COMP 760 Week 3: Generative Models Primer I

By Joey Bose and Prakash Panangaden

Mila

Generative models

Destroying structure by adding noise to data is straightforward; transforming noise to structured data is ***generative modelling.***

Ingredients Of A Generative Model

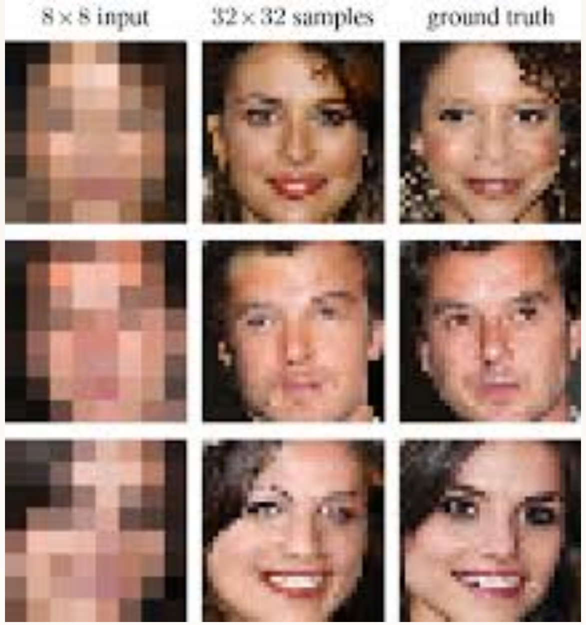
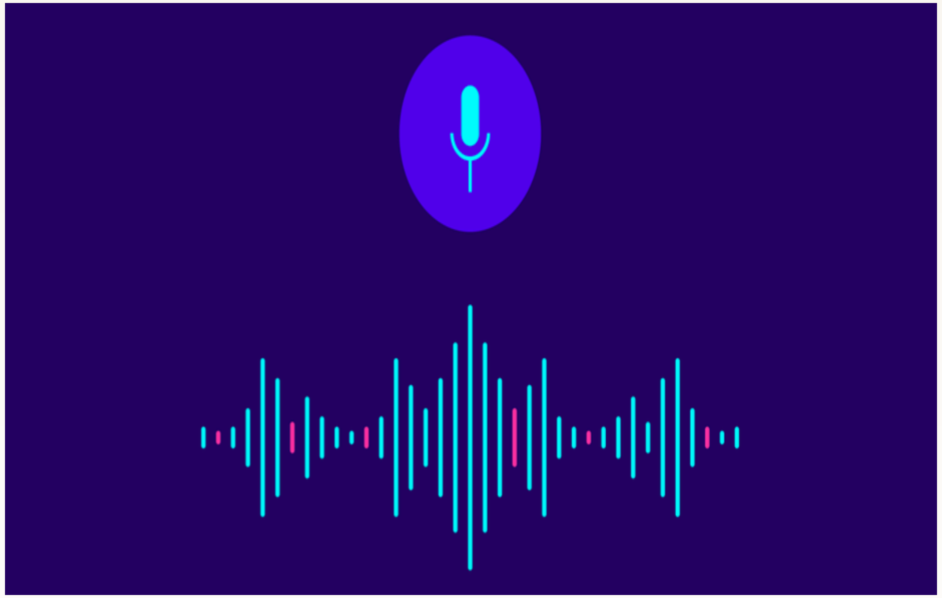


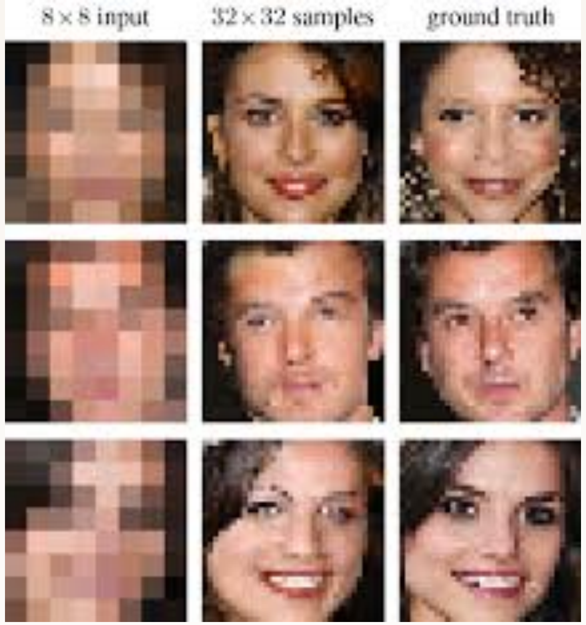
Image Super Resolution



Text to Speech

Drug Discovery

Ingredients Of A Generative Model



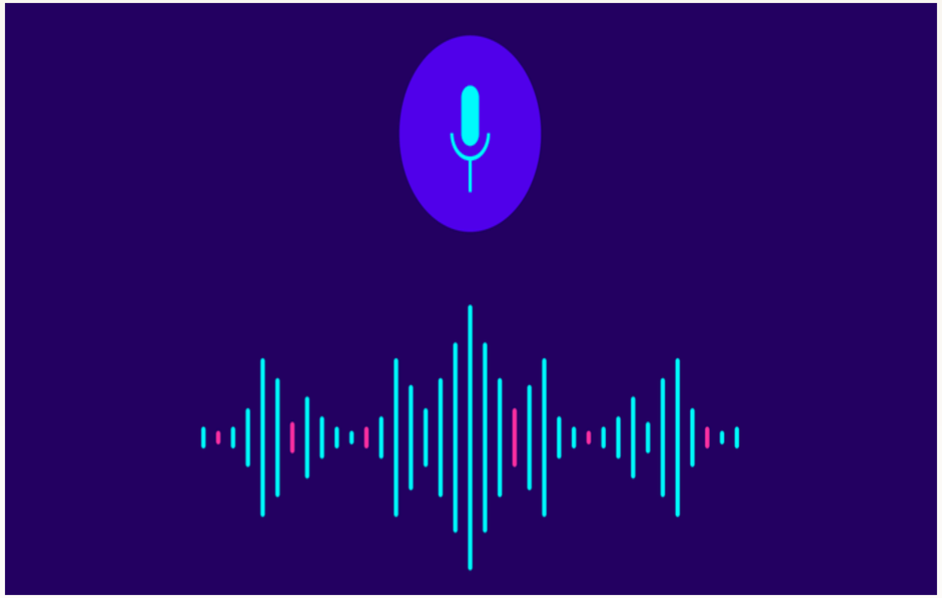


Image Super Resolution

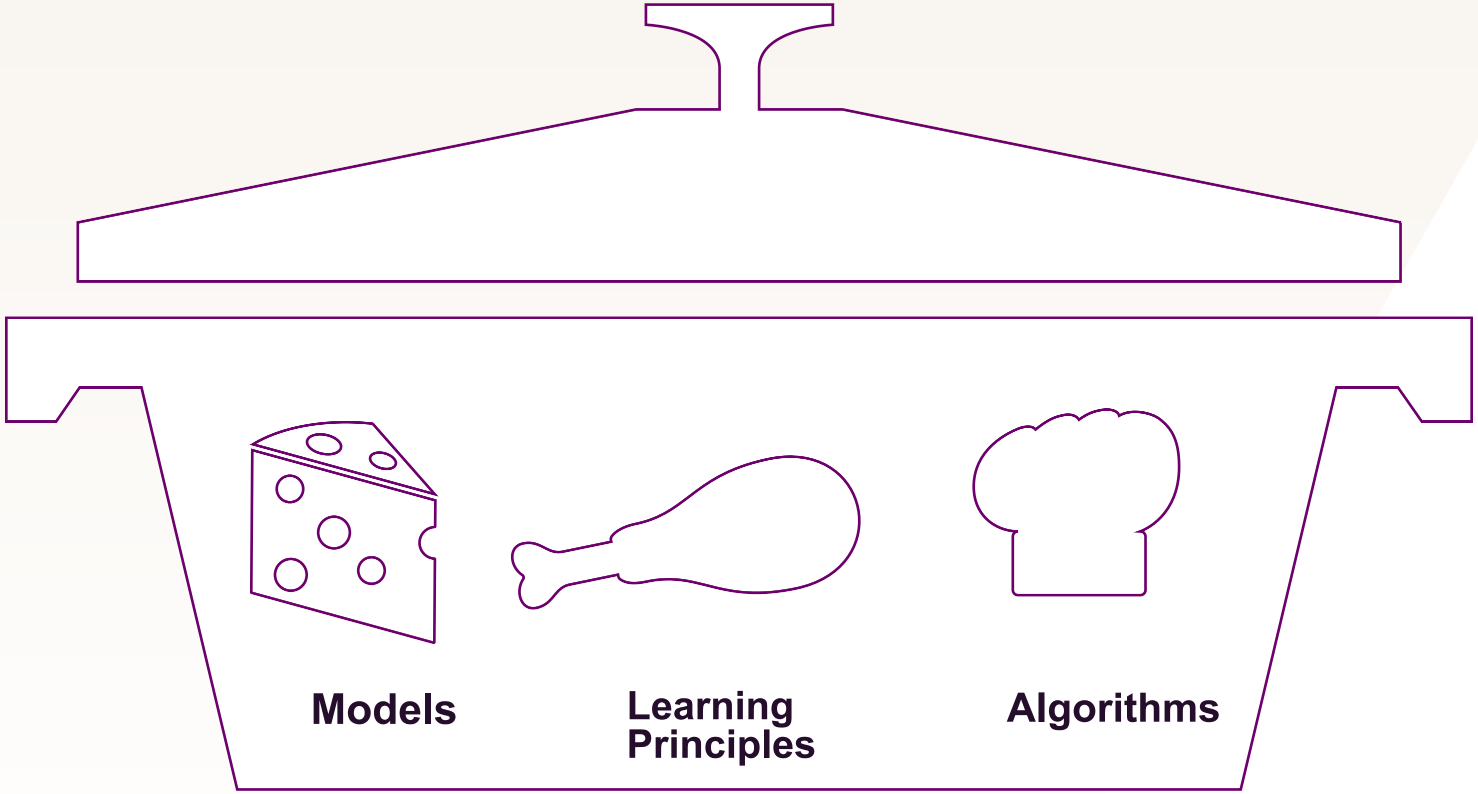
Text to Speech

Drug Discovery

What do we know about the data already?

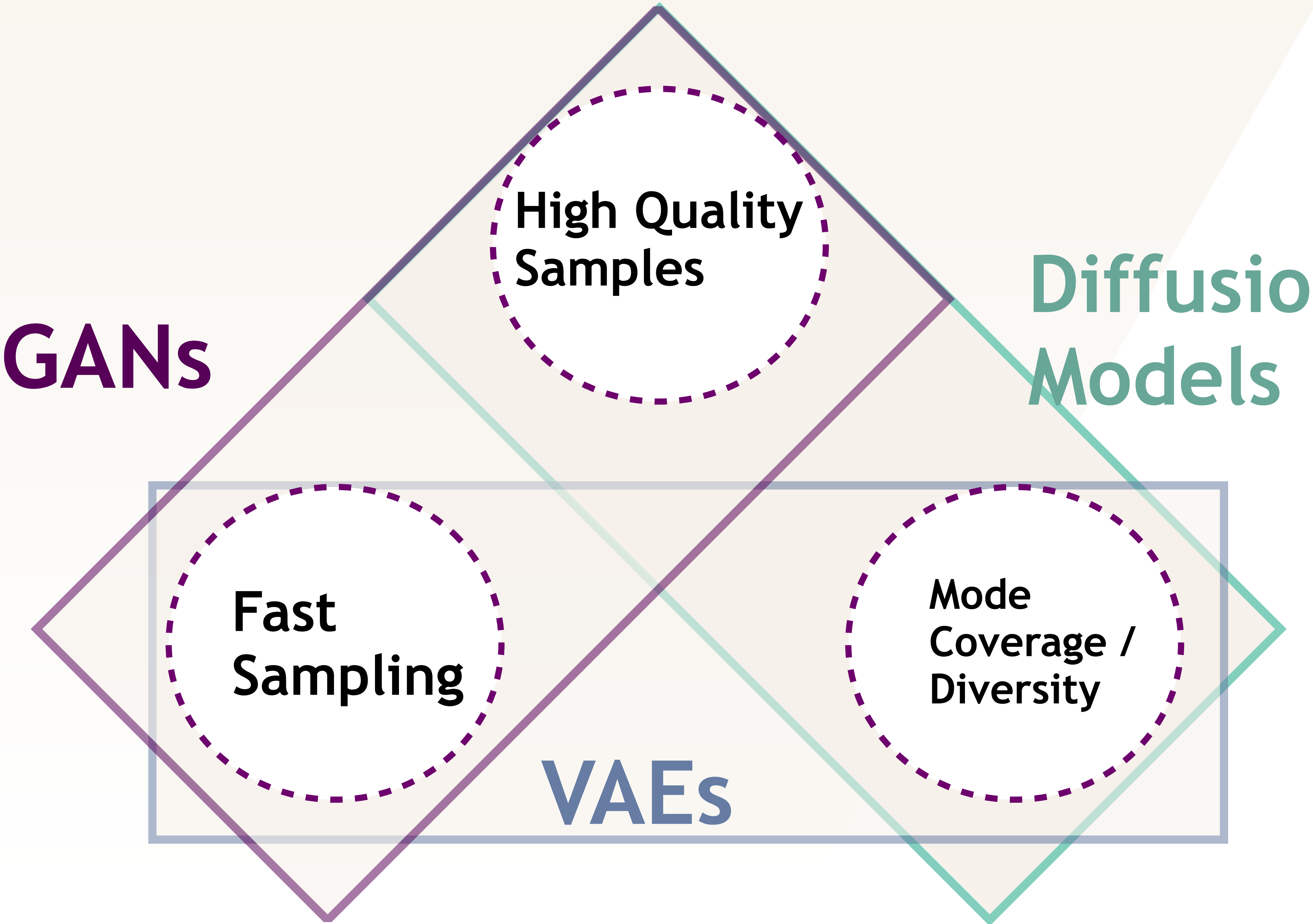
Mila

Ingredients Of A Generative Model

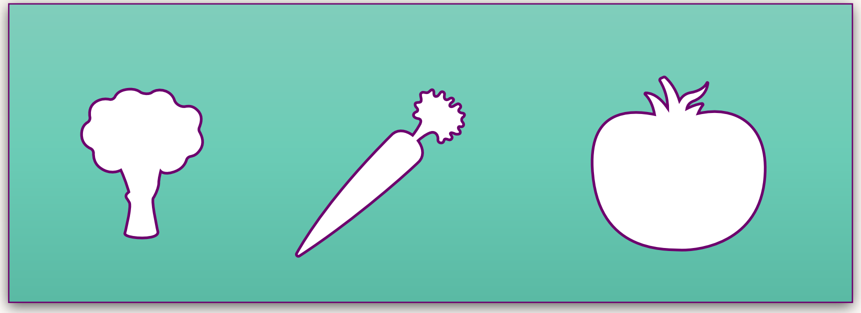


Generative Model Soup

Tradeoffs in a Generative Model

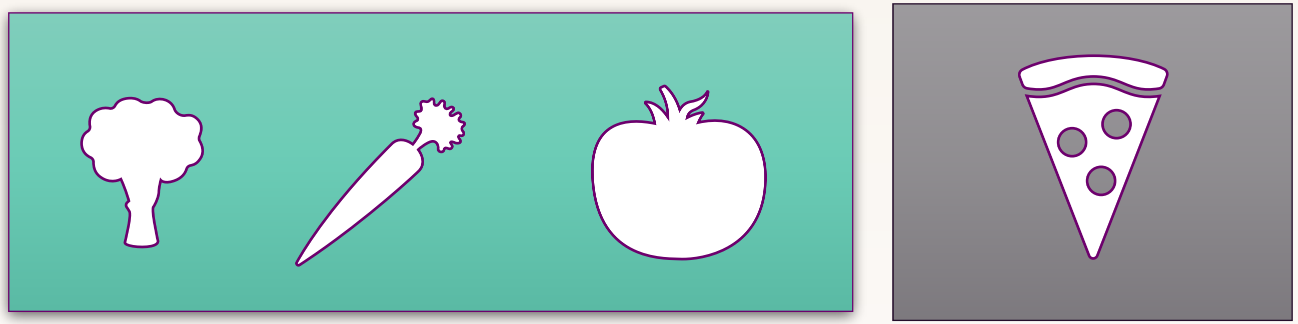


General Setup



Data (Empirical Data Distribution)

General Setup



Data + Noise

$$p(\mathscr{D})$$

We want to model this

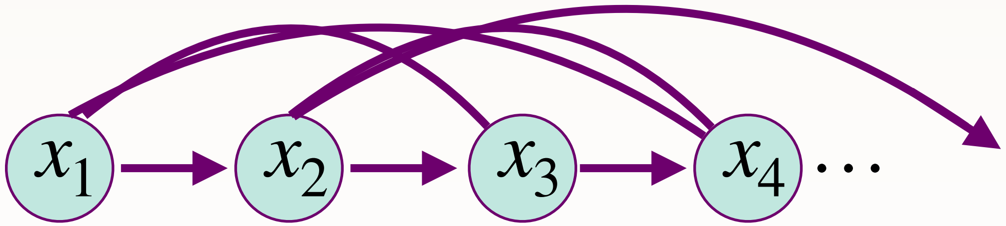
$$x \in \mathbb{R}^n \qquad \mathscr{D} = \{x_i\} \qquad i \in \{1, \ldots, N\}$$

Families of Generative Models

Models

- Fully Observed Models

Models that observe all data components directly without introducing any assumptions about unobserved variables.



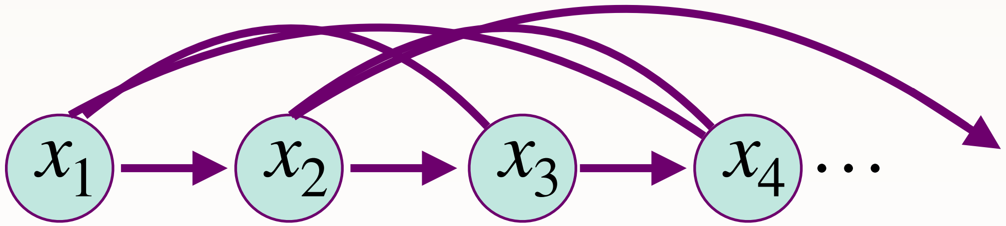
Mila

Families of Generative Models

Models

Fully Observed Models

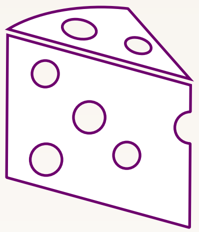
Models that observe all data components directly without introducing any assumptions about unobserved variables.



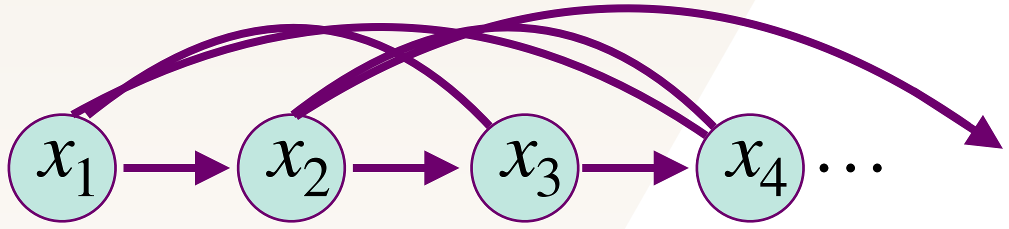
$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

Mila

Families of Generative Models



Fully Observed Models



$$p(x_1, \dots, x_n) = \prod_{i=1}^{n} p(x_i \,|\, x_1, \dots, x_{i-1})$$

- **Explicit form for Log-Likelihoods**
- **Efficient to scale up (e.g. Large Language Models)**

- **Order Sensitive**
- **Sequential Generation which can be slow**

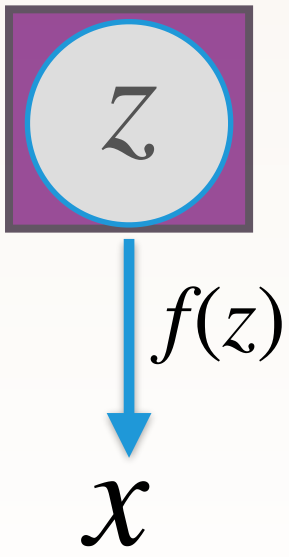
Families of Generative Models

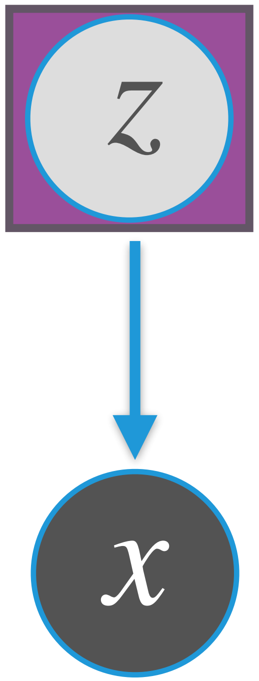
Models

- Latent Variable Models

Prescribed Models: Assume observed likelihoods plus some observation noise (VAE's, Flows).

Implicit Models: Likelihood Free Models (e.g. GANs).



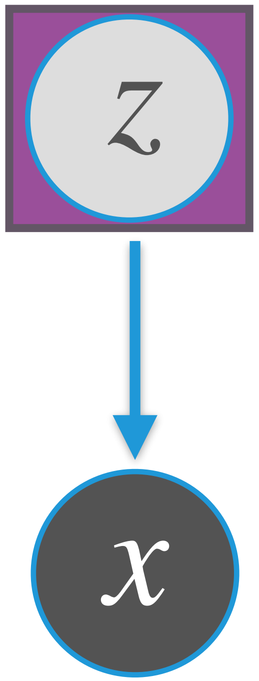


Families of Generative Models

Latent Variable Models

- Fast Sampling
- Avoids order dependency; can encode priors

- Model Class to Approximate the Posterior may not be rich enough
- Inversion process from inputs to latent might be hard



Families of Generative Models

Learning Principles

- Maximum Likelihood
- Expectation Maximization
- Contrastive Methods
- Monte Carlo
- Variational Methods

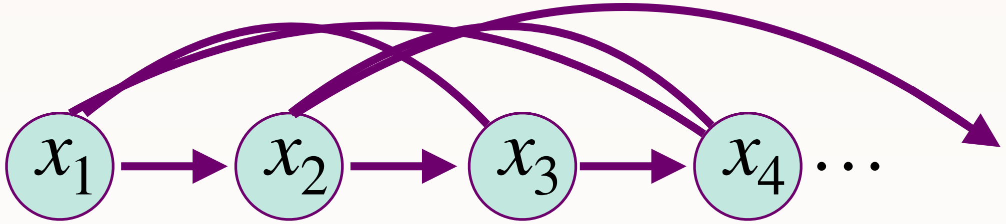
Many more not listed!!!

Families of Generative Models

Learning Principles

- Maximum Likelihood

$$\max_{\theta} \log p_{\theta}(\mathscr{D})$$

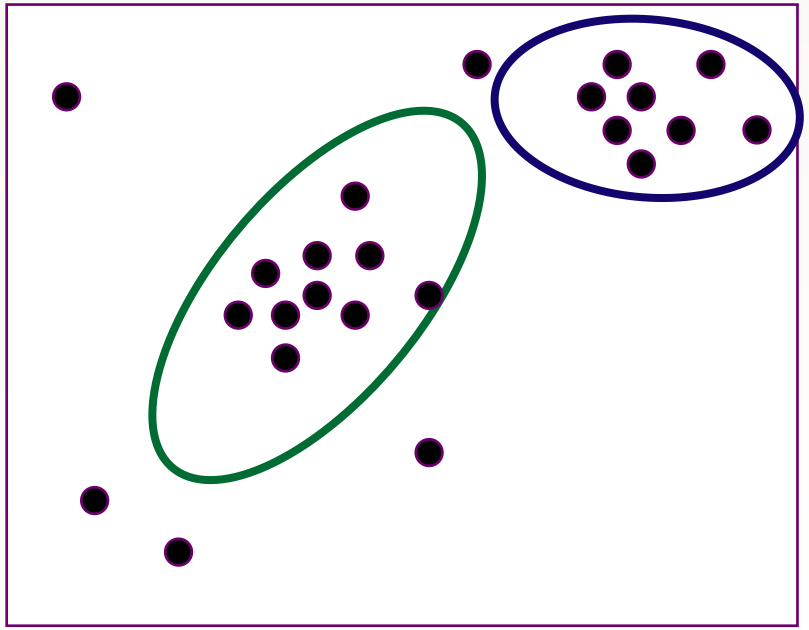


E.g. Autoregressive Models

Families of Generative Models

Learning Principles

- Expectation Maximization



E.g. Gaussian Mixture Models

Families of Generative Models

Learning Principles

- Contrastive Methods

E.g. Noise Contrastive Estimation

$$\mathscr{L} = -\log(s(t)) - \log\left(s(t) + \sum_{i=1}^{N} s(t') \right)$$

Score on positive samples — **Score on negative samples**

Mila

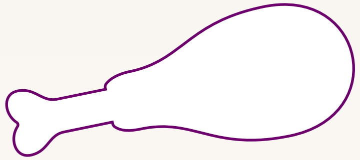
Families of Generative Models

Learning Principles

- Monte Carlo — **E.g. MCMC**

Think Monte-Carlo Gradient Estimation

Families of Generative Models



Learning Principles

- Variational Methods

E.g. Amortized Variational Inference

$$\log p(\mathscr{D}) \geq ELBO$$

Families of Generative Models

Algorithms

Combining models + Learning Principles

- LVMs + Variational Inference
- Implicit Generative Model + Two Sample Test
- Autoregressive Model + Maximum Likelihood

Latent Variable Models

Conventional Approaches to Latent Variable Modelling

$x \in \mathbb{R}^n \quad z \in \mathbb{R}^d \quad \mathcal{D} = \{x_i\} \quad i \in \{1, \ldots, N\}$

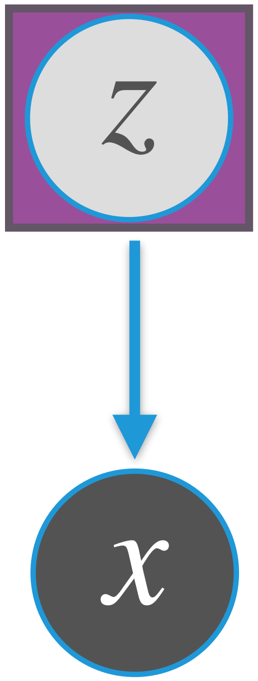
Mila

Conventional Approaches to Latent Variable Modelling

$x \in \mathbb{R}^n \quad z \in \mathbb{R}^d \quad \mathcal{D} = \{x_i\} \quad i \in \{1,\ldots,N\}$

$$\log p(x) = \log \int p(x \mid z)p(z)dz = \log \mathbb{E}_{p(z)}[p(x \mid z)]$$

$$\log p(\mathcal{D}) = \sum_{i=1}^{N} \log \mathbb{E}_{p(z)}[p(x_i \mid z)]$$



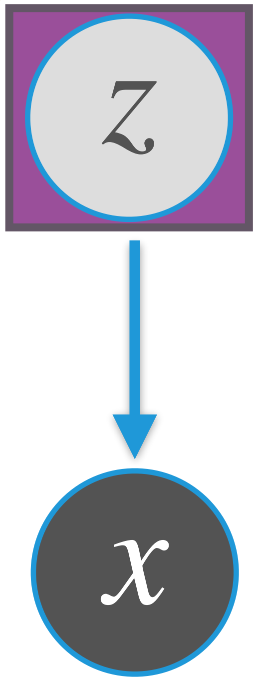
$p(x \mid z)$

Methods for Approximate Inference

- Laplace approximations
- Importance Sampling
- **Variational approximations** → This lecture
- Perturbative corrections
- Other methods: MCMC, Langevin, HMC etc…

Variational Inference

$$\log p(\mathscr{D}) = \log \int p(x \mid z) p(z) dz$$



$p(x \mid z)$

Variational Inference

$$\log p(\mathscr{D}) = \log \int p(x \mid z) p(z) dz$$

$$\log p(\mathscr{D}) = \log \int p(x \mid z) p(z) \frac{q(z)}{q(z)} dz$$

Importance Sampling

Mila

Variational Inference

$$\log p(\mathscr{D}) = \log \int p(x \mid z) p(z) dz$$

$$\log p(\mathscr{D}) = \log \int p(x \mid z) p(z) \frac{q(z)}{q(z)} dz$$

Importance Sampling

$$\log p(\mathscr{D}) \geq \mathbb{E}_{q_i(z)} \left[\log \frac{p(x_i, z_i)}{q_i(z)} \right]$$

Jensen's Inequality

Variational Inference

$$\log p(\mathcal{D}) = \log \int p(x \mid z) p(z) dz$$

$$\log p(\mathcal{D}) = \log \int p(x \mid z) p(z) \frac{q(z)}{q(z)} dz$$

Importance Sampling

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q_i(z)} \left[\log \frac{p(x_i, z_i)}{q_i(z)} \right]$$

Jensen's Inequality

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q_i(z)}[\log p(x_i \mid z)] + \mathbb{E}_{q_i(z)} \left[\log \frac{p(z)}{q_i(z)} \right]$$

Variational Inference

$$\log p(\mathscr{D}) \geq \mathbb{E}_{q(_i(z)}[\log p(x_i \,|\, z)] + \mathbb{E}_{q_i(z)}\left[\log \frac{p(z)}{q_i(z)}\right]$$

$$\log p(\mathscr{D}) \geq \boxed{\mathbb{E}_{q_i(z)}[\log p(x_i \,|\, z)] - D_{KL}(q_i(z) \,||\, p(z))}$$

Evidence Lower Bound (ELBO)

$$\log p(\mathscr{D}) \geq -\mathscr{F}(x, z)$$

Free Energy of the System

Variational Inference

$$\log p(\mathscr{D}) \geq \mathbb{E}_{q(_i(z)}[\log p(x_i \,|\, z)] + \mathbb{E}_{q_i(z)}\left[\log \frac{p(z)}{q_i(z)}\right]$$

$$\log p(\mathscr{D}) \geq \boxed{\mathbb{E}_{q_i(z)}[\log p(x_i \,|\, z)]} - \boxed{D_{KL}(q_i(z) \,||\, p(z))}$$

Reconstruction Error **Regularizer**

Variational Auto-Encoder

Arguably the catalyst **to the** Deep Generative Modelling Revolution.

Variational Auto-Encoders

Key Innovations:

- Amortized Inference
- Reparametrization Trick

Amortized Inference

- Introduce a parametric family of densities *shared over* all the data

Data $\longrightarrow$ Inference Network/ Encoder/ Recognition Model $\longrightarrow$ Approximate Posterior $q_\phi(z \,|\, x)$

$$\arg\max_{q_i} \mathbb{E}_{q_i^*(z)}[-\mathcal{F}(x_i, z)] \longrightarrow \arg\max_{\phi} \mathbb{E}_{q_\phi(z|x)}[-\mathcal{F}_\phi(x_i, z)]$$

VAE - Generative Model

- Sample a Reparametrized Latent z

Latent sample

$z \sim q_\phi(z|x) \longrightarrow$ **Generative Model** $p_\psi(x|z)$ $\longrightarrow \tilde{x}$

Reconstructed datum

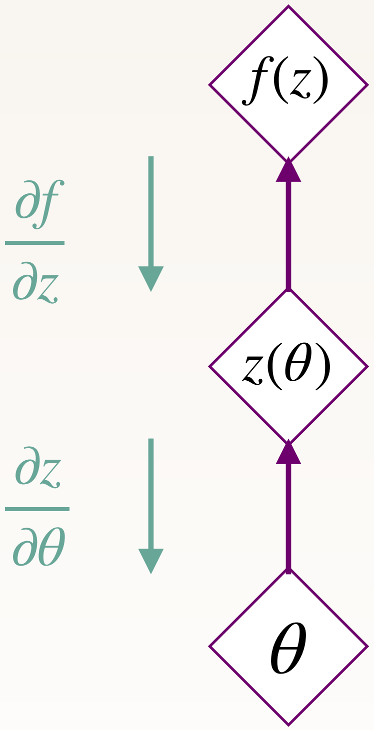
$$\log p(\mathcal{D}) \geq \boxed{\mathbb{E}_{q_\phi(z|x)}[\log p_\psi(x|z)]} - \boxed{D_{KL}(q_\phi(z|x)||p(z))}$$

Reconstruction Error **Regularizer**

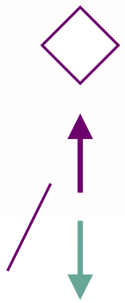
VAE - All the pieces

- Prior $p(z)$ usually a standard Normal $\mathcal{N}(0,I)$
- Encoder network $q_\phi(z|x)$
- Sample from the approximate posterior $z \sim q_\phi(z|x)$
- Generative Network $p_\psi(x|z)$
- Model parameters $\theta = \{\phi, \psi\}$
- Objective: Optimize ELBO

What about Reparametrization?



This is the classical setting in many DL problems

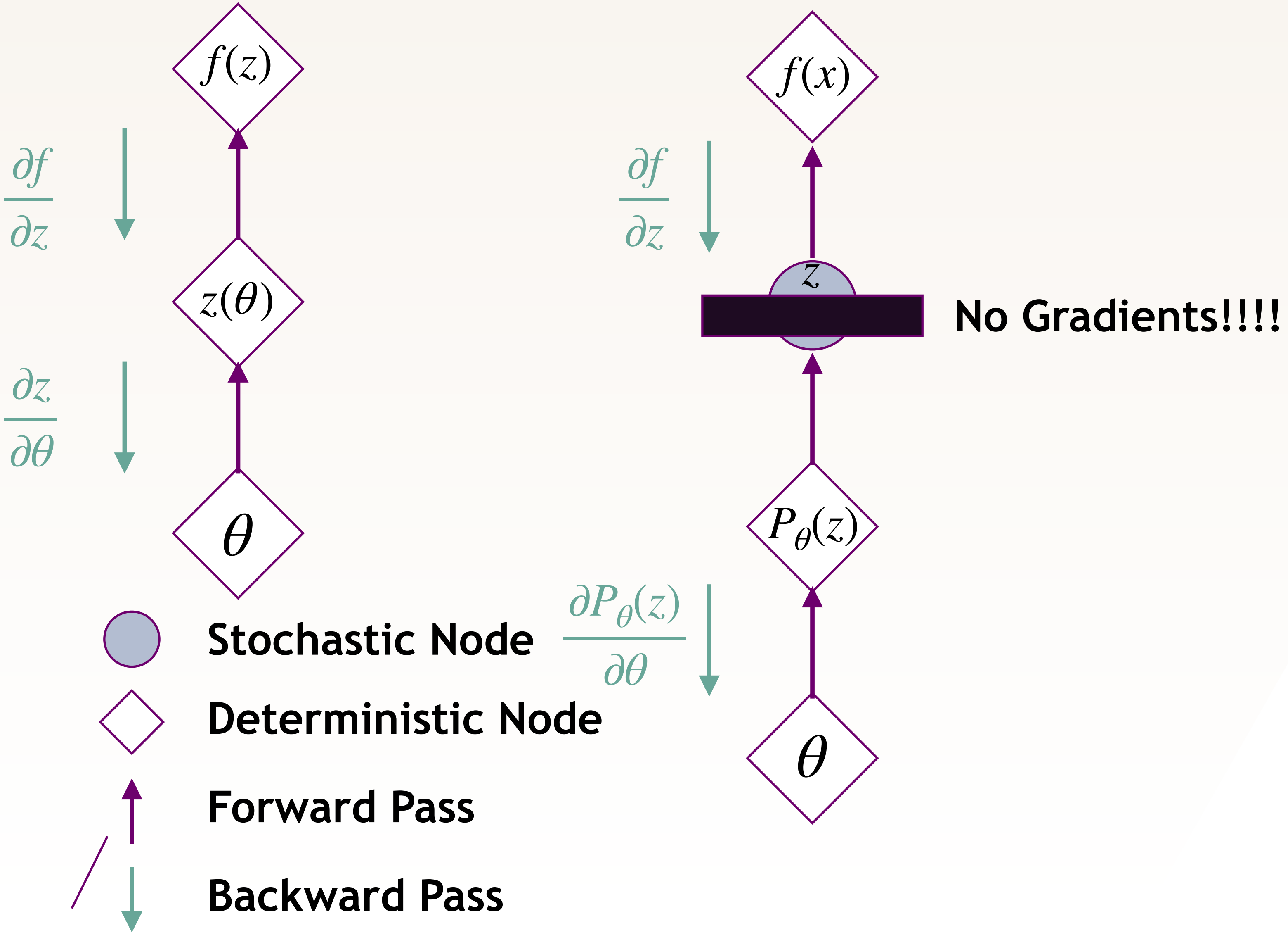


Deterministic Node

Forward Pass

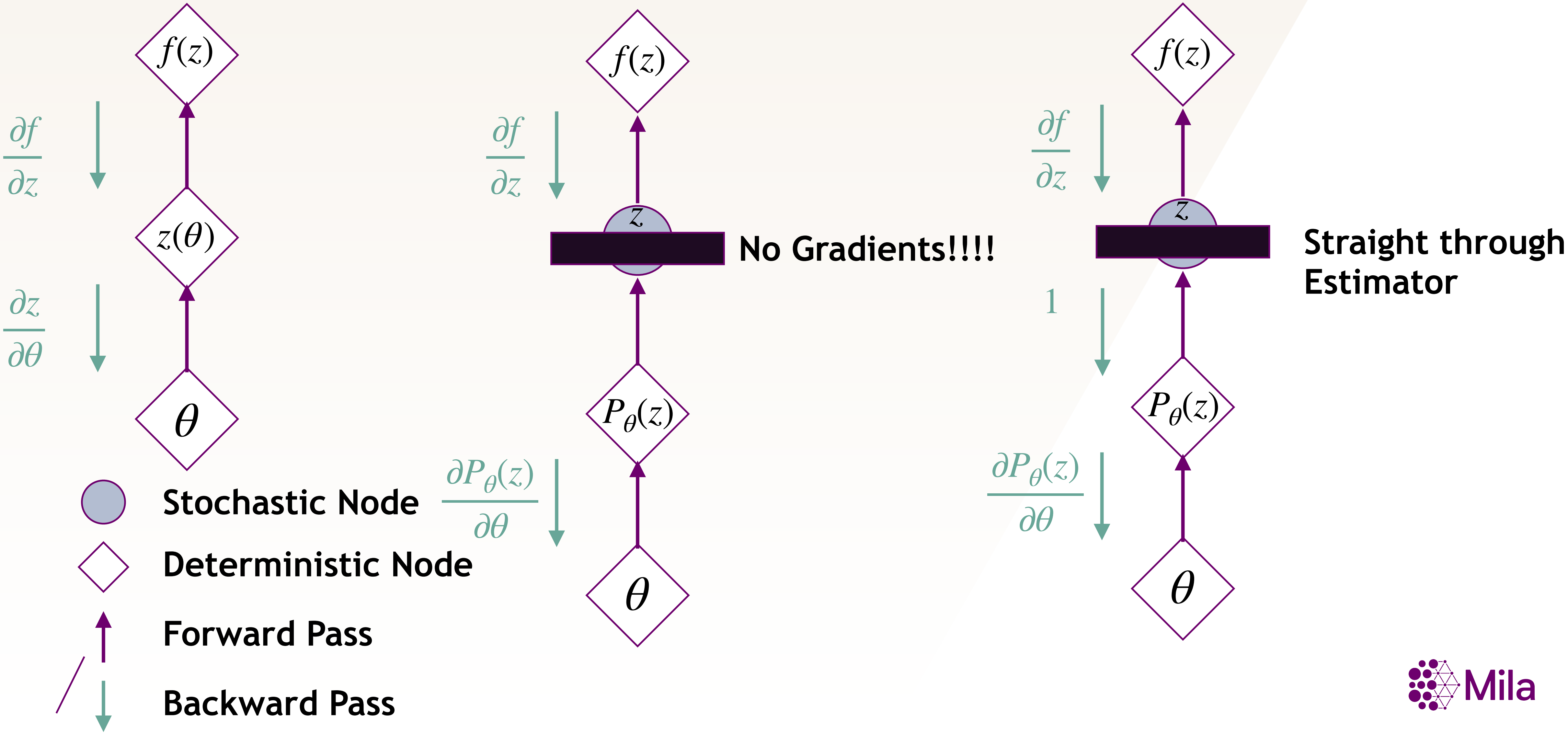
Backward Pass

What about Reparametrization?



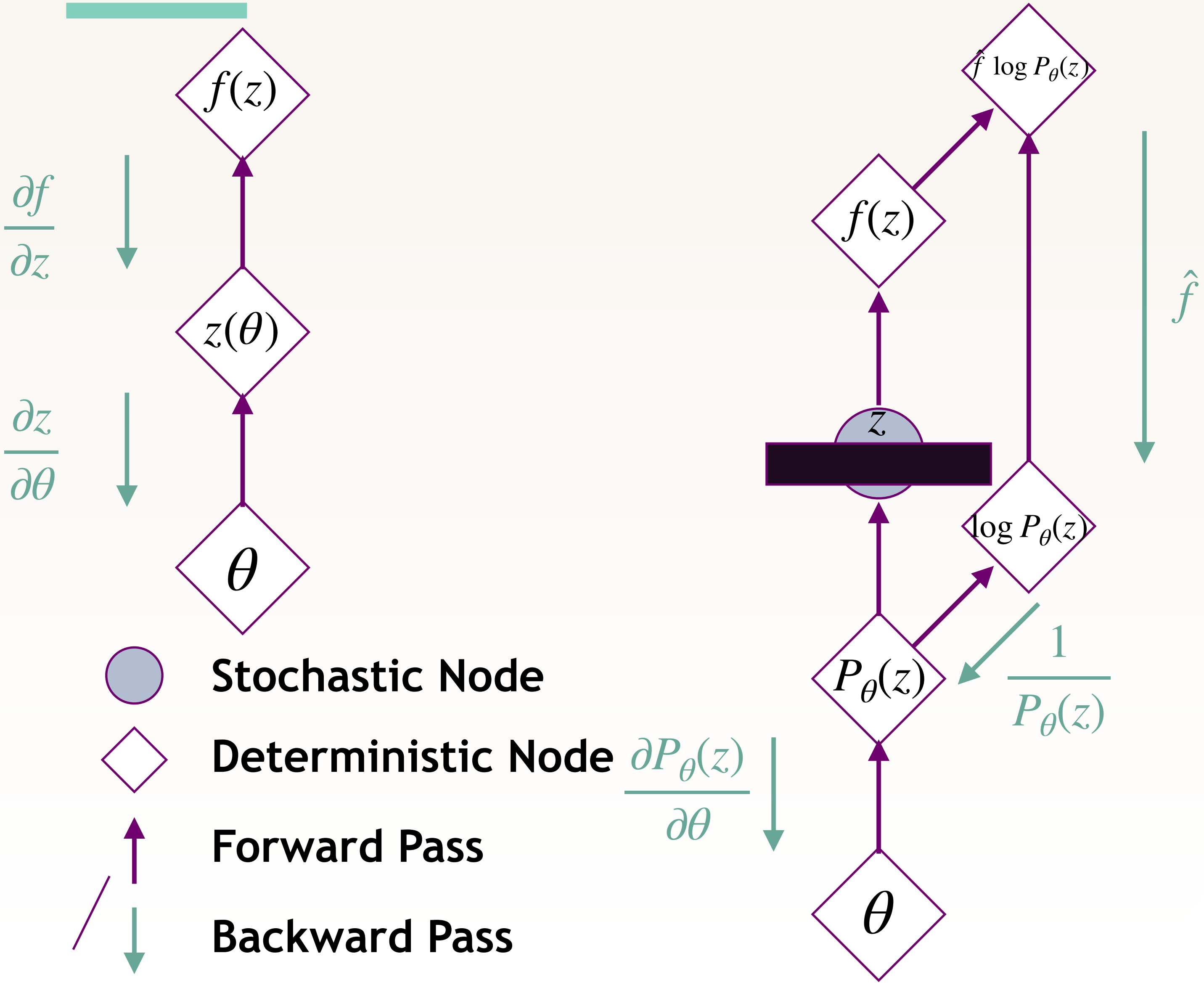
Mila

What about Reparametrization?



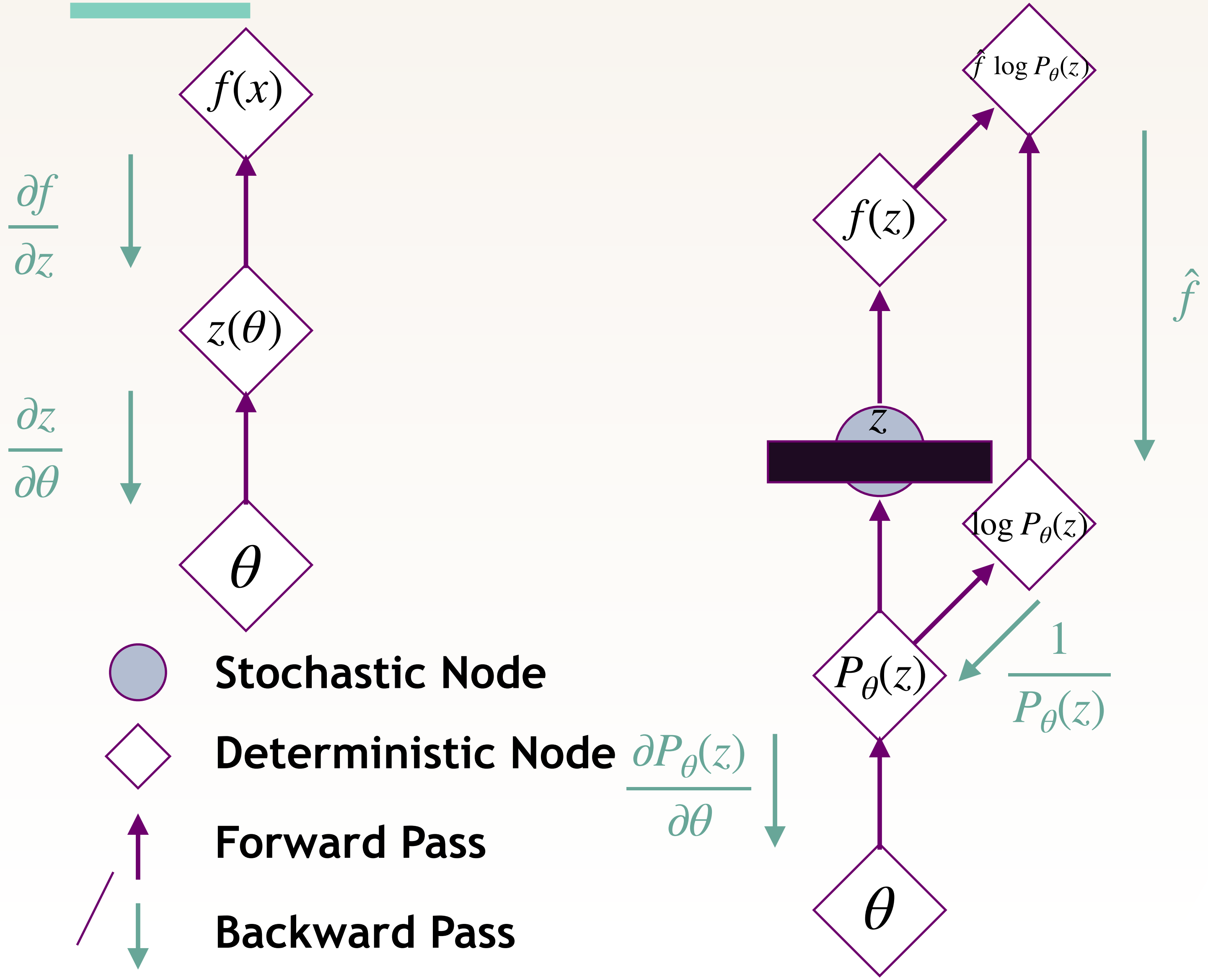
Mila

What about Reparametrization?



Does this look familiar to those who do RL?

What about Reparametrization?



Does this look familiar to those who do RL?

Problem:

$$\nabla_\theta \mathbb{E}_{z \sim p_\theta(z)}[f(z)]$$

REINFORCE:

$$\nabla_\theta \mathbb{E}_{z \sim p_\theta(z)}[f(z)] = \nabla_\theta \mathbb{E}_{z \sim p_\theta(z)}[f(z)\, \nabla_\theta \log p_\theta(z)]$$

Mila

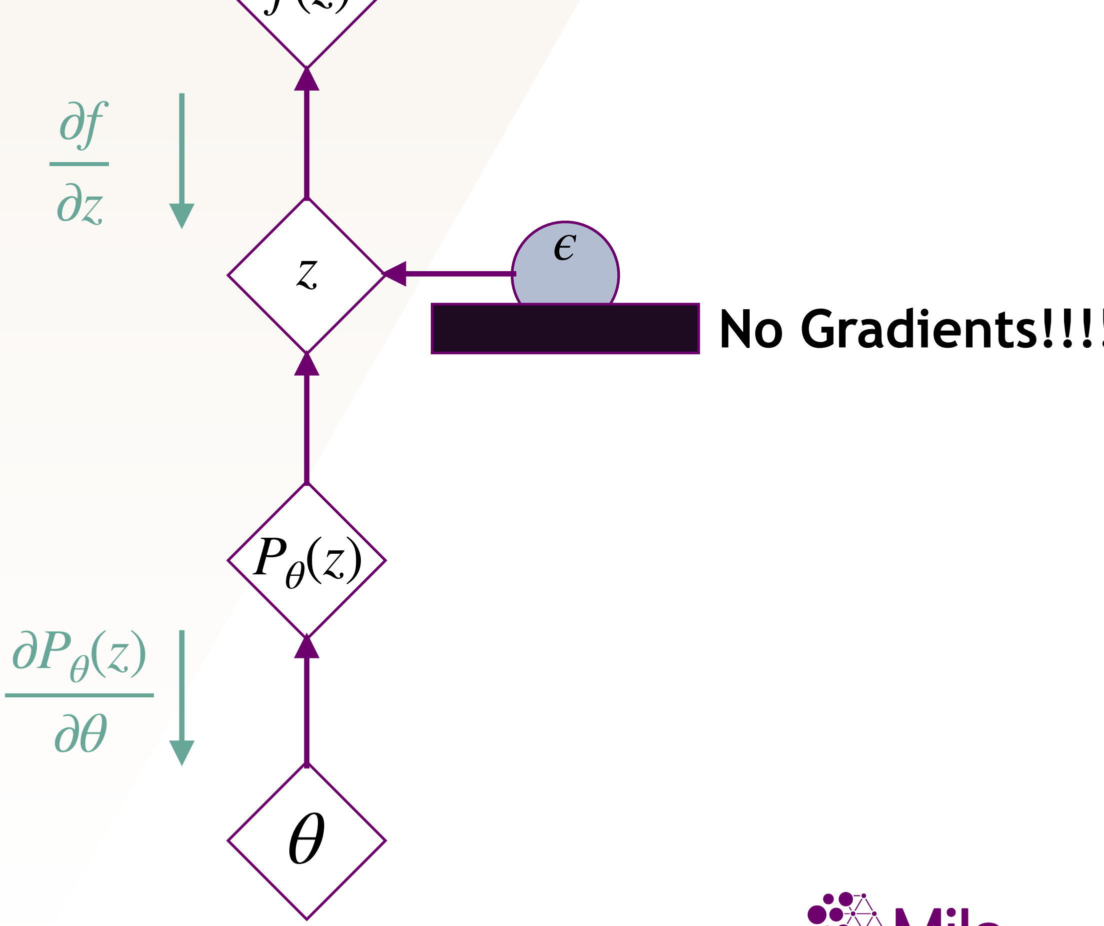
Reparametrization Trick in Gaussian VAE's

$$z \sim q_\phi(z|x)$$

$$z = \mu + \epsilon \odot \sigma^2$$

$$z = \mu + \epsilon \odot \sigma^2$$

$$\epsilon \sim \mathcal{N}(0,I)$$



We can backdrop normally now!

Application: Lossless Compression

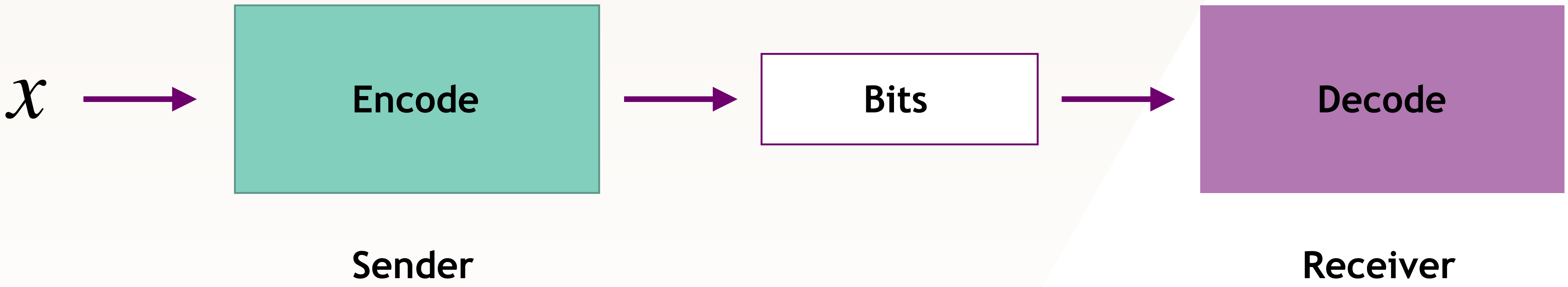
Statistical patterns in the data

"... randomising letters in the middle of words [has] little or no effect on the ability of skilled readers to understand the text. This is easy to denmtrasote. In a pubiltacion of New Scnieitst you could ramdinose all the letetrs, keipeng the first two and last two the same, and reibadailty would hadrly be aftcfeed. My ansaylis did not come to much beucase the thoery at the time was for shape and senqeuce retigcionon. Saberi's work sugsegts we may have some pofrweul palrlael prsooscers at work. The resaon for this is suerly that idnetiyfing coentnt by paarllel prseocsing speeds up regnicoiton. We only need the first and last two letetrs to spot chganes in meniang."

--- Graham Rawlinson, PhD Thesis 1976, Nottingham U.

Application: Lossless Compression

Compression is a process that encodes the original data into a representation that takes fewer bits.



Application: Lossless Compression

The Coding Problem:

- Suppose we are given M : $s_1, \ldots, s_M$
- We want to create a bit string for these symbols $(0,1)$
- How many bits do we need per symbol?

Without any further information we would need $\log_2 M$ bits

Mila

Application: Lossless Compression

The Coding Problem:

- Suppose we are given $M: s_1, \ldots, s_M$
- The sender and receiver both know the alphabet $\mathscr{A}$
- The sender and receiver both have access to a probabilistic model for each symbol.
- Assume Sender and Receiver have agreed upon a standard encoding/decoding paradigm: Arithmetic coding/ANS

How can we communicate the data with the fewest number of bits?

Mila

Application: Lossless Compression

Shannon's Source Coding Theorem

$$H(x) = -\mathbb{E}_{p(x)}[\log p(x)]$$

This is the best that we can do we both sender and receiver both use $p(x)$

In practice the receiver may not have access to $p(z \mid x)$.
What should they do then? Approximate it!

Mila

Application: Lossless Compression

Entropy Coding Example

A $p_A = 0.5$

B $p_B = 0.5$

$H(A,B) = -p_A \log_2 p_A - p_B \log_2 p_B$

$H(A,B) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5$

$H(A,B) = 1$

A $p_A = 0.8$

B $p_B = 0.2$

$H(A,B) = -p_A \log_2 p_A - p_B \log_2 p_B$

$H(A,B) = -0.8 \log_2 0.8 - 0.2 \log_2 0.2$

$H(A,B) \approx 0.7219$

Less than 1 bit/symbol

Application: Lossless Compression

Some common coding strategies

- Huffman Coding
- Arithmetic Coding
- Lempel-Ziv
- Entropy Coding

Handling continuous and high dimensional data is a problem!

Application: Lossless Compression

Notation:

- x - data we want to compress
- z - latent variable; compressed representation
- $p(x)$ - marginal or data distribution
- $p(z|x)$ - the true posterior distribution over data
- $q(z|x)$ - Approximate posterior distribution over data

Application: Lossless Compression

How can the sender communicate $\tilde{x} \sim p(x)$?

- Sender and Receiver both have access to $p(z)$ and $p(x \mid z)$

1. Sender samples $z \sim p(z)$ and then uses it to encode $\tilde{x}$ using $p(\tilde{x} \mid z)$

2. Effective code length is $-(\log p(\tilde{x}) + \log p(\tilde{x} \mid z))$ bits.

Can we do better?

Application: Bits-Back Coding

Suppose we have access to an LVM

$p(x \mid z)p(z)$ We compress z first using $p(z)$. Then we compress x using $p(x \mid z)$.

The receiver would execute this process in reverse to decode

$$H(x,z) = -\mathbb{E}_{p(x,z)}[\log p(x,z)]$$

$$= \underbrace{\boxed{H(p(x \mid z))}}_{\text{Size of bitstream to encode } x} + \underbrace{\boxed{H(p(z))}}_{\text{Size of bitstream to encode } z}$$

Mila

Application: Bits-Back Coding

Clearly,

$$H(p(x,z)) \geq H(p(x))$$

What's the optimality gap?

Recall $$p(x|z) = \frac{p(z|x)p(z)}{p(x)}$$

$$H(x) = -\mathbb{E}_{p(x)}\left[\log \frac{p(x|z)p(z)}{p(z|x)}\right]$$

$$= -\mathbb{E}_{p(x,z)}\left[\log p(x|z)\right] - \mathbb{E}_{p(x,z)}\left[\log p(z)\right] + \mathbb{E}_{p(x,z)}\left[\log p(z|x)\right]$$

$$= H(p(x|z)) + H(p(z)) - H(p(z|x))$$

We overestimate the length by $H(p(z|x))$ bits, Bits-Back will address this

Application: Bits-Back Coding

But the receiver doesn't have $p(z \mid x)$?

$q(z \mid x)$ We need to build an approximate posterior.

What's the minimum code length now?

$$\mathbb{E}_{q(z|x)}\left[-\log p(x \mid z) - \log p(z) + \log q(z \mid x)\right]$$

$$= \mathbb{E}_{q(z|x)}\left[-\log p(x, z) + \log q(z \mid x)\right]$$

$$= \mathbb{E}_{q(z|x)}\left[-\log p(z \mid x) - \log p(x) + \log q(z \mid x)\right]$$

$$= \mathbb{E}_{q(z|x)}\left[-\log p(x) + \log \frac{q(z \mid x)}{p(z \mid x)}\right]$$

Mila

Application: Bits-Back Coding

But the receiver doesn't have $p(z|x)$?

$q(z|x)$ We need to build an approximate posterior.

What's the minimum code length now?

$$= \mathbb{E}_{q(z|x)}\left[-\log p(x) + \log \frac{q(z|x)}{p(z|x)}\right]$$

$$= \mathbb{E}_{q(z|x)}[-\log p(x)] + D_{KL}(q(z|x)||p(z|x))$$

$$\log p(x) = ELBO + D_{KL}(q(z|x)||p(z|x))$$ KL-Gap due to mismatch between posteriors

$$\geq 0$$

Mila

Application: Bits-Back Coding

Suppose the Sender has additional bits (possibly random)

1. Sender samples uses random bits to generate a sample $\tilde{z}$ by using $p(z \mid x)$ to decode! Generating this sample costs $-\log p(\tilde{z})$ bits.

2. Sender uses $\tilde{z}$ to encode $\tilde{x}$ using $p(\tilde{x} \mid \tilde{z})$ and also encodes $\tilde{z}$ using $p(\tilde{z})$. This costs $-(\log p(\tilde{x}) + \log p(\tilde{x} \mid \tilde{z}))$ as before.

3. The Receiver can now recover this junk information by first decoding $\tilde{x}$ then $\tilde{z}$, effectively getting the "bits-back". The net cost for this scheme is $-\log p(\tilde{x} \mid \tilde{z}) - \log p(\tilde{z}) + \log p(\tilde{z}) = -\log p(\tilde{x} \mid \tilde{z})$.

Mila

Application: Bits-Back Coding

High Level Idea:

- Sender will use $p(x|z), p(z)$, and $p(z|x)$ as opposed to the first two in LVM.
- The goal is to annihilate the overestimation gap $-H(p(z|x))$.
- If Encoder sees terms from $H(p(x|z))$ or $H(p(z))$ we push to the bitstream making it longer. If decoder sees these terms we pop.
- If Encoder sees terms from $-H(p(z|x))$ we pop. Conversely, if decoder sees these terms we push to the bitstream.
- During encoding we pop $-H(p(z|x))$ and push $H(p(x|z))$ and $H(p(z))$. Same for Decoding.

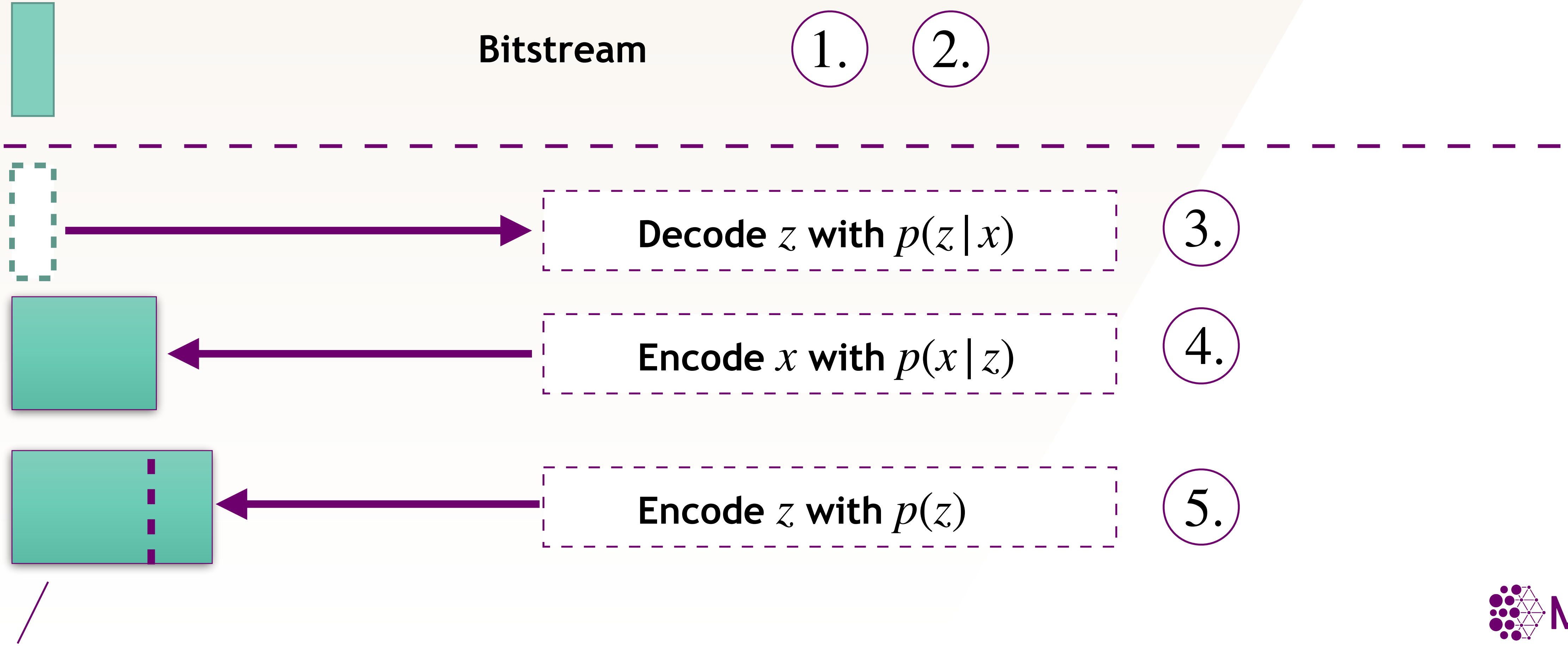
Application: Bits-Back Coding

Sender Perspective

1. Assume there is an initial Bitstream
2. Sender sends x; has access to $x, z, p(x|z), p(z), p(z|x)$
3. Sender uses initial bit-stream and decodes z using $p(z|x)$. Pop from bitstream
4. Sender encodes x using $p(x|z)$. Push onto bitstream.
5. Sender encodes z using $p(z)$. Push onto bitstream.

Application: Bits-Back Coding

Sender Perspective



Mila

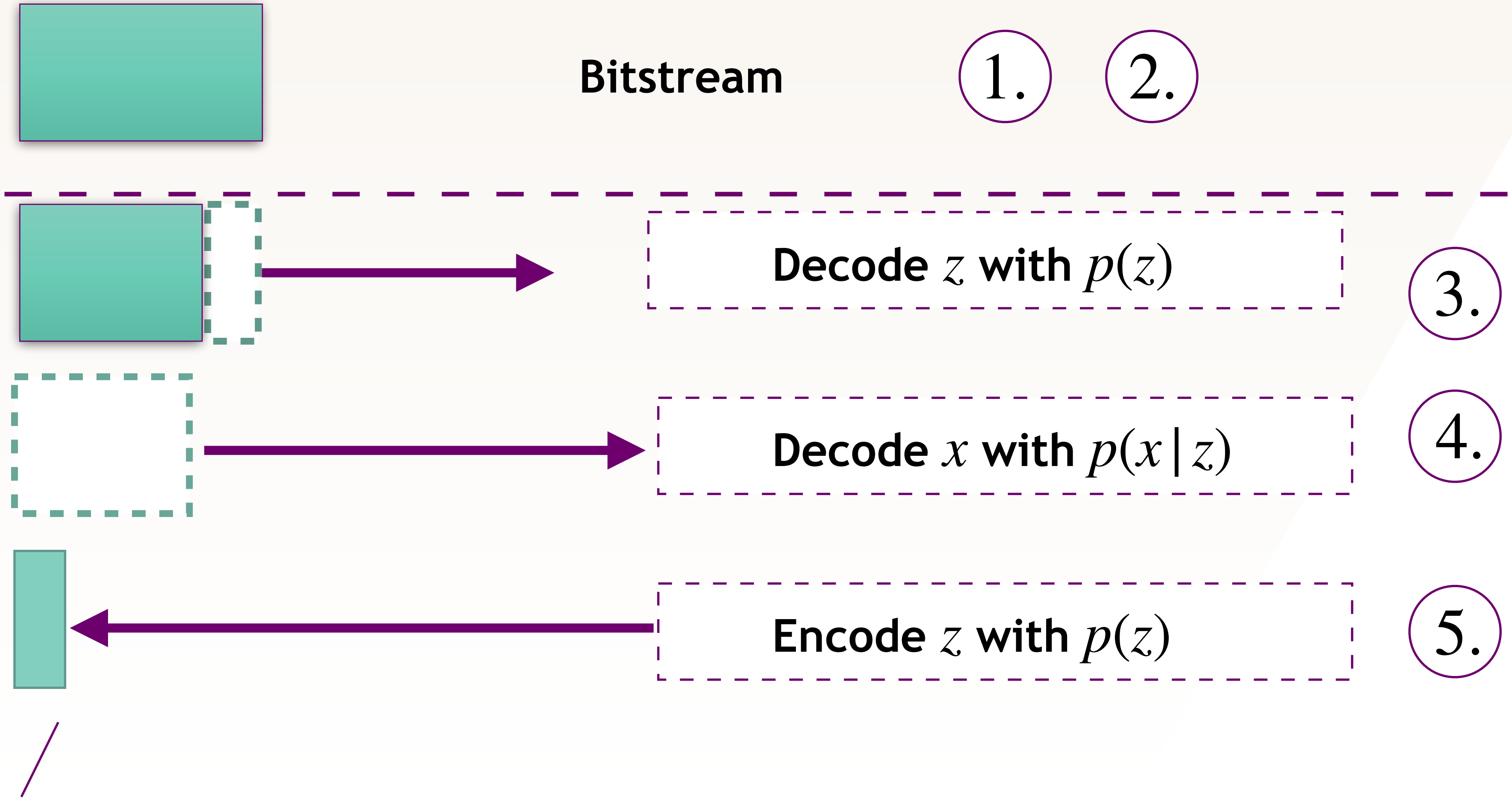
Application: Bits-Back Coding

Receiver Perspective

1. Receiver has access to $p(x \mid z), p(z), p(z \mid x)$.
2. Receiver decodes z using $p(z)$. Pop from bitstream.
3. Receiver decodes x using $p(x \mid z)$. Pop from bitstream.
4. Receiver encodes z using $p(z \mid x)$. Push to the bitstream.
5. Receiver successfully decodes x and recovers the initial bitstream.

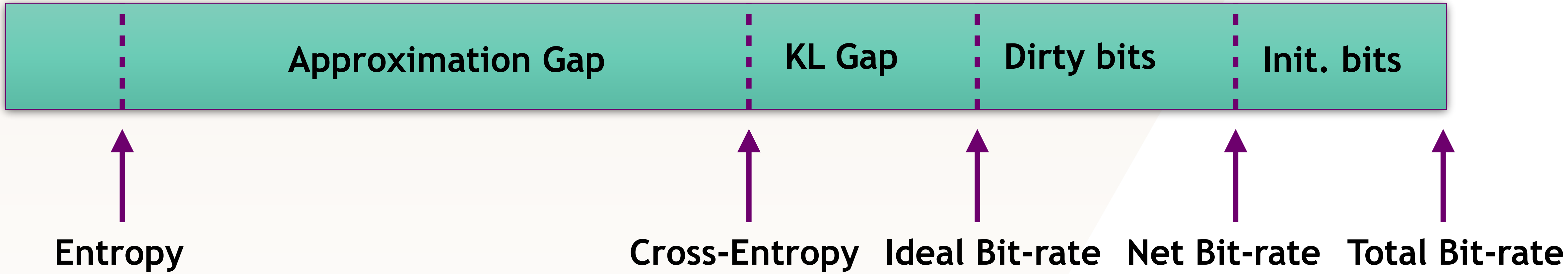
Application: Bits-Back Coding

Receiver Perspective



Application: Bits-Back Coding

The total Bitstream



Addition VAE Topics Not Covered

- Latent Space Disentanglement
- Hierarchical Latent Spaces
- Non-Gaussian Priors
- Identifiability and Causality

GANs

Game Theorists are still thanking Ian Goodfellow et. al 2014 for making their field profitable.

Implicit Generative Models

- What if we design a model where we cannot exactly compute the density?

We can no longer train using Maximum Likelihood and ELBO!

Implicit Generative Models

- What if we design a model where we cannot exactly compute the density?

  We can no longer train using Maximum Likelihood and ELBO!

- We can use a comparison metric between our model density q and p^*

This is known as Implicit Density Estimation

Mila

Implicit Generative Models

- We can use the Density Ratio as our Comparison Metric: $r = \frac{p^*(x)}{q(x)}$

Why is this sensible?

- Suppose we had class labels $y \in [-1,1]$, indicating Fake vs. Real samples

$$r = \frac{p^*(x)}{q(x)}$$

Implicit Generative Models

- Suppose we had class labels $y \in [-1,1]$, indicating Fake vs. Real samples

$$r = \frac{p^*(x)}{q(x)}$$

$$r = \frac{p^*(x \mid y = 1)}{q^*(x \mid y = -1)}$$

Implicit Generative Models

- Suppose we had class labels $y \in [-1,1]$, indicating Fake vs. Real samples

$$r = \frac{p^*(x)}{q(x)}$$

$$r = \frac{p^*(x \mid y = 1)}{q(x \mid y = -1)}$$

$$r = \frac{p^*(y = 1 \mid x)}{q(y = -1 \mid x)}$$ Bayes Rule + Cancellations

Implicit Generative Models

- Suppose we had class labels $y \in [-1,1]$, indicating Fake vs. Real samples

$$r = \frac{p^*(y=1|x)}{q(y=-1|x)}$$ Bayes Rule + Cancellations

- This is more manageable, and we can introduce a parametric scoring function

$$p^*(y=1|x) = D_\theta(x)$$

$$p^*(y=-1|x) = 1 - D_\theta(x)$$

Mila

Generative Adversarial Networks

Latent sample

$$z \sim p(z) \longrightarrow \text{Generator } G(z) \longrightarrow x = G(z)$$

Fake Sample

Generative Adversarial Networks

Latent sample

$z \sim p(z)$

Generator

$G(z)$

Fake Sample

$x = G(z)$

Real Sample

$x \sim p^*(x)$

Discriminator

$D_\theta(x)$

Generative Adversarial Networks

Latent sample

$z \sim p(z)$

Generator

$G(z)$

Fake Sample

$x = G(z)$

Real Sample

$x \sim p^*(x)$

Discriminator

$D_\theta(x)$

$$\min_G \max_D \mathbb{E}_{p^*(x)}[\log(D_\theta(x))] + \mathbb{E}_{q_\phi(\tilde{x})}[\log(1 - D_\theta(\tilde{x}))]$$

Generative Adversarial Networks

Discriminator Loss

$$\mathscr{L}_D = \mathbb{E}_{p^*(x)}[-\log(D_\theta(x))] + \mathbb{E}_z[\log(1 - D_\theta(G_\phi(z))]$$

Generator Loss

$$\mathscr{L}_G = -\mathscr{L}_D$$

Combined Loss

$$\min_G \max_D \mathbb{E}_{p^*(x)}[\log(D_\theta(x))] + \mathbb{E}_{q_\phi(\tilde{x})}[\log(1 - D_\theta(\tilde{x}))]$$

Generative Adversarial Networks

Discriminator Loss

$$\mathscr{L}_D = \mathbb{E}_{p^*(x)}[-\log(D_\theta(x))] + \mathbb{E}_z[\log(1 - D_\theta(G_\phi(z))]$$

Generator Loss

$$\mathscr{L}_G = -\mathscr{L}_D$$

Non-Saturating Generator Loss

$$\mathscr{L}_G = -\mathbb{E}_z[\log D_\theta(G(z))]$$

Combined Loss

$$\min_G \max_D \mathbb{E}_{p^*(x)}[\log(D_\theta(x))] + \mathbb{E}_{q_\phi(\tilde{x})}[\log(1 - D_\theta(\tilde{x}))]$$

Many Breeds of GANs

- Wasserstein GAN
- Wasserstein GAN with Gradient Penalty/Spectral Norm
- CycleGAN
- StyleGAN
- BigGAN

Normalizing Flows

Invented at Mila by Laurent Dinh et. al and at Deep Mind by Danilo Rezende et. al in parallel.

Motivation for Normalizing Flows

- What if we want to do exact Maximum Likelihood density estimation?
- What if we also want fast sampling as opposed to autoregressive models?
- What if we also want to have a rich family of models that are easier to train than GANs?

Setup Normalizing Flows

$$x \in \mathbb{R}^n \quad z \in \mathbb{R}^n \quad \mathscr{D} = \{x_i\} \quad i \in \{1, \ldots, N\}$$

Take a bijective Function

$$f : \mathbb{R}^n \to \mathbb{R}^n \qquad f^{-1} = g \qquad g \circ f(z) = z$$

How does the random variable z with distribution $q(z)$ transform under f?

$$z' = f(z)$$

Setup Normalizing Flows

How does the random variable z with distribution $q(z)$ transform under f?

$$z' = f(z)$$

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{dz'} \right|$$

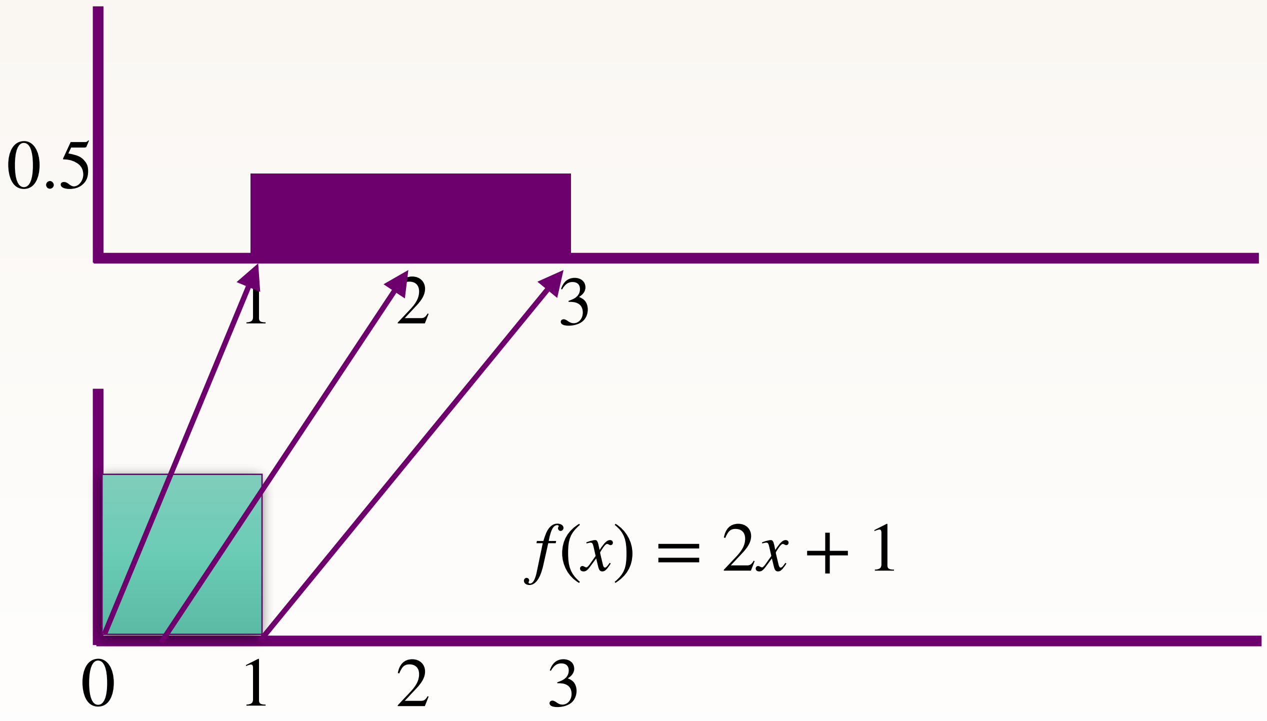
Change of variable formula for Probability Distribution

$$q(z') = q(z) \left| \det \frac{\partial f}{dz'} \right|^{-1}$$

Inverse function theorem

1D Example

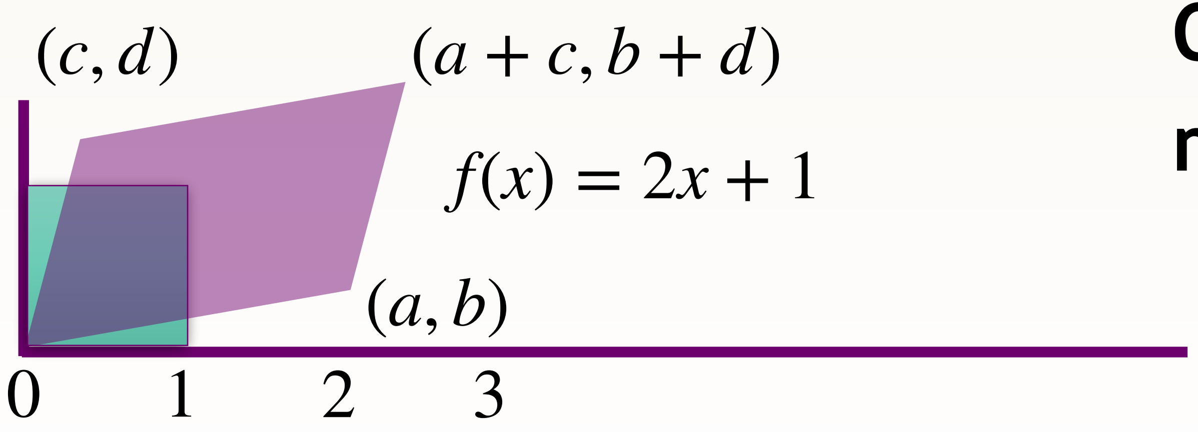
Let X be Uniform$(0,1)$. Let $Y = f(X) = 2X + 1$



Observe that the probability mass must integrate to 1.

1D Example

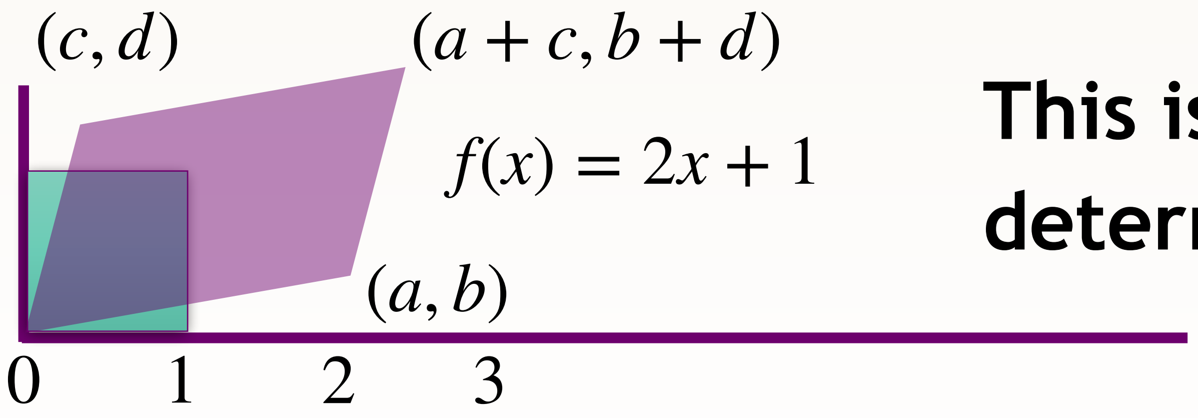
Let X be Uniform$(0,1)$. Let $Y = f(X) = 2X + 1$



Observe that the probability mass must integrate to 1.

1D Example

Let X be Uniform$(0,1)$. Let $Y = f(X) = 2X + 1$



Area of a parallelogram is $ad - bc$.

This is nothing but the determinant of a 2×2 matrix.

Density Estimation with Normalizing Flows

- Each function f_i must be invertible.
- We must be able to efficiently sample from the final distribution, $z_k = f_k \circ f_{k-1} \circ \ldots \circ f_1(z_0)$
- We must be able to efficiently compute the associated change in volume

Change in Volume

$$\boxed{\log p(z_k) = \log p(z_0) - \sum_{j=1}^{k} \log \det \left| \frac{\partial f_j}{\partial z_{j-1}} \right|}$$

Mila

Affine Coupling Flows

Forward Transform

$$y_{1:d} = x_{1:d}$$

$$y_{d+1:D} = x_{d+1:D} \odot exp(s(x_{1:d})) + t(x_{1:d})$$

Inverse Transform

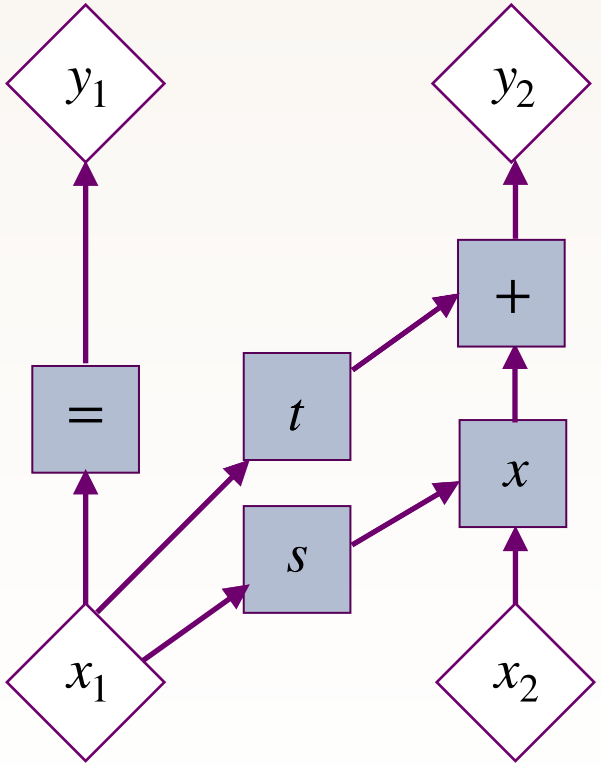
$$x_{1:d} = y_{1:d}$$

$$x_{d+1:D} = (y_{d+1:D} - t(y_{1:d}) \odot \exp(-s(y_{1:d}))$$

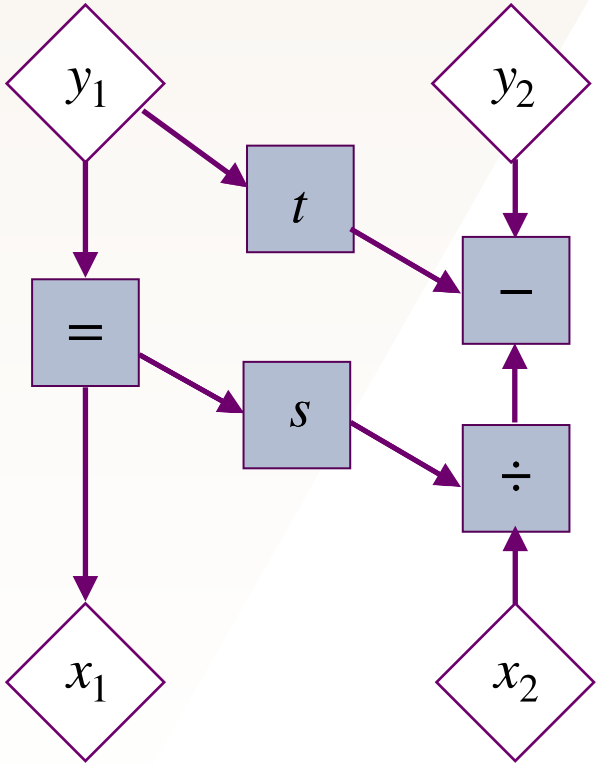
$$\frac{\partial y}{\partial x} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{x^T_{1:d}} & diag(exps(x_{1:d})) \end{bmatrix}.$$

Affine Coupling Flows

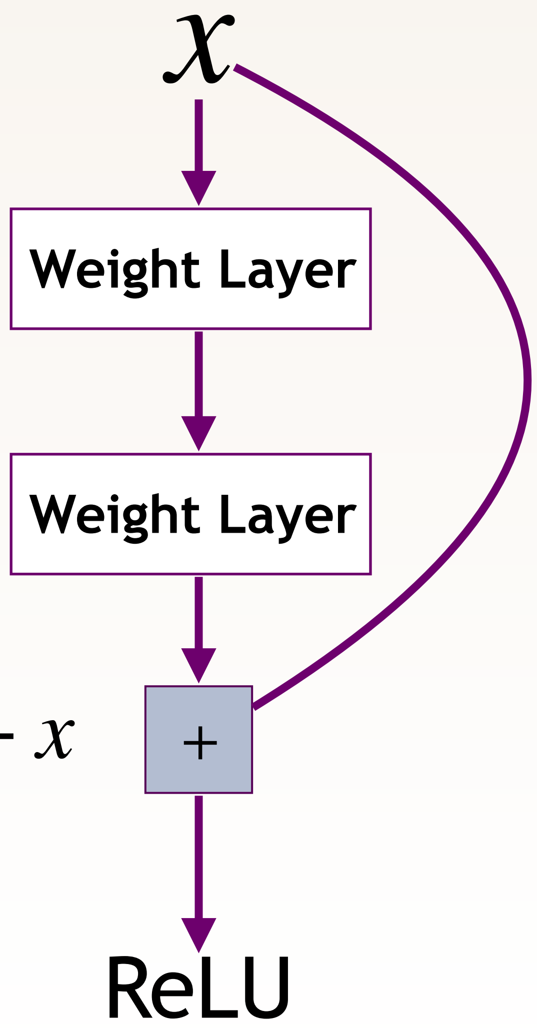
Forward Transform



Inverse Transform



Residual Flows

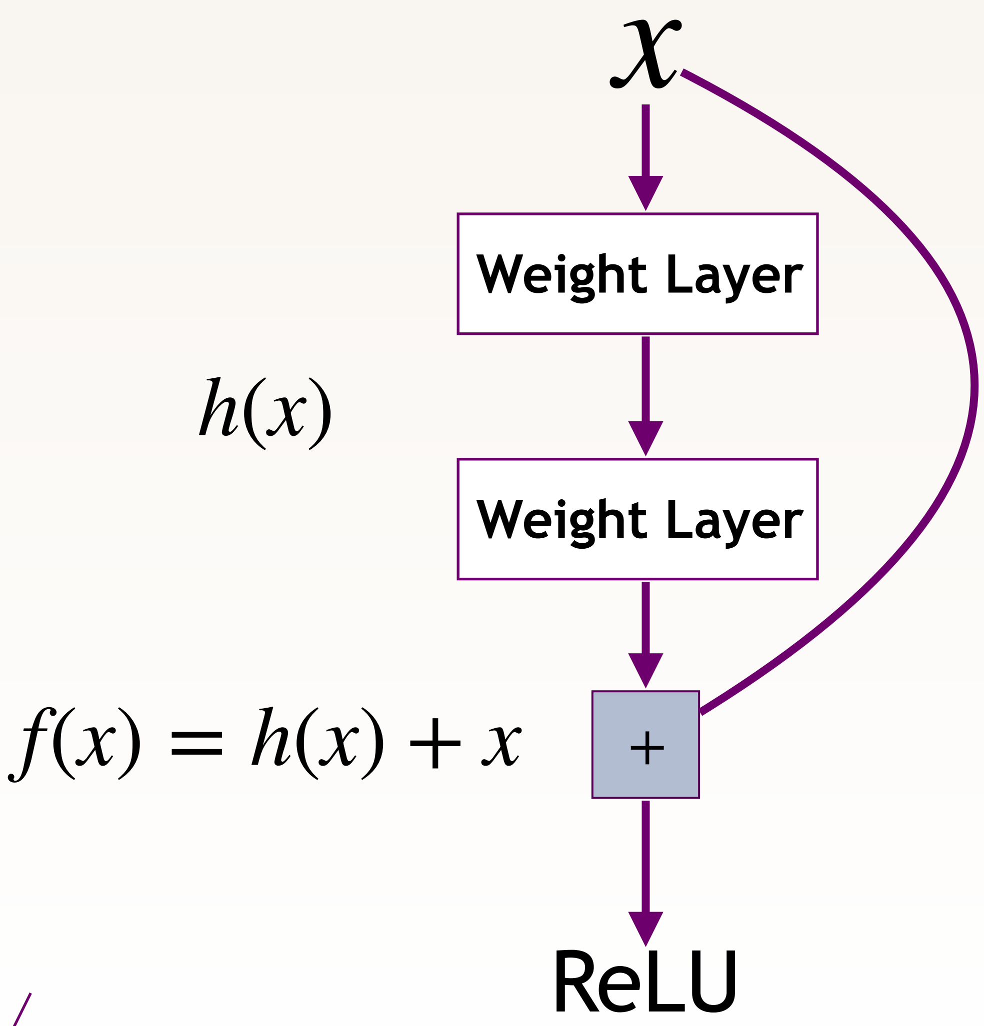


$h(x)$

$f(x) = h(x) + x$

ResNet Block

Residual Flows



Forward Dynamics

$$x_{t+1} \longleftarrow x_t + \alpha h(x_t)$$

This looks an Euler Discretization of an ODE.

Reverse Dynamics

$$x_t \longleftarrow x_{t+1} - \alpha h(x_t)$$

When can we invert a ResNet?

Residual Flows

Sufficient Condition

$$\mathrm{Lip}(h_t) \leq 1$$

This condition basically tells us the IVP has a solution that not only exists and is unique. See Picard-Lindelof Theorem.

Forward Dynamics

$$x_{t+1} \longleftarrow x_t + \alpha h(x_t)$$

This looks an Euler Discretization of an ODE

Reverse Dynamics

$$x_t \longleftarrow x_{t+1} - \alpha h(x_t)$$

When can we invert a ResNet?

Residual Flows

Change of Variable

$$f(x) = h(x) + x$$

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

$$\log p(x) = \log p(f(x)) + \boxed{\mathrm{tr}\left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} [J_h(x)]^k \right)} \qquad J_h = \frac{dh(x)}{dx}$$

Where did this come from?

Residual Flows

Change of Variable

$$\boxed{\mathrm{tr}\left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}[J_h(x)]^k\right)}$$

$$J_h = \frac{df(x)}{dx}$$

$$\left|\det J_f\right| = \det J_f$$

Lipschitz constrained perturbations of the form $x + h(x)$ have positive $\det$

Residual Flows

Change of Variable

$$\boxed{\mathrm{tr}\left(\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k}[J_h(x)]^k\right)}$$

$$J_h = \frac{dh(x)}{dx}$$

$$\left|\det J_f\right| = \det J_f$$

Lipschitz constrained perturbations of the form $x + h(x)$ have positive $\det$

$$\ln\det(J_f) = \mathrm{tr}(\ln(J_f))$$

$$\ln\det(A) = \mathrm{tr}(\ln(A))$$

For non-singular matrices A

Matrix Trace and Matrix Logarithm

Mila

Residual Flows

Change of Variable

$$\boxed{\mathrm{tr}\left(\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k}[J_h(x)]^k\right)}$$

$$J_h = \frac{dh(x)}{dx}$$

$$\left|\det J_f\right| = \det J_f$$

Lipschitz constrained perturbations of the form $x + h(x)$ have positive $\det$

$$\ln\det(J_f) = \mathrm{tr}(\ln(J_f)) \qquad \ln\det(A) = \mathrm{tr}(\ln(A))$$

For non-singular matrices A

$$\mathrm{tr}(\ln(I + J_h(x))) = \mathrm{tr}\left(\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k}[J_h(x)]^k\right)$$

The matrix logarithm has a convergent power series for $||J_h||_2 \leq 1$

Mila

Residual Flows

How do we compute the Trace efficiently?

$\mathrm{tr}(J_h)$ Naive computation takes $O(n^2)$

Trick #1

Vector-Jacobian products $v^T J_h$ can be computed at the same cost as a forward pass through h using reverse mode automatic differentiation.

Trick #2

We can compute a stochastic approximation of the trace using the Hutchinson's Trace Estimator

$$\mathrm{tr}(J_h) = \mathbb{E}_{v \sim p(v)}[v^T J_h v] \qquad \mathbb{E}[v] = 0 \qquad \mathrm{Cov}(v) = I$$

Mila

Different Normalizing Flows at a glance

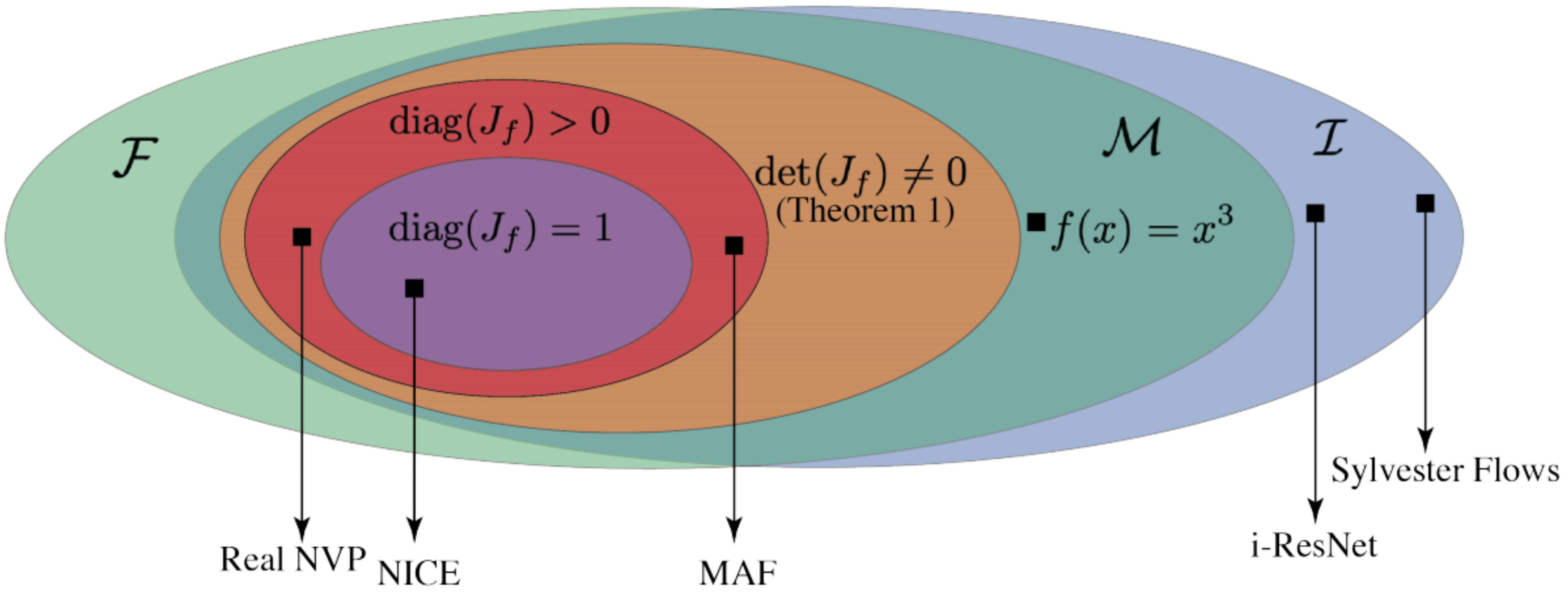


Fig credit: https://arxiv.org/pdf/1907.07945.pdf

Continuous Normalizing Flows

This is the framework that is most amenable to Manifold data.

Neural ODE

Forward Dynamics

$$x_{t+1} \longleftarrow x_t + \alpha h(x_t)$$

This looks an Euler Discretization of an ODE

Reverse Dynamics

$$x_t \longleftarrow x_{t+1} - \alpha h(x_t)$$

When can we invert a ResNet?

What if we take the discretization to be infinitesimal?

$$z_{t_0} = z \qquad z_{t_1} = z$$

$$\frac{dz_t}{dt} = h_t(t, z_t)$$

This is an ODE!

Mila

Continuous Normalizing Flows

Forward Dynamics

$$x = z_{t_1} = z_0 + \int_{t=t_0}^{t_1} h(z_t, t)dt$$

Reverse Dynamics

$$z_0 = x + \int_{t=t_1}^{t_0} h_\phi(z_t, t)dt = x - \int_{t=t_0}^{t_1} h_\phi(z_t, t)dt$$

What if we take the discretization to be infinitesimal?

$$z_{t_0} = z \qquad z_{t_1} = z$$

$$\frac{dz_t}{dt} = h_t(t, z_t)$$

This is an ODE!

Mila

Residual Flows

Forward Dynamics

$$x = z_{t_1} = z_0 + \int_{t=t_0}^{t_1} h(z_t, t)dt$$

Reverse Dynamics

$$z_0 = x + \int_{t=t_1}^{t_0} h_\phi(z_t, t)dt = x - \int_{t=t_0}^{t_1} h_\phi(z_t, t)dt$$

Instantaneous Change of Variable

$$\frac{d \log p(z_t)}{dt} = -\operatorname{Tr}\left(J_{h(t,\cdot)}(z_t)\right)$$

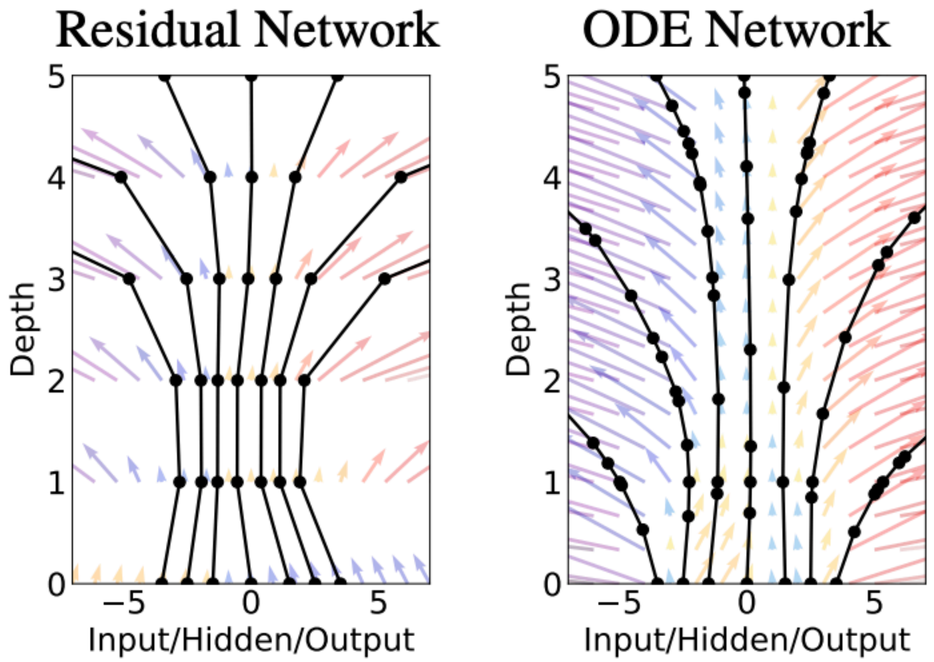


Fig credit: https://arxiv.org/abs/1806.07366

Continuous Normalizing Flows

Instantaneous Change of Variable

$$\frac{d \log p(z_t)}{dt} = -\operatorname{Tr}\left(J_{h(t,\cdot)}(z_t)\right)$$

Change of Variable

$$\log p(x) = \log p(z_0) - \int_{t=t_0}^{t_1} \operatorname{Tr}\left(J_{h_\phi(t,\cdot)}(z_t)\right)$$

Continuous Normalizing Flows

Numerically Solving ODE

- Discrete Solvers that use Euler method e.g. Runge-Kutta
- Adjoint Method

Con’s to using CNF’s

- CNF’s must be globally Lipschitz; this limits their expressive power
- ODE Solvers are slower as solutions must be to a pre-specified tolerance
- Prone to Numerical Errors and Noisy Gradients

Variational Inference with Normalizing Flows

Game Theorists are still thanking Ian Goodfellow et. al 2014 for making their field profitable.

ELBO in VAE

$$\log p(\mathscr{D}) \geq \underbrace{\boxed{\mathbb{E}_{q_\phi(z|x)}[\log p_\psi(x \mid z)]}}_{\text{Reconstruction Error}} - \underbrace{\boxed{D_{KL}(q_\phi(z \mid x) \,||\, p(z))}}_{\text{Regularizer}}$$

Reconstruction Error

Regularizer

Sources for a loose ELBO?

- Amortization cost
- Gaussian Posterior Approximation

Mila

Improving over the Gaussian Posterior Approximation

We can try to replace the Gaussian Posterior with a Flow

$$\mathcal{F}(x) = \mathbb{E}_{q_0(z_0)}[\log q_k(z_j) - \log p(x, z_j)]$$

$$= \mathbb{E}_{q_0(z_0)}\left[\log q_0(z_0) - \sum_{i=1}^{j} \ln \det \left| \frac{\partial f_i}{\partial z_{i-1}} \right| - \log p(x, z_i)\right]$$

$$= D_{KL}(q_0(z_0)\,||\,p(z_j)) - \mathbb{E}_{q_0(z_0)}\left[\sum_{i=1}^{j} \ln \det \left| \frac{\partial f_i}{\partial z_{i-1}} \right| - \log p(x \,|\, z_i)\right]$$