Generative models for discrete random variables

Rianne van den Berg, principal researcher @Microsoft Research Amsterdam

Previously at Google Brain & University of Amsterdam

Outline

Motivation for generative modeling for discrete random variables: lossless compression

- Basics of lossless compression
- Connecting likelihood-based generative models and lossless compression

Normalizing flows

Denoising diffusion models

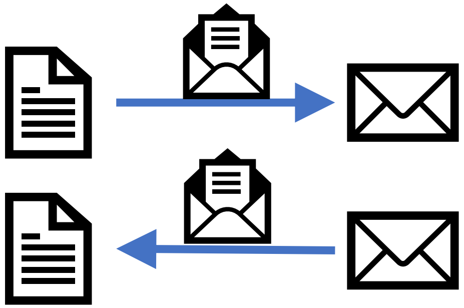
Autoregressive models

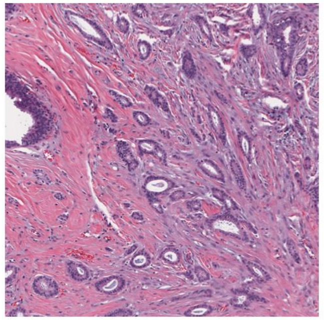
Compression

Message: object we'd like to compress. Files, messages, …

Encoding : message → compressed representation

Decoding: compressed representation → message





Lossless compression vs lossy compression.

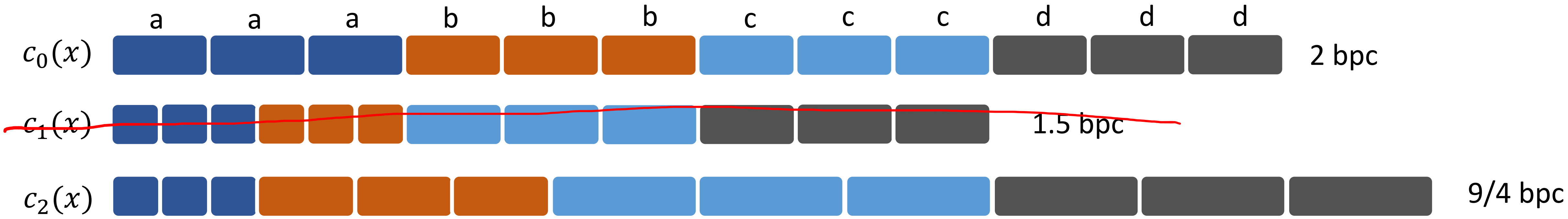
For lossless compression:

- Message must be perfectly reconstructed by decoding algorithm.
- Compressed representation must be uniquely decodable.
- *on average* the compressed representation will be shorter than the message.

Compressing messages

Shorter code length for some messages will necessarily lead to longer code lengths for others!

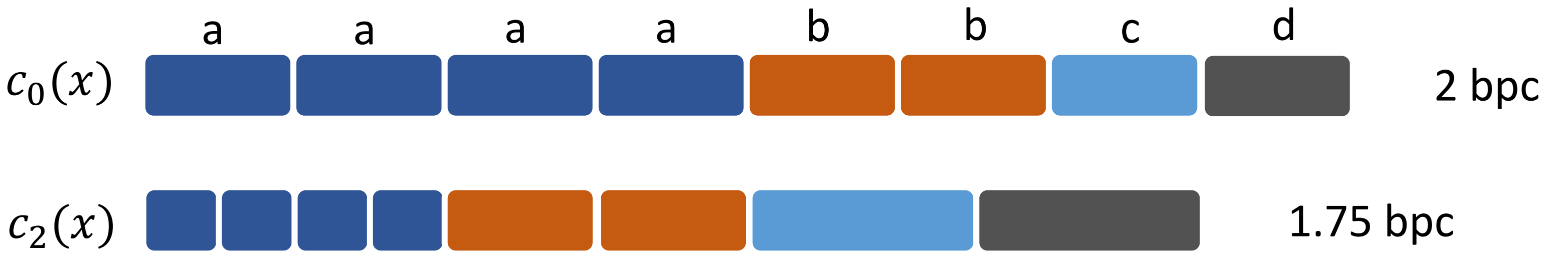
x	$p(x)$	$c_0(x)$	~~$c_1(x)$~~	$c_2(x)$
a	$^1/_4$	00	~~0~~	0
b	$^1/_4$	01	~~1~~	10
c	$^1/_4$	10	~~01~~	110
d	$^1/_4$	11	~~10~~	111



Compressing messages

However…. Trading off shorter and longer code lengths for different messages can be beneficial if not all messages occur with the same probability!

x	$p(x)$	$c_0(x)$	$c_2(x)$
a	$^1/_2$	00	0
b	$^1/_4$	01	10
c	$^1/_8$	10	110
d	$^1/_8$	11	111



Self-information of a message

How much information is contained in a message?

Shannon's definition: $h(x) =$ $\log_2 \frac{1}{p(x)}$

1. Info of two independent messages adds up:

$$h(xy) = \log\frac{1}{p(x,y)} = \log\frac{1}{p(x)p(y)} = \log\frac{1}{p(x)} + \log\frac{1}{p(y)}$$

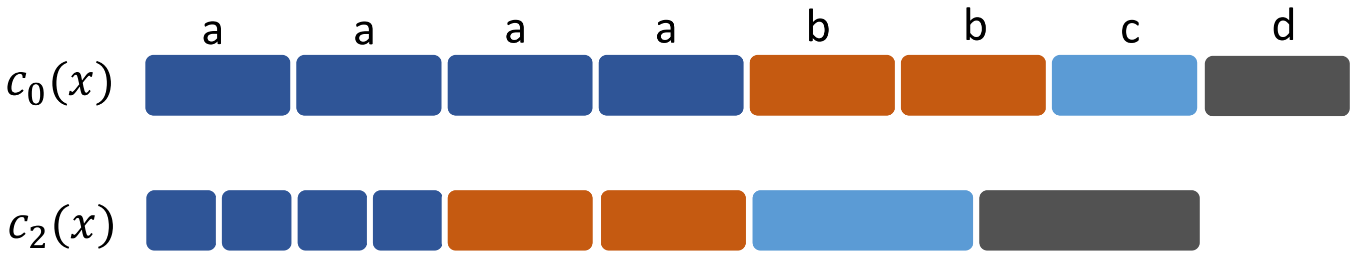
2. Messages with a large probability contain less information!

Example: Guess a particular day on which an event occurred in NL.
Not so informative message: It rained on that day.

Compressing messages: codelengths

However…. Trading off shorter and longer code lengths for different messages can be beneficial if not all messages occur with the same probability!

x	$p(x)$	$c_0(x)$	$c_2(x)$	$\log 1/p(x)$	$l_2(c(x))$
a	$^1/_2$	00	0	1.0	1
b	$^1/_4$	01	10	2.0	2
c	$^1/_8$	10	110	3.0	3
d	$^1/_8$	11	111	3.0	3



Shannon's source coding theorem

For data generated according to $x \sim p(x)$, what is the best average code length per symbol x?

$$l(C, X) = \sum_x p(x) l(c(x)) \geq \sum_x p(x) \underbrace{\log \frac{1}{p(x)}}_{\text{self-information}} = \mathbb{H}_p[X]$$

Source coding theorem: there exists a uniquely decodable code C for $X \sim p(X)$ such that

$$\mathbb{H}_p[X] \leq l_a(C) \leq \mathbb{H}_p[X] + 1$$

Resources/further reading

- Information theory, inference and learning algorithms. David MacKay
- Introduction to data compression, Guy Blelloch, Carnegie Mellon University. http://www.cs.cmu.edu/~guyb/realworld/compression.pdf
- CS294-158 course on deep unsupervised learning. L10 compression. Berkeley. Peter Abeel. https://www.youtube.com/watch?v=pPyOlGvWoXA

Outline

- Motivation for generative modeling for discrete random variables: lossless compression
  - Basics of lossless compression
  - **Connecting likelihood-based generative models and lossless compression**
- Normalizing flows
- Denoising diffusion models
- Autoregressive models

Generative likelihood-based models

Given: Data $\{x_n\}_{n=1}^{N}$generated by sampling $x \sim p_{data}(x)$

↳ unknown

Task: take a deep density estimator $p_\theta(x)$ and optimize it such that

$$p_\theta(x) \approx p(x)$$

Q: what happens when we try to use $p_\theta(x)$ to encode data generated by $p(x)$?

Encoding with an approximate distribution

Assume we have access to a prefix code such that it produces close to optimal codes for $p_\theta(x)$.

Code length:

If data was generated $x \sim p_{data}(x)$:

$\sum_x p_{data}(x) \log_2 1/p_\theta(x)$ = $\sum_x p_{data}(x) \log \frac{1}{p_\theta(x)} + \sum_x p_{data}(x) \log \frac{1}{p_{data}(x)}$

$- \sum_x p_{data}(x) \log \frac{1}{p_{data}(x)} = \mathbb{H}_{p_{data}}[X] + \underbrace{KL[p_{data} || p_\theta]}_{\geq 0}$

$KL[p_{data}||p_\theta]$ should be made small to achieve optimal compression

Likelihood-based generative models

Given: Data $\{x_n\}_{n=1}^{N}$ generated by sampling $x \sim p(x)$

Task: optimize $p_\theta(x) \approx p(x)$

Objective: $\arg\min_\theta \frac{1}{N}\sum_i -\log_2 p_\theta(x_i) = \arg\min_\theta \frac{1}{N}\sum_i \log\frac{1}{p_\theta(x_i)}$

$\approx \arg\min_\theta \sum_x p_{data}(x)\log\frac{1}{p_\theta(x)} = \arg\min_\theta H_{p_{data}}(x) + KL[p_{data}\|p_\theta]$

$= \arg\min_\theta KL[p_{data}\|p_\theta]$

Optimizing a likelihood-based generative model ←→ getting an optimal compressor

Likelihood-based models as lossless compressors

Loss function: $\mathbb{E}_{x \sim p_{data}}[-\log_2 p_\theta(x)] \geq \mathbb{E}_{x \sim p_{data}}[-\log_2 p_{data}(x)]$

Minimum expected code length

1. Entropy coders are designed for discrete data
   - Somewhere in your model you need to truncate / discretize your random variables
   - Truncating/discretizing leads to a loss of information!
   - High-precision discretization leads to larger entropy → longer codes!
2. Entropy coders either need to tractably enumerate $p(x)$ for all x, or they need to be able to evaluate $cdf(x)$ for all x.
   - Especially in high-D, we often don't have access to a closed form for $cdf(x)$
   - One solution: break down of high-D coding problem into coding problems for 1D data

Entropy coding for high-dim data

Data $x \in \{0, 1, 2, \ldots, K\}^D$, distributed according to $p_{data}(x)$

Entropy coders:

- Either need to tractably enumerate $p(x)$ for all x (undoable for high dims)
- or they need to be able to evaluate $cdf(x)$ for all x (in general not available for arbitrary $p(x)$)

Break down into D x 1dim problems with a factorization assumption:

Independent dimensions: $p_\theta(x) = \prod_{i=1}^{D} p_\theta(x_i)$ → parallel

Autoregressive dependencies: $p_\theta(x) = p_\theta(x_1)\, p_\theta(x_2 \mid x_1) \ldots p_\theta(x_D \mid x_{D-1}, \ldots, x_1)$

→ sequential

Ignoring dependencies → longer optimal codes

Example: $p(x_2, x_1) = p(x_2|x_1)p(x_1)$

Optimal average code length:

$\mathbb{H}[X_2, X_1] = \sum_{x_1, x_2} p(x_2|x_1)p(x_1)\left[\log \frac{1}{p(x_2|x_1)} + \log \frac{1}{p(x_1)}\right]$

$= \mathbb{H}[X_2|X_1] + \mathbb{H}[X_1]$

Approximate as independent: $p(x_1, x_2) \approx p(x_2)p(x_1)$

$\mathbb{H}[X_2, X_1] = \mathbb{H}[X_2] + \mathbb{H}[X_1]$

Use $\mathbb{H}[X_2|X_1] \leq \mathbb{H}[X_2]$ → making an independence assumption can make your optimal average code length larger!

Outline

- Motivation for generative modeling for discrete random variables: lossless compression
  - Basics of lossless compression
  - Connecting likelihood-based generative models and lossless compression
- **Integer discrete normalizing flows for lossless compression**
- Denoising diffusion models
- Autoregressive models

Density estimation with normalizing flows

Rezende & Mohamed, 2016. Dinh et al., 2016.

Idea: Find an invertible function that maps data from a complicated distribution with dependencies, to a distribution that is “easy” to sample from and evaluate.

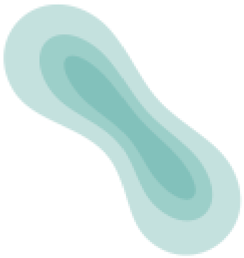
$p(z_2) = p_\theta(x)$

$z_1 = \left(f^{(2)}\right)^{-1}(z_2)$

$\longrightarrow$

$\longleftarrow$

$z_2 = f^{(2)}(z_1)$



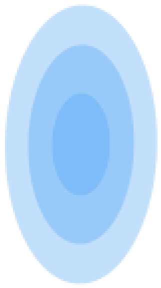
$p(z_1)$

$z_0 = \left(f^{(1)}\right)^{-1}(z_1)$

$\longrightarrow$

$\longleftarrow$

$z_1 = f^{(1)}(z_0)$

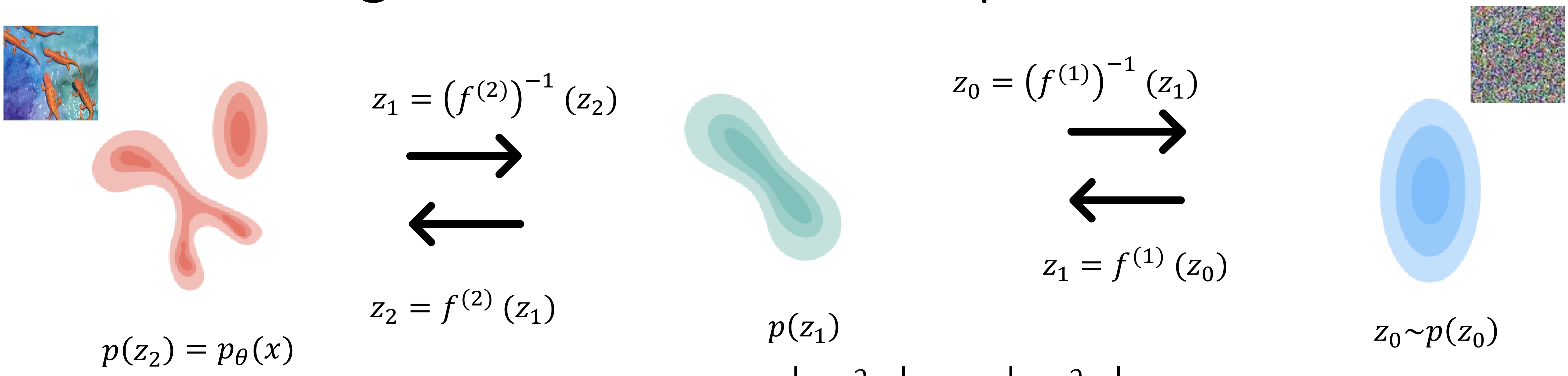


$z_0 \sim p(z_0)$

$$p_\theta(x) = p_{Z_2}(z_2) = p_{Z_0}(z_0)\,\left|\det\frac{\partial z_0}{\partial z_1}\right|\left|\det\frac{\partial z_1}{\partial z_2}\right|$$

$$\log p_\theta(x) = \log p_{z_0}(z_0) + \log\left|\det\frac{\partial z_0}{\partial z_1}\right| + \log\left|\det\frac{\partial z_1}{\partial z_2}\right|$$

Normalizing flows as source compressors



$$\log p_\theta(x) = \log p_{Z_2}(z_2) + \log\left|\det \frac{\partial z_0}{\partial z_1}\right| + \log\left|\det \frac{\partial z_1}{\partial z_2}\right|$$

$$p(z_0) = \prod_{i=1}^{D} p(z_{0,i})$$

- If the base distribution $p(z_0)$ is independent across dims: potentially easy compression! (turning D-dim coding problem into D 1-dim coding problems)
- However: normalizing flows were designed for continuous random variables…

Normalizing flows for integer valued data

Integer discrete flows for lossless compression, E Hoogeboom, J Peters, Rianne vd Berg, M Welling, NeurIPS 2019

Goal: define invertible $f\colon \mathbb{Z}^D \mapsto \mathbb{Z}^D$

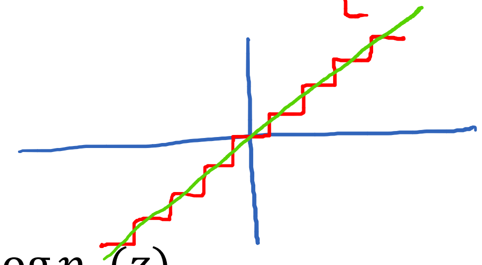
Simple solution: Take RealNVP [Dinh et al. ICLR 2017] and adjust to integers

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \to \begin{bmatrix} x_1 \\ \cancel{s_\theta(x_1)} \odot x_2 + \lfloor t_\theta(x_1) \rceil \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = z$$

round

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_1 \\ (z_2 - \lfloor t_\theta(z_1) \rceil) \,/\, \cancel{s_\theta(z_1)} \end{bmatrix} \leftarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = z$$

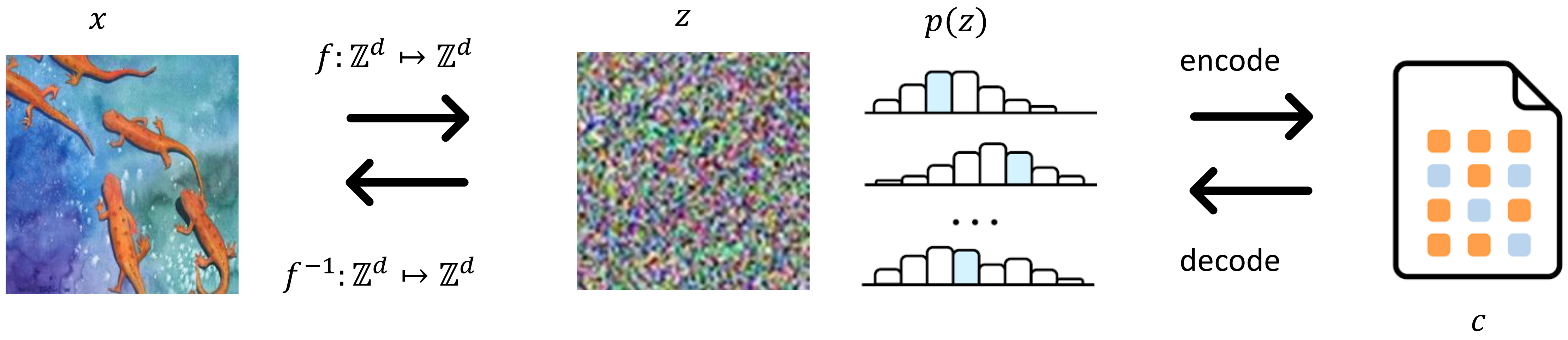
Gradients through rounding:



straight-through estimator

Data likelihood: $\log p_\theta(x) = \log p_Z(z)$

Lossless compression with integer discrete flows



Data likelihood: $\log p_\theta(x) = \log p_Z(z)$

High-probability z → short code
Low-probability z → long code

- Likelihood model for discrete random variables: can directly be used by entropy coders.
- The base distribution $p(z_0)$ is independent across dims: turned D-dim coding problem into D 1-dim coding problems!

Integer discrete flows and lossless compression. Emiel Hoogeboom*, Jorn Peters*, Rianne van den Berg, Max Welling, NeurIPS 2019

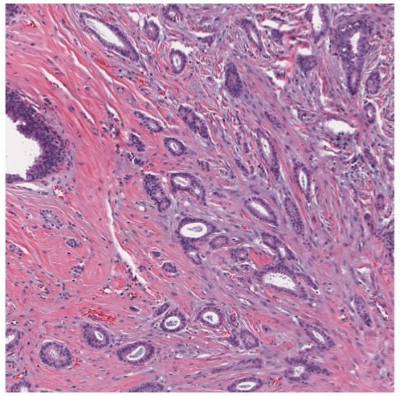
IDF++: Analyzing and improving Integer Discrete Flows for lossless compression. Rianne van den Berg, Alexey Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, Tim Salimans, ICLR 2021

Results: IDF & IDF ++

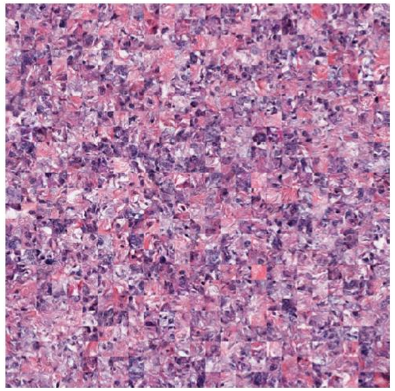
Dataset	IDF	JP2-WSI	FLIF [34]	JPEG2000
Histology	**2.42 (3.19×)**	3.04 (2.63×)	4.00 (2.00×)	4.26 (1.88×)

Compression models	CIFAR-10	IMAGENET-32	IMAGENET-64
PNG (Boutell & Lane (1997))	5.87*	6.39*	5.71*
JPEG-2000 (Rabbani (2002))	5.20†	6.48†	5.10†
FLIF (Sneyers & Wuille (2016))	4.19*	4.52*	4.19*
BIT-SWAP (Kingma et al. (2019))	3.82 (3.78)	4.50 (4.48)	-
HILLOC (Townsend et al. (2019a))	3.56 (3.55)	4.20 (4.18)	3.90 (3.89)
LBB (Ho et al. (2019b))	3.12 (3.12)	3.88 (3.87)	3.70 (3.70)
SREC (Cao et al. (2020))	-	-	4.29
IDF (Hoogeboom et al. (2019a))	3.32 (3.30)**	4.18 (4.15)	3.90 (3.90)
IDF++, SMALL: 4 FLOWS PER LEVEL	3.31 (3.29)	4.16 (4.14)	3.85 (3.85)
IDF++	3.26 (3.24)	4.12 (4.10)	3.81 (3.81)

Resolution: 2000 x 2000 pixels



Samples patched: 80 x 80 pixels



Integer discrete flows for lossless compression, E Hoogeboom, J Peters, Rianne vd Berg, M Welling, NeurIPS 2019

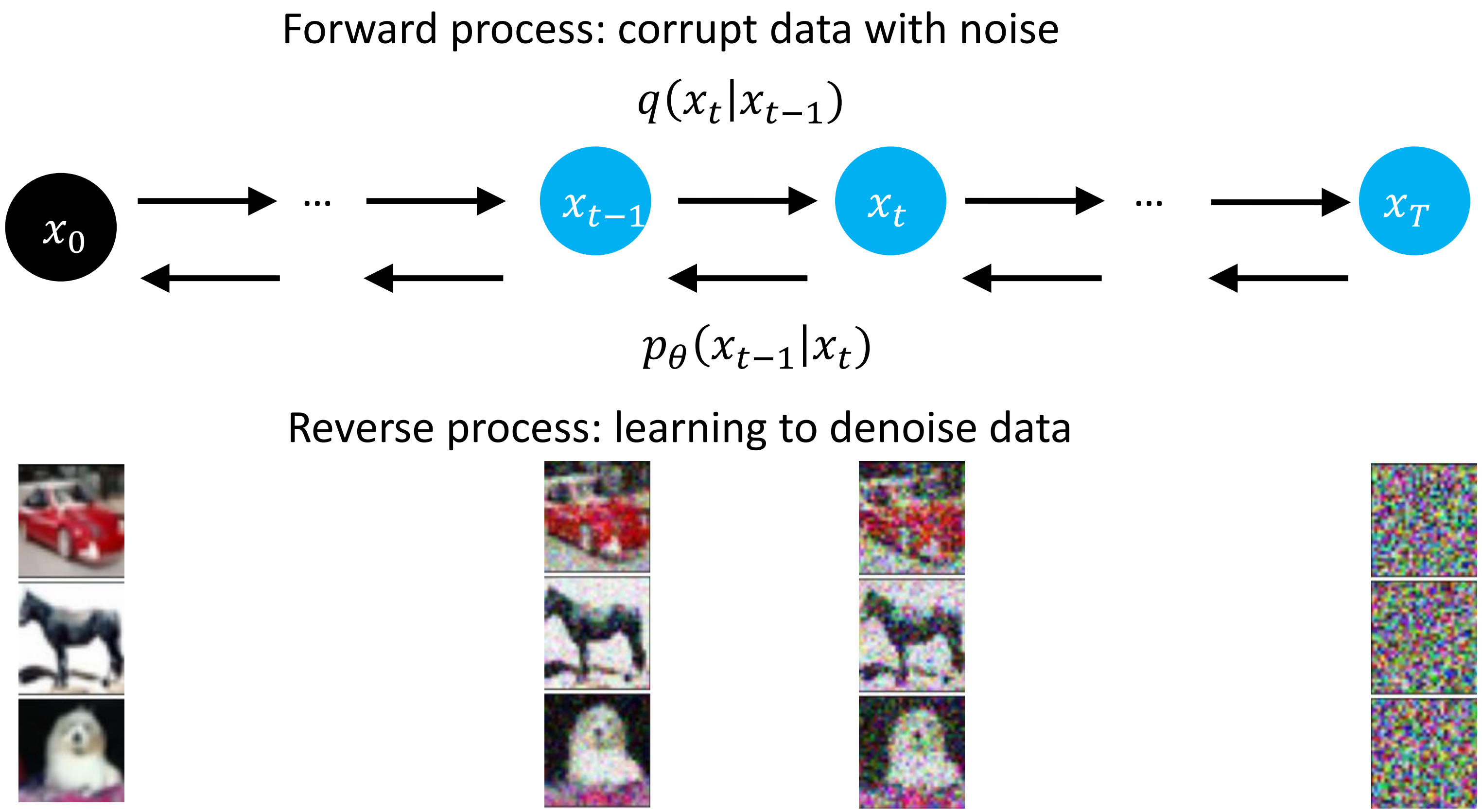
IDF++: Analyzing and improving Integer Discrete Flows for lossless compression.
Rianne vd Berg, Alexey Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, Tim Salimans, ICLR 2021

Outline

- Motivation for generative modeling for discrete random variables: lossless compression
  - Basics of lossless compression
  - Connecting likelihood-based generative models and lossless compression
- Integer discrete normalizing flows for lossless compression
- **Denoising diffusion models**
- Autoregressive models

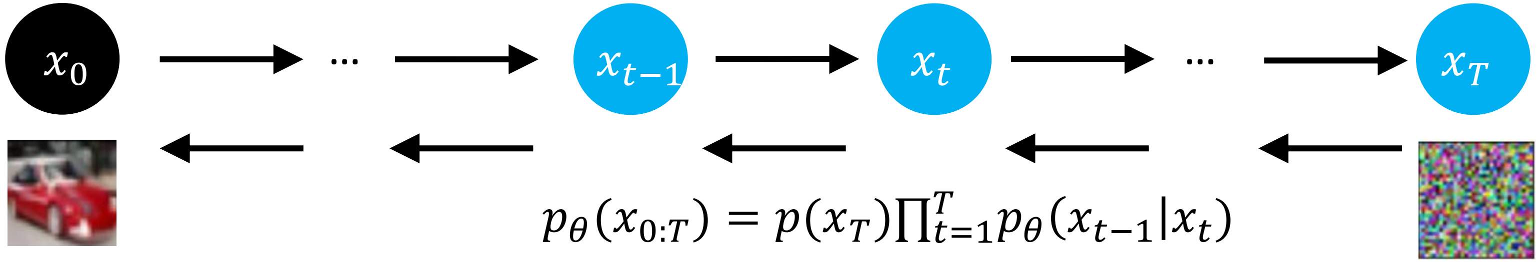
Denoising diffusion probabilistic models

Sohl-Dickstein et al., ICML 2015, Ho et al., NeurIPS 2020, Song et al., ICLR 2021



Training diffusion models

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$



$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$L_{vb} = \mathbb{E}_q(x_0)[D_{KL}[q(x_T|x_0)||p(x_T)] + \sum_{t=2}^{T} \mathbb{E}_{q(x_t|x_0)}[D_{KL}[q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)]] - \mathbb{E}_{q(x_1|x_0)}[\log p_\theta(x_0|x_1)]]$$

Practical requirements for $q(x_t|x_{t-1})$ to allow for efficient training of p_θ:

1. Efficient sampling of x_t from $q(x_t|x_0)$ for arbitrary time t
2. Tractable expression for $q(x_{t-1}|x_t, x_0)$.

If $x_t \in \mathbb{R}^D$, Gaussian $q(x_t|x_{t-1})$ (and $p_\theta(x_{t-1}|x_t)$): ☑ ☑

Diffusion models with discrete state spaces

Discrete random variables: $x_t \in \{0, \ldots., K-1\}$

Forward transition probabilities $q(x_t = j | x_{t-1} = j) = [Q_t]_{ij}$

In one-hot (row-based) representation: $q(x_t | x_{t-1}) = \mathrm{Cat}(x_t; p = x_{t-1} Q_t)$

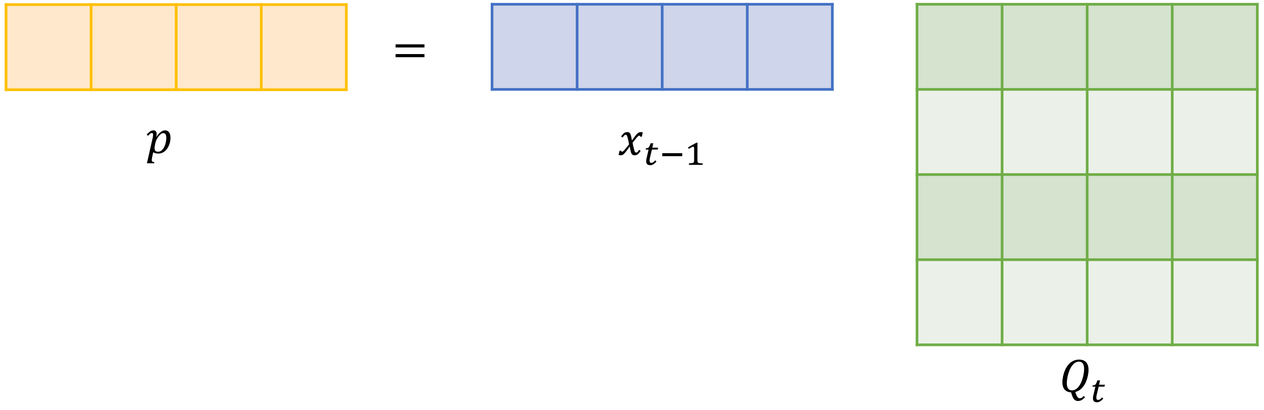
Practical requirements to allow for efficient training of p_θ:

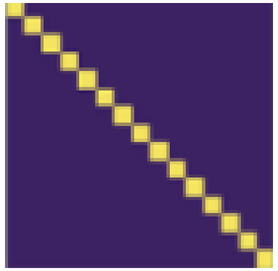
1. Efficient sampling of x_t from $q(x_t | x_0)$ for arbitrary time t.
   → $q(x_t | x_0)$ = $\mathrm{Cat}(x_t; p = x_0 \bar{Q}_t)$ with $\bar{Q}_t = Q_1 Q_2 \ldots Q_t$
2. Tractable expression for $q(x_{t-1} | x_t, x_0)$.
   → $q(x_{t-1} | x_t, x_0)$ = $\mathrm{Cat}(x_{t-1}; p = \frac{x_t Q_t^T \odot x_0 \bar{Q}_{t-1}}{x_0 \bar{Q}_t x_t^T})$

Sohl-Dickstein et al., ICML 2015
Hoogeboom et al., NeurIPS 2021
Jacob Austin*, Daniel Johnson*, Jonathan Ho, Daniel Tarlow, Rianne van den Berg, NeurIPS 2021

Choice of Markov transition matrix

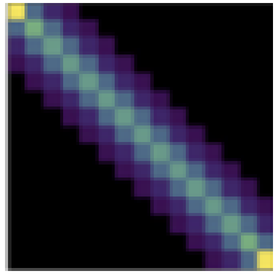
$$q(x_t|x_{t-1}) = \mathrm{Cat}(x_t; p = x_{t-1}Q_t)$$





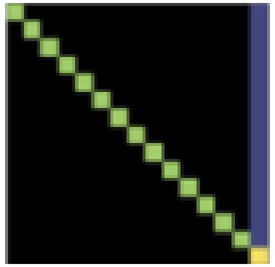
Structureless corruption:
Uniform transition probabilities:

Multinomial diffusion, Hoogeboom et al., NeurIPS 2021



Locality-sensitive transitions:
Transition with larger probability to nearby classes

- Ordinal data: images
- Similarity based on nearest-neighbour graph of token embeddings



Stay or transition to absorbing state:

Example: [MASK] token in mask-based language models

Jacob Austin*, Daniel Johnson*, Jonathan Ho, Daniel Tarlow, Rianne van den Berg, NeurIPS 2021

D3PMs for text generation: text8

Table 1: Quantitative results on text8. NLL is reported on the entire test set. Sample times are for generating a single example of length 256. Results are reported on two seeds. All models are standard 12-layer transformers unless otherwise noted. †Transformer XL is a 24-layer transformer, using a 784 context window. ‡Results reported by [20] by running code from official repository.

Model	Model steps	NLL (bits/char) (↓)	Sample time (s) (↓)
Discrete Flow [49] (8×3 layers)	-	1.23	0.16
Argmax Coupling Flow [20]	-	1.80	0.40 ± 0.03
IAF / SCF [57]‡	-	1.88	0.04 ± 0.0004
Multinomial Diffusion (D3PM uniform) [20]	1000	≤ 1.72	26.6 ± 2.2
D3PM uniform [20] (ours)	1000	$\leq 1.61 \pm 0.02$	3.6 ± 0.4
D3PM NN (L_{vb}) (ours)	1000	$\leq 1.59 \pm 0.03$	3.1474 ± 0.0002
D3PM mask ($L_{\lambda=0.01}$) (ours)	1000	$\leq 1.45 \pm 0.02$	3.4 ± 0.3
D3PM uniform [20] (ours)	256	$\leq 1.68 \pm 0.01$	0.5801 ± 0.0001
D3PM NN (L_{vb}) (ours)	256	$\leq 1.64 \pm 0.02$	0.813 ± 0.002
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	256	$\leq 1.47 \pm 0.03$	0.598 ± 0.002
Transformer decoder (ours)	256	1.23	0.3570 ± 0.0002
Transformer decoder [1]	256	1.18	-
Transformer XL [10]†	256	1.08	-
D3PM uniform [20] (ours)	20	$\leq 1.79 \pm 0.03$	0.0771 ± 0.0005
D3PM NN (L_{vb}) (ours)	20	$\leq 1.75 \pm 0.02$	0.1110 ± 0.0001
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	20	$\leq 1.56 \pm 0.04$	0.0785 ± 0.0003

D3PMs for text generation: LM1B

Table 2: Quantitative results on LM1B. Perplexity reported on the test set. Results are reported on two seeds. All models have context window length 128 and 12 layers unless otherwise noted. †Transformer XL is a 24 layer transformer. ‡rounded for readability, see Appendix B.2.2.

Metric:	Perplexity (↓)			Sample time‡ (s) (↓)		
inference steps:	1000	128	64	1000	128	64
D3PM uniform	137.9 ± 2.1	139.2 ± 1.2	145.0 ± 1.2	1.82	0.21	0.08
D3PM NN	149.5 ± 1.3	158.6 ± 2.2	160.4 ± 1.2	21.29	6.69	5.88
D3PM absorbing	76.9 ± 2.3	80.1 ± 1.2	83.6 ± 6.1	1.90	0.19	0.10
Transformer (ours)	-	43.6	-	-	0.26	-
Transformer XL [10]†	-	21.8	-	-	-	-

D3PMs for image generation

Table 3: Inception scores (IS), Frechet Inception Distance (FID) and negative log-likehood (NLL) on the image dataset CIFAR-10. The NLL is reported on the test set in bits per dimension. We report our results as averages with standard deviations, obtained by training five models with different seeds.

Model	IS (↑)	FID (↓)	NLL (↓)
Sparse Transformer [9]			2.80
NCSN [45]	8.87 ± 0.12	25.32	
NCSNv2 [46]	8.40 ± 0.07	10.87	
StyleGAN2 + ADA [22]	9.74 ± 0.05	3.26	
Diffusion (original), L_{vb} [43]			≤ 5.40
DDPM L_{vb} [19]	7.67 ± 0.13	13.51	≤ 3.70
DDPM L_{simple} [19]	9.46 ± 0.11	3.17	≤ 3.75
Improved DDPM L_{vb} [30]		11.47	≤ 2.94
Improved DDPM L_{simple} [30]		2.90	≤ 3.37
DDPM++ cont [47]		2.92	2.99
NCSN++ cont. [47]	9.89	2.20	
D3PM uniform L_{vb}	5.99 ± 0.14	51.27 ± 2.15	$\leq 5.08 \pm 0.02$
D3PM absorbing L_{vb}	6.26 ± 0.10	41.28 ± 0.65	$\leq 4.83 \pm 0.02$
D3PM absorbing $L_{\lambda=0.001}$	6.78 ± 0.08	30.97 ± 0.64	$\leq 4.40 \pm 0.02$
D3PM Gauss L_{vb}	7.75 ± 0.13	15.30 ± 0.55	$\leq 3.966 \pm 0.005$
D3PM Gauss $L_{\lambda=0.001}$	8.54 ± 0.12	8.34 ± 0.10	$\leq 3.975 \pm 0.006$
D3PM Gauss + logistic $L_{\lambda=0.001}$	8.56 ± 0.10	7.34 ± 0.19	$\leq 3.435 \pm 0.007$

D3PMs for image generation

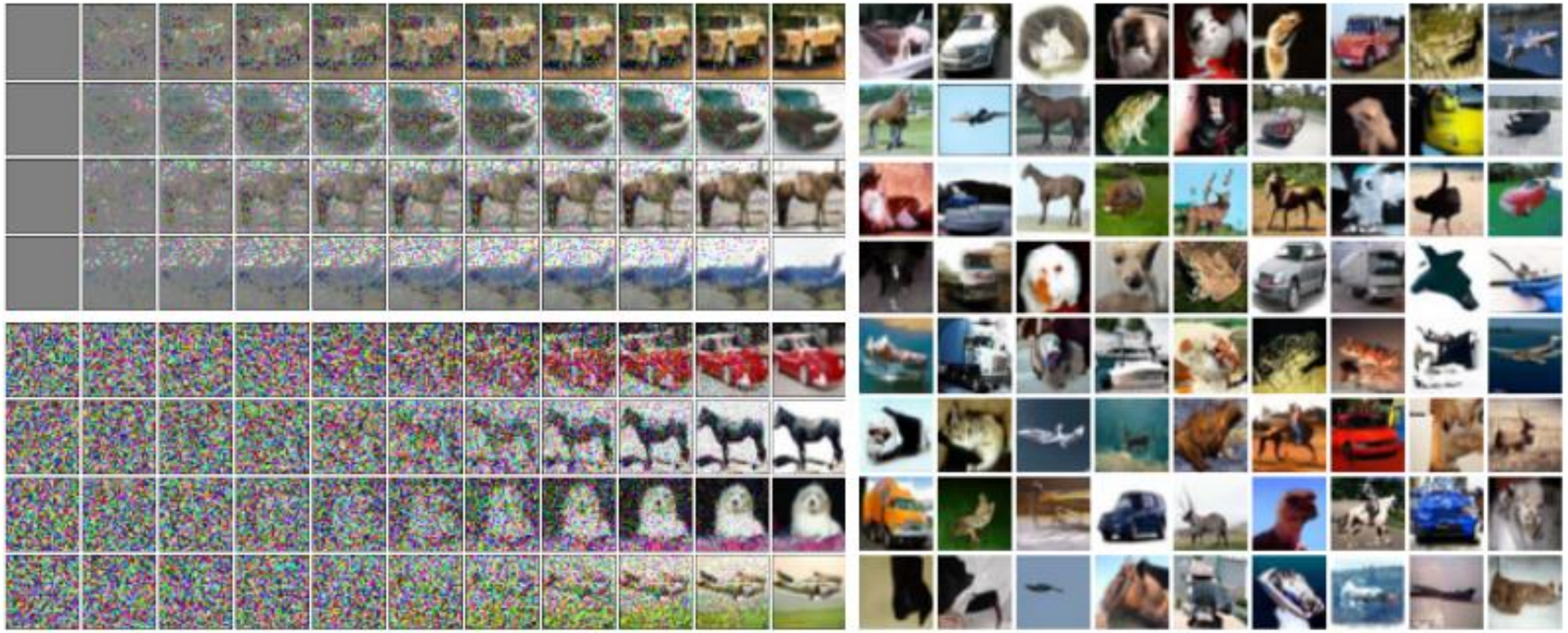


Figure 3: Left: progressive sampling at $t = 1000, 900, 800, ..., 0$ for D3PM absorbing (top) and D3PM Gauss + logistic (bottom), trained with L_λ loss on CIFAR-10. These samples were cherry picked. Right: (non cherry picked) samples from the D3PM Gauss + logistic model.

Outline

- Motivation for generative modeling for discrete random variables: lossless compression
  - Basics of lossless compression
  - Connecting likelihood-based generative models and lossless compression
- Integer discrete normalizing flows for lossless compression
- Denoising diffusion models
- **Autoregressive models**

Autoregressive models

Factorized density: $p_\theta(x_1, \ldots, x_D) = \prod_{i=1}^{D} p_\theta(x_i | x_{i-1}, \ldots, x_1)$

Pros:

- No problem handling discrete data.
- among SOTA models for density estimation.

Cons:

- Requires D sequential steps encoding and decoding → very slow
- Fixed factorization → Not ideal for inpainting.
- Implementing autoregressive architecture is tricky: causal masking in conv filters.

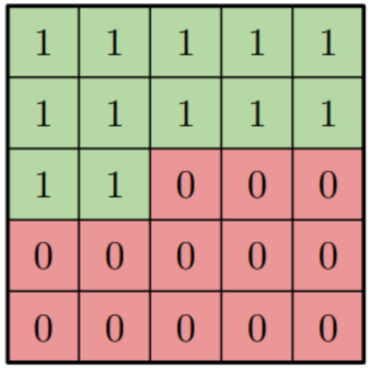


Figure source: van den Oord., NeurIPS 2016

Work that tries to speed up autoregressive compression by combining it with super-resolution: Cao et al., 2020, arXiv:2004.02872

Order agnostic autoregressive models

Uria et al., a deep and tractable density estimator, ICML 2014

AR with fixed ordering σ:

$$p_\theta(x_1, \ldots, x_D; \sigma) = \prod_{i=1}^{D} p_\theta(x_{\sigma(i)} | x_{\sigma(i-1)}, \ldots, x_{\sigma(1)})$$

Order-agnostic model:

$$\log p(x_1, \ldots, x_D) \geq \mathbb{E}_{\sigma \sim U(S_D)} \sum_t^D \log p(x_{\sigma(t)} | x_{\sigma(<t)})$$

$$= \mathbb{E}_t [\frac{D}{D-t+1} \mathbb{E}_{\sigma \sim U(S_D)} \sum_{k \in \sigma(\geq t)} \log p(x_k | x_{\sigma(<t)})]$$

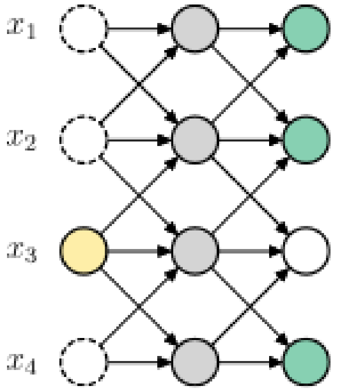


Figure 2: ARDM training step. This step optimizes for step $t = 2$ for all possible permutations σ simultaneously which satisfy $\sigma(1) = 3$.

Emiel Hoogeboom, Alexey Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, Tim Salimans. arXiv:2110.02037

Autoregressive diffusion models

Emiel Hoogeboom, Alexey Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, Tim Salimans. arXiv:2110.02037

Same loss as in [Uria et al. 2014]:

$$\log p(x_1, \ldots, x_D) \geq \mathbb{E}_t [\frac{D}{D-t+1} \mathbb{E}_{\sigma \sim U(S_D)} \sum_{k \in \sigma(\geq t)} \log p(x_k | x_{\sigma(<t)})]$$

Makes connection to

1. absorbing state discrete diffusion models [Austin et al. 2021]
2. Dynamics programming to reduce the number of diffusion steps [Watson et al. 2021]

→Parallel sampling of multiple variables.

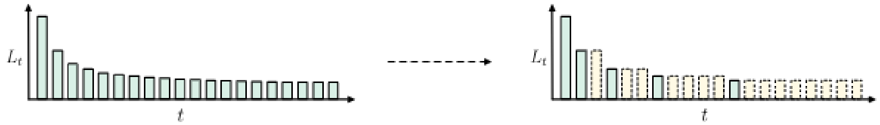


Figure 3: Loss components for Parallelized ARDMs using a budget of 5 steps for a problem of 20 steps. Left: individual loss component for every step. Right: parallelized policy extracted from the dynamic programming algorithm. Components of the same height are modelled simultaneously, so they are inferred and generated in parallel.

Other benefit: No need for causal masking of conv filters, just input and output masking.

Results: text8 and CIFAR-10

Table 1: Order Agnostic model performance (in bpc) on the text8 dataset. The OA-Transformer learns arbitrary orders by permuting inputs and outputs as described in XLNet. A Transformer learning only a single order achieves 1.35 bpc.

Model	Steps	NLL
OA-Transformer	250	1.64
D3PM-uniform	1000	1.61 $\pm$0.020
D3PM-absorbing	1000	1.45 $\pm$0.020
D3PM-absorbing	**256**	1.47
OA-ARDM (ours)	**250**	**1.43** $\pm$0.001
D3PM-absorbing	**20**	1.56 $\pm$0.040
Parallelized OA-ARDM (ours)	**20**	**1.51** $\pm$0.007

Table 2: Order Agnostic modelling performance (in bpd) on the CIFAR-10 dataset. The upscaling model generates groups of four most significant categories, equivalent to 2 bits at a time.

Model	Steps	NLL
ARDM-OA	3072	2.69 $\pm$ 0.005
Parallel ARDM-OA	50	2.74
ARDM-Upscale 4	4×3072	**2.64** $\pm$ 0.002
Parallel ARDM-Upscale 4	4×50	2.68
D3PM Absorbing	1000	4.40
D3PM Gaussian	1000	3.44 $\pm$ 0.007

Figure 5: Visualization of x through the generative process for an ARDM Upscale 4 model.

Lossless compression CIFAR-10

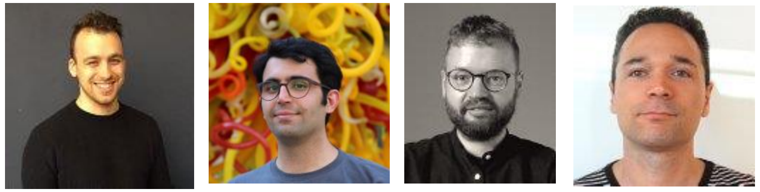
Table 3: CIFAR-10 lossless compression performance (in bpd).

Model	Steps	Compression per image	Dataset compression
VDM (Kingma et al., 2021)	1000	≥ 8	2.72
VDM (Kingma et al., 2021)	500	≥ 8	2.72
OA-ARDM (ours)	500	2.73	2.73
ARDM-Upscale 4 (ours)	500	**2.71**	**2.71**
VDM (Kingma et al., 2021)	100	≥ 8	2.91
OA-ARDM (ours)	100	2.75	2.75
ARDM-Upscale 4 (ours)	100	2.76	2.76
LBB (Ho et al., 2019)		≥ 8	3.12
IDF (Hoogeboom et al., 2019)		3.34	3.34
IDF++ (van den Berg et al., 2021)		3.26	3.26
HiLLoC (Townsend et al., 2020)		4.19	3.56
FLIF (Sneyers & Wuille, 2016)		4.19	4.19

Collaborators for this work

Integer discrete flows and lossless compression
Emiel Hoogeboom*, Jorn Peters*, Rianne van den Berg, Max Welling, NeurIPS 2019

IDF++: Analyzing and improving Integer Discrete Flows for lossless compression
Rianne van den Berg, Alexey Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, Tim Salimans, ICLR 201



Structured denoising diffusion models in discrete state-spaces
Jacob Austin*, Daniel Johnson*, Jonathan Ho, Daniel Tarlow, Rianne van den Berg, NeurIPS 2021

Autoregressive diffusion models
Emiel Hoogeboom, Alexey Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, Tim Salimans, arXiv:2110.02037

* Equal contributions