of ill-conditioned).
  - The distribution $p(x_1)$ and $p(x_2 \,|\, x_1)$ has small condition numbers.

Numerical Experiments

- **Task 1:** generating the samples that satisfy the physical rule of straight-line light propagation.
  - Patches with size (16*16) **ensure** correlated features (sun and shadow) are sequentially discovered via AR.

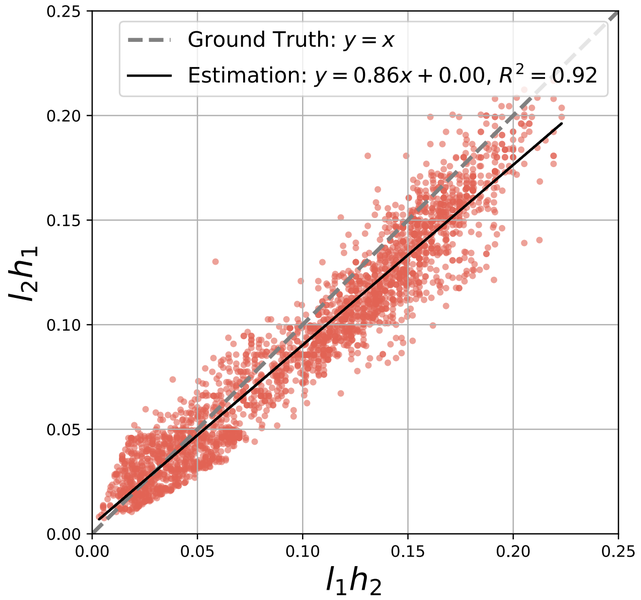


Task

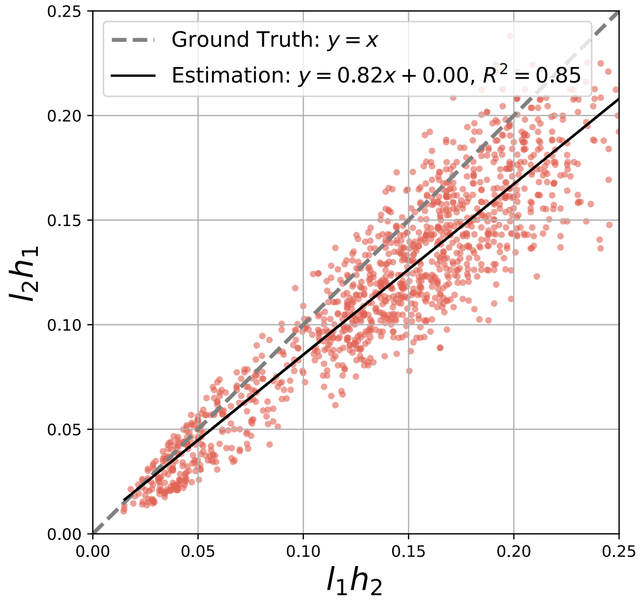


Training data

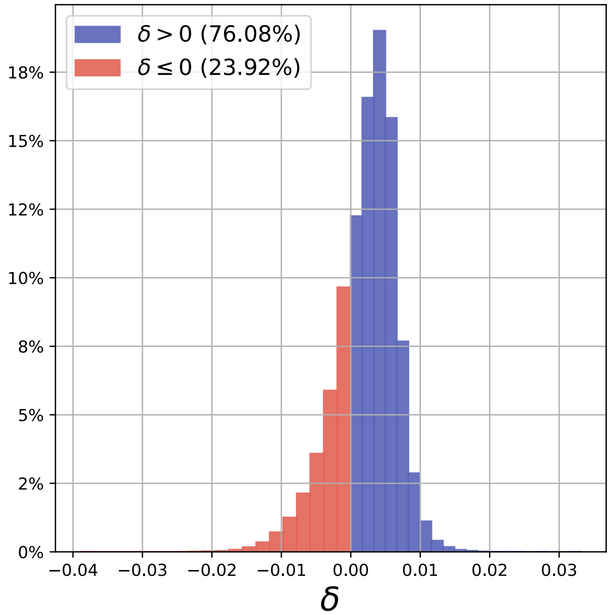
Autoregressive Diffusion Model: Conditional Dependency



AR DM



DM

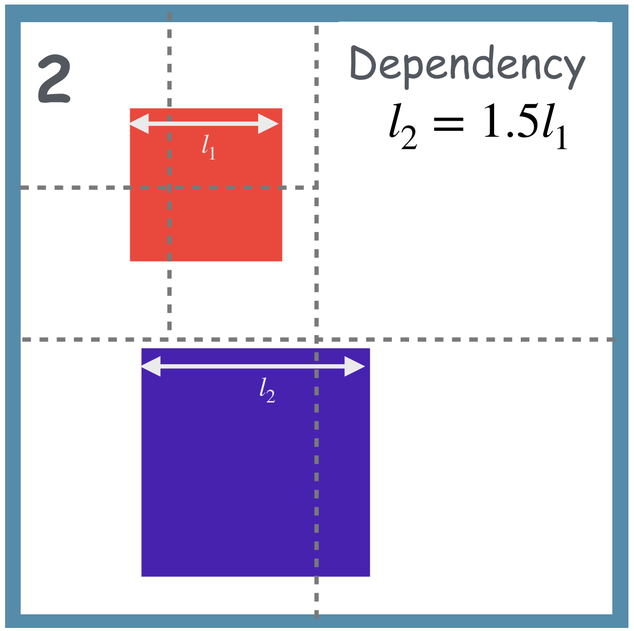


$$\delta = L_{dm} - L_{ar}$$

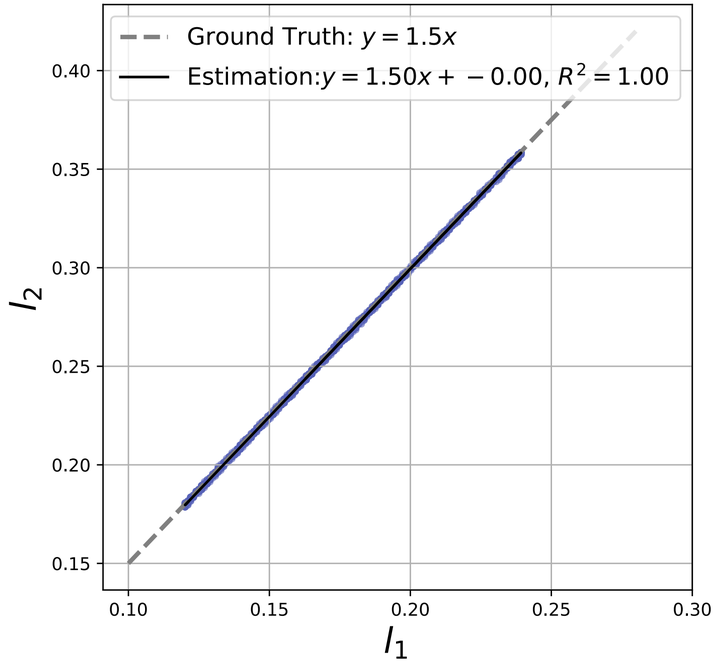
- When the dependency appear among different patches of the data, AR DM can better capture the inter-patch dependencies, and achieves lower training loss.
- This aligns with the theoretical results on conditional dependency.

Numerical Experiments

- **Task 1:** generating the samples that satisfy the constraints between the lengths of two square (32*32).
  - Small patch size (8*8) **disrupts** these dependencies by segmenting the squares into patches.

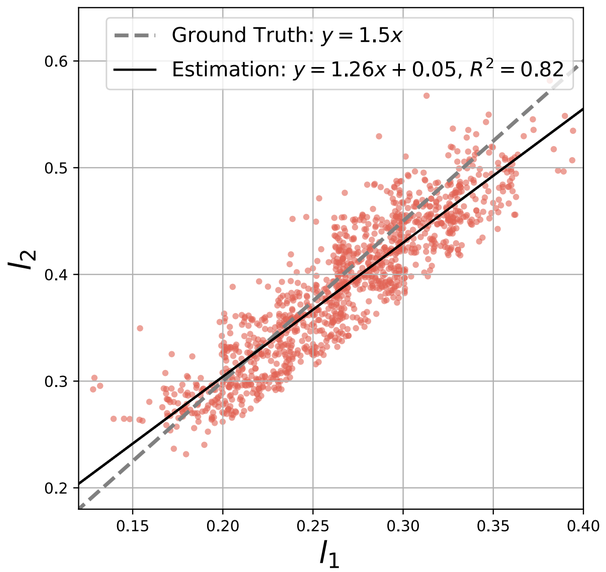


Task

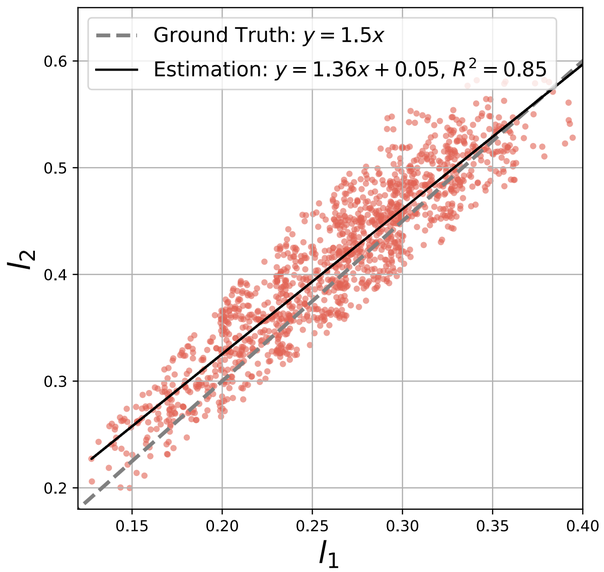


Training data

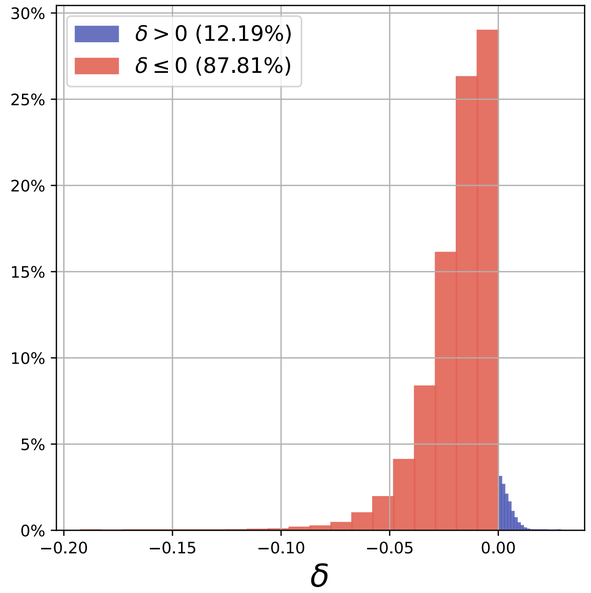
Numerical Experiments: Task 1



AR DM



DM



$$\delta = L_{dm} - L_{ar}$$

- When the dependency does not appear among different patches of the data, AR DM can not capture the inter-patch dependencies, and achieves substantially larger training loss.
- This aligns with the theoretical results in general setting.

Takeaway

- AR diffusion model and Standard diffusion model
  - We developed an RTK framework, which can be used to develop a family of diffusion inference algorithms.
  - We developed the sampling theory for AR diffusion model, which shows that:
    - AR diffusion model behaves worse than standard DM, in general settings.
    - AR diffusion model can better capture inter-patch dependencies than standard DM.