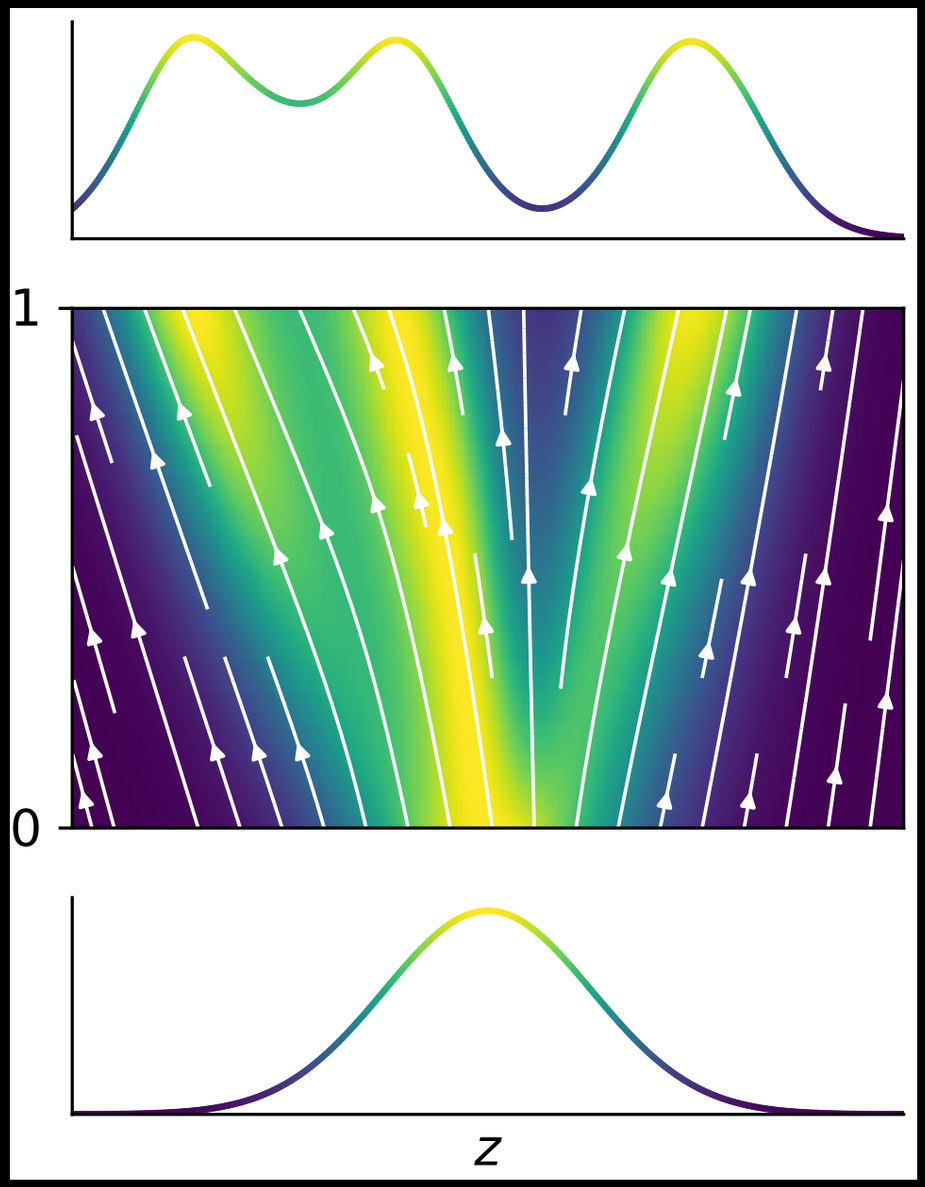
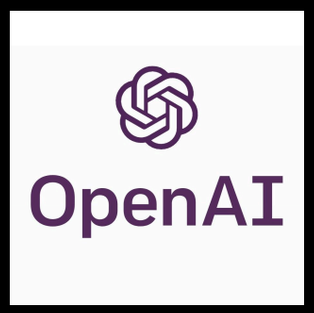
FFJORD:
reversible generative models with unrestricted architectures



FFJORD:
reversible generative models with unrestricted architectures

Will Grathwohl Ricky Chen
Jesse Bettencourt
Ilya Sutskever
David Duvenaud



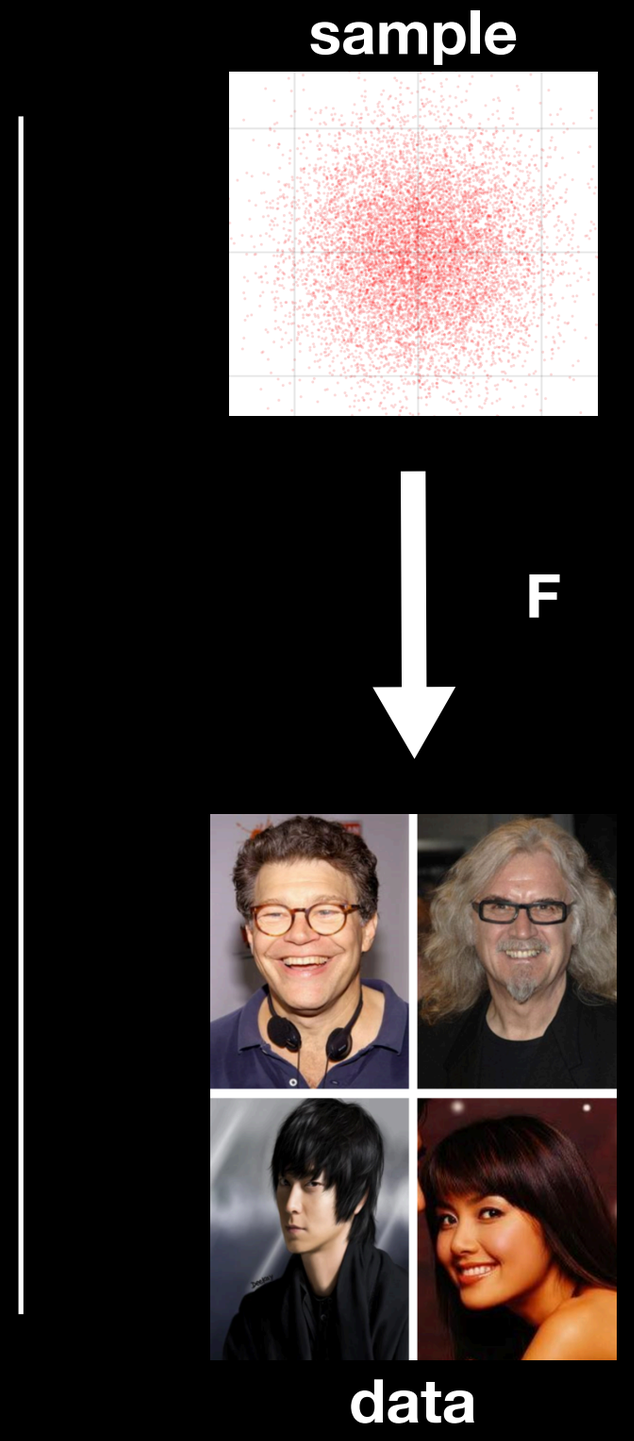
generative modeling with the change of variables formula

we can define a simple generative model as

$$z \sim p(z)$$
$$x = F_\theta(z)$$

if f is invertible…

$$\log p(x) = \log p(F_\theta^{-1}(x)) - \log |\partial F_\theta / \partial x|$$



challenges

Requirements:

F_θ is invertible

$\log |\partial F_\theta / \partial x|$ can be computed efficiently

Problem:

we cannot easily compute or estimate the determinant of arbitrary functions

Previous Solution:

use restricted network architectures to deliver these properties

Examples:

NICE (Dinh et al. 2014),
Real-NVP (Dinh et al. 2016),
Glow (Kingma & Dhariwal 2018)

Example: Real-NVP

forward

$$x = [x_a; x_b]$$

$$F_\theta^t(x) = [x_a; x_b \cdot s_\theta^t(x_a) + t_\theta^t(x_a)]$$

$$F_\theta(x) = F_\theta^T \circ \cdots \circ F_\theta^1(x)$$

inverse

$$z = [z_a; z_b]$$

$$(F_\theta^t)^{-1}(z) = [z_a; (z_b - t_\theta^t(z_a))/s_\theta^t(z_b)]$$

log-determinant

$$\log |\partial F / \partial x| = \sum_{t=1}^{T} \sum_{i=1}^{N} \log s_\theta^t(z_t)_i$$

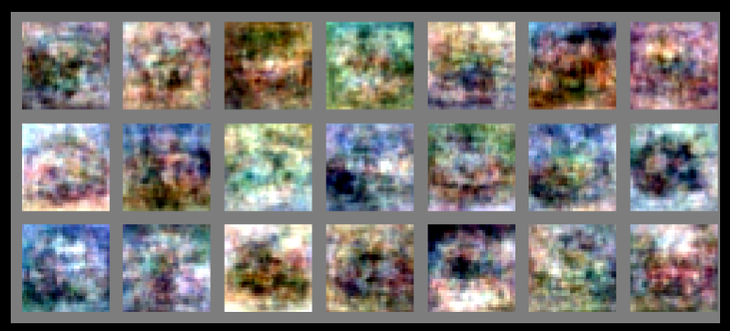
realities

each transformation is simple $\Longrightarrow$ many must be composed

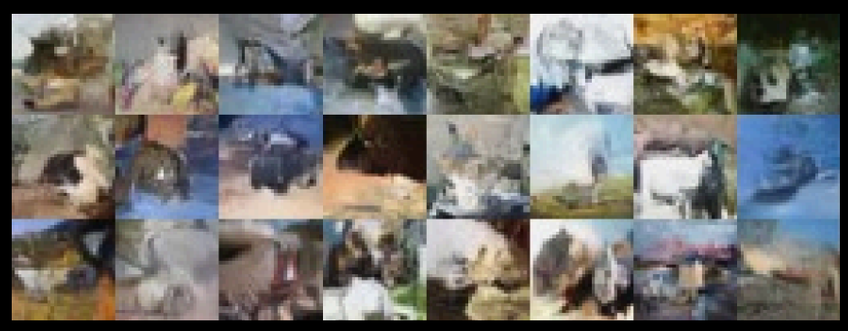
SOTA Glow model trained on cifar-10 has 400 layers and over 100M parameters

despite these drawbacks considerable progress has been made in recent years

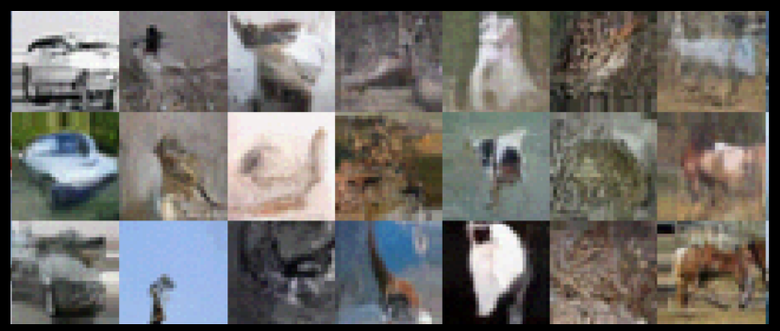
NICE (2014)



real-nvp (2016)



Glow (2018)



an alternate view

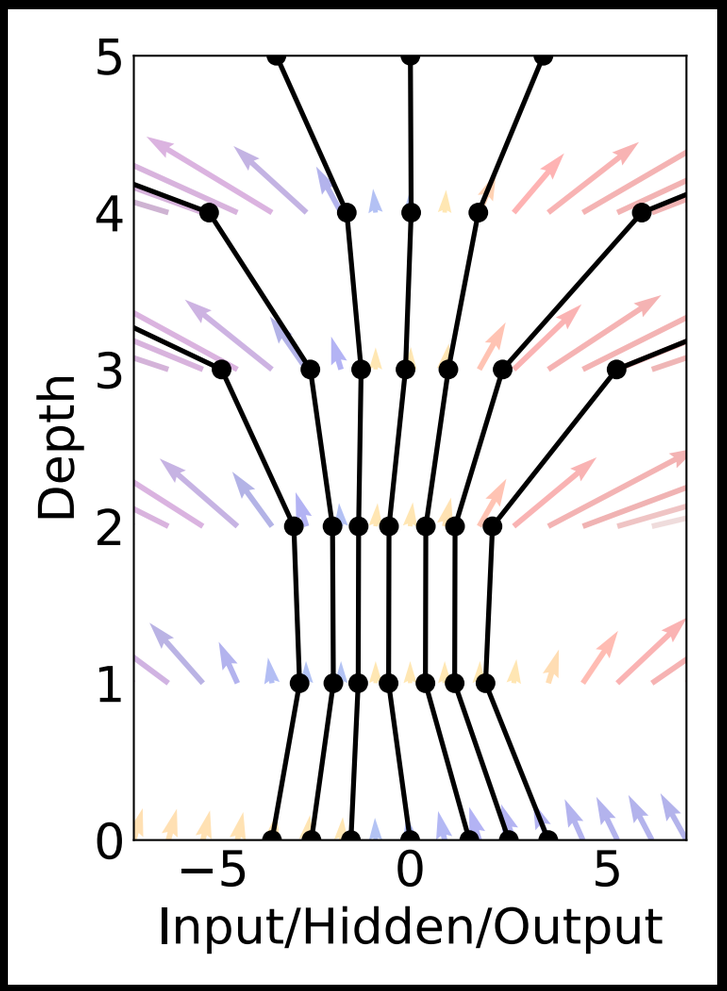
most reversible generative models compose many small building blocks

$$F_\theta(x) = F_\theta^T \circ \cdots \circ F_\theta^1(x)$$

this can be thought of as a discrete-time dynamics process

$$x = z_0$$
$$z_t = F_\theta^t(z_{t-1})$$
$$F_\theta(x) = z_T$$

$$\log p(x) = \log p(z_T) + \sum_{t=0}^{T} \log |\partial F^t / \partial z_t|$$



(Chen et al. 2018)

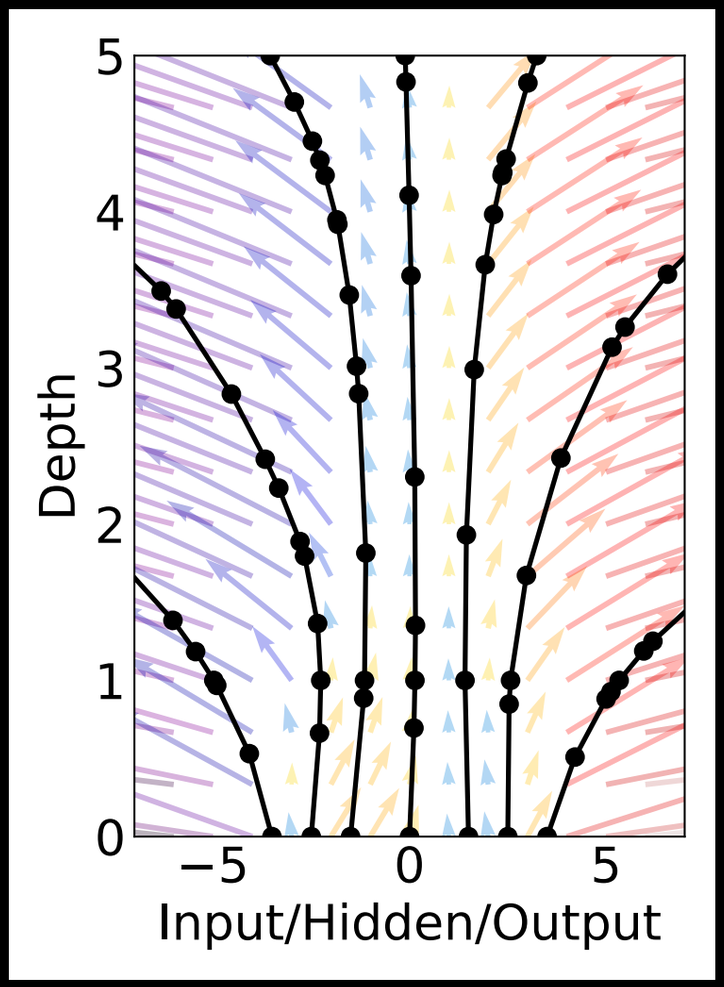
take a few limits...

if we replace these discrete-time dynamics with a continuous-time process something interesting happens

$$x = z_0$$

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

$$z_T = z_0 + \int_0^T f_\theta(z_t, t) dt$$



under this model:

$$\log p(x) = \log p(z_T) + \int_0^1 \mathrm{Tr}\left(\frac{\partial f}{\partial z_t}\right) dt$$

known as the instantaneous change of variables introduced in Neural ODEs (Chen, Rubanova, Bettencourt, Duvenaud, 2018)

what this means

log-probability of the data under the ***discrete*** model

$$\log p(x) = \log p(z_T) + \sum_{t=0}^{T} \log \left| \partial F^t / \partial z_t \right|$$

log-probability of the data under the continuous model

$$\log p(x) = \log p(z_T) + \int_0^1 \operatorname{Tr} \left(\frac{\partial f}{\partial z_t} \right) dt$$

sum of jacobian log-determinants $\Longrightarrow$ integral of jacobian trace

computational considerations

for a general function $f : \mathcal{R}^N \to \mathcal{R}^N$:

Quantity	Time-cost
$f(x)$	$O(N)$
$\partial f / \partial x$	$O(N^2)$
$\log \lvert \partial f / \partial x \rvert$	$O(N^3)$
$\mathrm{Tr}(\partial f / \partial z_t)$	$O(N^2)$

vjp's and stochastic trace estimation

$\partial f/\partial x$ requires $\mathcal{O}(N^2)$ to compute using automatic differentiation
but $e^T(\partial f/\partial x)$ can be computed in for any e in $\mathcal{O}(N)$

for any matrix A, we have

$$\mathrm{Tr}(A) = \mathbb{E}_{p(e)}[e^T A e]$$ (Hutchinson's estimator)

if $\mathbb{E}[e] = 0, \mathrm{Cov}(e) = I$

applying to the jacobian we have

$$\mathrm{Tr}(\partial f/\partial z_t) = \mathbb{E}_{p(e)}\left[e^T(\partial f/\partial z)e\right]$$ (vector-jacobian products)

which can be estimated in $O(N)$

3-line tf implementation

```
dfdz = f(z, t)
e = tf.random_normal(tf.shape(z))
div = tf.reduce_sum(
      tf.gradients(dfdz, z, grad_ys=e) * e
)
```

unbiased log-likelihood estimation

stochastic trace estimates can be incorporated into the instantaneous change of variables with

$$\log p(x) = \log p(z_T) + \int_0^1 \mathrm{Tr}\left(\frac{\partial f}{\partial z_t}\right) dt$$

$$= \log p(z_T) + \int_0^1 \mathbb{E}_{p(e)}\left[e^T \frac{\partial f}{\partial z_t} e\right] dt$$

$$= \log p(z_T) + \mathbb{E}_{p(e)}\left[\int_0^1 e^T \frac{\partial f}{\partial z_t} e dt\right] \quad \textbf{(swap order of integration)}$$

we can sample a single e and integrate the divergence estimates to obtain an unbiased estimate of log p(x) for unrestricted f

that’s all fine and dandy but…

our model represents the gradients of a continuous-time process

z_T is the solution to an ordinary differential equation (ODE) initial value problem (IVP)

can compute with numerical ODE-solver

how to compute $\partial L / \partial \theta$?

neural ODEs

recent work from Chen et al. (2018) provides a solution

given an objective

$$L(z(t_1)) = L\left(\int_{t_0}^{t_1} f(z(t), t, \theta)dt\right)$$
$$= L(\mathrm{ODESolve}(z(t_0), f, t_0, t_1, \theta))$$

$\frac{\partial L}{\partial \theta}$ can be found by solving a different IVP backwards in time

adjoint sensitivity

define new quantity, the adjoint: $a(t) = -\frac{\partial L}{\partial z(t)}$

it is governed by dynamics: $\frac{\partial a(t)}{\partial t} = -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z(t)}$

derivatives of original system are solutions IVPs based on these dynamics

$$\frac{\partial L}{\partial \theta} = \int_{t_1}^{t_0} a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt$$

putting it all together

in this work we define a generative model for data

$$z_0 \sim p(z_0)$$

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

$$x = z_1$$

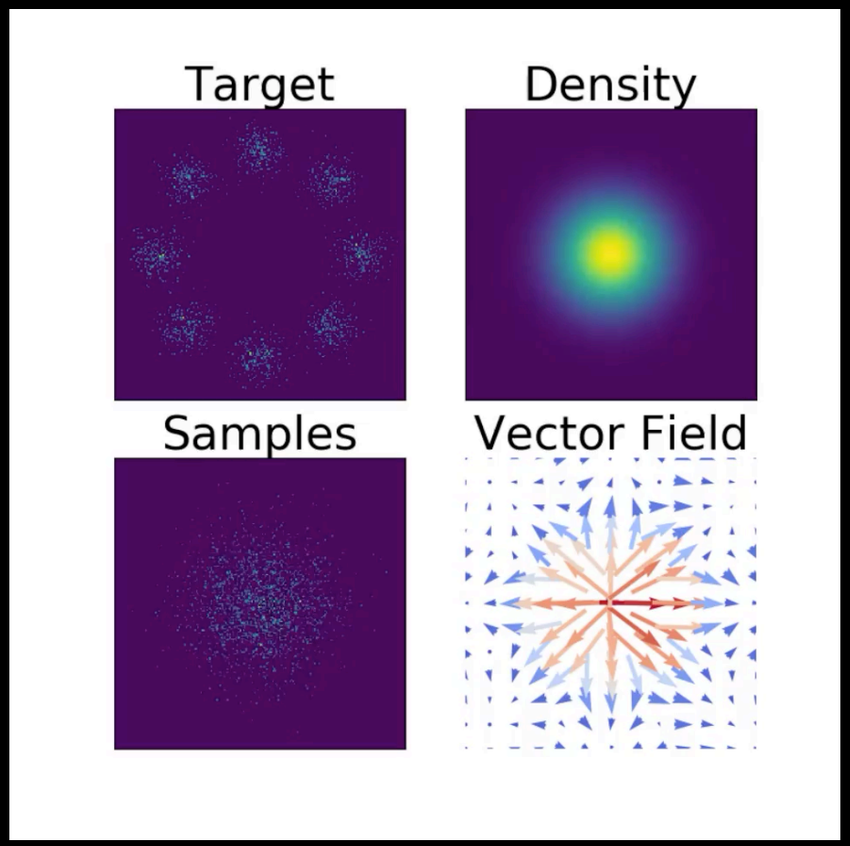
where θ is trained using adjoint backpropagation to maximize stochastic estimates of

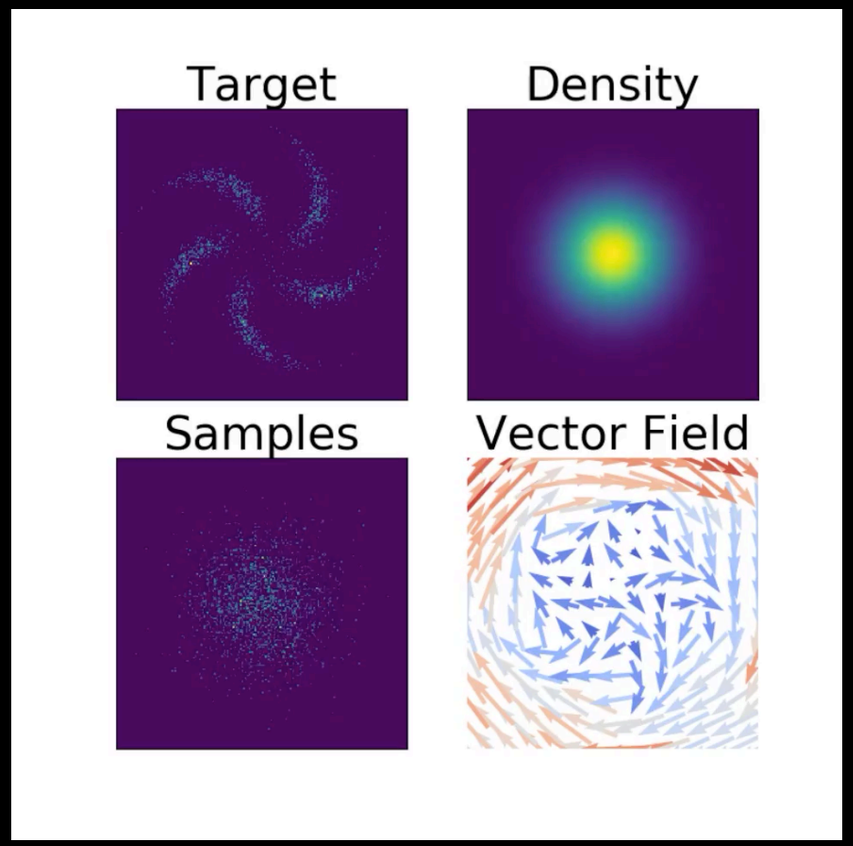
$$\log p(x) = \log p(z_0) + \mathbb{E}_{p(e)} \left[\int_0^1 e^T \frac{\partial f}{\partial z(t)} e dt \right]$$

giving us the first invertible generative model which allows unrestricted architectures to specify the dynamics

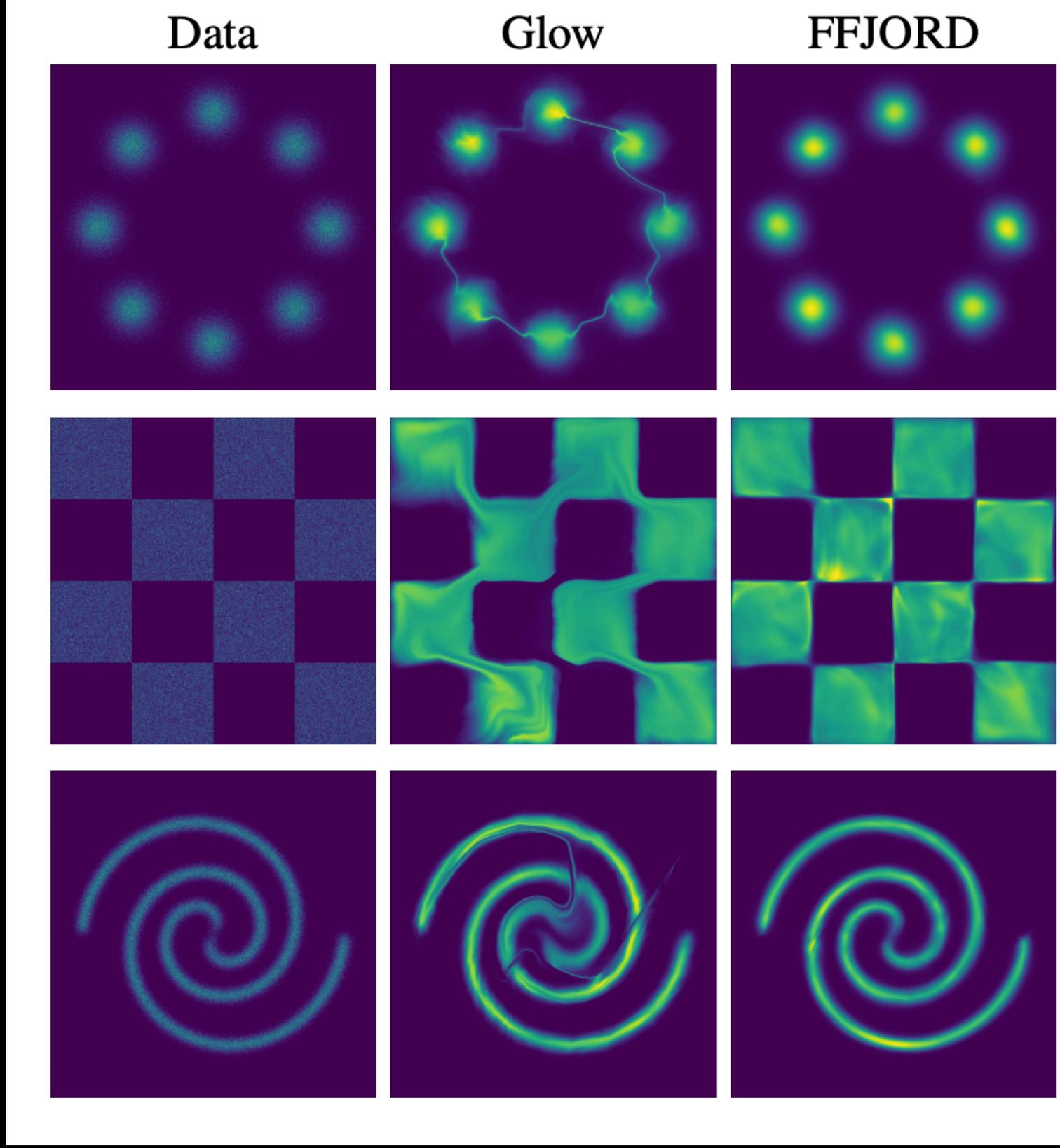
hence the name Free-Form Jacobian of Reversible Dynamics (FFJORD)

FFJORD in action





comparison to prior work



tabular density estimation

FFJORD flows defined by unrestricted neural networks

outperforms all other models with efficient sampling by a wide margin

even outperforms some autoregressive models without efficient sampling

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Real NVP	-0.17	-8.33	18.71	13.55	-153.28
Glow	-0.17	-8.15	18.92	11.35	-155.07
FFJORD	**-0.46**	**-8.59**	**14.92**	**10.43**	**-157.40**
MADE	3.08	-3.56	20.98	15.59	-148.85
MAF	-0.24	-10.08	17.70	11.75	-155.69
TAN	-0.48	-11.19	15.12	11.01	-157.03
MAF-DDSF	-0.62	-11.96	15.09	8.86	-157.73

-log p(x)

cannot be sampled from efficiently

image density estimation

FFJORD outperforms both real-nvp and Glow on MNIST and can match their performance using a single flow step

performs comparably to Glow on CIFAR10 while using 2% as many parameters

	MNIST	CIFAR10
Real NVP	1.06*	3.49*
Glow	1.05*	**3.35***
FFJORD	**0.99*** (1.05†)	3.40*
MADE	2.04	5.67
MAF	1.89	4.31
TAN	-	-
MAF-DDSF	-	-

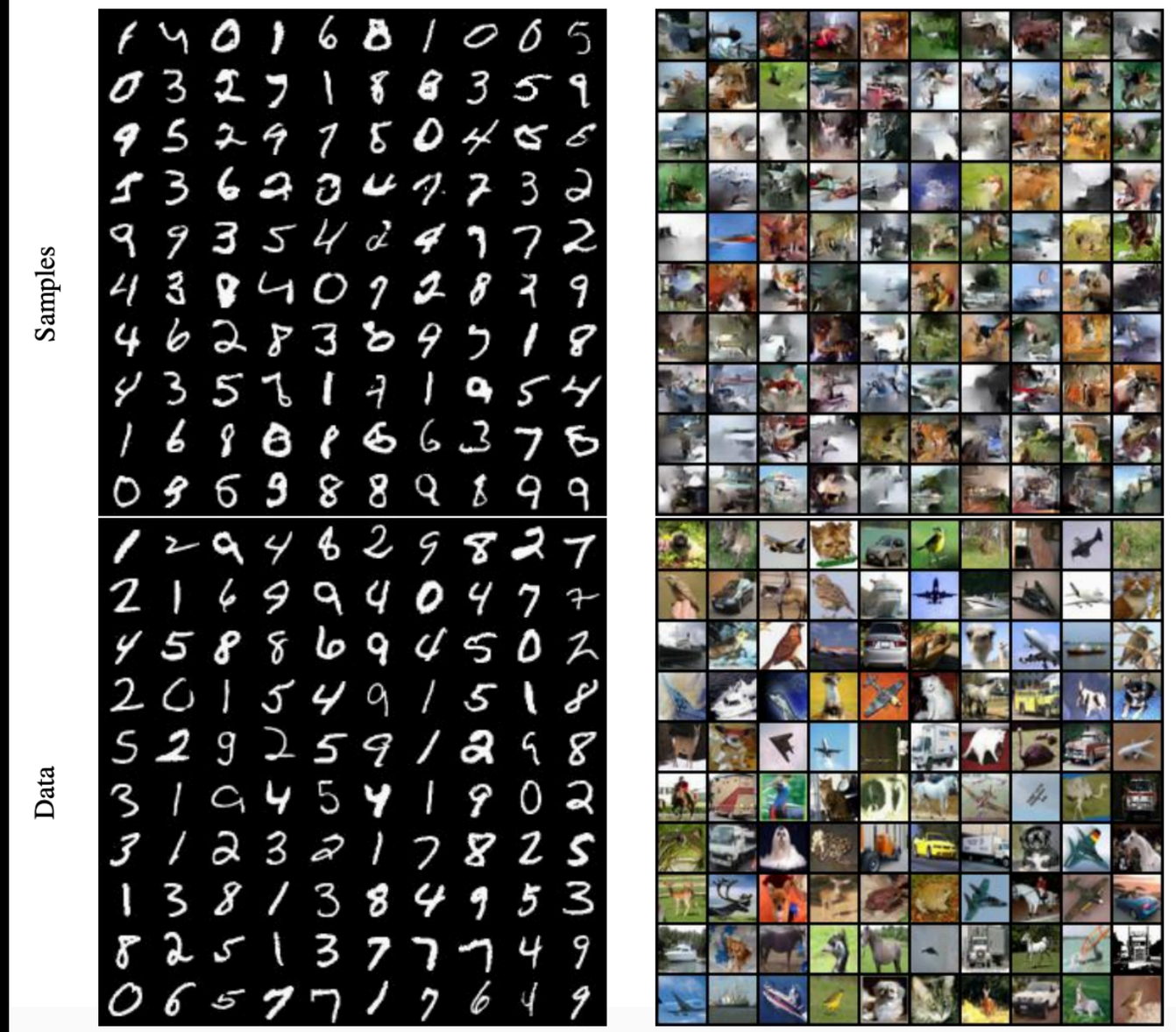


image density estimation

FFJORD outperforms both real-nvp and Glow on MNIST and can match their performance using a single flow step

performs comparably to Glow on CIFAR10 while using 2% as many parameters

	MNIST	CIFAR10
Real NVP	1.06*	3.49*
Glow	1.05*	**3.35***
FFJORD	**0.99*** (1.05†)	3.40*
MADE	2.04	5.67
MAF	1.89	4.31
TAN	-	-
MAF-DDSF	-	-

FFJORD for variational autoencoders

we can use FFJORD to improve posterior approximation in variational inference like normalizing flows

we use the standard inference network to propose a gaussian distribution which is sampled from this sample provides the initial value of an IVP

the dynamics are governed by a neural net

outperforms all competing normalizing flows (on test set ELBO)

	MNIST	Omniglot	Frey Faces	Caltech Silhouettes
No Flow	$86.55 \pm .06$	$104.28 \pm .39$	$4.53 \pm .02$	$110.80 \pm .46$
Planar	$86.06 \pm .31$	$102.65 \pm .42$	$4.40 \pm .06$	$109.66 \pm .42$
IAF	$84.20 \pm .17$	$102.41 \pm .04$	$4.47 \pm .05$	$111.58 \pm .38$
Sylvester	$83.32 \pm .06$	$99.00 \pm .04$	$4.45 \pm .04$	$104.62 \pm .29$
FFJORD	$\mathbf{82.82 \pm .01}$	$\mathbf{98.33 \pm .09}$	$\mathbf{4.39 \pm .01}$	$\mathbf{104.03 \pm .43}$

drawbacks

non-constant computation time

currently 4-5x slower than other methods (Glow, Real-NVP)

divergence estimator means we cannot estimate log p(x) with importance samples for use in VAE

code available

implementation of FFJORD (https://github.com/rtqichen/ffjord)

implementation of the adjoint backprop method (https://github.com/rtqichen/torchdiffeq)

we have implemented a number of GPU-enabled ODE solvers

- explicit and implicit methods
- adaptive and fixed-step methods
- Runge-Kutta and Adams-Bashforth methods

we hope these will be used by the community to encourage the development of related methods

thanks

feel free to reach out to me at wgrathwohl@cs.toronto.edu

or my coauthors

Ricky Chen (rtqichen@cs.toronto.edu)

Jesse Bettencourt (jessebett@cs.toronto.edu)

Ilya Sutskever (ilyasu@openai.com)

David Duvenaud (duvenaud@cs.toronto.edu)

next steps and future work

regularizing dynamics of ODE to improve performance (spectral normalization)

drawing inspiration from continuous dynamics to produce discrete time models with fewer architecture restrictions

extend to stochastic differential equations