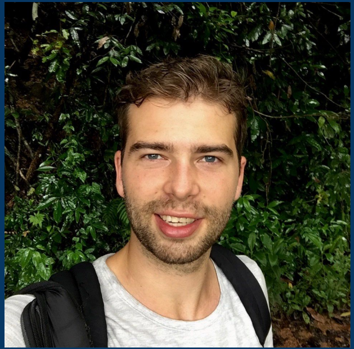
Normalizing flows for discrete data

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg* & Max Welling.



*Work done while at University of Amsterdam

Integer discrete flows and lossless compression

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, Max Welling

Normalizing flows | **Quantized RV's** | **Images** | **Compression**

Discrete Flows: Invertible Generative Models of Discrete Data

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, Ben Poole

Normalizing flows | **Quantized RV's** | **Text** | **Fast generation**

Compression with Flows via Local Bits-Back Coding

Jonathan Ho, Evan Lohn, Pieter Abbeel

Normalizing flows | **Continuous RV's** | **Images** | **Compression**

Integer discrete flows and lossless compression

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, Max Welling

Normalizing flows | Quantized RV’s | Images | Compression

Discrete Flows: Invertible Generative Models of Discrete Data

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, Ben Poole

Normalizing flows | Quantized RV’s | Text | Fast generation

Compression with Flows via Local Bits-Back Coding

Jonathan Ho, Evan Lohn, Pieter Abbeel

Normalizing flows | Continuous RV’s | Images | Compression

LOSSLESS SOURCE COMPRESSION

Probable inputs map to shorter codes and improbable inputs are mapped to longer codes.

Minimum code length for a symbol x is close to

$$-\log D(x)$$

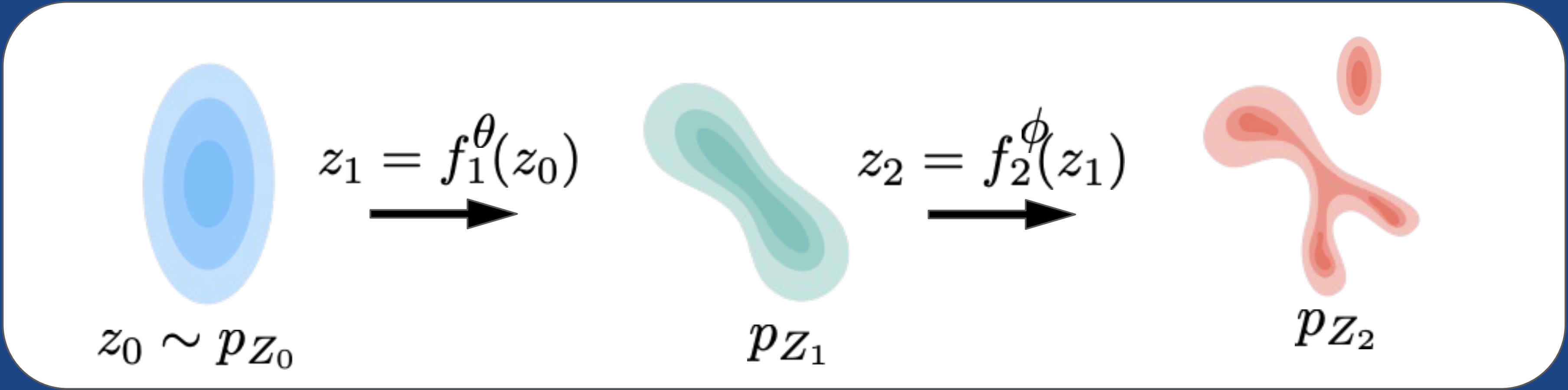
Minimum expected code length:

$$\mathbb{E}_{x\sim D}[|c(x)|] \geq \mathbb{E}_{x\sim \mathcal{D}}[-\log p_\theta(x)] \geq \mathcal{H}(\mathcal{D}) = \mathbb{E}_{x\sim \mathcal{D}}[-\log \mathcal{D}(x)]$$

a_i	$c(a_i)$	p_i	$h(p_i)$	l_i
a	0	$1/2$	1.0	1
b	10	$1/4$	2.0	2
c	110	$1/8$	3.0	3
d	111	$1/8$	3.0	3

NORMALIZING FLOWS: CONTINUOUS RANDOM VARIABLES

Idea: Apply a sequence of invertible transformations to a random variable.



$$p(z_1|z_0) = \delta(z_1 - f_1(z_0))$$

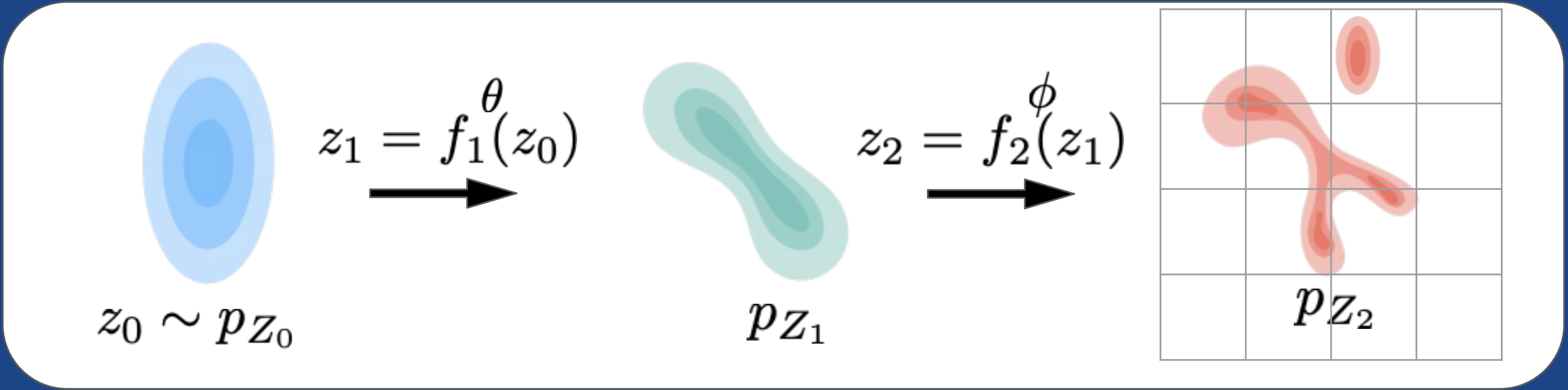
$$p(z_1) = \int p(z_1|z_0)p(z_0)\,\mathrm{d}z_0 = p(f_1^{-1}(z_1))\left|\frac{\partial z_1}{\partial z_0}\right|^{-1}$$

NAIVE LOSSLESS COMPRESSION FOR NORMALIZING FLOWS

Entropy encoders require as input:

- Input in the form of symbols
- The distribution of the symbols

Quantize latent distribution



Leads to reconstruction error → needs to be encoded too for lossless compression.

NORMALIZING FLOWS FOR INTEGER VALUED DATA

Problem formulation: Define invertible $f_\theta : \mathbb{Z}^d \mapsto \mathbb{Z}^d$

Simplest solution: Take RealNVP and make suitable for integers.

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ s^\theta(x_1) \odot x_2 + t^\theta(x_1) \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} z_1 \\ (z_2 - t^\theta(z_1)) \oslash s^\theta(z_1) \end{bmatrix}$$

Integer discrete flows for lossless compression, E Hoogeboom, J Peters, RvdB, M Welling, NeurIPS 2019

NORMALIZING FLOWS FOR INTEGER VALUED DATA

Problem formulation: Define invertible $f_\theta : \mathbb{Z}^d \mapsto \mathbb{Z}^d$

Simplest solution: Take RealNVP and make suitable for integers.

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 + \boxed{t^\theta(x_1)} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} z_1 \\ z_2 - \boxed{t^\theta(z_1)} \end{bmatrix}$$

Integer discrete flows for lossless compression, E Hoogeboom, J Peters, RvdB, M Welling, NeurIPS 2019

NORMALIZING FLOWS FOR INTEGER VALUED DATA

Problem formulation: Define invertible $f_\theta : \mathbb{Z}^d \mapsto \mathbb{Z}^d$

Simplest solution: Take RealNVP and make suitable for integers.

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 + \boxed{\lfloor t^\theta(x_1) \rceil} \end{bmatrix}$$

Use straight-through estimator to backprop gradients

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} z_1 \\ z_2 - \boxed{\lfloor t^\theta(z_1) \rceil} \end{bmatrix}$$

Integer discrete flows for lossless compression, E Hoogeboom, J Peters, RvdB, M Welling, NeurIPS 2019

OBTAINING THE DENSITY

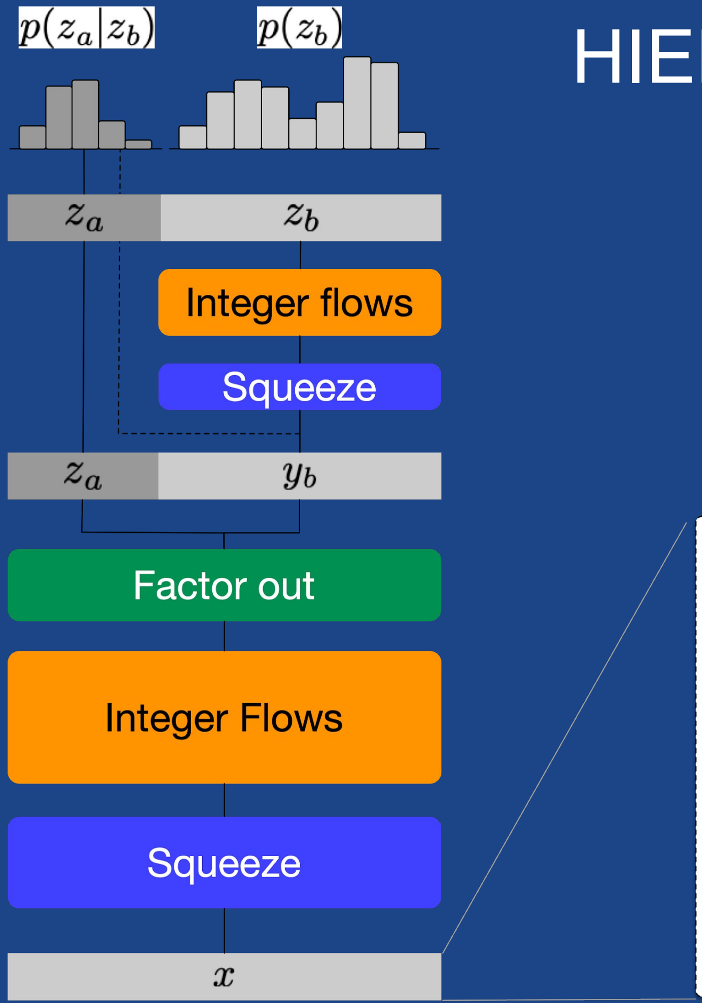
Continuous random variables:

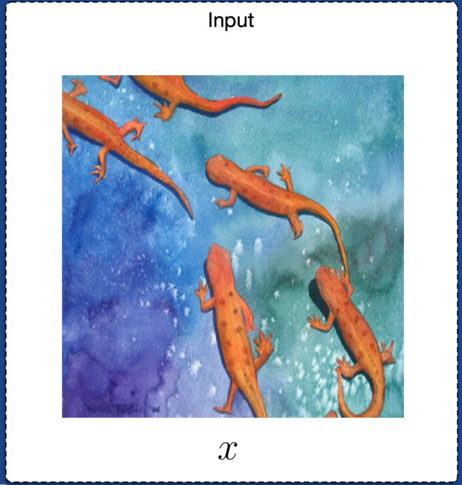
$$p(x) = \int p(x|z)p(z)\ \mathrm{d}z = \int \delta(x - f(z))p(z)\ \mathrm{d}z = p(f^{-1}(x))\left|\frac{\partial x}{\partial z}\right|^{-1}$$

Discrete random variables:

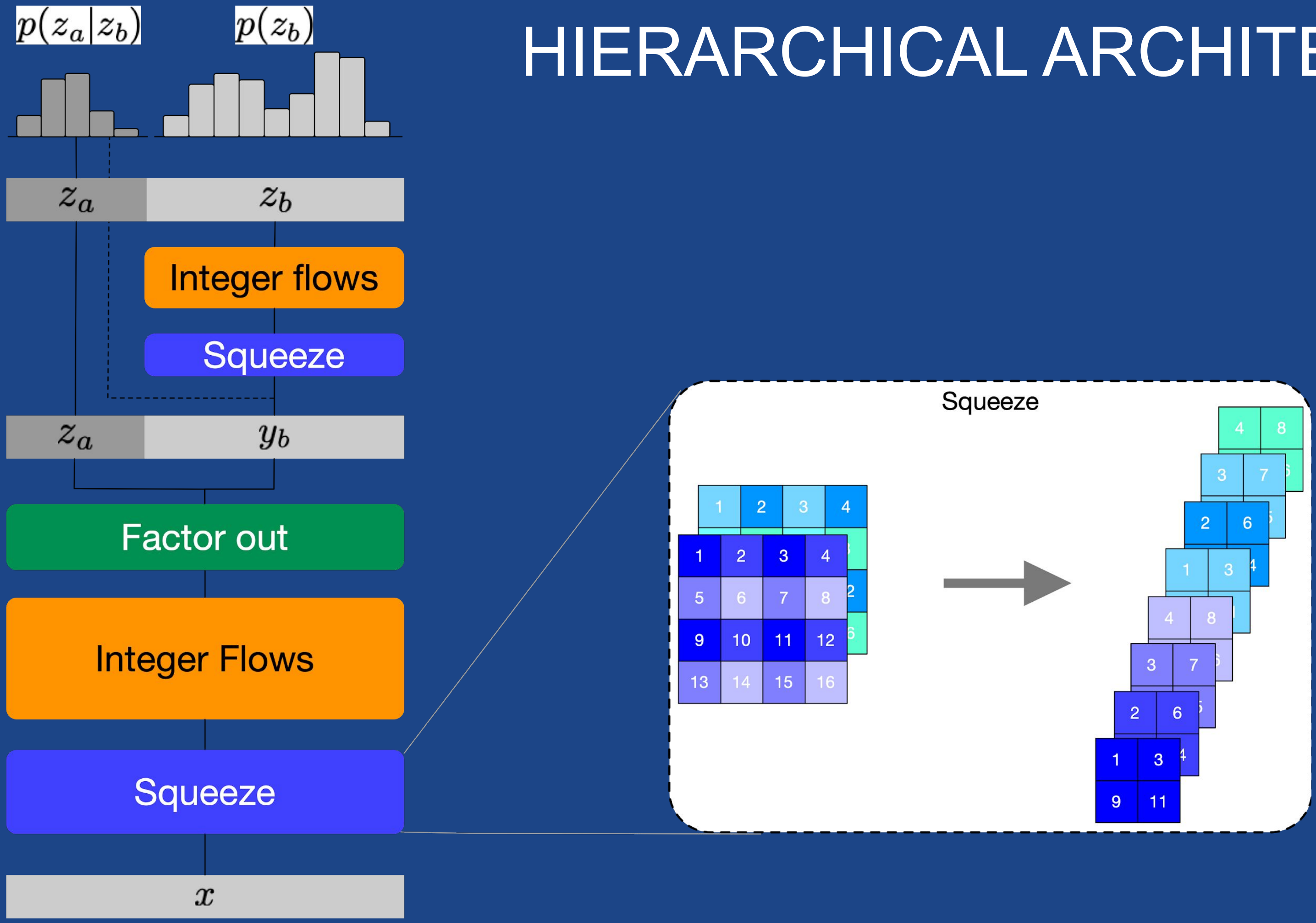
$$p(x) = \sum_z p(x|z)p(z) = \sum_z \delta_{z,f^{-1}(x)}p(z) = p(f^{-1}(x))$$

HIERARCHICAL ARCHITECTURE

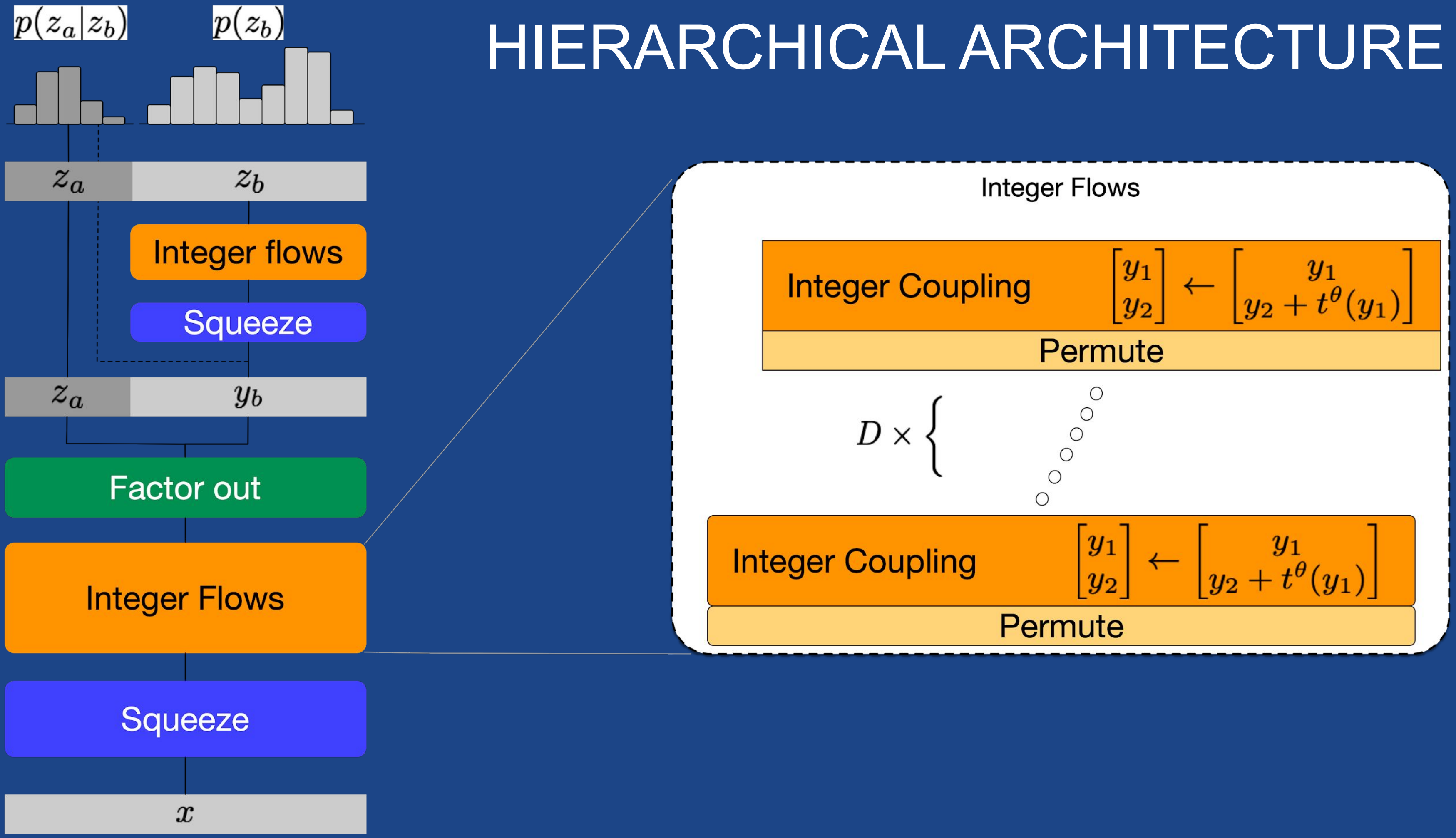




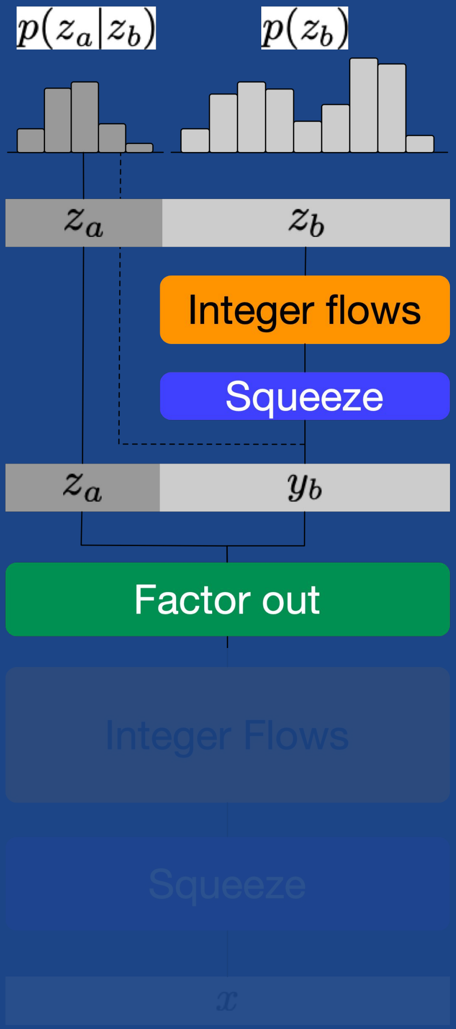
HIERARCHICAL ARCHITECTURE



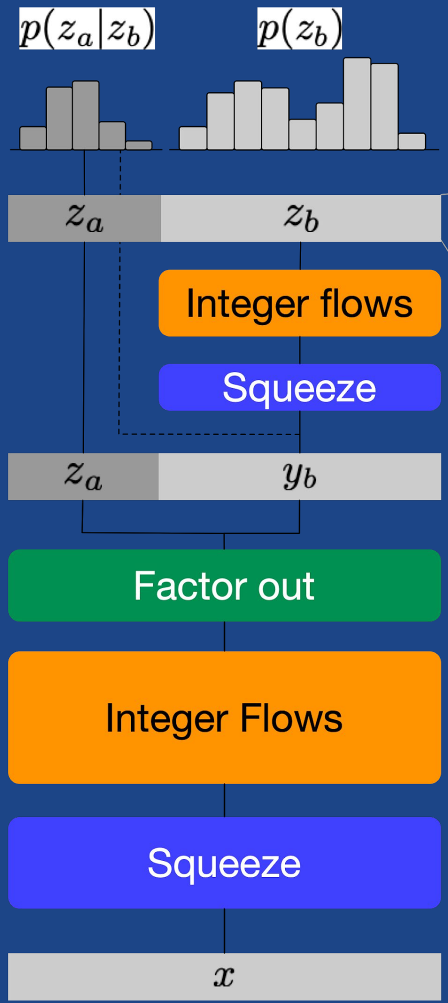
HIERARCHICAL ARCHITECTURE

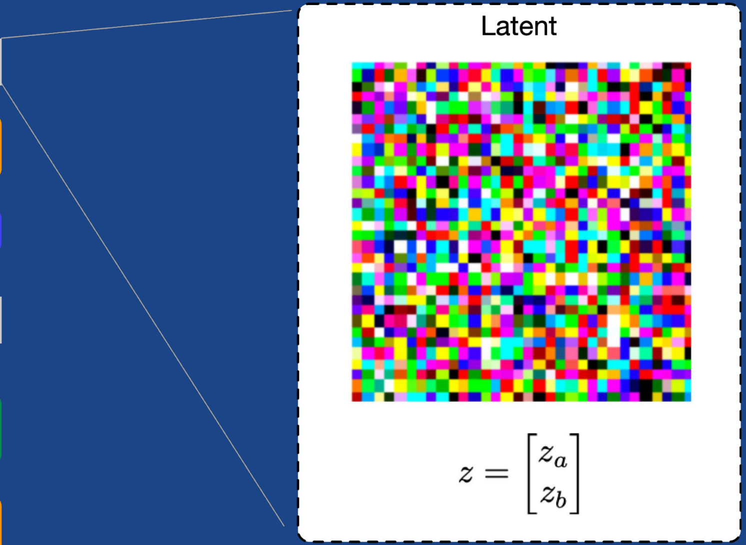


HIERARCHICAL ARCHITECTURE

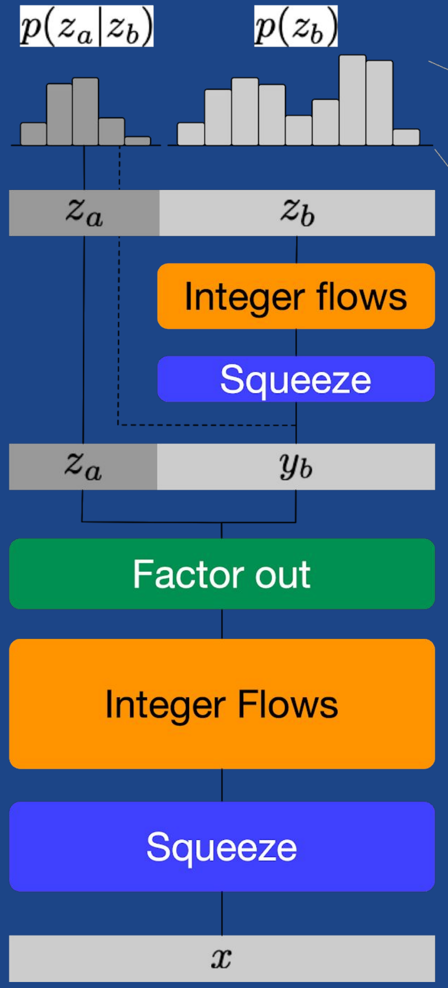


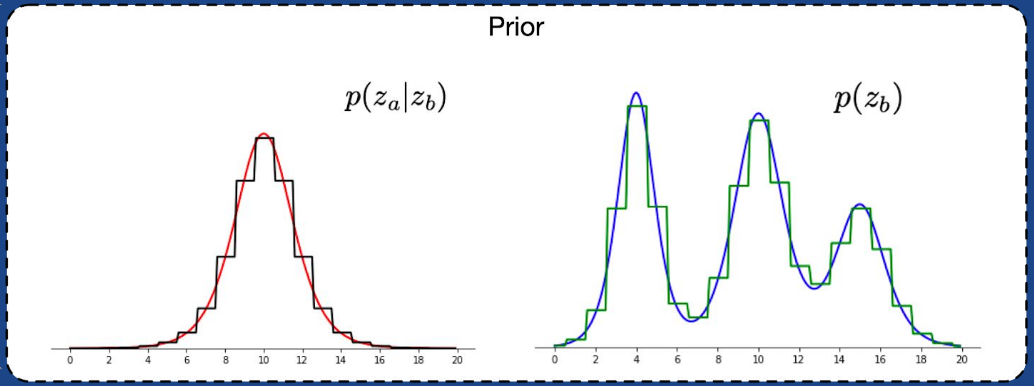
HIERARCHICAL ARCHITECTURE



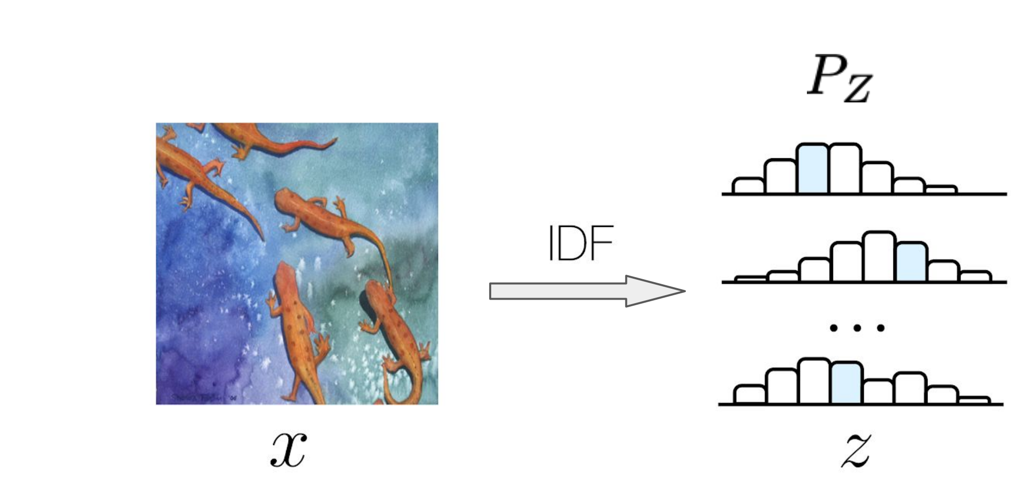


HIERARCHICAL ARCHITECTURE

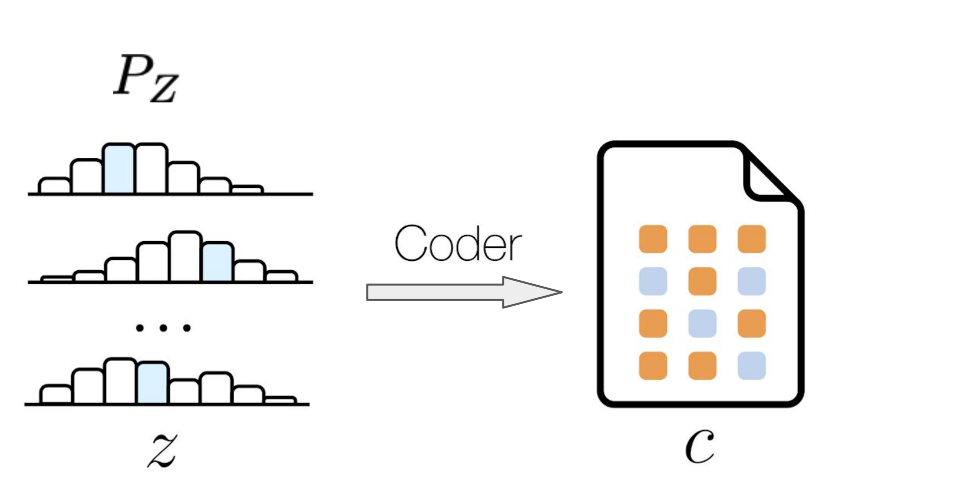




LOSSLESS SOURCE COMPRESSION



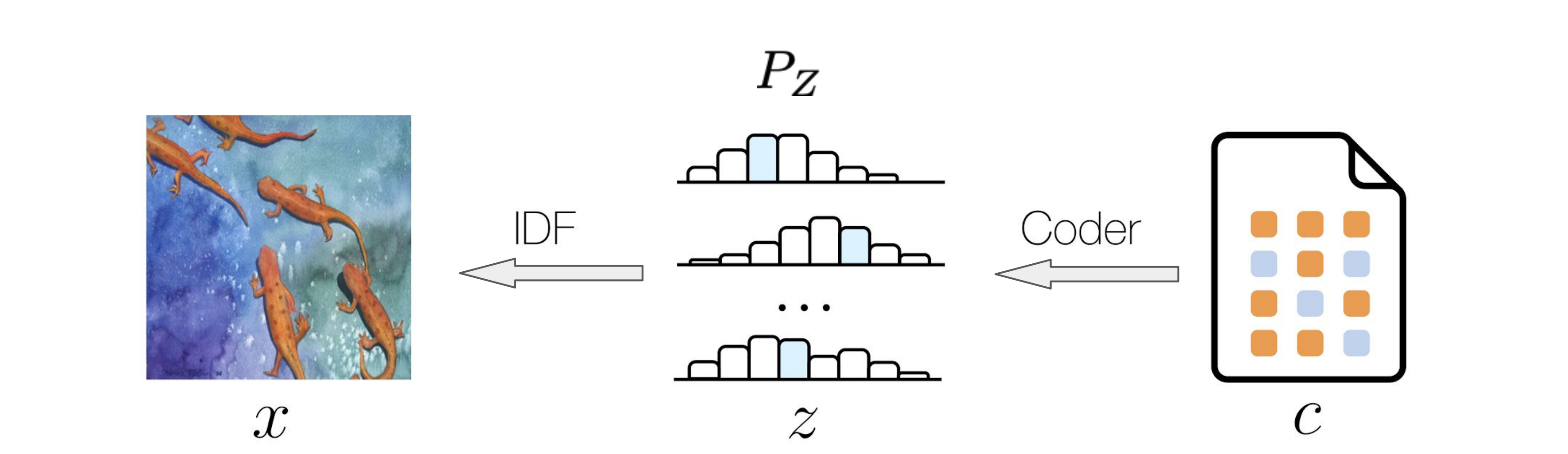
LOSSLESS SOURCE COMPRESSION



High-probability z → Short code

Low-probability z → Long code

LOSSLESS SOURCE COMPRESSION



High-probability z → Short code

Low-probability z → Long code

RESULTS

Table 1: Compression performance of IDFs on CIFAR10, ImageNet32 and ImageNet64 in bits per dimension, and compression rate (shown in parentheses). The Bit-Swap results are retrieved from [23]. The column marked IDF† denotes an IDF trained on ImageNet32 and evaluated on the other datasets.

Dataset	IDF	IDF†	Bit-Swap	FLIF [34]	PNG	JPEG2000
CIFAR10	**3.34 (2.40**$\times$**)**	3.60 (2.22$\times$)	3.82 (2.09$\times$)	4.37 (1.83$\times$)	5.89 (1.36$\times$)	5.20 (1.54$\times$)
ImageNet32	**4.18 (1.91**$\times$**)**	**4.18 (1.91**$\times$**)**	4.50 (1.78$\times$)	5.09 (1.57$\times$)	6.42 (1.25$\times$)	6.48 (1.23$\times$)
ImageNet64	**3.90 (2.05**$\times$**)**	3.94 (2.03 $\times$)	–	4.55 (1.76$\times$)	5.74 (1.39$\times$)	5.10 (1.56$\times$)

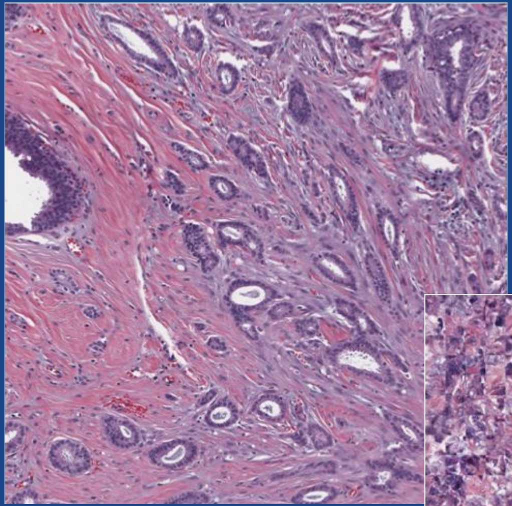
Table 3: Generative modeling performance of IDFs and comparable flow-based methods in bits per dimension (negative $\log_2$-likelihood).

Dataset	IDF	Continuous	RealNVP	Glow	Flow++
CIFAR10	3.32	3.31	3.49	3.35	3.08
ImageNet32	4.16	4.13	4.28	4.09	3.86
ImageNet64	3.90	3.85	3.98	3.81	3.69

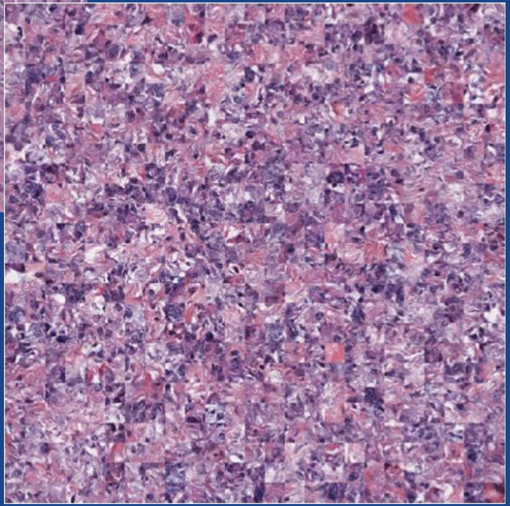
MEDICAL DATA: HISTOLOGY DATA

Resolution: 2000 x 2000 pixels

IDF trained on 80 x 80 px patches



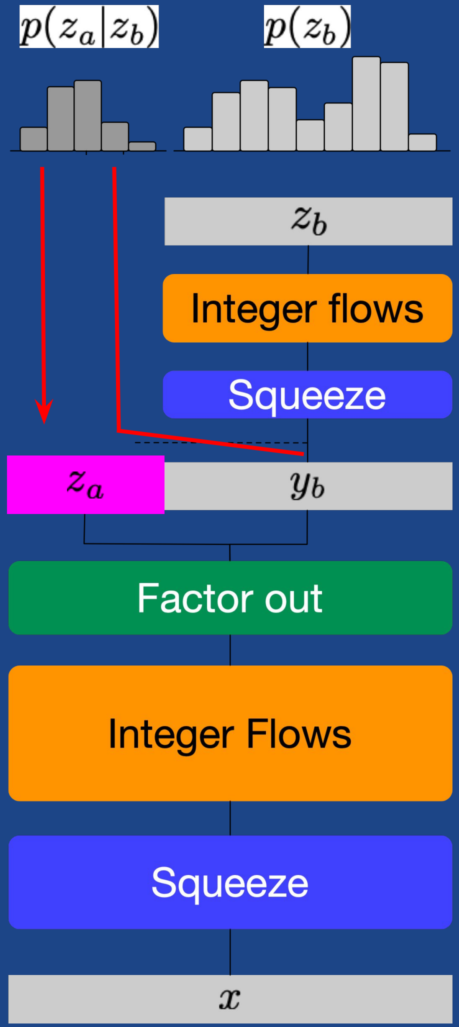
Sampled patches:
80 x 80 pixels



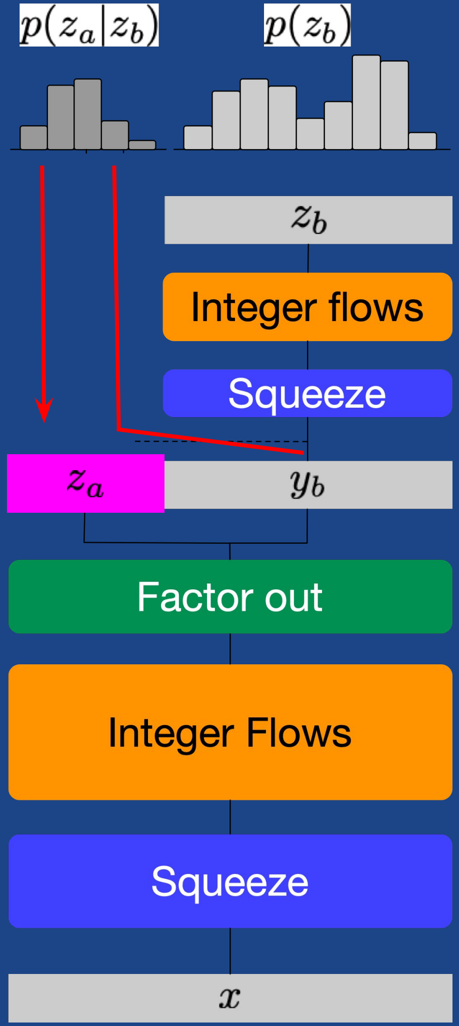
patch-wise compression (each patch is considered independent)

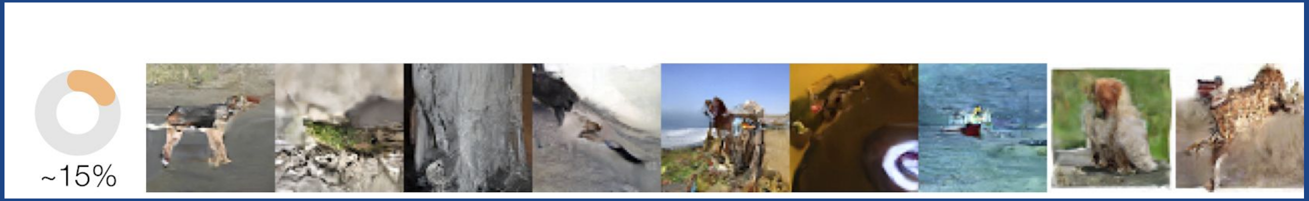
Dataset	IDF	JP2-WSI	FLIF [34]	JPEG2000
Histology	**2.42 (3.19×)**	3.04 (2.63×)	4.00 (2.00×)	4.26 (1.88×)

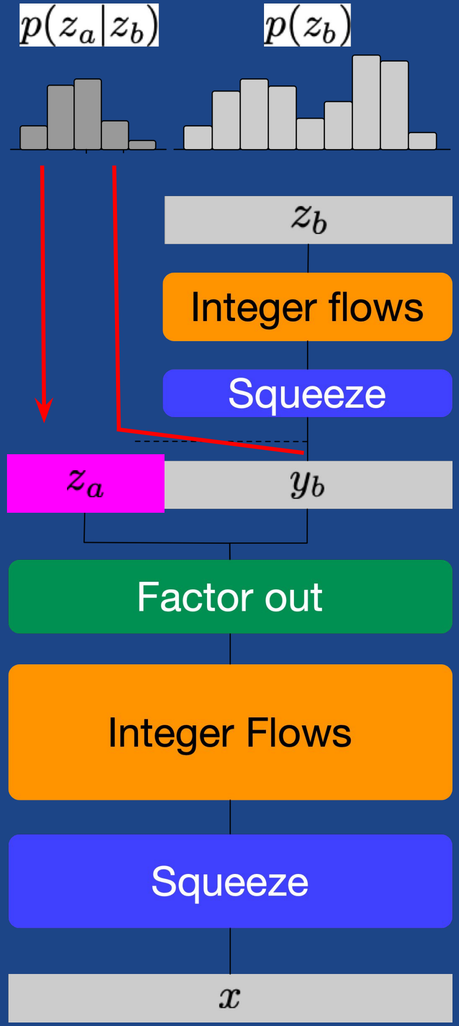
PROGRESSIVE IMAGE RENDERING



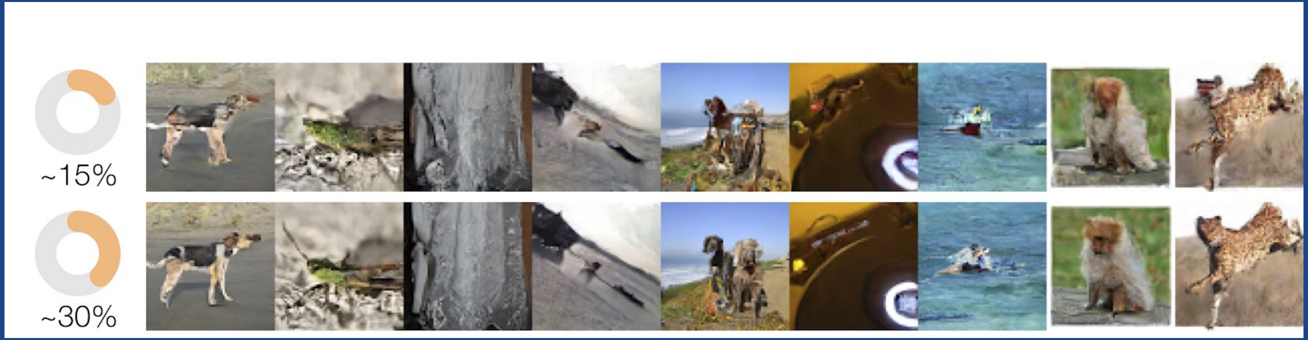
PROGRESSIVE IMAGE RENDERING



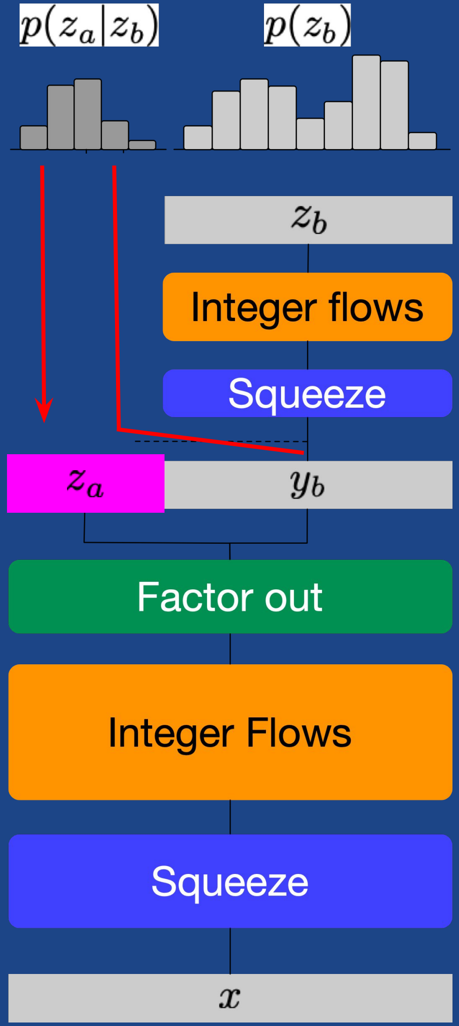


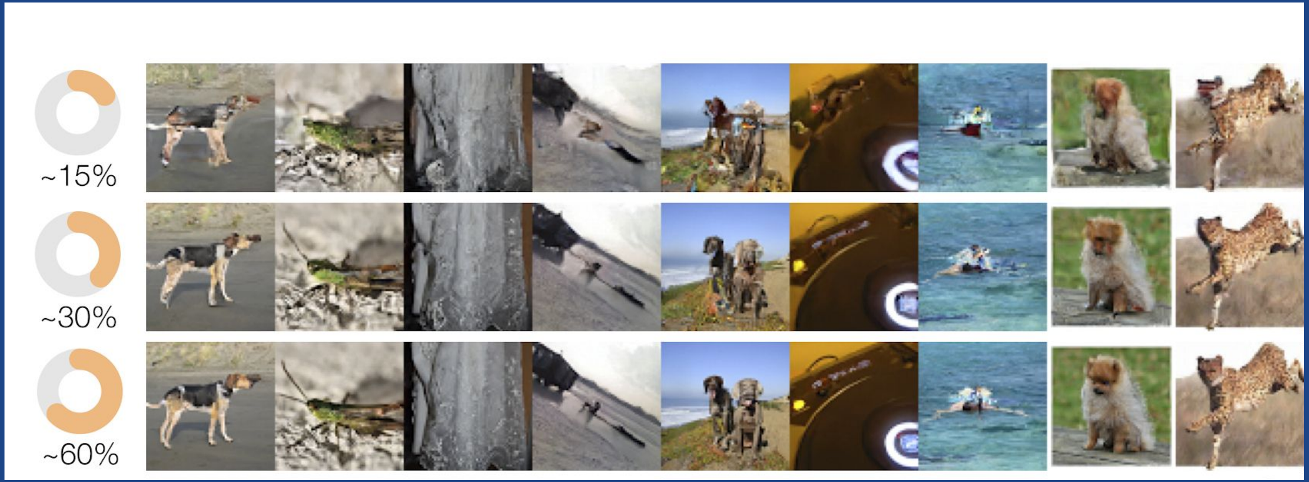


PROGRESSIVE IMAGE RENDERING

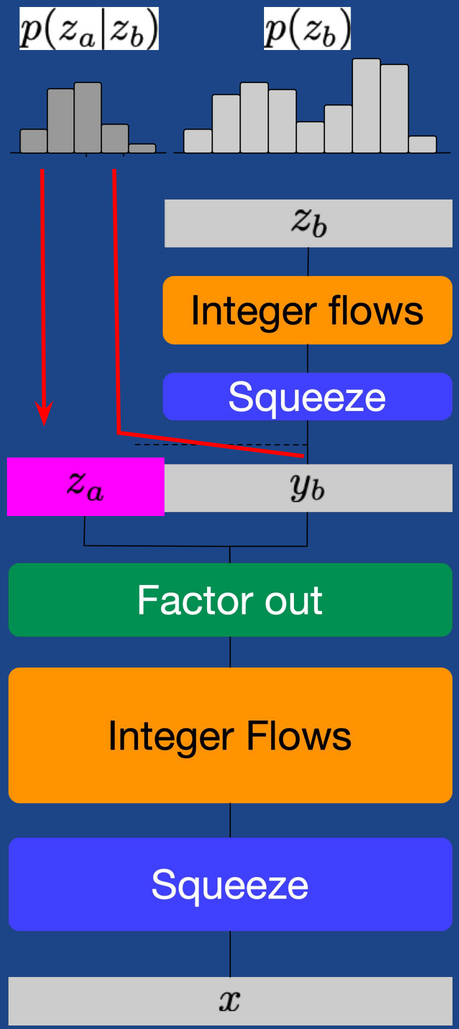


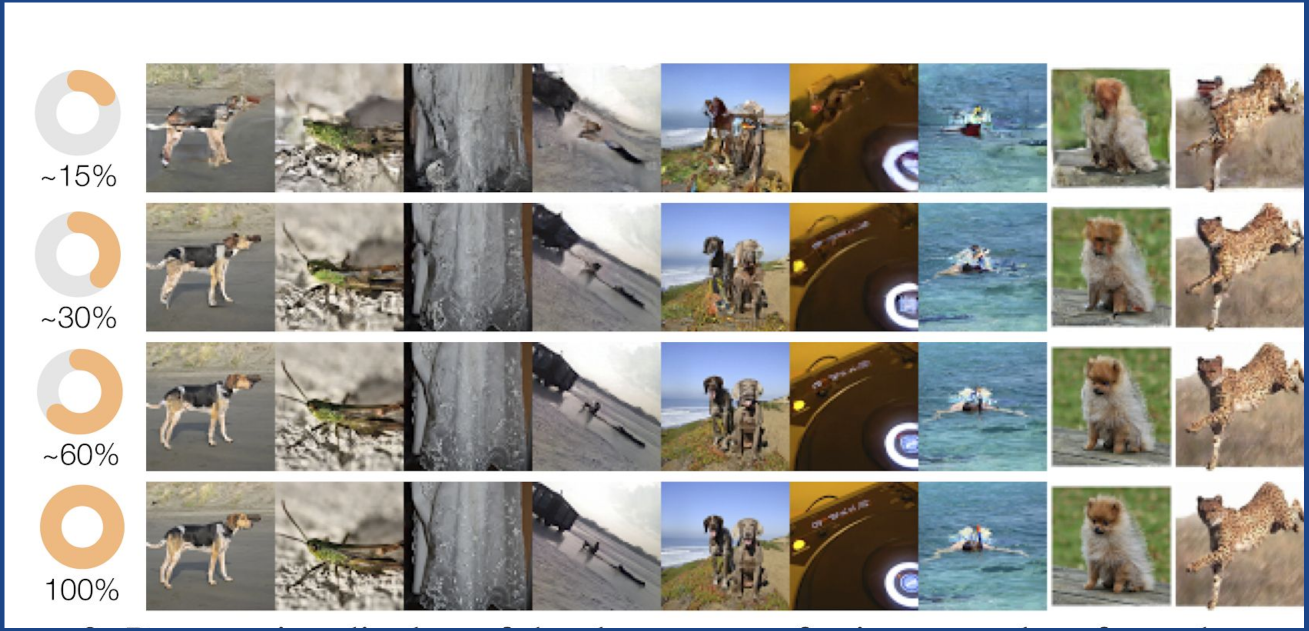
PROGRESSIVE IMAGE RENDERING





PROGRESSIVE IMAGE RENDERING

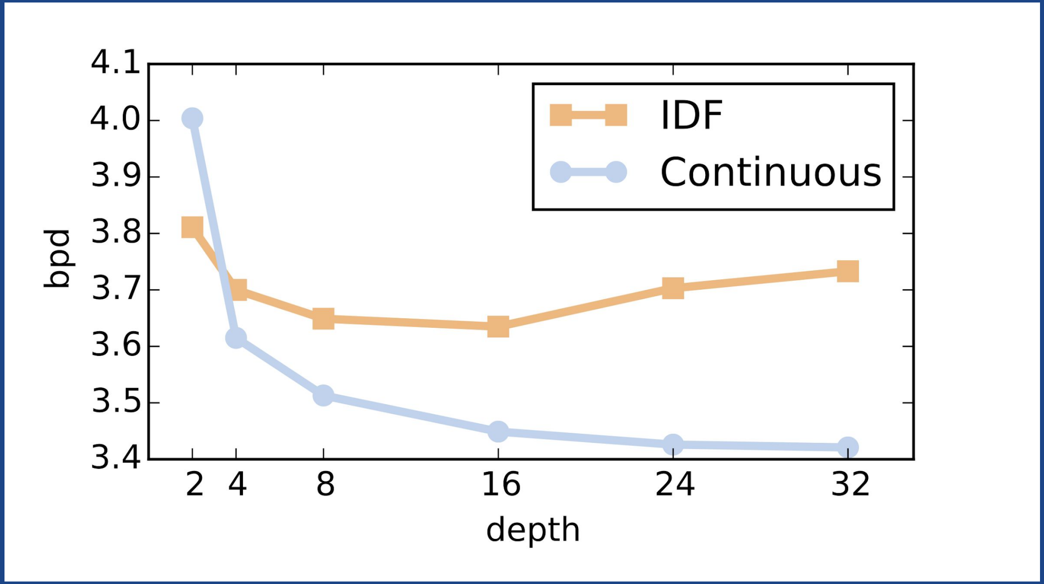




DIRECTIONS FOR IMPROVEMENT

Problem: discrete flows don't always benefit from more flow layers.

Hypothesis: gradient bias in straight through estimator is the cause.



Integer discrete flows and lossless compression

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, Max Welling

Normalizing flows | Quantized RV's | Images | Compression

Discrete Flows: Invertible Generative Models of Discrete Data

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, Ben Poole

Normalizing flows | Quantized RV's | Text | Fast generation

Compression with Flows via Local Bits-Back Coding

Jonathan Ho, Evan Lohn, Pieter Abbeel

Normalizing flows | Continuous RV's | Images | Compression

DISCRETE FLOWS

Discrete but not ordinal data with finite number of classes

Focus: generative modeling for text (character level)

Architecture variants:

- Autoregressive layers
- Coupling layer/bi-partite layers

INTEGER DISCRETE FLOWS

Ordinal discrete data with possibly infinite number of classes

Focus: lossless source compression for images

Architecture variants:

- Coupling layer/bi-partite layers

DISCRETE FLOWS

Bipartite bijector

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ [s(x_1) \odot x_2 + t(x_1)] \bmod K \end{bmatrix}$$

Autoregressive bijector

$$z_d = [s_d(x_{1:d-1}) \odot x_d + t_d(x_{1:d-1})] \bmod K$$

$$t : \{0, 1, ..., K-1\} \mapsto \{0, 1, ..., K-1\}$$
$$s : \{0, 1, ..., K-1\} \mapsto \{1, ..., K-1\}$$

Only invertible if s and K are coprime

$$t_d = \mathrm{one\_hot}(\arg\max(\theta_d))$$

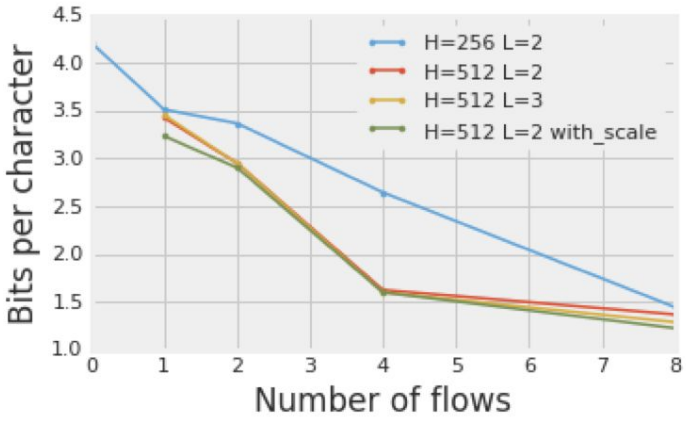
$$\frac{\partial t_d}{\partial \theta_d} \approx \frac{\partial}{\partial \theta_d} \mathrm{softmax}\left(\frac{\theta_d}{\tau}\right)$$

RESULTS

	Test NLL (bpc)	Generation
3-layer LSTM (Merity et al., 2018)	**1.18**[3]	3.8 min
Ziegler and Rush (2019) (AF/SCF)	1.46	-
Ziegler and Rush (2019) (IAF/SCF)	1.63	-
Bipartite flow	1.38	**0.17 sec**

Table 3: Character-level language modeling results on Penn Tree Bank.

RESULTS



	bpc	Gen.
LSTM (Coojimans+2016)	1.43	19.8s
64-layer Transformer (Al-Rfou+2018)	**1.13**	35.5s
Bipartite flow (4 flows, w/ σ)	1.60	**0.15s**
Bipartite flow (8 flows, w/o σ)	1.29	**0.16s**
Bipartite flow (8 flows, w/ σ)	1.23	**0.16s**

Figure 3: Character-level language modeling results on text8. The test bits per character decreases as the number of flows increases. More hidden units H and layers L in the Transformer per flow, and applying a scale transformation instead of only location, also improves performance.

Integer discrete flows and lossless compression

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, Max Welling

Normalizing flows | Quantized RV's | Images | Compression

Discrete Flows: Invertible Generative Models of Discrete Data

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, Ben Poole

Normalizing flows | Quantized RV's | Text | Fast generation

Compression with Flows via Local Bits-Back Coding

Jonathan Ho, Evan Lohn, Pieter Abbeel

Normalizing flows | Continuous RV's | Images | Compression