Normalizing Flows

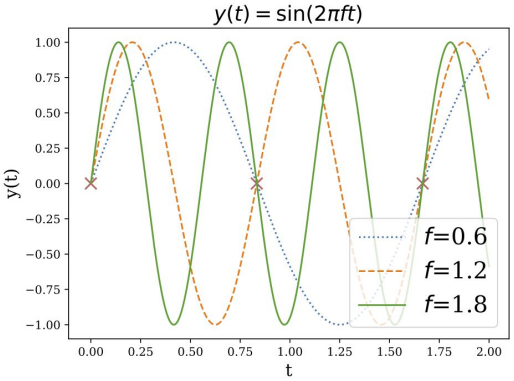
Chin-Wei Huang

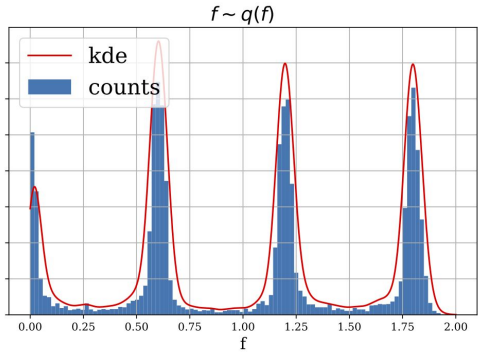
(2020/03/16)

Invertible Neural network and why is it appealing?

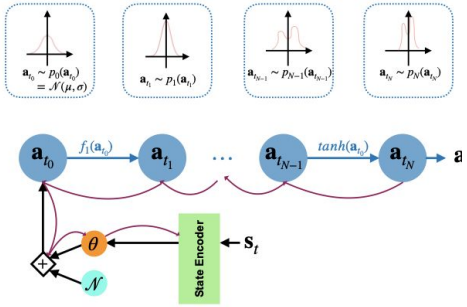
Generative modeling

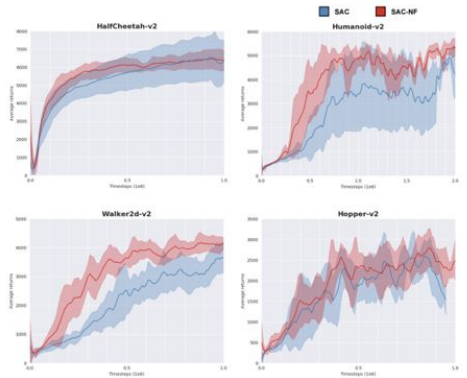
Bayesian inference



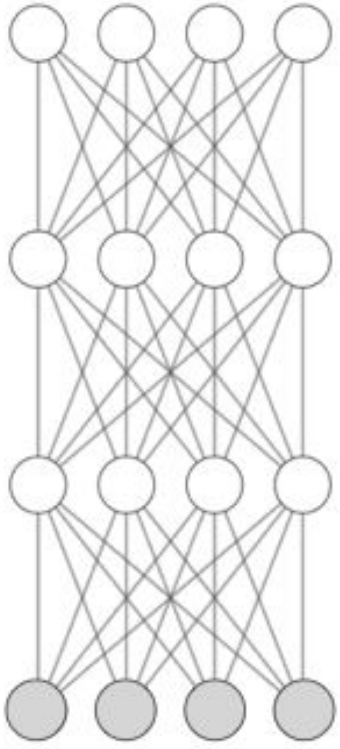


Exploration in RL

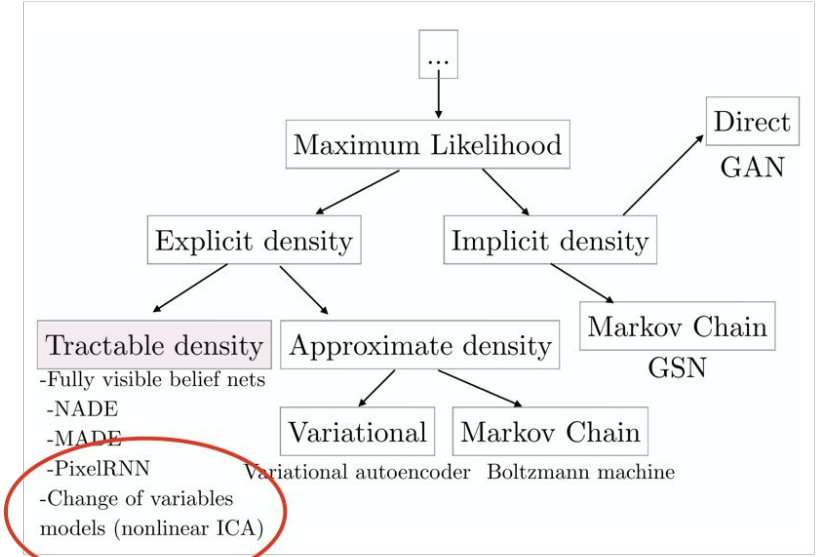


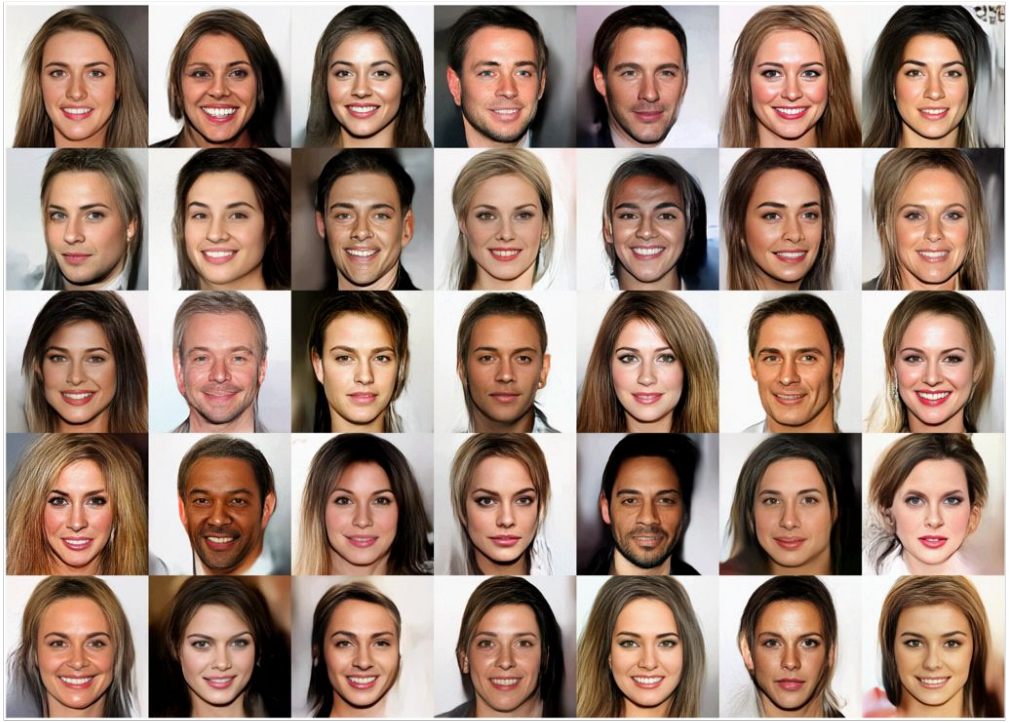


O(1) mem backprop



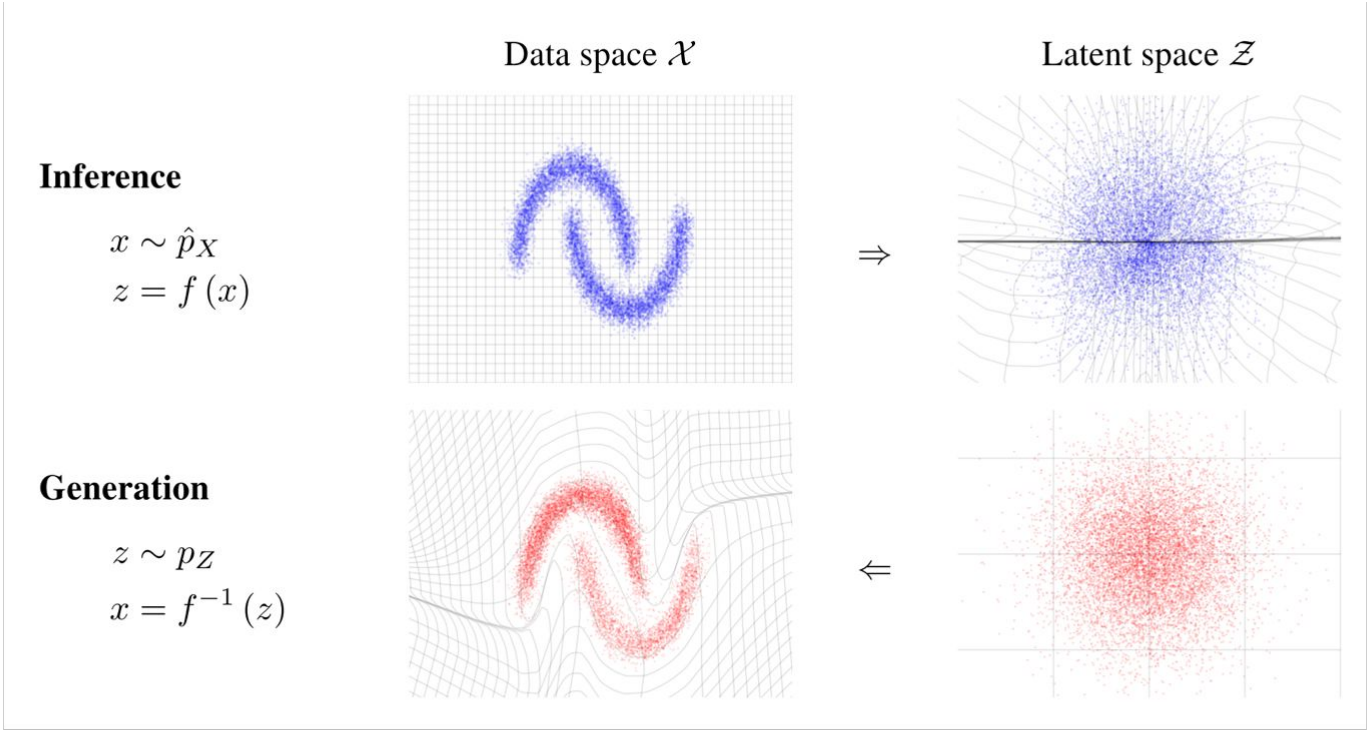
Explicit Density Model Without Approximation





Goodfellow et al. 2016

Kingma and Dhariwal 2018



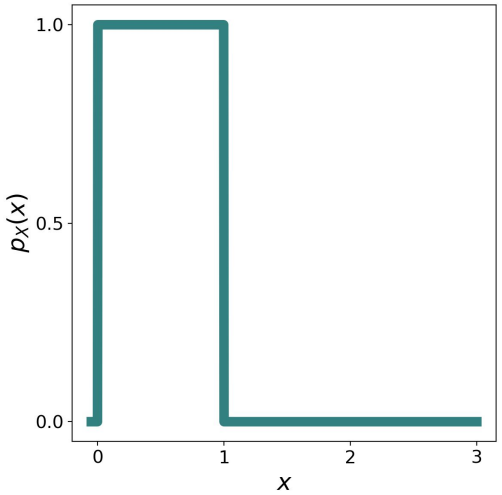
Normalizing Flows

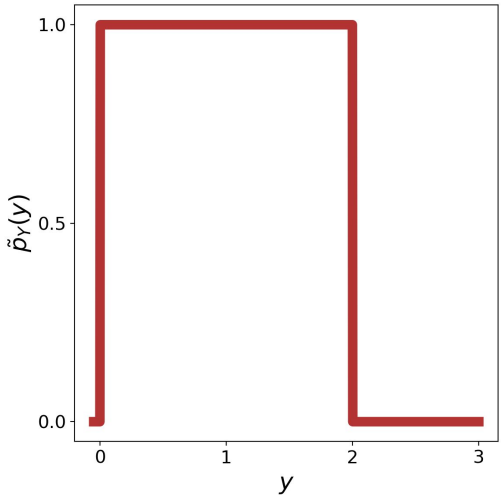
Translating probability distributions

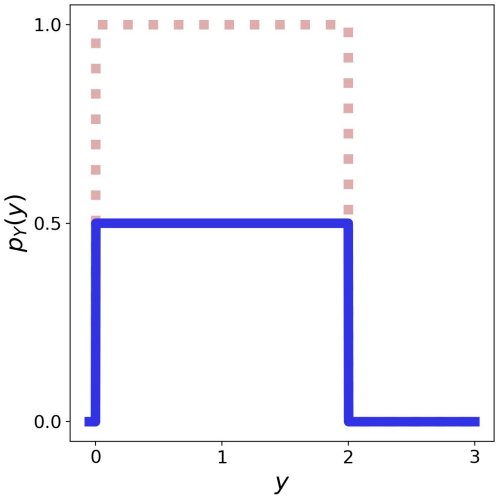
Change of Variable Density Needs to Be Normalized

$$X \sim p_X \qquad Y := 2X$$

$$p_X(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{else} \end{cases} \qquad \tilde{p}_Y(y) = p_X(y/2) \qquad p_Y(y) = p_X(y/2)/2$$



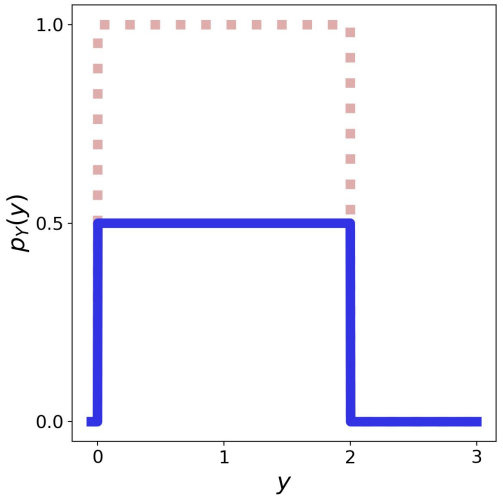




Change of Variable Density (m-Dimensional)

For a multivariable invertible mapping $f : \mathbb{R}^m \to \mathbb{R}^m \quad X \sim p_X \quad Y := f(X)$

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$



$$Y := 2X$$

$$p_Y(y) = p_X(y/2)/2$$

Local change of volume

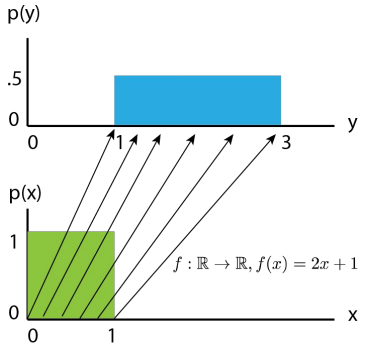
mass = density * volume

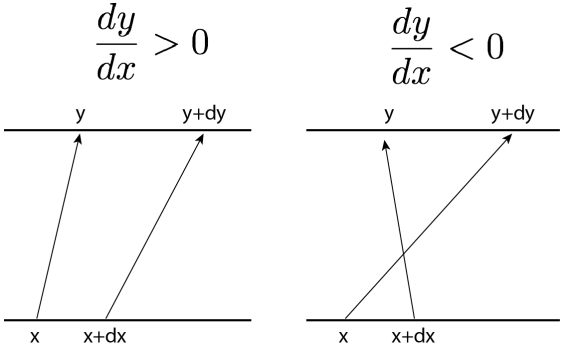
Change of Variable Density (m-Dimensional)

For a multivariable invertible mapping $f : \mathbb{R}^m \to \mathbb{R}^m \quad X \sim p_X \quad Y := f(X)$

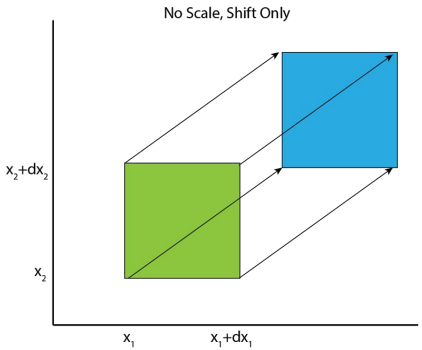
$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{\partial f^{-1}(y)}{\partial y} \right|$$

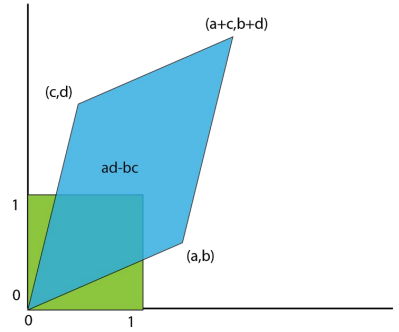
1-D





2-D

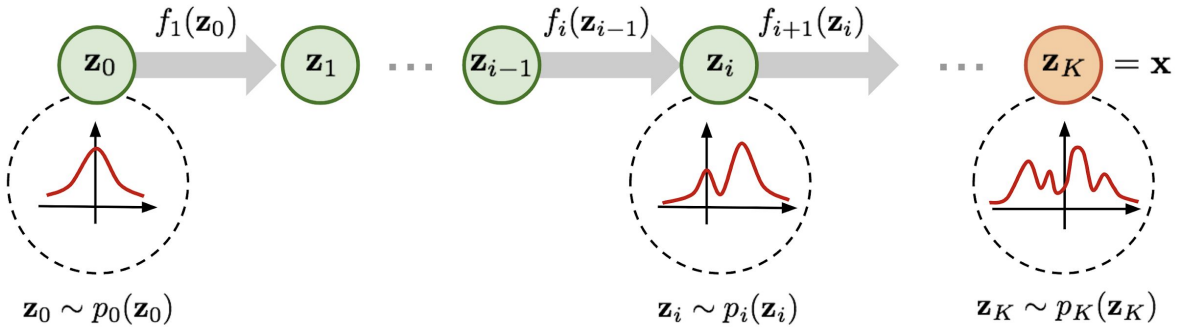




Figures from blog post: *Normalizing Flows Tutorial, Part 1: Distributions and Determinants* by Eric Jang

Chaining Invertible Mappings (Composition)

$$f = f_S \circ \cdots \circ f_2 \circ f_1 \qquad f(x) = f_S(\cdots f_2(f_1(x)))$$



$$\frac{\partial f(x)}{\partial x} = \frac{f_S(x_{S-1})}{\partial x_{S-1}} \cdots \frac{f_2(x_1)}{\partial x_1} \frac{f_1(x_0)}{\partial x_0} \qquad x_s = f_s(x_{s-1}) \quad x_0 = x$$

Chain rule

$$\det\left(\frac{\partial f(x)}{\partial x}\right) = \det\left(\frac{f_S(x_{S-1})}{\partial x_{S-1}}\right) \cdots \det\left(\frac{f_2(x_1)}{\partial x_1}\right) \det\left(\frac{f_1(x_0)}{\partial x_0}\right)$$

Determinant of matrix product

Figure from blog post: *Flow-based Deep Generative Models* by Lilian Weng, 2018

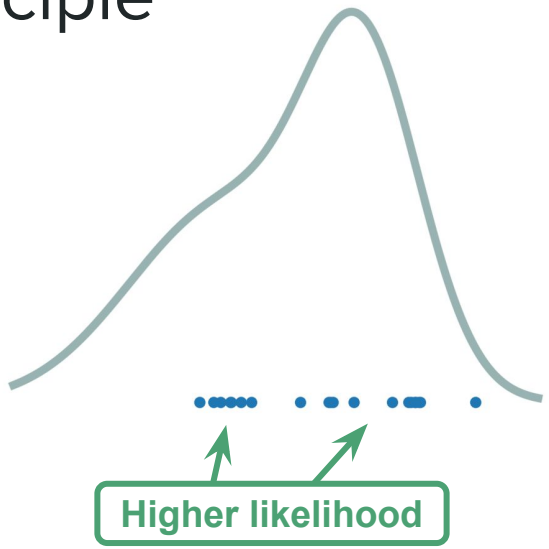
Training with Maximum Likelihood Principle

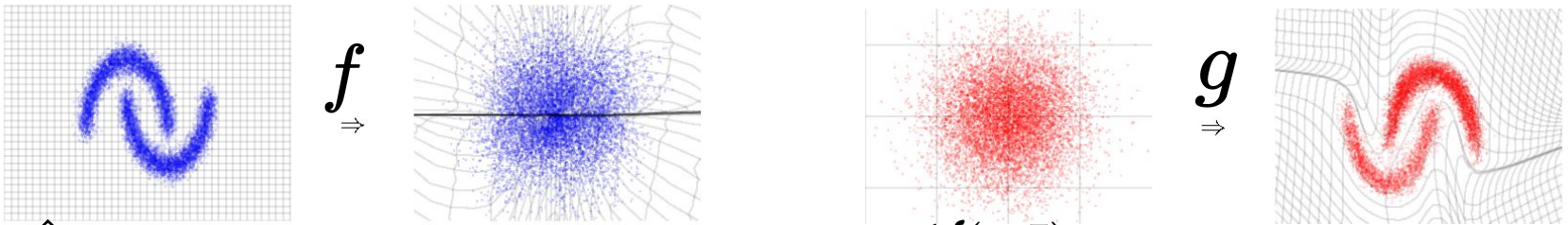
$$Z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad X = g(Z)$$

$$g = f^{-1} \text{ bijective}$$

$$\mathbb{E}_x [\log p(x)] = \mathbb{E}_x \left[\log \mathcal{N}(f(x); 0, I) \left| \det \frac{\partial f(x)}{\partial x} \right| \right]$$

Regularizes the entropy





$x \sim \hat{p}_X$ Inference $z \sim p_Z = \mathcal{N}(\mathbf{0}, \mathbf{I})$ Generation

Figures from *Density Estimation Using Real NVP* by Dinh et al., 2017

Pathways to Designing a Normalizing Flow

1. Require an invertible architecture.

   - Coupling layers, autoregressive, etc.

2. Require efficient computation of a change of variables equation.

$$\log p(x) = \log p(f(x)) + \log \left| \det \frac{df(x)}{dx} \right|$$

<Model distribution> <Base distribution>

$\mathcal{O}(m^3)$

(or a continuous version) $\log p(x(t_N)) = \log p(x(t_0)) + \int_{t_0}^{t_N} \operatorname{tr}\left(\frac{\partial f(x(t), t)}{\partial x(t)}\right) dt$

Slide by Ricky Chen

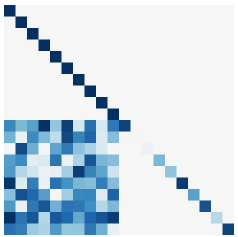
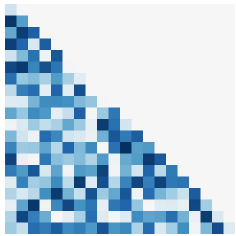
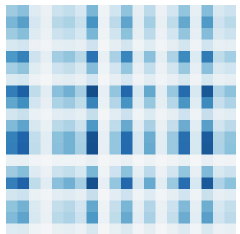
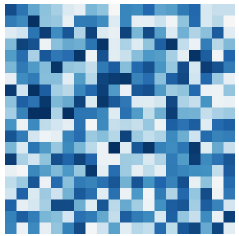
Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling	2. Autoregressive	3. Det identity	4. Stochastic estimation
NICE/RealNVP/Glow Cubic Spline Flow Neural Spline Flow	IAF/MAF/NAF SOS polynomial UMNN	Planar/Sylvester flows Radial flow	Residual Flow FFJORD
			
(Lower triangular + structured)	(Lower triangular)	(Low rank)	(Arbitrary)

Jacobian

Figures from Ricky Chen

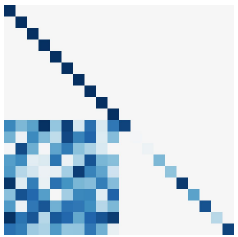
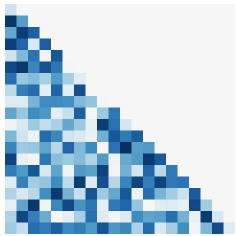
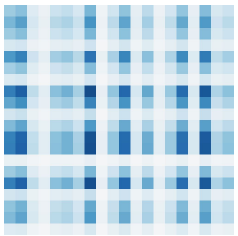
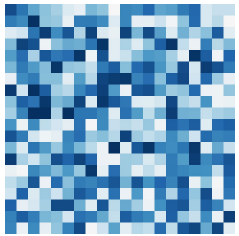
Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

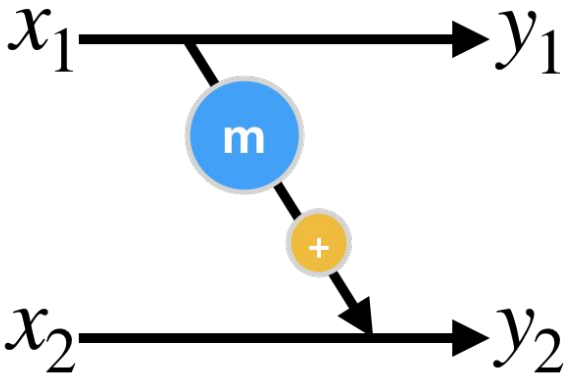
1. Block coupling	2. Autoregressive	3. Det identity	4. Stochastic estimation
NICE/RealNVP/Glow Cubic Spline Flow Neural Spline Flow	IAF/MAF/NAF SOS polynomial UMNN	Planar/Sylvester flows Radial flow	Residual Flow FFJORD
			
(Lower triangular + structured)	(Lower triangular)	(Low rank)	(Arbitrary)

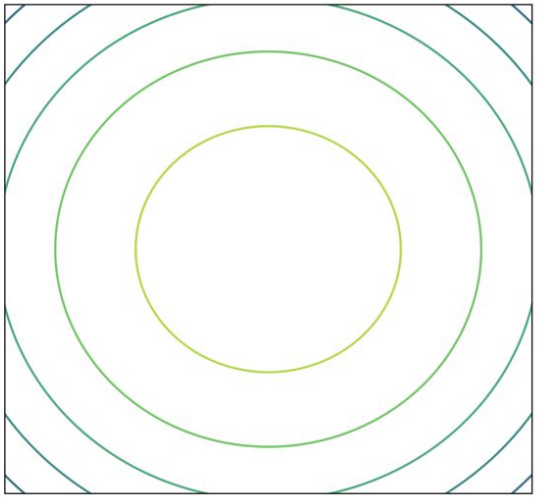
Jacobian

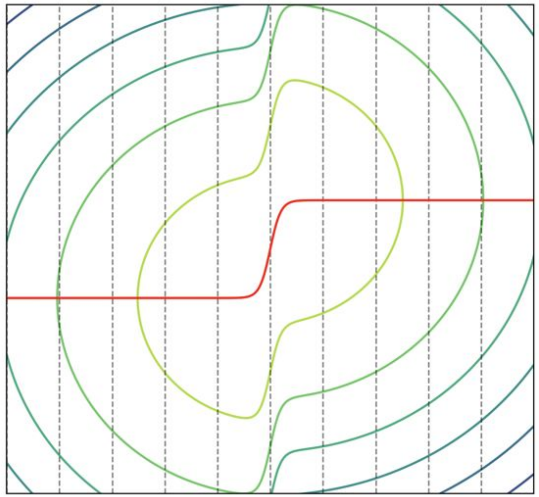
Figures from Ricky Chen

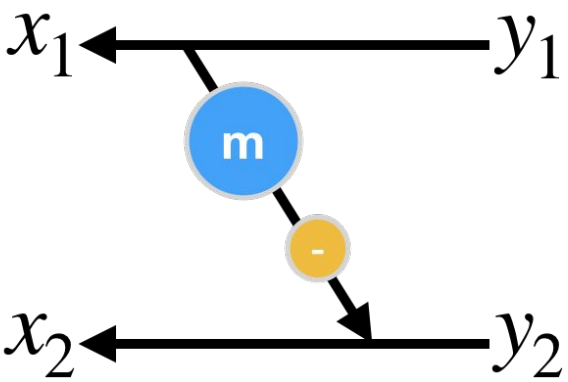
Coupling Law - NICE

- General form $f(\boldsymbol{x})_1 = \boldsymbol{x}_1, \quad f(\boldsymbol{x})_2 = \boldsymbol{x}_2 + \mathcal{F}(\boldsymbol{x}_1)$
- Invertibility no constraint
- Jacobian determinant =1 (volume preserving)









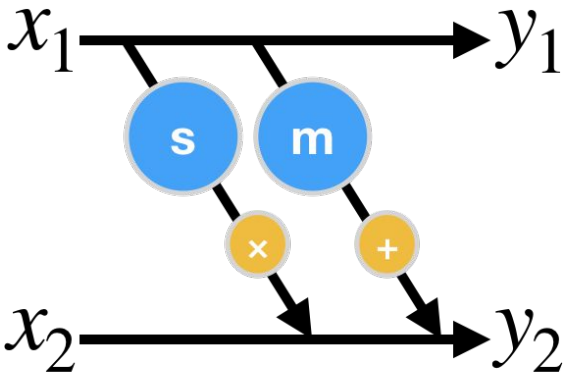
Dinh et al. 2014

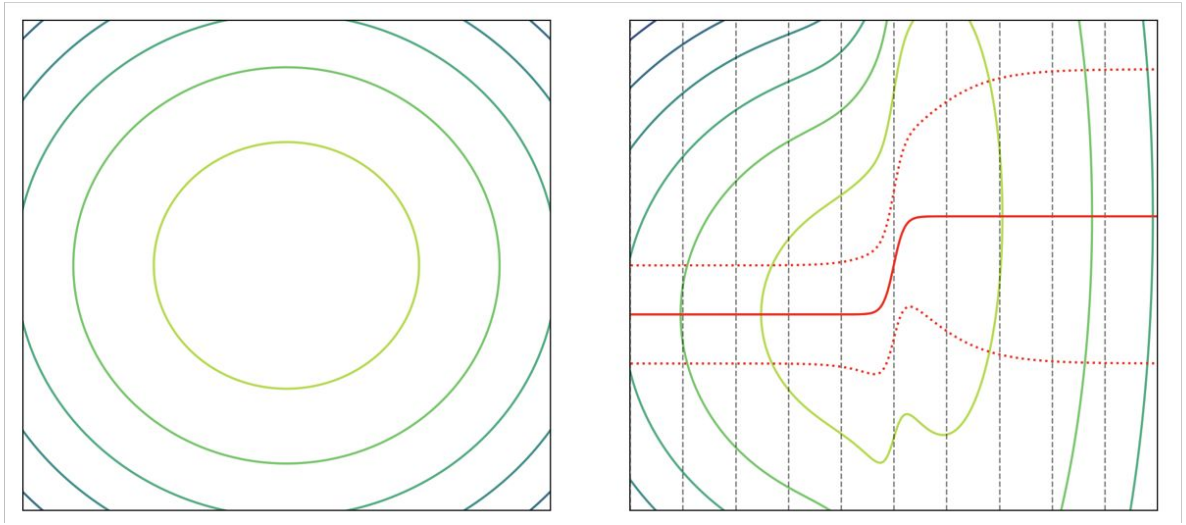
Coupling Law - RealNVP

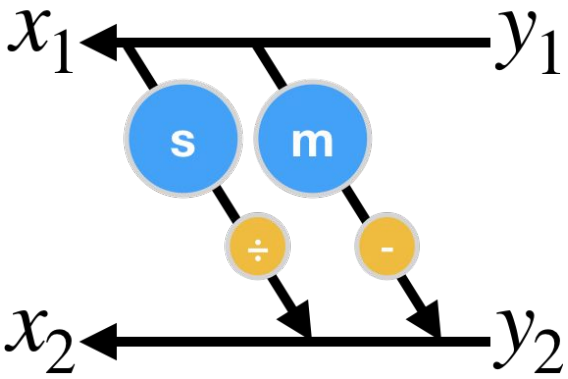
- General form

$$f(\boldsymbol{x})_1 = \boldsymbol{x}_1,$$
$$f(\boldsymbol{x})_2 = s(\boldsymbol{x}_1) \odot \boldsymbol{x}_2 + m(\boldsymbol{x}_1)$$

- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s







Dinh et al. 2016

Real NVP via Masked Convolution

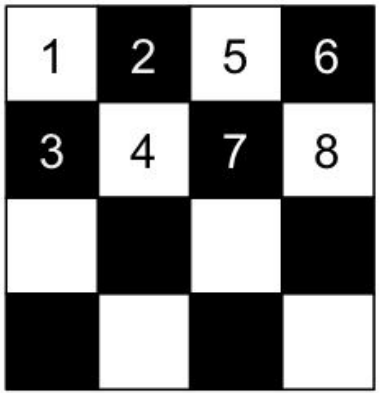
Partitioning can be implemented using a binary mask b, and using the functional form for y

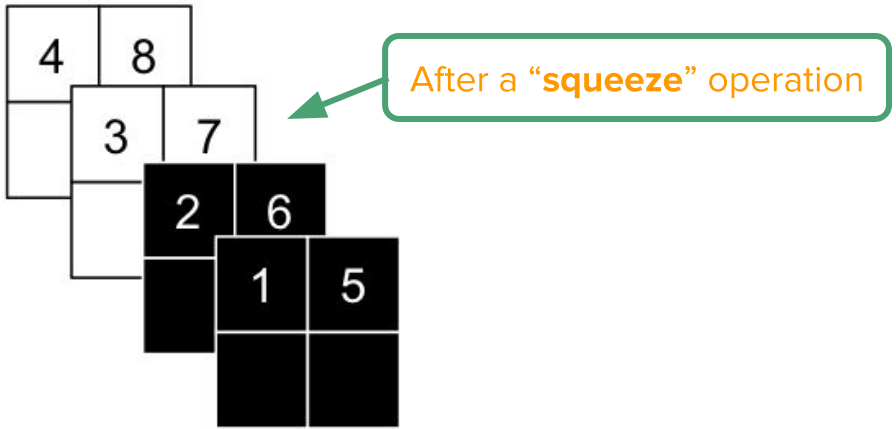
$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s\_(b \odot x)) + m(b \odot x))$$

Real NVP via Masked Convolution

Partitioning can be implemented using a binary mask b, and using the functional form for y

$$f(x) = b \odot x + (1 - b) \odot (x \odot \exp(s\_(b \odot x)) + m(b \odot x))$$





Figures from *Density Estimation Using Real NVP* by Dinh et al., 2017

Celeba-64 (left) and LSUN bedroom (right)

Figures from *Density Estimation Using Real NVP* by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

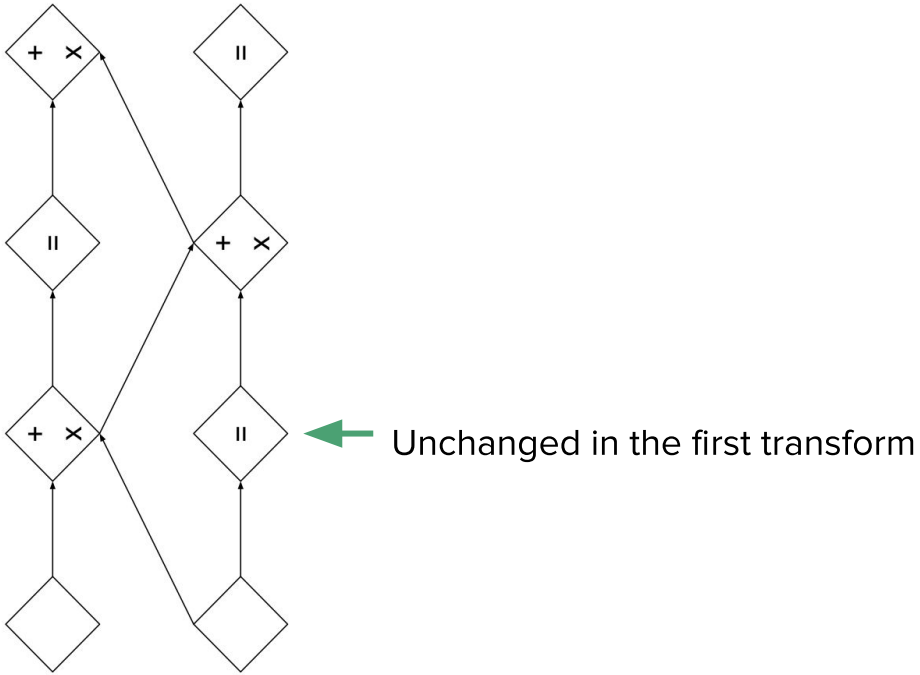
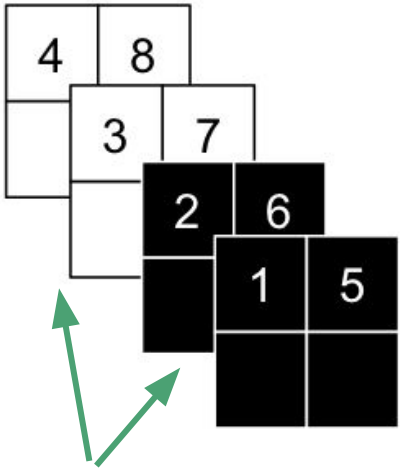


Figure from *Density Estimation Using Real NVP* by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)

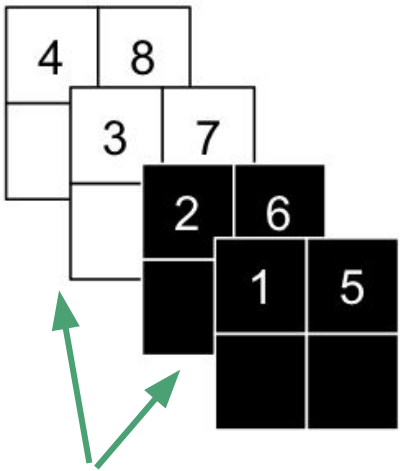


Alternating masks

Figure from *Density Estimation Using Real NVP* by Dinh et al., 2017

Glow: Generative Flow with 1x1 Convolutions

Replacing permutation with 1x1 convolution (soft permutation)



Alternating masks

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ x_1 \\ x_2 \end{bmatrix}$$

Replace with a general invertible matrix W

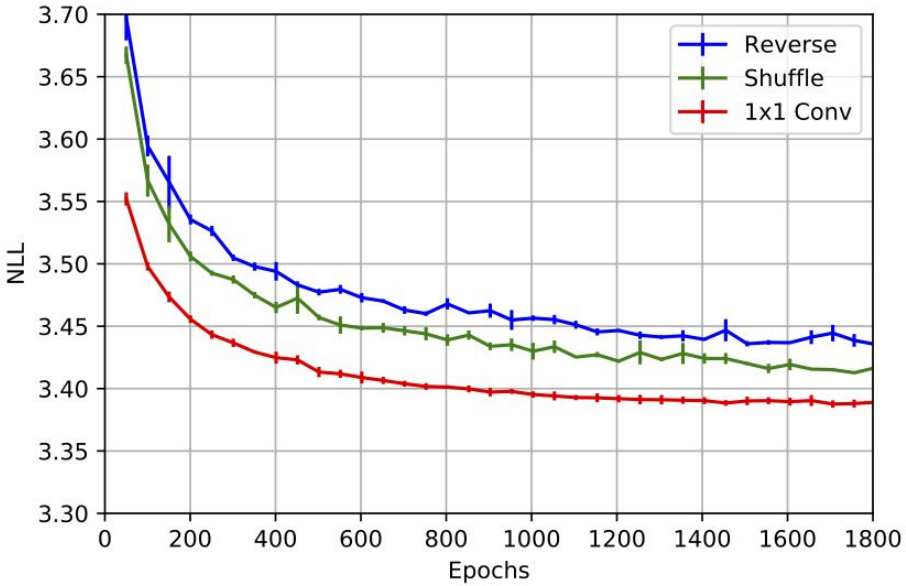
Represent W as a 1x1 convolutional kernel of shape [c, c, 1, 1]; c being # channels

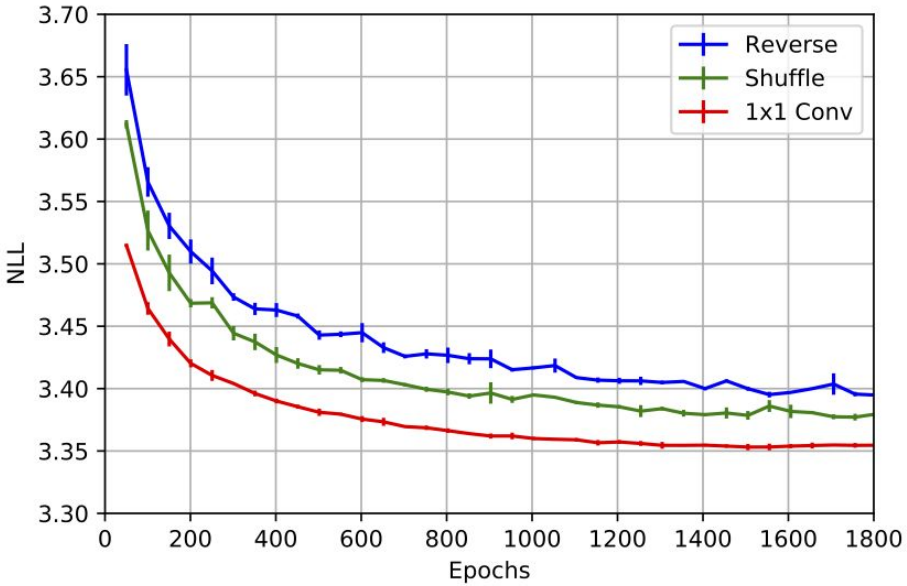
$$\log\left|\det\left(\frac{\partial\,\mathrm{conv2D}(h;W)}{\partial h}\right)\right| = h \cdot w \cdot \log\left|\det(W)\right|$$

Figure from *Density Estimation Using Real NVP* by Dinh et al., 2017

Ablation: Permutation vs 1x1 Convolution

Model	CIFAR-10	ImageNet 32x32	ImageNet 64x64	LSUN (bedroom)	LSUN (tower)	LSUN (church outdoor)
RealNVP	3.49	4.28	3.98	2.72	2.81	3.08
Glow	**3.35**	**4.09**	**3.81**	**2.38**	**2.46**	**2.67**

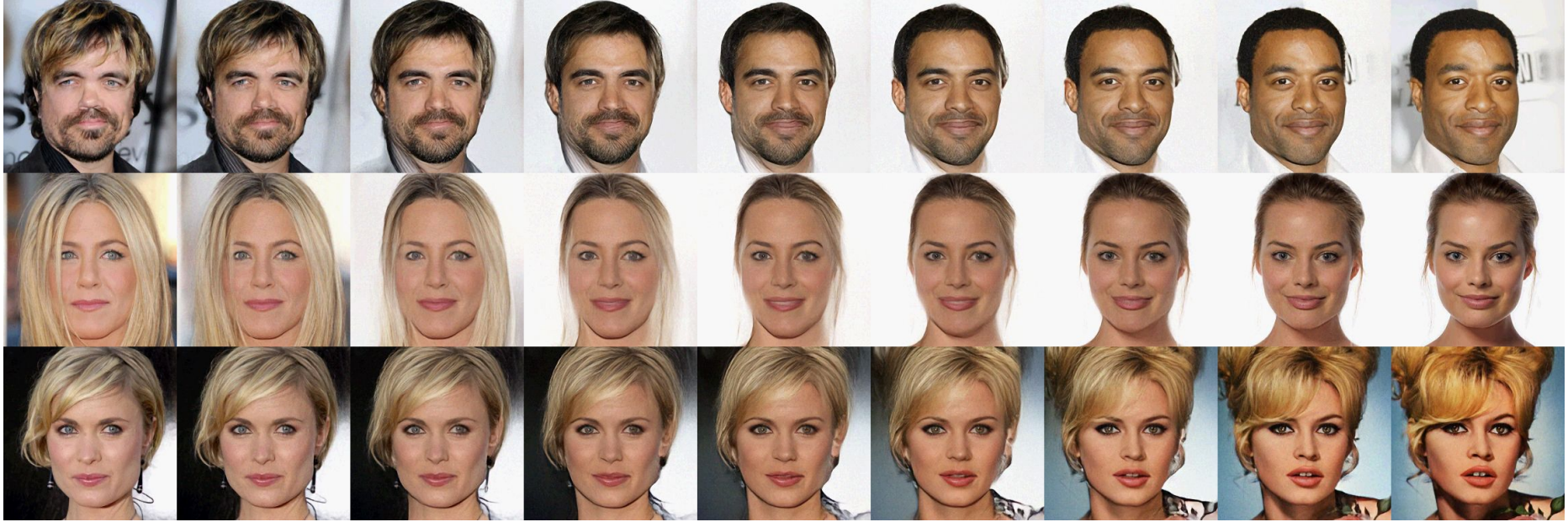




Bits-per-dim on CIFAR: left: additive, right: affine

Results from *Glow: Generative Flow with Invertible 1×1 Convolutions* by Kingma and Dhariwal, 2018

Figure from *Glow: Generative Flow with Invertible 1×1 Convolutions* by Kingma and Dhariwal, 2018



Interpolation with Generative Flows

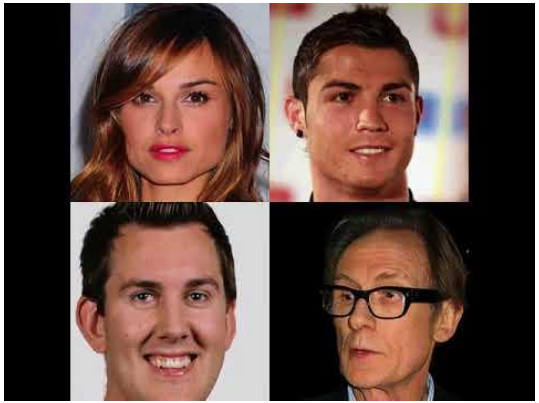


Figure from *Glow: Generative Flow with Invertible 1×1 Convolutions* by Kingma and Dhariwal, 2018
Video from Durk Kingma's youtube channel

Architectural Taxonomy

Sparse connection

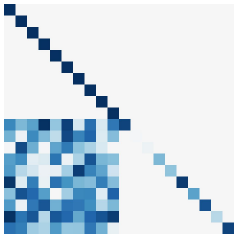
$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

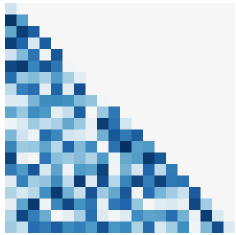
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular + structured)

2. Autoregressive

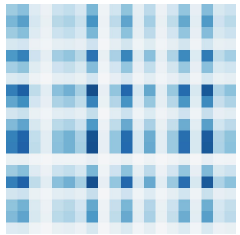
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

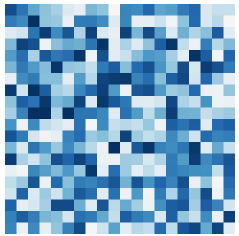
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD



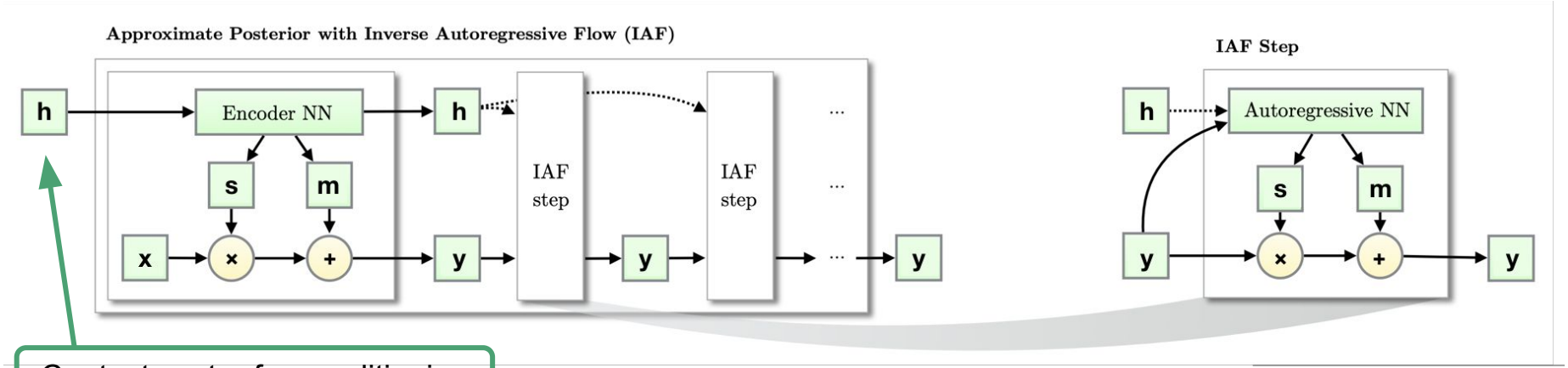
(Arbitrary)

Jacobian

Figures from Ricky Chen

Inverse (Affine) Autoregressive Flows

- General form $f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$
- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s

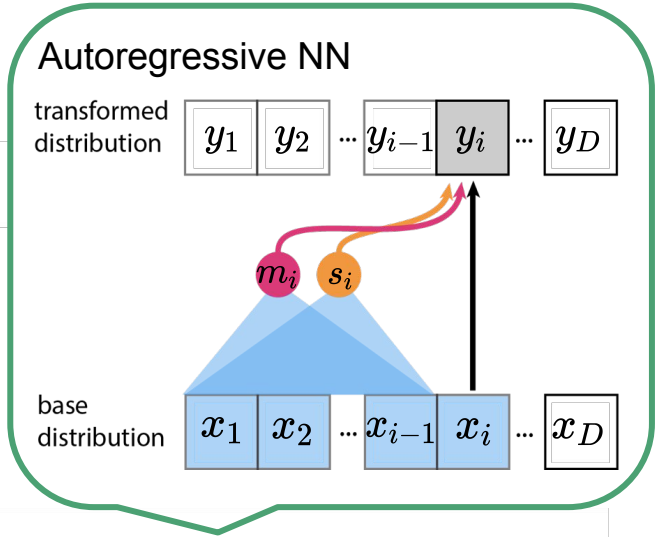


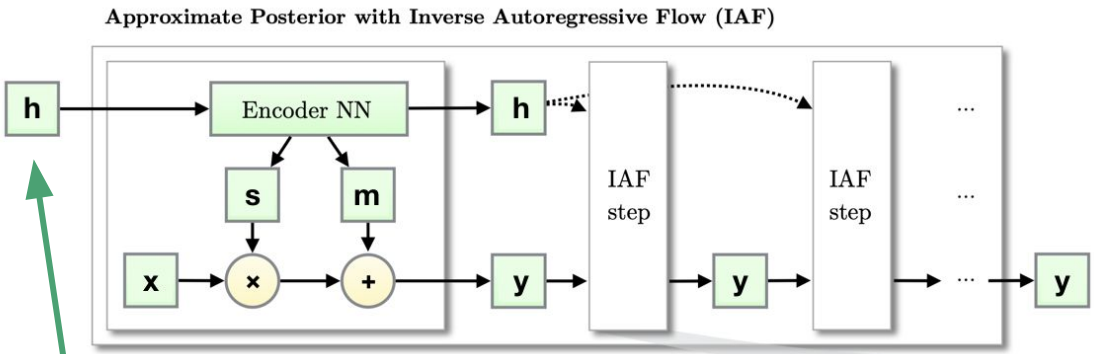
Context vector for conditioning

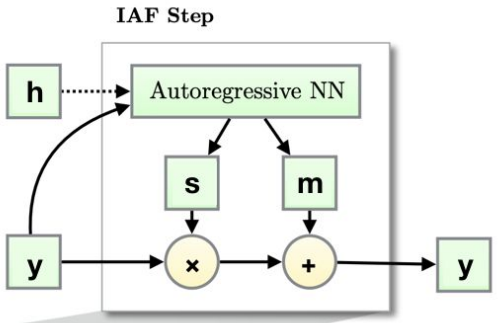
Modified from Kingma et al. 2016

Inverse (Affine) Autoregressive Flows

- General form $f(\boldsymbol{x})_t = s(\boldsymbol{x}_{<t}) \cdot \boldsymbol{x}_t + m(\boldsymbol{x}_{<t})$
- Invertibility s>0 (or simply non-zero)
- Jacobian determinant product of s





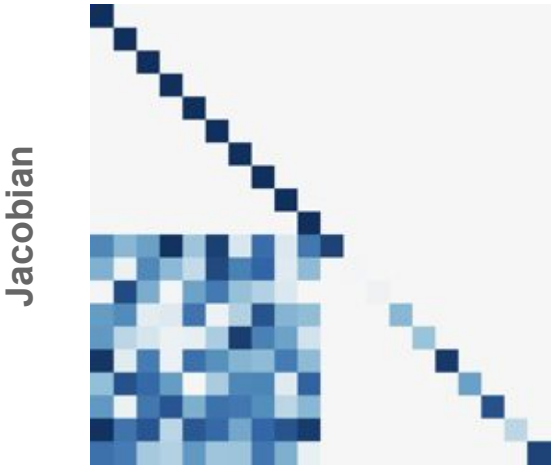


Context vector for conditioning

Trade-off between Expressivity and Inversion Cost

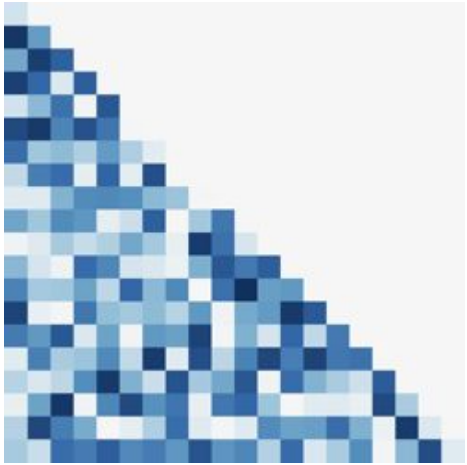
Block autoregressive

- Limited capacity
- Inverse takes constant time



Autoregressive

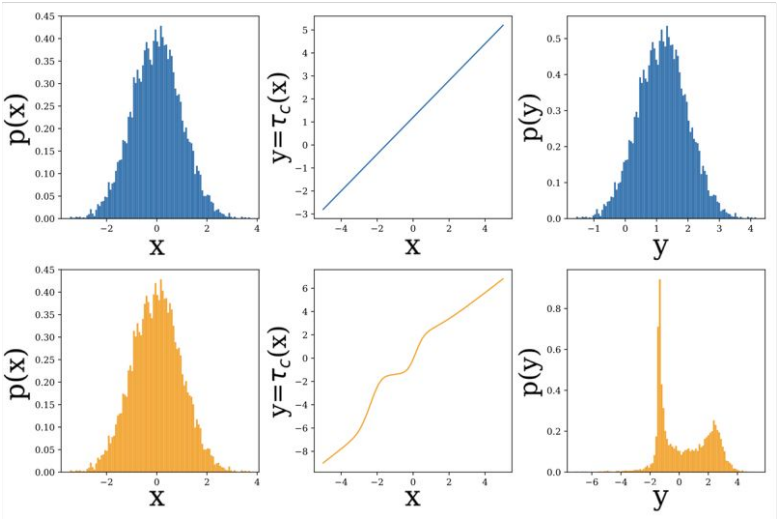
- Higher capacity
- Inverse takes linear time (dimensionality)

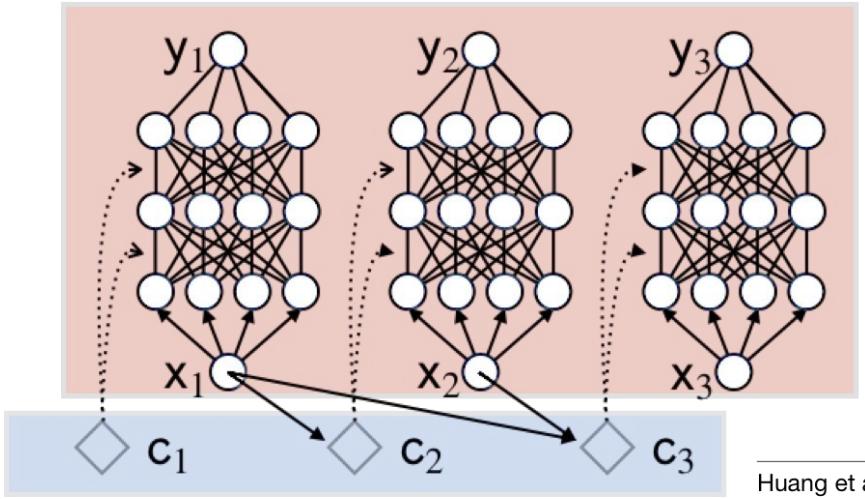


Figures from Ricky Chen

Neural Autoregressive Flows

- General form $f(\boldsymbol{x})_t = \mathcal{P}(\boldsymbol{x}_t; \mathcal{H}(\boldsymbol{x}_{<t}))$
- Invertibility monotonic activation and positive weight in $\mathcal{P}$
- Jacobian determinant product of derivatives (elementwise)





Huang et al. 2018

Inverse Transform Sampling

Let F be a cumulative distribution function (CDF) on the real space with inverse

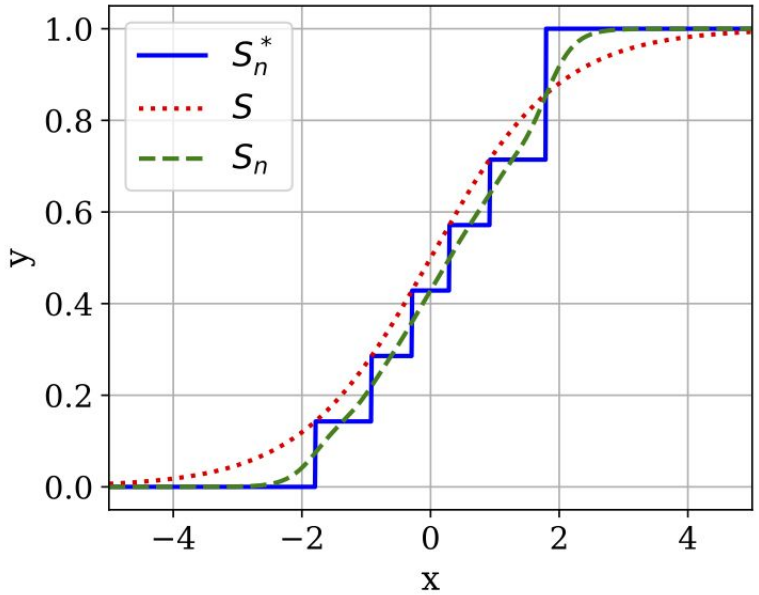
$$G(u) := F^{-1}(u) := \inf\{x : F(x) = u\} \quad \forall\, 0 < u < 1$$

Theorem (Probability Integral transform & Inverse Probability integral transform)

> If F is the CDF of X, then Y:=F(X) has a uniform distribution on [0,1].
>
> Conversely, if Y follows a uniform distribution on [0,1], X:=G(Y) has F as Its CDF.

Universal Flows in 1D

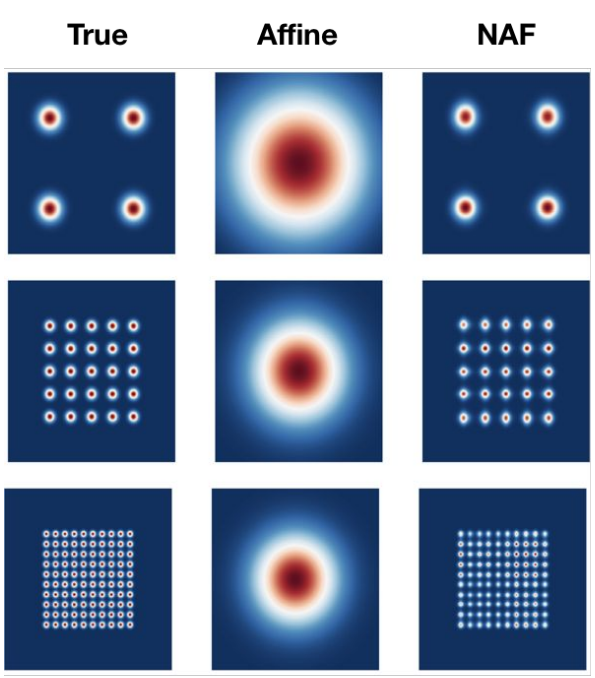
Assume the inverse CDF G(u) of a random variable X is continuous, then one can use a monotonic neural network to approximate G (to an arbitrary precision).



Further reading: the Knothe-Rosenblatt rearrangement for higher dimensionality.

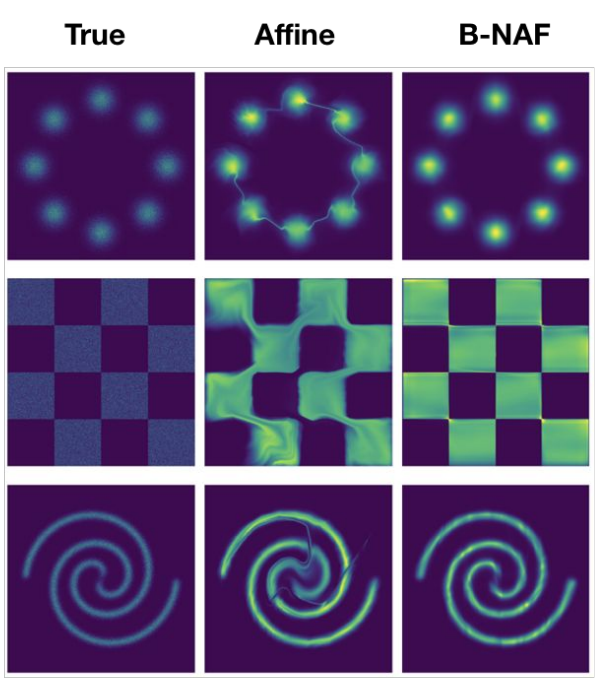
Figure from *Neural Autoregressive Flows* by Huang et al., 2018

Examples of Universal Flows



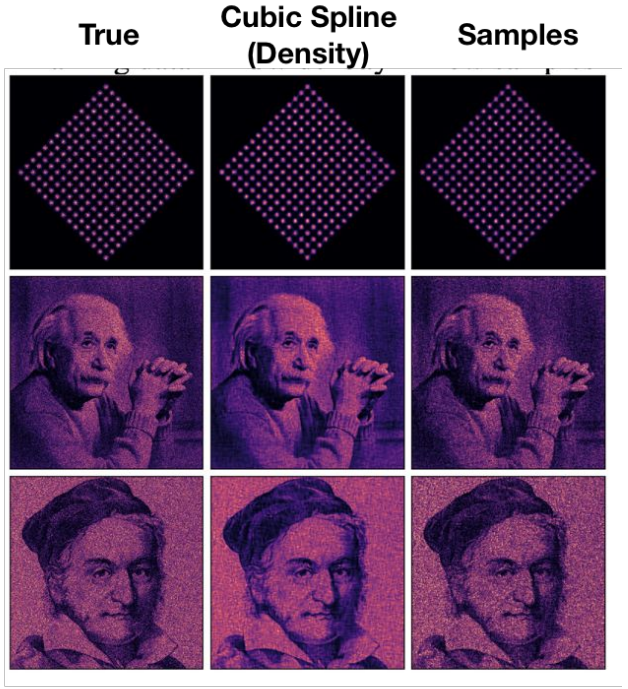
Neural Autoregressive Flows

Huang et al. 2018



Block Neural Autoregressive Flows

De Cao et al. 2019



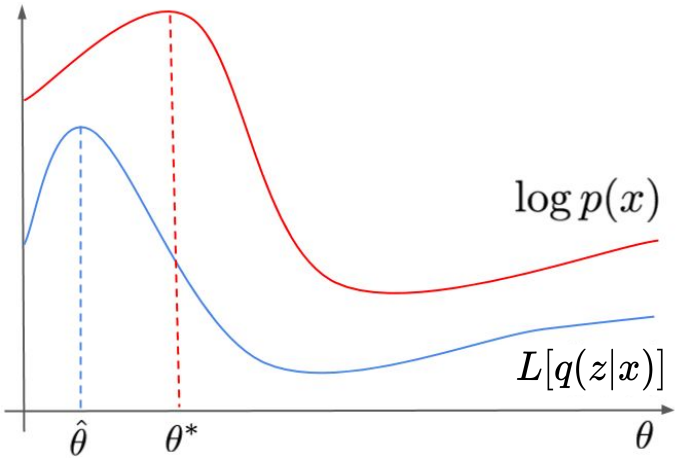
Cubic-Spline Flows

Durkan et al. 2019

A Digression on Variational Inference

The variational gap can be reduced by considering a larger family of distributions.

$$\log p(x) - L[q(z|x)] = \underbrace{\log p(x) - L[q^*]}_{\text{Approximation gap}} + \underbrace{L[q^*] - L[q(z|x)]}_{\text{Amortization gap}}$$



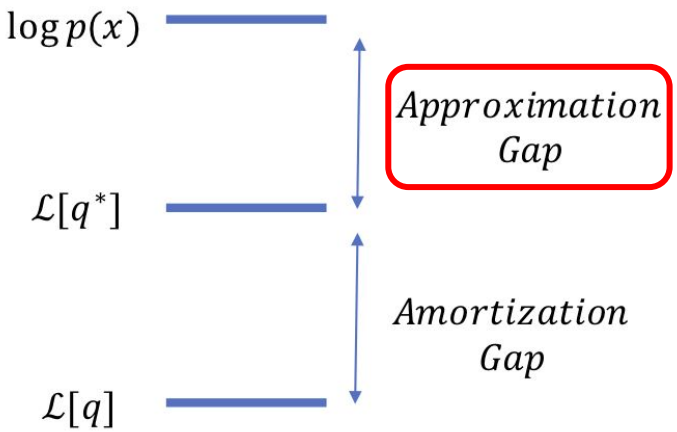


Figure from *Sylvester Normalizing Flows for Variational Inference* by Berg et al., 2018

Figure from *Inference Suboptimality in Variational Autoencoders* by Cremer et al., 2018

A Digression on Variational Inference

The variational gap can be reduced by considering a larger family of distributions.

Recall the *reparameterization trick*:

Given some function g_ϕ such that for some $\epsilon \sim q_\epsilon$, $g_\phi(\epsilon) \overset{d}{=} z \sim q_\phi(z)$
g_ϕ depends on ϕ but q_ϵ does not.

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z)\right] = \nabla_\phi \mathbb{E}_{\epsilon \sim q_\epsilon(\epsilon)} \left[\log p_\theta(x, g_\phi(\epsilon)) - \log q_\phi(g_\phi(\epsilon))\right]$$

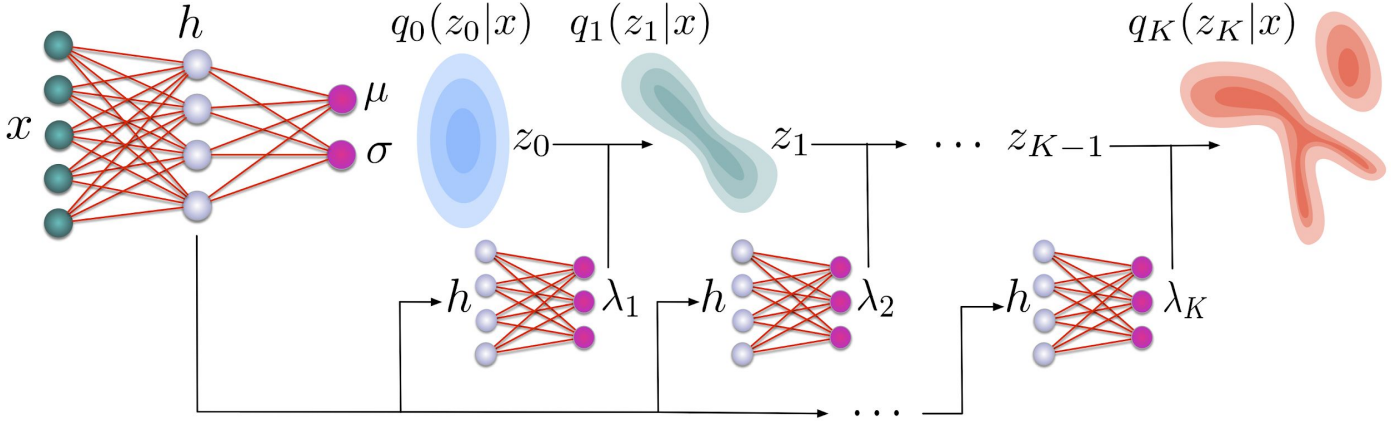
If furthermore, g_ϕ is invertible, the entropy of z can be estimated by

$$-\mathbb{E}_\epsilon \left[\log q_\phi(g_\phi(\epsilon))\right] = -\mathbb{E}_\epsilon \left[\log q_\epsilon(\epsilon) \left| \det \frac{\partial g_\phi(\epsilon)}{\partial \epsilon} \right|^{-1}\right]$$

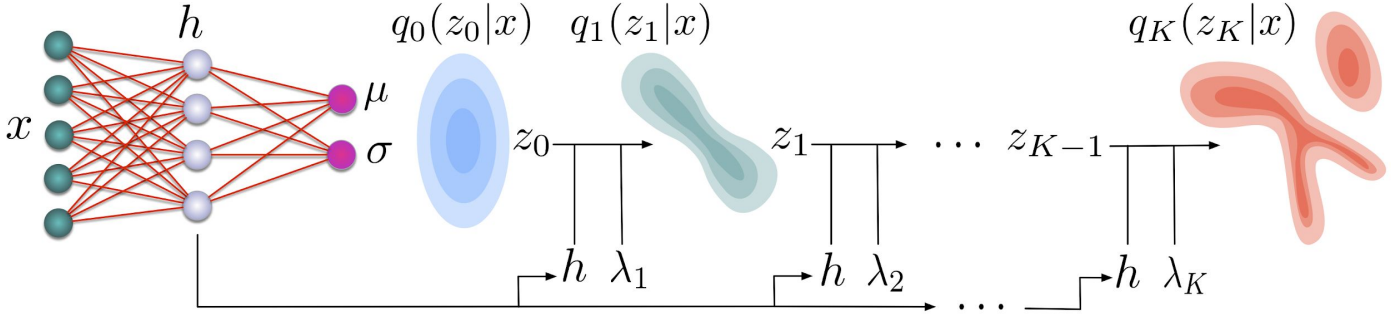
Prevents the distribution of z from collapsing to a point mass

Amortization Strategies

Hypernetwork



Concatenation



Figures from *Sylvester Normalizing Flows for Variational Inference* by Berg et al., 2018

Architectural Taxonomy

Sparse connection

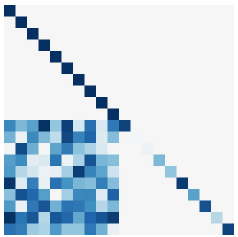
$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

1. Block coupling

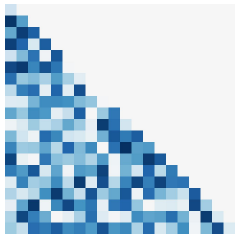
NICE/RealNVP/Glow
Cubic Spline Flow
Neural Spline Flow



(Lower triangular + structured)

2. Autoregressive

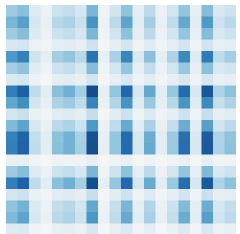
IAF/MAF/NAF
SOS polynomial
UMNN



(Lower triangular)

3. Det identity

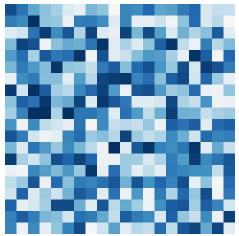
Planar/Sylvester flows
Radial flow



(Low rank)

4. Stochastic estimation

Residual Flow
FFJORD



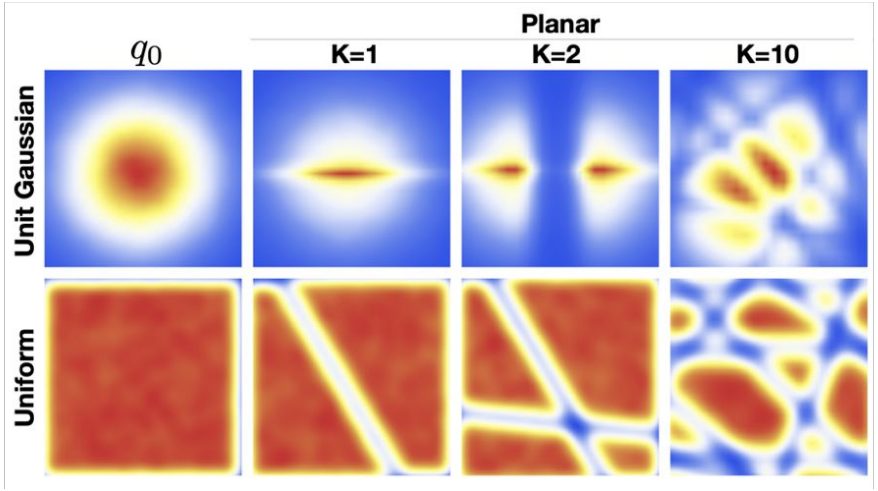
(Arbitrary)

Jacobian

Figures from Ricky Chen

Determinant Identity - Planar Flows

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{x} + b)$
- Invertibility $\boldsymbol{u}^\top \boldsymbol{w} > -1 \text{ if } h = \tanh$
- Jacobian determinant $\left|\det \frac{\partial f}{\partial \boldsymbol{x}}\right| = \left|\det \left(\boldsymbol{I} + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}\boldsymbol{w}^\top\right)\right| = \left|1 + h'(\boldsymbol{w}^\top \boldsymbol{x} + b)\boldsymbol{u}^\top \boldsymbol{w}\right|$



VAE on binary MNIST

Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	$\leq$ 89.9
DLGM+NF (k = 10)	$\leq$ 87.5
DLGM+NF (k = 20)	$\leq$ 86.5
DLGM+NF (k = 40)	$\leq$ 85.7
DLGM+NF (k = 80)	$\leq$ 85.1

Rezende et al. 2015

Determinant Identity - Sylvester Flows

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{A}h(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{b})$ $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{B} \in \mathbb{R}^{d \times m}$, $\boldsymbol{b} \in \mathbb{R}^{d}$, and $d \leq m$
- Invertibility Similar to planer flows
- Jacobian determinant Using Sylvester's Thm: $\det(\boldsymbol{I}_m + \boldsymbol{A}\boldsymbol{B}) = \det(\boldsymbol{I}_d + \boldsymbol{B}\boldsymbol{A})$

Model	Freyfaces		Omniglot		Caltech 101	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE	4.53 ± 0.02	4.40 ± 0.03	104.28 ± 0.39	97.25 ± 0.23	110.80 ± 0.46	99.62 ± 0.74
Planar	$\mathbf{4.40 \pm 0.06}$	$\mathbf{4.31 \pm 0.06}$	102.65 ± 0.42	96.04 ± 0.28	109.66 ± 0.42	98.53 ± 0.68
IAF	4.47 ± 0.05	4.38 ± 0.04	102.41 ± 0.04	96.08 ± 0.16	111.58 ± 0.38	99.92 ± 0.30
O-SNF	4.51 ± 0.04	4.39 ± 0.05	99.00 ± 0.29	93.82 ± 0.21	106.08 ± 0.39	94.61 ± 0.83
H-SNF	4.46 ± 0.05	4.35 ± 0.05	$\mathbf{99.00 \pm 0.04}$	$\mathbf{93.77 \pm 0.03}$	$\mathbf{104.62 \pm 0.29}$	$\mathbf{93.82 \pm 0.62}$
T-SNF	4.45 ± 0.04	4.35 ± 0.04	99.33 ± 0.23	93.97 ± 0.13	105.29 ± 0.64	94.92 ± 0.73

Van den Berg et al. 2018

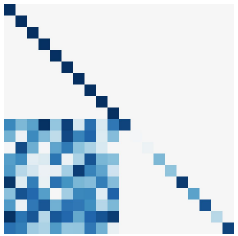
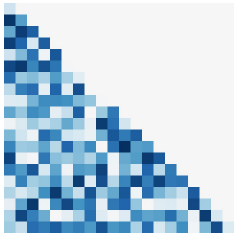
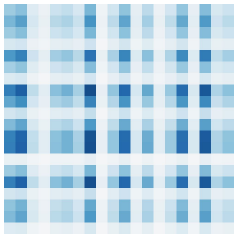
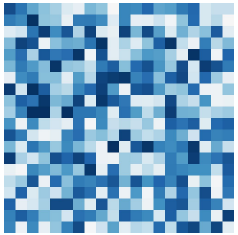
Architectural Taxonomy

Sparse connection

$$f(\boldsymbol{x})_t = g(\boldsymbol{x}_{1:t})$$

Residual Connection

$$f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$$

	1. Block coupling	**2. Autoregressive**	**3. Det identity**	**4. Stochastic estimation**
	NICE/RealNVP/Glow Cubic Spline Flow Neural Spline Flow	IAF/MAF/NAF SOS polynomial UMNN	Planar/Sylvester flows Radial flow	Residual Flow FFJORD
Jacobian				
	(Lower triangular + structured)	(Lower triangular)	(Low rank)	(Arbitrary)

Figures from Ricky Chen

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

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

Theorem 1. *If $g(x)$ is L-Lipschitz for $0 < L < 1$, then $f(x) := x + g(x)$ is invertible.*

Proof Fix any y in the codomain of f. Define $x_0 = y$ and $x_t = y - g(x_{t-1})$ for $t \geq 1$, which is a contractive mapping. Thus, by the Banach fixed-point theorem, x_t converges to a unique fixed point x^* which satisfies $f(x^*) = y$.

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x}^{k} \right)$$

Power series expansion

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x}^k \right)$$

Power series expansion

$$\approx \mathbb{E}_v \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^\top \left(\frac{\partial g(x)}{\partial x}^k \right) v \right]$$

Truncation &
Hutchinson trace estimator

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x}^k \right)$$

Power series expansion

$$\approx \mathbb{E}_v \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} v^\top \left(\frac{\partial g(x)}{\partial x}^k \right) v \right]$$

Truncation &
Hutchinson trace estimator

Bias

Behrmann et al. 2018

Stochastic Estimation for General Residual Form

- General form $f(\boldsymbol{x}) = \boldsymbol{x} + g(\boldsymbol{x})$
- Invertibility $\left\| \frac{\partial g(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_2 < 1$
- Jacobian determinant

$$\log \left| \det \frac{\partial f(x)}{\partial x} \right| = \operatorname{tr} \left(\log \frac{\partial f(x)}{\partial x} \right)$$

Jacobi's formula

$$\operatorname{tr} \left(\log \left(\mathbf{I} + \frac{\partial g(x)}{\partial x} \right) \right) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \operatorname{tr} \left(\frac{\partial g(x)}{\partial x}^{k} \right)$$

Power series expansion

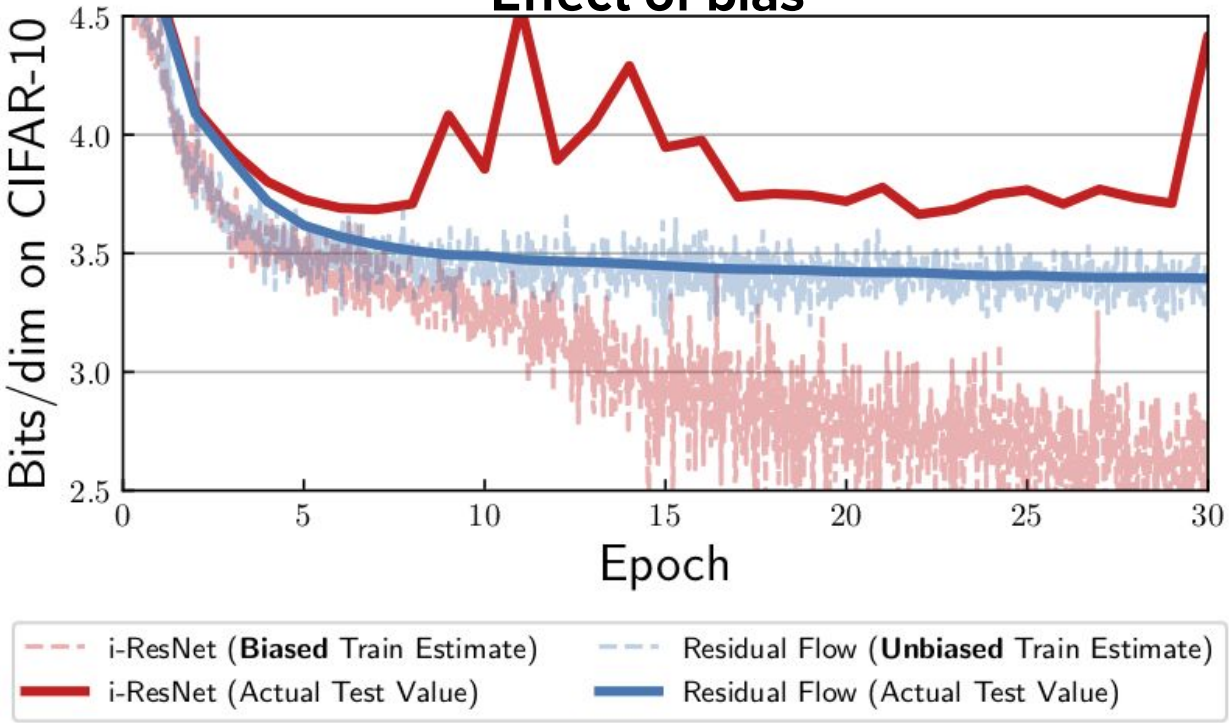
$$= \mathbb{E}_{v,n} \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k \cdot \mathbb{P}(N \geq k)} v^{\top} \left(\frac{\partial g(x)}{\partial x}^{k} \right) v \right]$$

Russian roulette estimator & Hutchinson trace estimator

Behrmann et al. 2018

Chen et al. 2019

Effect of bias



CelebA samples

Cifar10 samples

Imagenet-32 samples

Figures from *Residual Flows for Invertible Generative Modeling* by Chen et al., 2019

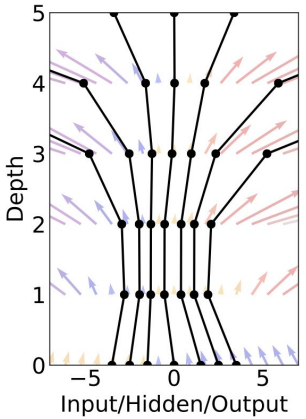
Continuous Time Flows

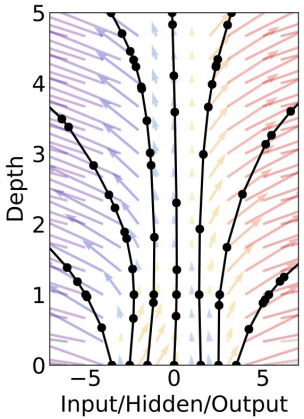
Neural ODE

A neural ordinary differential equation (neural ODE) parameterizes the **time derivative** of a function using a neural network

$$\frac{dz(t)}{dt} = f(z(t), t, \theta) \qquad z(t_1) = z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta) dt$$

ResNet





ODE Network

Figures from *Neural Ordinary Differential Equations* by Chen et al., 2018

Backpropagation of Neural ODE

Adjoint sensitivity method enables reverse-mode differentiation (backprop)

$$a(t) := \frac{\partial L(z(t_1))}{\partial z(t)} \qquad \frac{da(t)}{dt} = -a(t)^\top \frac{\partial f(z(t),t,\theta)}{\partial z}$$

Adjoint variable

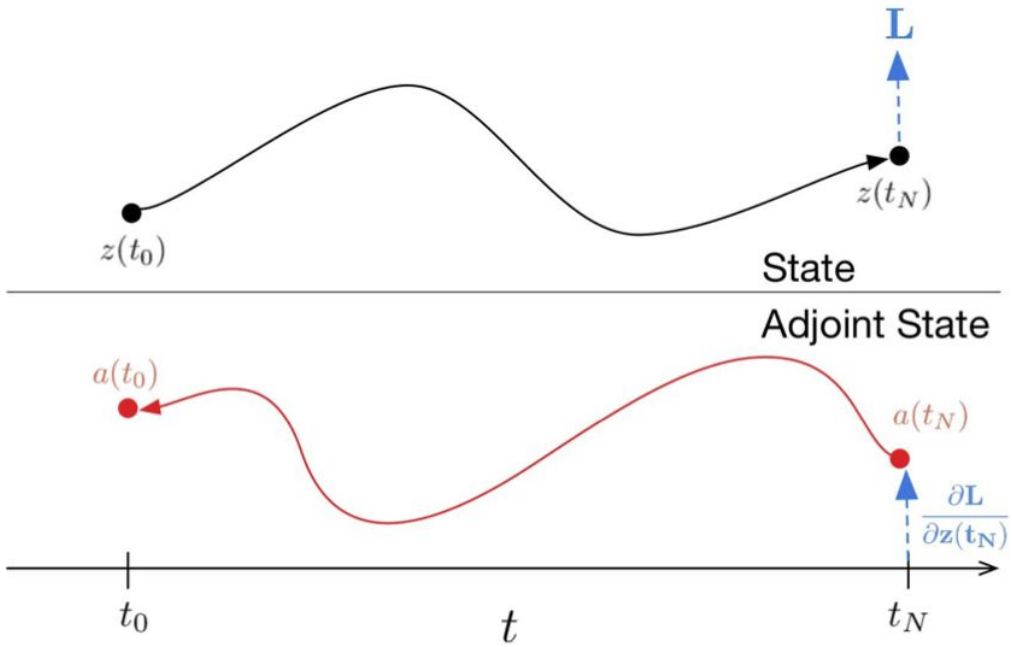
Solve for a in reverse time

Recompute z backwards
Constant memory!

A Differentiable Primitive for AutoDiff

Forward:

Backward:

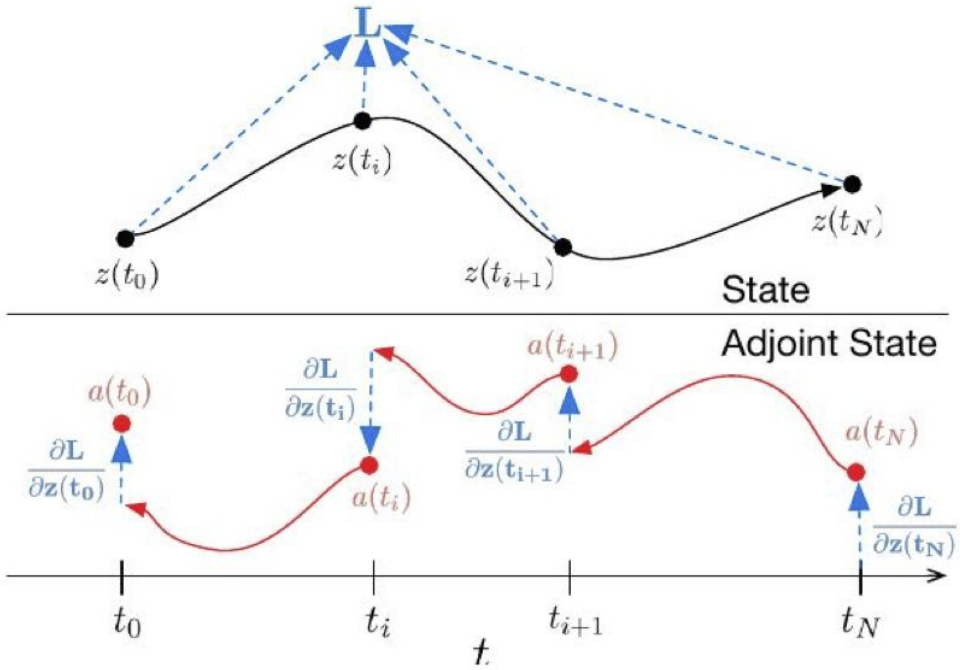


Slide by Ricky Chen

A Differentiable Primitive for AutoDiff

Forward:

Backward:



Slide by Ricky Chen

Backpropagation of Neural ODE (cont'd)

Adjoint sensitivity method enables reverse-mode differentiation (backprop)

$$a(t) := \frac{\partial L(z(t_1))}{\partial z(t)} \qquad \frac{da(t)}{dt} = -a(t)^\top \frac{\partial f(z(t),t,\theta)}{\partial z}$$

Adjoint variable

Solve for a in reverse time (2)

Recompute z backwards (1)
Constant memory!

Gradient wrt parameters

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} a(t)^\top \frac{\partial f(z(t),t,\theta)}{\partial \theta} dt$$ (3)

Solve 1 ODE forward
Solve 3 ODEs backward

Uniqueness of Solution ⇒ Invertibility

Theorem (Picard-Lindelof)

The initial value problem $\frac{dz(t)}{dt} = f(z(t), t, \theta) \quad z(0) = z_0$

has a unique solution if f is ***uniformly Lipschitz in time***.

Reverse time always has a unique solution ⇒ invertible!

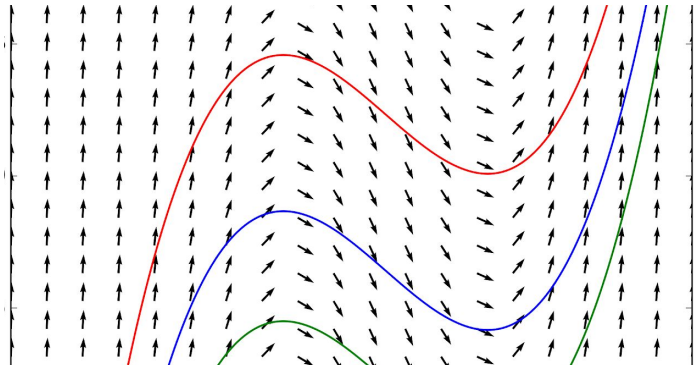


Figure from blog post: *Neural Ordinary Differential Equations and Dynamics Models* by Aidan Abdulali, 2019

Instantaneous Change of Variables

Theorem (Instantaneous Change of Variables, Chen et al., 2018)

Assume f is ***uniformly Lipschitz in time***, and z(t) follows the probability density function p(z(t), t). Then the change in log probability function follows the ODE

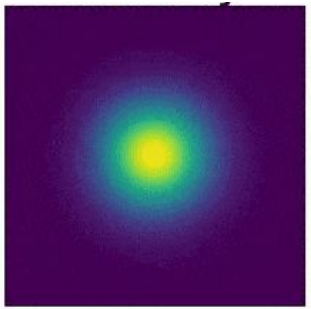
$$\frac{\partial \log p(z(t),t)}{\partial t} = -\mathrm{tr}\left(\frac{df}{dz(t)}\right)$$

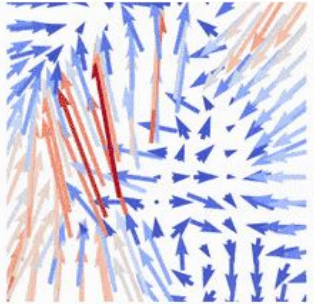
Instantaneous Change of Variables

Theorem (Instantaneous Change of Variables, Chen et al., 2018)

Assume f is ***uniformly Lipschitz in time***, and z(t) follows the probability density function p(z(t), t). Then the change in log probability function follows the ODE

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





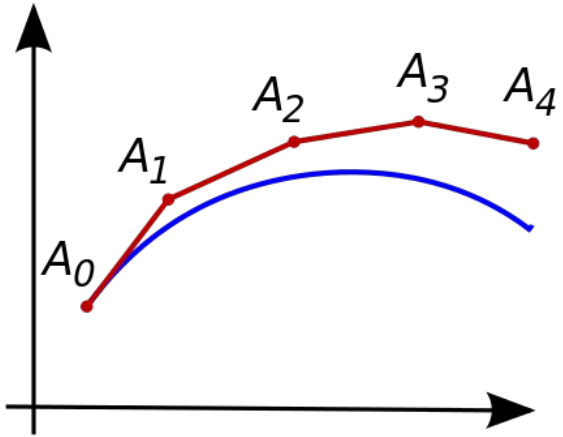
Gif from *https://github.com/rtqichen/ffjord*

Figures from *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models* by Grathwohl et al., 2018

Residual Flows as Finite Neural ODE

If f is Lipschitz, then the Euler method for numerical integration is invertible given that the step size is sufficiently small

$$z_{t+\epsilon} = z_t + \epsilon \cdot f(z_t, t, \theta)$$



Make it <1-Lipschitz by choosing step size small

Figure from *Euler method (wikipedia)*