Integer Discrete Flows and Lossless Compression

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

Lossless compression methods aim to reduce the size of data in expectation, using a statistical model. Flow-based models are attractive in this setting because they admit exact likelihood optimization, which is equivalent to minimizing the expected number of bits per message. Conventional flows learn continuous distribution, whereas entropy coders are based on *discrete* distributions. As a solution, we introduce a flow-based generative model for ordinal discrete data called Integer Discrete Flow (IDF). To the best of our knowledge, this is the first lossless compression method that uses invertible neural networks.

Background: Lossless Compression

In lossless compression one aims to use short codes for likely symbols and long codes for unlikely symbols. Since the likely symbols occur more frequently, the expected code length is short. The optimal length for a sample x is $-\log \mathcal{D}(x)$. The challenge lies in finding a good statistical models $p_X(x)$. In the example below we show that by taking the likelihood into account, one can obtain a shorter expected code.

Sym	code	p
a	00	1/2
b	01	1/4
c	10	1/8
d	11	1/8

$\frac{1}{2}\ell(00)+\frac{1}{4}\ell(01)+\frac{1}{8}\ell(10)+\frac{1}{8}\ell(11)=\mathbf{2\ bits}$

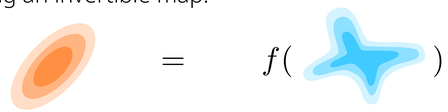
Sym	code	p
a	0	1/2
b	10	1/4
c	110	1/8
d	111	1/8

$\frac{1}{2}\ell(00)+\frac{1}{4}\ell(10)+\frac{1}{8}\ell(110)+\frac{1}{8}\ell(111)=\mathbf{1.75\ bits}$

The expected code length is equal to the log-likelihood objective. Hence, optimizing log-likelihood is equivalent to minimizing the expected code length.

Background: Normalizing Flows

Normalizing Flows are an attractive statistical model because they admit exact likelihood optimization. They map complicated distributions to simple distributions using an invertible map.



$p(z)$ $p(x)$

The likelihood can be computed using the change of variables formula:

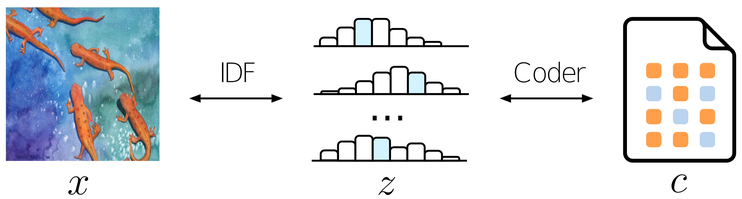
$$p_X(x) = p_Z(z)\left|\frac{\mathrm{d}z}{\mathrm{d}x}\right|, \quad z = f(x)$$

Integer Discrete Flows

We introduce integer discrete flows (IDFs) to make flows suitable for lossless compression. IDFs are invertible integer maps that can represent rich transformations. They can be used to learn the probability mass function on (high-dimensional) ordinal discrete data. The model distribution can then be expressed as:

$$p_X(x) = p_Z(z), \quad z = f(x)$$

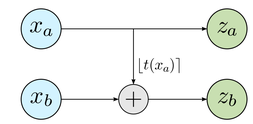
Because IDFs define discrete distributions, they can directly be used by an entropy encoder for lossless compression:



x z c

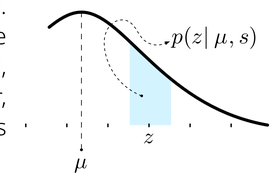
Integer Coupling

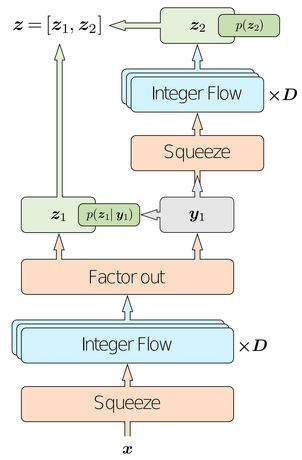
We modify the standard coupling layer by constraining the translations in the coupling layer to integers by rounding. A deep architecture is obtained by composing multiple integer coupling layers. As each coupling layer is an integer map, so is the composition.



Discrete Base Distribution

The discrete change of variables formula requires pZ(z) to be discrete. We propose the Discretized Logistic distribution. The discretized logistic captures the inductive bias that values close together are related, which is well-suited for ordinal data. Moreover, evaluation is cheap as the CDF of the logistic is related to the sigmoid.





Architecture

The IDF architecture utilizes splitpriors for hierarchical structure. Empirically we found this improves performance. Moreover, since a part of the flow operators on lower dimensional data, there is also a computational benefit.

In our experiments, we found that too many coupling layers are difficult to optimize, possibly due to the introduced gradient bias. To circumvent this, we use more expressive coupling layers while using fewer coupling layers in total.

Experiments

To test the compression performance of IDFs, we compare with a number of established lossless compression methods: PNG; FLIF, a recent format that uses machine learning to build decision trees for efficient coding; and Bit-Swap, a VAE based lossless compression method. We show that IDFs outperform all these formats on CIFAR10, ImageNet32 and ImageNet64.

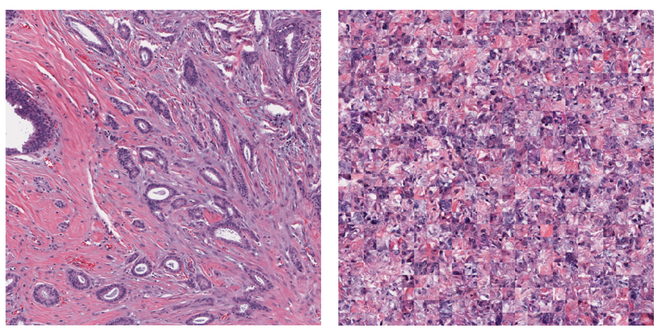
Dataset	IDF	IDF†	Bit-Swap	FLIF	PNG
Cifar10	**3.34 (2.40x)**	3.60 (2.22x)	3.82 (2.09x)	4.37 (1.83x)	5.89 (1.36x)
ImageNet32	**4.18 (1.91x)**	**4.18 (1.91x)**	4.50 (1.78x)	5.09 (1.57x)	6.54 (1.25x)
ImageNet64	**3.90 (2.05x)**	3.94 (2.03x)	—	4.55 (1.76x)	5.74 (1.39x)

IDF† indicates our IDF model trained on ImageNet32, but evaluated on the indicated dataset.

Tunable Compression

Without domain knowledge, we can learn a domain specific compressor using only data. We evaluate this on a histology dataset.

Dataset	IDF	JP2-WSI	FLIF	JPEG2000
Histology	**2.42 (3.19x)**	3.04 (2.63x)	4.00 (2.00x)	4.26 (1.88x)



Progressive Image Rendering

IDFs support progressive rendering naturally. To partially render an image using IDFs, first the received variables are decoded. All unknown remaining variables can be sampled from the IDF. Below we show various images decoded using approximately 15, 30, 60 and 100% of the bitstream.