FFJORD: reversible generative models with unrestricted architectures

Will Grathwohl

FFJORD: reversible generative models with unrestricted architectures

Will Grathwohl

Joint work with:
Tian Qi Chen (equal contribution)
Jesse Bettencourt
Ilya Sutskever
David Duvenaud

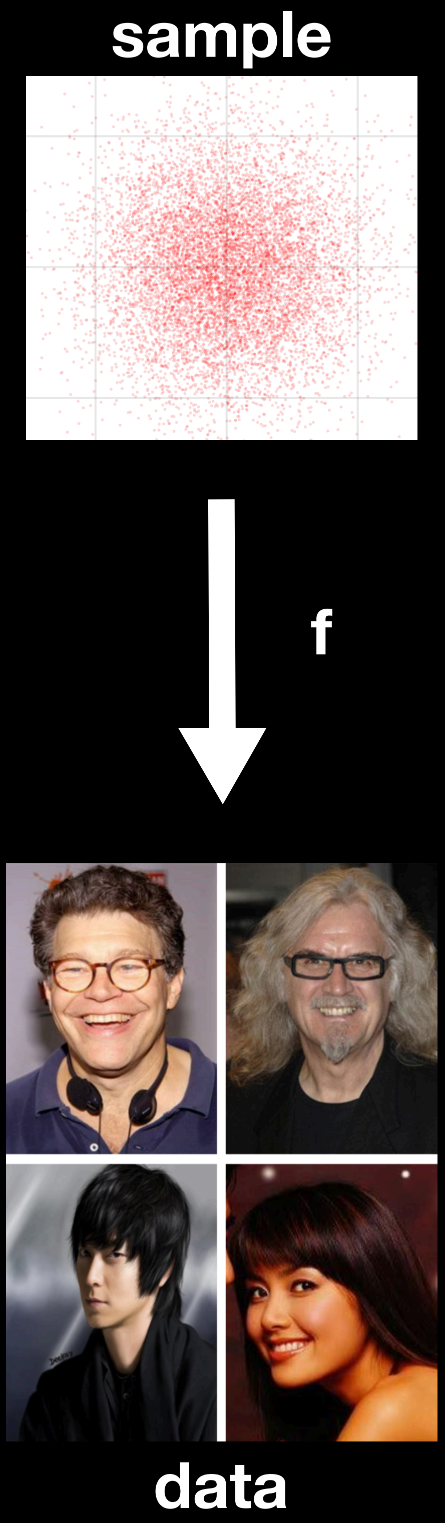
generative modeling with the change of variables formula

we can define a simple generative model as

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

if f is invertible…

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



challenges

care must be taken to design f_θ which is invertible and where $\log|\partial f/\partial x|$ can be efficiently computed

we cannot easily compute or estimate the determinant of arbitrary functions

typically, we use restricted network architectures to deliver these properties

examples include nice (Dinh et al. 2014), real-nvp (Dinh et al. 2016), Glow (Kingma & Dhariwal 2018)

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

$$z = [z_a; z_b]$$

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

log-determinant

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

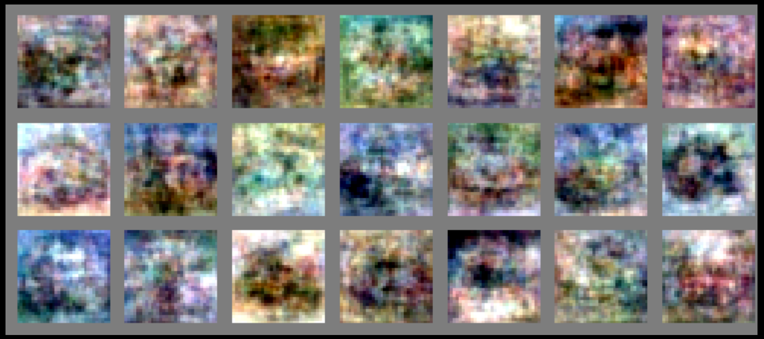
realities

since each layer consists of a fairly simple transformation of the data, many must be composed

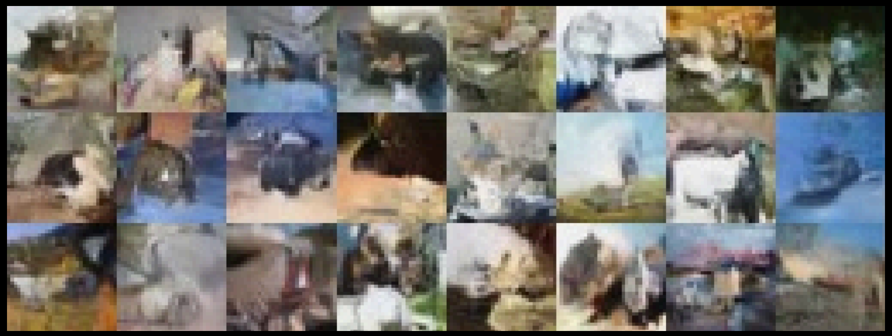
the published Glow model trained on cifar-10 has ~400 layers and over 100M parameters

despite these drawbacks considerable progress has been made in recent years

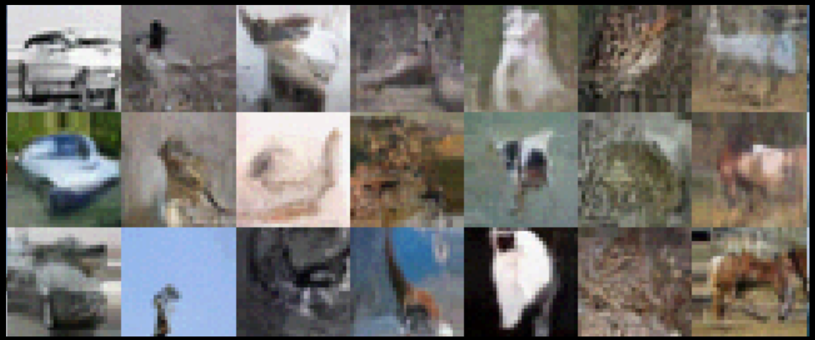
NICE (2014)



real-nvp (2016)



Glow (2018)



what if…

we could use more expressive functions while still obeying the main constraints of flows?

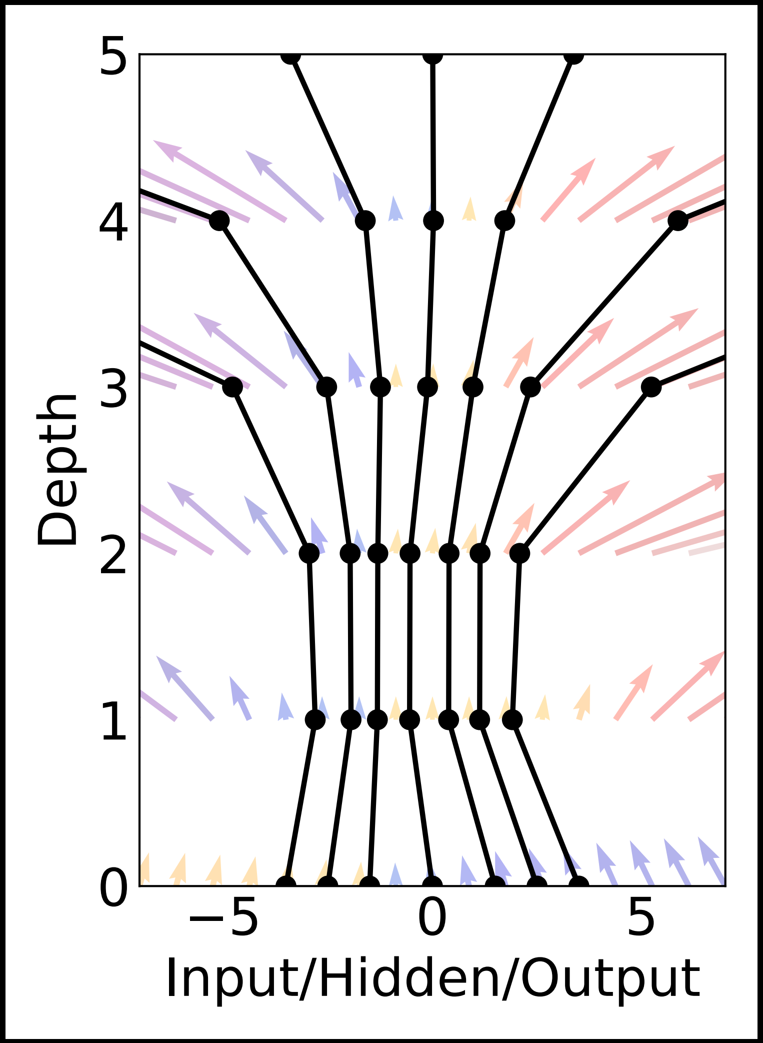
if we look at the problem in a different way, something interesting happens…

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

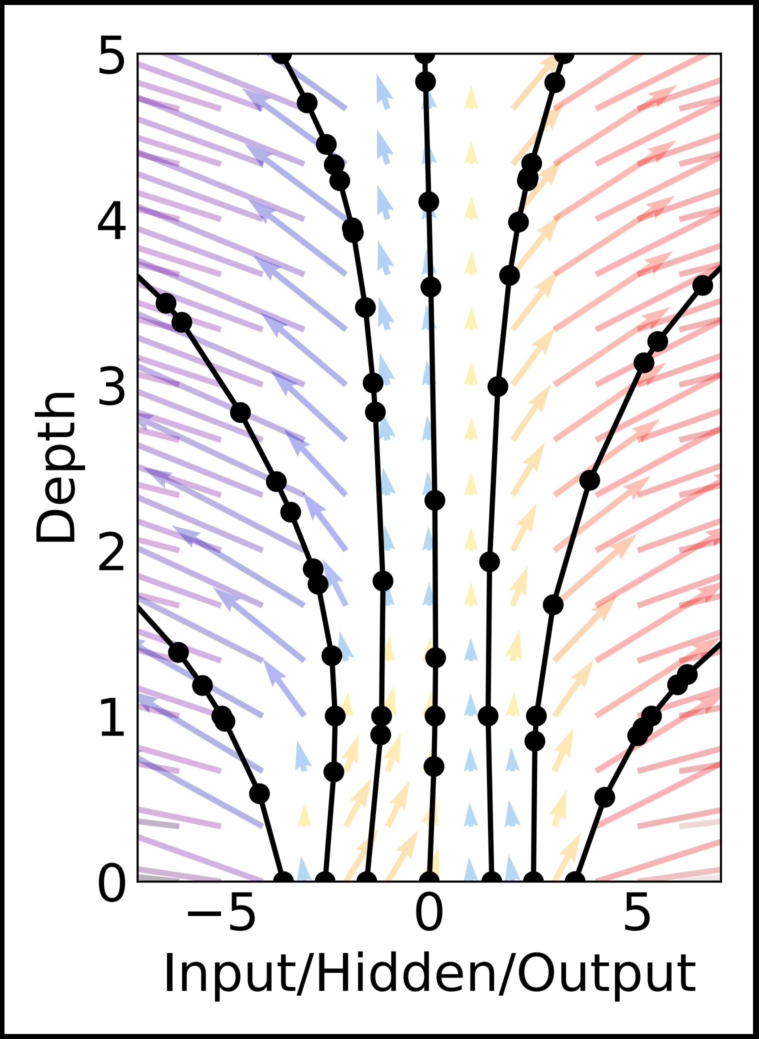


$$\begin{aligned} x &= z_0 \\ z_t &= f_\theta^t(z_{t-1}) \\ f_\theta(x) &= z_T \end{aligned}$$

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

take a few limits…

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



$$x = z_0$$

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

$$f_\theta(x) = z_T = z_0 + \int_0^1 f_\theta(z_t, t)dt$$

most interestingly…

$$\log p(x) = \log p(z_T) + \int_0^1 \nabla f(z_t, t)dt$$

known as the continuous change of variables (Chen et al. 2018)

what this means

log-probability of the data under the discrete model

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

log-probability of the data under the continuous model

$$\log p(x) = \log p(z_T) + \int_0^1 \nabla f(z_t, t) dt$$

we replace the sum of jacobian log-determinants with the integral of a divergence

log-dets vs divergence

for a general function $f : \mathcal{R}^N \to \mathcal{R}^N$, $\partial f / \partial x$ can be computed in $O(N^2)$ time using automatic differentiation

given $\partial f / \partial x$ computing $\log |\partial f / \partial x|$ requires $O(N^3)$ computation and there is no known efficient unbiased estimator

given $\partial f / \partial x$ $\nabla f(z_t, t)$ can be computed in $O(N)$ thus we are constrained by the $O(N^2)$ cost of computing the jacobian

but, using two tricks we can produce an unbiased estimator for this quantity with $O(N)$ computation

stochastic divergence estimation

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

for any matrix A, we have

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

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

given that $\nabla f(z) = \mathrm{Tr}(\partial f/\partial z)$ we have

$$\nabla f(z) = \mathbb{E}_{p(e)}[e^T(\partial f/\partial z)e]$$

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 divergence estimates can be incorporated into the continuous change of variables with

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

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 is defined by f which represents the gradients of a continuous-time dynamics process

computing z_T consists of solving an ordinary differential equation (ODE) initial value problem (IVP)

if f is a neural network we must compute the gradient of a solution to an IVP wrt to the parameters of the function that governs its dynamics

neural ODEs

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

given an objective

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

Chen et al. (2018) demonstrates that $\frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}, \frac{\partial L}{\partial t_1}$ can all be found by solving a different IVP backwards in time

adjoint backpropagation (in theory)

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

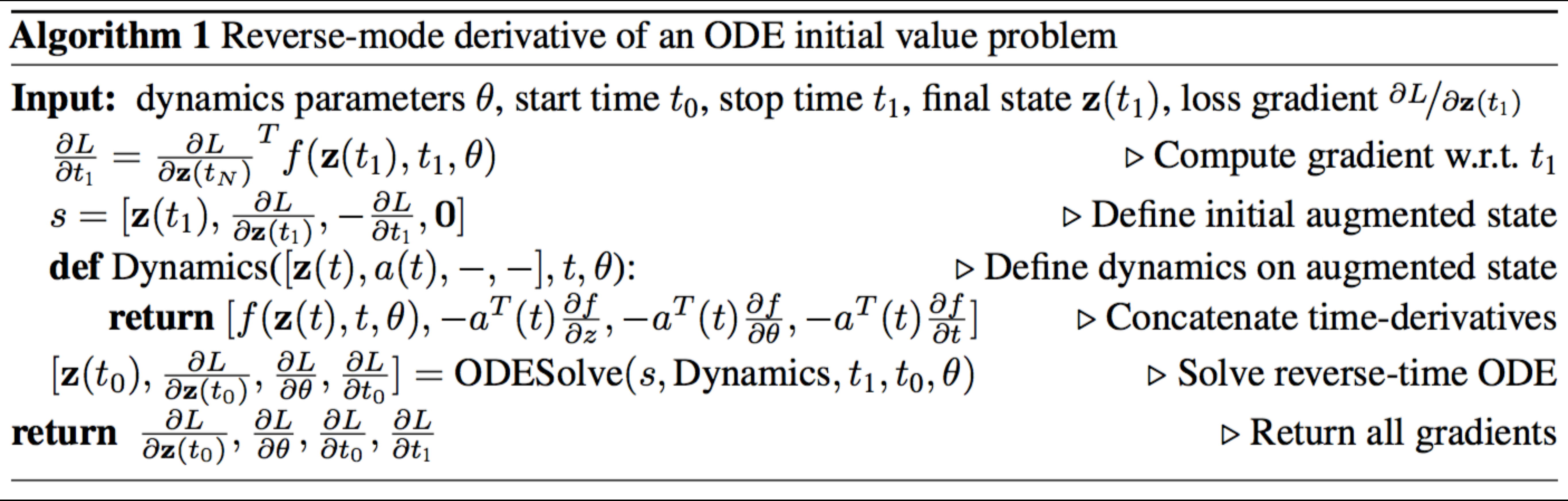
$$\frac{\partial L}{\partial \theta} = \int_{t_1}^{t_0} a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt$$

[Scalable Inference of Ordinary Differential Equation Models of Biochemical Processes", Froehlich, Loos, Hasenauer, 2017]

adjoint backpropagation (in practice)

Solve original IVP using a numerical solver

compute its gradients with a second call to a numerical solver



Algorithm 1 Reverse-mode derivative of an ODE initial value problem

```
Input: dynamics parameters $\theta$, start time $t_0$, stop time $t_1$, final state $\mathbf{z}(t_1)$, loss gradient $\partial L/\partial \mathbf{z}(t_1)$
  $\frac{\partial L}{\partial t_1} = \frac{\partial L}{\partial \mathbf{z}(t_N)}^T f(\mathbf{z}(t_1), t_1, \theta)$                ▷ Compute gradient w.r.t. $t_1$
  $s = [\mathbf{z}(t_1), \frac{\partial L}{\partial \mathbf{z}(t_1)}, -\frac{\partial L}{\partial t_1}, \mathbf{0}]$                ▷ Define initial augmented state
  def Dynamics($[\mathbf{z}(t), a(t), -, -], t, \theta$):                ▷ Define dynamics on augmented state
      return $[f(\mathbf{z}(t), t, \theta), -a^T(t)\frac{\partial f}{\partial z}, -a^T(t)\frac{\partial f}{\partial \theta}, -a^T(t)\frac{\partial f}{\partial t}]$                ▷ Concatenate time-derivatives
  $[\mathbf{z}(t_0), \frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}] = \text{ODESolve}(s, \text{Dynamics}, t_1, t_0, \theta)$                ▷ Solve reverse-time ODE
return $\frac{\partial L}{\partial \mathbf{z}(t_0)}, \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}, \frac{\partial L}{\partial t_1}$                ▷ Return all gradients
```

from Chen et al. (2018)

putting it all together

in this work we define a generative model for data

$$z_0 \sim p(z_0)$$

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

$$x = z_1$$

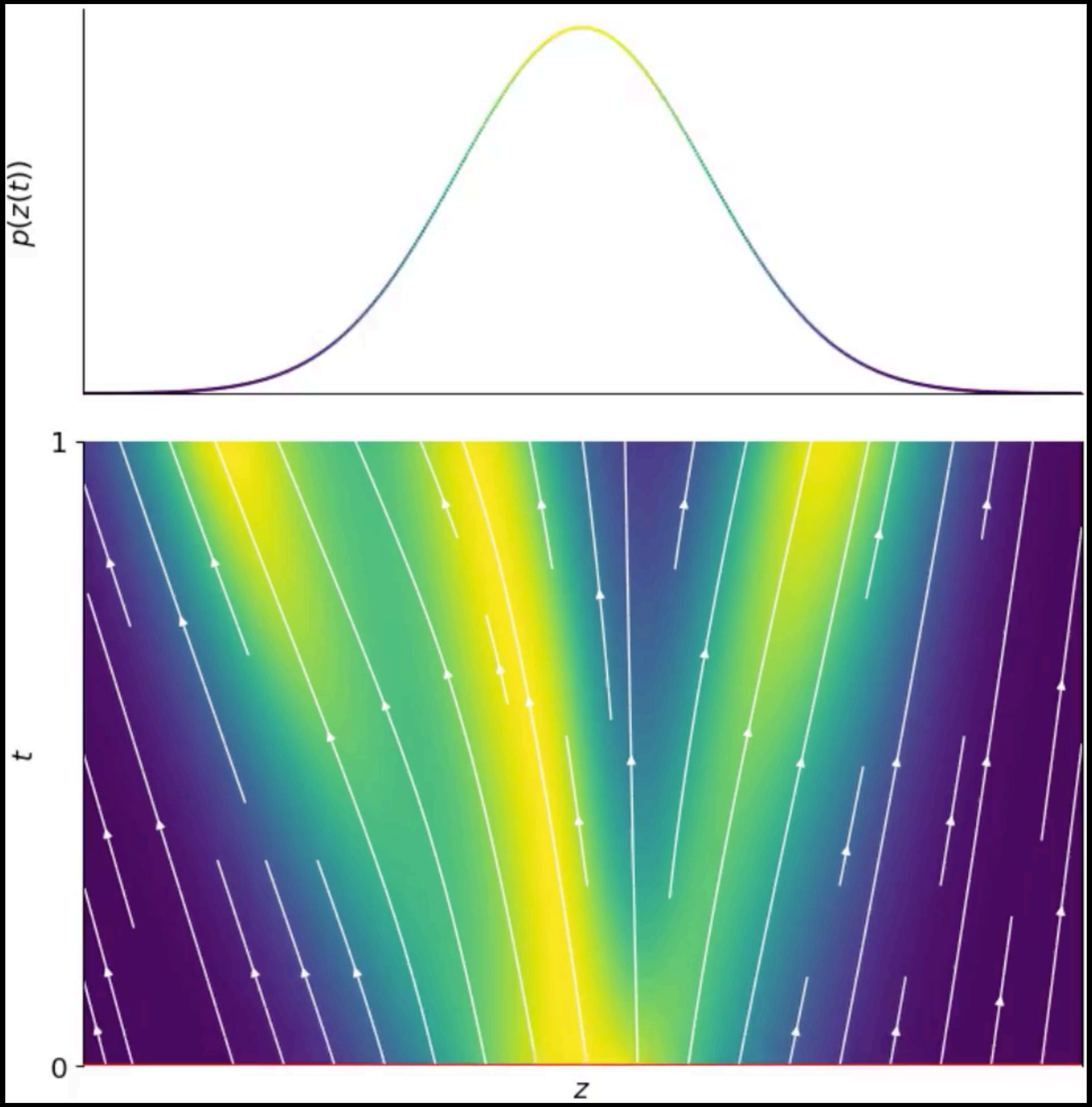
where θ is trained using adjoint backpropagation to maximize stochastic estimates of

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

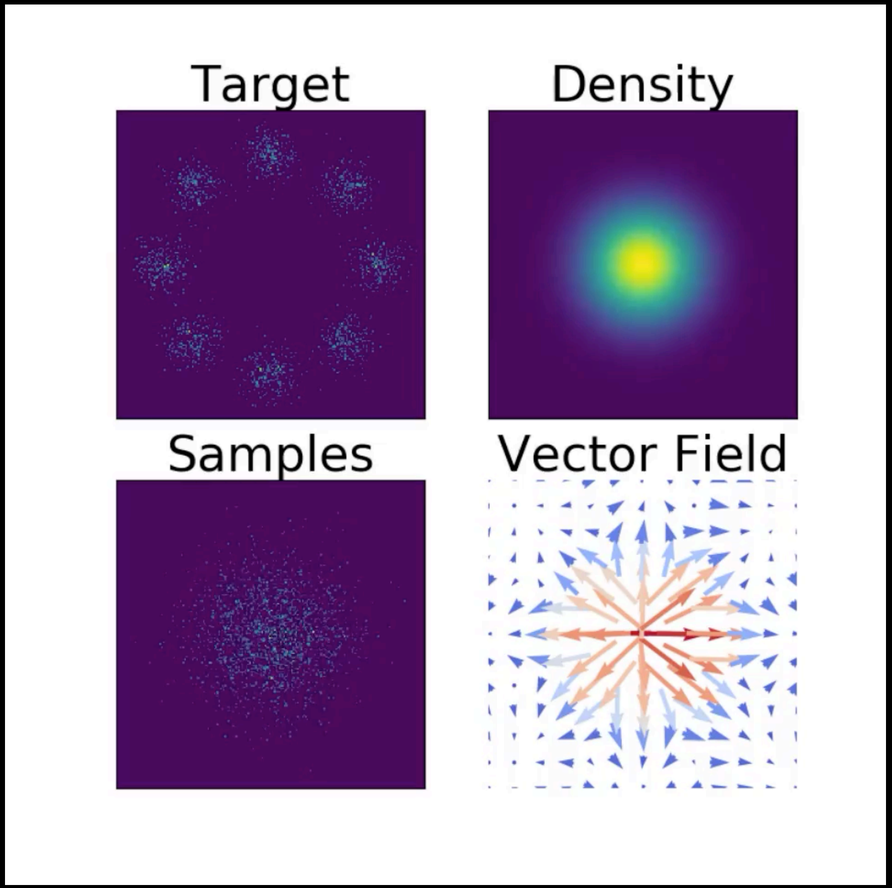
giving us the first invertible generative model which allows unrestricted architectures to specify the dynamics

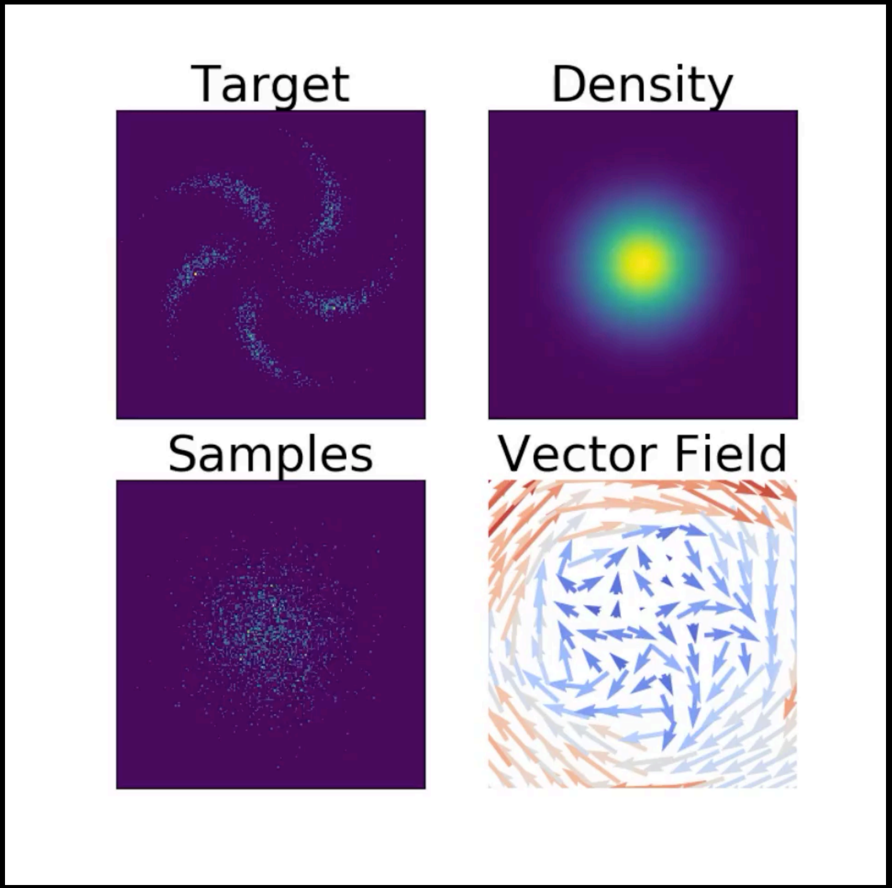
hence the name Free-Form Jacobian of Reversible Dynamics (FFJORD)

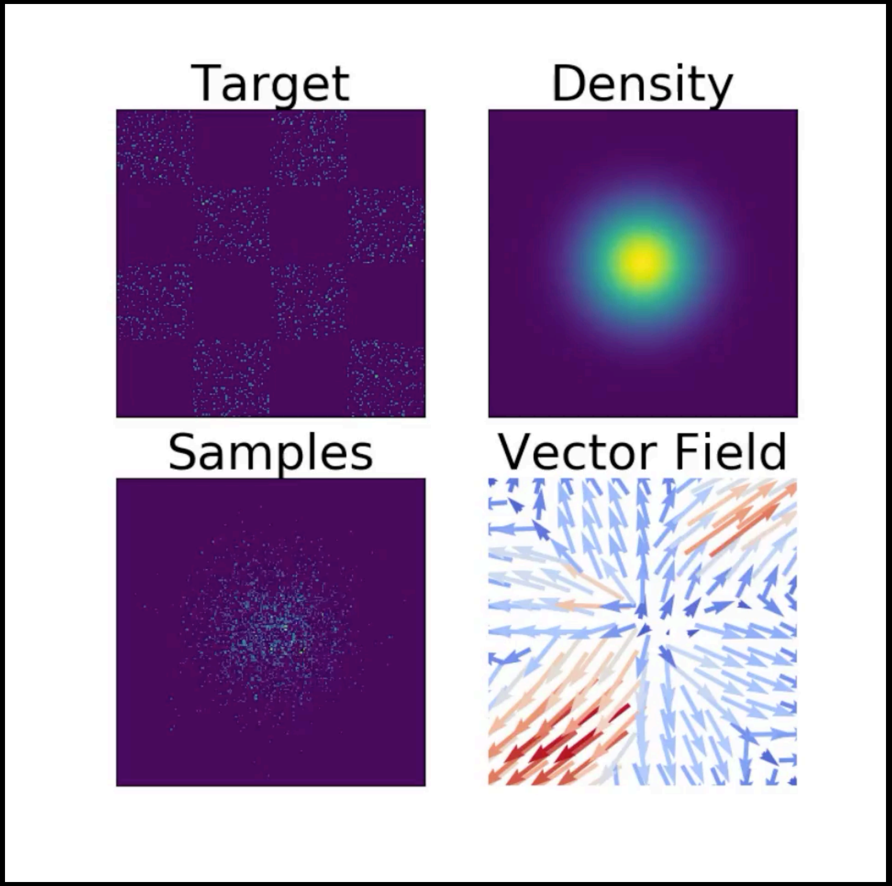
FFJORD in action

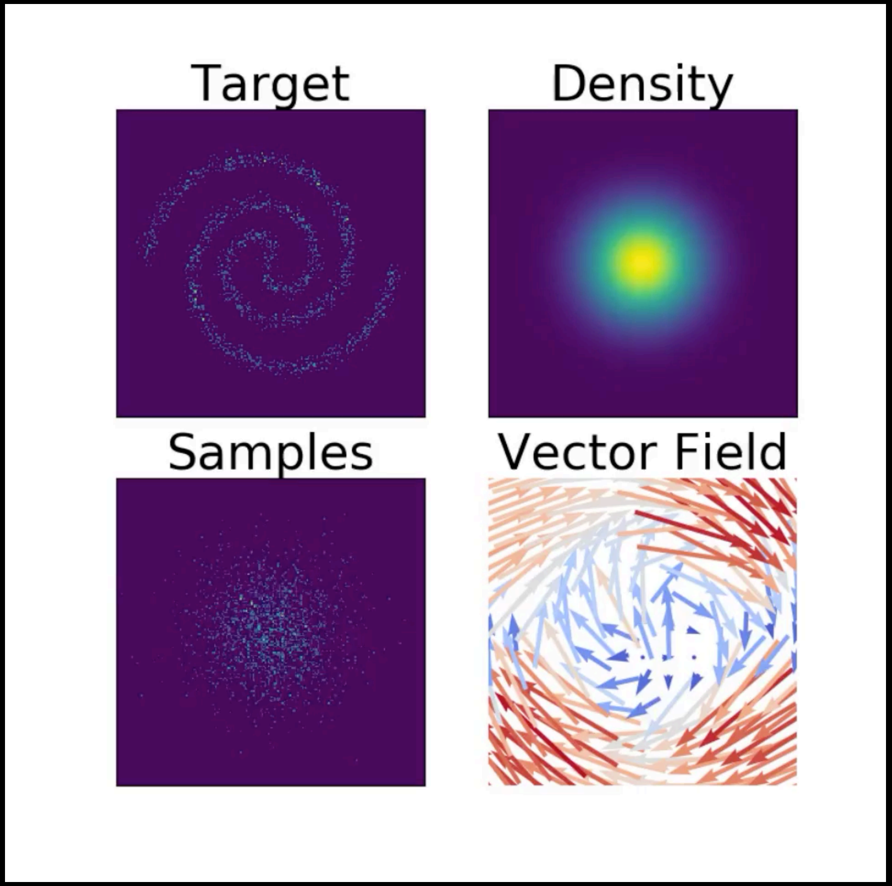


FFJORD in action

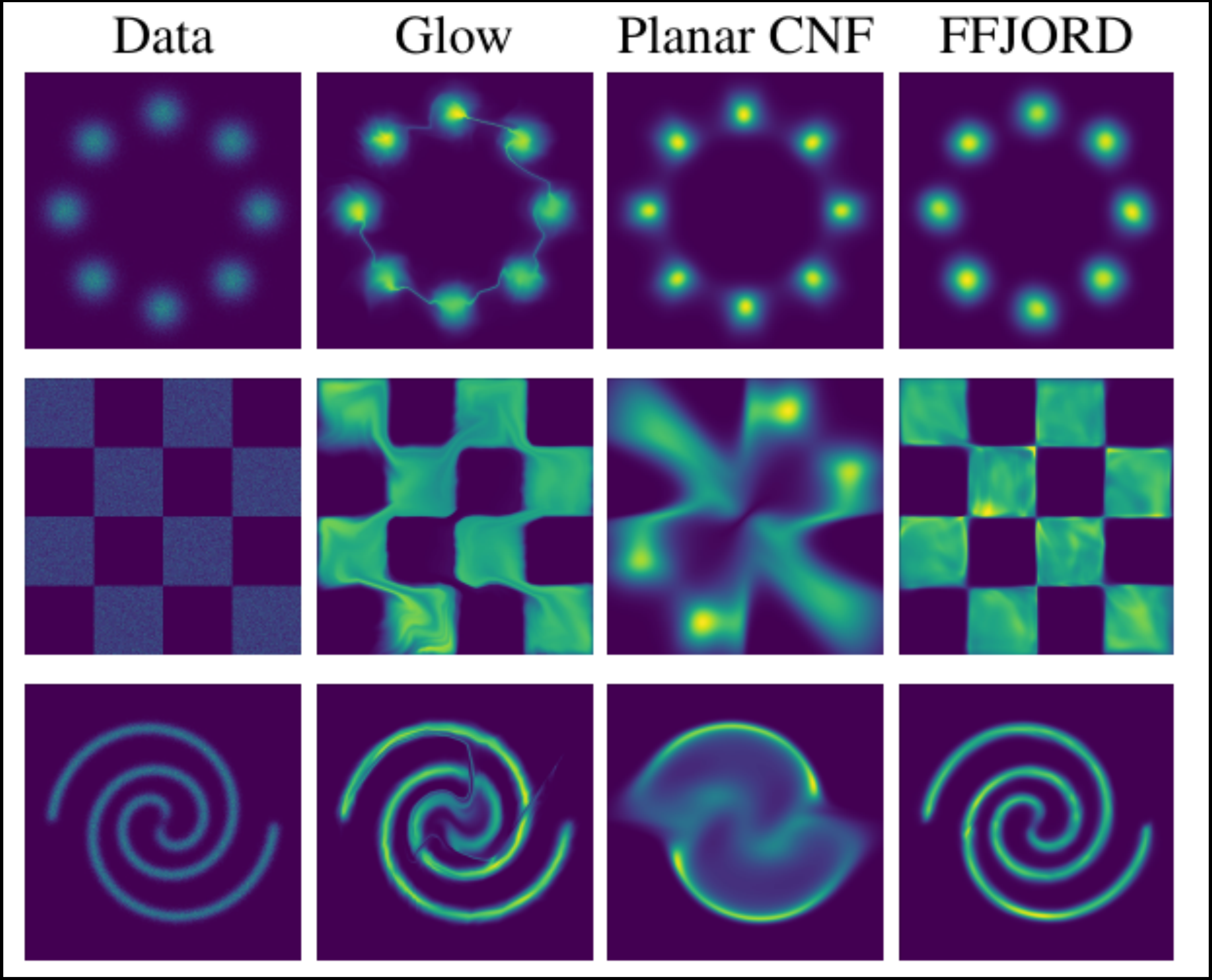








comparison to prior work



summary of quantitative experiments

we evaluate FFJORD on density estimation for tabular and image data

as a normalizing flow for improved posterior estimation in variational autoencoders

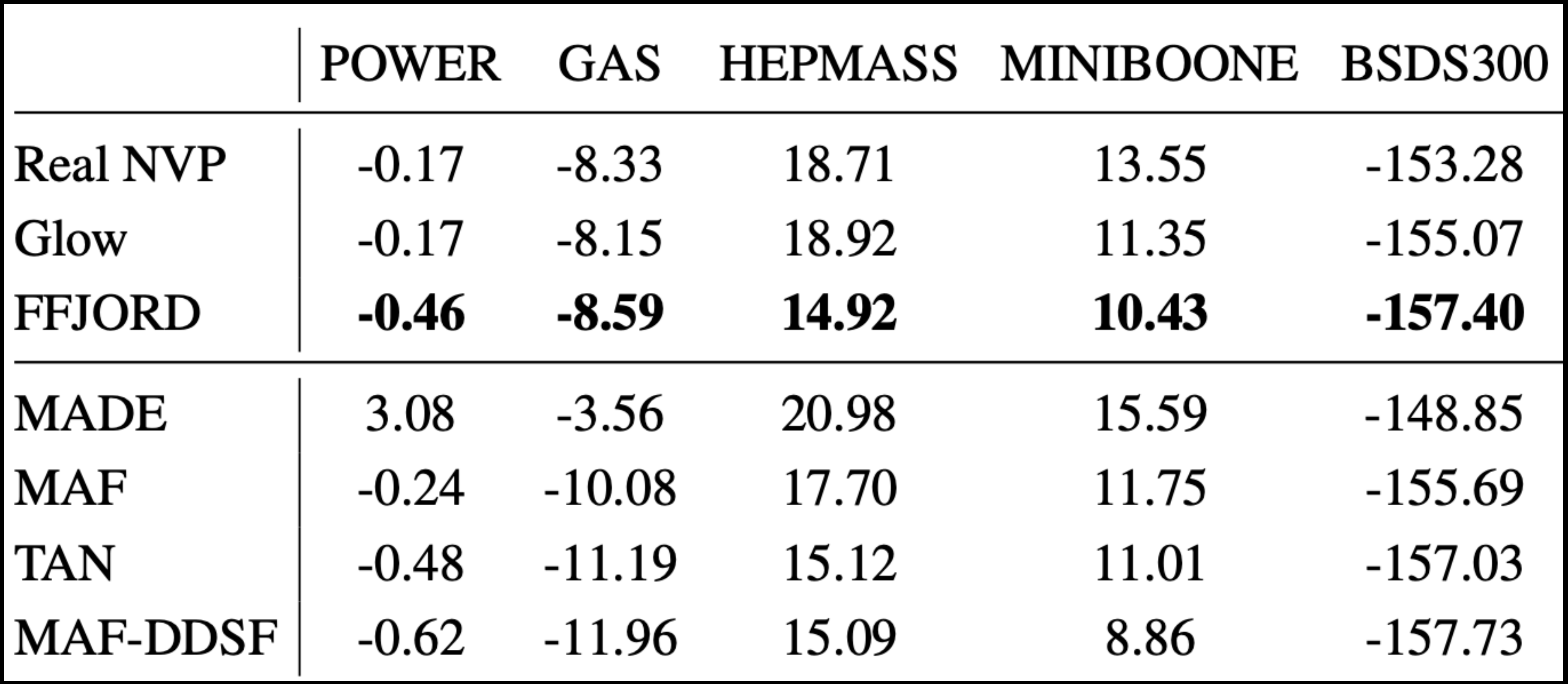
compare to state of the art generative models, normalizing flows, and autoregressive density estimation models

tabular density estimation

FFJORD flows defined by unrestricted neural networks

outperforms other models with efficient sampling by a wide margin

outperforms some autoregressive models without efficient sampling



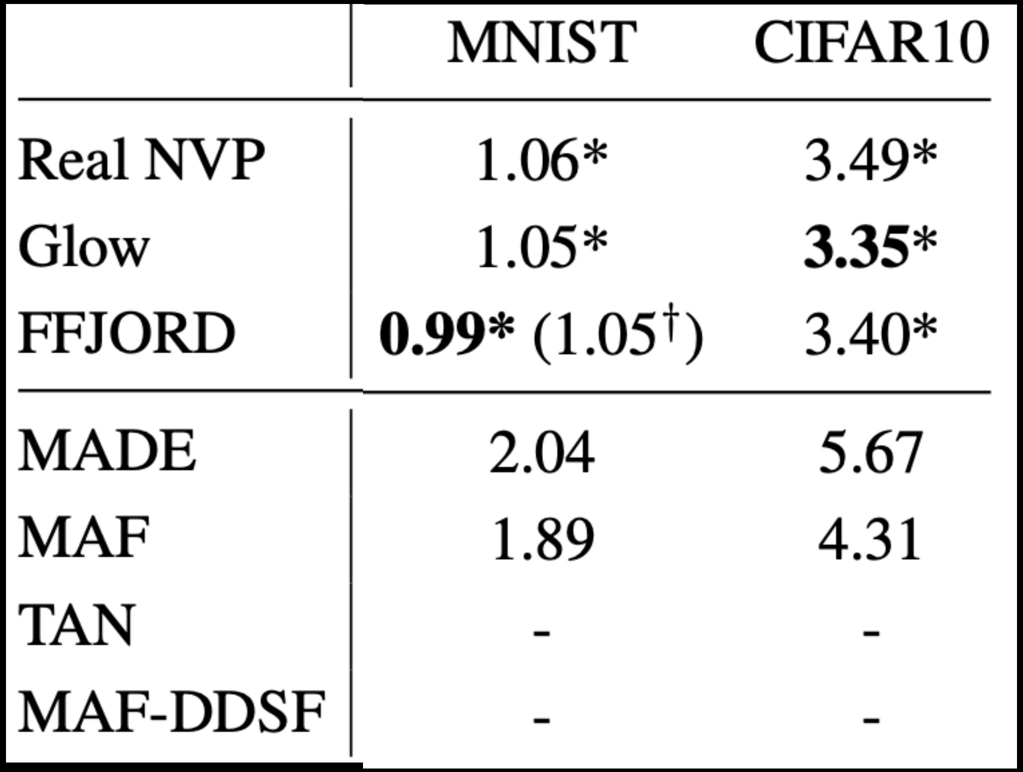
	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Real NVP	-0.17	-8.33	18.71	13.55	-153.28
Glow	-0.17	-8.15	18.92	11.35	-155.07
FFJORD	**-0.46**	**-8.59**	**14.92**	**10.43**	**-157.40**
MADE	3.08	-3.56	20.98	15.59	-148.85
MAF	-0.24	-10.08	17.70	11.75	-155.69
TAN	-0.48	-11.19	15.12	11.01	-157.03
MAF-DDSF	-0.62	-11.96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

cannot be sampled from efficiently

image density estimation

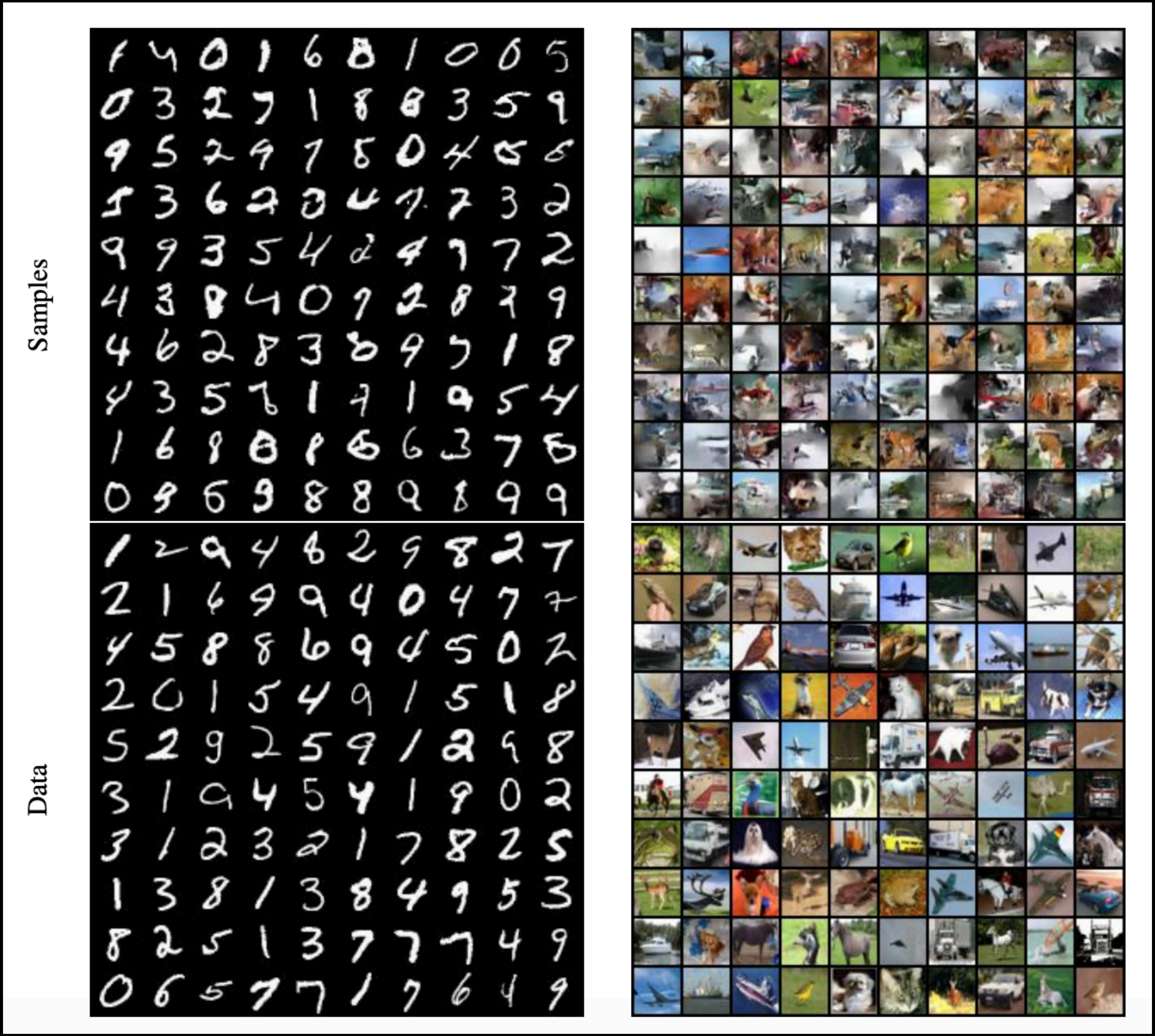
FFJORD outperforms both real-nvp and Glow on MNIST and can match their performance using a single flow step

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 samples



FFJORD for variational autoencoders

$$\text{layer}(h; \mathbf{x}, W, b) = \sigma\left(\left(\underbrace{W}_{D_{out}\times D_{in}} + \underbrace{\hat{U}(\mathbf{x})}_{D_{out}\times k}\underbrace{\hat{V}(\mathbf{x})}_{D_{in}\times k}^{T}\right)h + \underbrace{b}_{D_{out}\times 1} + \underbrace{\hat{b}(\mathbf{x})}_{D_{out}\times 1}\right)$$

	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	$\mathbf{82.82} \pm \mathbf{.01}$	$\mathbf{98.33} \pm \mathbf{.09}$	$\mathbf{4.39} \pm \mathbf{.01}$	$\mathbf{104.03} \pm \mathbf{.43}$