Integer Discrete Flows and Lossless Compression

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

Universiteit van Amsterdam

UvA - BOSCH DELTA LAB

* Now at Google.

Lossless source compression

- Map every input to unique output such that probable inputs map to shorter codes and improbable inputs are mapped to longer codes.
- Minimum code length for a symbol x is close to $-\log \mathcal{D}(x)$

- Minimum expected code length:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[|c(x)|\right] \approx \mathbb{E}_{x\sim\mathcal{D}}\left[-\log p_X(x)\right] \geq \mathcal{H}(\mathcal{D})$$

Normalizing flows for integer-valued data

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

Integer Discrete Flows (IDFs): Remove scaling in coupling layers (step 1).

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

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

Normalizing flows for integer-valued data

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

Integer Discrete Flows (IDFs): Constrain translations to be integer valued (step 2).

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

Use straight-through estimator to backprop gradients

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

Obtaining the density

Continuous random variables:

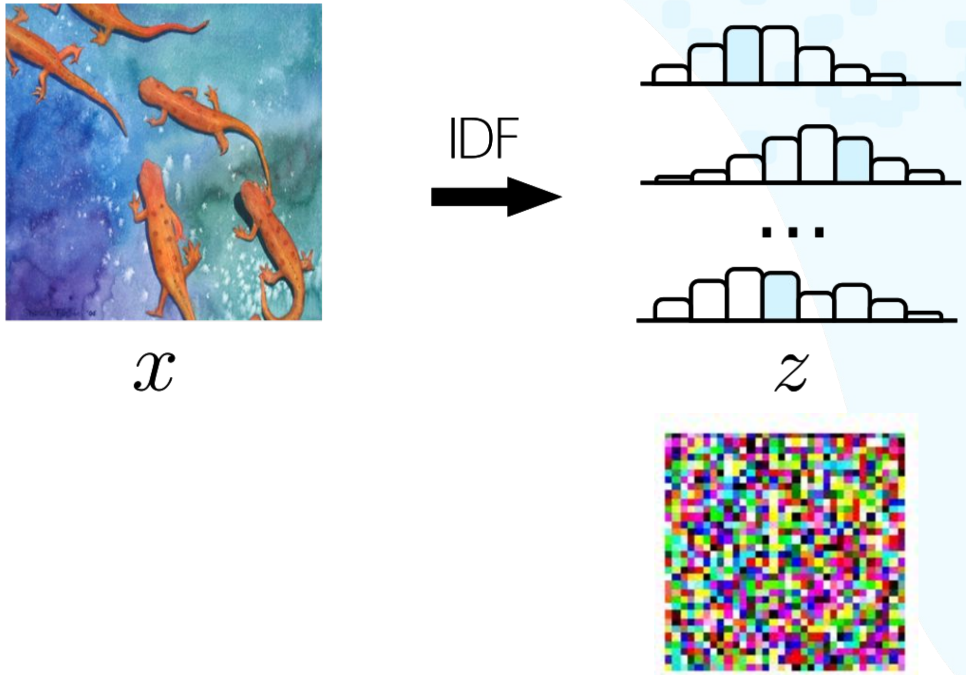
$$p(x) = \int p(x|z)p(z)\ \mathrm{d}z = \int \delta(x - f(z))p(z)\ \mathrm{d}z = p(f^{-1}(x))\left|\frac{\partial x}{\partial z}\right|^{-1}$$

Discrete random variables:

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

Lossless source compression

Step 1: transform input data to z-space using the IDF.

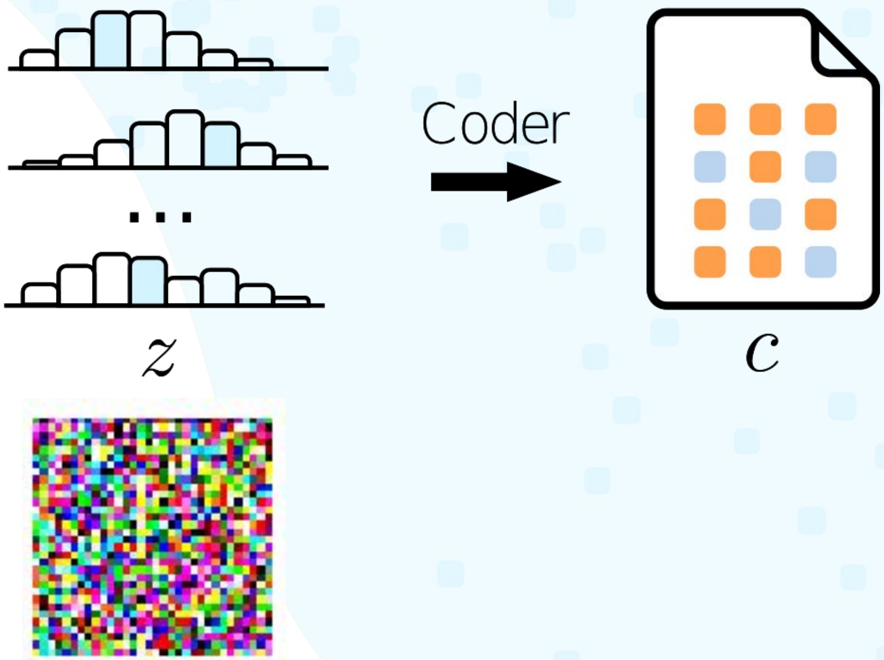


Lossless source compression

Step 2: encode z using off-the-shelve entropy encoder

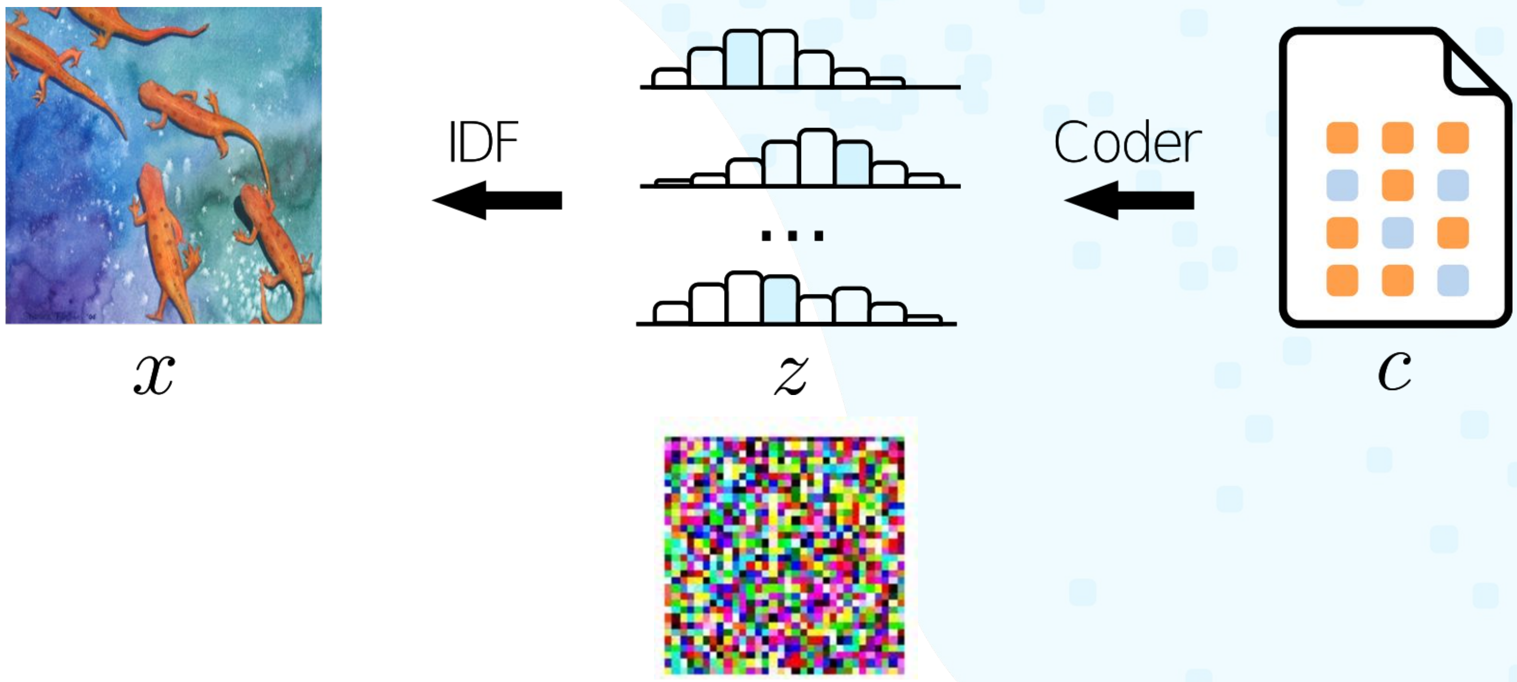
High-probability z → Short code

Low-probability z → Long code



Lossless source compression

Decompression works analogously in inverse order, using the inverse transformation: the entropy based decoder following by the inverse mapping defined by the IDF.



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

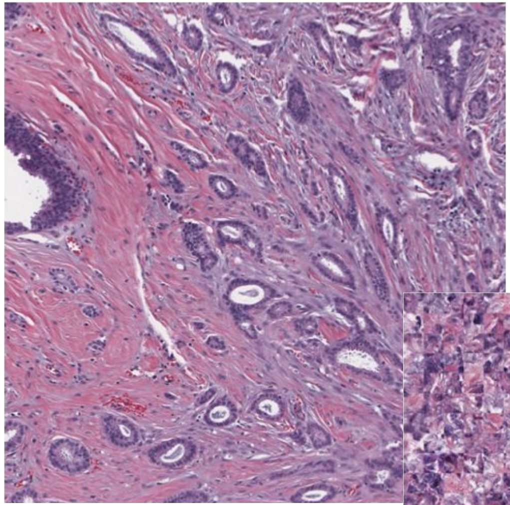
Medical data: Histology dataset

IDF trained on 80 x 80 px patches

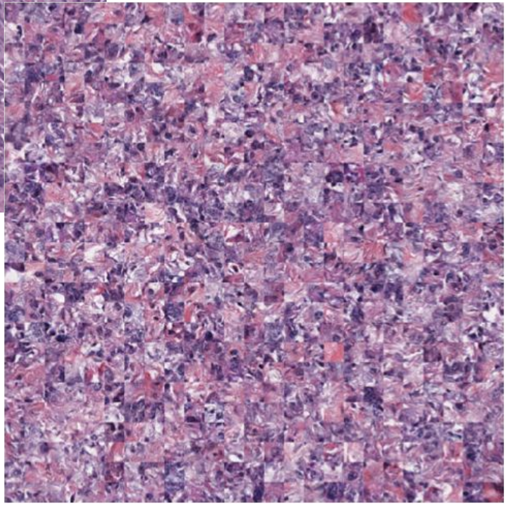
Compression is done patch-wise (each patch is considered independent)

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

Resolution: 2000 x 2000 pixels



Sampled patches: 80 x 80 pixels



Progressive image rendering

To partially render an image using IDFs, first the received variables are decoded. Next, using the hierarchical structure of the prior and ancestral sampling, the remaining dimensions are obtained. Below, the decoded images using 15, 30, 60, and 100% of the encoded data was user to decode the images.