

Normalizing Flows

Didrik Nielsen



Where are flows useful?

Anywhere you need a flexible density $p(\mathbf{x})$ with

Sampling:

$$\mathbf{x} \sim p(\mathbf{x})$$

Density:

$$\mathbf{x} \mapsto \log p(\mathbf{x})$$



Why Flows?

	Flow	ARM	VAE	GAN	EBM
Density	Exact Fast	Exact Fast	Approx. Fast	NA	Unnorm.
Sampling	Fast	Slow	Fast	Fast	Slow & Approx.

History

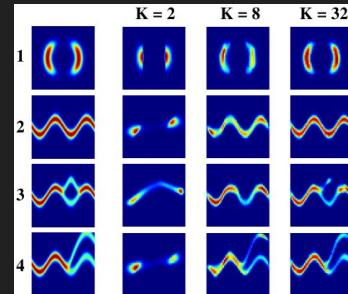
Normalizing Flows: Originates from (Tabak & Turner 2013; Tabak & Vanden-Eijnden 2010)
Describes the **change of probability density** through a series of **invertible maps**

Popularized by (Rezende & Mohammed 2015):

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende
Shakir Mohamed
Google DeepMind, London

DANILOR@GOOGLE.COM
SHAKIR@GOOGLE.COM



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

First deep generative flows: NICE (Dinh et al. 2014); RealNVP (Dinh et al. 2016):

NICE: NON-LINEAR INDEPENDENT COMPONENTS ESTIMATION

Laurent Dinh David Krueger Yoshua Bengio*
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7



DENSITY ESTIMATION USING REAL NVP

Laurent Dinh*
Montreal Institute for Learning Algorithms
University of Montreal
Montreal, QC H3T1J4

Jascha Sohl-Dickstein
Google Brain

Samy Bengio
Google Brain



Outline

1. Framework
2. Coupling Flows,
Autoregressive Flows
3. Continuous-time Flows,
Residual Flows
4. Discrete Flows,
Surjective Flows

The Flow Framework

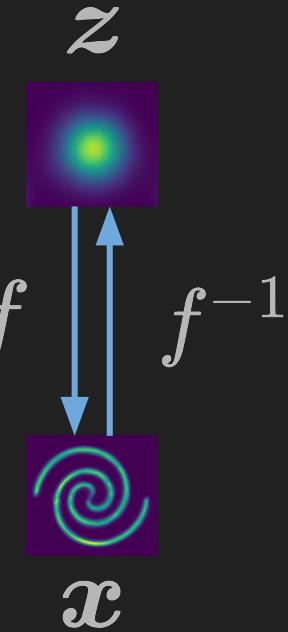
Construct $p(\mathbf{x})$ using

- Base distribution $p(\mathbf{z})$
- Bijective mapping f

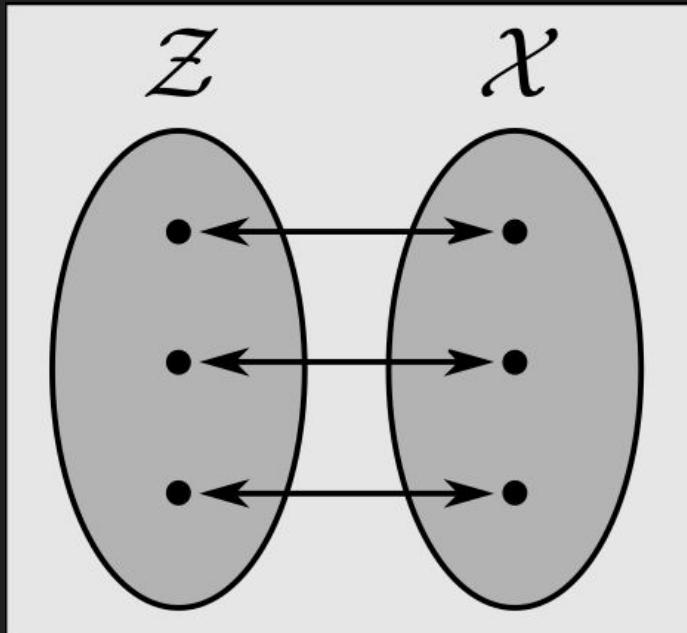
Sampling:

$$\mathbf{z} \sim p(\mathbf{z})$$

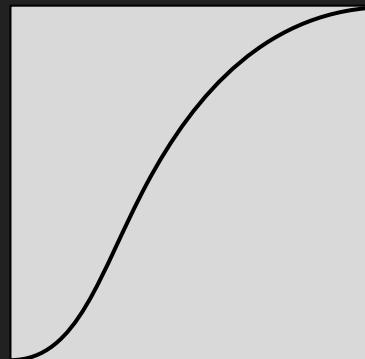
$$\mathbf{x} = f(\mathbf{z})$$



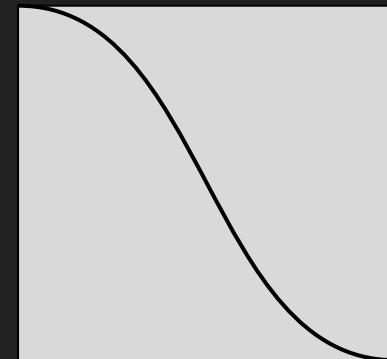
Flows: Invertibility



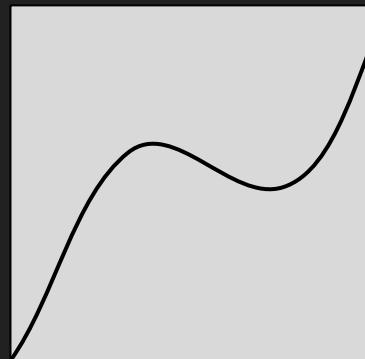
In 1D:
Strictly increasing or decreasing



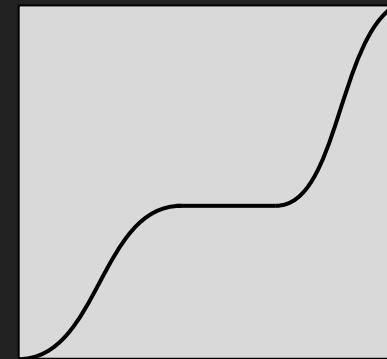
Invertible



Invertible



Not Invertible



Not Invertible

The Flow Framework

Construct $p(\mathbf{x})$ using

- Base distribution $p(\mathbf{z})$
- Bijective mapping f

Sampling:

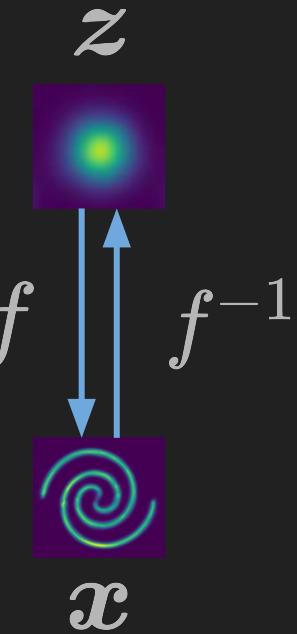
$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} = f(\mathbf{z})$$

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

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

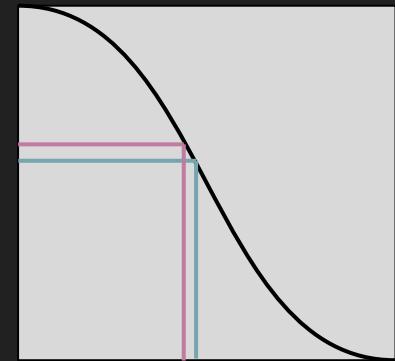
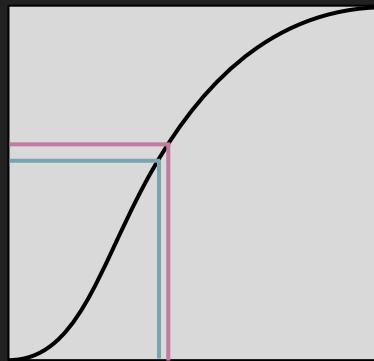


Change-of-Variables in 1D

$$\begin{aligned} \int_{x_1}^{x_2} \underline{p(x)dx} &= \int_{z_1}^{z_2} p(z)dz \\ &= \int_{?}^? \underline{p(z) \frac{dz}{dx} dx} \end{aligned}$$

$$\frac{dz}{dx} > 0$$

$$\frac{dz}{dx} < 0$$



$$p(x) = p(z) \left| \frac{dz}{dx} \right|$$

$$\int_{x_1}^{x_2} p(x)dx$$

$$= \int_{\boxed{z_1}}^{\boxed{z_2}} p(z)dz$$

$$= \int_{\boxed{x_1}}^{\boxed{x_2}} p(z) \frac{dz}{dx} dx$$

$$\int_{x_1}^{x_2} p(x)dx$$

$$= \int_{\boxed{z_1}}^{\boxed{z_2}} p(z)dz$$

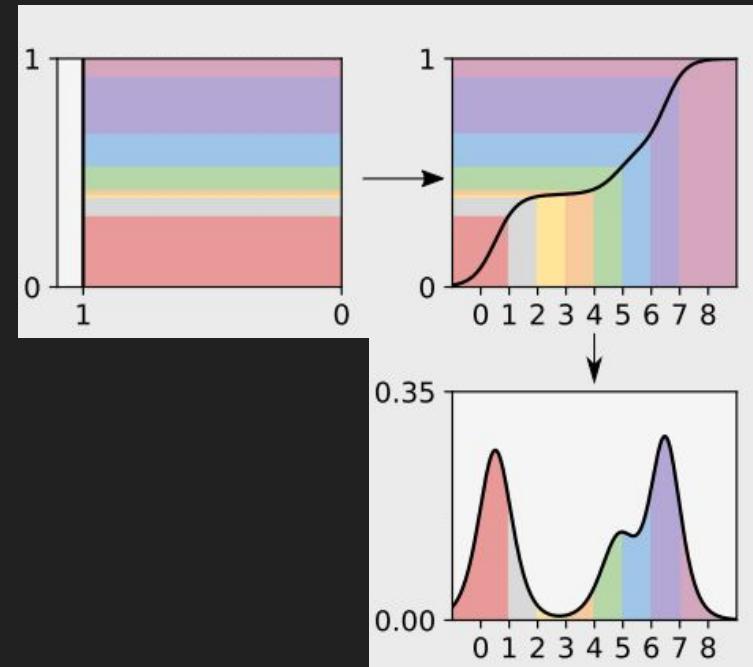
$$= \int_{\boxed{x_2}}^{\boxed{x_1}} p(z) \frac{dz}{dx} dx$$

$$= \int_{x_1}^{x_2} p(z) \left[-\frac{dz}{dx} \right] dx$$

Flows: 1D Example

Inverse transform sampling:

$$\begin{aligned} p(x) &= \text{Unif}(z|0, 1) \left| \frac{dCDF(x)}{dx} \right| \\ &= \left| \frac{dCDF(x)}{dx} \right| \end{aligned}$$

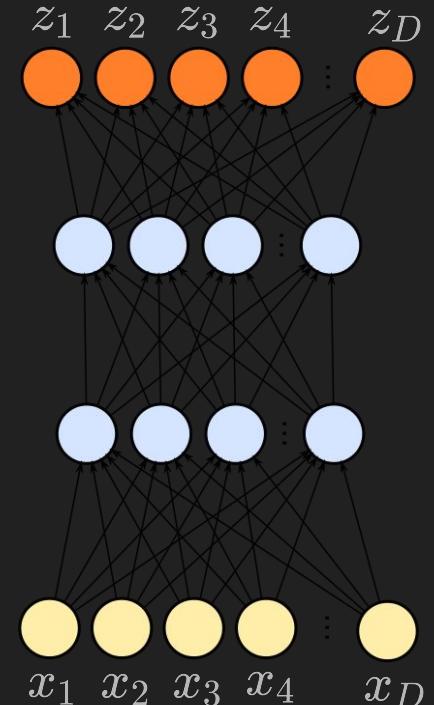


$$p(x) = p(z) \left| \frac{dz}{dx} \right|$$

Change-of-Variables

$$\mathbf{J} = \boxed{\frac{\partial \mathbf{z}}{\partial \mathbf{x}}} = \underbrace{\begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_D} \\ \vdots & & \vdots \\ \frac{\partial z_D}{\partial x_1} & \cdots & \frac{\partial z_D}{\partial x_D} \end{bmatrix}}_{D \times D}$$

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \boxed{\frac{\partial \mathbf{z}}{\partial \mathbf{x}}} \right|$$



The Flow Framework

Construct $p(\mathbf{x})$ using

- Base distribution $p(\mathbf{z})$
- Bijective mapping f

Sampling:

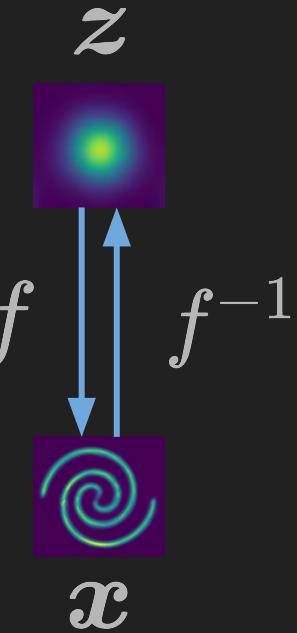
$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} = f(\mathbf{z})$$

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

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



The Flow Framework

Sampling:

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\underline{\mathbf{x} = f(\mathbf{z})}$$

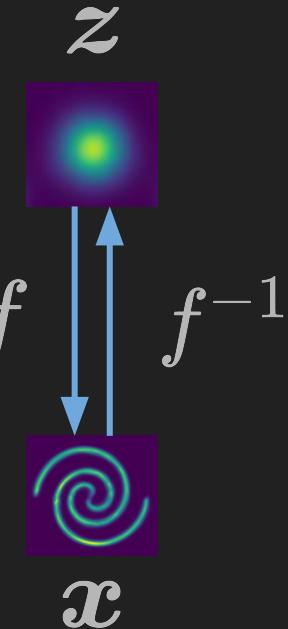
Forward

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

$$\underline{\mathbf{z} = f^{-1}(\mathbf{x})}$$

Inverse



The Flow Framework

Sampling:

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} = f_T \circ \dots \circ f_1(\mathbf{z})$$

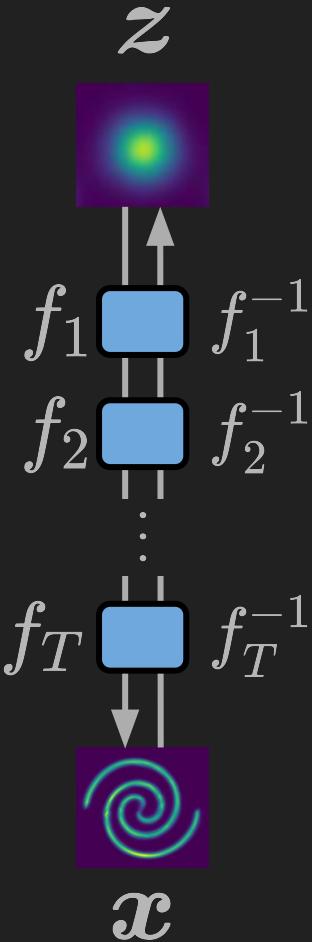
Forward

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \sum_{t=1}^T \log \left| \det \frac{\partial \mathbf{z}_{t-1}}{\partial \mathbf{z}_t} \right|, \quad \text{for } \mathbf{z}_T = \mathbf{x}$$

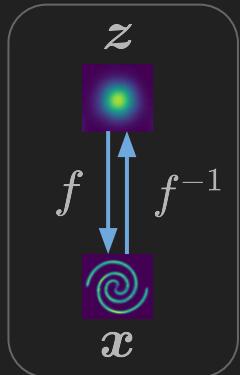
$$\mathbf{z} = f_1^{-1} \circ \dots \circ f_T^{-1}(\mathbf{x})$$

Inverse



Outline

1. Framework



2. Coupling Flows, Autoregressive Flows

3. Continuous-time Flows, Residual Flows

4. Discrete Flows, Surjective Flows

Sampling:

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} = \underline{f(\mathbf{z})}$$

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

$$\mathbf{z} = \underline{f^{-1}(\mathbf{x})}$$

How to Build Efficient Flows?

All about developing layers that:

- 1) Are **expressive**
- 2) Are **invertible**
- 3) Have **cheap-to-compute** Jacobian determinants

Det. Identities:

Planar, radial, Sylvester, etc.

Autoregressive:

MAF, IAF, NAF, TAN, etc.

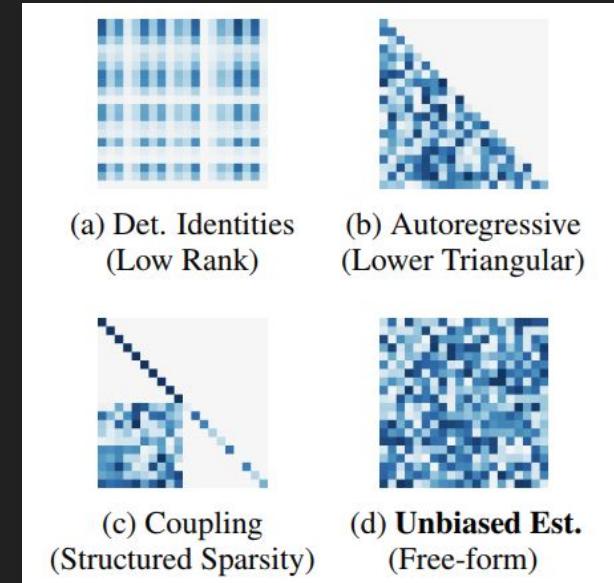
Coupling:

NICE, RealNVP, Glow, Flow++, etc.

Popular!

Unbiased:

FFJORD, Residual Flows, etc.



Taken from (Chen et al., 2019)

Coupling Flows

Forward:

$$\mathbf{x}_{1:d} = \mathbf{z}_{1:d}$$

$$\mathbf{x}_{d+1:D} = (\mathbf{z}_{d+1:D} - \mu_{d+1:D}) \cdot e^{-\alpha_{d+1:D}}$$

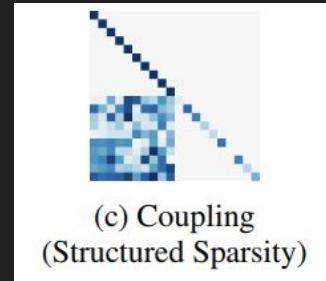
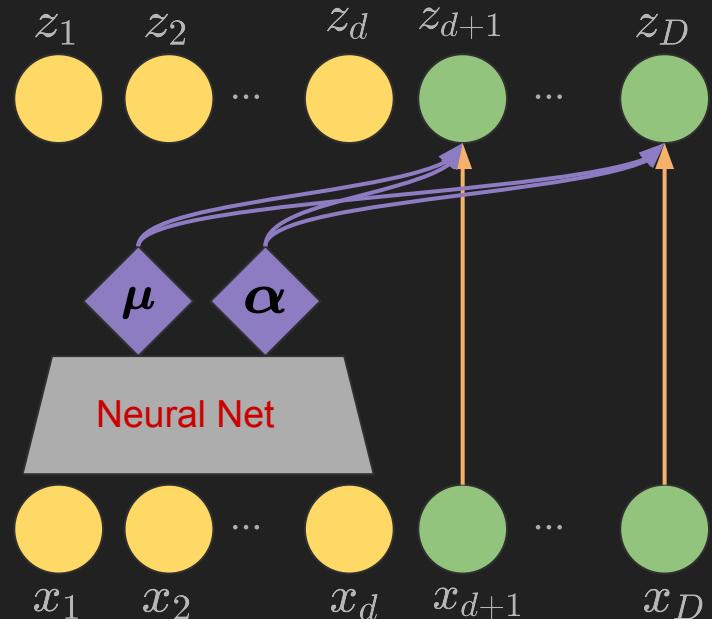
Inverse:

$$\mathbf{z}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{z}_{d+1:D} = \mathbf{x}_{d+1:D} \cdot e^{\alpha_{d+1:D}} + \mu_{d+1:D}$$

Jac. Det:

$$\log |\det J| = \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right| = \sum_{i=d+1}^D \alpha_i$$



Are Coupling Layers Enough?

Problem?

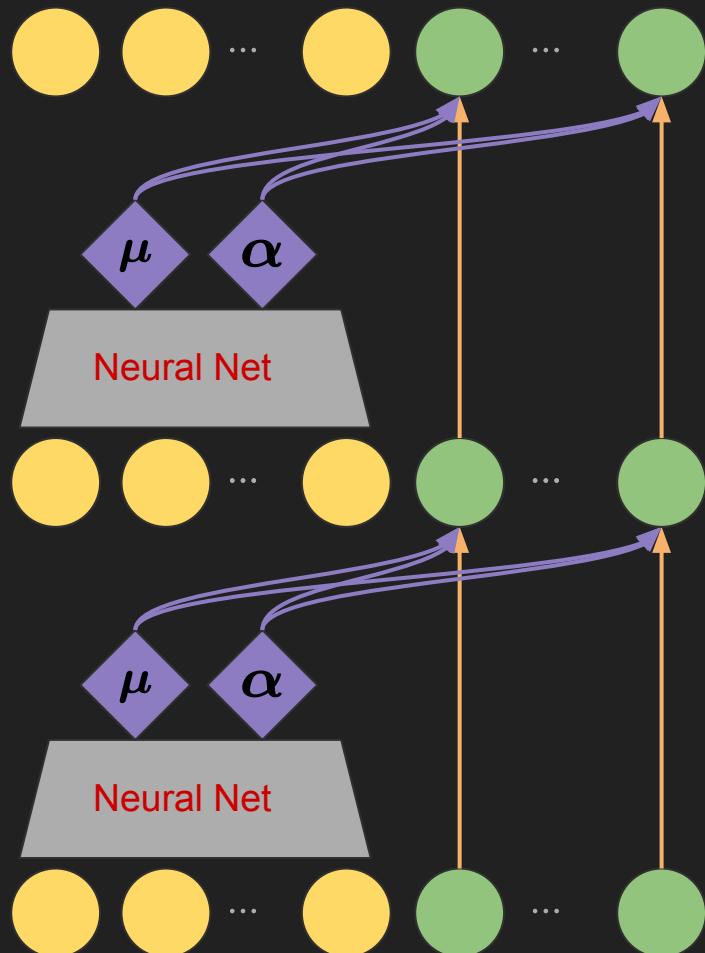
No mixing, need permutation layers!

Reverse:

$$x = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} z$$

Shuffle:

$$x = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} z$$



Time to Code!

Tasks

We will implement a coupling flow like RealNVP/Glow and fit it to 2D toy data.

Todo:

1. Implement the `Coupling` bijection class
2. Implement the `make_net` function
3. Train flow and sample
4. **Bonus:** Study the effect of some parameter (e.g. flow layers, NN layers, NN dim, NN activation, etc.) and improve performance.

Useful functions:

Problem 1: `torch.chunk`, `torch.cat`
Problem 2: `torch.nn.Sequential`

Forward:

$$\begin{aligned}\mathbf{x}_{1:d} &= \mathbf{z}_{1:d} \\ \mathbf{x}_{d+1:D} &= (\mathbf{z}_{d+1:D} - \mu_{d+1:D}) \cdot e^{-\alpha_{d+1:D}}\end{aligned}$$

Inverse:

$$\begin{aligned}\mathbf{z}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{z}_{d+1:D} &= \mathbf{x}_{d+1:D} \cdot e^{\alpha_{d+1:D}} + \mu_{d+1:D}\end{aligned}$$

Jac. Det:

$$\log |\det J| = \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right| = \sum_{i=d+1}^D \alpha_i$$

`.inverse(self, z)`

`⋮`

`return x
(B, 2)`

`.forward(self, x)`

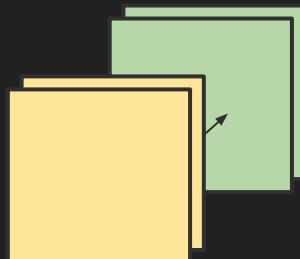
`⋮
return z, ldj
(B, 2) (B,)`

(Taken from Kingma & Dhariwal 2018)

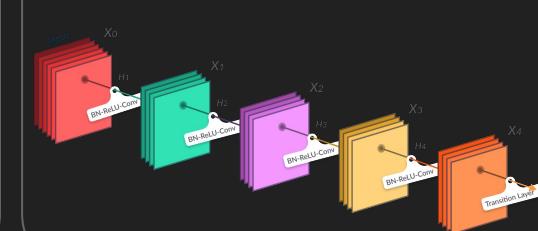
Coupling Flows on Images

While toy data had shape (B,2),
e.g. CIFAR-10 has shape (B,3,32,32)

Split along channels:



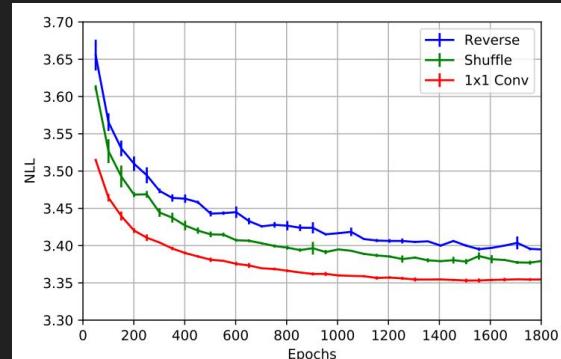
Use e.g. CNNs:



Need dequantization
since images are
discrete in $\{0, 1, \dots, 255\}$



Permutation using Conv1x1:



How to Build Efficient Flows?

All about developing layers that:

- 1) Are **expressive**
- 2) Are **invertible**
- 3) Have **cheap-to-compute** Jacobian determinants

Det. Identities:

Planar, radial, Sylvester, etc.

Autoregressive:

MAF, IAF, NAF, TAN, etc.

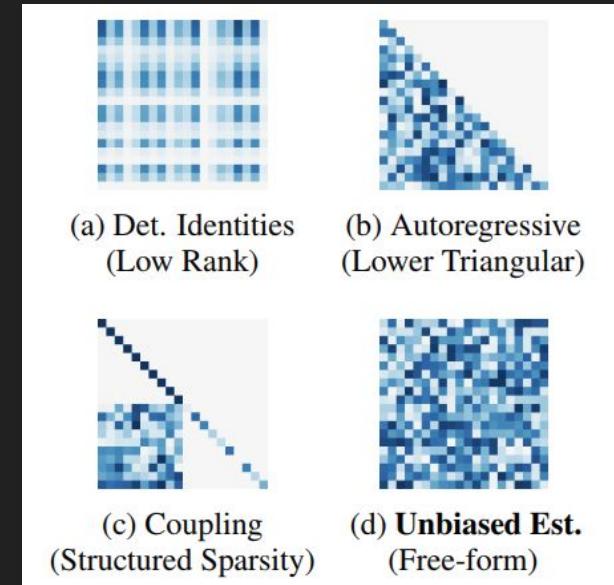
Related to
AR models

Coupling:

NICE, RealNVP, Glow, Flow++, etc.

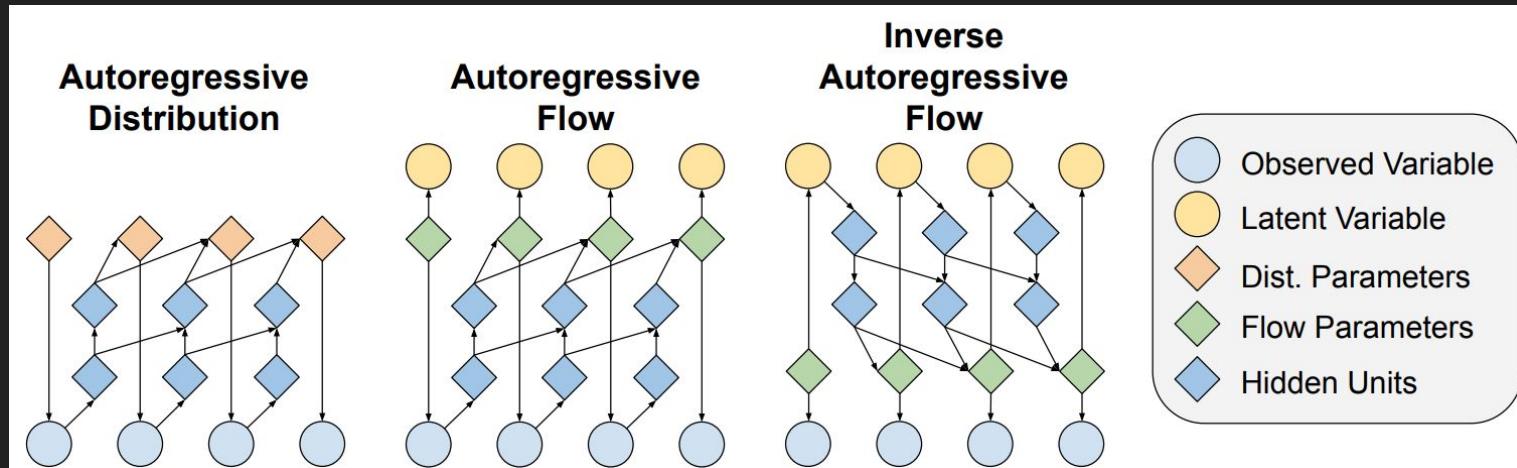
Unbiased:

FFJORD, Residual Flows, etc.



Taken from (Chen et al., 2019)

Autoregressive Flows



Equivalent for continuous distributions
(Papamakarios et al. 2017)

Sampling: D passes (slow)
Density: 1 pass (fast)

Inverse possible for flows
(Kingma et al. 2016)

Sampling: 1 pass (fast)
Density: D passes (slow)

(for comparison)
Coupling flows:

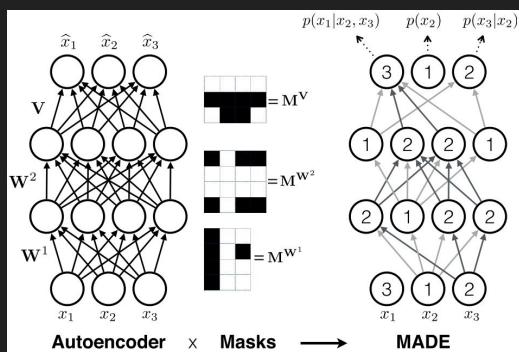
Sampling: 1 pass (fast)
Density: 1 pass (fast)

RNNs (e.g. LSTM, GRU)
require sequential computation

Masked Autoregressive NNs

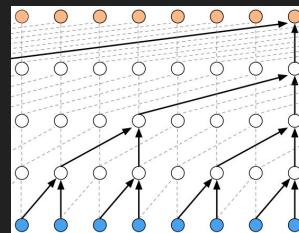
Vectors (B,D)

MADE
(Larochelle et al. 2015)



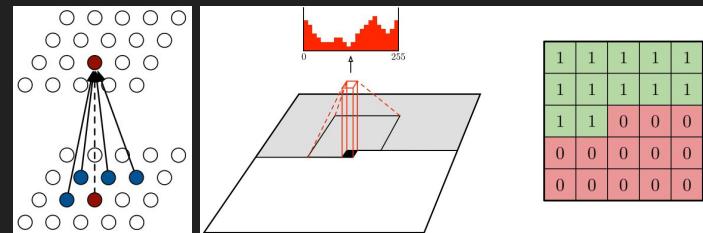
Sequences (B,D,L)

WaveNet
(van den Oord et al. 2016)



Images (B,C,H,W)

PixelCNN
(van den Oord et al. 2016)



...

Transformer
(Vaswani et al. 2017)

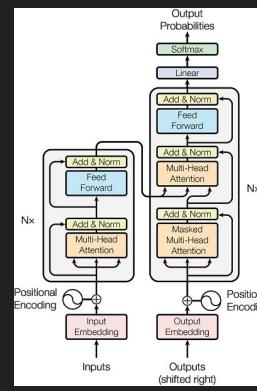
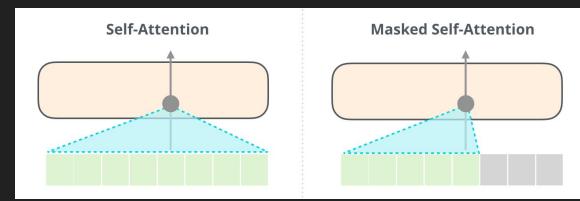
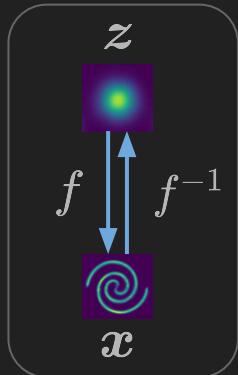


Image Transformer
(Parmar et al. 2018)



Outline

1. Framework



Sampling:

$$\mathbf{z} \sim p(\mathbf{z})$$

$$\mathbf{x} = \underline{f(\mathbf{z})}$$

Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$

$$\mathbf{z} = \underline{f^{-1}(\mathbf{x})}$$

2. Coupling Flows, Autoregressive Flows



Sampling: 1 pass (fast)
Density: 1 pass (fast)



Sampling: D passes (slow)
Density: 1 pass (fast)

Sampling: 1 pass (fast)
Density: D passes (slow)

3. Continuous-time Flows, Residual Flows

4. Discrete Flows, Surjective Flows

Continuous-time Flows

Neural ODE (Chen et al. 2018):

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

Instantaneous Change-of-Variables formula:

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\text{Tr} \left(\frac{\partial f}{\partial \mathbf{z}(t)} \right)$$

FFJORD (Grathwohl et al. 2018):

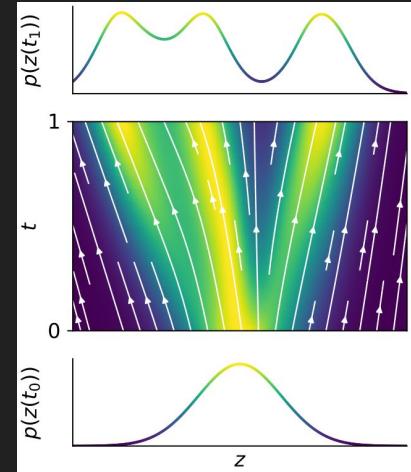
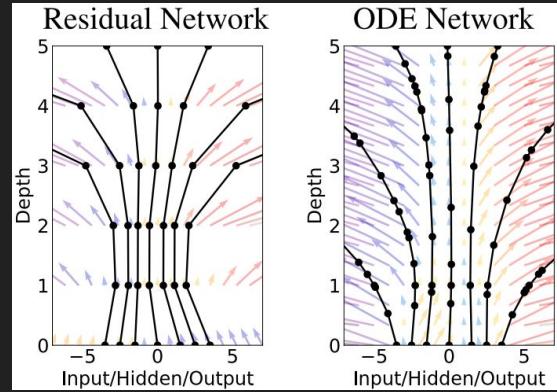
Use Hutchinson's Trace Estimator:

$$\text{Tr} \left(\frac{\partial f}{\partial \mathbf{z}(t)} \right) = \mathbb{E}_{p(\boldsymbol{\epsilon})} \left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)} \boldsymbol{\epsilon} \right] \quad \text{for} \quad \mathbb{E} [\boldsymbol{\epsilon}] = \mathbf{0}, \text{Cov} [\boldsymbol{\epsilon}] = \mathbf{I}$$

Jacobian $\frac{\partial f}{\partial \mathbf{z}(t)}$ costs $O(D^2)$

vjp $\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}(t)}$ costs $O(D)$

Connects to score-based / diffusion models!
(Song et al. 2020)



Residual Flows

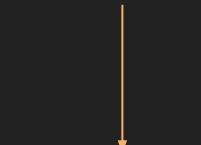
i-ResNets (Behrmann et al. 2019):

$$\mathbf{z} = f(\mathbf{x}) = \mathbf{x} + h(\mathbf{x}) \quad \text{for contractive } h$$

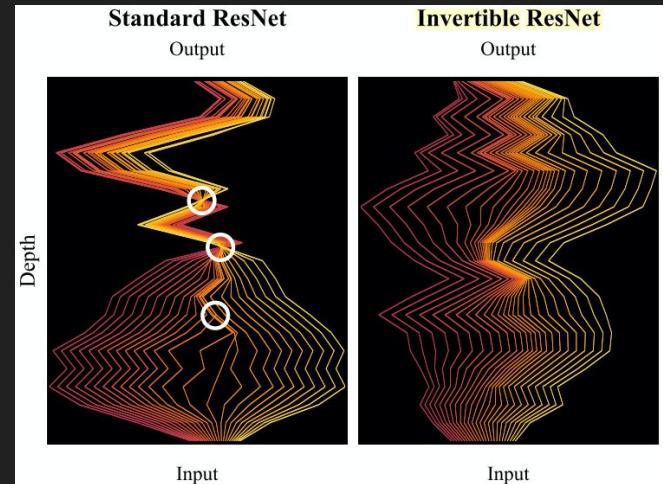
$$\log J_f = \text{Tr} \left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \mathbf{J}_h^k \right)$$

Hutchinson's
Trace Estimator

Truncate
(introduces bias)



Enforce using
SpectralNorm

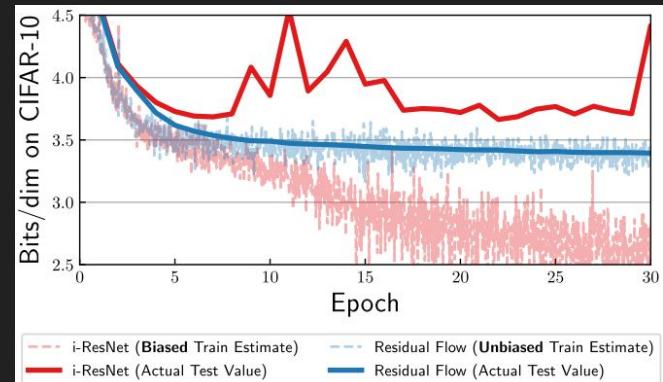


Residual Flows (Chen et al. 2019):

Replace truncation by (unbiased) Russian Roulette Estimator:

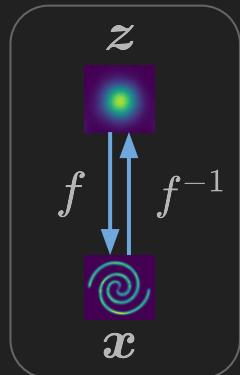
$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{p(N)} \left[\sum_{k=1}^N \frac{\Delta_k}{P(N \geq k)} \right] \quad \text{for } p(N) \text{ w/ integer support}$$

+ introduce memory-efficient gradient estimators



Outline

1. Framework



Sampling:

$$z \sim p(z)$$

$$x = \underline{f(z)}$$

Density:

$$\log p(x) = \log p(z) + \log \left| \det \frac{\partial z}{\partial x} \right|$$

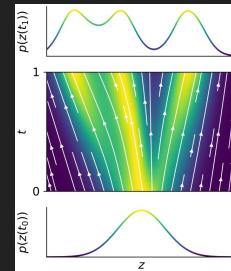
$$z = \underline{f^{-1}(x)}$$

2. Coupling Flows, Autoregressive Flows



Sampling: 1 pass (fast)
Density: D passes (slow)

3. Continuous-time Flows, Residual Flows



4. Discrete Flows, Surjective Flows

Discrete Flows

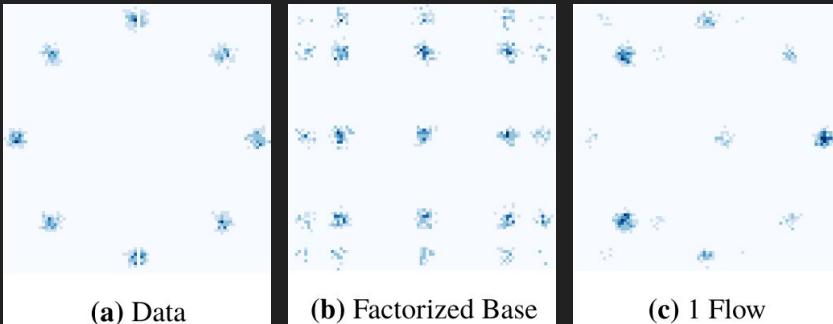
No volume: $\log p(\mathbf{x}) = \log p(\mathbf{z})$

Discrete Flows (Tran et al. 2019):

$$z_d = (\mu_d + \sigma_d x_d) \bmod K$$

Use Straight-Through Estimator:

Forward: $\mu_d = \text{one_hot}(\text{argmax}(\theta_d))$
Backward: $\text{softmax}(\theta_d/\tau)$

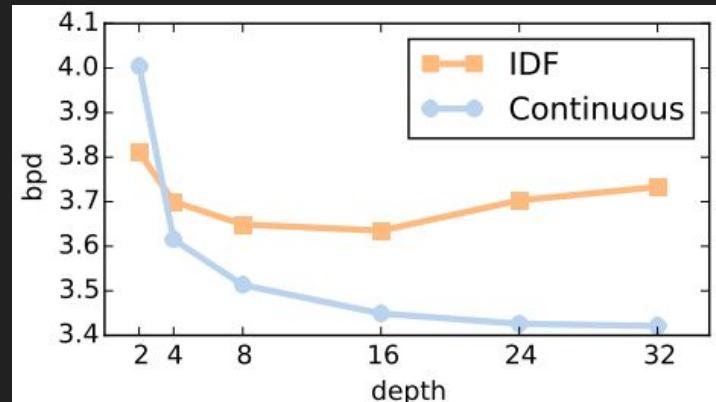


Integer Discrete Flows (Hoogeboom et al. 2019):

$$z_d = x_d + \mu_d$$

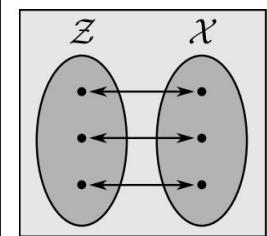
Also use Straight-Through Estimator:

Forward: $\mu_d = \lfloor \theta_d \rfloor$
Backward: θ_d

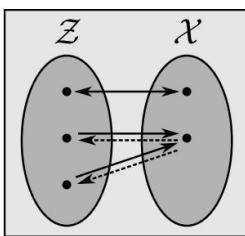


Surjective Flows

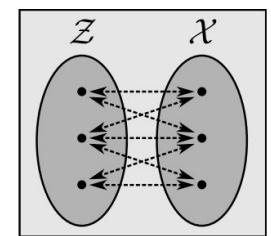
SurVAE Flows (Nielsen et al. 2020):



Bijective



Surjective



Stochastic

Examples:

$x = \text{round}(z)$

Dequantization (Uria et al., 2013; Ho et al. 2019)

$x = z[:n]$

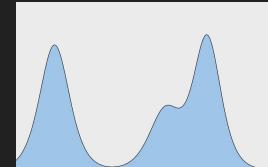
Augmented Normalizing Flows (Huang et al., 2020), VFlow (Chen et al., 2020)

$x = \text{argmax}(z)$

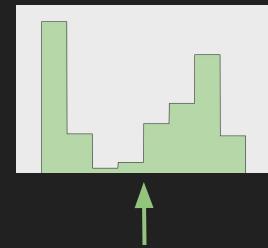
Argmax Flows (Hoogeboom et al., 2021)

⋮

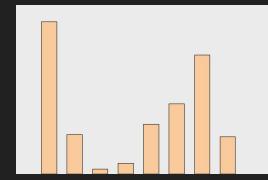
Flow



Dequantized Data



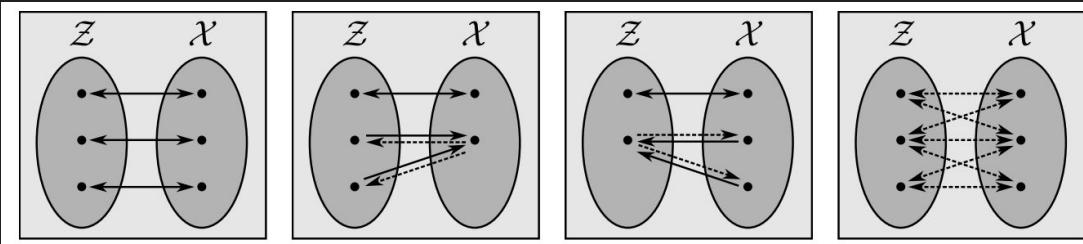
Data



heartedness frege thematically inferred by the famous existence of a function f from the laplace definition we can analyze a definition of binary operations with additional size so their functionality cannot be viewed here there is no change because its total cost of learning objects from language to platonic linguistics examines why animate to indicate wild amphibious substances animal and marine life constituents of animals and bird sciences medieval biology biology and central medicine full discovery re

Surjective Flows

SurVAE Flows (Nielsen et al. 2020):



Bijective

Surjective

Stochastic

Transformation	Forward $x \leftarrow z$	Inverse $z \leftarrow x$	Likelihood Contribution $\mathcal{V}(x, z)$
Bijective	$x = f(z)$	$z = f^{-1}(x)$	$\log \det \nabla_x z $
Stochastic	$x \sim p(x z)$	$z \sim q(z x)$	$\log \frac{p(x z)}{q(z x)}$
Surjective (Gen.)	$x = f(z)$	$z \sim q(z x)$	$\log \frac{p(x z)}{q(z x)}$ as $\frac{p(x z)}{\delta(x - f(z))} \rightarrow$
Surjective (Inf.)	$x \sim p(x z)$	$z = f^{-1}(x)$	$\log \frac{p(x z)}{q(z x)}$ as $\frac{q(z x)}{\delta(z - f^{-1}(x))} \rightarrow$

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|, \quad \mathbf{z} = f^{-1}(\mathbf{x})$$

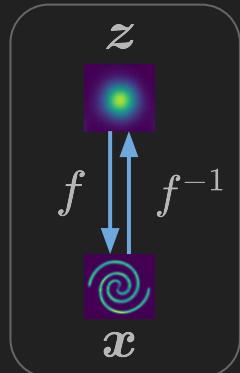
$$\log p(\mathbf{x}) \geq \log p(\mathbf{z}) + \log \frac{p(\mathbf{x}|\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}, \quad \mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$$



**Likelihood
contribution**

Outline

1. Framework



Density:

$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \log \left| \det \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right|$$
$$\mathbf{z} = \underline{f^{-1}(\mathbf{x})}$$

2. Coupling Flows, Autoregressive Flows



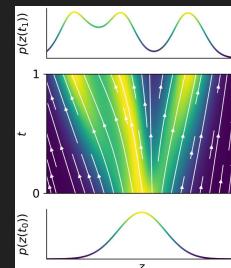
Sampling: 1 pass (**fast**)
Density: 1 pass (**fast**)



Sampling: D passes (**slow**)
Density: 1 pass (**fast**)

Sampling: 1 pass (**fast**)
Density: D passes (**slow**)

3. Continuous-time Flows, Residual Flows



ODE



Invertible
ResNet

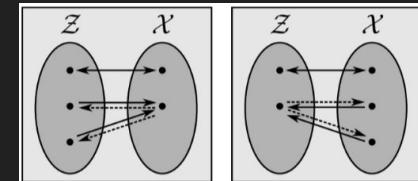
4. Discrete Flows, Surjective Flows

Discrete Flows:

$$\log p(\mathbf{x}) = \log p(\mathbf{z})$$
$$\mathbf{z} = f^{-1}(\mathbf{x})$$

Straight-Through Estimator

Surjective Flows:



$$\log p(\mathbf{x}) = \log p(\mathbf{z}) + \mathcal{V}(\mathbf{x}, \mathbf{z})$$

Thanks for listening!