Autoregressive Diffusion Models

ICLR 2022

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, Tim Salimans

Introduction

- Deep generative models have made great progress in modelling different sources of data, such as images, text and audio.
- Two of the popular types are Autoregressive Models and Diffusion Models.

Autoregressive Models

- require a **pre-specified order** in which to generate data.
- although order can be retrieved with a single neural network call, sampling from a model requires **the same number of network calls as the dimensionality** of the data.

Diffusion Models

- **A large number of samples** are required for good performance.

This paper introduces Autoregressive Diffusion Models (ARDMs) that learns to generate in any order.

- In contrast to standard ARMs, they impose no constraints on the model architecture.
- Further, ARDMs require significantly fewer steps than diffusion models to attain the same performance.

This paper introduces Autoregressive Diffusion Models (ARDMs) that learns to generate in any order.

And additionally proposed two methods: **Parallelized** to speed up and **Upscaling** to increase performance.

Order Agnostic Autoregressive Diffusion Models

OA-ARDMs

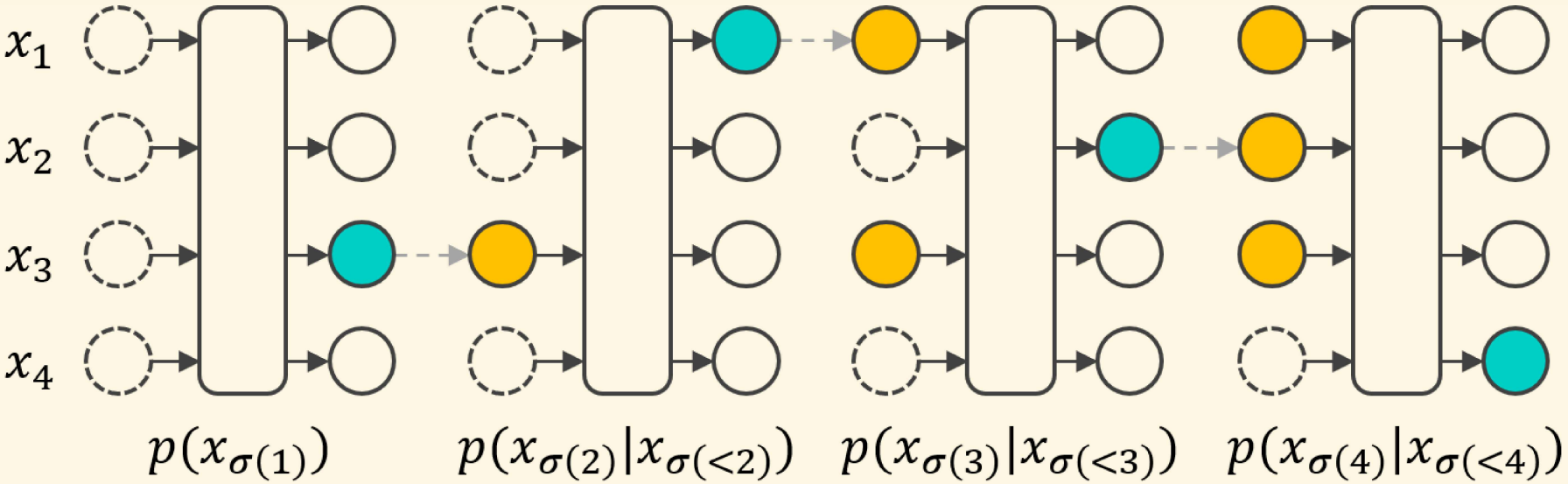
Based on

- **Order Agnostic** Autoregressive Models
- **Discrete Diffusion** Models
  Discrete Input, Discrete Output

Symbol Table

Symbol	Meaning	e.g.
D	Dimensions of input data	$\mathrm{height} \times \mathrm{width} \times \mathrm{channel}$
S_D	a set of all permutations of the integers $1, \ldots, D$	$\lvert S_D \rvert = !D$
K	number of categories	Quantize each pixel into 256 classes

OA-ARDMs (cont.)



Sampling from OA-ARDMs

Input: Network f

Output: Sample x

Initialize $x = \mathbf{0}$

Sample $\sigma \sim \mathcal{U}(S_D)$

for t in $\{1, \ldots, D\}$ **do**

$m \leftarrow (\sigma < t)$ and $n \leftarrow (\sigma = t)$

$x' \sim \mathcal{C}(x | f(m \odot x))$

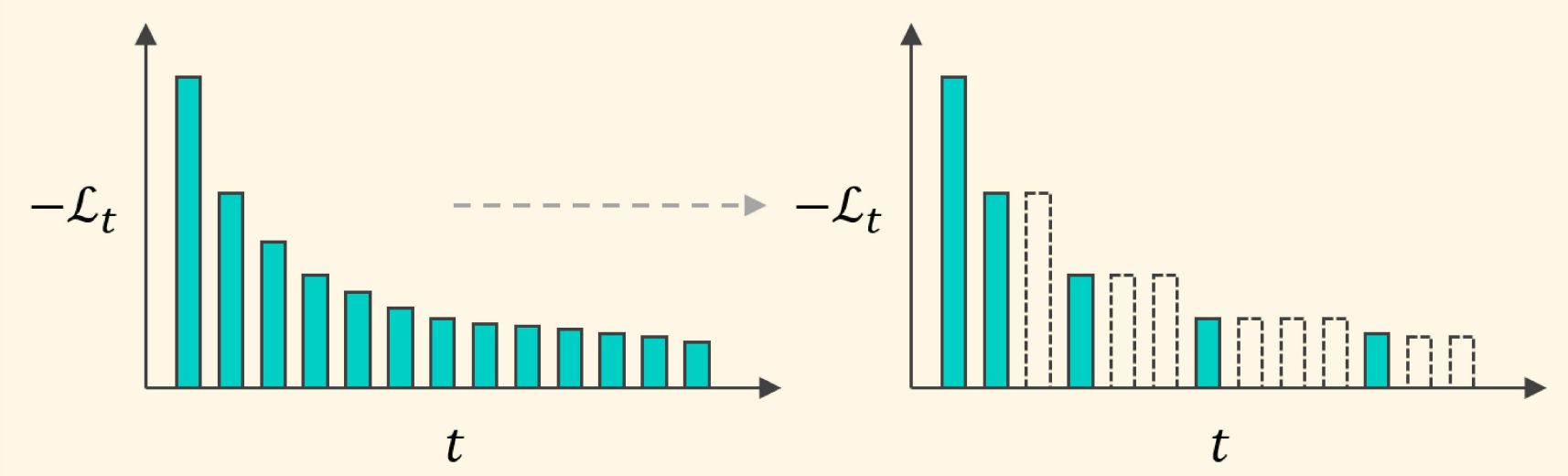
$x \leftarrow (1 - n) \odot x + n \odot x'$

Optimizing OA-ARDMs

$$
\begin{aligned}
&\textbf{Input: } \text{Datapoint } x, \text{ Network } f \\
&\textbf{Output: } \text{ELBO } \mathcal{L} \\
&\text{Sample } t \sim \mathcal{U}(1, \ldots, D) \\
&\text{Sample } \sigma \sim \mathcal{U}(S_D) \\
&\text{Compute } m \leftarrow (\sigma < t) \\
&l \leftarrow (1 - m) \odot \log \mathcal{C}(x | f(m \odot x)) \\
&\mathcal{L}_t \leftarrow \frac{1}{D - t + 1} \text{sum}(l) \\
&\mathcal{L} \leftarrow D \cdot \mathcal{L}_t
\end{aligned}
$$

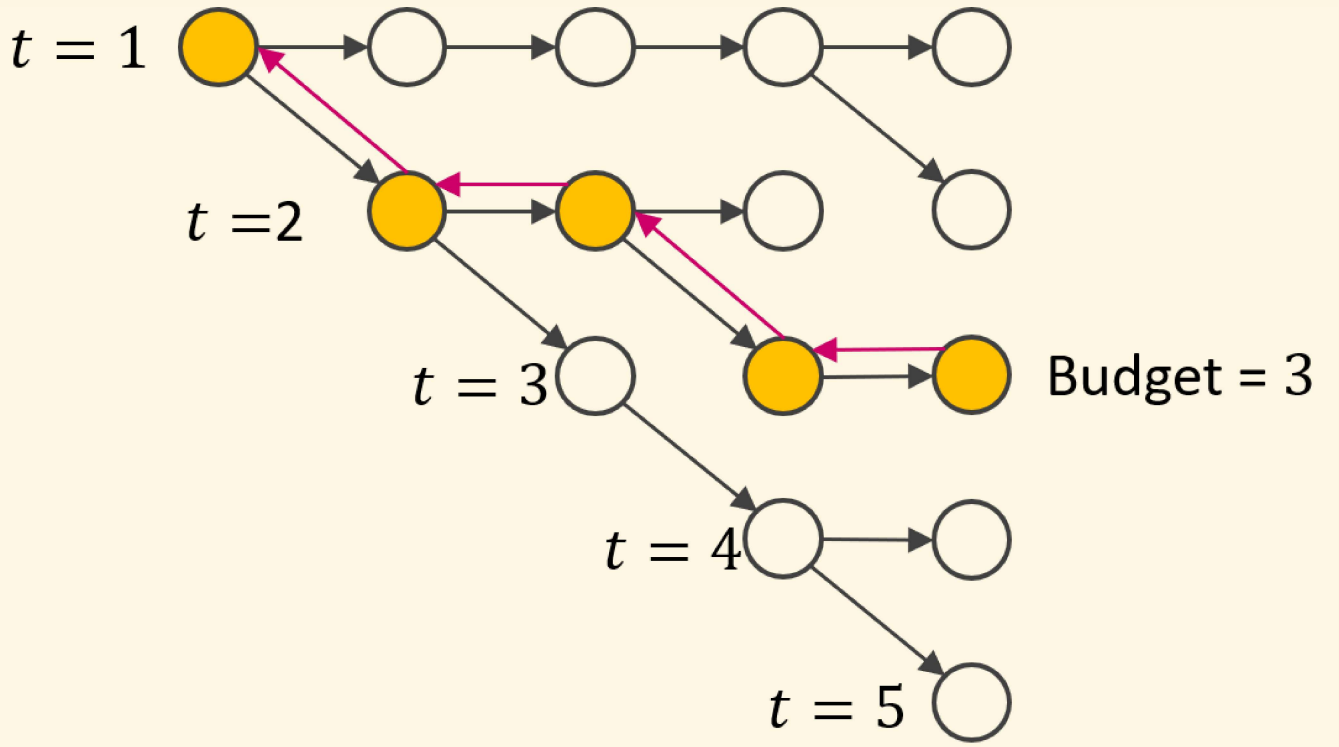
- Faster – **Parallelized**
  - Generate multiple outputs in one step to speed up.
  - Loss Components + Dynamic Programming.
- Higher Quality – **Upscaling**
  - Multi Stage.
  - Low Resolution Quantization $\rightarrow$ High Resolution Quantization.

Parallelized



Parallelized (cont.)

Find the lowest cost steps within a limited budget with Dynamic Programming.



Parallelized (cont.)

$$\text{Cost Table } \hat{L} = \begin{bmatrix} 0 & \infty & \infty & \cdots & \infty \\ \infty & -\mathcal{L}_1 & -\mathcal{L}_1 & \cdots & -\mathcal{L}_1 \\ \infty & \infty & -\mathcal{L}_2 & \cdots & -\mathcal{L}_2 \\ \vdots & & & \ddots & \vdots \\ \infty & \infty & \cdots & \infty & -\mathcal{L}_D \end{bmatrix} \in \mathbb{R}^{(D+1)\times(D+1)}$$

Parallelized (cont.)

Transition Table

Input: Cost Table $\hat{L} \in \mathbb{R}^{(D+1)\times(D+1)}$

Output: Transition Table $T \in \mathbb{R}^{(D+1)\times(D+1)}$

Initialize $T = \mathbf{0}$ and $L = \hat{L}$

for t in $\{1, \ldots, D\}$ **do**

  for k in $\{t, \ldots, D\}$ **do**

    if $L_{t-1,k-1} < L_{t-1,k}$ **do**

      $L_{t,k} = \hat{L}_{t,k} + L_{t-1,k-1}$ and $T_{t,k} = t - 1$

    else do

      $L_{t,k} = \hat{L}_{t,k} + L_{t,k-1}$ and $T_{t,k} = t$

Parallelized (cont.)

Transition Path

Input: Transition Table $T \in \mathbb{R}^{(D+1)\times(D+1)}$

Budget of Steps N

Output: Transition Path $Q \in \mathbb{R}^{N}$

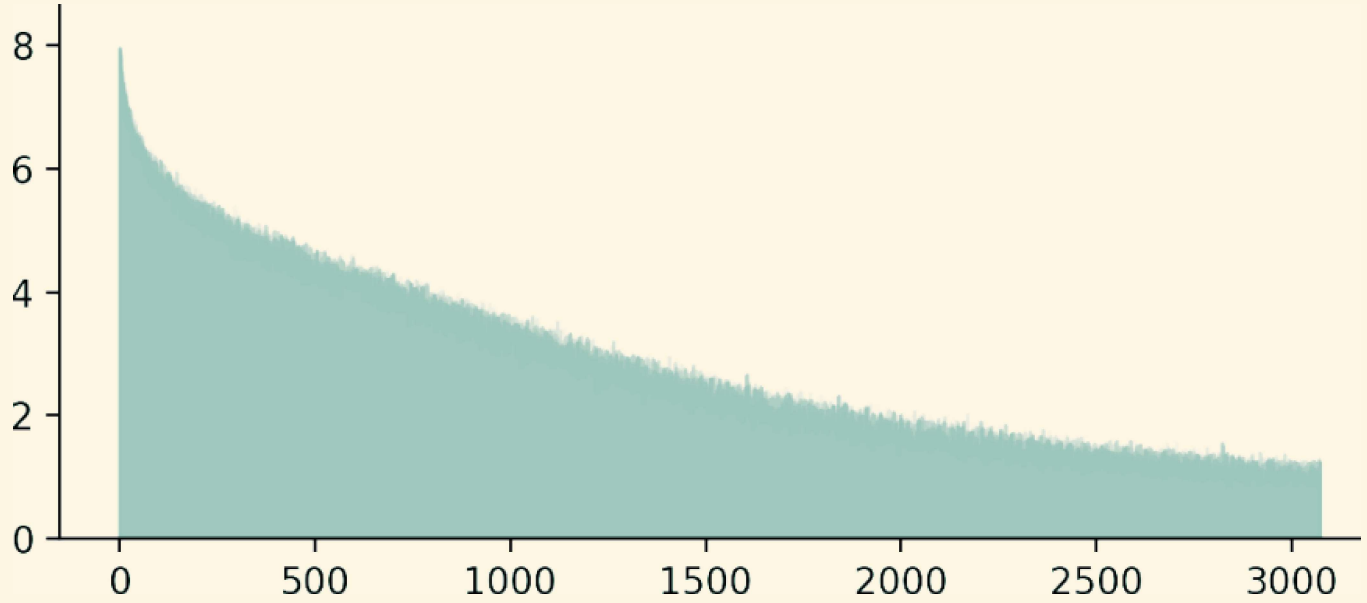
Initialize $k = N$

for t in $\{D, \ldots, 1\}$ **do**

$Q_{t-1} = k$

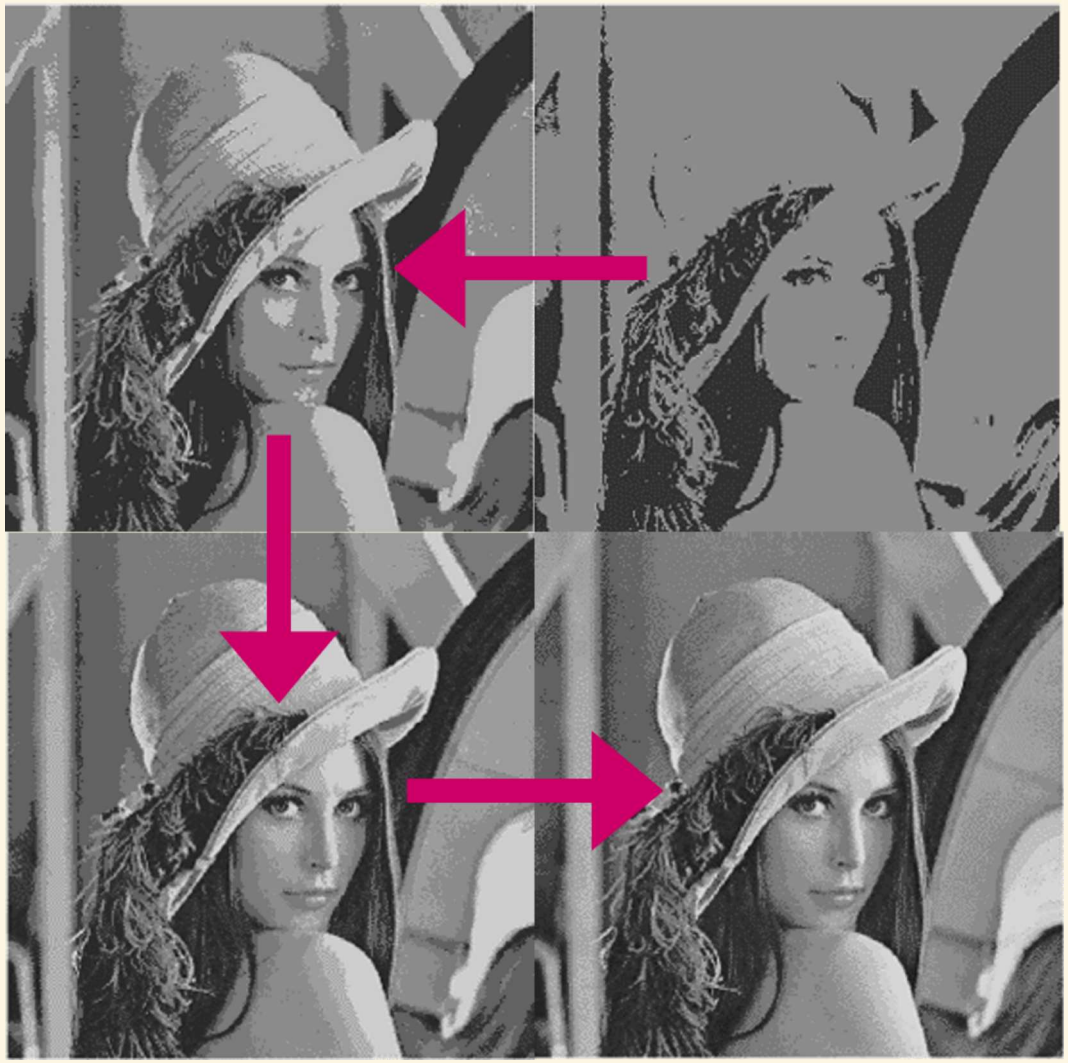
$k = T_{k,t}$

Parallelized (cont.)



The $-\mathcal{L}_t$ used by Cost Table $\hat{L}$ is the result of statistics from the training samples.

Upscaling



Upscaling (cont.)

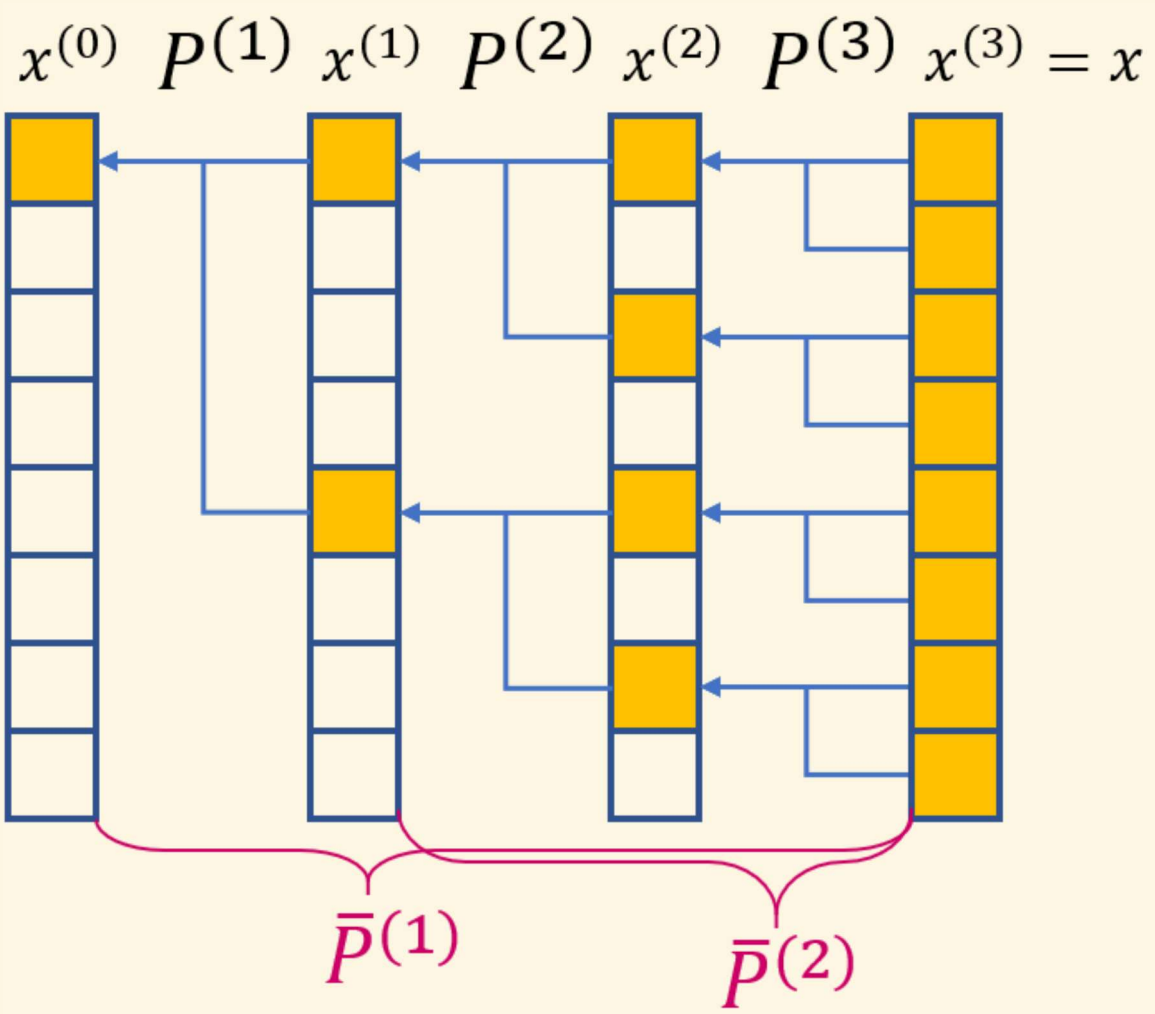
- Low Resolution Quantization $\rightarrow$ High Resolution Quantization.
- The amount of computation becomes $S = \lceil \log_b(K) \rceil$ times, b is upscaling level.

Upscaling (cont.)

Reduce the quantization resolution by using the transition matrix $\bar{P}_s$.

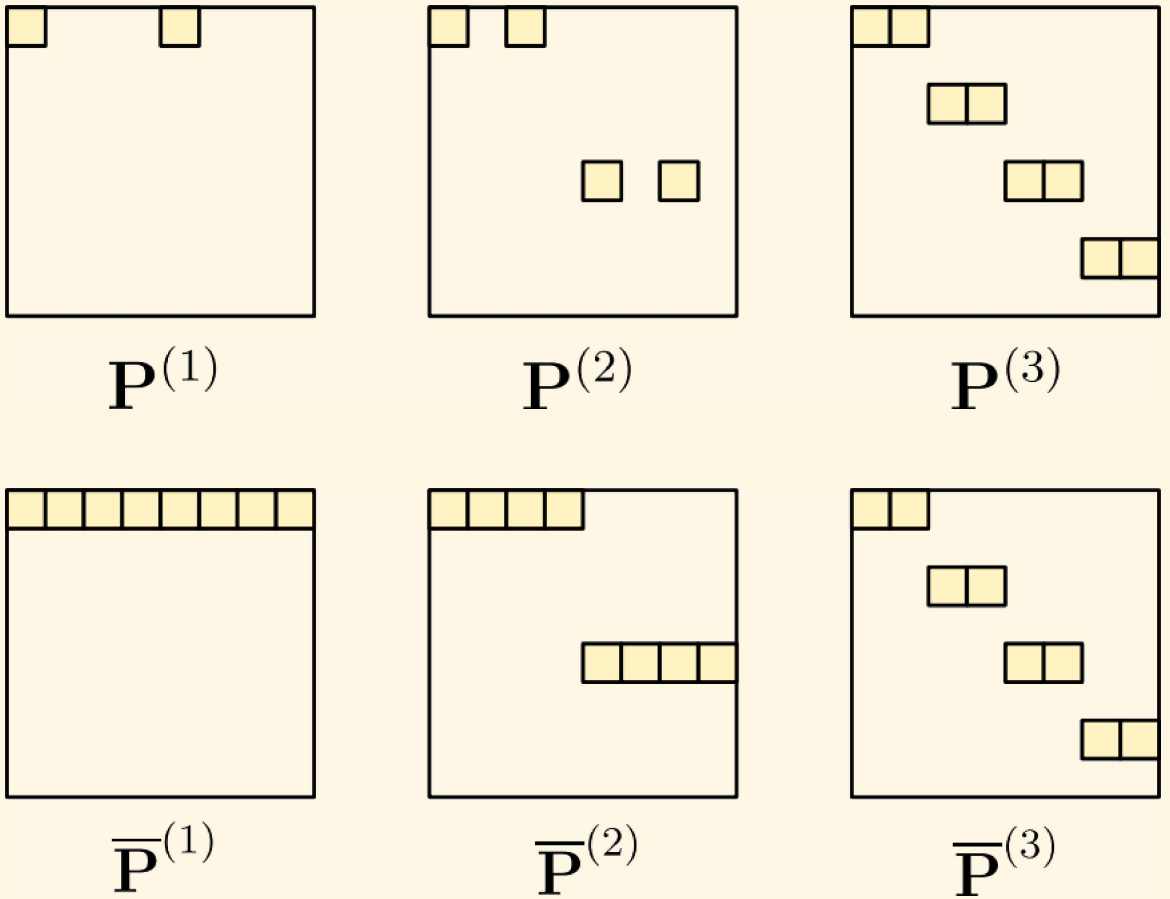
$$
\begin{aligned}
x^{(s-1)} &= P^{(s)} x^{(s)} \\
&= P^{(s)} P^{(s+1)} x^{(s+1)} \\
&= P^{(s)} P^{(s+1)} \dots P^{(S)} x^{(S)} \\
&= P^{(s)} P^{(s+1)} \dots P^{(S)} x \\
&= \bar{P}^{(s)} x
\end{aligned}
$$

Upscaling (cont.)



Upscaling (cont.)

Visualization P and $\bar{P}$



Sampling from Upscale OA-ARDMs

Input: Network f

Output: Sample x

Initialize $x = x^{(0)}$

for s in $\{1, \dots, S\}$ **do**

  Sample $\sigma \sim \mathcal{U}(S_D)$

  for t in $\{1, \dots, D\}$ **do**

    $m \leftarrow (\sigma < t)$ and $n \leftarrow (\sigma = t)$

    $\theta^{(s)} \propto P^{(s)\intercal} x \odot \bar{P}^{(s+1)} f(x, m, s, t)$

    $x' \sim \mathcal{C}(x^{(s)} | \theta^{(s)})$

    $x \leftarrow (1 - n) \odot x + n \odot x'$

Optimizing Upscale OA-ARDMs

Input: Datapoint x, Network f

Output: ELBO $\mathcal{L}$

Sample $s \sim \mathcal{U}(1, \ldots, S)$

Sample $t \sim \mathcal{U}(1, \ldots, D)$ and $\sigma \sim \mathcal{U}(S_D)$

$x^{(s)} \leftarrow \bar{P}^{(s+1)} x$ and $x^{(s-1)} \leftarrow \bar{P}^{(s)} x$

Compute $m \leftarrow (\sigma < t)$

$i \leftarrow m \odot x^{(s)} + (1 - m) \odot x^{(s-1)}$

$\theta^{(s)} \propto P^{(s)\intercal} x^{(s-1)} \odot \bar{P}^{(s+1)} f(i, m, s, t)$

$l \leftarrow (1 - m) \odot \log \mathcal{C}(x^{(s)} | \theta^{(s)})$

$\mathcal{L}_t \leftarrow \frac{1}{D - t + 1} \text{sum}(l)$ and $\mathcal{L} \leftarrow D \cdot \mathcal{L}_t$

Main Results

text8

Model	Steps	NLL (bits/char)(↓)
Discrete Flow (8 × 3 layers)	-	1.23
Argmax Coupling Flow	-	1.80
IAF / SCF	-	1.88
Multinomial Diffusion (D3PM uniform)	1000	1.72
OA-Transformer	250	1.64
D3PM-uniform	1000	1.61 ±0.020
D3PM-absorbing	1000	1.45 ±0.020
D3PM-absorbing	256	1.47
OA-ARDM (ours)	250	1.43 ±0.001
Transformer decoder	256	1.18
Transformer XL	256	1.08
D3PM-absorbing	20	1.56 ±0.001
Parallelized OA-ARDM (ours)	20	1.51 ±0.001

CIFAR-10, k=256

Model	Steps	NLL (bits/dimension)(↓)
ARDM-OA	3072	2.69 ±0.005
Parallel ARDM-OA	50	2.74
ARDM-Upscale 4	4×3072	2.64 ±0.002
Parallel ARDM-Upscale 4	4×50	2.68
D3PM Absorbing	1000	4.40
D3PM Gaussian	1000	3.44 ±0.007

CIFAR-10 Upscale

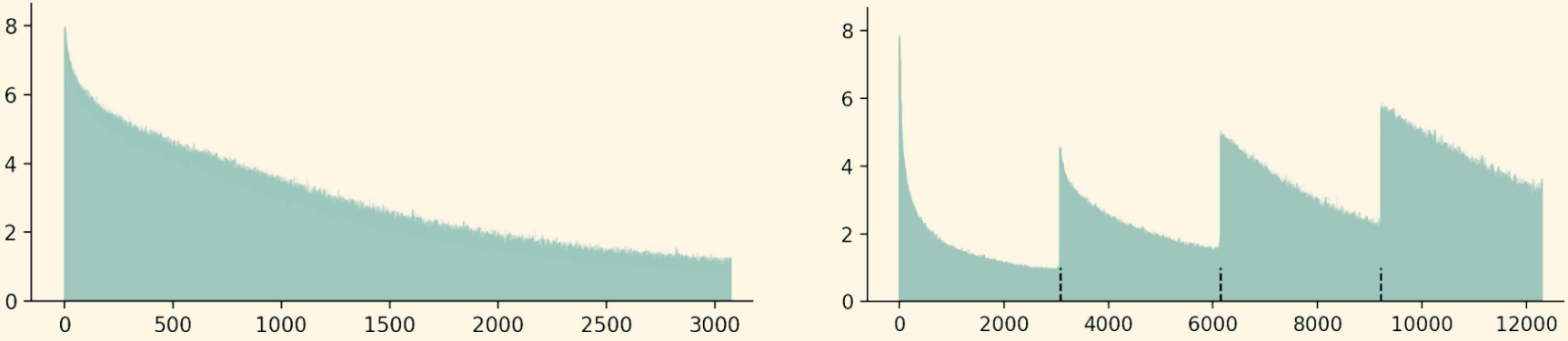
Model	Steps	Performance
OA-ARDM	$D = 3072$	2.69
ARDM Upscale 16	$2 \times D$	2.67
ARDM Upscale 4	$4 \times D$	**2.64**
ARDM Upscale 2	$8 \times D$	2.67

SC09-10 Upscale, k=65536

Model	Steps	Performance
OA-ARDM	$D = 16000$	7.93
ARDM Upscale 256	$2 \times D$	6.36
ARDM Upscale 16	$4 \times D$	6.30
ARDM Upscale 4	$8 \times D$	**6.29**
ARDM Upscale 2	$16 \times D$	**6.29**

Parallelized (cont.)

The Loss components $-\mathcal{L}_t$ over model step on CIFAR-10.



- Left: loss terms for the OA-ARDM.
- Right: loss terms for the ARDM-Upscale 4, which comprises four stages.

Conclusion

- You can choose to have faster or higher quality.
- Achieved SOTA in order-agnostic approaches
- On text
  - Still not as good as the single-order autoregressive models.
  - Using the upscaling method failed to achieve an increase in quality.

在沒有給定初始條件情況下，生成順序並不重要。
但如果已有給定的起始條件，不考慮生成順序感覺是不合理的。

→ 利用 TD error 來學習適合的生成順序？

Thanks for your attention.

Q&A