Deep Generative Models with Stick-Breaking Priors

Eric Nalisnick

University of California, Irvine

Outline

1 Overview of Deep Generative Models.

- [ ] Definition
- [ ] Applications
- [ ] Training

Outline

1 Overview of Deep Generative Models.

- [ ] Definition
- [ ] Applications
- [ ] Training

2 The Dirichlet Process.

- [ ] Definition
- [ ] Stochastic Gradient Variational Bayes for the DP

Outline

1. Overview of Deep Generative Models.
   - [ ] Definition
   - [ ] Applications
   - [ ] Training

2. The Dirichlet Process.
   - [ ] Definition
   - [ ] Stochastic Gradient Variational Bayes for the DP

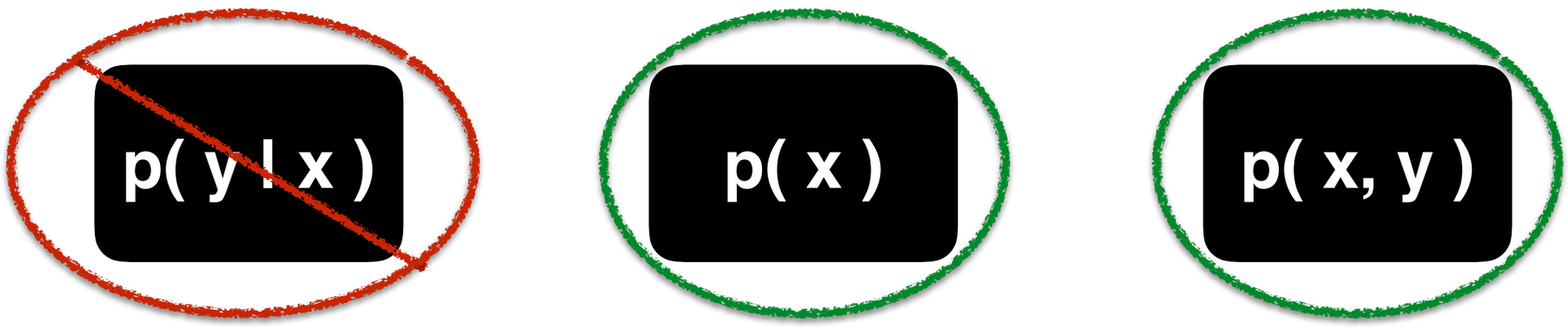
3. Deep Generative Models with Stick-Breaking Priors.
   - [ ] Stick-Breaking Variational Autoencoders
   - [ ] Dirichlet Process Variational Autoencoders
   - [ ] Stick-Breaking for Adaptive Computation

Overview of Deep Generative Models

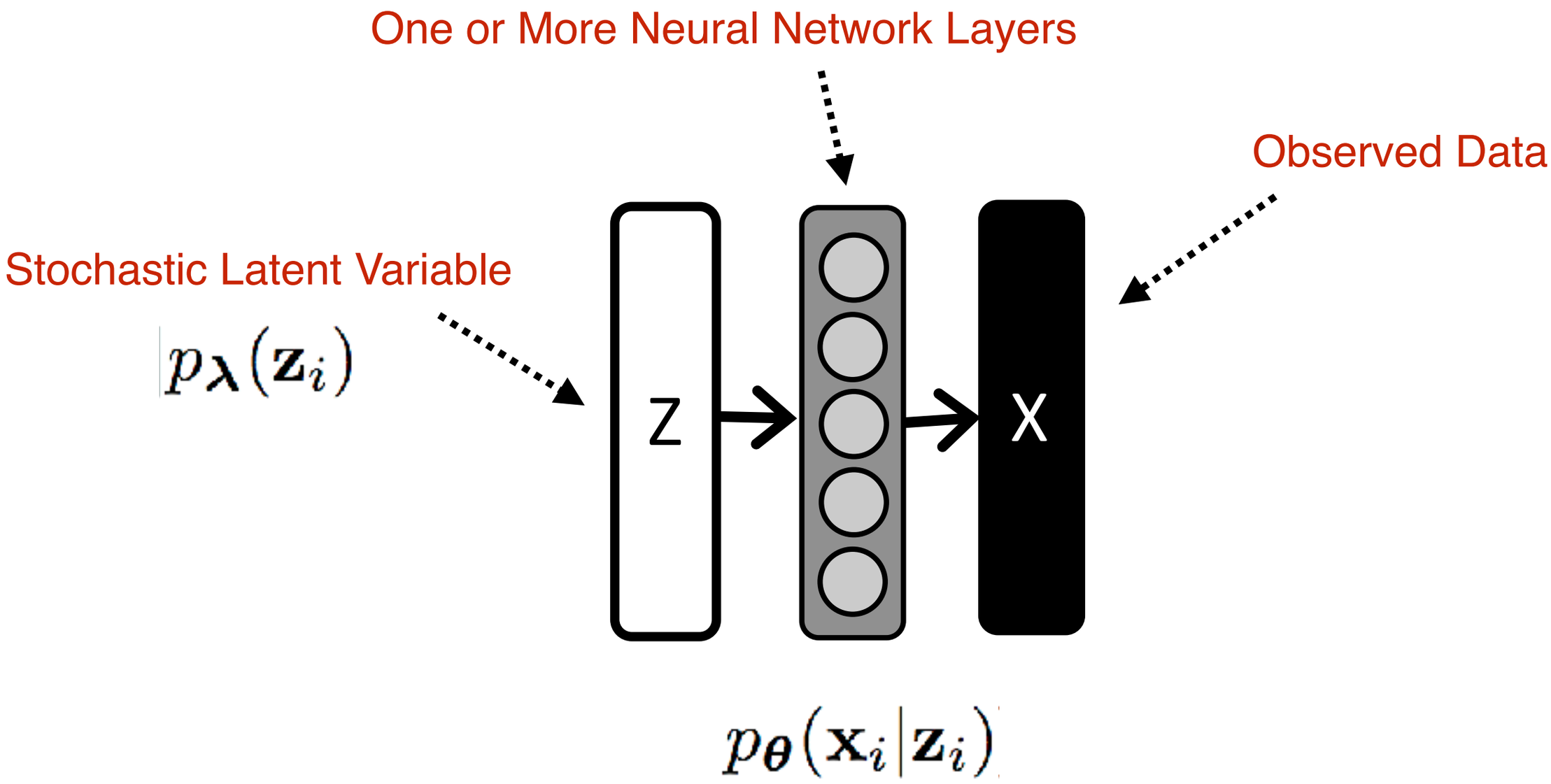
> “Generative modeling…denotes modeling tasks in which a density over *all* the observable quantities is constructed.”

David MacKay

Information Theory, Inference, and Learning Algorithms



Deep Generative Models



Density Network
(MacKay & Gibbs, 1997)

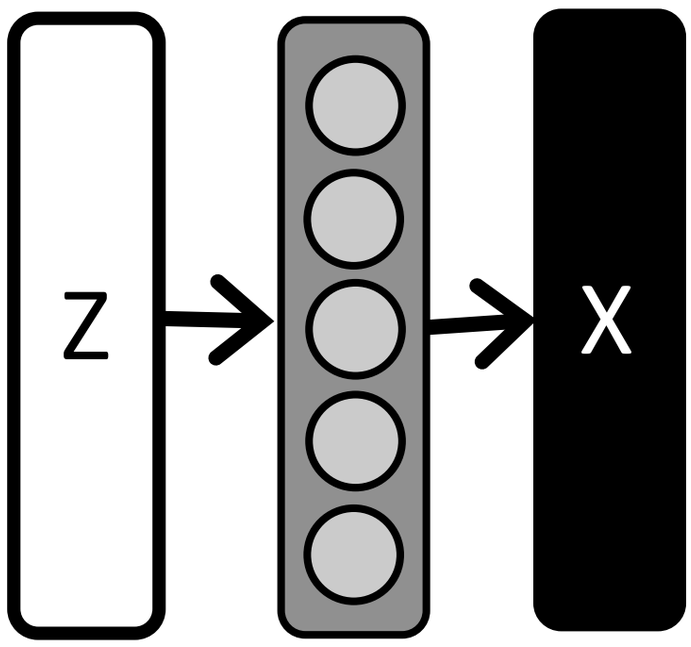
Deep Generative Models

One or More Neural Network Layers

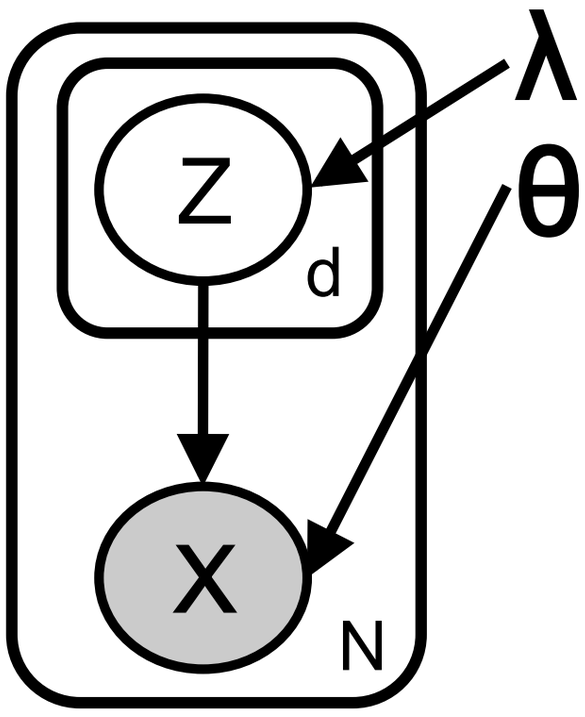
Observed Data

Stochastic Latent Variable

$p_{\boldsymbol{\lambda}}(\mathbf{z}_i)$

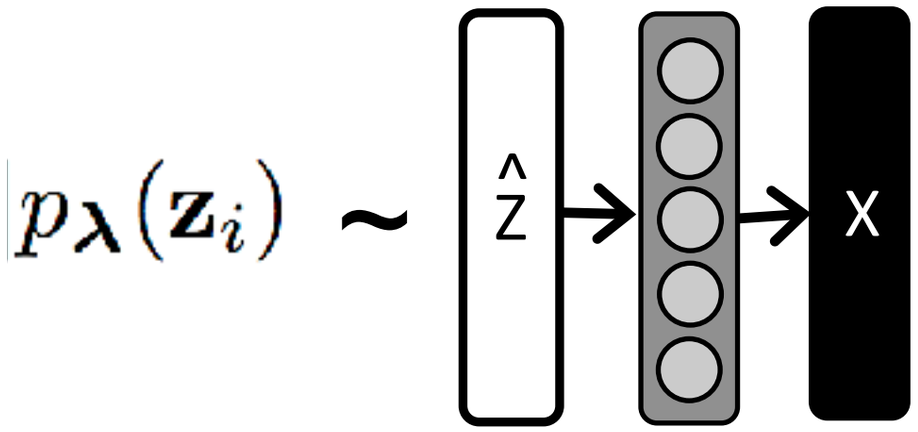


$p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)$



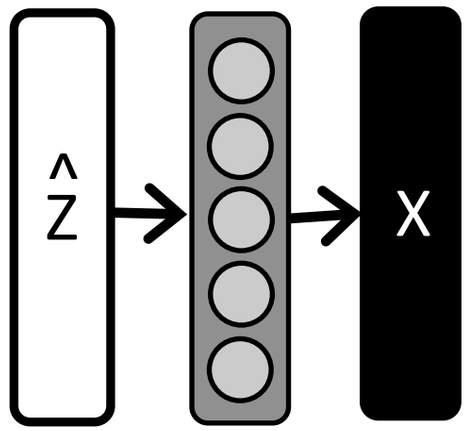
Density Network
(MacKay & Gibbs, 1997)

Sampling from a Deep Generative Model

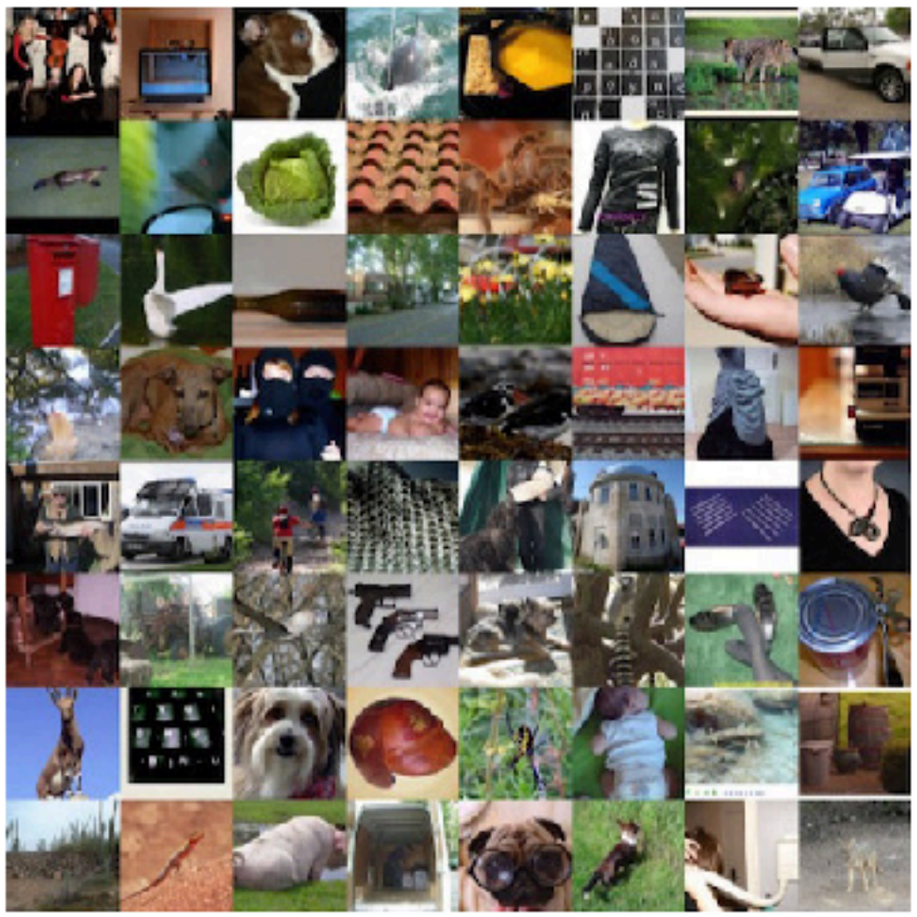


Sampling from a Deep Generative Model

$$p_{\boldsymbol{\lambda}}(\mathbf{z}_i) \sim$$

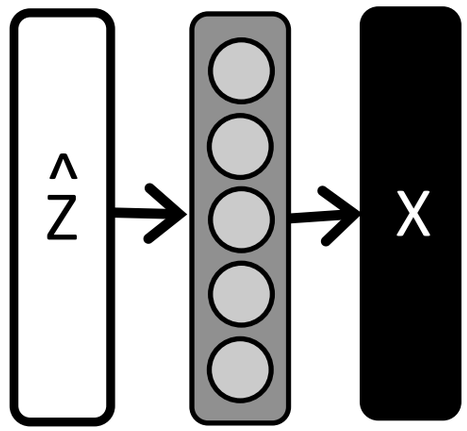


ImageNet

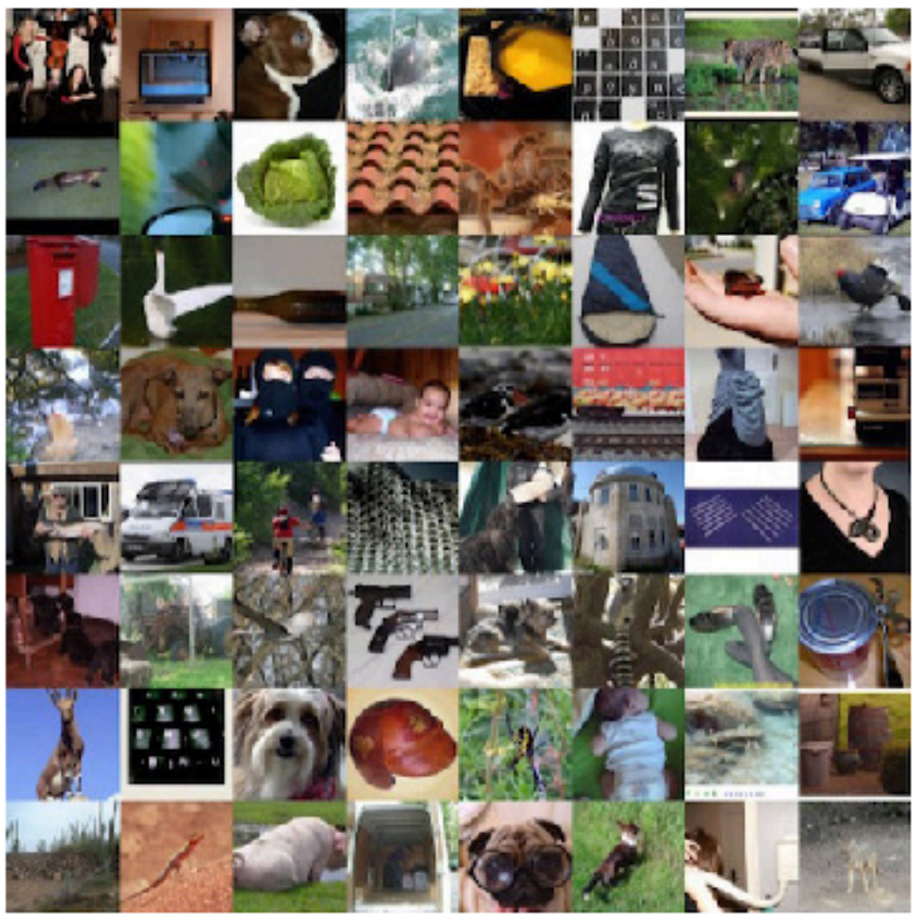


Sampling from a Deep Generative Model

$$p_{\boldsymbol{\lambda}}(\mathbf{z}_i) \sim$$

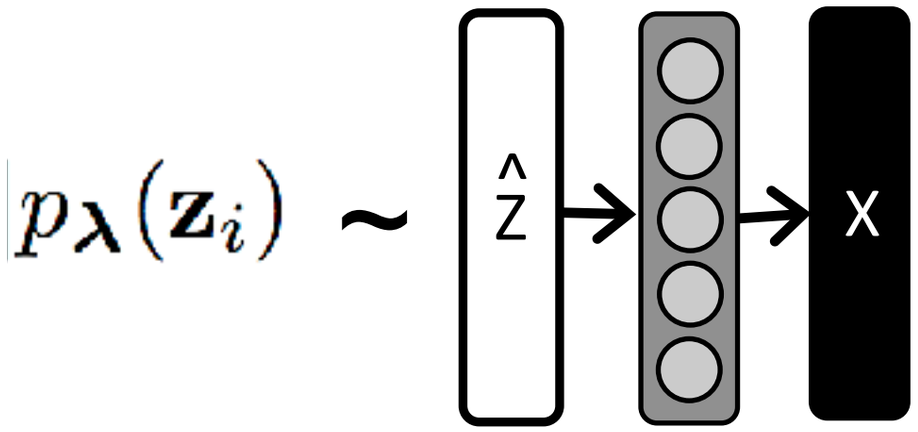


ImageNet



Samples from a DGM

Sampling from a Deep Generative Model



Samples from a DGM

he had been unable to conceal the fact that there was a logical explanation for his inability to alter the fact that they were supposed to be on the other side of the house .
with a variety of pots strewn scattered across the vast expanse of the high ceiling , a vase of colorful flowers adorned the tops of the rose petals littered the floor and littered the floor .
atop the circular dais perched atop the gleaming marble columns began to emerge from atop the stone dais, perched atop the dais .

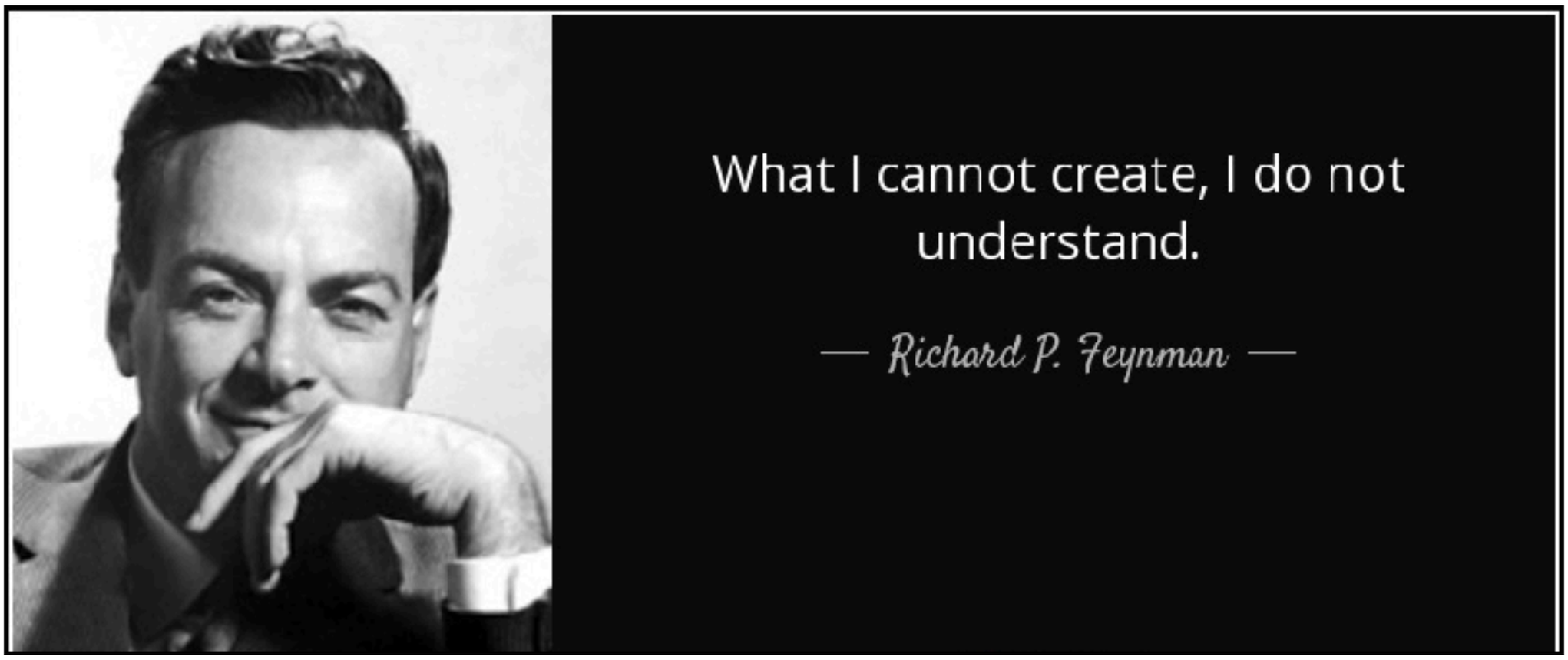
(Bowman et al., 2016)

Overview of Deep Generative Models

1.1 Why Generative Modeling?

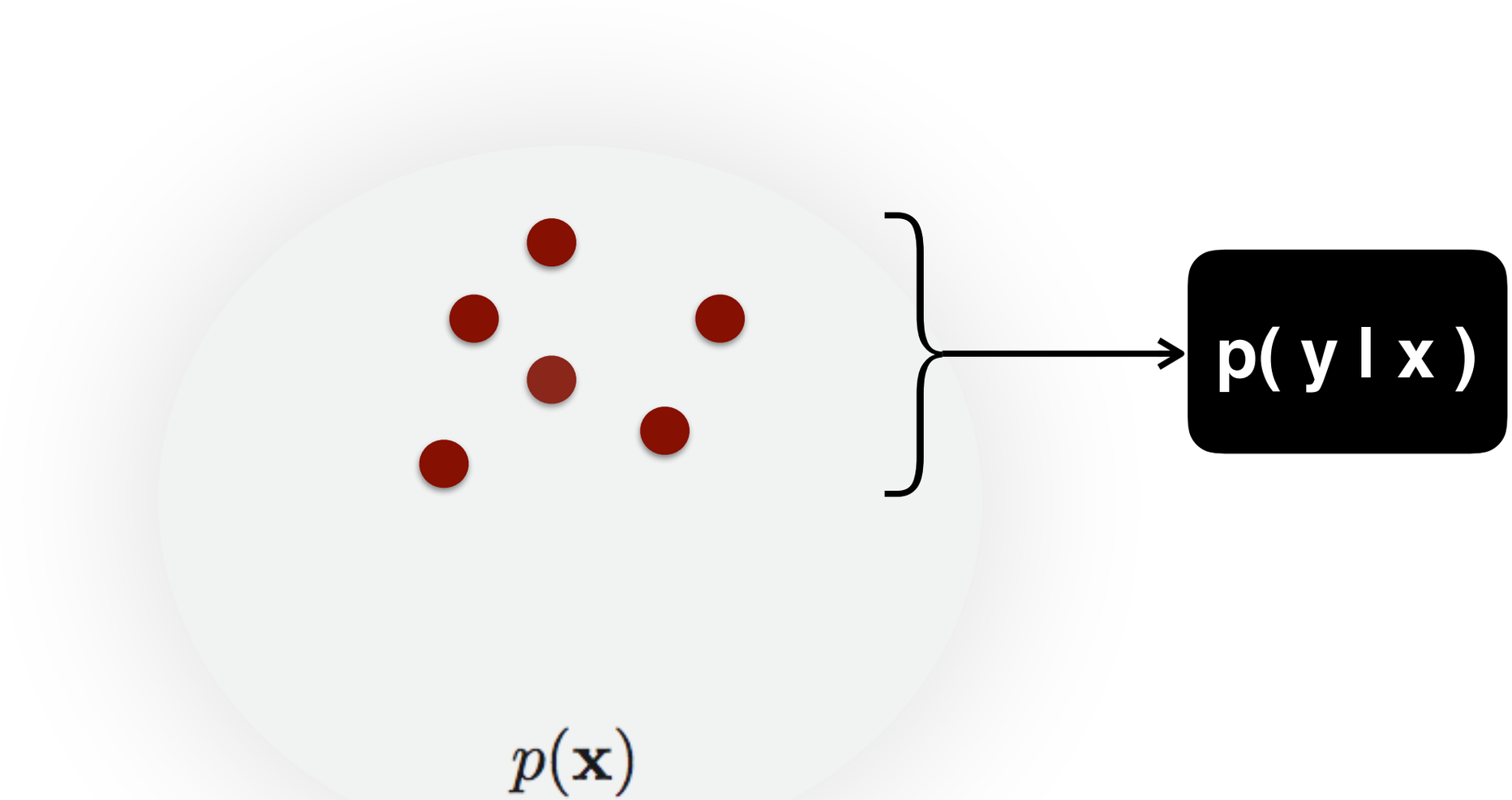
Why Generative Modeling?

1 Cognitive Motivations.



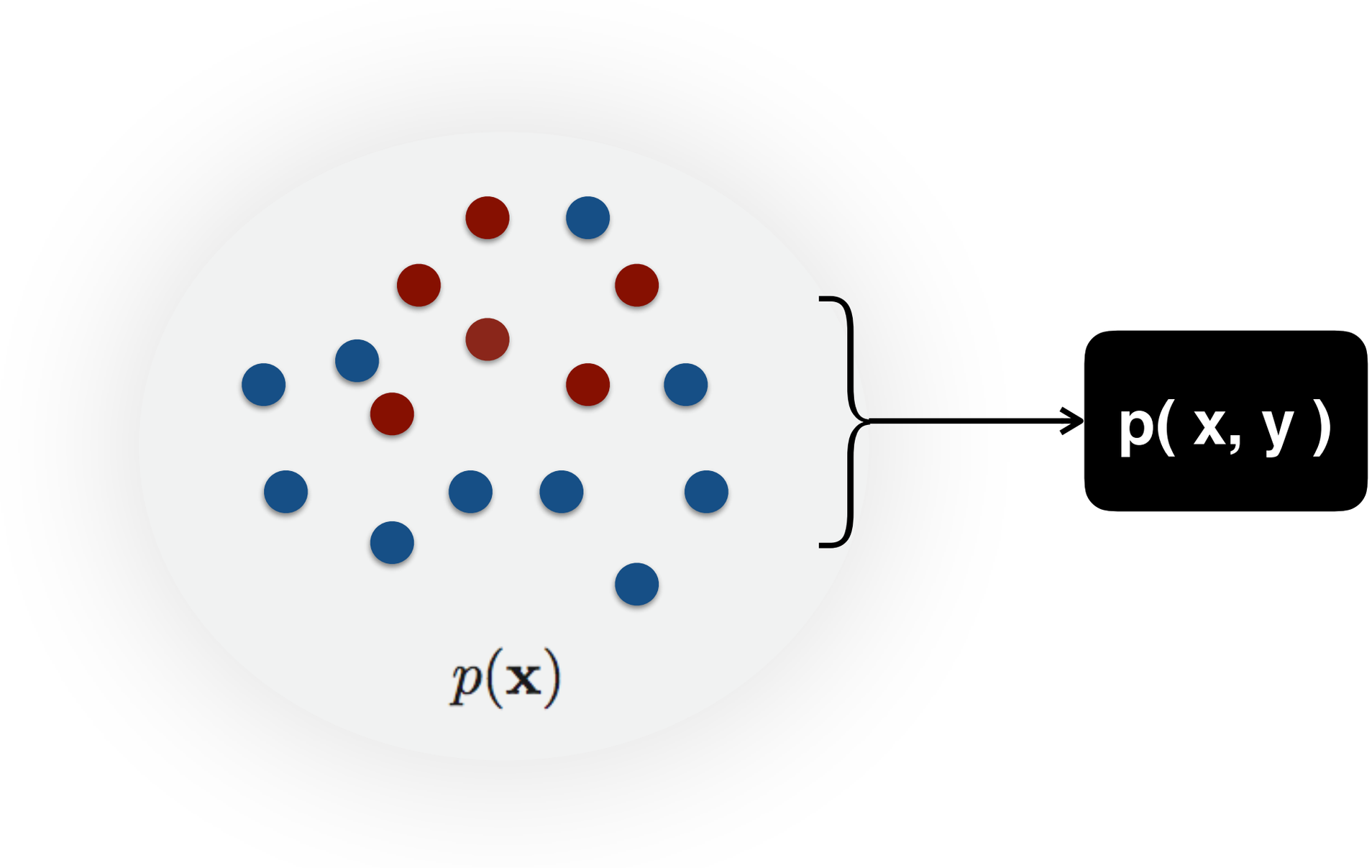
Why Generative Modeling?

2 **Semi-Supervised Learning:** Regularizing discriminative models with unlabeled data.



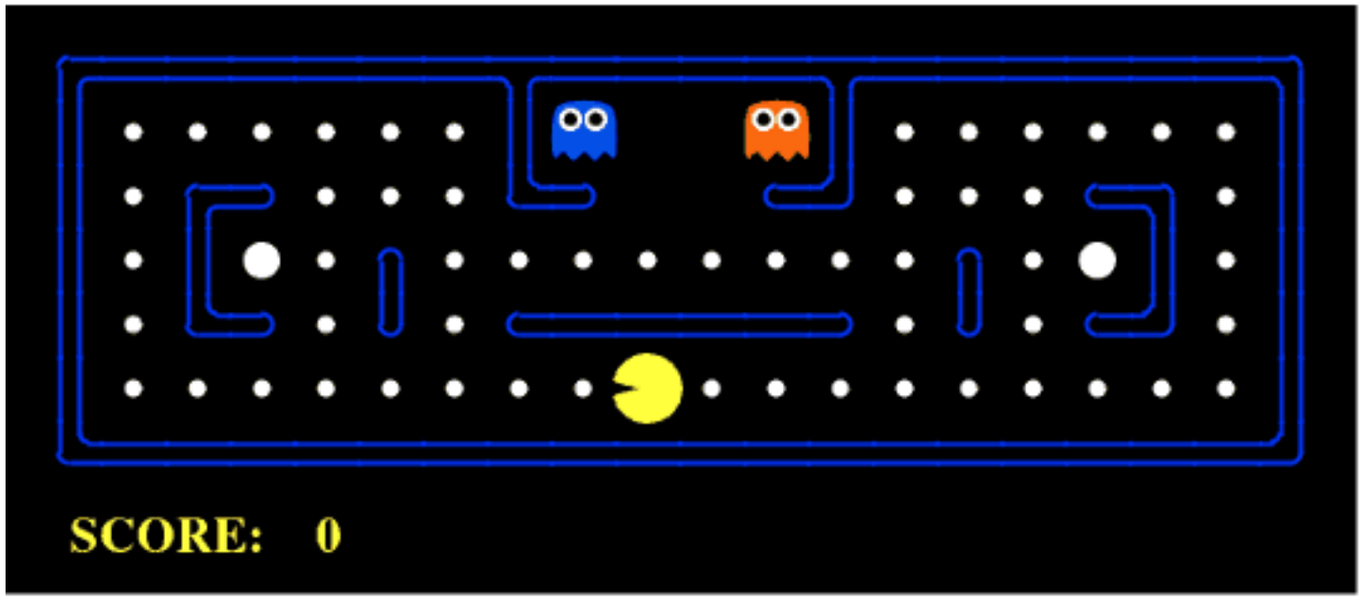
Why Generative Modeling?

2 **Semi-Supervised Learning:** Regularizing discriminative models with unlabeled data.



Why Generative Modeling?

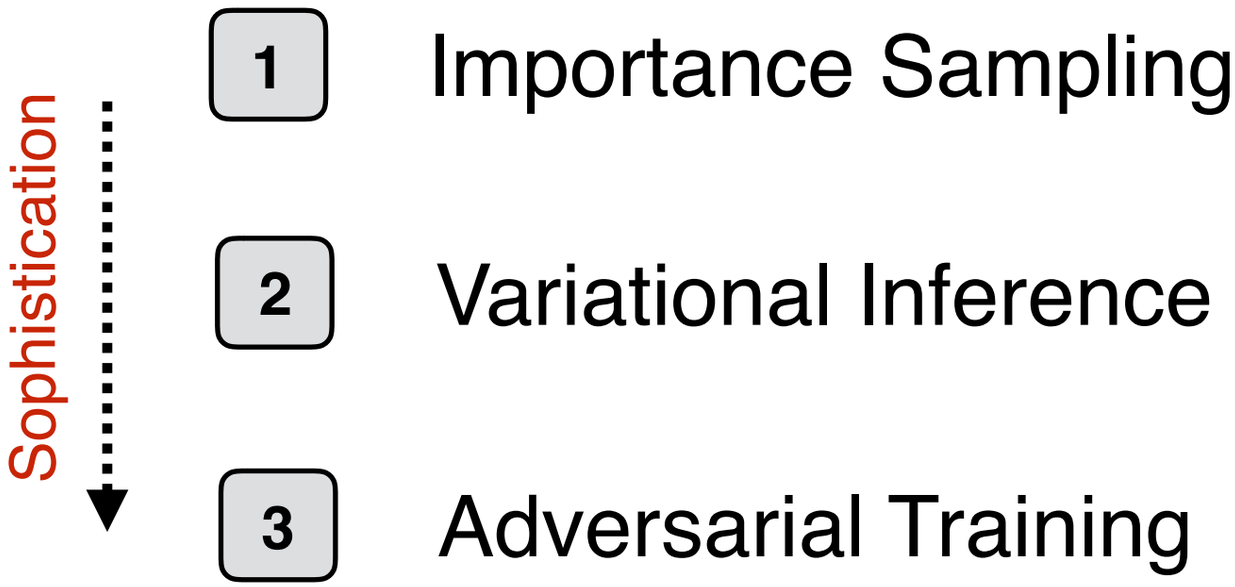
3 Simulating Environments for Reinforcement Learning.



Overview of Deep Generative Models

1.2 How to Train a Deep Generative Model

Learning a Deep Generative Model



Learning a Deep Generative Model

1 Importance Sampling

(MacKay & Gibbs, 1997)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

Learning a Deep Generative Model

1 Importance Sampling

(MacKay & Gibbs, 1997)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$\approx \log \frac{1}{R} \sum_{r}^{R} \exp\{\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,r})\}$$

where $\quad \hat{\mathbf{z}}_{i,r} \sim p_{\boldsymbol{\lambda}}(\mathbf{z}_i)$

Learning a Deep Generative Model

1 Importance Sampling

(MacKay & Gibbs, 1997)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$\approx \log \frac{1}{R} \sum_{r}^{R} \exp\{\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,r})\}$$

$$\text{where} \quad \hat{\mathbf{z}}_{i,r} \sim p_{\boldsymbol{\lambda}}(\mathbf{z}_i)$$

Works only in low dimensions

Learning a Deep Generative Model

2 Variational Inference

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

Learning a Deep Generative Model

2 Variational Inference

Posterior Approximation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$= \log \int_{\mathbf{z}} \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

Learning a Deep Generative Model

2 Variational Inference

Posterior Approximation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$= \log \int_{\mathbf{z}} \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$\geq \int_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} d\mathbf{z}$$

Learning a Deep Generative Model

2 Variational Inference

Posterior Approximation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$= \log \int_{\mathbf{z}} \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$\geq \int_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} d\mathbf{z}$$

$$= \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

Data Reconstruction

Regularization

Learning a Deep Generative Model

2 Variational Inference

Posterior Approximation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) = \log \int_{\mathbf{z}} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$= \log \int_{\mathbf{z}} \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i) d\mathbf{z}$$

$$\geq \int_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)\, p_{\boldsymbol{\lambda}}(\mathbf{z}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)} d\mathbf{z}$$

$$= \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

Data Reconstruction

Regularization

Problem: For neural network likelihoods, these integrals are analytically intractable

Learning a Deep Generative Model

2 Variational Inference

Step #1: Use Monte Carlo Approximations

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

Learning a Deep Generative Model

2 Variational Inference

Step #1: Use Monte Carlo Approximations

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

Problem: Still can't compute gradient w.r.t. variational parameters

$$\frac{\partial}{\partial \boldsymbol{\phi}} \sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) = \sum_{s=1}^{S} \frac{\partial}{\partial \hat{\mathbf{z}}_{i,s}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) \frac{\partial \hat{\mathbf{z}}_{i,s}}{\partial \boldsymbol{\phi}}.$$

Learning a Deep Generative Model

2 Variational Inference

Step #1: Use Monte Carlo Approximations

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)] - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

Problem: Still can't compute gradient w.r.t. variational parameters

$$\frac{\partial}{\partial \boldsymbol{\phi}} \sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) = \sum_{s=1}^{S} \frac{\partial}{\partial \hat{\mathbf{z}}_{i,s}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) \frac{\partial \hat{\mathbf{z}}_{i,s}}{\partial \boldsymbol{\phi}}.$$

Step #2: Sample via Differentiable Non-Centered Parametrizations

$$q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) = \mathrm{N}(\mathbf{z}_i; \boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^2_{\boldsymbol{\phi}})$$

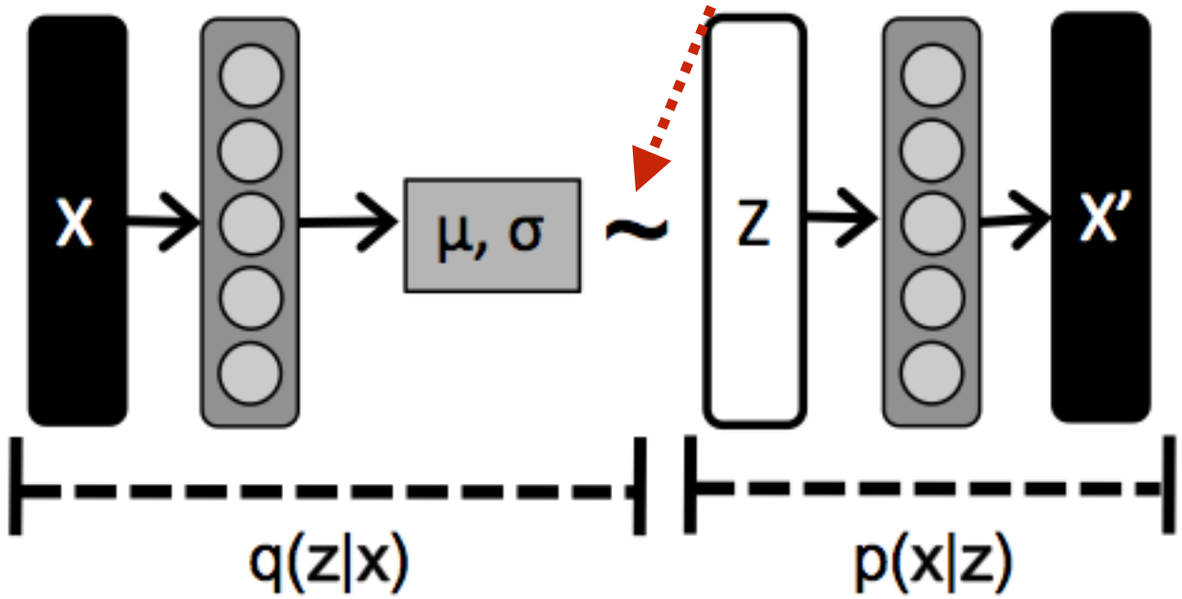
$$\hat{\mathbf{z}}_{i,s} = \boldsymbol{\mu}_{\boldsymbol{\phi}} + \boldsymbol{\sigma}_{\boldsymbol{\phi}} \odot \hat{\boldsymbol{\epsilon}}_s \qquad \hat{\boldsymbol{\epsilon}}_s \sim \mathrm{N}(\mathbf{0}, \mathbb{I})$$

The Variational Autoencoder (VAE)

(Kingma & Welling, 2014), (Rezende et al., 2014)

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\hat{\mathbf{z}}_{i,s}) - KLD[q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\mathbf{z}_i)]$$

$$\text{where } \hat{\mathbf{z}}_{i,s} = \boldsymbol{\mu}_{\boldsymbol{\phi}} + \boldsymbol{\sigma}_{\boldsymbol{\phi}} \odot \hat{\boldsymbol{\epsilon}}_s \qquad \hat{\boldsymbol{\epsilon}}_s \sim \mathbf{N}(\mathbf{0}, \mathbb{I})$$



Learning a Deep Generative Model

3 Adversarial Training

(Goodfellow et al., 2014)

$$\mathcal{J}_{\text{Adver.}} = -y_j \log f_\psi(\hat{\mathbf{x}}_j) - (1 - y_j) \log(1 - f_\psi(\hat{\mathbf{x}}_j))$$

Classifier predicting if **x** is generated or observed data

$$y_j = \begin{cases} 1 & \text{if } \hat{\mathbf{x}}_j \sim p^*(\mathbf{x}) \\ 0 & \text{if } \hat{\mathbf{x}}_j \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \end{cases}$$

Learning a Deep Generative Model

3 Adversarial Training

(Goodfellow et al., 2014)

$$\mathcal{J}_{\text{Adver.}} = -y_j \log f_{\psi}(\hat{\mathbf{x}}_j) - (1 - y_j) \log(1 - f_{\psi}(\hat{\mathbf{x}}_j))$$

Classifier predicting if **x** is generated or observed data

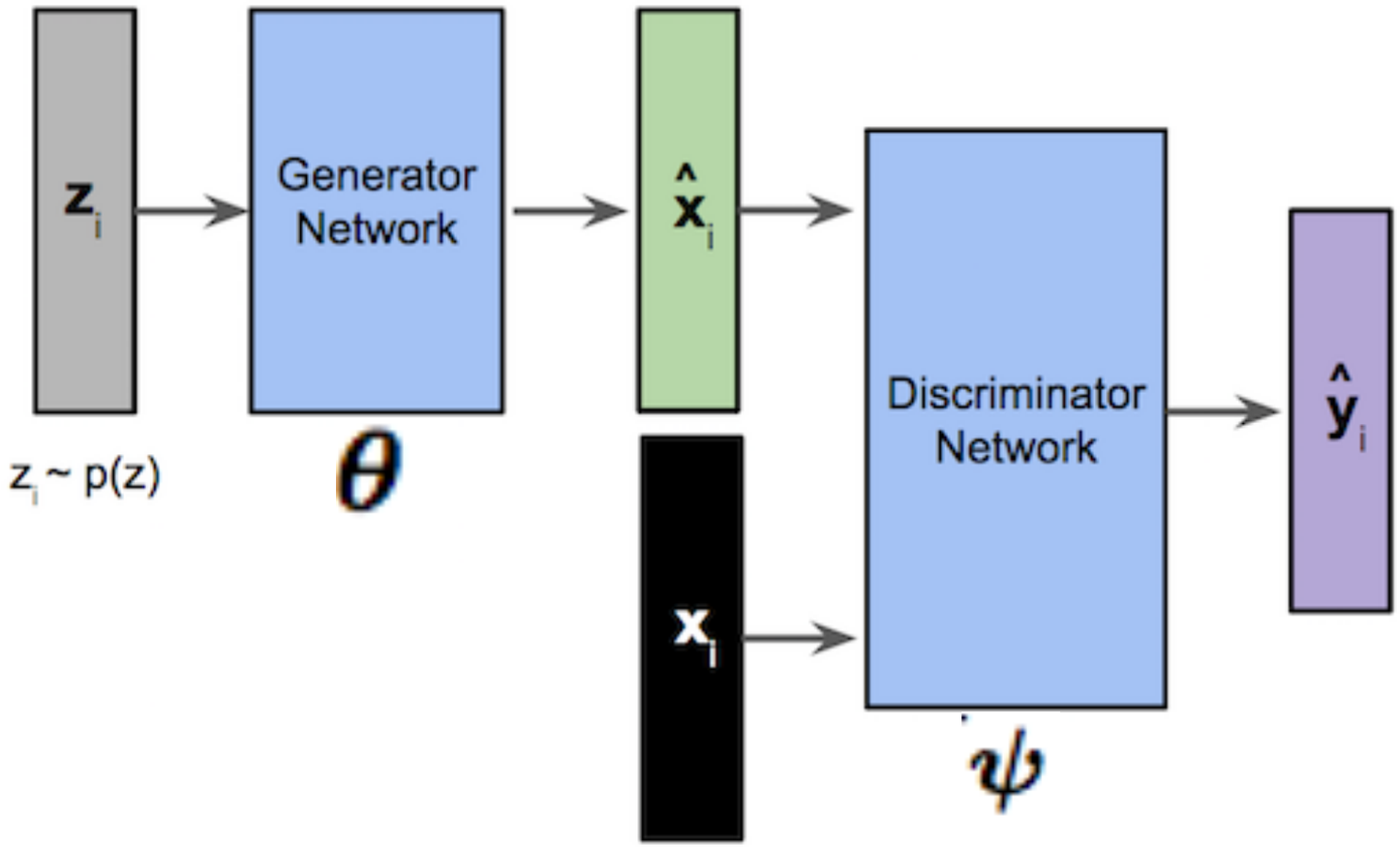
$$y_j = \begin{cases} 1 & \text{if } \hat{\mathbf{x}}_j \sim p^*(\mathbf{x}) \\ 0 & \text{if } \hat{\mathbf{x}}_j \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \end{cases}$$

Two step training:

1. Minimize $\mathcal{J}_{\text{Adver.}}$ with respect to $\boldsymbol{\psi}$
2. Maximize $\mathcal{J}_{\text{Adver.}}$ with respect to $\boldsymbol{\theta}$

Generative Adversarial Networks (GANs)

(Goodfellow et al., 2014)

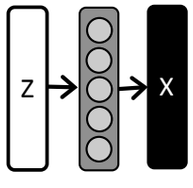
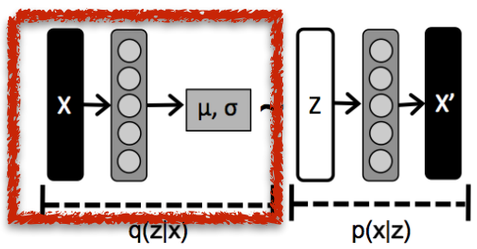
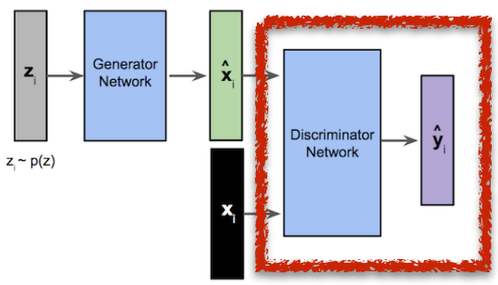


Variational Bayes vs Adversarial Training

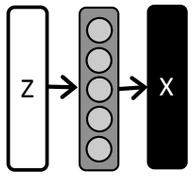
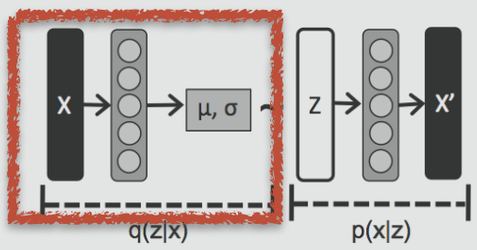
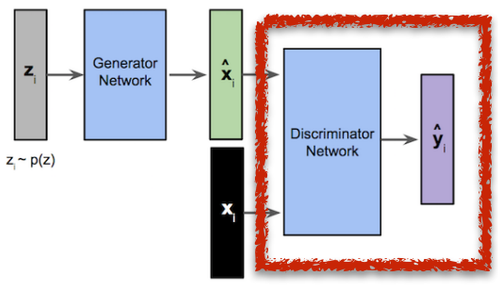
VAE

GAN

Deep Generative Models Summary

	Training Method	
 (Original) Density Network	Importance Sampled Estimate of Marginal Likelihood	No auxiliary model
 Variational Autoencoder	Stochastic Gradient Variational Bayes	Backend auxiliary model
 Generative Adversarial Network	Adversarial Training	Frontend auxiliary model

Deep Generative Models Summary

	Training Method	
 (Original) Density Network	Importance Sampled Estimate of Marginal Likelihood	No auxiliary model
 Variational Autoencoder	Stochastic Gradient Variational Bayes	Backend auxiliary model
 Generative Adversarial Network	Adversarial Training	Frontend auxiliary model

The Dirichlet Process

The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$$

The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$$

1 $\zeta_k \sim G_0$

The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$$

1 $\zeta_k \sim G_0$

2 $\boldsymbol{\pi} \sim \mathrm{GEM}(\alpha_0)$

$$v_k \sim \mathrm{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$$

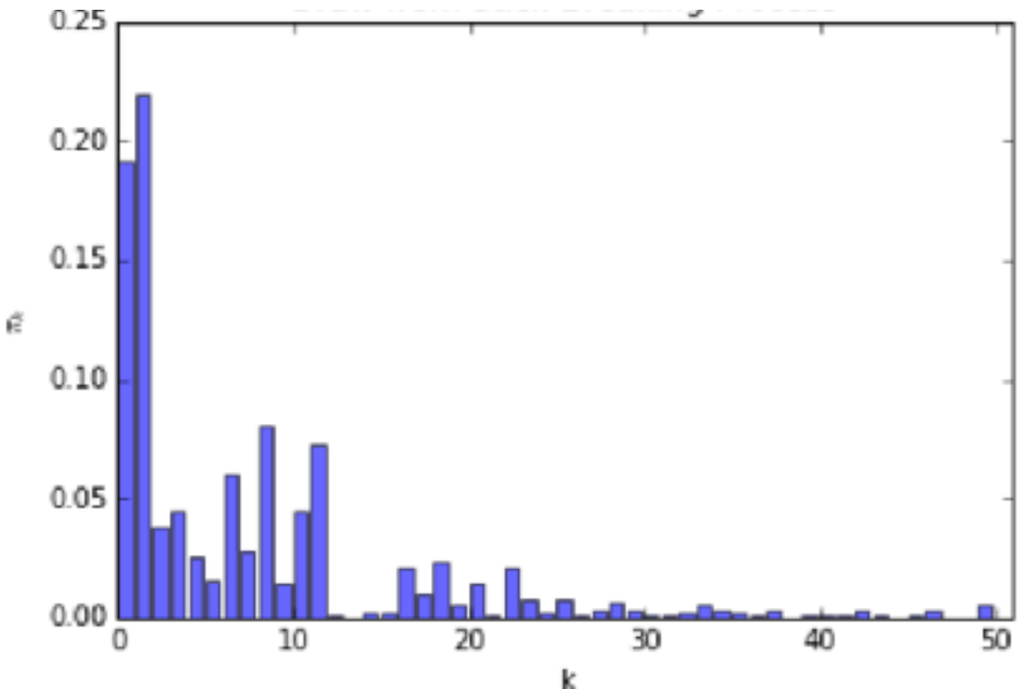
1 $\zeta_k \sim G_0$

2 $\boldsymbol{\pi} \sim \mathrm{GEM}(\alpha_0)$

$$v_k \sim \mathrm{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

Draw from GEM distribution



The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\zeta_k}$$

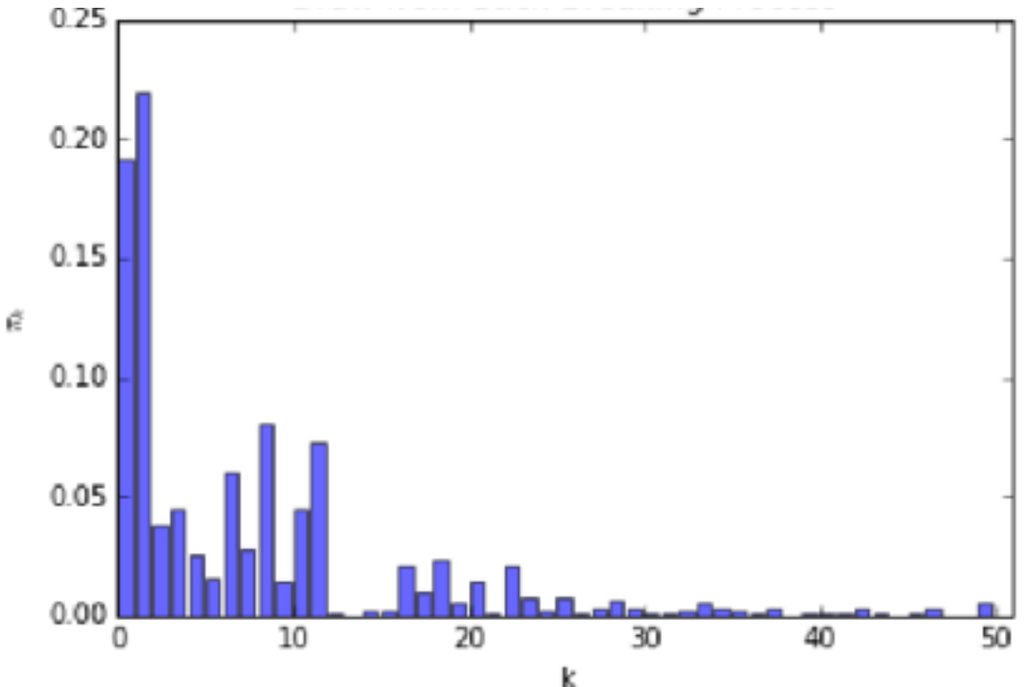
1 $\zeta_k \sim G_0$

2 $\pi \sim \text{GEM}(\alpha_0)$

$$v_k \sim \text{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

Draw from GEM distribution



The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{T} \pi_k \delta_{\zeta_k}$$

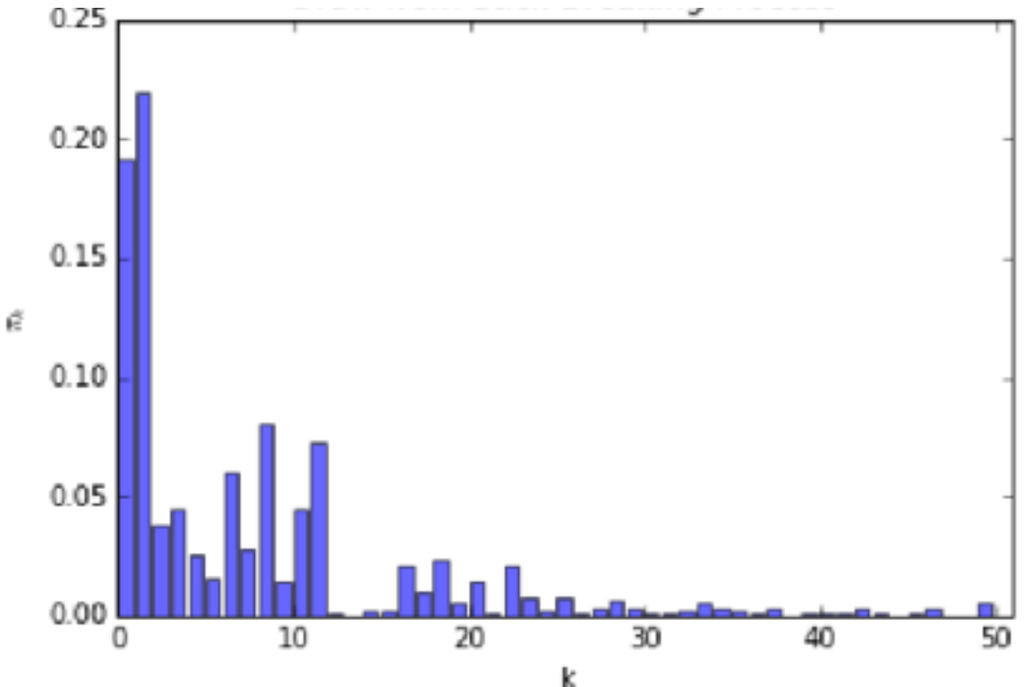
1 $\zeta_k \sim G_0$

2 $\pi \sim \text{GEM}(\alpha_0)$

$$v_k \sim \text{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

Draw from GEM distribution



The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{T} \pi_k \mathbf{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

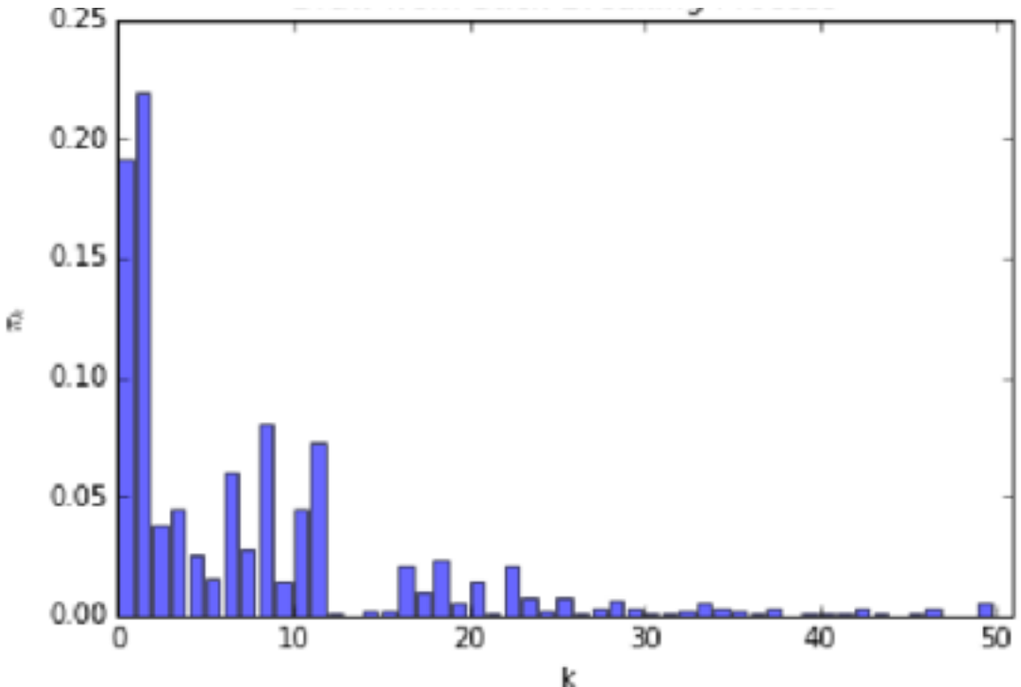
1 $\zeta_k \sim G_0$

2 $\boldsymbol{\pi} \sim \mathbf{GEM}(\alpha_0)$

$$v_k \sim \mathbf{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

Draw from GEM distribution



The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{T} \pi_k \mathbf{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

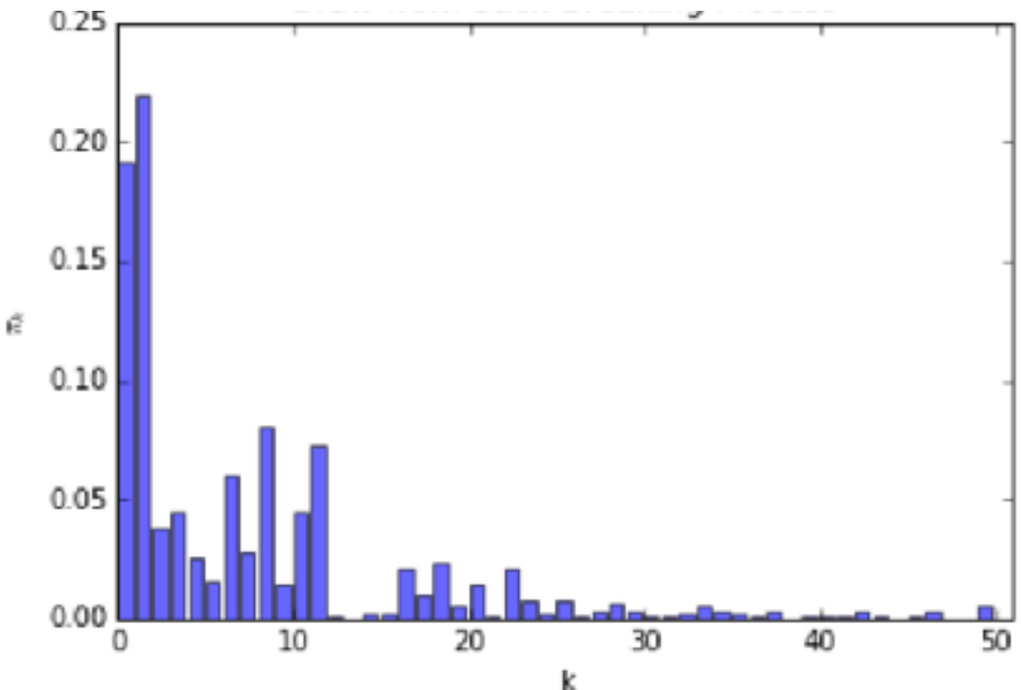
1 $\zeta_k \sim G_0$

2 $\boldsymbol{\pi} \sim \text{Dirichlet}(\alpha_0)$

$$v_k \sim \mathbf{Beta}(1, \alpha_0)$$

$$\pi_k = \begin{cases} v_1 \text{ if } k = 1 \\ v_k \prod_{j<k}(1 - v_j) \text{ for } k > 1 \end{cases}$$

Draw from GEM distribution



SGVB for The Dirichlet Process

$$G(\cdot) = \sum_{k=1}^{T} \pi_k \mathrm{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Two Obstacles:

1. Gradients through $\boldsymbol{\pi}_k$
2. Gradients thought samples from the mixture $G(\cdot)$

Differentiating Through π_k

Obstacle: The Beta distribution does not have a non-centered parametrization (except in special cases)

Differentiating Through π_k

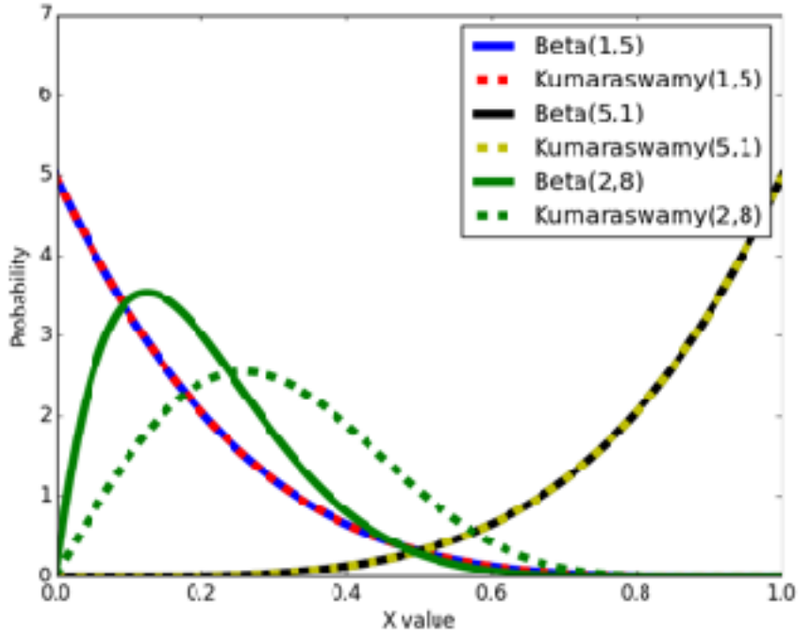
Obstacle: The Beta distribution does not have a non-centered parametrization (except in special cases)

Poondi Kumaraswamy
(1930-1988)

Kumaraswamy Distribution: A Beta-like distribution with a closed-form inverse CDF. Use as variational posterior.

$$\text{Kumaraswamy}(x; a, b) = abx^{a-1}(1-x^a)^{b-1}$$

$$x \sim (1-u^{\frac{1}{b}})^{\frac{1}{a}} \text{ where } u \sim \text{Uniform}(0,1)$$



Stochastic Backpropagation through Mixtures

Obstacle: Not obvious how to use the *reparametrization trick* for samples from a mixture distribution.

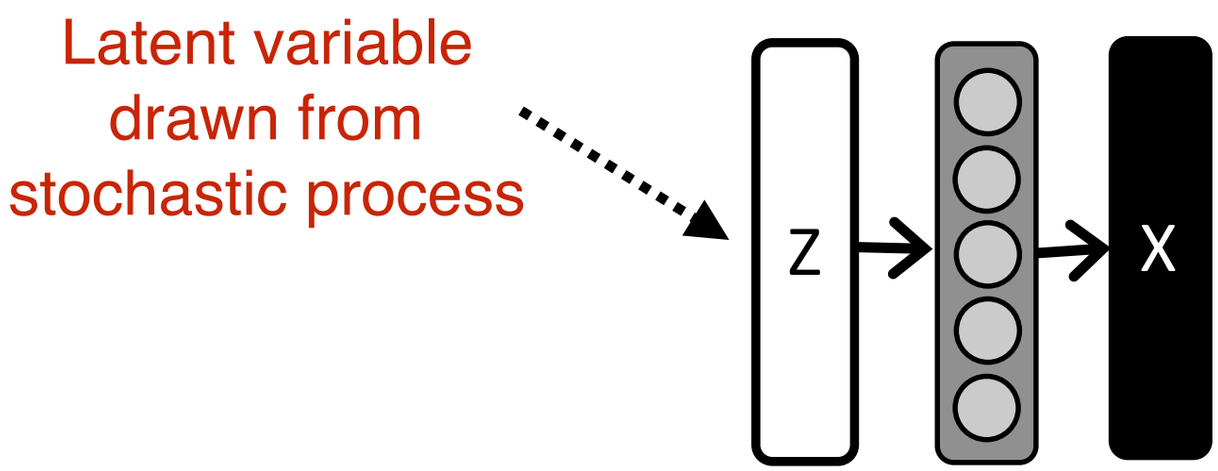
Stochastic Backpropagation through Mixtures

Obstacle: Not obvious how to use the *reparametrization trick* for samples from a mixture distribution.

Brute Force Solution: Summing over all components results in a tractable ELBO but requires O(T) decoder propagations.

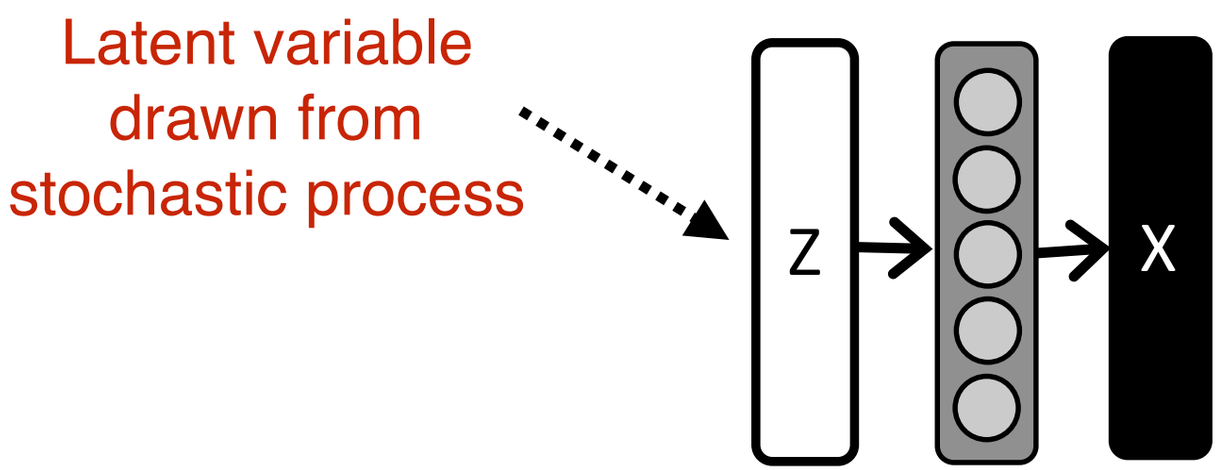
Deep Generative Models with Stick-Breaking Priors

Why Combine the DP with DGMs?



Why Combine the DP with DGMs?

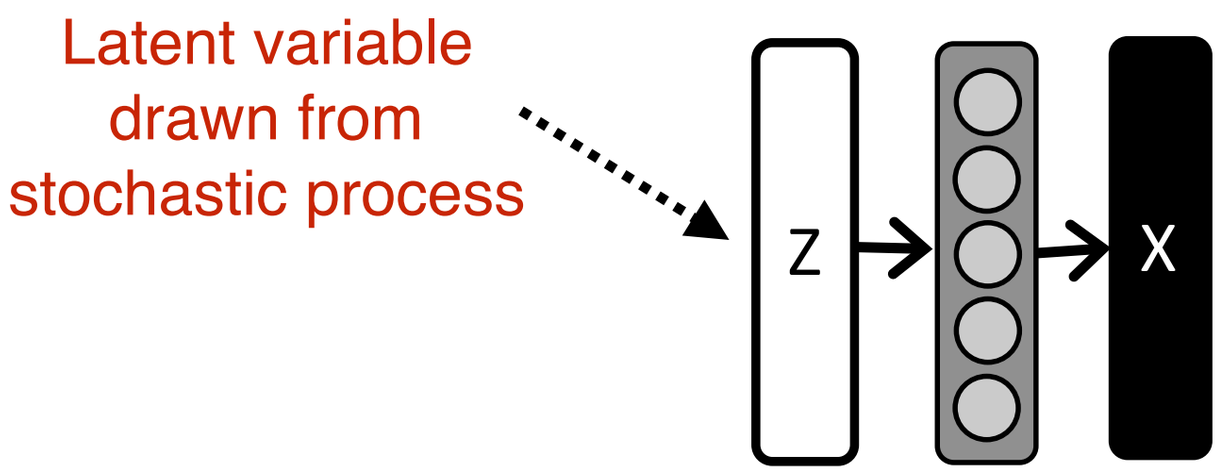
1 Dynamic Latent Representations (Width)



Why Combine the DP with DGMs?

1. Dynamic Latent Representations (Width)

2. Neural Networks with Adaptive Computation (Depth)

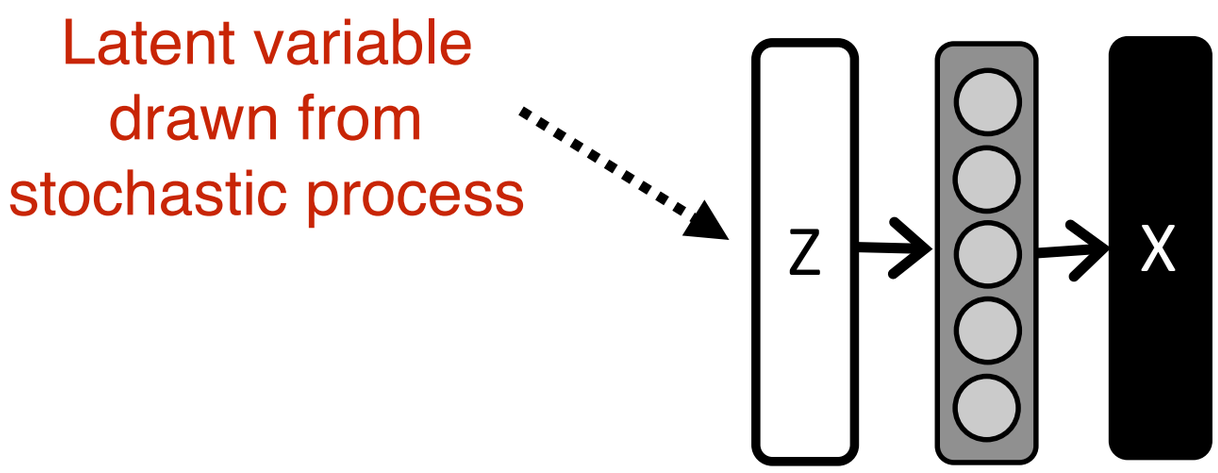


Why Combine the DP with DGMs?

1. Dynamic Latent Representations (Width)

2. Neural Networks with Adaptive Computation (Depth)

3. Automatic and Principled Model Selection (Based on Marginal Likelihood)



Deep Generative Models with Stick-Breaking Priors

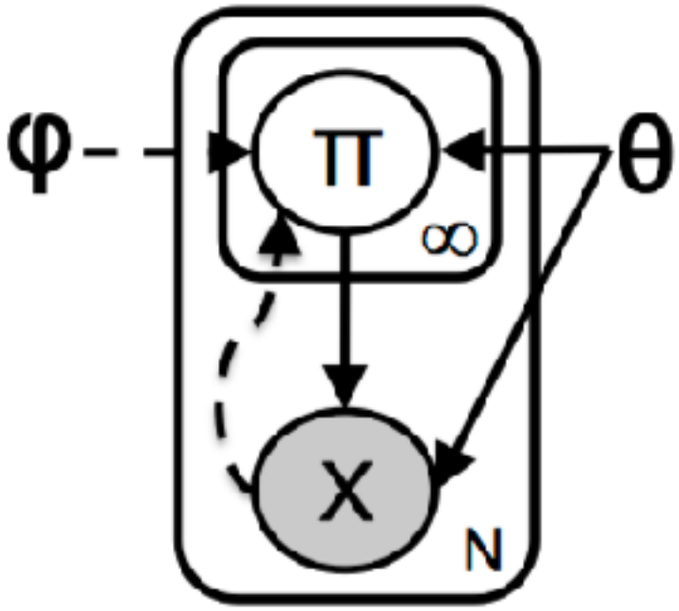
3.1 The Stick-Breaking Variational Autoencoder

In collaboration with

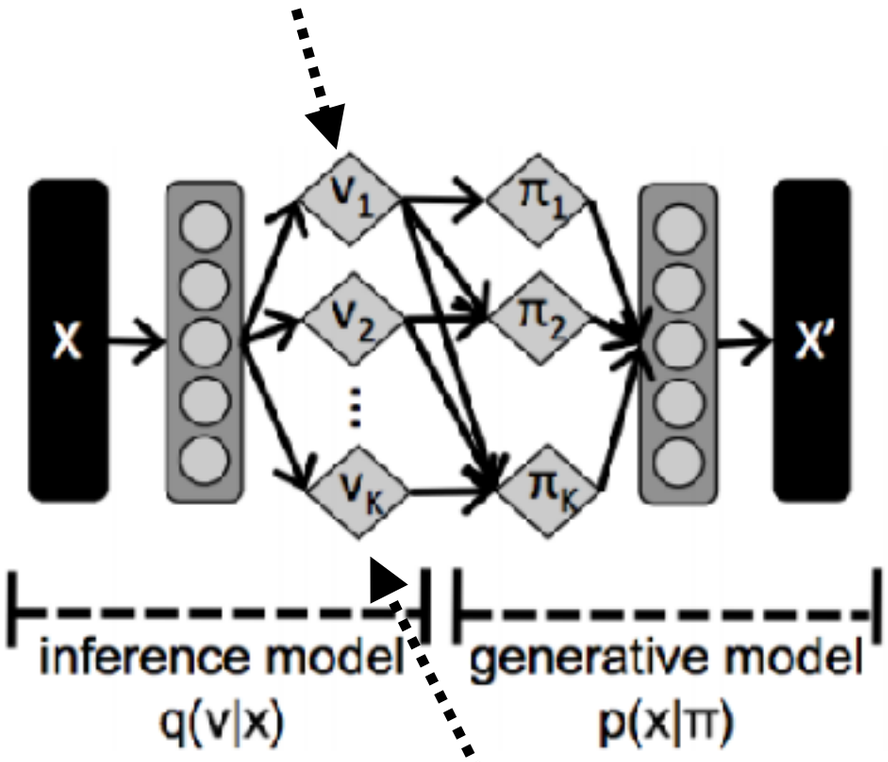
Padhraic Smyth

MODEL #1: Stick-Breaking Variational Autoencoder

(Nalisnick & Smyth, 2017)



Kumaraswamy Samples

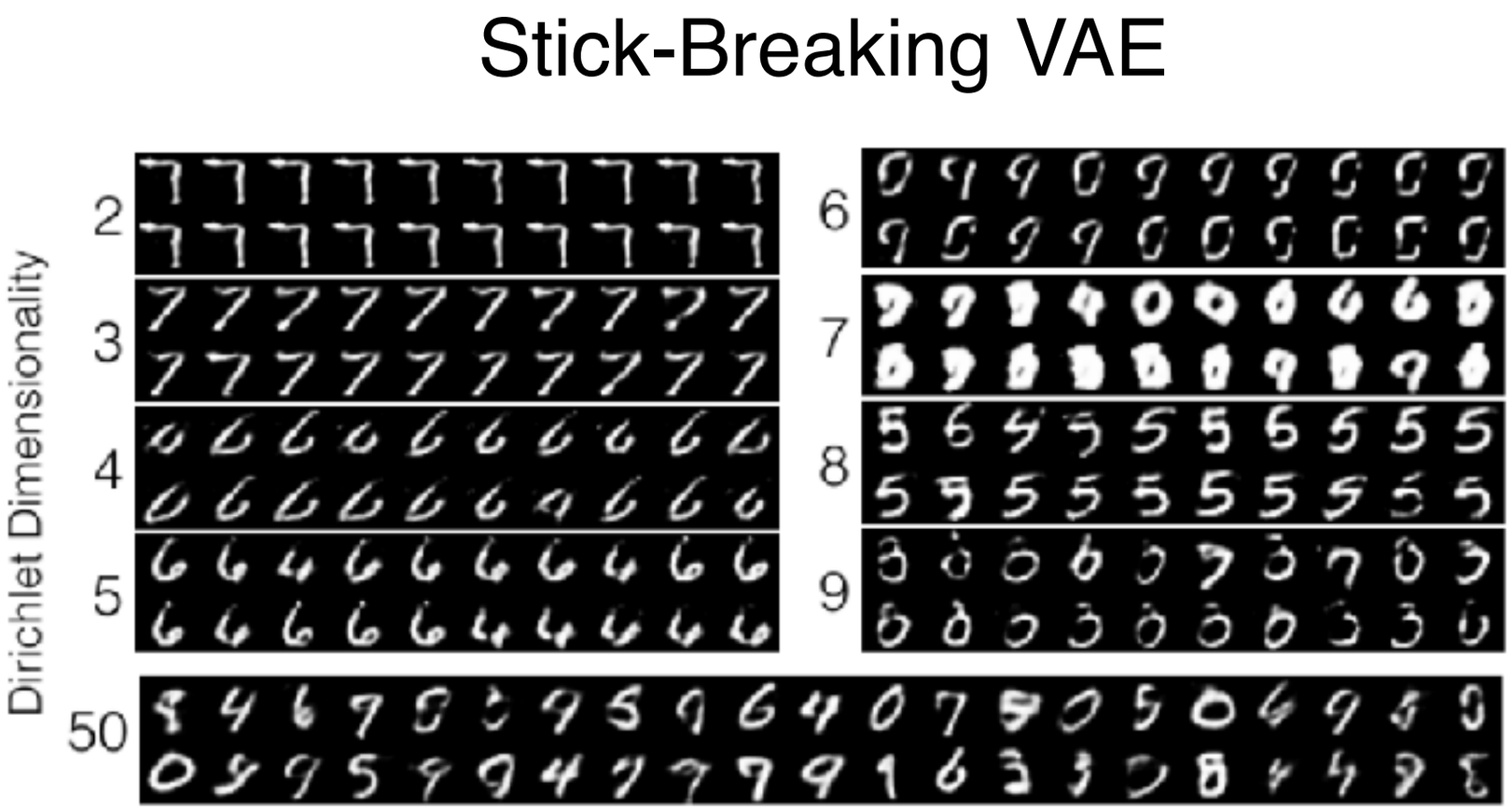


Truncated posterior;
not necessary but learns faster

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i) = \frac{1}{S} \sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i | \boldsymbol{\pi}_{i,s}) - KL(q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i | \mathbf{x}_i) || p(\boldsymbol{\pi}_i; \boldsymbol{\alpha}_0))$$

Samples from Generative Model

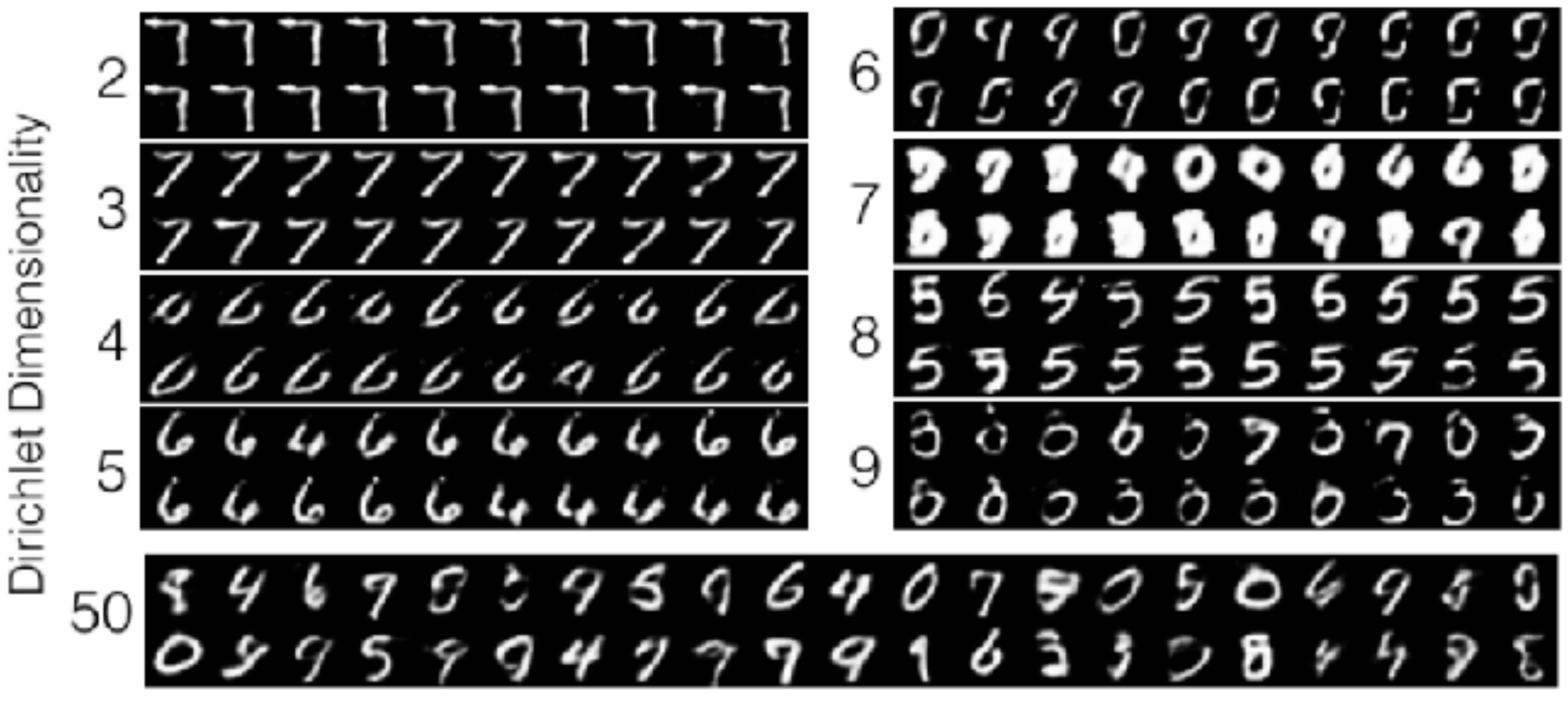
(Nalisnick & Smyth, 2017)



Samples from Generative Model

(Nalisnick & Smyth, 2017)

Stick-Breaking VAE



Truncation level of 50 dimensions, Beta(1,5) Prior

Gaussian VAE

50 dimensions, N(0,1) Prior

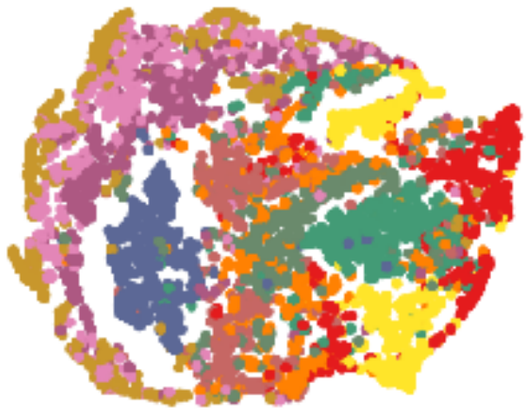
Quantitative Results for SB-VAE

(Nalisnick & Smyth, 2017)

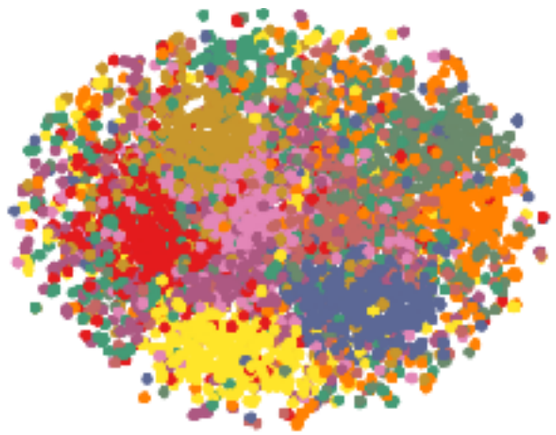
	k=3	k=5	k=10
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33
Raw Pixels	2.95	3.12	3.35

MNIST: kNN Classifier on Latent Space

Unsupervised



MNIST: Dirichlet
Latent Space (t-SNE)



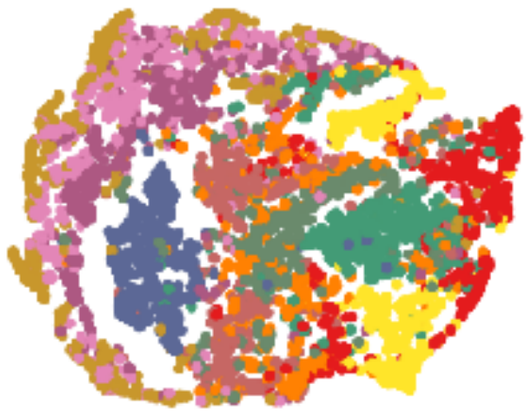
MNIST: Gaussian
Latent Space (t-SNE)

Semi-Supervised

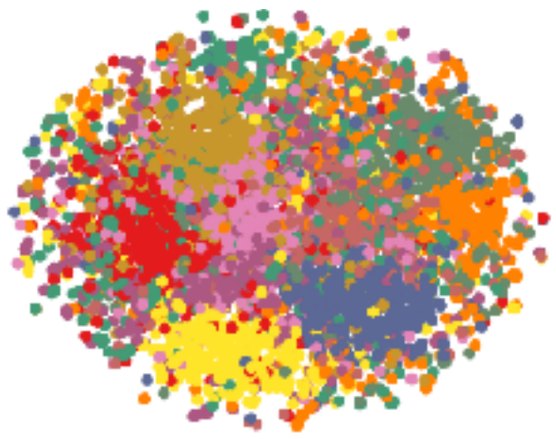
Quantitative Results for SB-VAE

(Nalisnick & Smyth, 2017)

Unsupervised



MNIST: Dirichlet
Latent Space (t-SNE)



MNIST: Gaussian
Latent Space (t-SNE)

	k=3	k=5	k=10
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33
Raw Pixels	2.95	3.12	3.35

MNIST: kNN Classifier on Latent Space

Model	$-\log p(\mathbf{x}_i)$ MNIST	MNIST+rot
Gauss VAE	96.80	108.40
Kumar-SB VAE	98.01	112.33
Logit-SB VAE	99.48	114.09
Gamma-SB VAE	100.74	113.22

(Estimated) Marginal Likelihoods

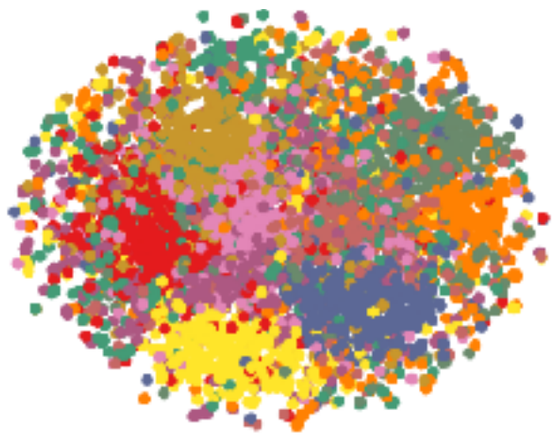
Semi-Supervised

Quantitative Results for SB-VAE

(Nalisnick & Smyth, 2017)

Unsupervised

MNIST: Dirichlet
Latent Space (t-SNE)



MNIST: Gaussian
Latent Space (t-SNE)

	k=3	k=5	k=10
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33
Raw Pixels	2.95	3.12	3.35

MNIST: kNN Classifier on Latent Space

Model	$-\log p(\mathbf{x}_i)$ MNIST	MNIST+rot
Gauss VAE	96.80	108.40
Kumar-SB VAE	98.01	112.33
Logit-SB VAE	99.48	114.09
Gamma-SB VAE	100.74	113.22

(Estimated) Marginal Likelihoods

Semi-Supervised

Nonparametric version of (Kingma et al., NIPS 2014)'s M2 model

	MNIST (N=45,000)			**SVHN** (N=65,000)		
	10%	5%	1%	10%	5%	1%
SB-DGM	4.86$_{\pm.14}$	5.29$_{\pm.39}$	**7.34**$_{\pm.47}$	**32.08**$_{\pm4.00}$	**37.07**$_{\pm5.22}$	**61.37**$_{\pm3.60}$
Gauss-DGM	**3.95**$_{\pm.15}$	**4.74**$_{\pm.43}$	11.55$_{\pm2.28}$	36.08$_{\pm1.49}$	48.75$_{\pm1.47}$	69.58$_{\pm1.64}$
kNN	6.13$_{\pm.13}$	7.66$_{\pm.10}$	15.27$_{\pm.76}$	64.81$_{\pm.34}$	68.94$_{\pm.47}$	76.64$_{\pm.54}$

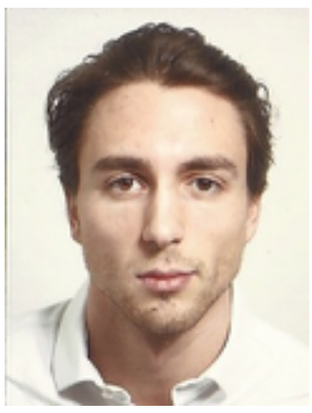
Deep Generative Models with Stick-Breaking Priors

3.2 The Dirichlet Process

Variational Autoencoder

In collaboration with

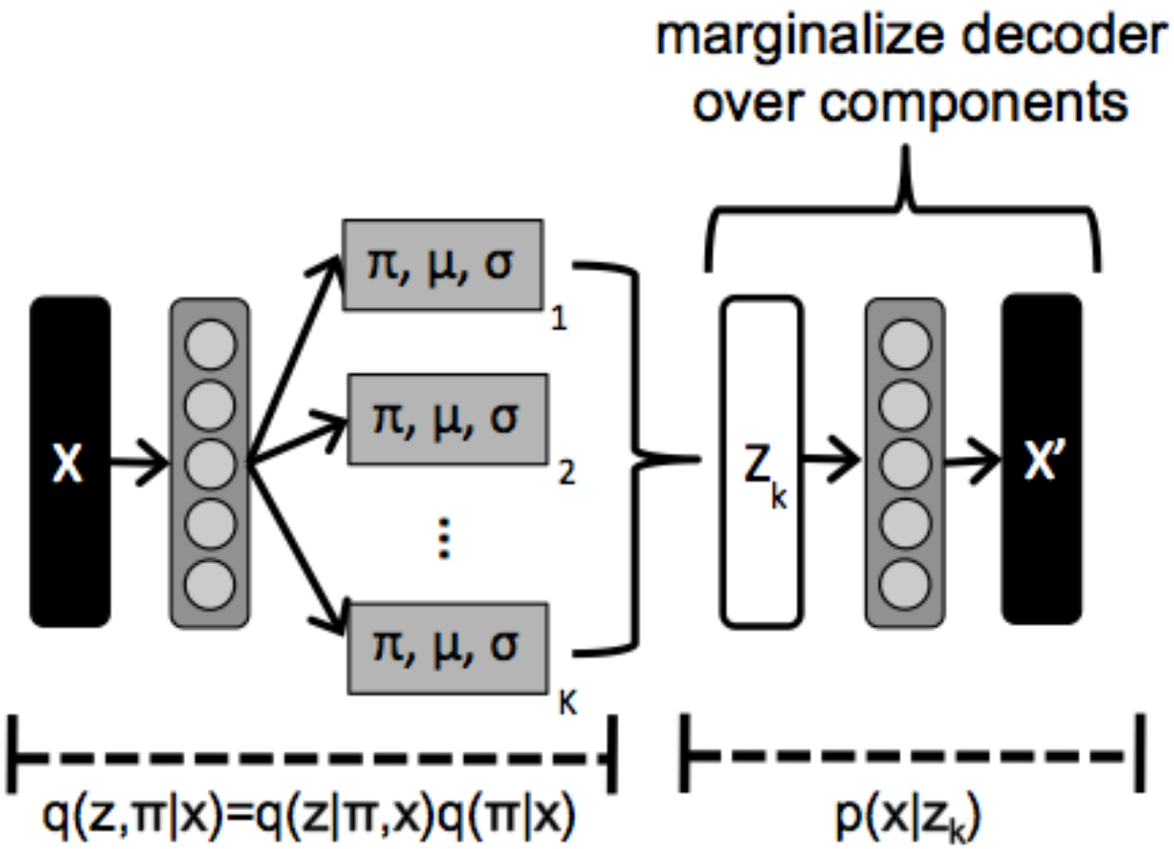
Padhraic Smyth



Lars Hertel

MODEL #2: Dirichlet Process Variational Autoencoder

(Nalisnick et al., 2016)



$$\mathcal{L}_{\text{SGVB}} = \sum_k \mu_{\pi_k} [\frac{1}{S} \sum_s \log p_{\boldsymbol{\theta}}(\mathbf{x}_i | \hat{\mathbf{z}}_{i,k,s}) + \mathbb{E}_{q_k}[\log p(\mathbf{z}_i)]]$$

$$- \text{KLD}[q(\boldsymbol{\pi}_k|\mathbf{x}_i)||p(\boldsymbol{\pi}_k)] - \frac{1}{S} \sum_s \log \sum_k \hat{\pi}_{i,k,s} q(\hat{\mathbf{z}}_{i,k,s}; \boldsymbol{\phi}_k)$$

Samples from Generative Model

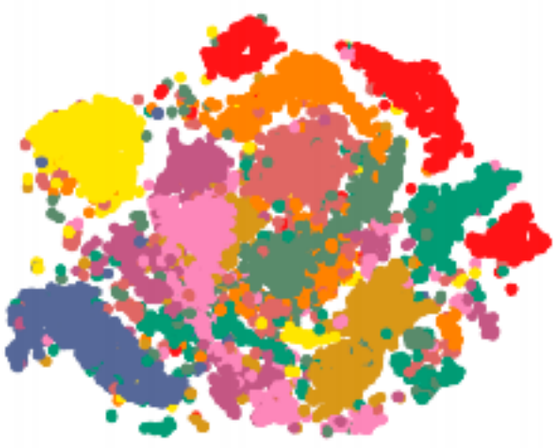
(Nalisnick et al., 2016)

MNIST Samples from Component #1

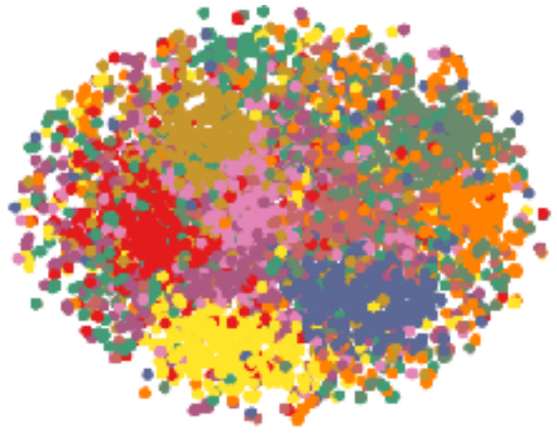
MNIST Samples from Component #5

Quantitative Results for DP-VAE

(Nalisnick et al., 2016)



MNIST: Dirichlet Process Latent Space (t-SNE)



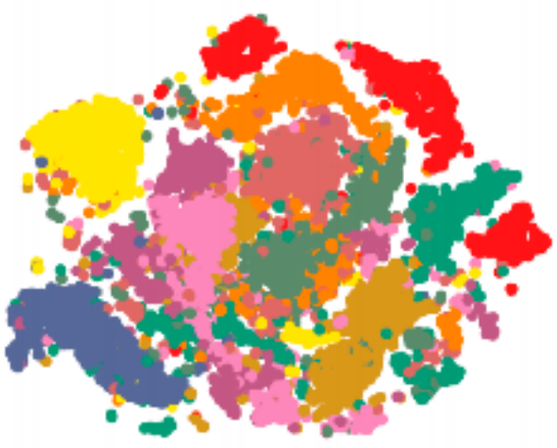
MNIST: Gaussian Latent Space (t-SNE)

	k=3	k=5	k=10
DLGMM	**9.14**	**8.38**	**8.42**
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33

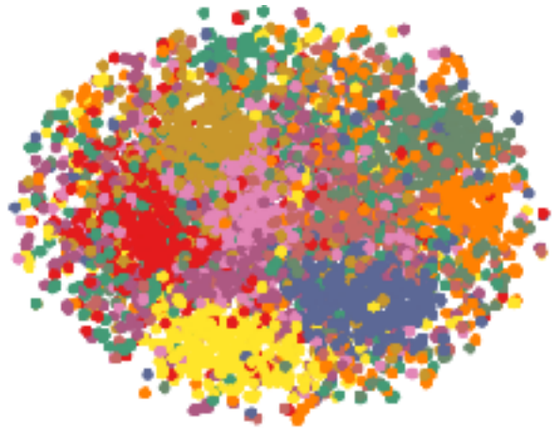
MNIST: kNN Classifier on Latent Space

Quantitative Results for DP-VAE

(Nalisnick et al., 2016)



MNIST: Dirichlet Process Latent Space (t-SNE)



MNIST: Gaussian Latent Space (t-SNE)

	k=3	k=5	k=10
DLGMM	**9.14**	**8.38**	**8.42**
SB-VAE	9.34	8.65	8.90
Gauss-VAE	28.4	20.96	15.33

MNIST: kNN Classifier on Latent Space

	$-\log p_{\theta}(\mathbf{x}_i)$	
	MNIST	OMNIGLOT
DLGMM (500d-3x25s)	**96.50**	123.50
DLDPMM (500d-17tx25s)	96.91	123.76
Gauss-VAE (500d-25s)	96.80	**119.18**
SB-VAE (500d-25t)	98.01	–

(Estimated) Marginal Likelihoods

Deep Generative Models with Stick-Breaking Priors

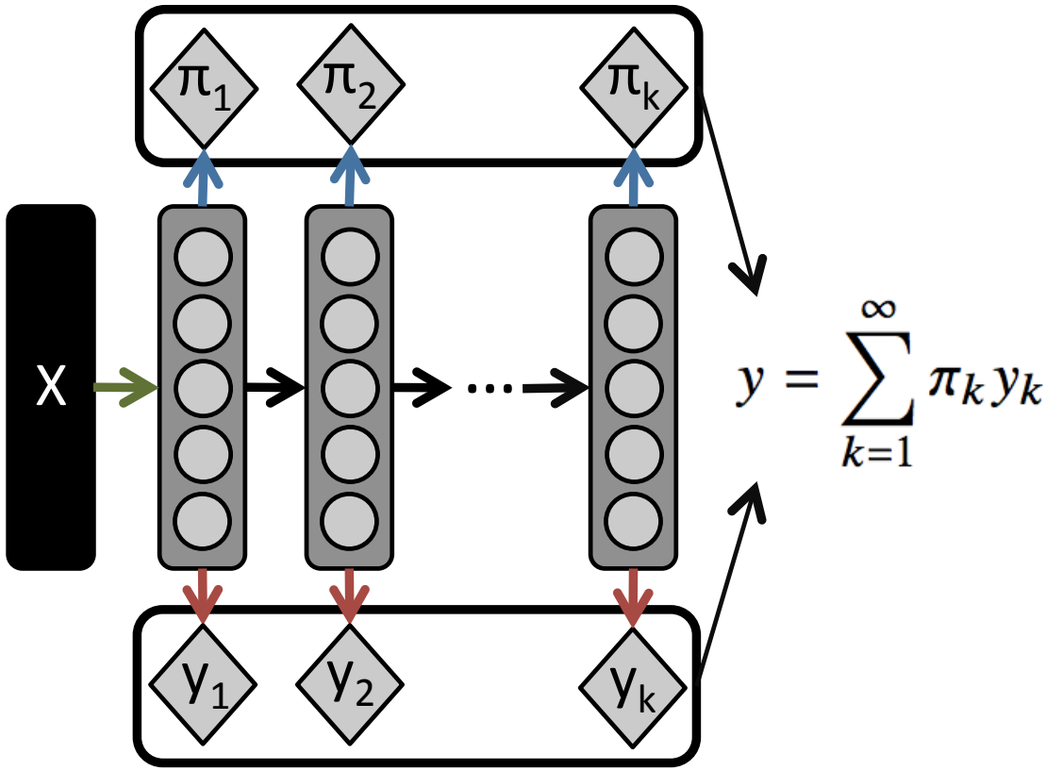
3.3 Adaptive Computation Time via Stick-Breaking

In collaboration with

Hugo Larochelle

Marc-Alexandre Cote

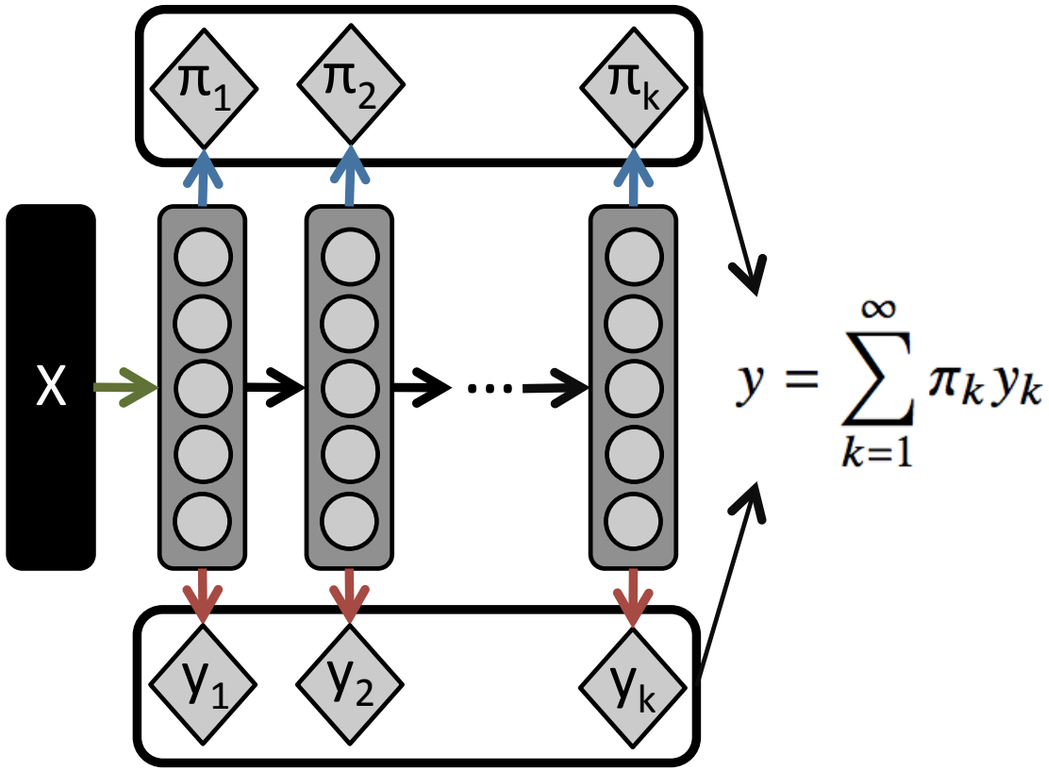
MODEL #3: Stick-Breaking Adaptive Computation



$$\text{stop when } \prod_k (1 - v_k) \leq \tau$$

Remaining stick is negligibly small

MODEL #3: Stick-Breaking Adaptive Computation



$$\text{stop when } \prod_k (1 - v_k) \leq \tau$$

Remaining stick is negligibly small

$$\log p_{\boldsymbol{\theta}}(y_i|\mathbf{x}_i) \geq \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(y_i|\hat{\boldsymbol{\pi}}_{i,s}, \mathbf{x}_i) - KLD[q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i|\mathbf{x}_i)||p_{\boldsymbol{\lambda}}(\boldsymbol{\pi}_i)]$$

Quantitative Results for SB-RNN: Image Modeling

Model	**MNIST NLL**
Stick-Breaking NADE (w/ LSTM init.)	81.19
Stick-Breaking NADE	81.58
Diagonal BiLSTM	**79.20**
PixelCNN	81.30
DBN	84.55
EoNADE	84.68
DARN	≈ 84.13
DLGM	≈ 85.51

Negative log likelihood on MNIST. Various baseline results (lower block)

KL-Divergence from the prior per pixel.

Quantitative Results for SB-RNN: Text Modeling

Model	**Perplexity**
Stick-Breaking LSTM	**1.27**
Adaptive Computation Time LSTM (Graves, 2016)	1.30
LSTM w/ Batch Norm. (Cooijmans et al., 2016)	1.32
GRU (our implementation)	1.34
LSTM (our implementation)	1.35
LSTM (Cooijmans et al., 2016)	1.38

Character-level modeling of Penn Treebank.

Quantitative Results for SB-RNN: Text Modeling

Model	**Perplexity**
Stick-Breaking LSTM	**1.27**
Adaptive Computation Time LSTM (Graves, 2016)	1.30
LSTM w/ Batch Norm. (Cooijmans et al., 2016)	1.32
GRU (our implementation)	1.34
LSTM (our implementation)	1.35
LSTM (Cooijmans et al., 2016)	1.38

Character-level modeling of Penn Treebank.

Model	**Perplexity**
Stick-Breaking LSTM (ens. 1)	1.41
Stick-Breaking LSTM (ens. 32)	**1.36**
Stick-Breaking GRU (ens. 1)	1.45
Stick-Breaking GRU (ens. 32)	1.43
Adaptive Computation Time LSTM (Graves, 2016)	1.42
LSTM w/ Batch Norm. (Cooijmans et al., 2016)	**1.36**
GRU (our implementation)	1.48
LSTM (our implementation)	1.42
LSTM (Cooijmans et al., 2016)	1.43

Character-level modeling of Text8.

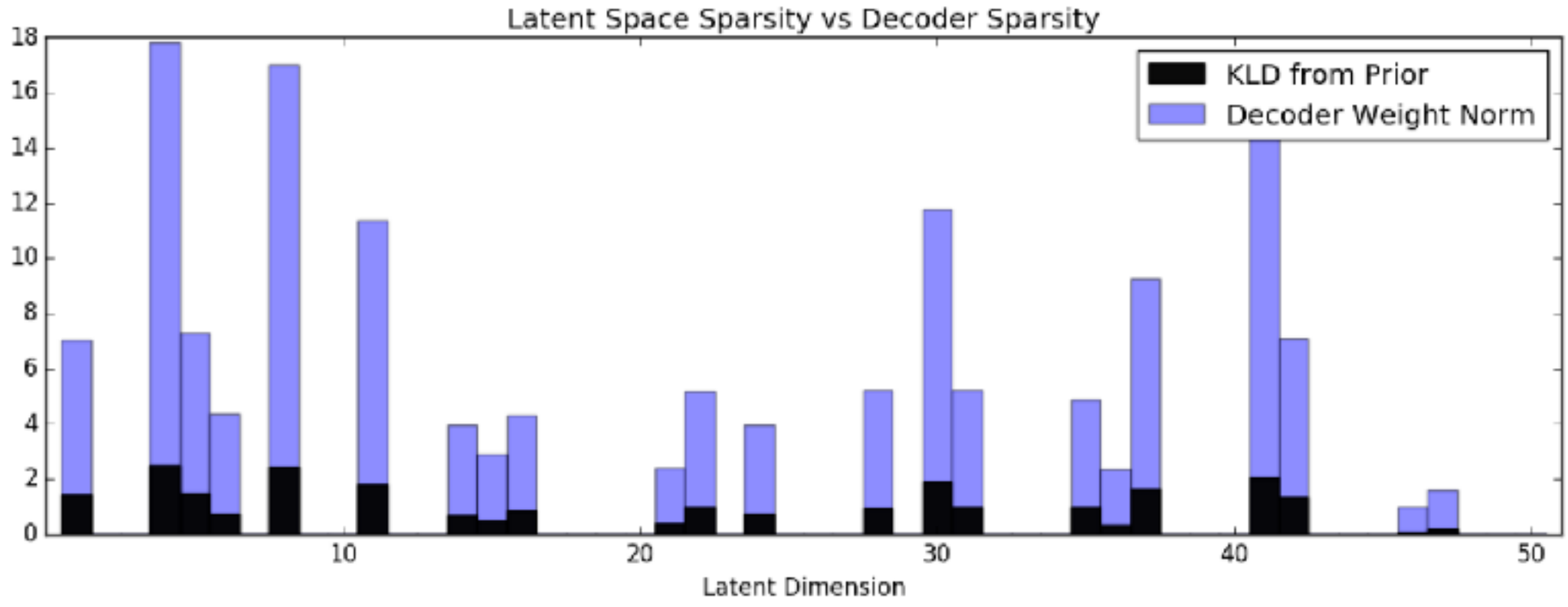
Conclusions

1. **Superior Latent Spaces with NP priors:** Seem to preserve class structure well, resulting in better discriminative properties. Their dynamic capacity naturally encodes factors of variation.

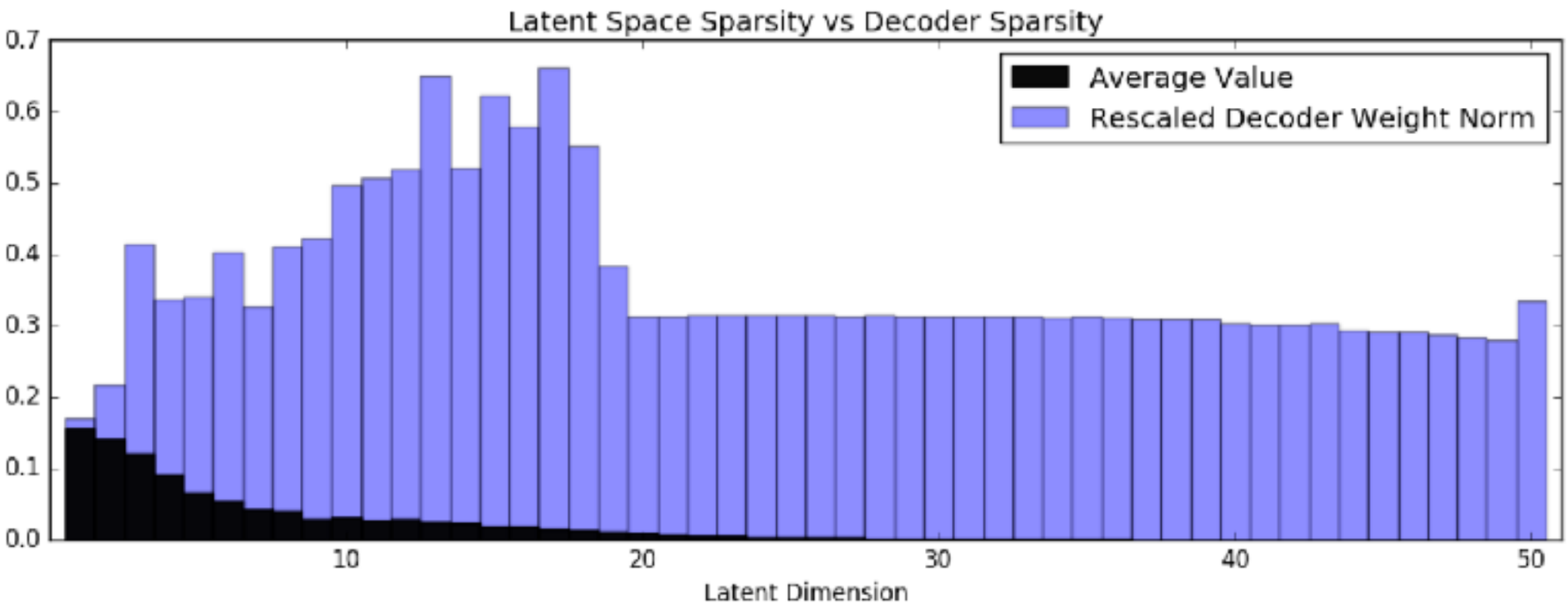
2. **Adaptive Computation:** Fusing NNs with nonparametric priors endows NNs with natural model selection properties.

Thank you. Questions?

Appendix



(a) Gauss VAE



(b) Stick-Breaking VAE