Generative Modeling

Advanced Normalizing flows

Denis Derkach, Artem Ryzhikov, Maxim Artemev

Laboratory of methods for big data analysis

LAMBDA • HSE

Spring 2021

In this Lecture

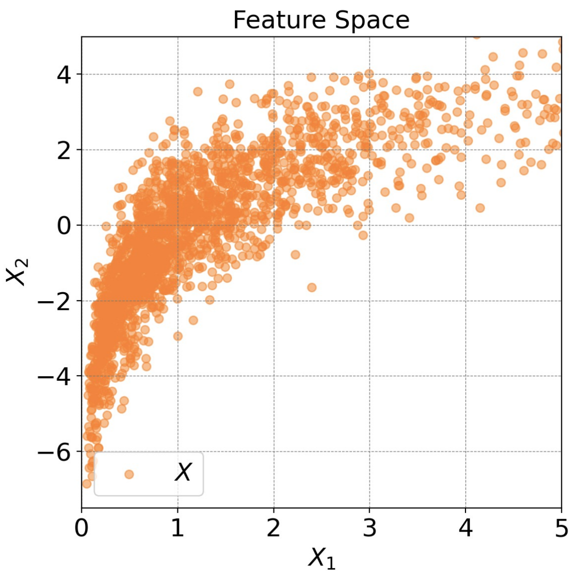
- Ordering in block models.
- Full Jacobian models.
- Continuous time models.

Reminder: Normalizing Flows

Problem Statement

$x_i \sim p_x(x)$ – data

$p_x(x)$ – ?

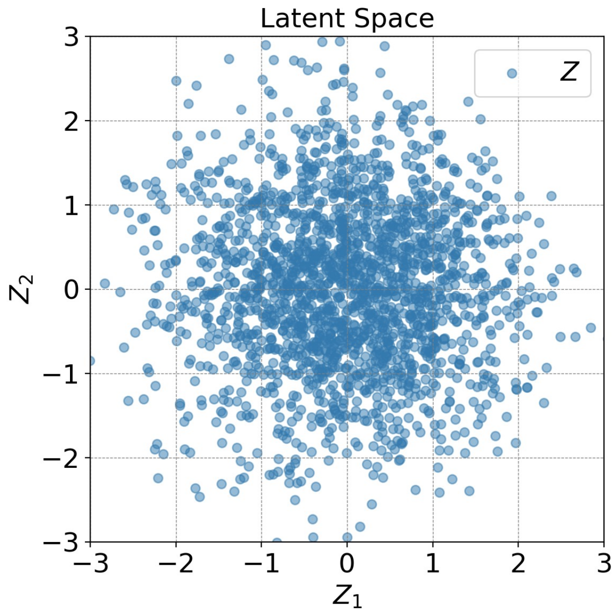


$z = f(x)$ – ?

$x = f^{-1}(z)$ - ?

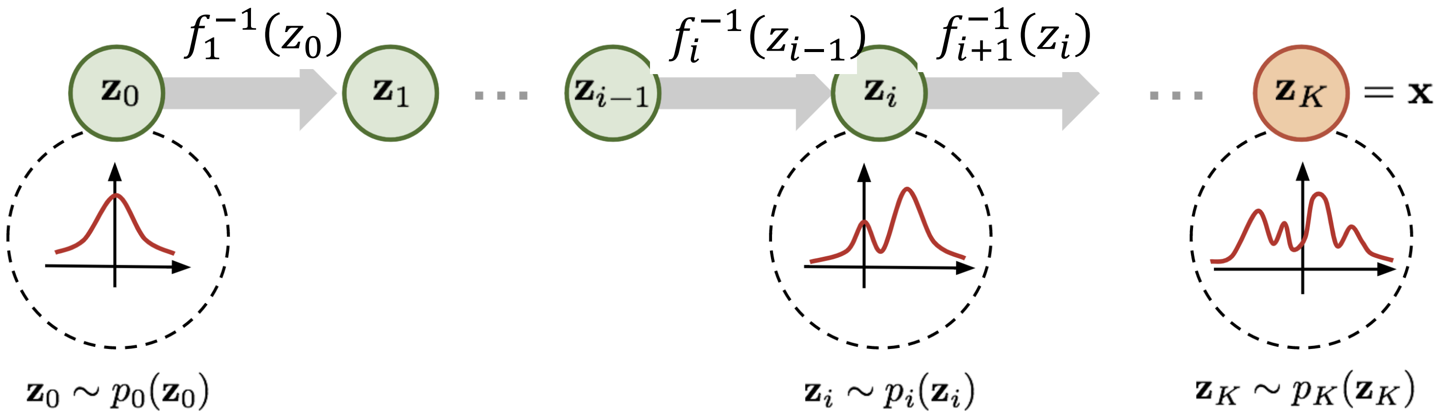
$z_i \sim p_z(z)$

$p_z(z)$ known



- **We have:** real objects $\{x_i\}$
- **Task:** find invertible and differentiable f: $z_i = f(x_i)$, such that $z_i \sim p_z(z)$. For some known: $p_z(z)$.

Flow discussion



$$p_x(x) = \boldsymbol{p}_{\boldsymbol{z}_0}\left(\boldsymbol{f}_{\boldsymbol{K}}\left(\ldots\left(\boldsymbol{f}_{\boldsymbol{1}}(x)\right)\right)\right)\left|\det\frac{\partial \boldsymbol{f}_{\boldsymbol{1}}(z_1)}{\partial z_1}\right| \ldots \left|\det\frac{\partial \boldsymbol{f}_{\boldsymbol{K}}(z_k)}{\partial z_k}\right|.$$

- It is possible to obtain $p_x(x)$ consecutively changing observables.
- The overall transformation is invertible if individual layers are invertible.
- Dimensions of each observable is the same.
- Fit is performed using ML estimate.
- Need to calculate determinant.

Block matrix for Jacobian

$$\frac{\partial \boldsymbol{f}(x)}{\partial x} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial z_{d+1:D}}{\partial x_{d+1:D}} & S \end{pmatrix}$$

- Randomly choose d such that we have two disjoint subsets of observables: $z_{1:d}$ and $z_{d+1:D}$.
- Insert a block transform.
- Repeat for several layers.
- If needed insert scaling layers.
- Fit simultaneously.

Real-NVP

$$z = \boldsymbol{f}(x) = \begin{cases} z_{1:d} = x_{1:d} \\ z_{d+1:D} = x_{d+1:D} \odot \exp\big(\boldsymbol{s}(x_{1:d})\big) + \boldsymbol{t}(x_{1:d}) \end{cases}$$

- $\boldsymbol{s}(x_{1:d})$ и $\boldsymbol{t}(x_{1:d})$ – **neural networks** with d inputs and $D - d$ outputs.
- Invertible.
- $\det J_k = \exp \sum_{i=d+1}^{D} \big(\alpha_\theta(z_{1:d})\big)_i$ for k-th layer.
- Inspired by RNADE.

https://arxiv.org/abs/1605.08803

Jacobian Choices

1. Det Identities

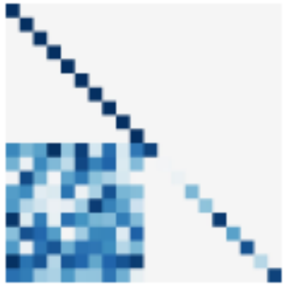
Planar NF
Sylvester NF
...

Jacobian

(Low rank)

2. Coupling Blocks

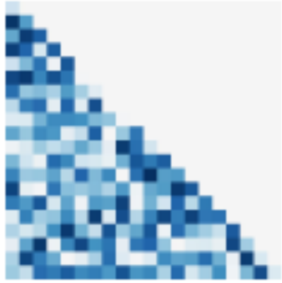
NICE
Real NVP
Glow
...



(Lower triangular + structured)

3. Autoregressive

Inverse AF
Neural AF
Masked AF
...

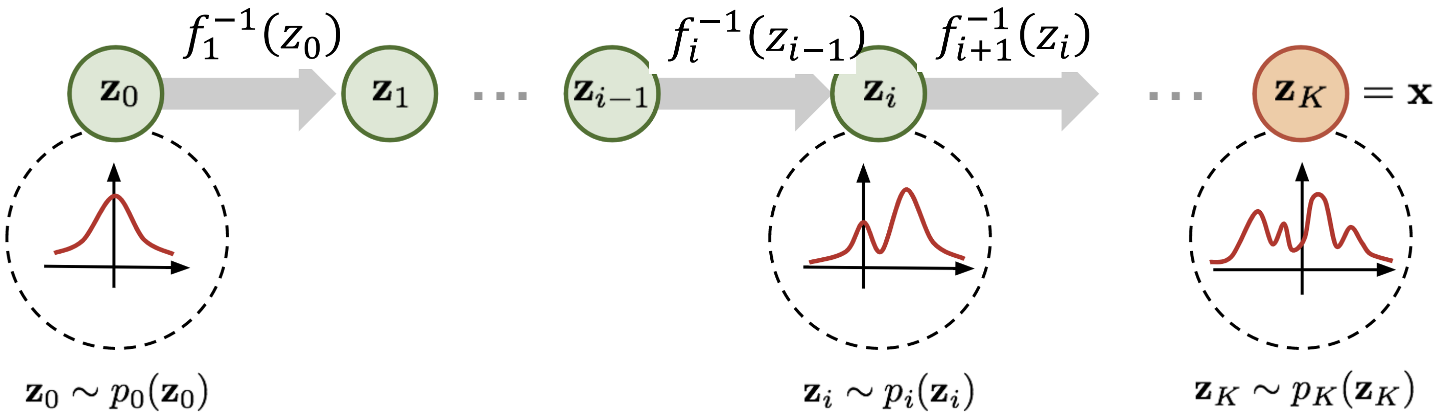


(Lower triangular)

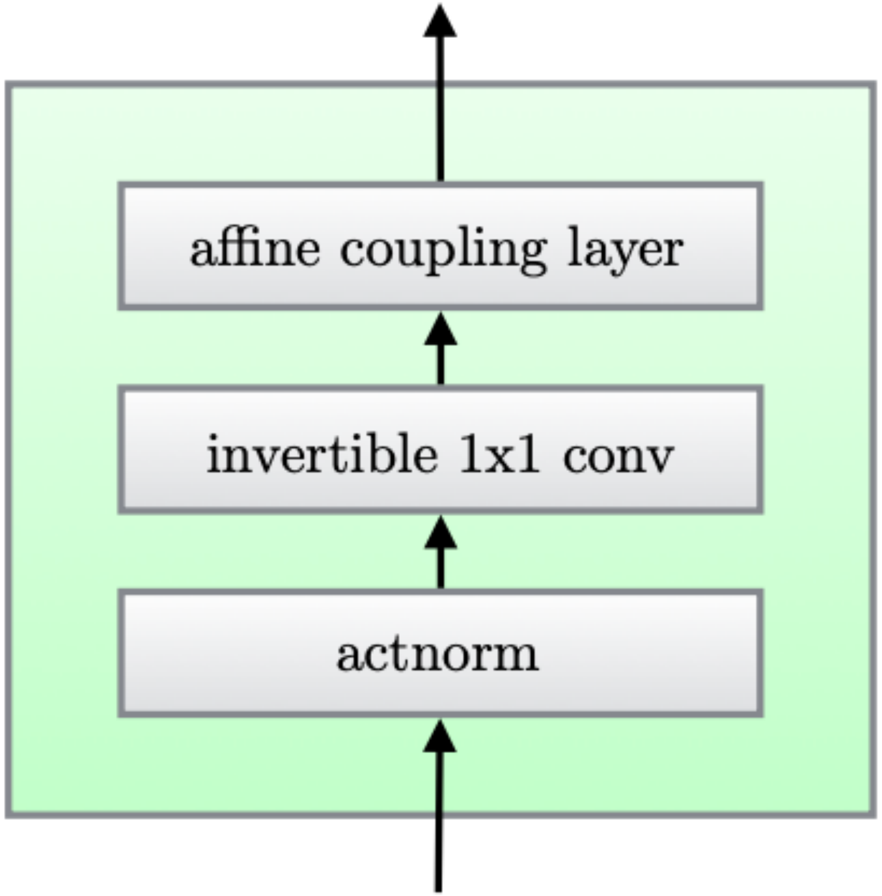
Generative Flow with Invertible 1×1 Convolutions

Motivation

- rNVP needs permutations to include all dimensions into consideration.
- This requires additional layers and several permutations.



One step of flow



- **Activation normalization.**
  - Trainable scale and bias
  $$z_{i,j} = \boldsymbol{s} \odot x_{i:j} + \boldsymbol{b}.$$
- **Invertible 1x1 convolution.**
  - Trainable weight matrix.
  $$z_{i,j} = \boldsymbol{W} x_{i:j}.$$
- **Affine coupling layer.**
  - Similar to rNVP.
  $$x_a, x_b = \text{split}(x)$$
  $$z_a = \boldsymbol{s} \odot x_a + \boldsymbol{t}.$$
  $$z_b = x_b.$$

https://arxiv.org/pdf/1807.03039.pdf

1x1 convolution layer

- Permutation is just a special case of a linear operator:

$$y = Wx = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} x.$$

- Convolution of an input h×w×c tensor **h** with a weight matrix W (c×c):

$$f = \mathrm{conv2d}(h; W).$$

  - Need change of variables formula:

$$\log|\det\left(\frac{df}{dh}\right)| = \log|\det(\frac{\mathrm{dconv2D}(\boldsymbol{h}; W)}{\mathrm{dh}})| = h \cdot w \log\det|W|$$

https://arxiv.org/pdf/1807.03039.pdf

Jacobian calculations

- PLU decomposition:

$$W = PL\big(U + \mathrm{diag}(s)\big).$$

$$\log|\det W| = \sum \log|s|.$$

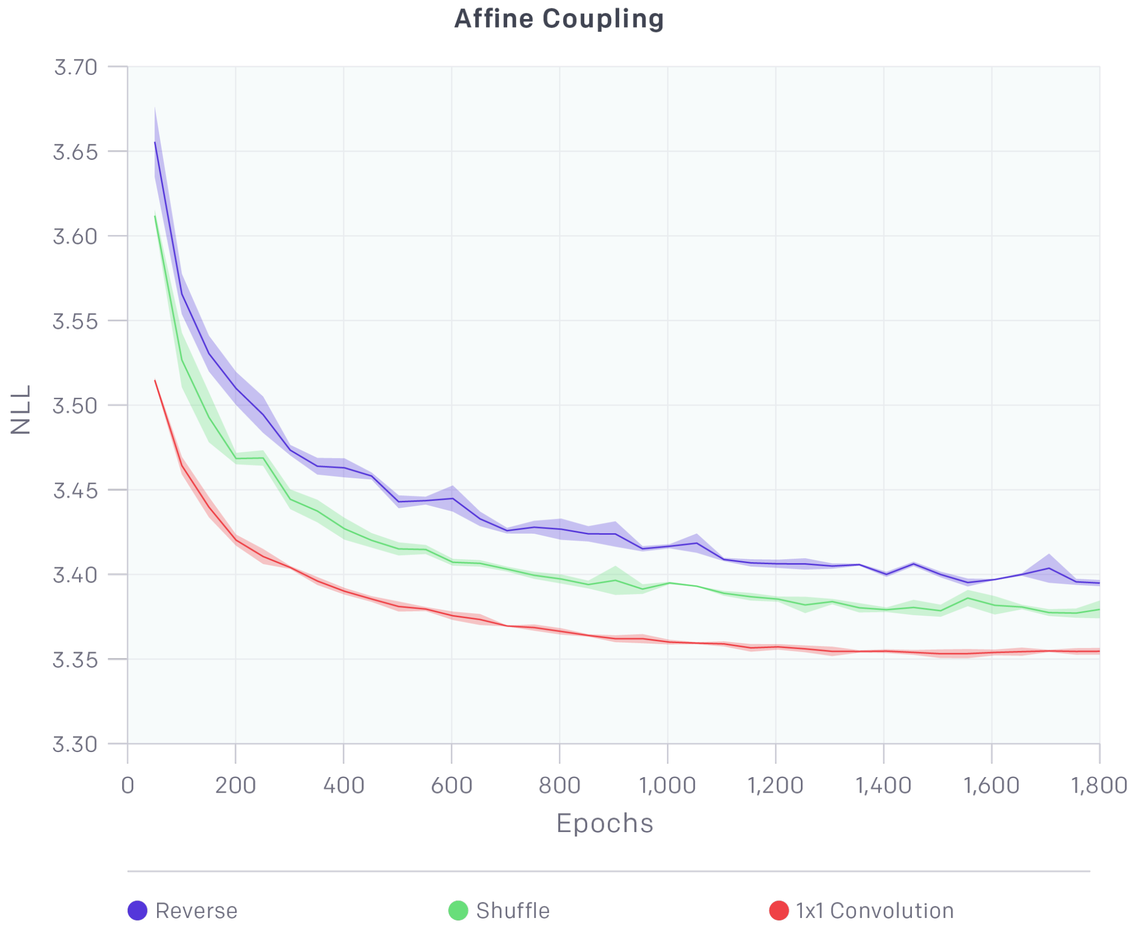
Initiate from random W.

P is permutation (remains fixed), L, U, and s are optimized

https://arxiv.org/pdf/1807.03039.pdf

Affine coupling

- Faster to converge than additive.
- 1x1 convolution performs like better randomization.



https://slideslive.com/38917897/glow-generative-flow-with-invertible-1x1-convolutions

Sampling Temperature

- In order to get more realistic sampling, one can use a reduced-temperature model.
- In this work:

$$p_{\theta,T}(x) \sim p_{\theta}^{T^2}(x)$$

Temperature is a free parameter for sampling.

https://arxiv.org/abs/1702.00783

Temperature Dependence

Figure 8: Effect of change of temperature. From left to right, samples obtained at temperatures $0, 0.25, 0.6, 0.7, 0.8, 0.9, 1.0$

GLOW: results

Figure 4: Random samples from the model, with temperature 0.7

GLOW: depth dependence

Figure 9: Samples from shallow model on left vs deep model on right. Shallow model has $L = 4$ levels, while deep model has $L = 6$ levels

Conclusions

- Addresses problem of choosing permutation with 1x1 convolutional layer.
- Uses triangular Jacobian idea.
- Quite slow.

https://arxiv.org/abs/1702.00783

More Linear Flows

- QR Flows:

$$W = QR$$

Q is orthogonal, R is upper triangular.

$$\det W = \prod R_{ii}$$

- Orthogonal flows

W – orthogonal, $\det W = 1$

https://arxiv.org/pdf/1912.02762.pdf

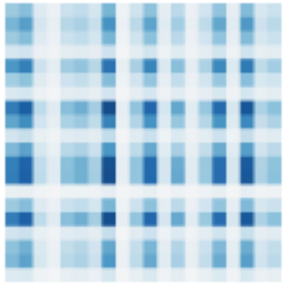
Residual Flows

Jacobian Choices

1. Det Identities

Planar NF
Sylvester NF
...

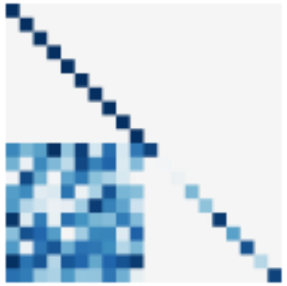
Jacobian



(Low rank)

2. Coupling Blocks

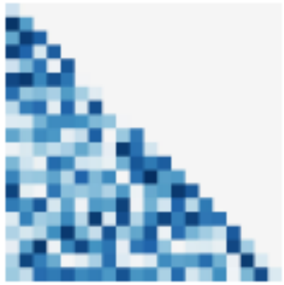
NICE
Real NVP
Glow
...



(Lower triangular + structured)

3. Autoregressive

Inverse AF
Neural AF
Masked AF
...



(Lower triangular)

4. Unbiased Estimation

FFJORD
Residual Flows

(Arbitrary)

Invertible Residual Networks (i-ResNet)

Residual blocks:

$$y = F(x) = x + \boldsymbol{g}(x)$$

can be inverted by **fixed point iteration**:

$$x = y - \boldsymbol{g}(x)$$

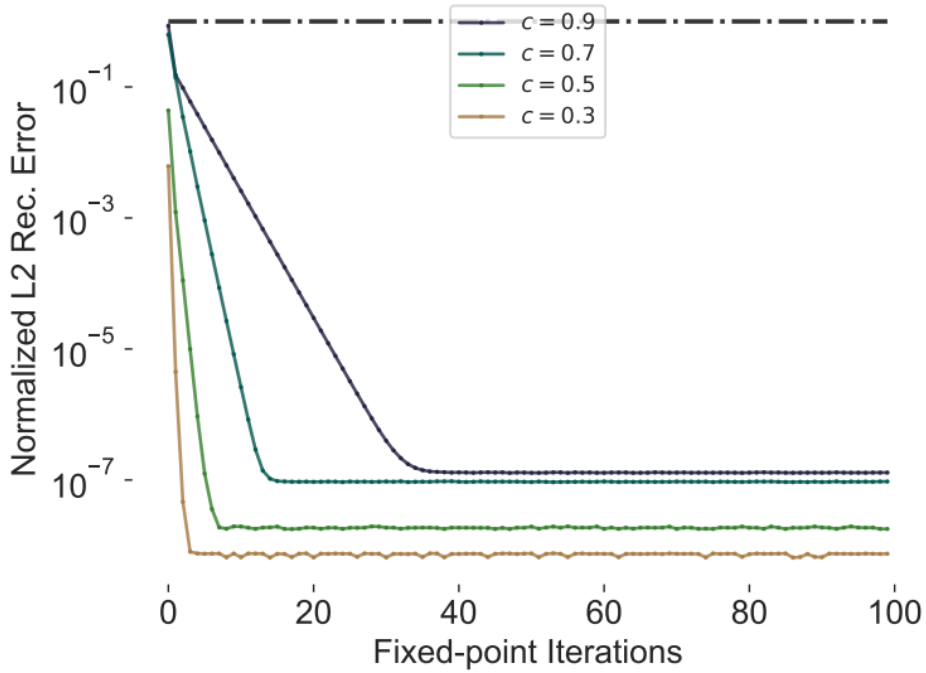
and has unique inverse if

$$\mathrm{Lip}(\boldsymbol{g}) < 1$$

Thus, effectively making $\boldsymbol{g}$ contractive.

Algorithm 1. Inverse of i-ResNet layer via fixed-point iteration.

Input: output from residual layer y, contractive residual block g, number of fixed-point iterations n
Init: $x^0 := y$
for $i = 0, \ldots, n$ **do**
  $x^{i+1} := y - g(x^i)$
end for



https://arxiv.org/abs/1811.00995

Invertible Residual Networks (i-ResNet)

Residual blocks:

$$y = F(x) = x + \boldsymbol{g}(x)$$

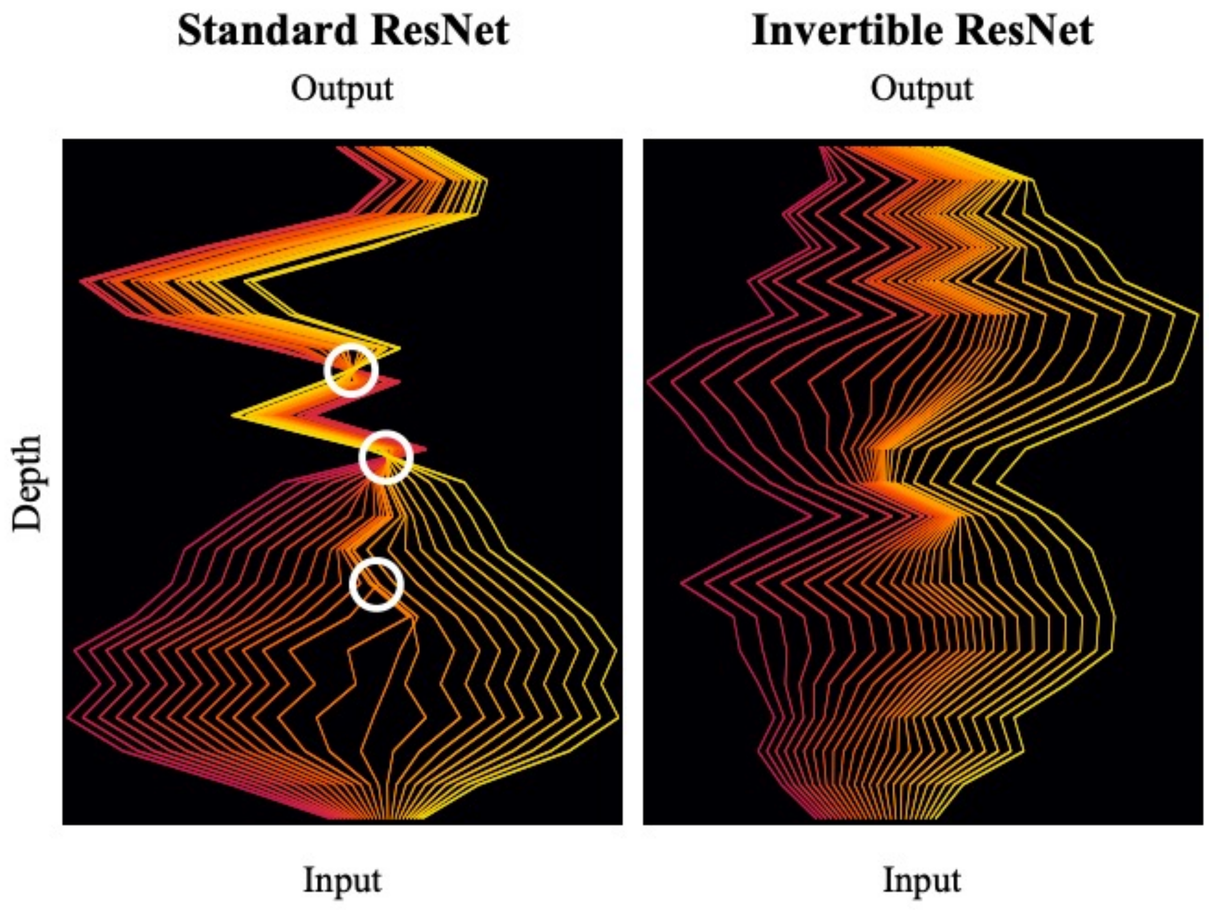
can be inverted by **fixed point iteration**:

$$x = y - \boldsymbol{g}(x)$$

and has unique inverse if

$$\mathrm{Lip}(\boldsymbol{g}) < 1$$

Thus, effectively making $\boldsymbol{g}$ contractive.



https://arxiv.org/abs/1811.00995

Satisfying Lipschitz Condition on g(x)

- Contruct $\boldsymbol{g}$ as a network

$$\boldsymbol{g} = W_3 \circ \phi \circ W_2 \circ \phi \circ W_1 \circ \phi.$$

- Can construct data independent Lip-constraint:

$$\mathrm{Lip}(g) < \left||W_3|\right|_2 \left||W_2|\right|_2 \left||W_1|\right|_2$$

- Use spectral norm of weight matrices (σ the largest singular layer):

$$\widetilde{W} = \frac{cW}{\sigma}$$

https://arxiv.org/abs/1811.00995

Satisfying Lipschitz Condition on g(x)

Figure 3. Original images (top) and reconstructions from i-ResNet with $c = 0.9$ (middle) and a standard ResNet with the same architecture (bottom), showing that the fixed point iteration does not recover the input without the Lipschitz constraint.

https://arxiv.org/abs/1811.00995

Change of Variables Formula

$$\log\ p_X(x) = \log \boldsymbol{p_Z}(F(x)) + \log|\det J_F(x)|$$

- Need a way to compute Jacobian:

$$F = (I + g)x.$$

- Insert into the calculations and $\ln \det A = \mathrm{tr}(\log A)$):

$$\log\ p_X(x) = \log \boldsymbol{p_Z}(F(x)) + \mathrm{tr}\ (\log\ (I + J_g(x)).$$

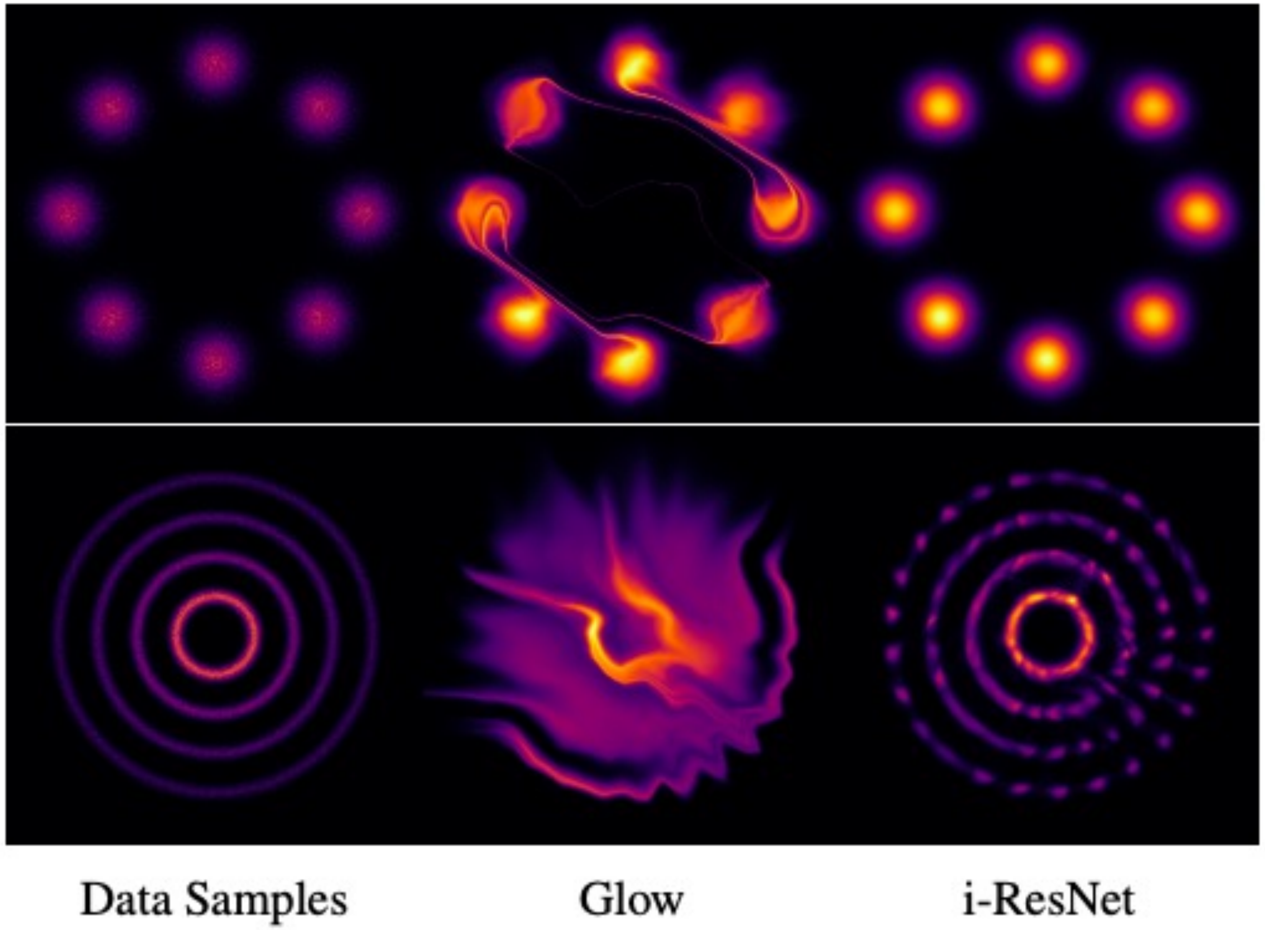
- Trace of the matrix logarithm can be expressed as power series (for constrained $||J_g|| < 2$):

$$\log\ p_X(x) = \log \boldsymbol{p_Z}(F(x)) + \sum_{k=1}^{\infty}(-1)^{k+1}\frac{tr(J_g^k)}{k}.$$

- The sum can be truncated and reduced to biased Hutchinson estimate.

Invertible Residual Networks (i-ResNet)

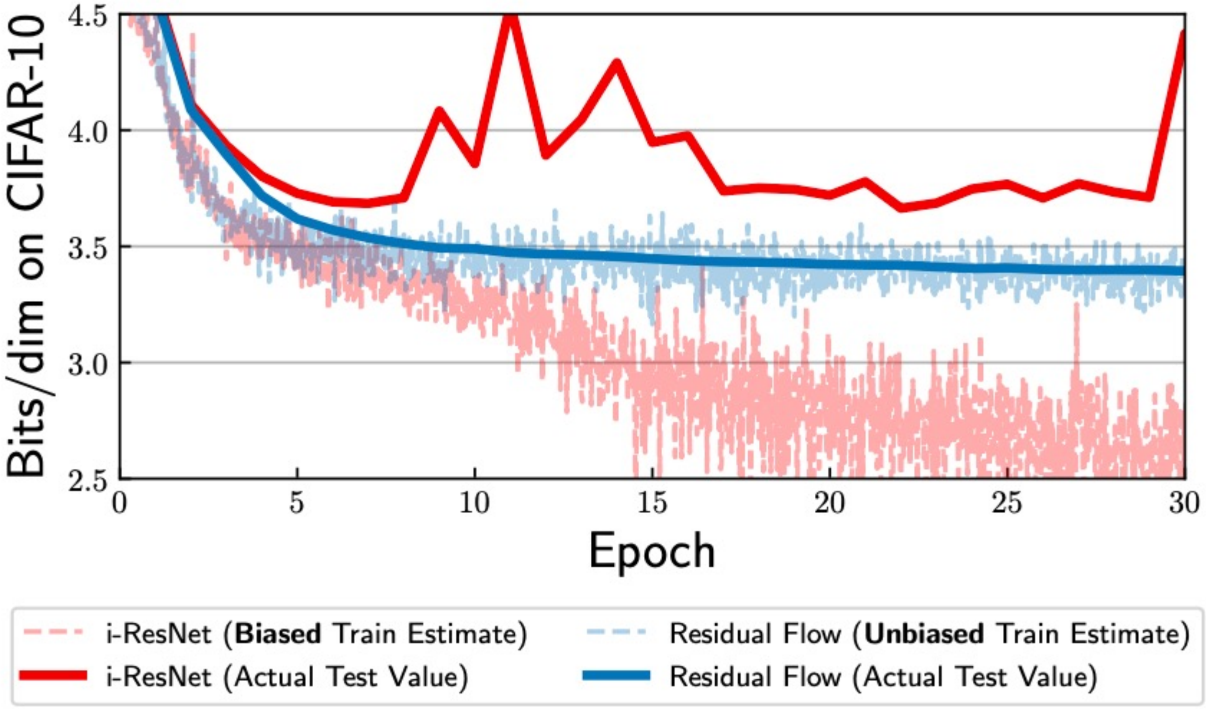
- Performs better than glow.
- Still some troubles are visible.
- Can we estimate in a different way?



Residual Flows

- Use Russian roulette estimate for infinite series:

$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{n \sim p(N)} \left[\sum_{k=1}^{n} \frac{\Delta_k}{\mathbb{P}(N \geq k)} \right]$$



https://arxiv.org/pdf/1906.02735.pdf

Residual Flows results

Figure 5: **Qualitative samples.** Real (left) and random samples (right) from a model trained on 5bit 64×64 CelebA. The most visually appealing samples were picked out of 5 random batches.

Table 2: Lower FID implies better sample quality. *Results taken from Ostrovski et al. (2018).

Model	CIFAR10 FID
PixelCNN*	65.93
PixelIQN*	49.46
i-ResNet	65.01
Glow	46.90
Residual Flow	**46.37**
DCGAN*	37.11
WGAN-GP*	36.40

- Performs better than many flow models.
- Still worse than GANs

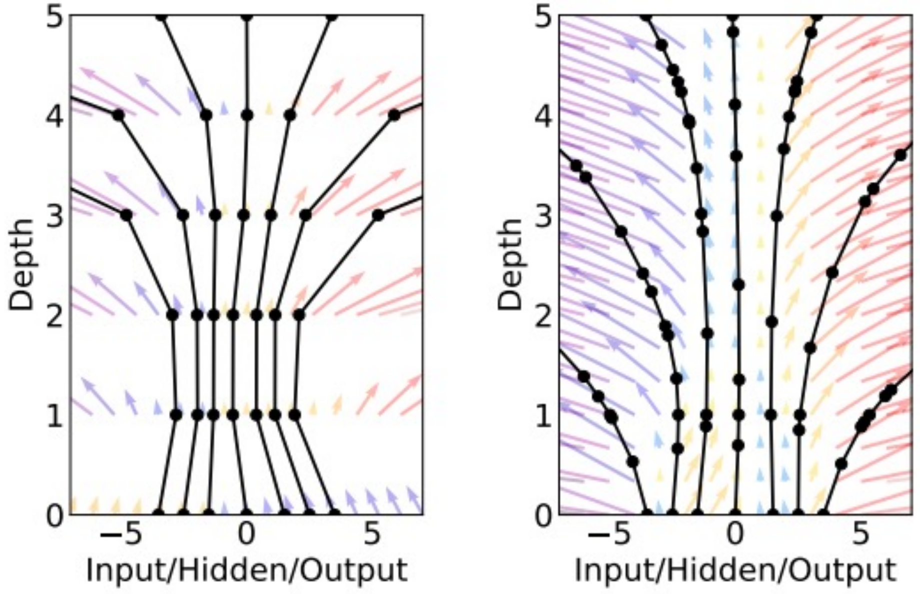
Residual Flows Discussion

- RF has dense Jacobian, thus are more flexible than AR flows.
- Density evaluation needs to be done iteratively, thus is slow.

https://www.cs.toronto.edu/~rtqichen/pdfs/residual_flows_
slides.pdf

FFJORD

Motivation

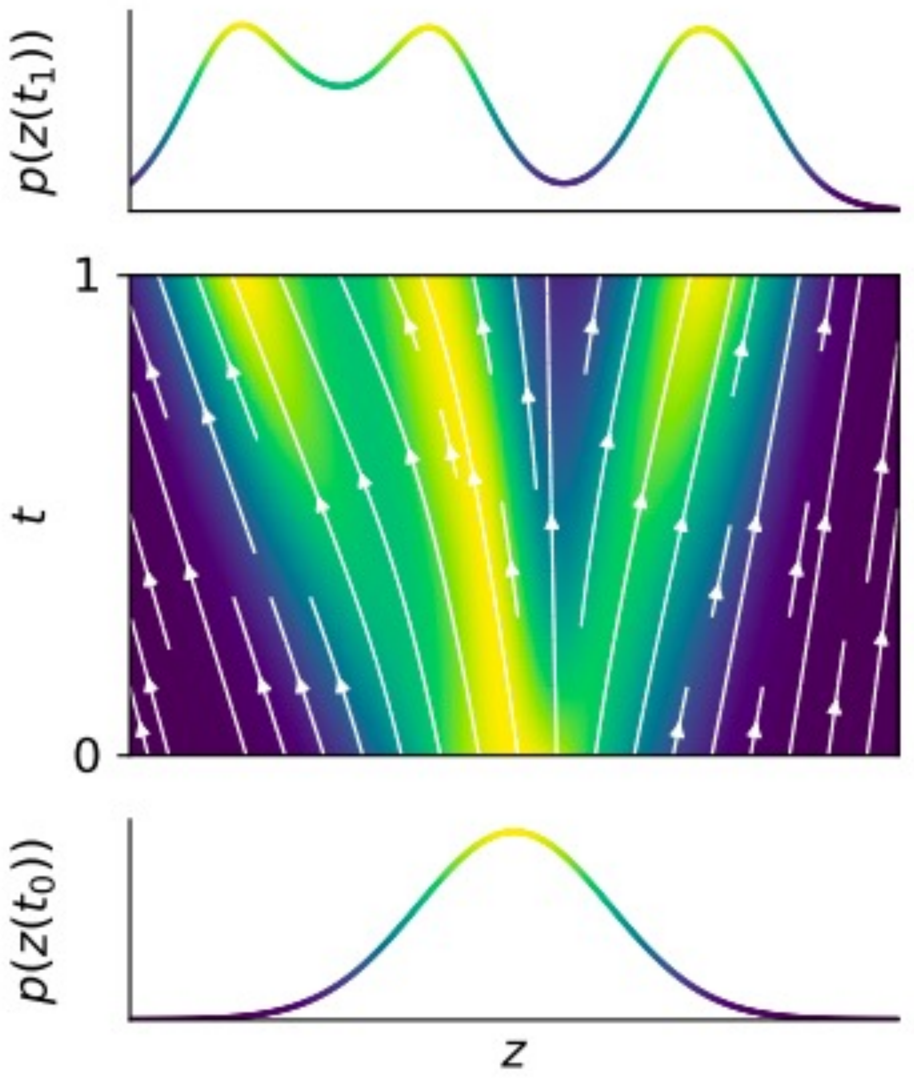


- Do we really care of having discrete steps?
- Can we change the Jacobian to something more stochastic?
- System of continuous-time dynamics.
- These ideas led to the NeuralODE model.

https://arxiv.org/abs/1806.07366

https://stanniszhou.github.io/discussion-group/post/ffjord/

Continuous Normalizing Flows



- Model generative process with continuous dynamics:

$$z_0 \sim p(z_0)$$

$$\frac{\partial z}{\partial t} = f_\theta(z_t, t)$$

$$x = z_t = z_0 + \int_{t_0}^{t_1} f_\theta(z_t, t)dt$$

Instantaneous Change-of-variable Formula

- For f uniformly Lipschitz continuous in z and continuous in t

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}}(t)\right)$$

- The complete initial value problem is then given as

$$\frac{d}{dt}\begin{bmatrix} \mathbf{z}(t) \\ \log p(\mathbf{z}(t), t) \end{bmatrix} = \begin{bmatrix} f(\mathbf{z}(t), t, \boldsymbol{\theta}) \\ -\mathrm{tr}\, \partial_{\mathbf{z}} f \end{bmatrix}, \quad t \in [0, T],$$

$$\mathbf{z}(0) \sim p_{\mathbf{z}_0}(\mathbf{z}(0)),$$

$$\log p(\mathbf{z}(0), 0) = \log p_{\mathbf{z}_0}(\mathbf{z}(0)).$$

Unbiased Log-Density Estimation

- log-probability of the data under continuous model

$$\log(p_x) = \log(p_{z_0}) - \int_0^1 \mathrm{Tr}\frac{\partial f(z(t))}{\partial z(t)}dt$$

- This gives $O(N^3)$ calculations.
- Need a smart trick to increase the speed of calculations.

$$\begin{aligned}
\log p(\mathbf{z}(t_1)) &= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt \\
&= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon}\right] dt \\
&= \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\int_{t_0}^{t_1} \boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon} dt\right]
\end{aligned}$$

FFJORD: results

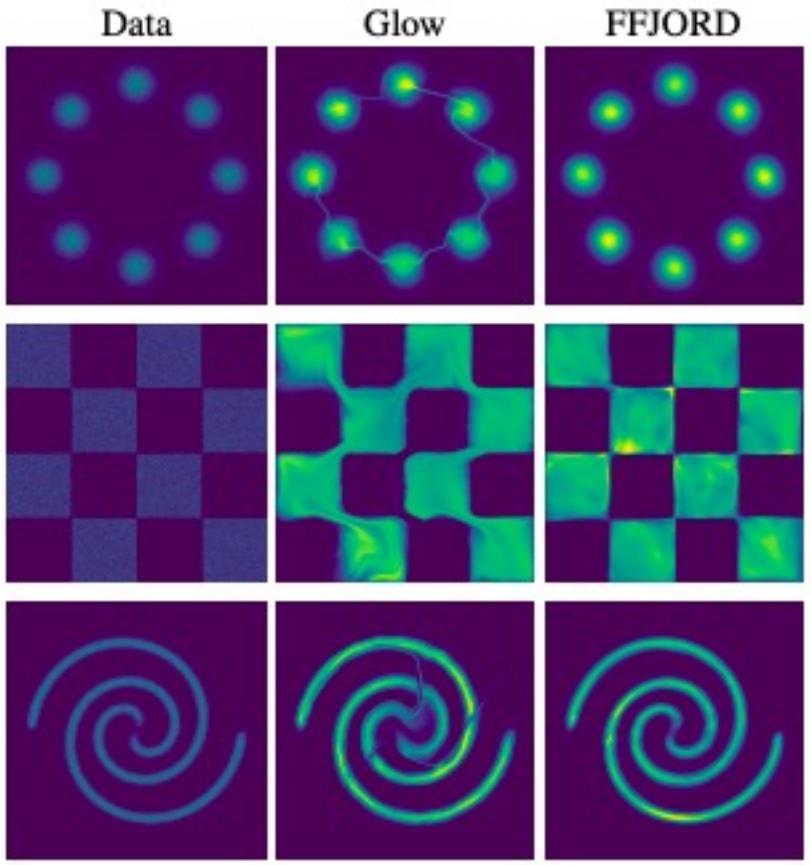


Figure 2: Comparison of trained FFJORD and Glow models on 2-dimensional distributions including multi-modal and discontinuous densities.

	POWER	GAS	HEPMASS	MINIB	BSDS	MNIST	CIFAR10
Real NVP	-.17	-8.33	18.71	13.55	-153.28	1.06	3.49
Glow	-.17	-8.15	18.92	11.35	-155.07	1.05	**3.35**
FFJORD	**-.46**	**-8.59**	**15.26**	**10.43**	**-157.67**	**0.99***	3.40
MADE	3.08	-3.56	20.98	15.59	-148.85	2.04	5.67
MAF	-.24	-10.08	17.70	11.75	-155.69	1.89	4.31
TAN	-.48	-11.19	15.12	11.01	-157.03	-	-
DDSF	-.62	-11.96	15.09	8.86	-157.73	-	-

Table: Density estimation experiments. Negative log-likelihood on test set.

- Performs better than many flow models.

FFJORD: discussion

- Advantages
  - Guaranteed inverse regardless of model parameterization
  - Efficient, unbiased log-probability estimation without restricting the Jacobian of the transformation
  - Reversible generative models can now be defined with standard neural network architectures.
- Disadvantages
  - Relies on adaptive numerical ODE solvers for stable training
  - Computation time determined by solver, not user
  - 4-5x slower than other reversible generative models (Glow, Real-NVP)

FFJORD: discussion

	Method	Train on data	One-pass Sampling	Exact log-likelihood	Free-form Jacobian
	Variational Autoencoders	✓	✓	✗	✓
	Generative Adversarial Nets	✓	✓	✗	✓
	Likelihood-based Autoregressive	✓	✗	✓	✗
Change of Variables	Normalizing Flows	✗	✓	✓	✗
Change of Variables	Reverse-NF, MAF, TAN	✓	✗	✓	✗
Change of Variables	NICE, Real NVP, Glow, Planar CNF	✓	✓	✓	✗
Change of Variables	**FFJORD**	✓	✓	✓	✓

Table 1: A comparison of the abilities of generative modeling approaches.

Conclusions

Normalizing flows: conclusions

- Advantages:
  - Explicit likelihood.
  - Straightforward sampling.
- Disadvantages:
  - Less realistic samples than GANs.
  - Some models have suppressed one of the advantages.