Normalizing Flows

Didrik Nielsen

Where are flows useful?

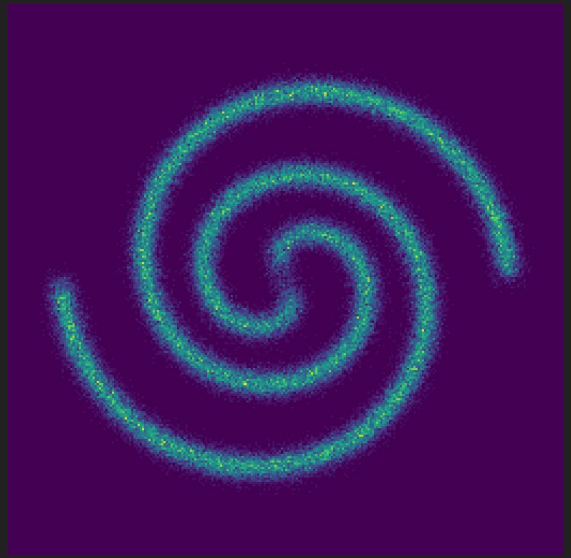
Anywhere you need a flexible density $p(\boldsymbol{x})$ with

Sampling:

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

Density:

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



Why Flows?

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

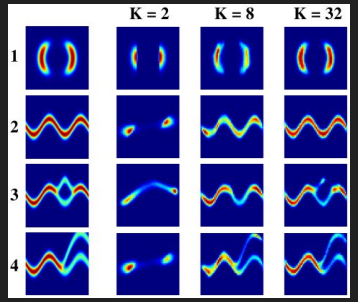
History

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 DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

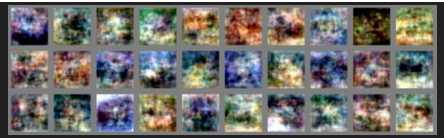


Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	≤ 89.9
DLGM+NF (k = 10)	≤ 87.5
DLGM+NF (k = 20)	≤ 86.5
DLGM+NF (k = 40)	≤ 85.7
DLGM+NF (k = 80)	≤ 85.1

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 **Samy Bengio**
Google Brain Google Brain

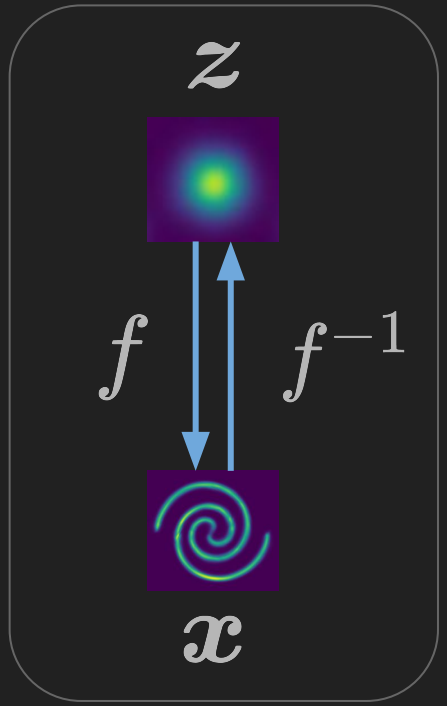
Outline

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

The Flow Framework

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

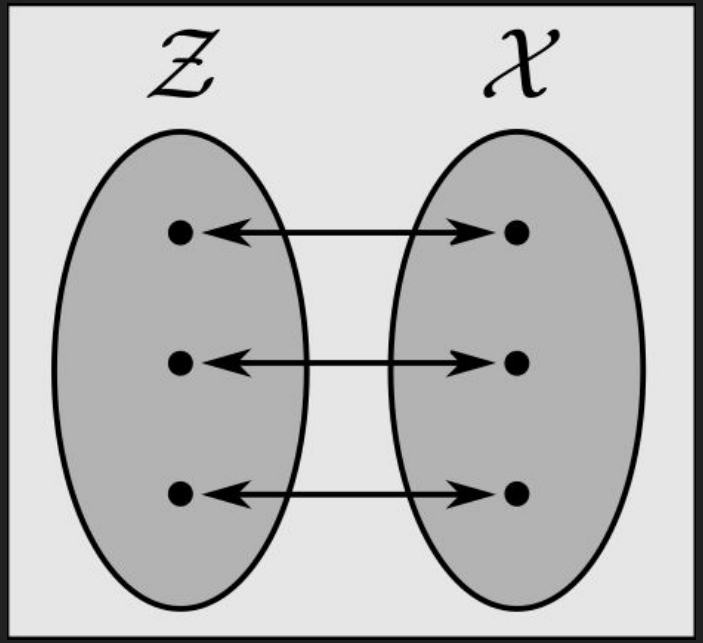
- Base distribution $p(\boldsymbol{z})$
- Bijective mapping f



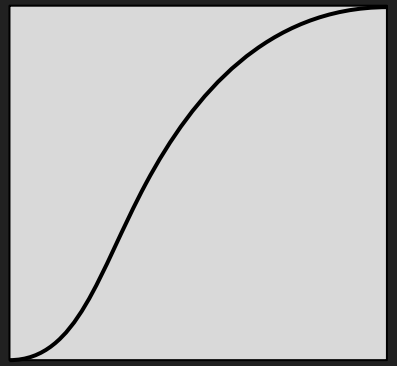
Sampling:

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

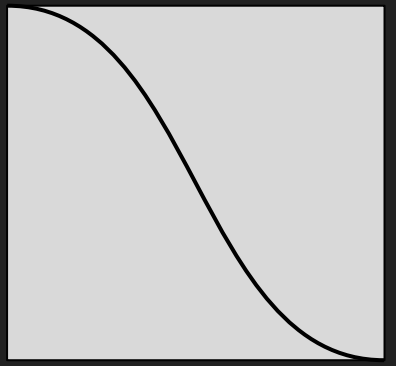
Flows: Invertibility



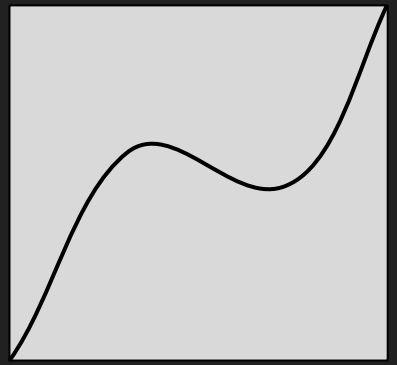
In 1D:
Strictly increasing or decreasing



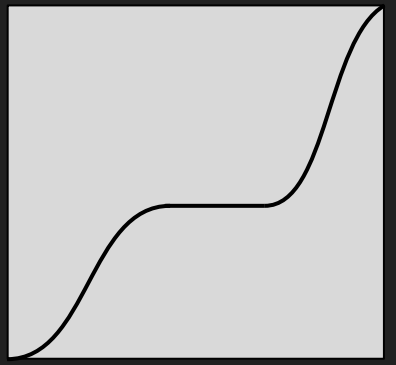
Invertible



Invertible



Not Invertible

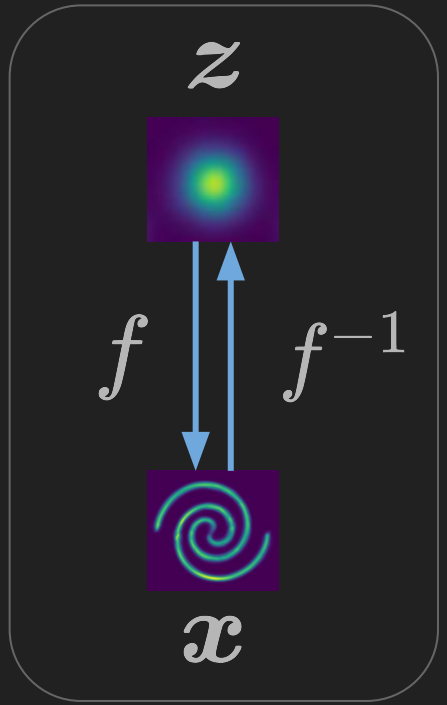


Not Invertible

The Flow Framework

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

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



Sampling:

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

Density:

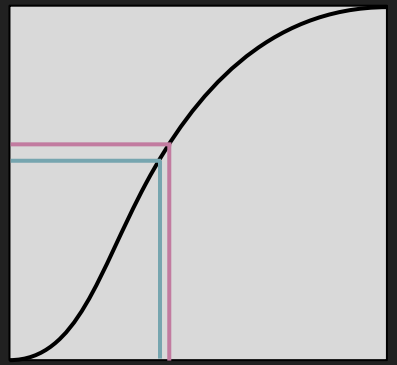
$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \log \underbrace{\left|\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right|}_{\text{Jac. Det.}}$$
$$\boldsymbol{z} = f^{-1}(\boldsymbol{x})$$

Change-of-Variables in 1D

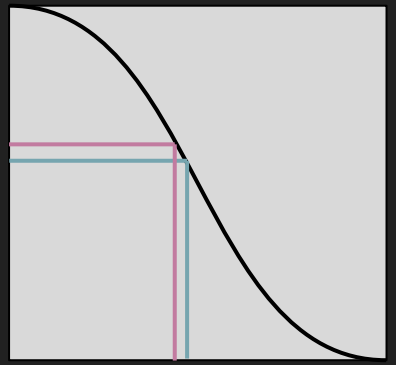
$$\int_{x_1}^{x_2} p(x)dx = \int_{z_1}^{z_2} p(z)dz$$

$$= \int_{?}^{?} p(z)\frac{dz}{dx}dx$$

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



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



Let's derive it!

$$\int_{x_1}^{x_2} p(x)dx = \int_{z_1}^{z_2} p(z)dz = \int_{x_1}^{x_2} p(z)\frac{dz}{dx}dx$$

$$\int_{x_1}^{x_2} p(x)dx = \int_{z_1}^{z_2} p(z)dz = \int_{x_2}^{x_1} p(z)\frac{dz}{dx}dx = \int_{x_1}^{x_2} p(z)\left[-\frac{dz}{dx}\right]dx$$

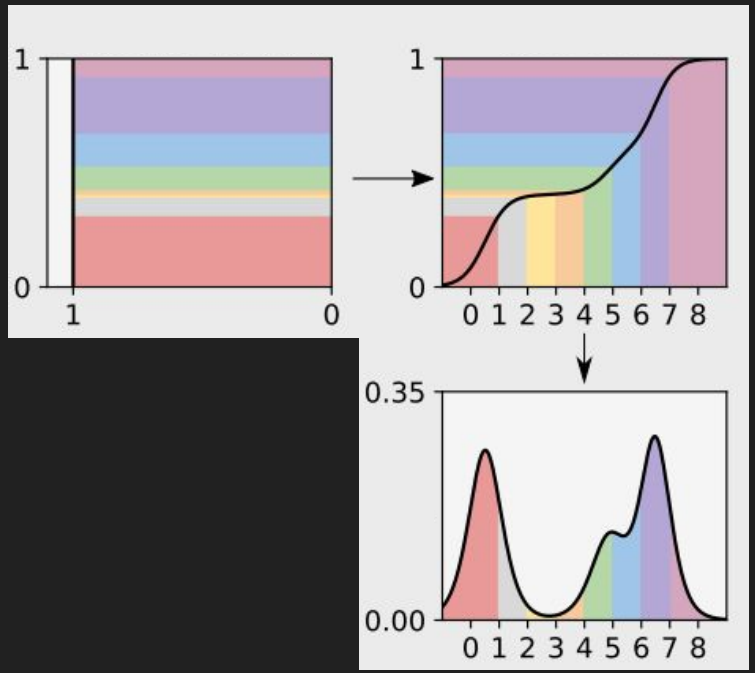
$$p(x) = p(z)\left|\frac{dz}{dx}\right|$$

Flows: 1D Example

Inverse transform sampling:

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

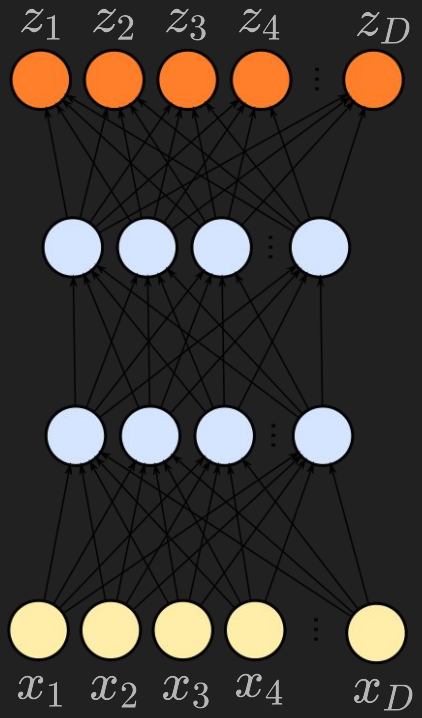
$$= \left|\frac{dCDF(x)}{dx}\right|$$



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

Change-of-Variables

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

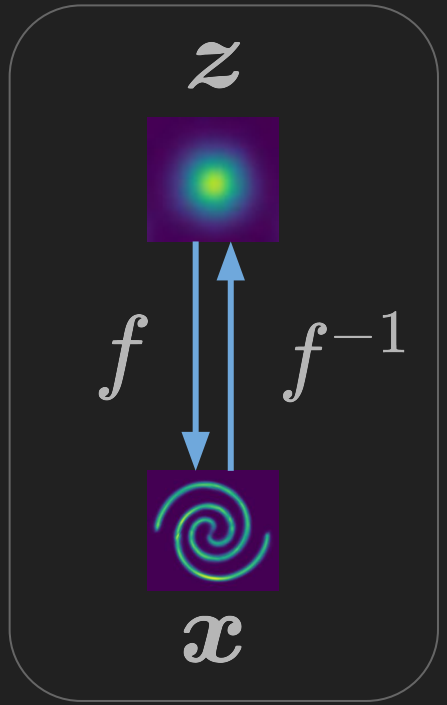


$$p(x) = p(z) \left| \det \boxed{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}} \right|$$

The Flow Framework

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

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



Sampling:

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$
$$\boldsymbol{x} = f(\boldsymbol{z})$$

Density:

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \log \underbrace{\left|\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right|}_{\text{Jac. Det.}}$$
$$\boldsymbol{z} = f^{-1}(\boldsymbol{x})$$

The Flow Framework

Sampling:

$$z \sim p(z)$$

$$x = f(z)$$

Forward

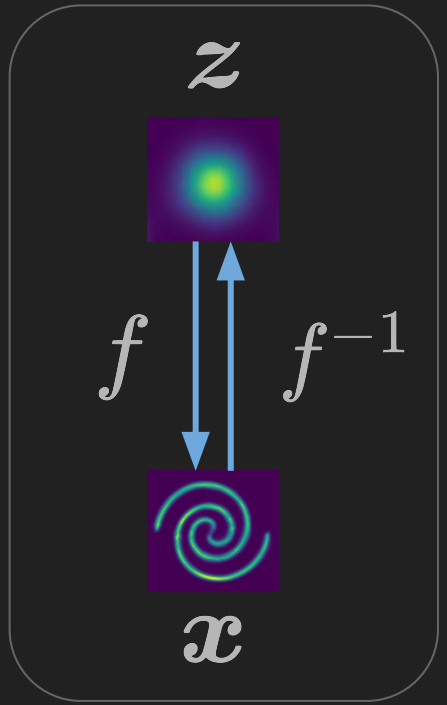
Density:

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

Jac. Det.

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

Inverse



The Flow Framework

Sampling:

$$z \sim p(z)$$

$$x = f_T \circ \ldots \circ f_1(z)$$

Forward

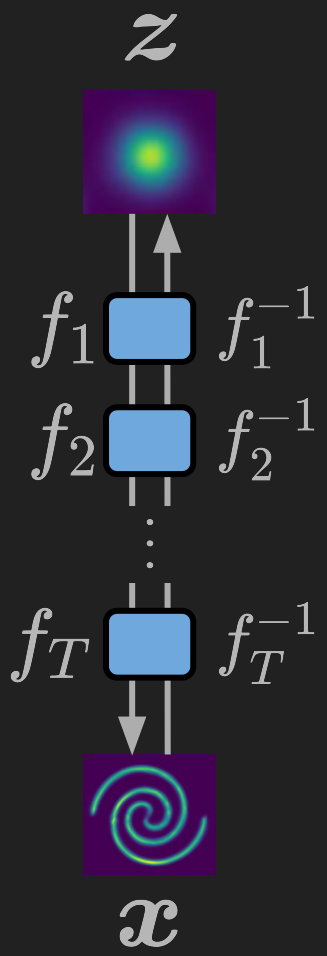
Density:

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

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

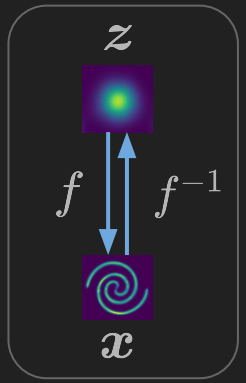
Jac. Det.

Inverse



Outline

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



Sampling:

$$z \sim p(z)$$
$$x = f(z)$$

Density:

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

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.

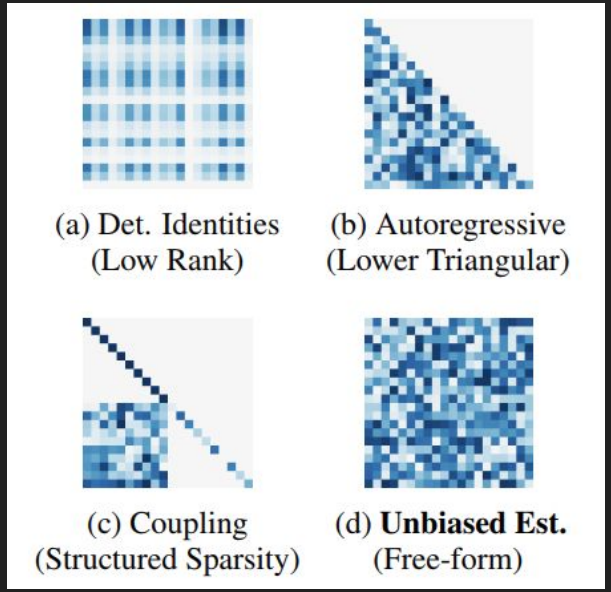
Coupling:

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

Popular!

Unbiased:

FFJORD, Residual Flows, etc.



Taken from (Chen et al., 2019)

Coupling Flows

Forward:

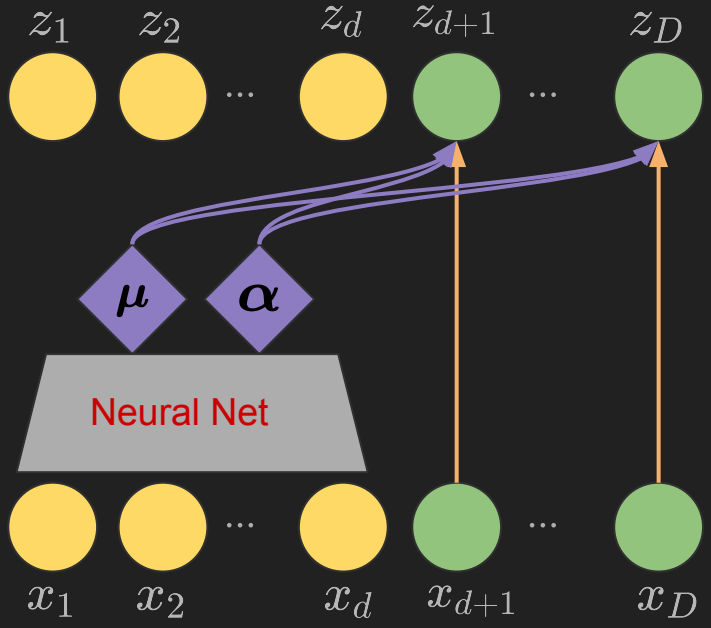
$$\boldsymbol{x}_{1:d} = \boldsymbol{z}_{1:d}$$
$$\boldsymbol{x}_{d+1:D} = \left(\boldsymbol{z}_{d+1:D} - \mu_{d+1:D}\right) \cdot e^{-\alpha_{d+1:D}}$$

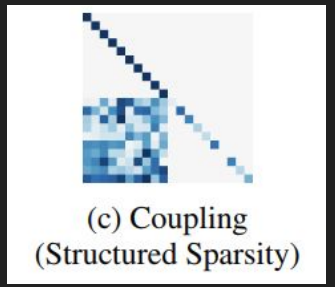
Inverse:

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

Jac. Det:

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





(c) Coupling
(Structured Sparsity)

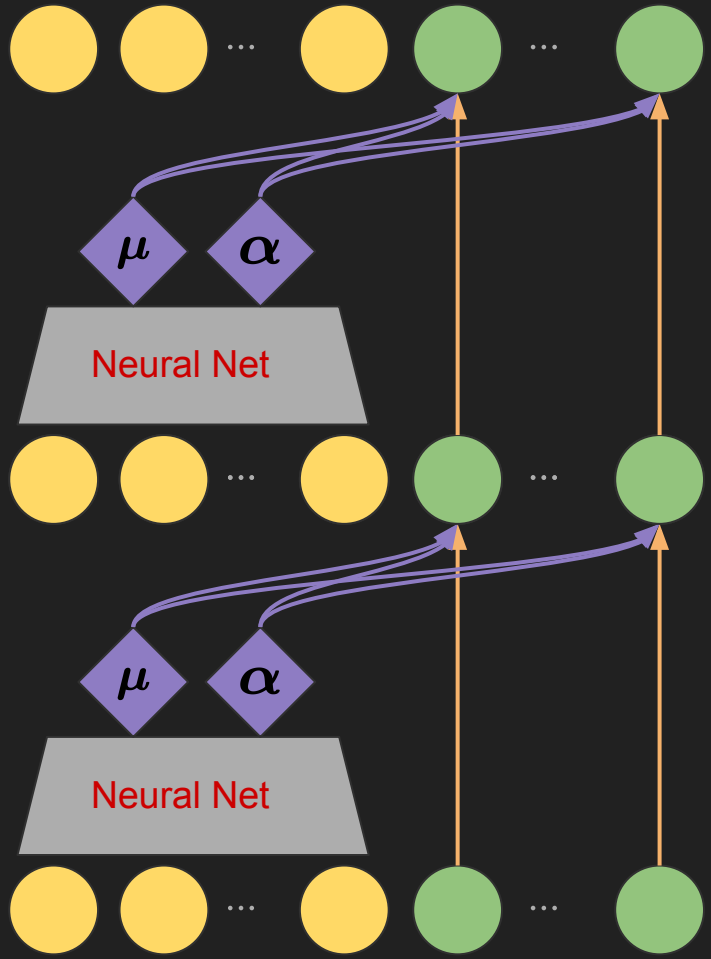
Are Coupling Layers Enough?

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:

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

Inverse:

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

Jac. Det:

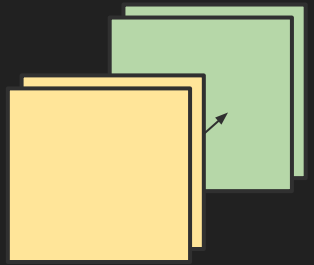
$$\log\left|\det J\right| = \log\left|\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}\right| = \sum_{i=d+1}^{D} \alpha_i$$

```
.inverse(self, z)
    ⋮
    return x
           (B, 2)
.forward(self, x)
    ⋮
    return z,    ldj
           (B, 2) (B,)
```

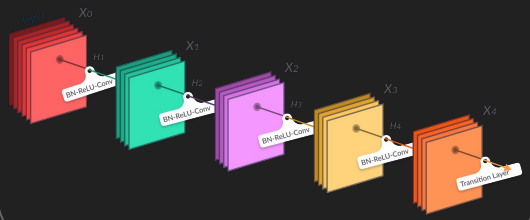
Coupling Flows on Images

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

Split along channels:



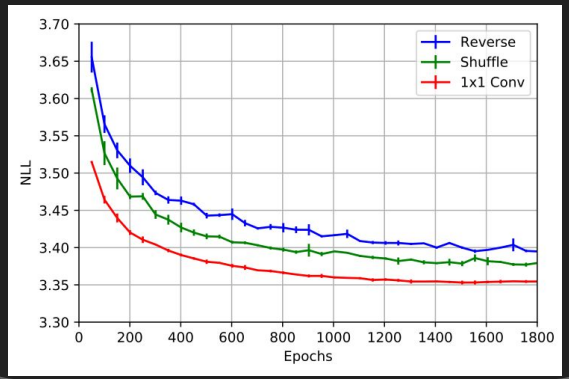
Use e.g. CNNs:



Need dequantization since images are discrete in {0,1,...,255}

(Taken from Kingma & Dhariwal 2018)

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.

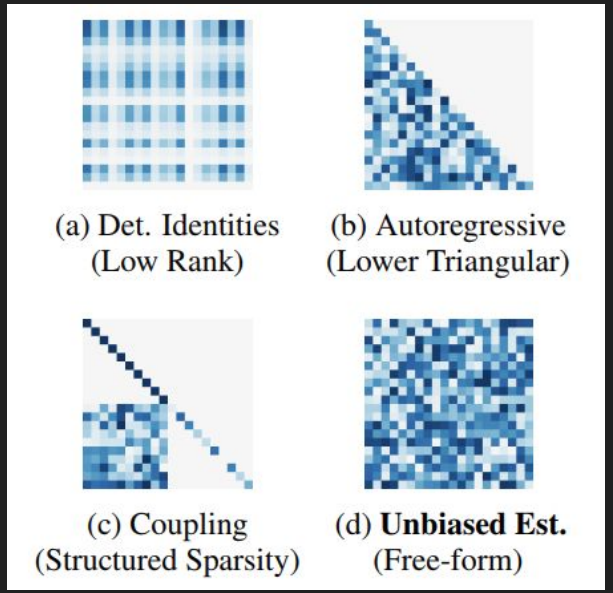
Autoregressive:
MAF, IAF, NAF, TAN, etc.

Related to AR models

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

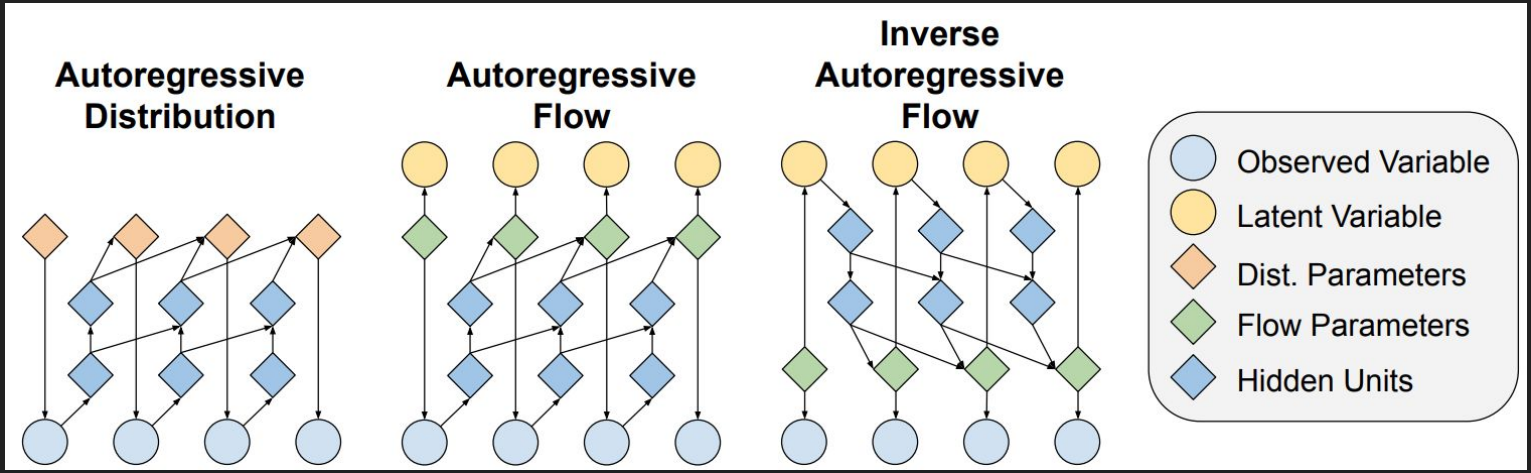
Popular!

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:

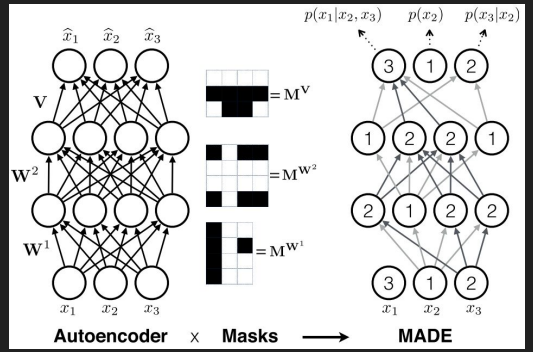
Sampling: 1 pass (fast)
Density: 1 pass (fast)

Masked Autoregressive NNs

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

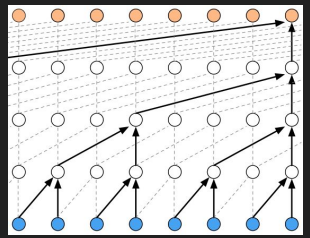
Vectors (B,D)

MADE
(Larochelle et al. 2015)

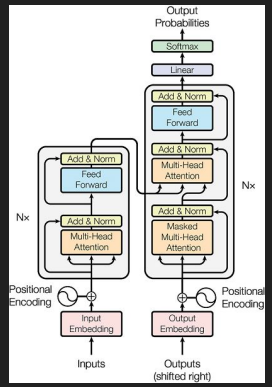


Sequences (B,D,L)

WaveNet
(van den Oord et al. 2016)



Transformer
(Vaswani et al. 2017)



Images (B,C,H,W)

PixelCNN
(van den Oord et al. 2016)

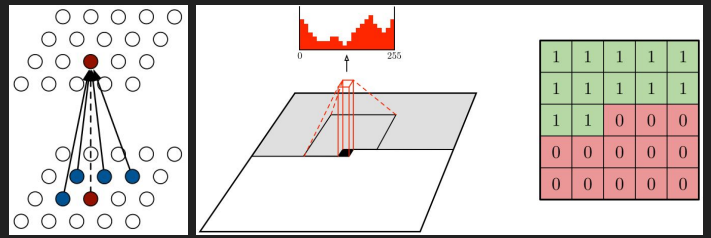
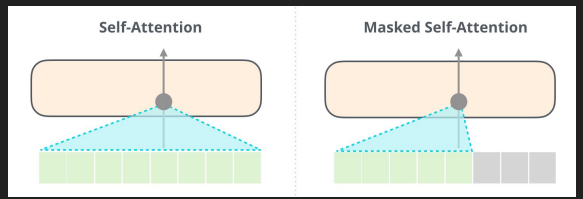


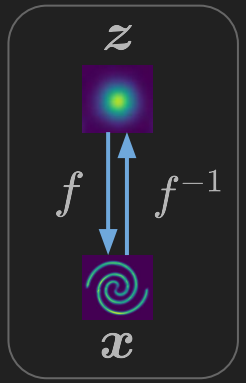
Image Transformer
(Parmar et al. 2018)



…

Outline

1. Framework



Sampling:

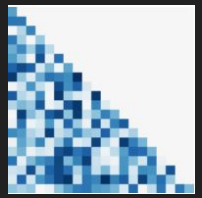
$$z \sim p(z)$$
$$x = f(z)$$

Density:

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

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

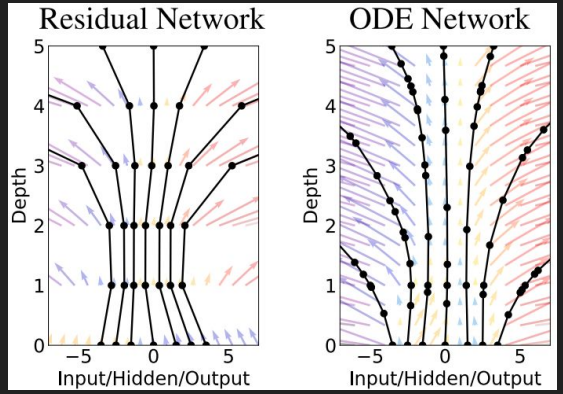
Continuous-time Flows

Neural ODE (Chen et al. 2018):

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

Instantaneous Change-of-Variables formula:

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



FFJORD (Grathwohl et al. 2018):

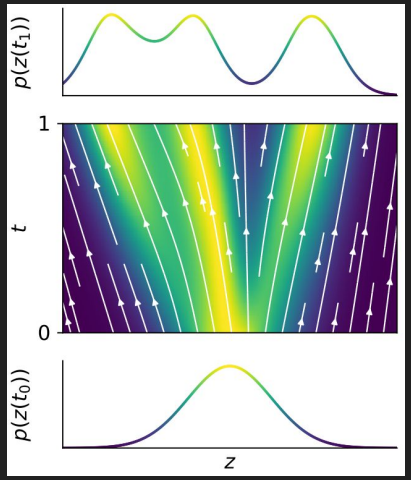
Use Hutchinson's Trace Estimator:

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

Jacobian $\frac{\partial f}{\partial \boldsymbol{z}(t)}$ costs O(D²)

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

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



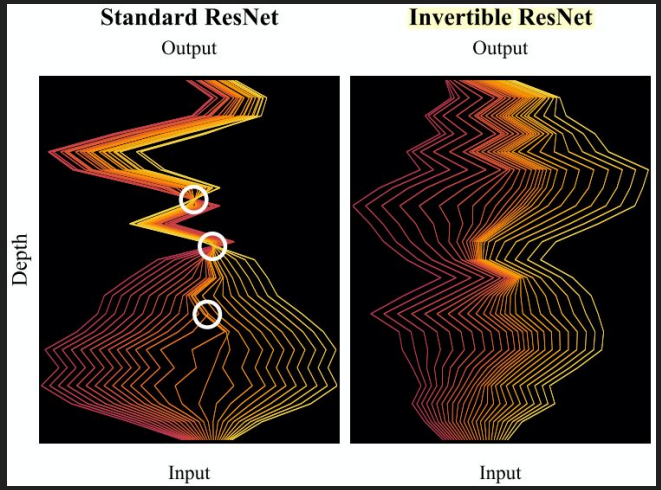
Residual Flows

i-ResNets (Behrmann et al. 2019):

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

$$\log J_f = \mathrm{Tr}\left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} J_h^k\right)$$

- Hutchinson's Trace Estimator
- Truncate (introduces bias)
- Enforce using SpectralNorm

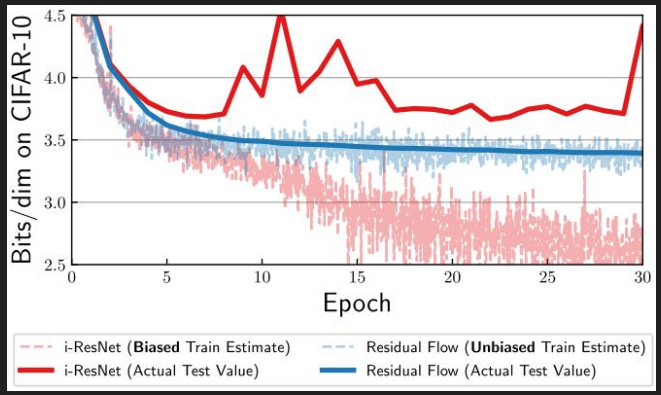


Residual Flows (Chen et al. 2019):

Replace truncation by (unbiased) Russian Roulette Estimator:

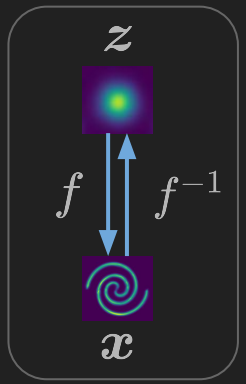
$$\sum_{k=1}^{\infty} \Delta_k = \mathbb{E}_{p(N)}\left[\sum_{k=1}^{N} \frac{\Delta_k}{P(N \geq k)}\right] \quad \text{for } p(N) \text{ w/ integer support}$$

+ introduce memory-efficient gradient estimators



Outline

1. Framework



Sampling:

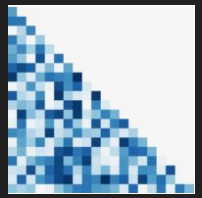
$$z \sim p(z)$$
$$x = f(z)$$

Density:

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

2. Coupling Flows, Autoregressive Flows

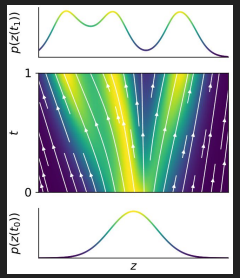
Sampling: 1 pass (fast)
Density: 1 pass (fast)



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

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

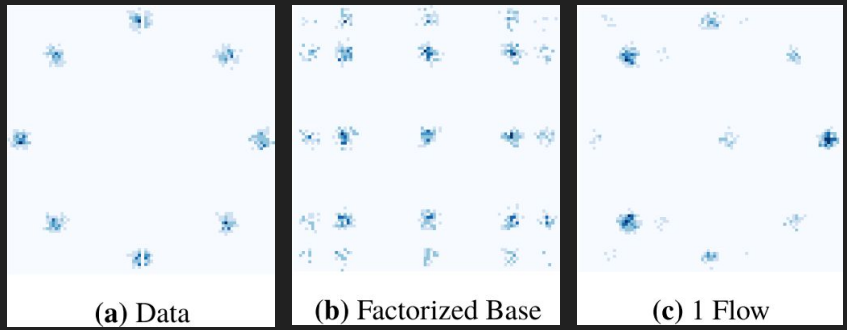
Discrete Flows (Tran et al. 2019):

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

Use Straight-Through Estimator:

Forward: $\mu_d = \text{one\_hot}(\text{argmax}(\theta_d))$

Backward: $\text{softmax}(\theta_d / \tau)$



(a) Data (b) Factorized Base (c) 1 Flow

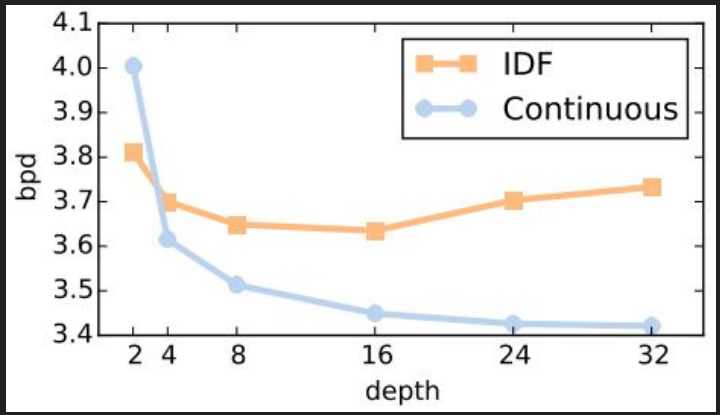
Integer Discrete Flows (Hoogeboom et al. 2019):

$$z_d = x_d + \mu_d$$

Also use Straight-Through Estimator:

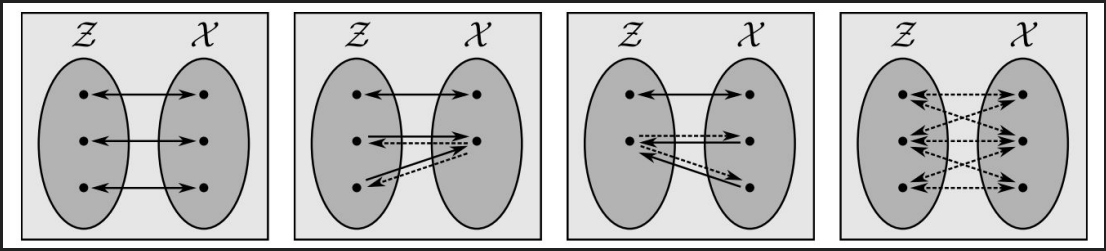
Forward: $\mu_d = \lfloor \theta_d \rceil$

Backward: θ_d

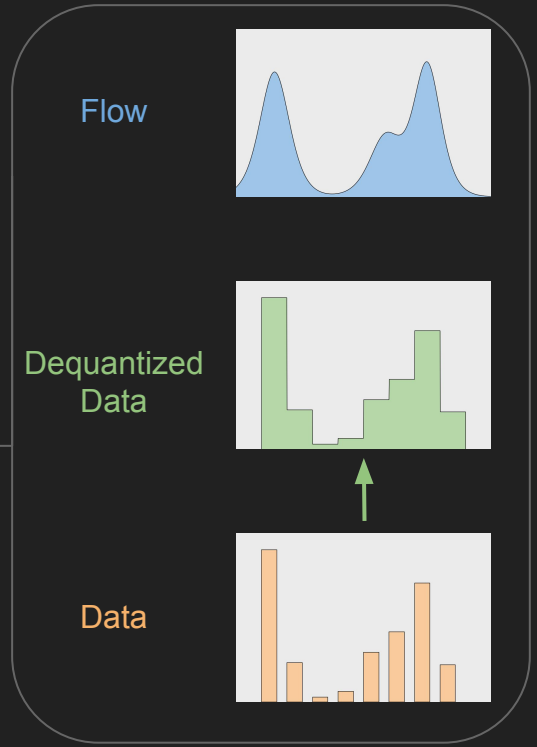


Surjective Flows

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)

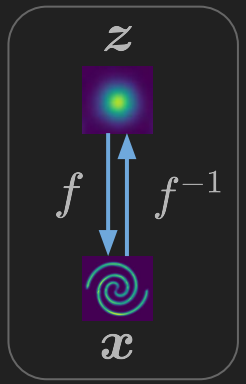
$\vdots$

heartedness frege thematically infered 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 reviewed here there is no change because its

otal 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

Outline

1. Framework



Sampling:

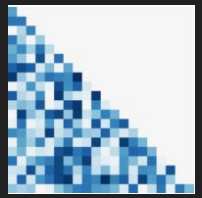
$$z \sim p(z)$$
$$x = f(z)$$

Density:

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

2. Coupling Flows, Autoregressive Flows

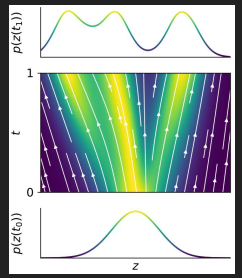
Sampling: 1 pass (fast)
Density: 1 pass (fast)



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

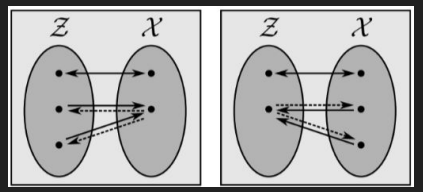
4. Discrete Flows, Surjective Flows

Discrete Flows:

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

Straight-Through Estimator

Surjective Flows:



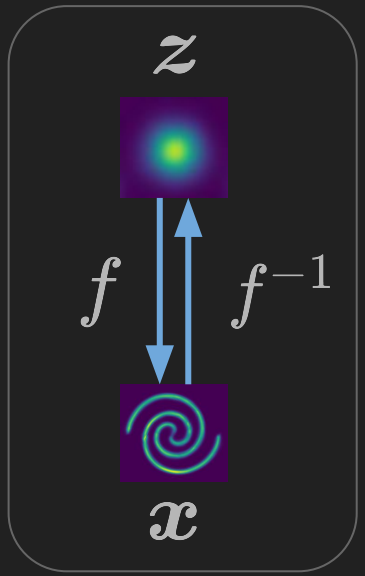
$$\log p(x) = \log p(z) + \mathcal{V}(x, z)$$

Bonus: Variational Inference with Flows

Generative Modeling

Train to learn
data distribution $q(\boldsymbol{x})$

$p(\boldsymbol{x})$



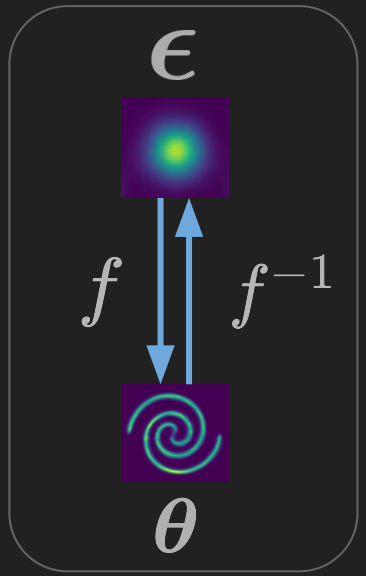
Minimize:

$$\mathbb{D}_{KL}\left[q(\boldsymbol{x})\|p(\boldsymbol{x})\right]$$

Variational Inference

Train to approximate
posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$

$q(\boldsymbol{\theta})$



Minimize:

$$\mathbb{D}_{KL}\left[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})\right]$$

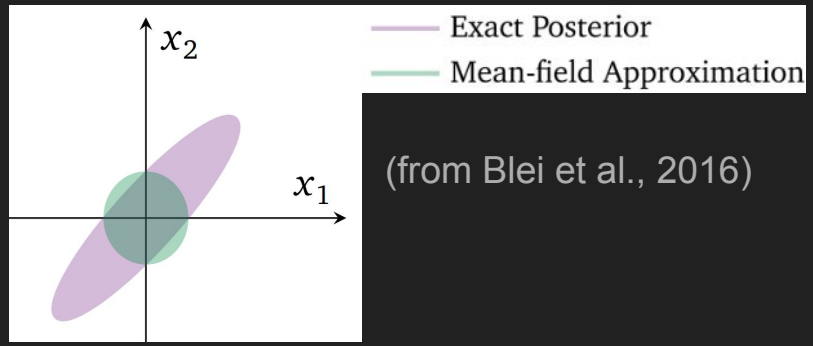
Variational Inference

Learn posterior approximation:

$$q_\lambda(\boldsymbol{\theta}) \approx p(\boldsymbol{\theta}|\mathcal{D})$$

By minimizing the KL:

$$\mathbb{D}_{KL}\left[q_\lambda(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|\mathcal{D})\right]$$



(from Blei et al., 2016)

Example: Mean-field approximation:

Factorized distribution:

$$q_\lambda(\boldsymbol{\theta}) = \prod_{d=1}^{D} q_{\lambda_d}(\theta_d)$$

Common choice:

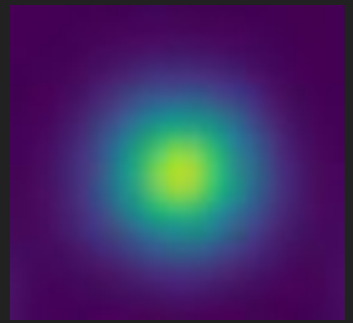
$$q_\lambda(\boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\theta_d|\mu_d, \sigma_d^2)$$

$$\boldsymbol{\lambda} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$$

Variational Inference with Flows

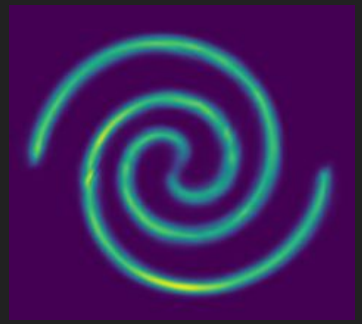
Mean-field Gaussian:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I})$$
$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$
$$q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$$

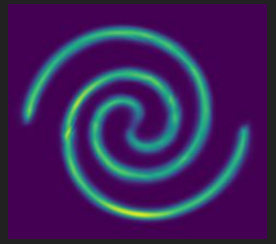


Flow:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{I})$$
$$\boldsymbol{\theta} = f_{\boldsymbol{\lambda}}(\boldsymbol{\epsilon})$$
$$q(\boldsymbol{\theta}) = q(\boldsymbol{\epsilon})\,|\det \boldsymbol{J}|$$



Which Flows to use for VI?



$$\epsilon \sim \mathcal{N}(\epsilon|\mathbf{0}, \boldsymbol{I})$$
$$\boldsymbol{\theta} = f_{\boldsymbol{\lambda}}(\boldsymbol{\epsilon})$$
$$q(\boldsymbol{\theta}) = q(\boldsymbol{\epsilon}) \left|\det \boldsymbol{J}\right|$$

Desired properties:

→ Fast sampling

→ Density for generated samples

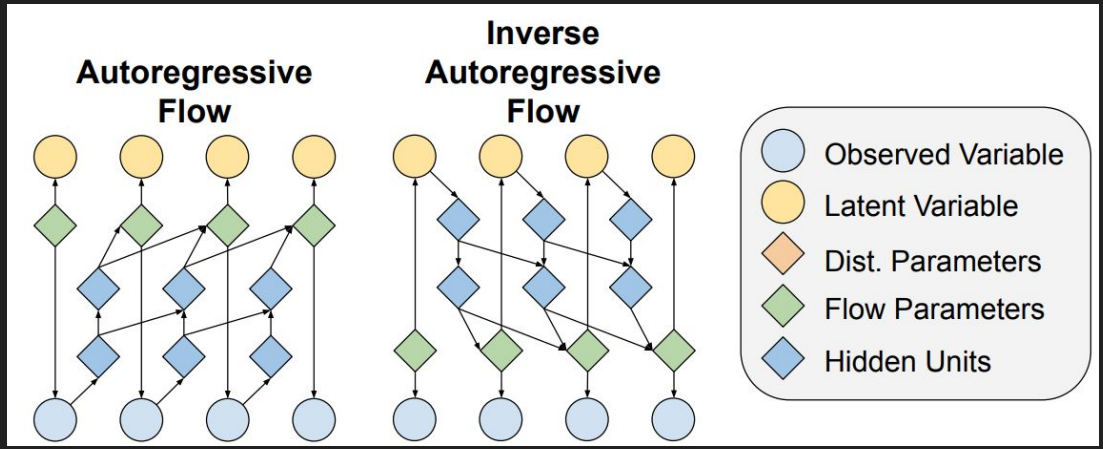
Autoregressive Flow

Forward: D passes (slow)
Inverse: 1 pass (fast)

Inverse Autoregressive Flow

Forward: 1 pass (fast)
Inverse: D passes (slow)

We only need the forward direction!



Other Flows for VI?

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

Sylvester Normalizing Flows for Variational Inference

Rianne van den Berg* University of Amsterdam
Leonard Hasenclever* University of Oxford
Jakub M. Tomczak University of Amsterdam
Max Welling University of Amsterdam

Det. Identities:

Planar, radial, Sylvester, etc.

Autoregressive:

MAF, IAF, NAF, TAN, etc.

Coupling:

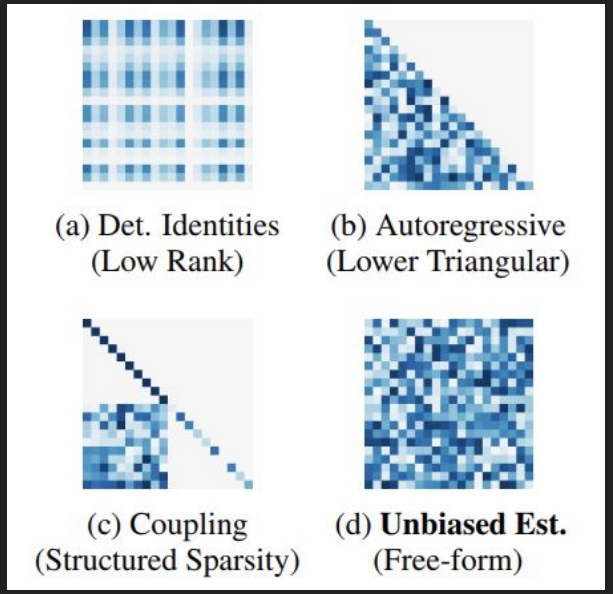
NICE, RealNVP, Glow, Flow++, etc.

Unbiased:

FFJORD, Residual Flows, etc.

Improved Variational Inference with Inverse Autoregressive Flow

Diederik P. Kingma dpkingma@openai.com
Tim Salimans tim@openai.com
Rafal Jozefowicz rafal@openai.com
Xi Chen peter@openai.com
Ilya Sutskever ilya@openai.com
Max Welling* M.Welling@uva.nl



Taken from (Chen et al., 2019)

Example: Variational Autoencoders

Prior: $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$

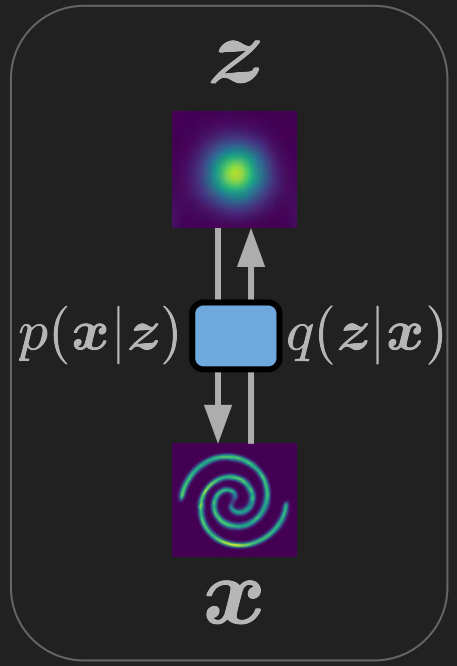
Likelihood: $p(\boldsymbol{x}|\boldsymbol{z}) = p(\boldsymbol{x}|\mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{z}))$

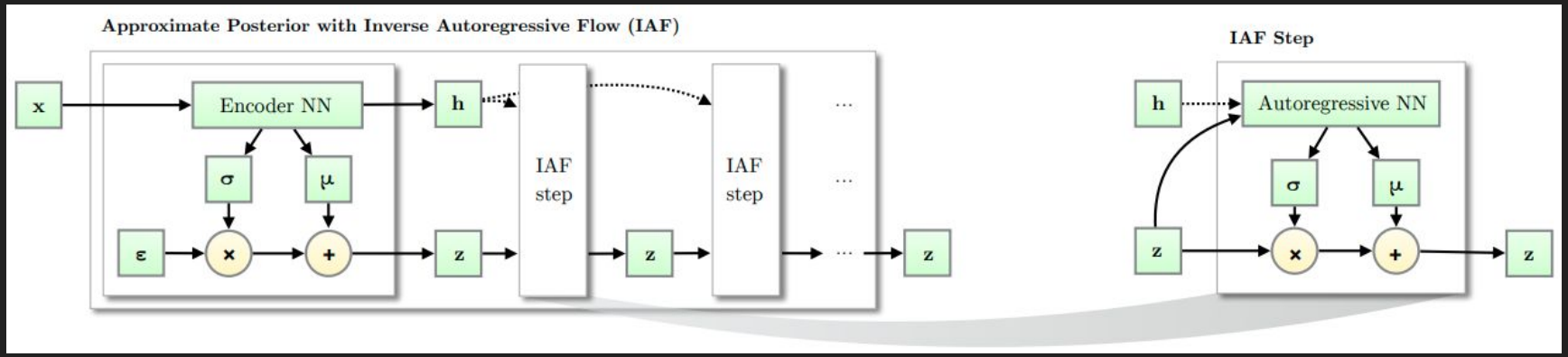
Mean-field:

$$q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$$
$$\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \leftarrow \mathrm{NN}_{\boldsymbol{\phi}}(\boldsymbol{x})$$

Flow:

$$q(\boldsymbol{z}|\boldsymbol{x}) = q(\boldsymbol{\epsilon}) \left| \det \frac{\partial f_{\boldsymbol{\lambda}}(\boldsymbol{\epsilon}; \boldsymbol{x})}{\partial \boldsymbol{\epsilon}} \right|^{-1}$$





Example: Bayesian Neural Networks

$$\mu_n = \mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$$

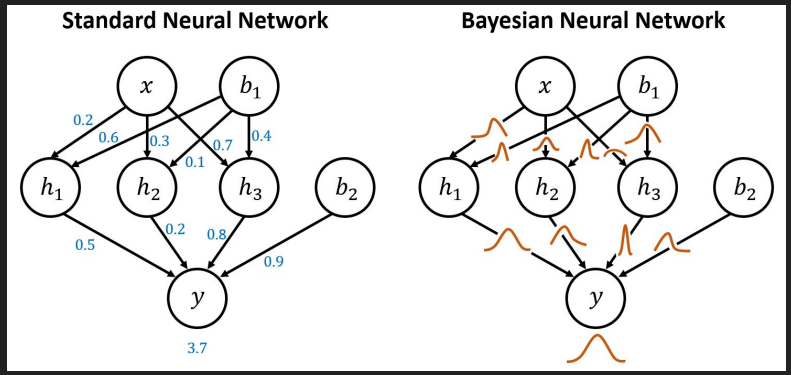
Likelihood: $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$

$\mathrm{Bern}(y_n|\mu_n)$

$\mathcal{N}(y_n|\mu_n, \sigma^2)$

$$\boldsymbol{\theta} = \{\boldsymbol{w}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{w}_L, \boldsymbol{b}_L\}$$

Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \boldsymbol{I})$



Mean-field:

$$q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$$

Flow:

$$q(\boldsymbol{\theta}) = q(\boldsymbol{\epsilon}) \left| \det \frac{\partial f_{\boldsymbol{\lambda}}(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} \right|^{-1}$$

Scalable Bayesian Neural Networks

Local Reparameterization Trick (Kingma et al. 2015):

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \qquad \boldsymbol{W}, \boldsymbol{b} \sim q(\boldsymbol{W}, \boldsymbol{b})$$

$$q(\boldsymbol{W}, \boldsymbol{b}) \sim \mathcal{N} \implies q(\boldsymbol{h}) \sim \mathcal{N}$$

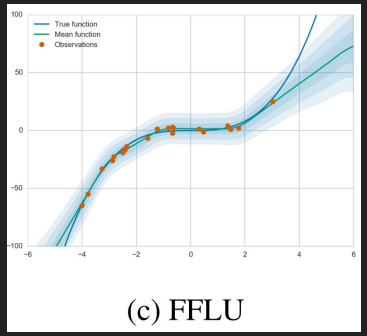
Much lower gradient variance!

Multiplicative Normalizing Flows (Louizos & Welling 2017):

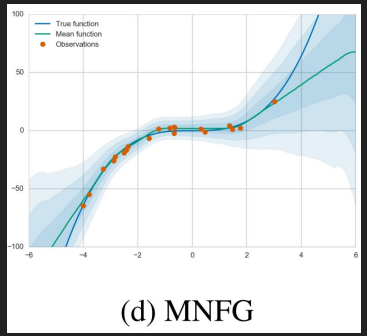
$$q(\boldsymbol{z}) = q(\boldsymbol{\epsilon}) |\det \boldsymbol{J}|$$

$$q(\boldsymbol{W}|\boldsymbol{z}) = \prod_i \prod_o \mathcal{N}(z_i \mu_{io}, \sigma_{io}^2)$$

- Can still apply LRT
- More flexible q



(c) FFLU



(d) MNFG

Thanks for listening!