Deep Generative Models: GANs and VAE

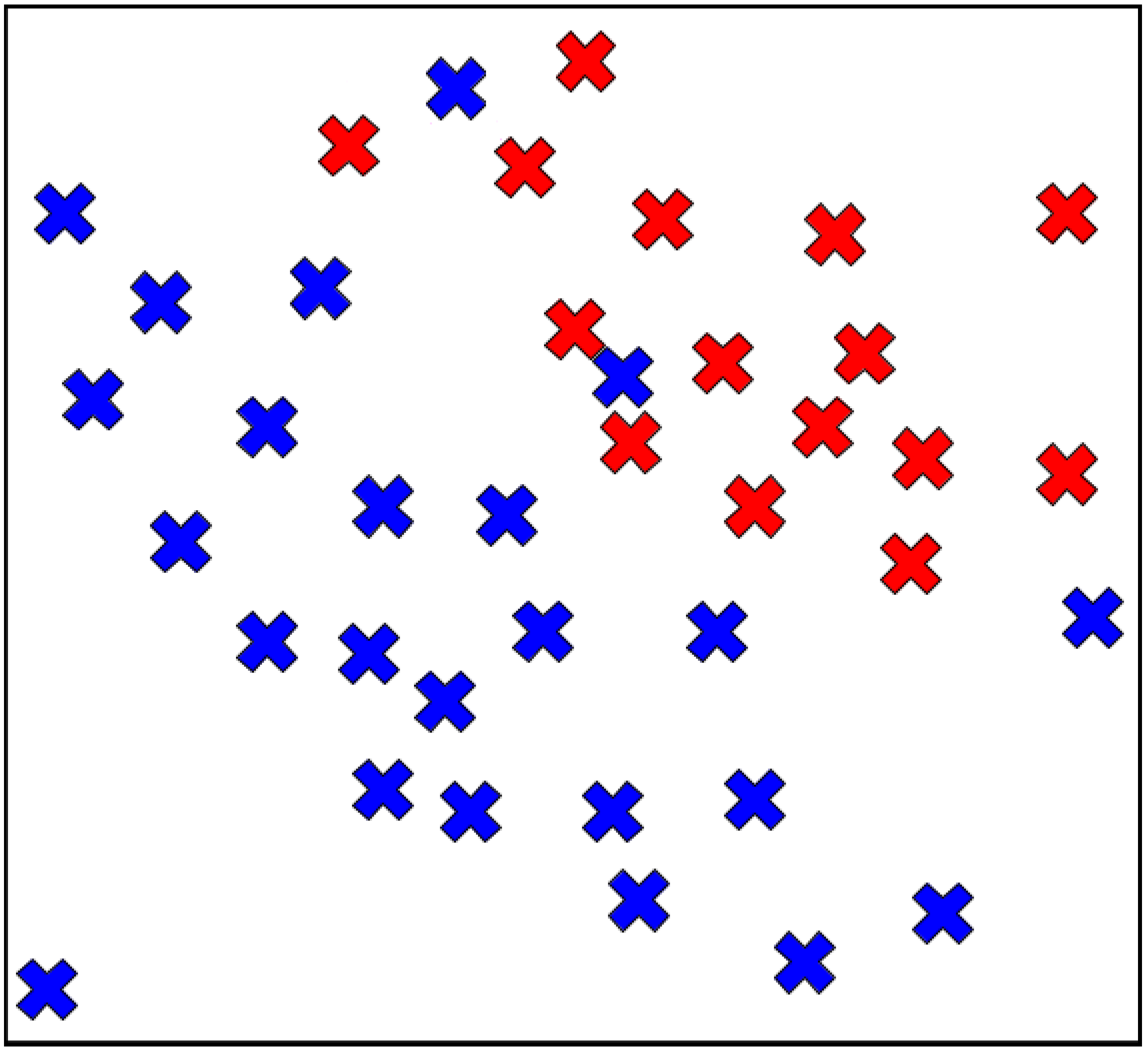
Jakub M. Tomczak
AMLAB, Universiteit van Amsterdam

Split, Croatia 2017

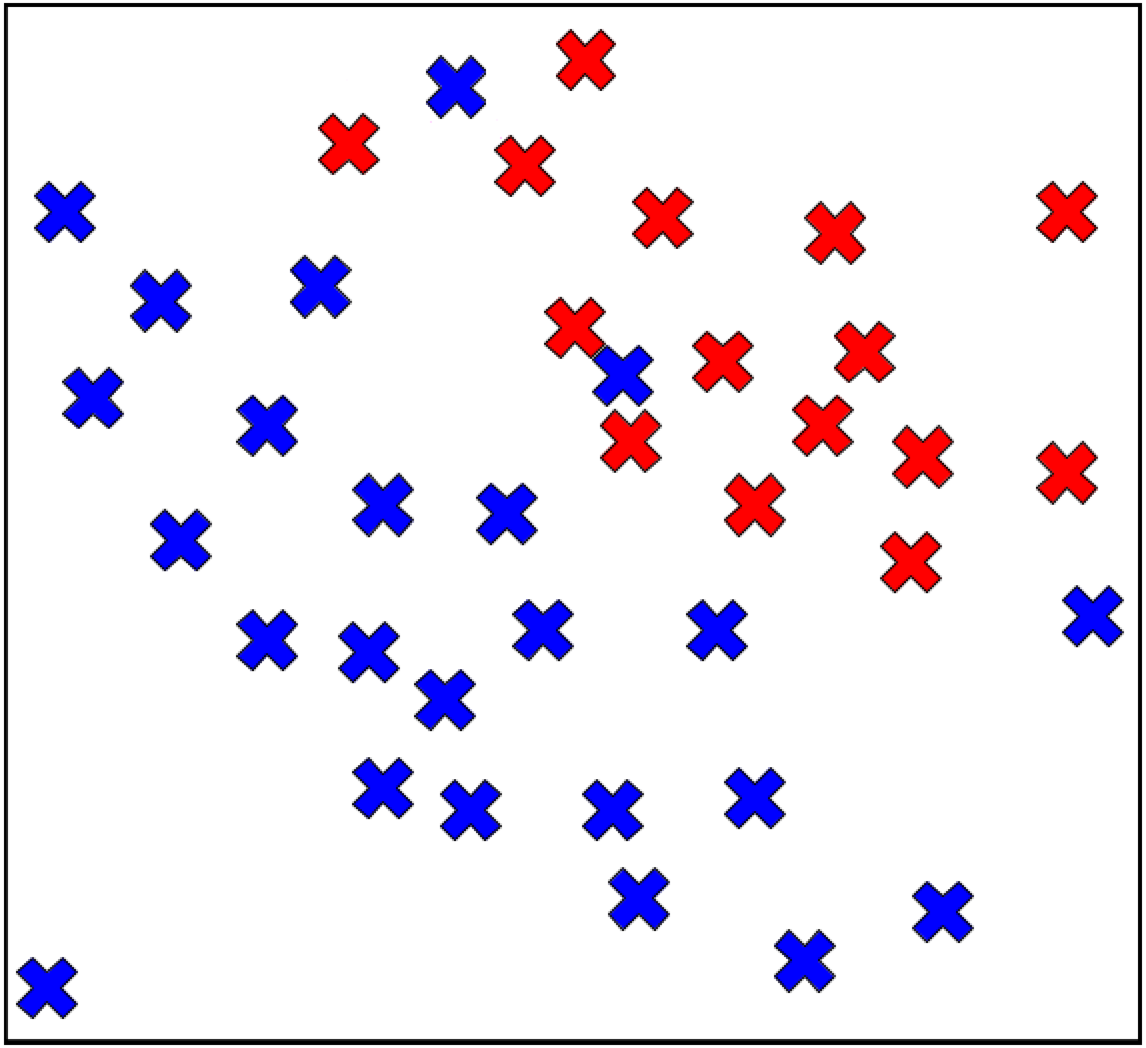
UNIVERSITY OF AMSTERDAM

Do we need generative modeling?

Do we need generative modeling?

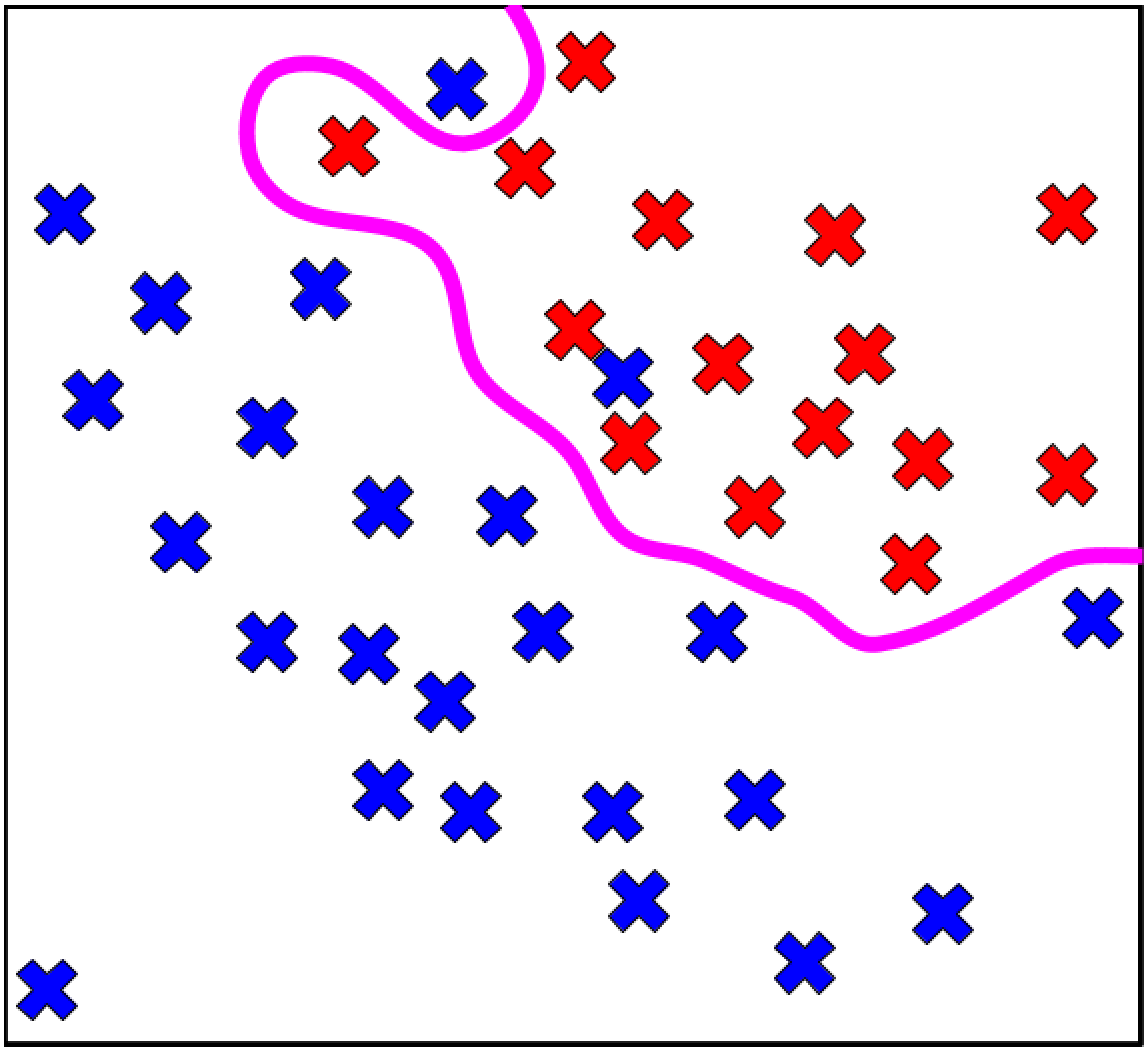


Do we need generative modeling?



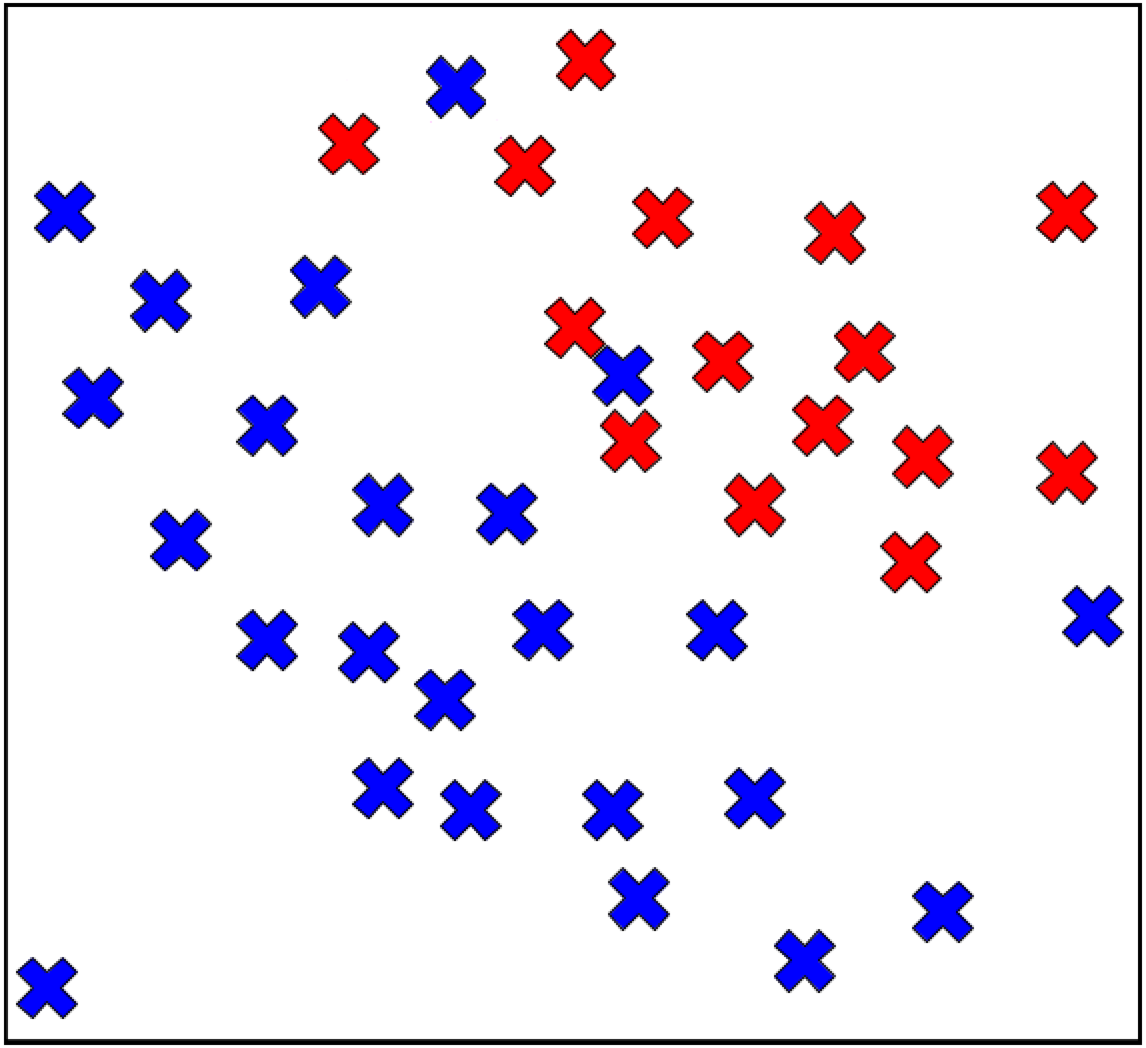
$p_\theta(y|x)$

Do we need generative modeling?



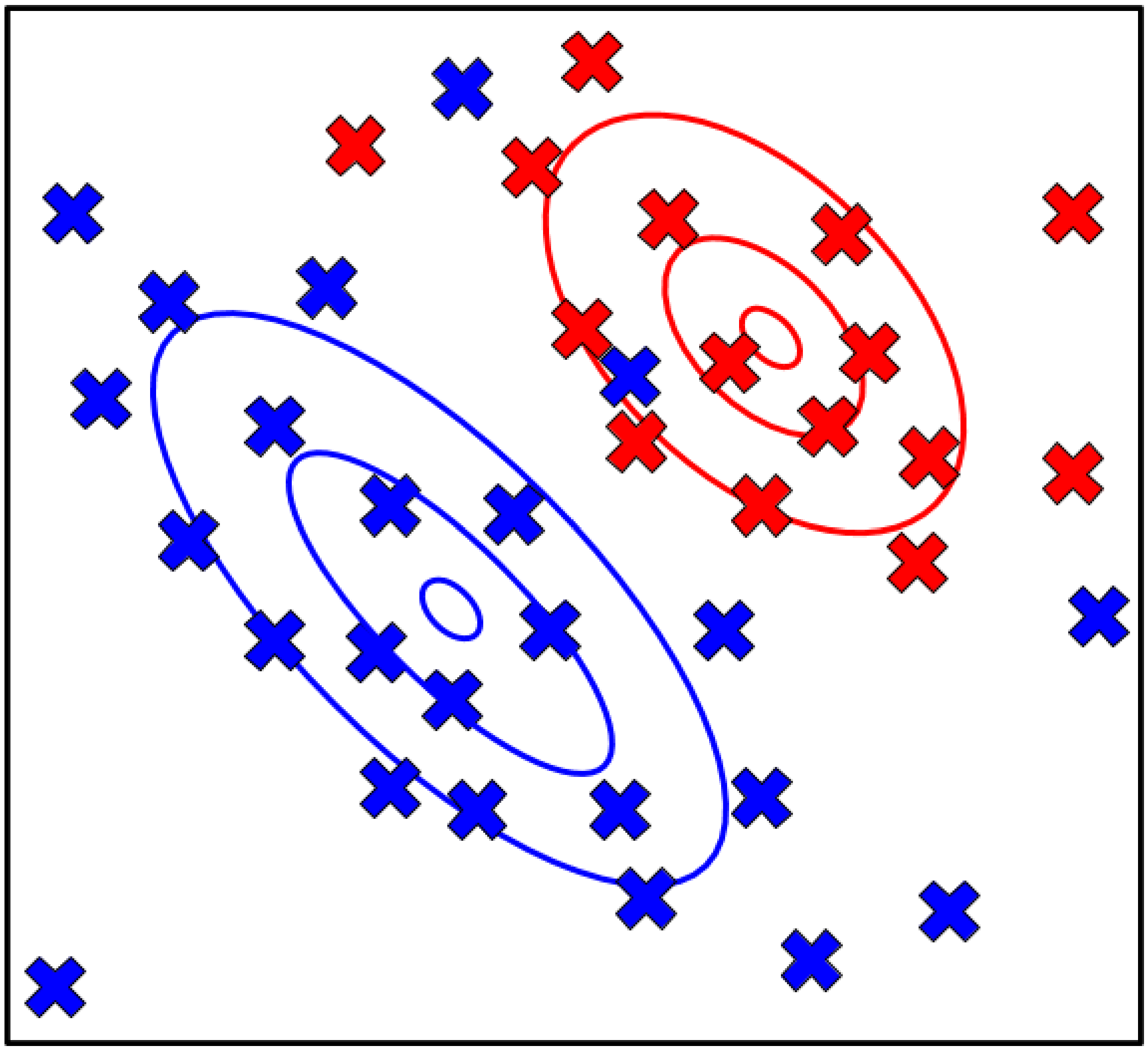
$p_\theta(y|x)$

Do we need generative modeling?



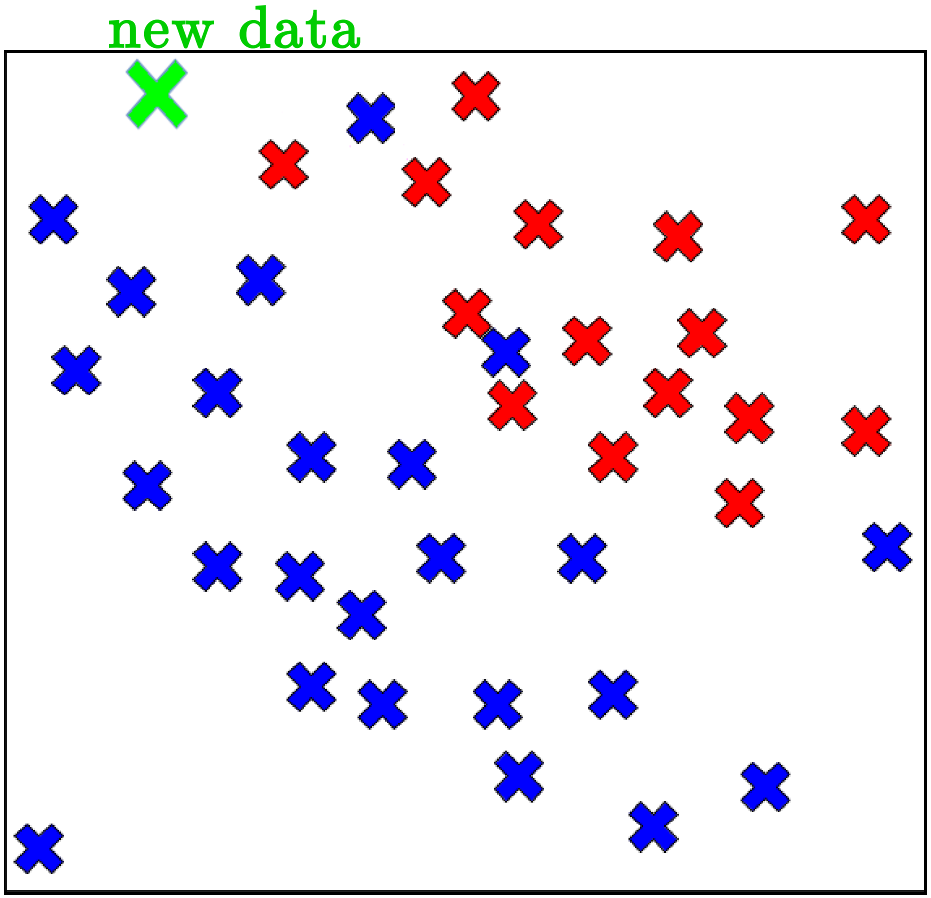
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

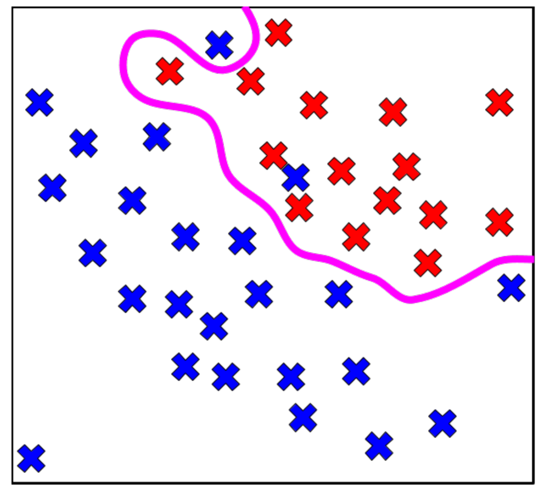
Do we need generative modeling?



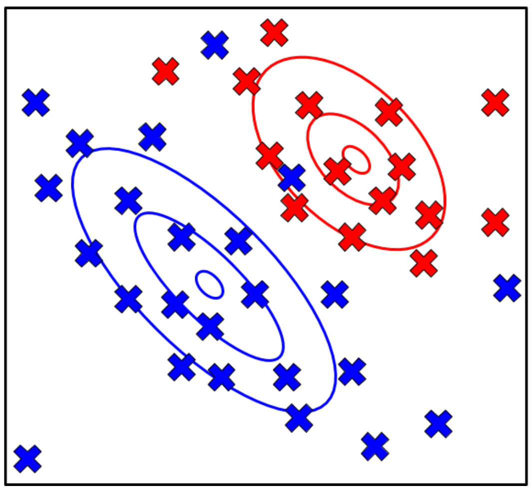
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

Do we need generative modeling?



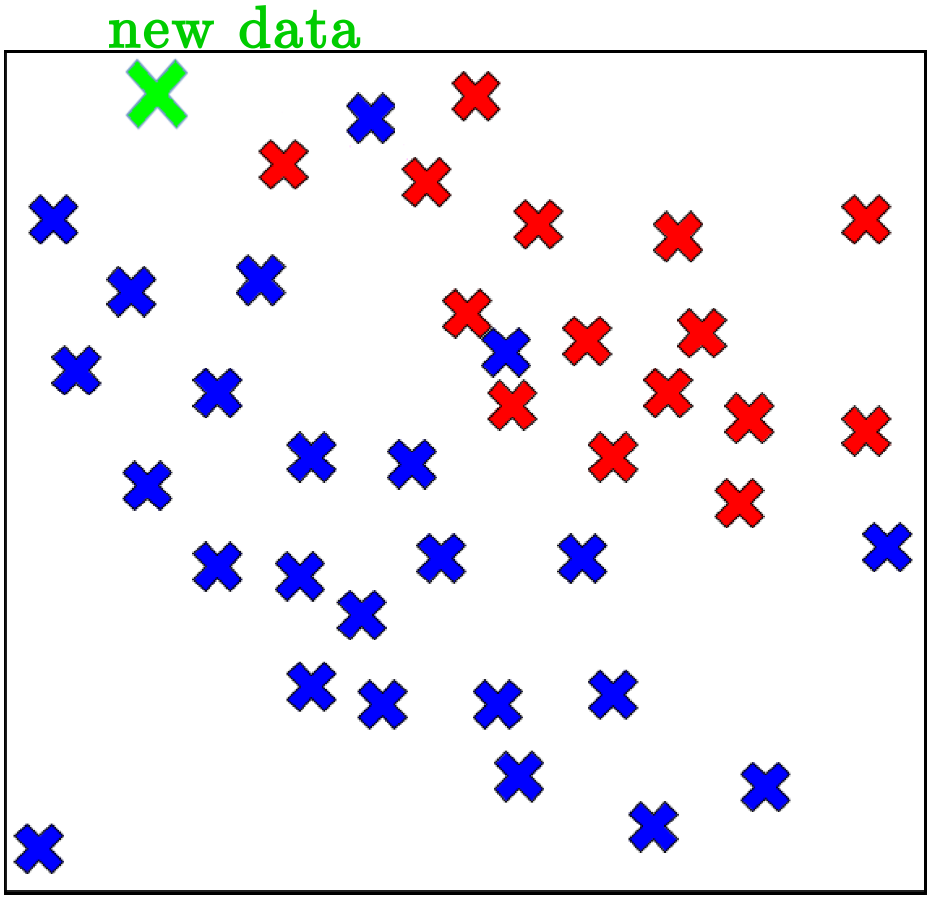


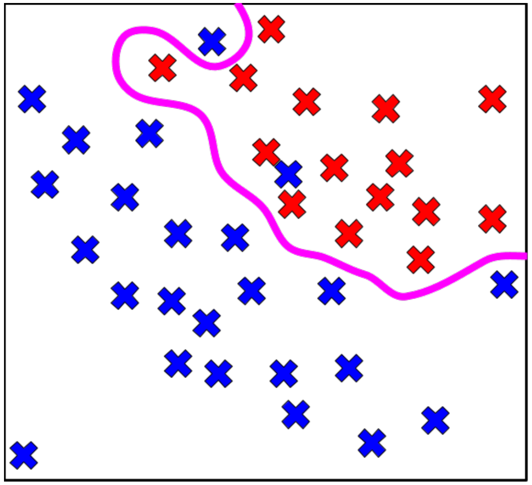
$p_\theta(y|x)$



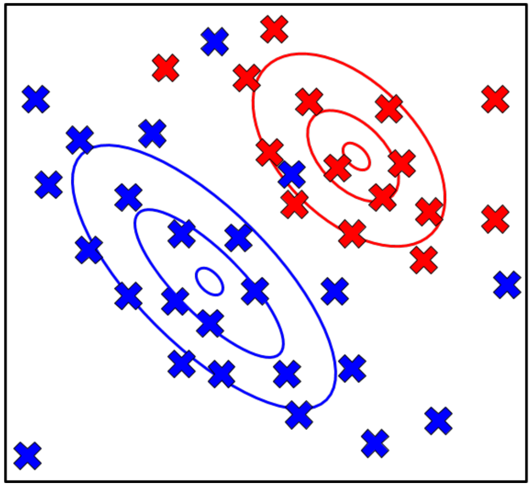
$p_\theta(x, y)$

Do we need generative modeling?





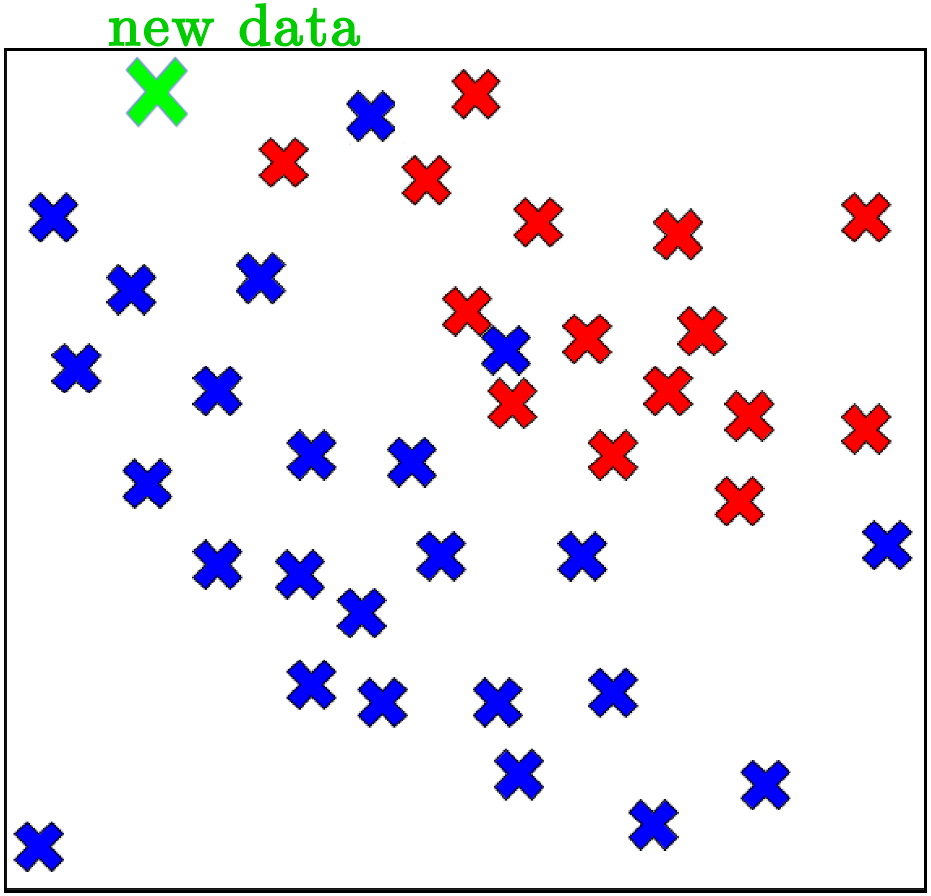
$p_\theta(y|x)$

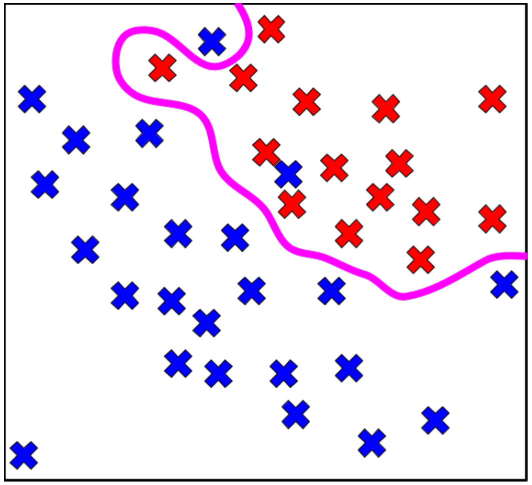


$p_\theta(x, y)$

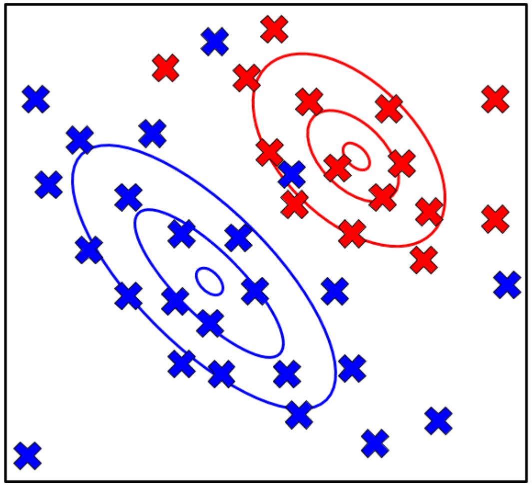
High probability
of the **blue label**.
=
**Highly probable
decision!**

Do we need generative modeling?





$p_\theta(y|x)$



$p_\theta(x, y)$

High probability of the blue **label**.

$=$

Highly probable decision!

High probability of the blue **label**.

x

Low probability of the **object**.

$=$

Uncertain decision!

Generative Modeling

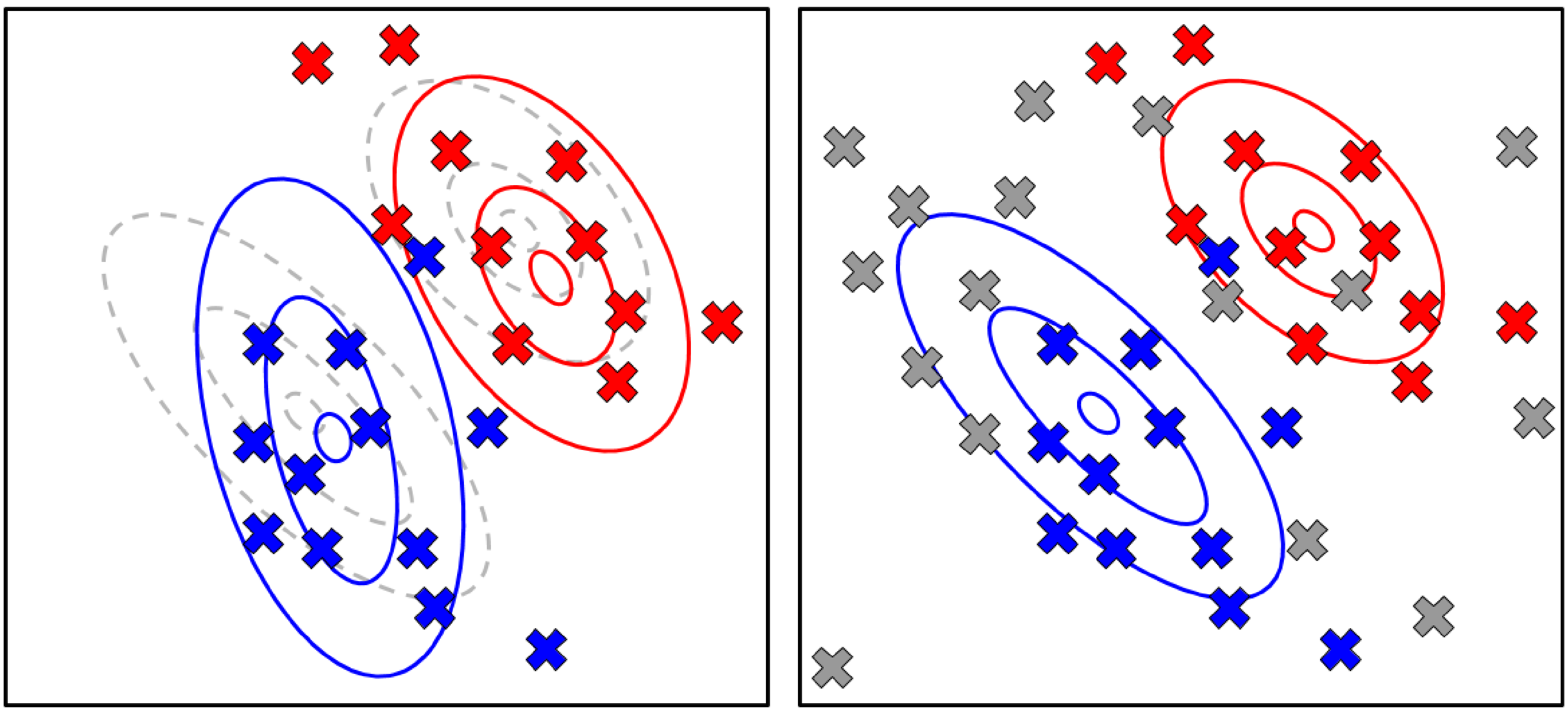
- Providing decision is not enough. *How to evaluate uncertainty? Distribution of y is only a part of the story.*

- Generalization problem. *Without knowing the distribution of $\mathbf{x}$ how we can generalize to new data?*

- Understanding the problem is crucial ("**What I cannot create, I do not understand**", Richard P. Feynman). *Properly modeling data is essential to make better decisions.*

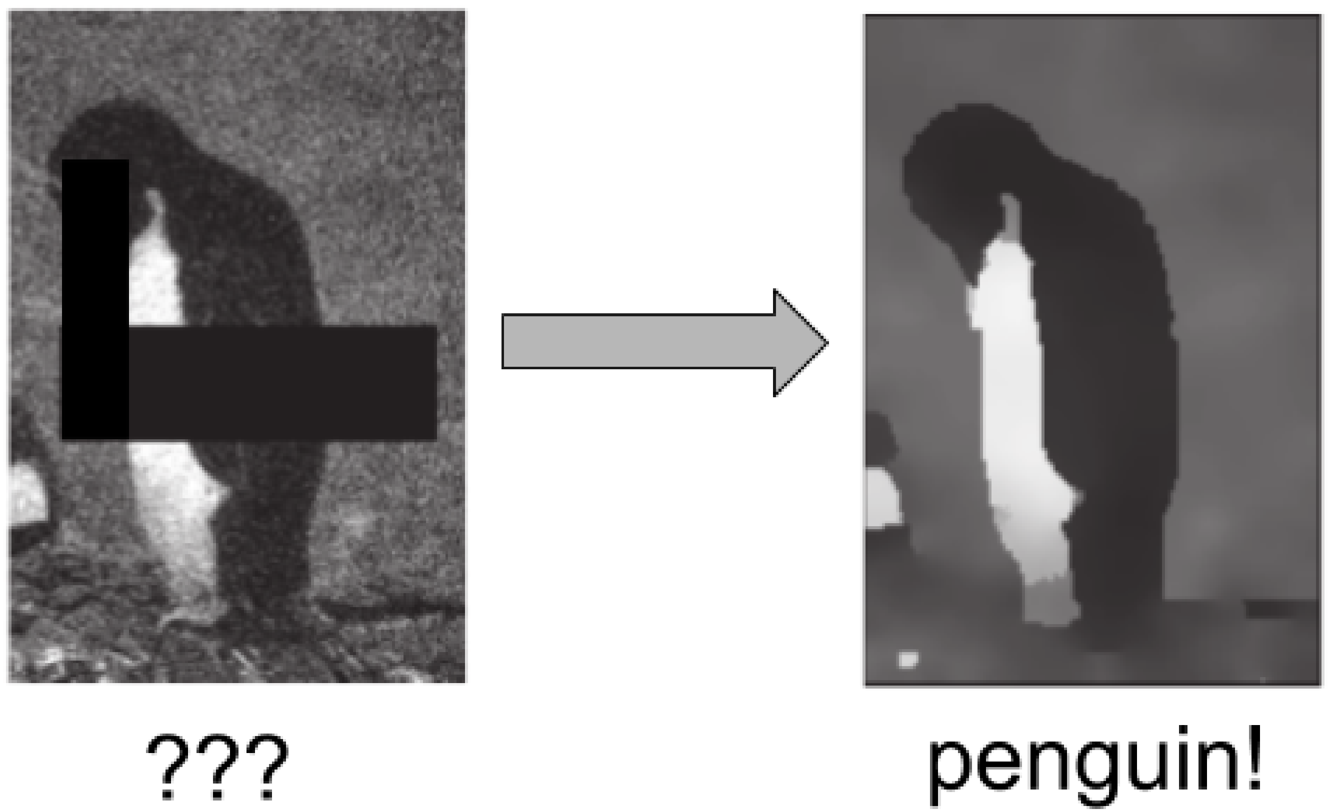
Generative Modeling

- Semi-supervised learning.
  Use unlabeled data to train a better classifier.



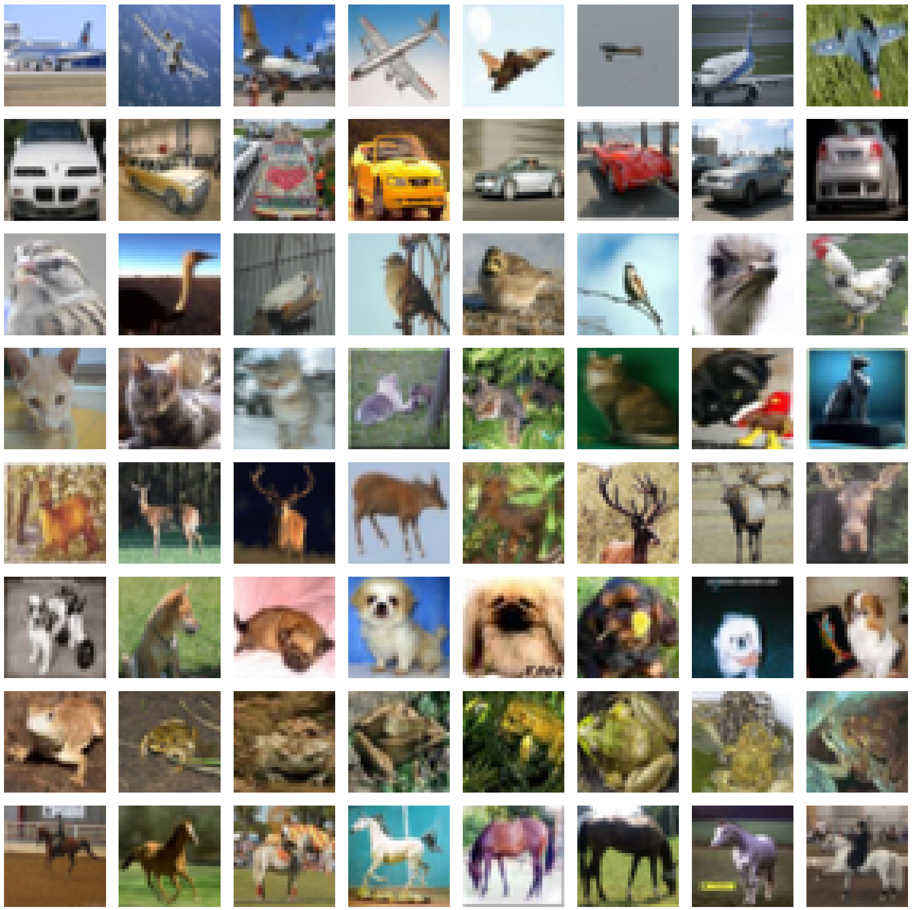
Generative Modeling

- Handling missing or distorted data.
  Reconstruct and/or denoise data.

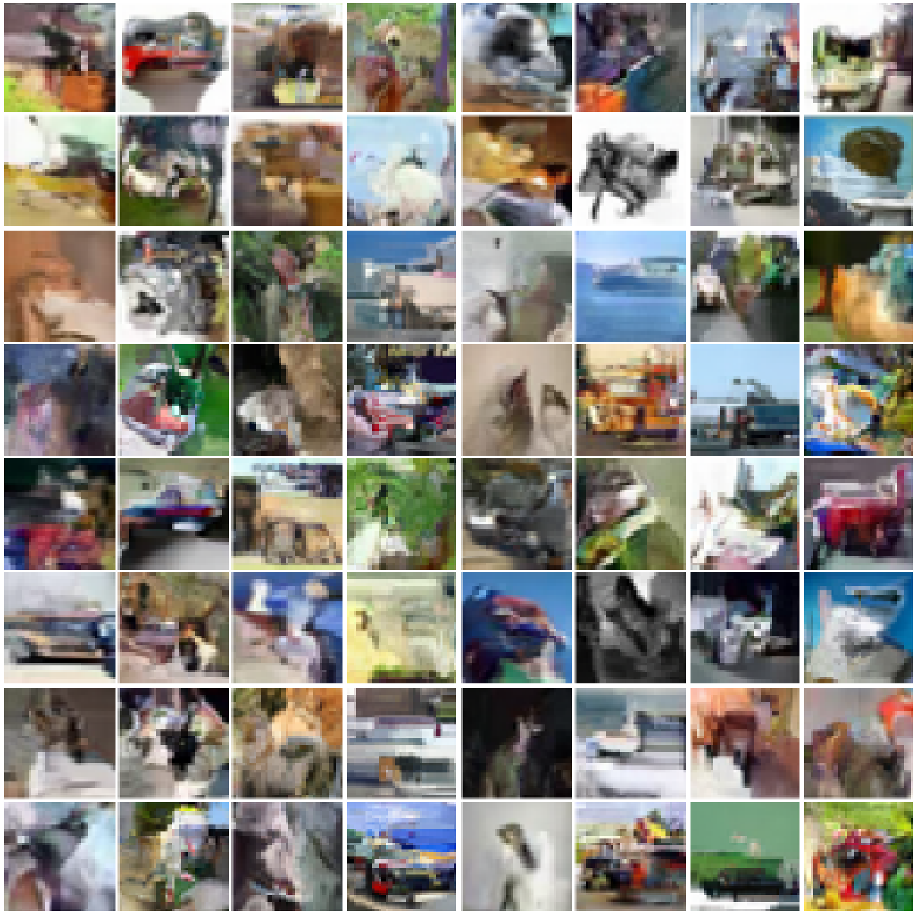


Generative Modeling

Image generation



Real



Generated

CHEN, Xi, et al. Variational lossy autoencoder. arXiv preprint arXiv:1611.02731, 2016.

Generative Modeling

Sequence generation

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 .

Generated

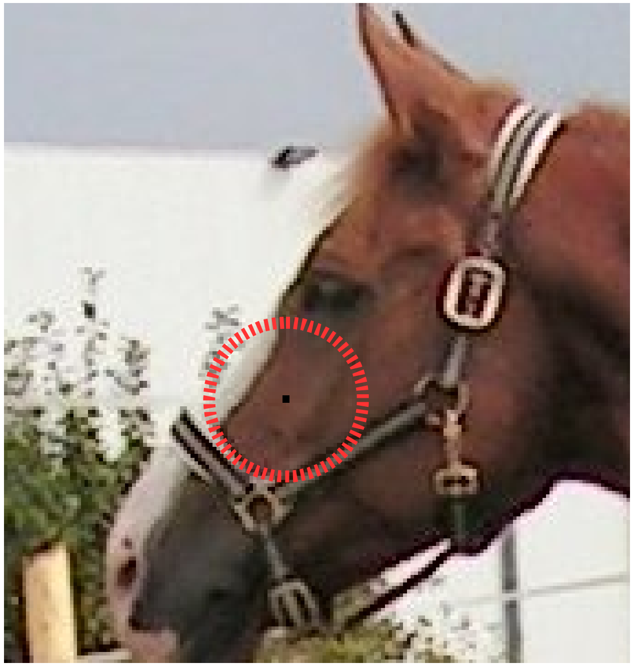
BOWMAN, Samuel R., et al. Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349, 2015.

How to formulate a generative model?

Modeling in high-dimensional space is difficult.

How to formulate a generative model?

Modeling in high-dimensional space is difficult.



How to formulate a generative model?

Modeling in high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

How to formulate a generative model?

Modeling in high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

very inefficient!

How to formulate a generative model?

Modeling in high-dimensional space is difficult.

$\rightarrow$ modeling all dependencies among pixels.

$$p(x) = \prod_{d=1}^{c} \psi_c(x_c)$$

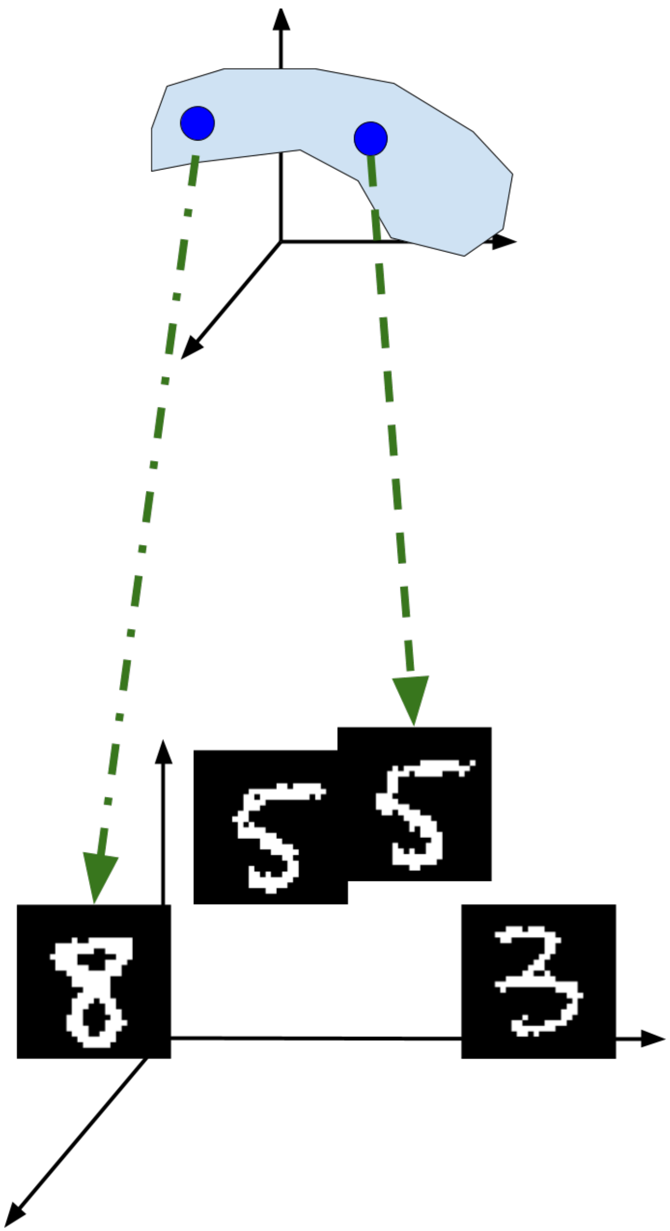
very inefficient!

A possible **solution**? $\rightarrow$ **Latent variable models**

Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$



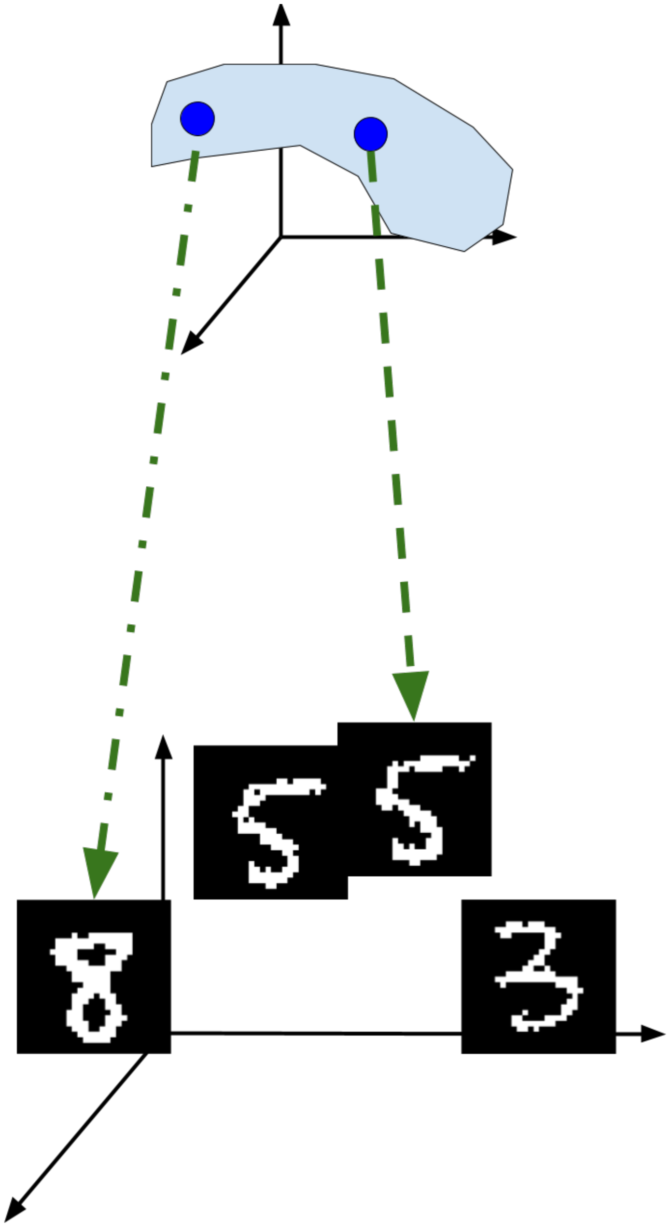
Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$

First sample $\mathbf{z}$.

Second, sample $\mathbf{x}$ for given $\mathbf{z}$.



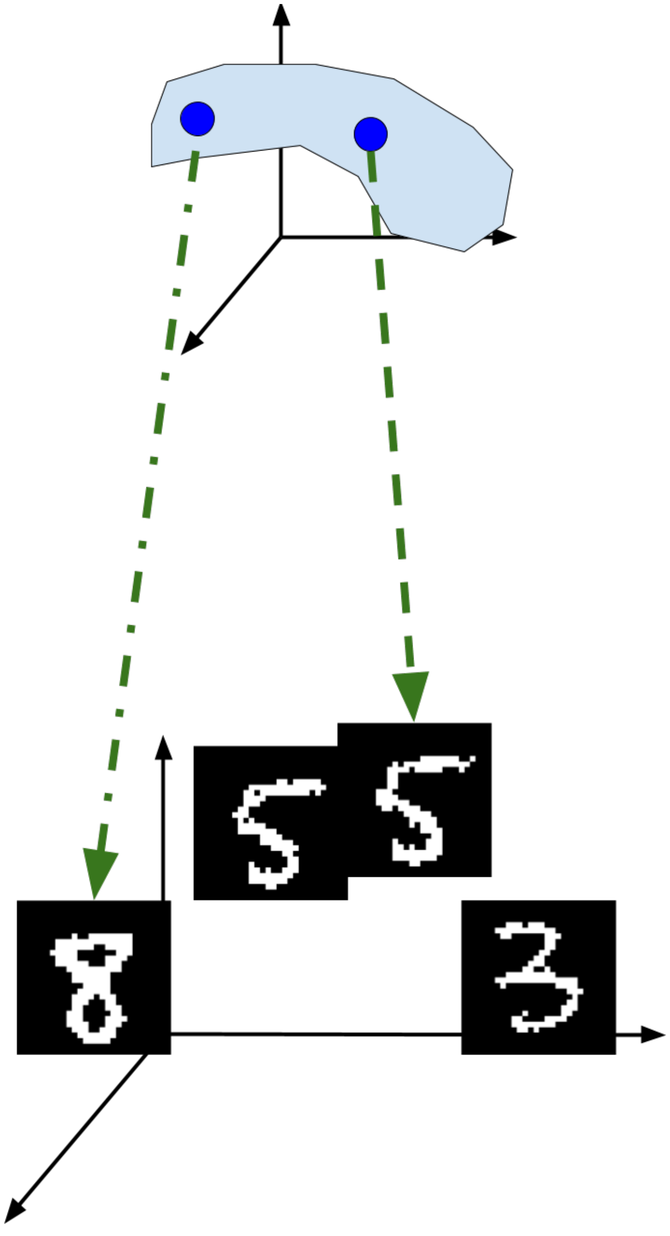
Latent Variable Models

- **Latent variable model**:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

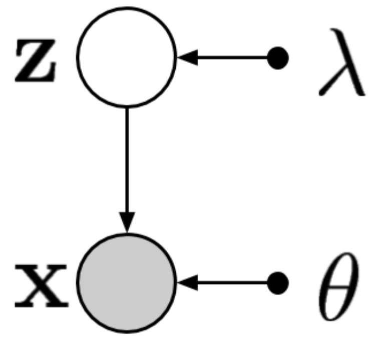
First sample $\mathbf{z}$.

Second, sample $\mathbf{x}$ for given $\mathbf{z}$.



Latent Variable Models

- Latent variable model:



$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

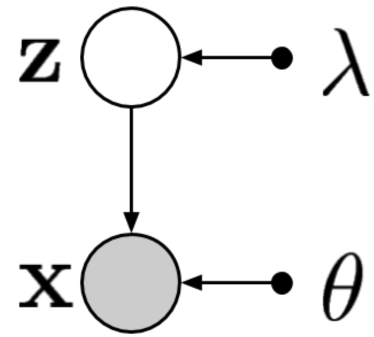
- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ ***an infinite mixture of Gaussians.***

Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$



- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

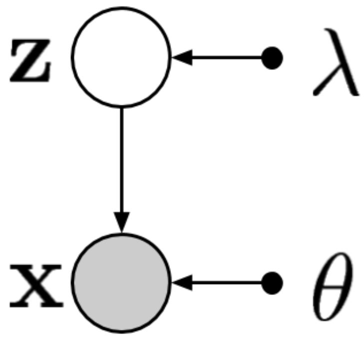
Convenient but limiting!

- What if we take a **non-linear transformation** of z? $\rightarrow$ *an infinite mixture of Gaussians.*

Latent Variable Models

- **Latent variable model**:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$



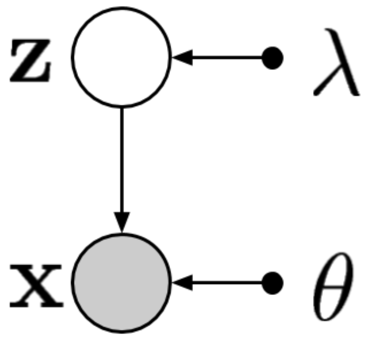
- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ *an infinite mixture of Gaussians.*

Latent Variable Models

- Latent variable model:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\; \mathrm{d}\mathbf{z}$$

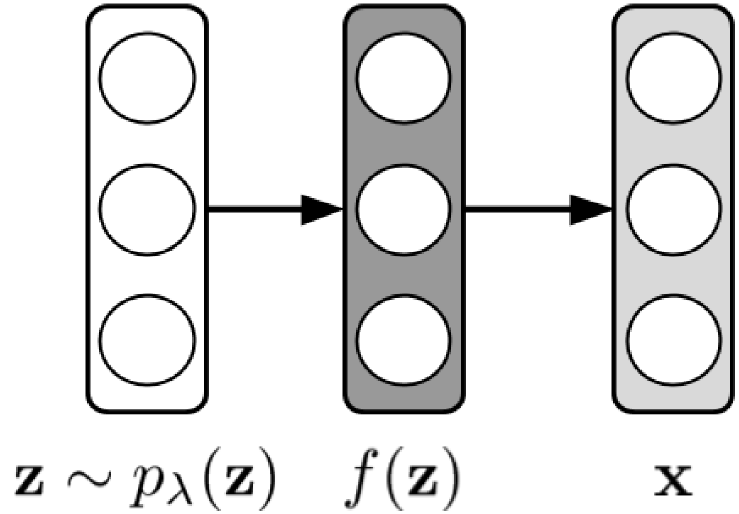


- If $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mathbf{b}, \Psi)$ and $p_\lambda(\mathbf{z}) = \mathcal{N}(\mu_0, \Sigma_0)$, then $\rightarrow$ **Factor Analysis**.

- What if we take a **non-linear transformation** of z? $\rightarrow$ *an infinite mixture of Gaussians.*

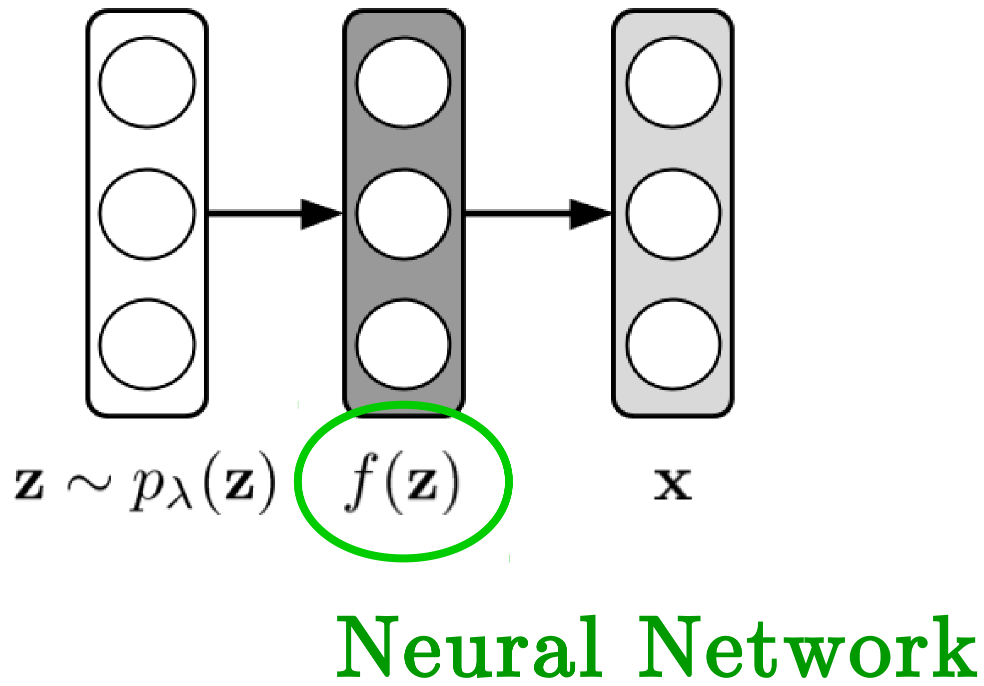
Neural network

Deep Generative Models (DGM): Density Network



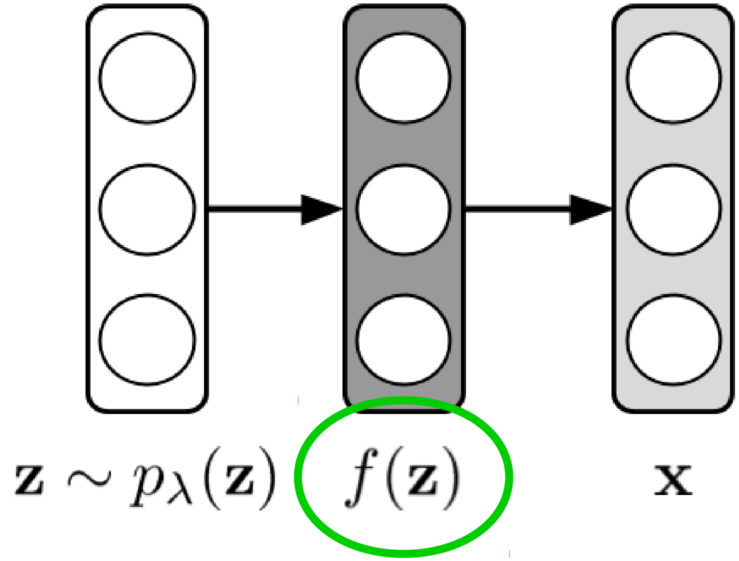
MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

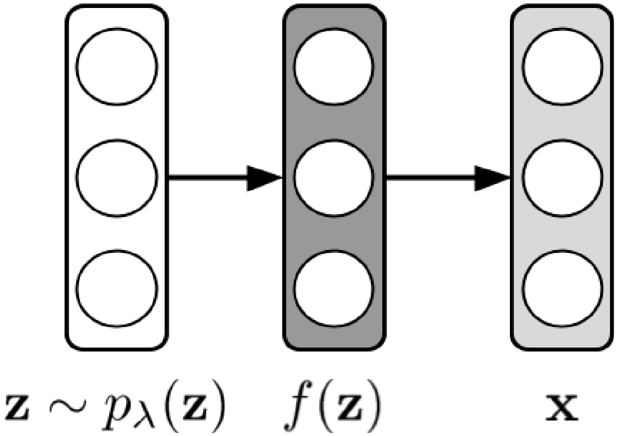


Neural Network

How to train this model?!

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

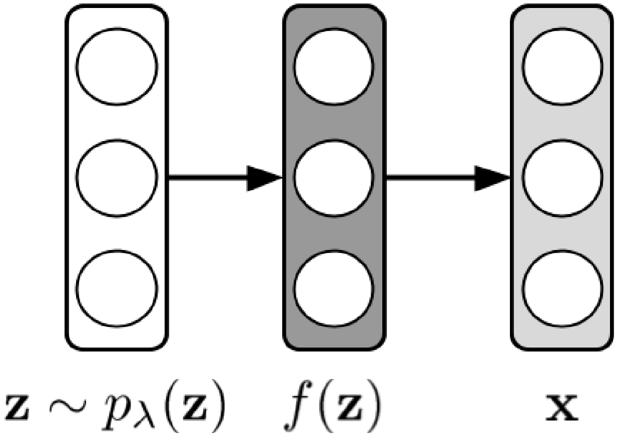
- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

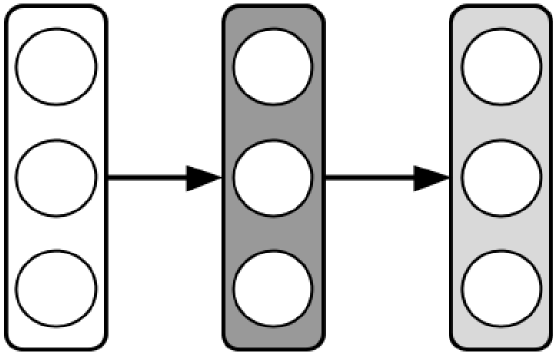
$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$



Sample $\mathbf{z}$ many times, apply log-sum-exp trick and maximize log-likelihood.

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network



- MC approximation:

$$\log p(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

$$\approx \log \frac{1}{S} \sum_{s=1}^{S} \exp\Big(\log p_\theta(\mathbf{x}|\mathbf{z}_s)\Big)$$

where:

$$\mathbf{z}_s \sim p_\lambda(\mathbf{z})$$

Sample $\mathbf{z}$ many times, apply log-sum-exp trick and maximize log-likelihood.

It scales badly in high dimensional cases!

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

PROS

Log-**likelihood** approach

Easy **sampling**

Training using **gradient-based methods**

CONS

Requires **explicit** models

Fails in **high dim. cases**

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Density Network

PROS

Log-**likelihood** approach

Easy **sampling**

Training using **gradient-based methods**

CONS

Requires **explicit** models

Fails in **high dim. cases**

Can we do better?

MacKay, D. J., & Gibbs, M. N. (1999). *Density networks*. Statistics and neural networks: advances at the interface. Oxford University Press, Oxford, 129-144.

DGM: Generative Adversarial Nets

Let image two agents:

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

A fraud

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

A fraud

An art expert

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

A fraud

... and a real artist

An art expert

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

The fraud aims to copy the real artist and cheat the art expert.

A fraud

... and a real artist

An art expert

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

A fraud

The fraud aims to copy the real artist and cheat the art expert.

... and a real artist

The expert assesses a painting and gives her opinion.

An art expert

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

A fraud

... and a real artist

The fraud aims to copy the real artist and cheat the art expert.

The expert assesses a painting and gives her opinion.

An art expert

The fraud learns and tries to fool the expert.

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

Hmmm... fake!

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Let image two agents:

Hmmm... fake!

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

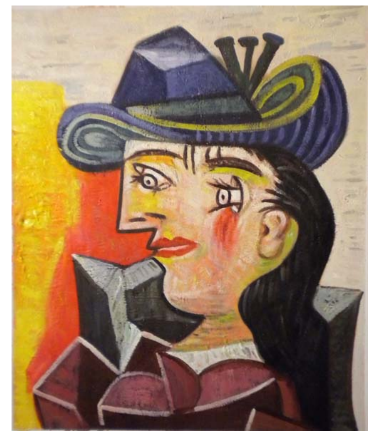
Let image two agents:

Hmmm... Pablo!

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

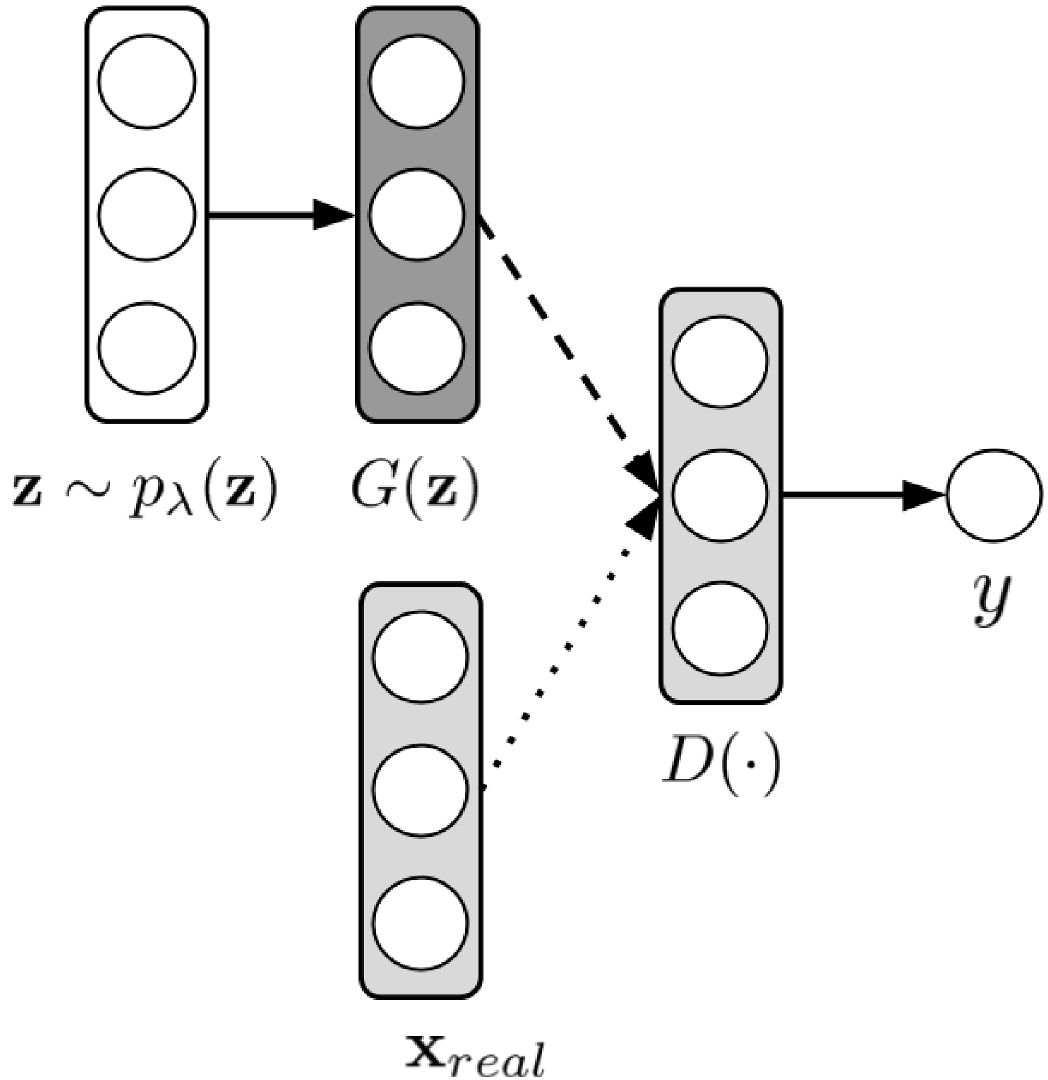
Let image two agents:



Hmmm... Pablo!

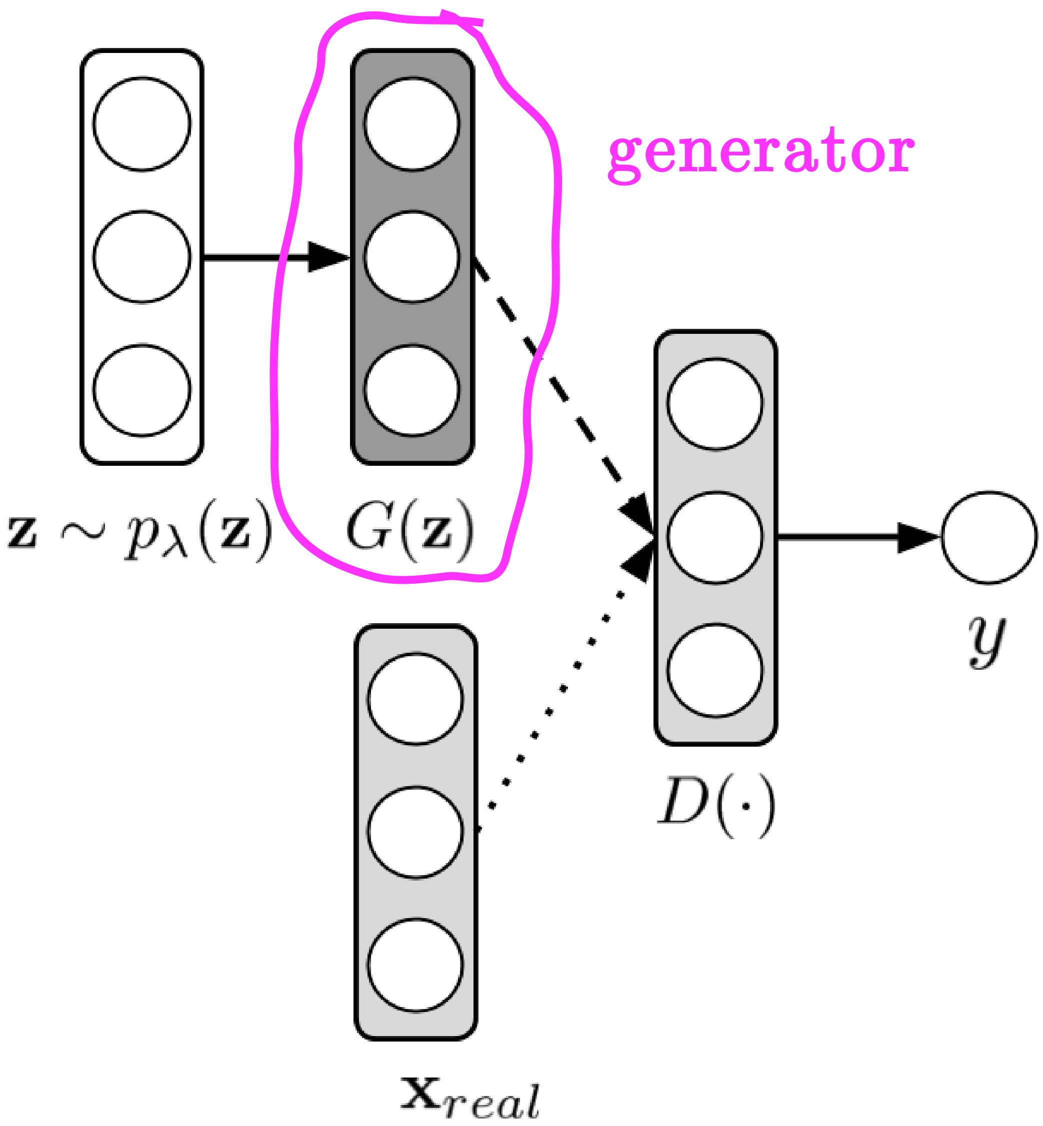
Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets



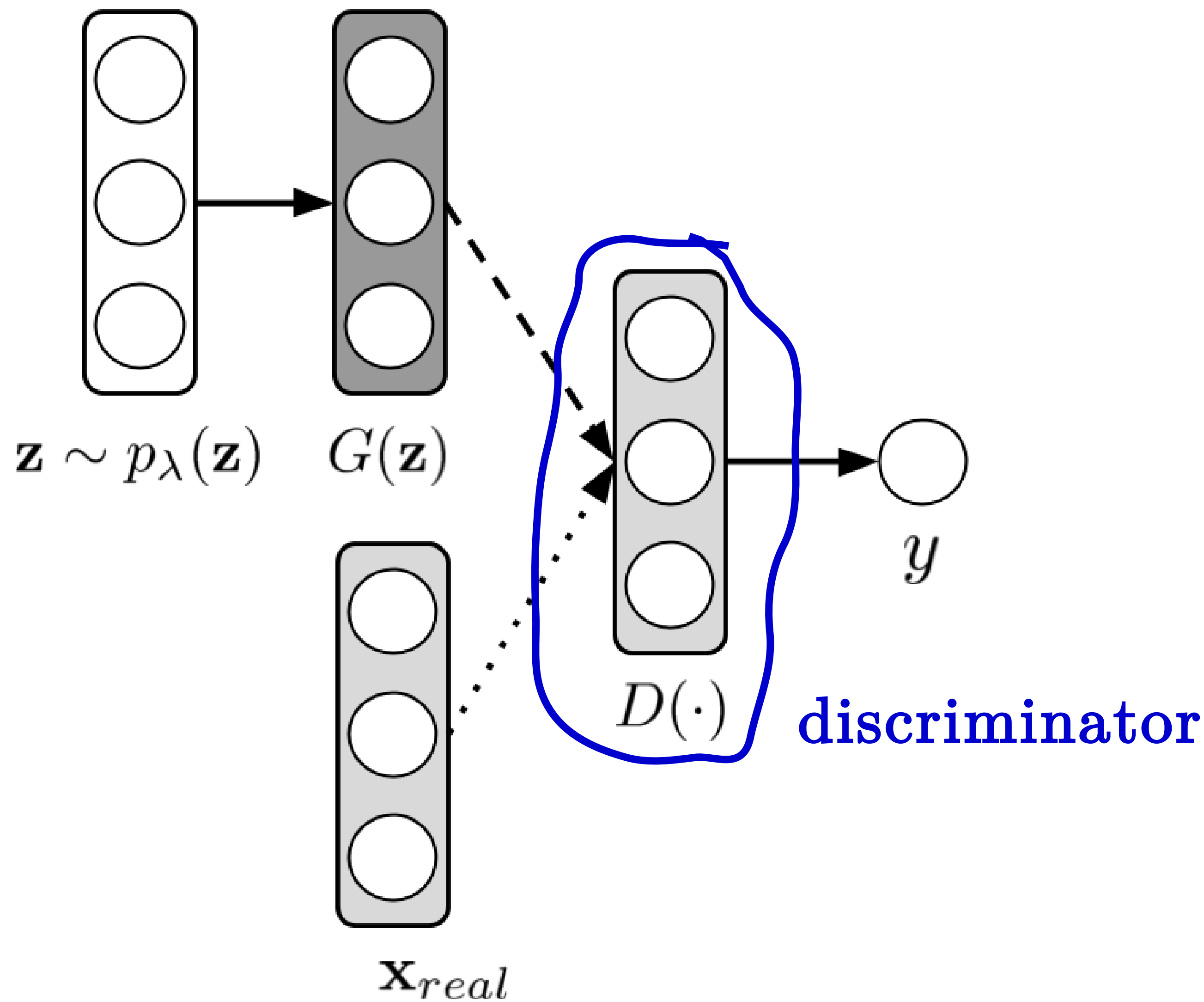
Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets



Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets



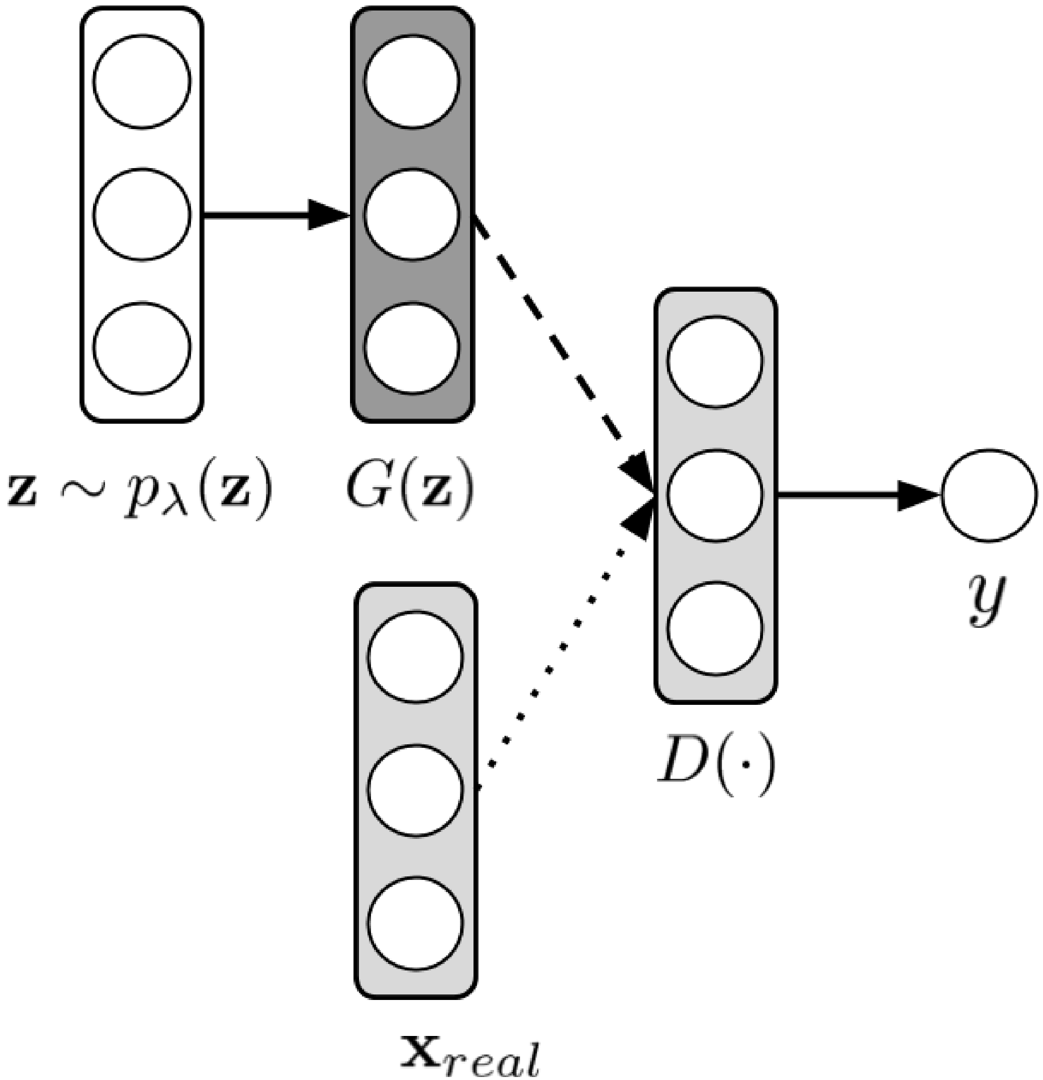
Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

1. Sample **z**.

2. Generate $G(\mathbf{z})$.

3. Discriminate whether given image is **real** or **fake**.



Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Formally, the problem is the following:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{real}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Formally, the problem is the following:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{real}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Minimize wrt. generator

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Formally, the problem is the following:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{real}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Maximize wrt. discriminator

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Formally, the problem is the following:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{real}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Once we converge, we can generate images that are almost **indistinguishable** from real images.

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Formally, the problem is the following:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{real}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Once we converge, we can generate images that are almost **indistinguishable** from real images.

BUT training is very unstable...

Goodfellow, I., et al. (2014). *Generative adversarial nets*. In Advances in neural information processing systems

DGM: Generative Adversarial Nets

Training Data | Samples

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. In Advances in Neural Information Processing Systems (pp. 2234-2242).

DGM: GANs

PROS

Allows **implicit** models

Easy **sampling**

Training using **gradient-based methods**

Works in **high dim. cases**

CONS

Unstable training

Does not correspond to likelihood solution

No clear way for **quantitative assessment**

Missing mode problem

DGM: Wasserstein GAN

We can consider an earth-mover distance to formulate GAN-like optimization problem as follows:

$$\min_{G} \max_{D \in \mathcal{W}} \mathbb{E}_{\mathbf{x} \sim p_{real}} [D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [D(G(\mathbf{z}))]$$

where the discriminator is a 1-Lipshitz function.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. arXiv preprint arXiv:1701.07875.

DGM: Wasserstein GAN

We can consider an earth-mover distance to formulate GAN-like optimization problem as follows:

$$\min_{G} \max_{D \in \mathcal{W}} \mathbb{E}_{\mathbf{x} \sim p_{real}} [D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [D(G(\mathbf{z}))]$$

where the discriminator is a 1-Lipshitz function.

It means we need to **clip** weights of the discriminator, *i.e.*,

`clip(weights, -c, c)`.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. arXiv preprint arXiv:1701.07875.

DGM: Wasserstein GAN

We can consider an earth-mover distance to formulate GAN-like optimization problem as follows:

$$\min_{G} \max_{D \in \mathcal{W}} \mathbb{E}_{\mathbf{x} \sim p_{real}} [D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [D(G(\mathbf{z}))]$$

where the discriminator is a 1-Lipshitz function.

Wasserstein GAN **stabilizes training** (but other problems **remain**).

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. arXiv preprint arXiv:1701.07875.

DGM: More GANs (selected)

Deep convolutional generative adversarial networks

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Auxiliary classifier GANs

Odena, A., Olah, C., & Shlens, J. (2016). Conditional image synthesis with auxiliary classifier gans. arXiv preprint arXiv:1610.09585.

From optimal transport to generative modeling: the VEGAN cookbook

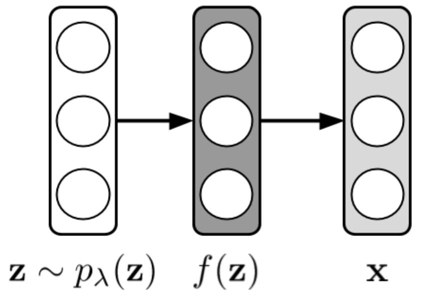
Bousquet, O., Gelly, S., Tolstikhin, I., Simon-Gabriel, C. J., & Schoelkopf, B. (2017). From optimal transport to generative modeling: the VEGAN cookbook. arXiv preprint arXiv:1705.07642.

Bidirectional Generative Adversarial Networks

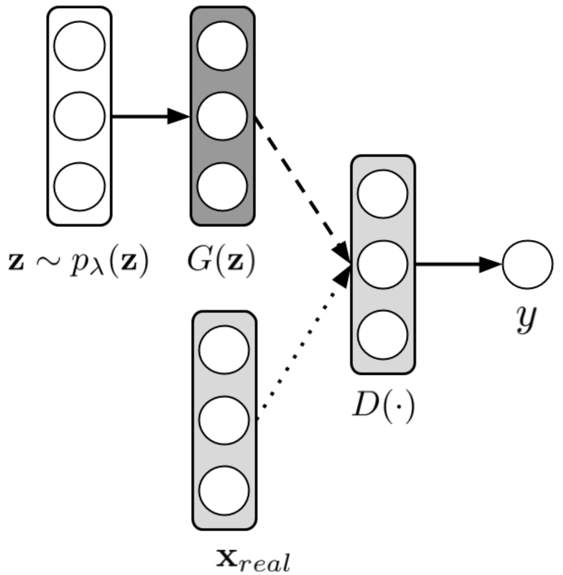
Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. arXiv preprint arXiv:1605.09782.

Questions?

DGM: so far we have

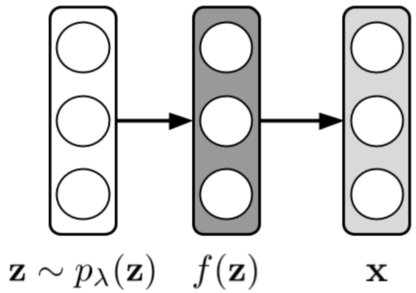


Density Network

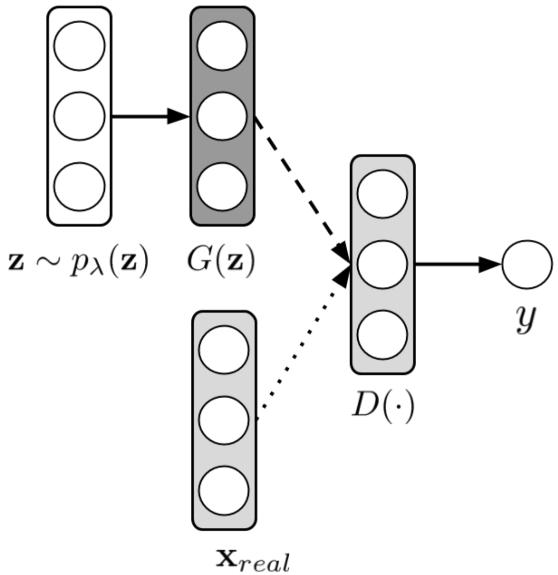


Generative Adversarial Net

DGM: so far we have



Density Network

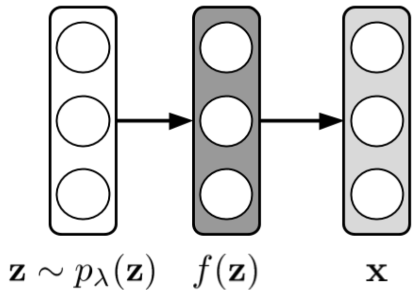


Generative Adversarial Net

Works only for low dim. cases...

Inefficient training...

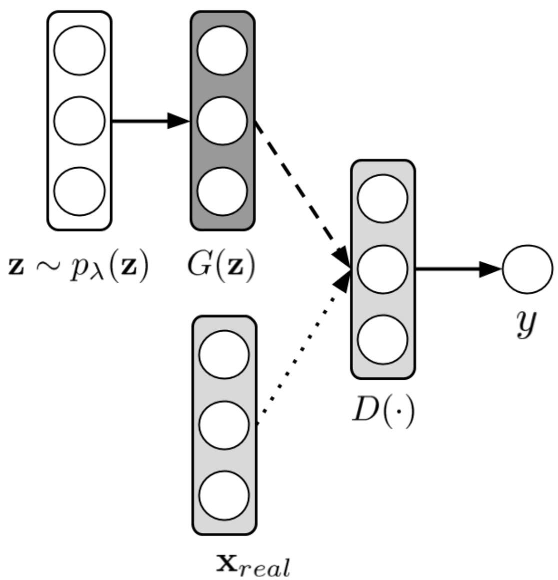
DGM: so far we have



Density Network

Works only for low dim. cases...

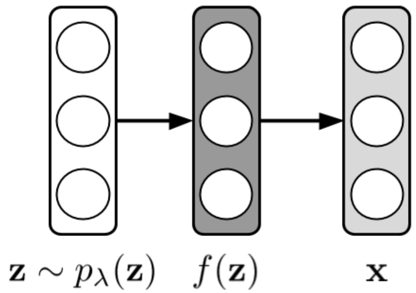
Inefficient training...



Generative Adversarial Net

Works for high dim. cases!

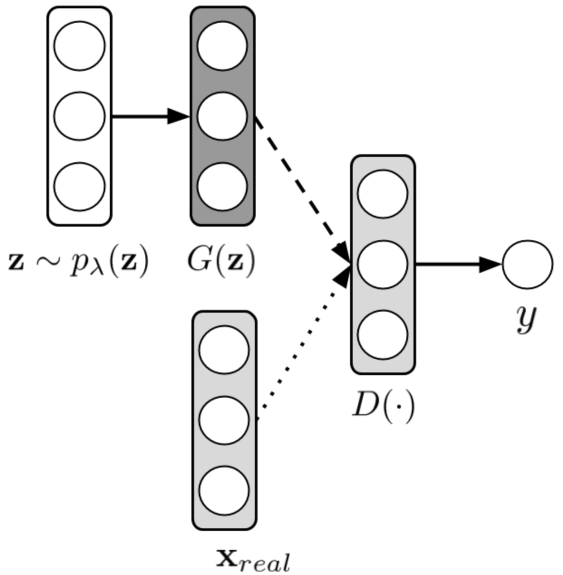
DGM: so far we have



Density Network

Works only for low dim. cases...

Inefficient training...



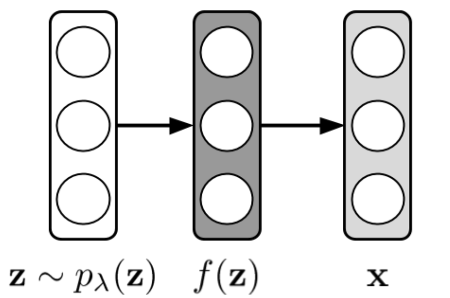
Generative Adversarial Net

Works for high dim. cases!

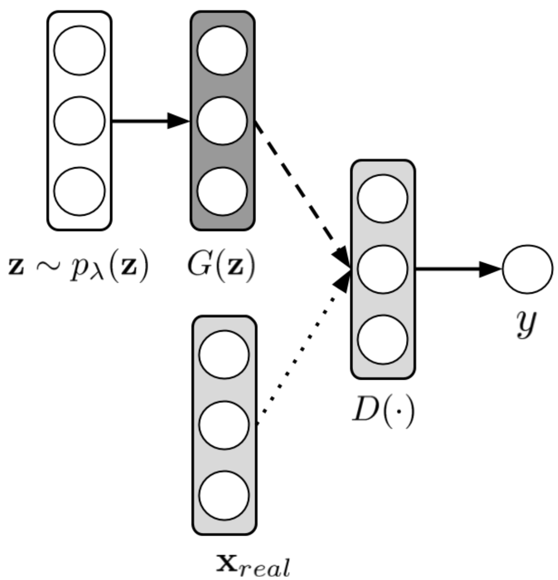
Doesn't train a distribution...

Unstable training...

DGM: so far we have



Density Network

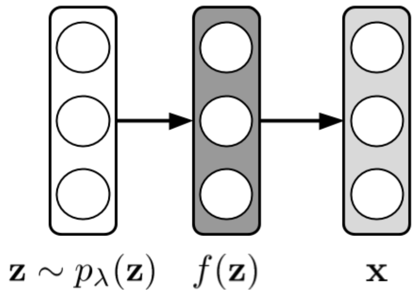


Generative Adversarial Net

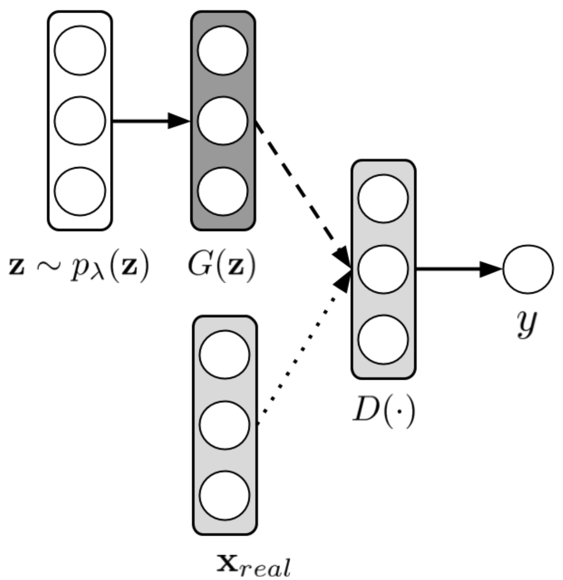
QUESTION

Can we stick to the log-likelihood approach but with a simple training procedure?

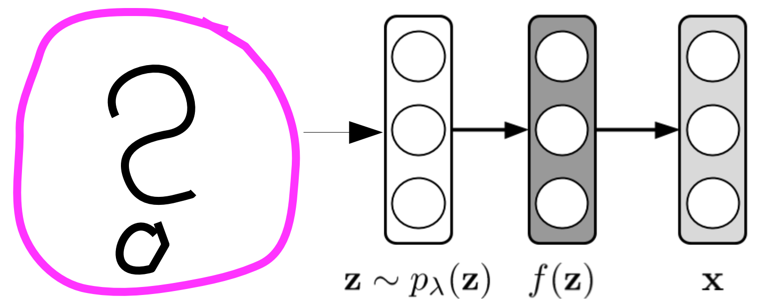
DGM: so far we have



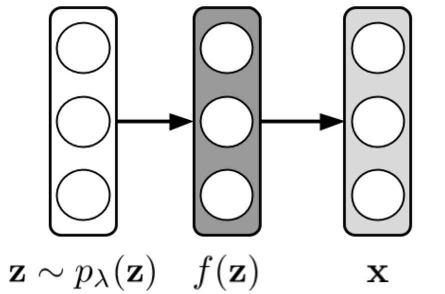
Density Network



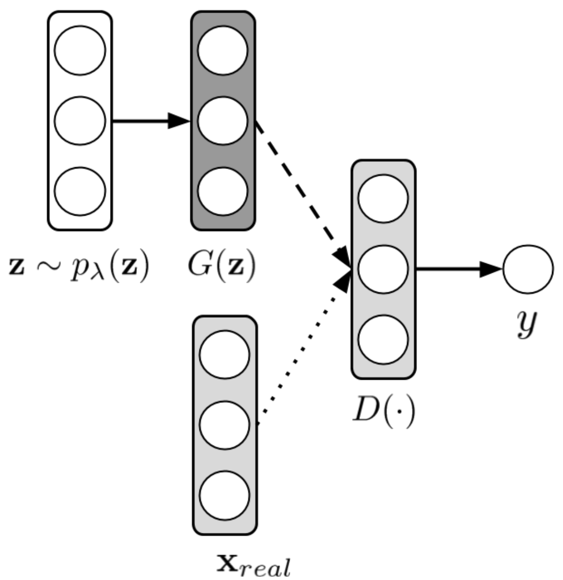
Generative Adversarial Net



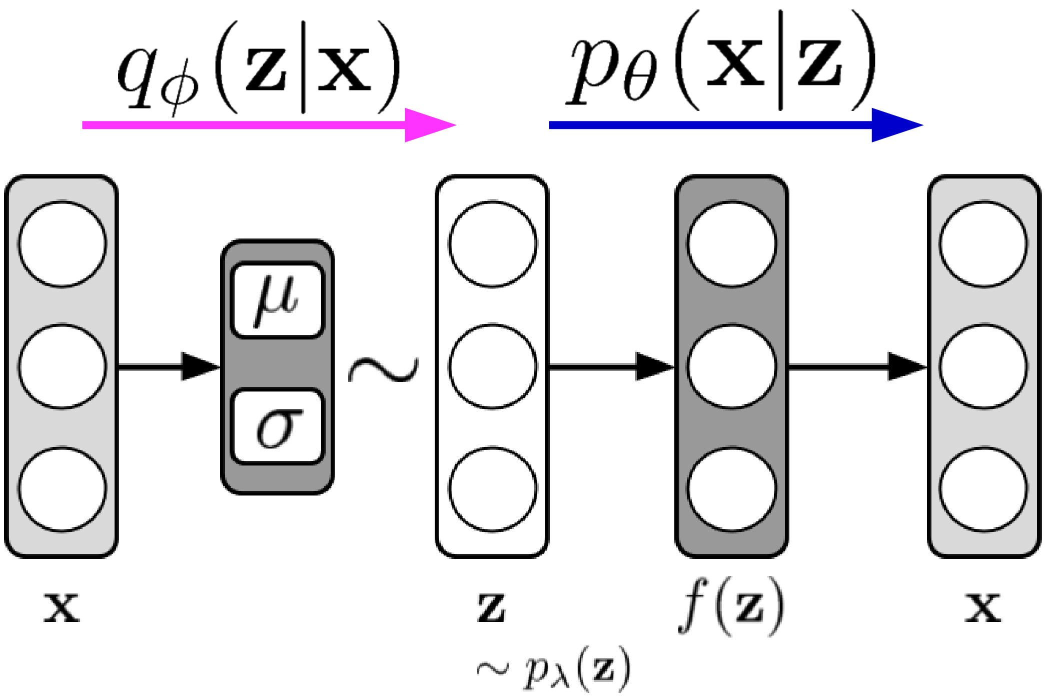
DGM: Variational Auto-Encoder



Density Network



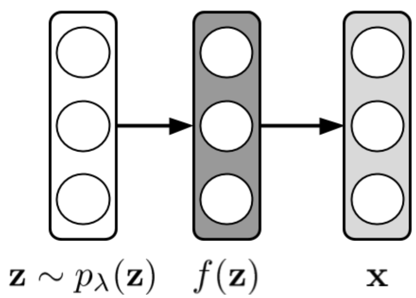
Generative Adversarial Net



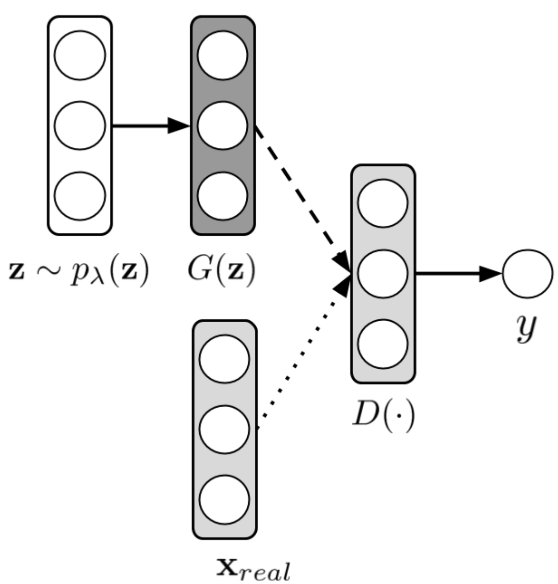
Variational Auto-Encoder

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

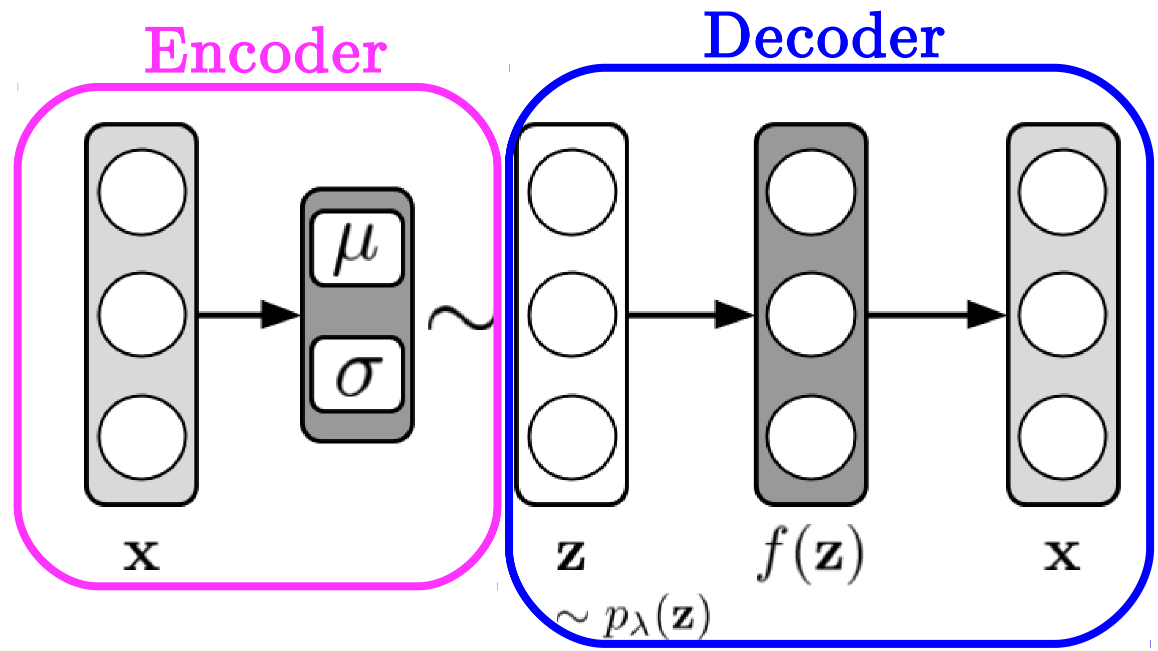
DGM: Variational Auto-Encoder



Density Network



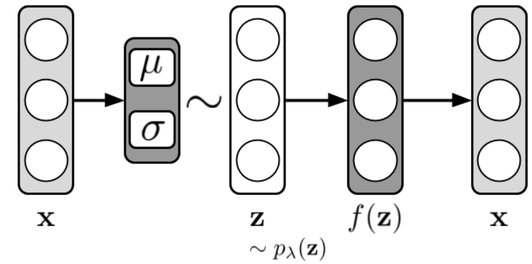
Generative Adversarial Net



Variational Auto-Encoder

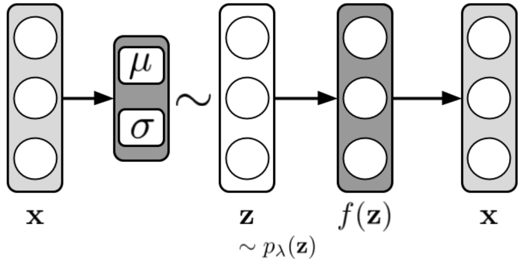
Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder



$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]
\end{aligned}$$

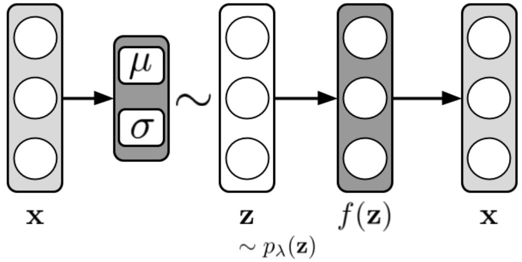
DGM: Variational Auto-Encoder



Variational posterior

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]
\end{aligned}
$$

DGM: Variational Auto-Encoder



$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\ p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\ \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x})\ \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\ \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]}_{\text{Regularization}}
\end{aligned}
$$

DGM: Variational Auto-Encoder

Our objective it **the evidence lower bound.**

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]$$

We can approximate it using **MC sample**.

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^{S} [\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

DGM: Variational Auto-Encoder

Our objective it **the evidence lower bound.**

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})]$$

We can approximate it using **MC sample**.

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

How to properly calculate gradients (*i.e.*, train the model)?

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_{\theta}(\mathbf{x}|\mathbf{z}_s) - \log q_{\phi}(\mathbf{z}_s|\mathbf{x}) + \log p_{\lambda}(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

SOLUTION: use a non-centered parameterization (a.k.a. ***reparameterization trick***).

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu, \sigma^2)$$

$$\mathbf{z}_s = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

DGM: Variational Auto-Encoder

$$\mathcal{L}(\mathbf{x}) \approx \frac{1}{S}\sum_{s=1}^{S}[\log p_\theta(\mathbf{x}|\mathbf{z}_s) - \log q_\phi(\mathbf{z}_s|\mathbf{x}) + \log p_\lambda(\mathbf{z}_s)]$$

PROBLEM: calculating gradient wrt parameters of the variational posterior (*i.e.*, sampling process).

SOLUTION: use a non-centered parameterization (a.k.a. ***reparameterization trick***).

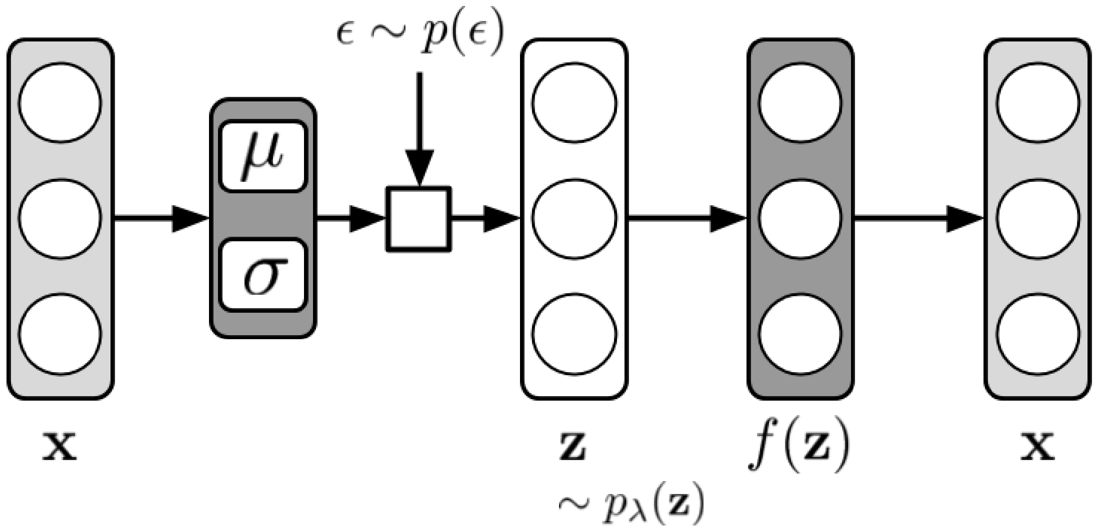
$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu, \sigma^2)$$

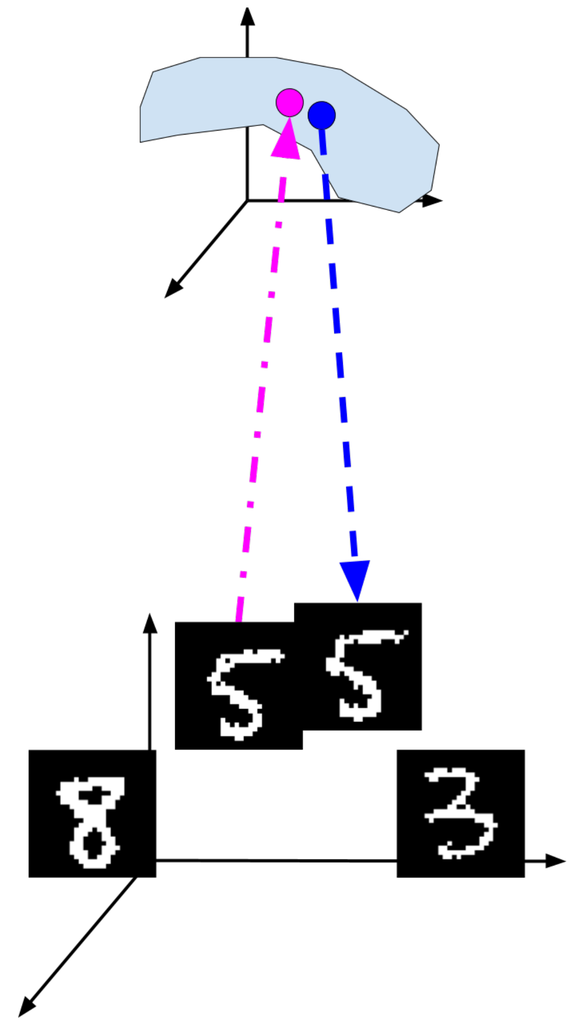
$$\mathbf{z}_s = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

Output of a neural network

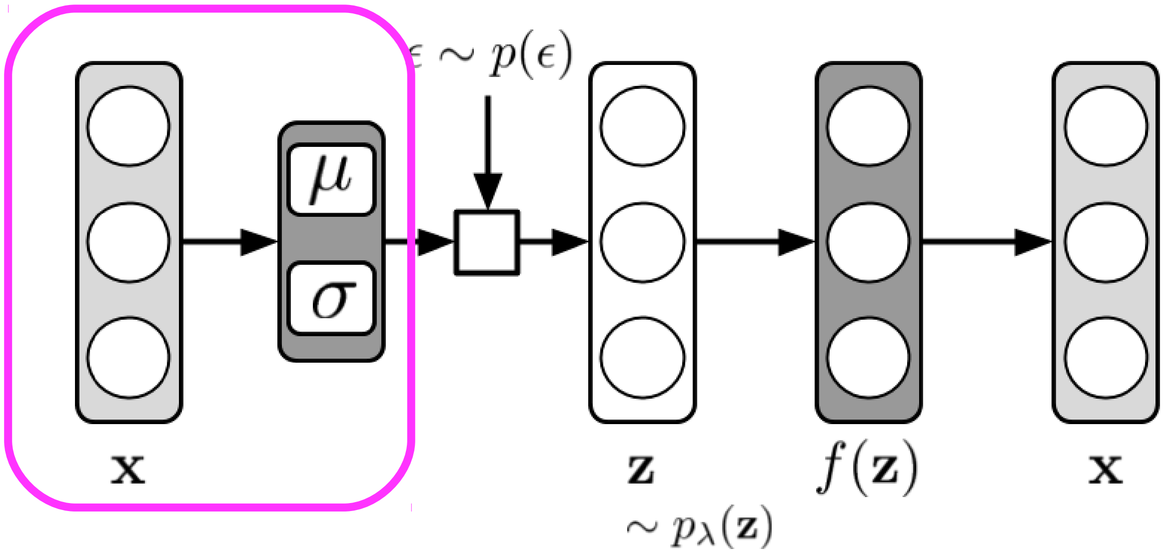
Kingma, D. P., & Welling, M. (2013). Auto-encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

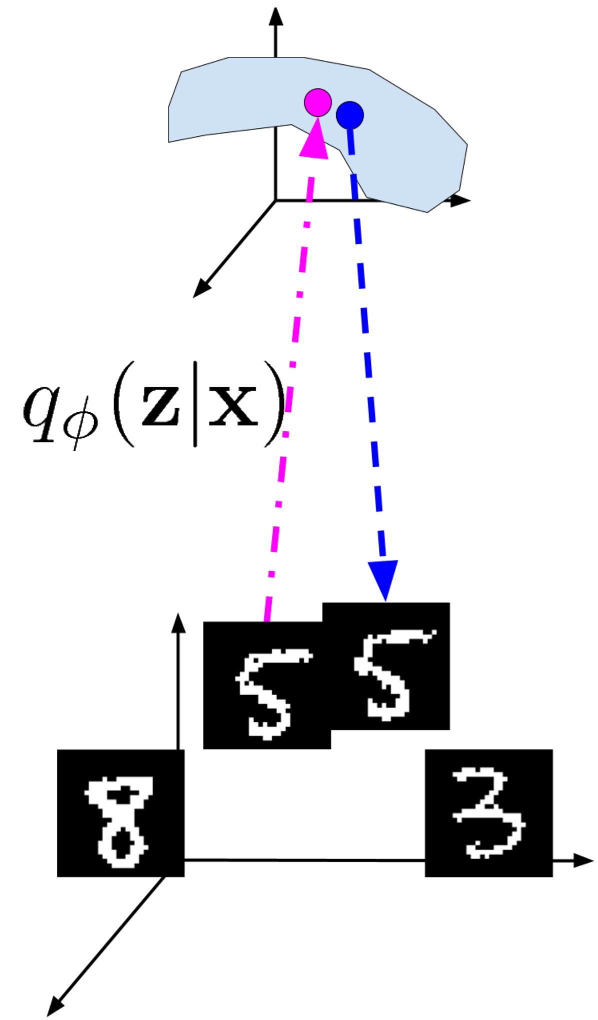
DGM: Variational Auto-Encoder





DGM: Variational Auto-Encoder

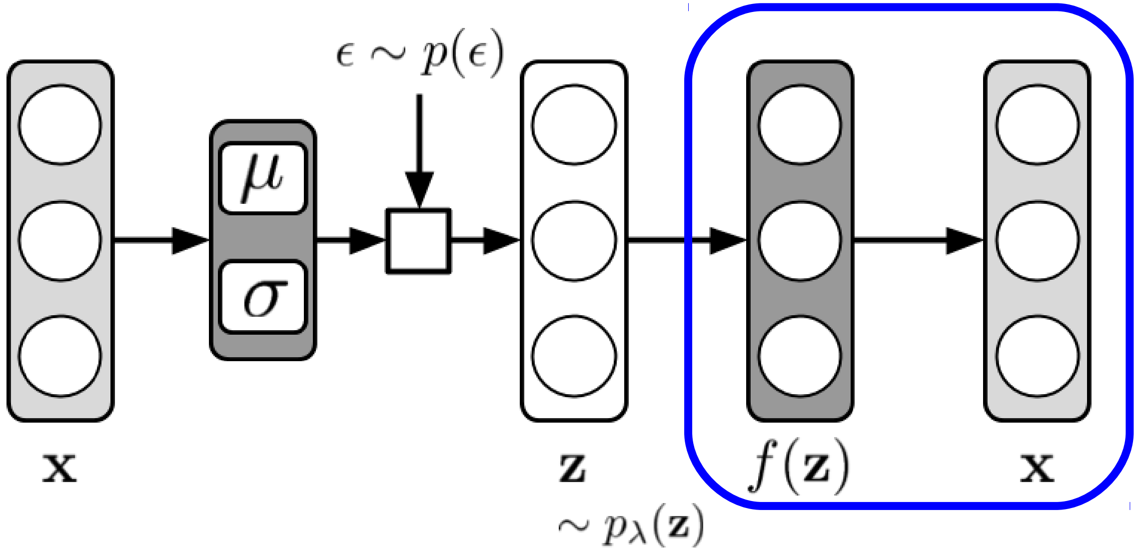


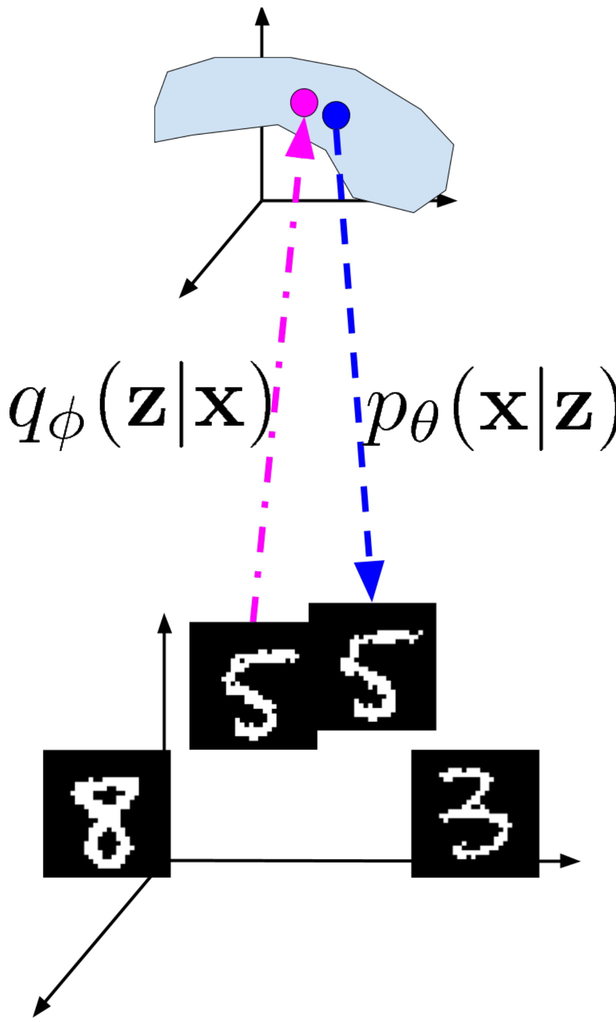


A **deep neural net** that outputs parameters of the variational posterior (**encoder**):

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}\{\sigma^2(\mathbf{x})\})$$

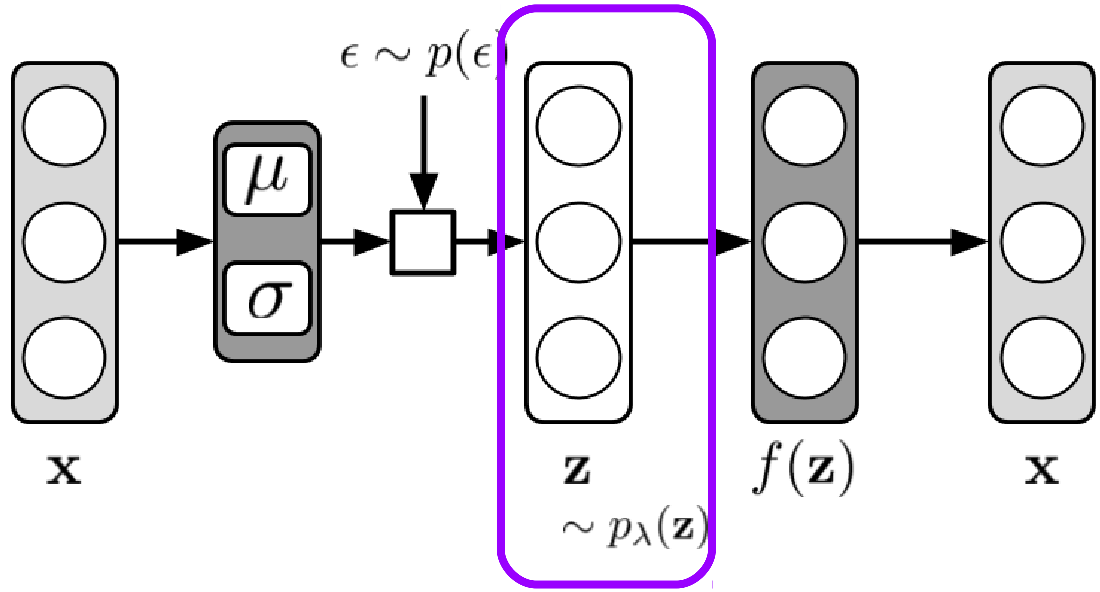
DGM: Variational Auto-Encoder

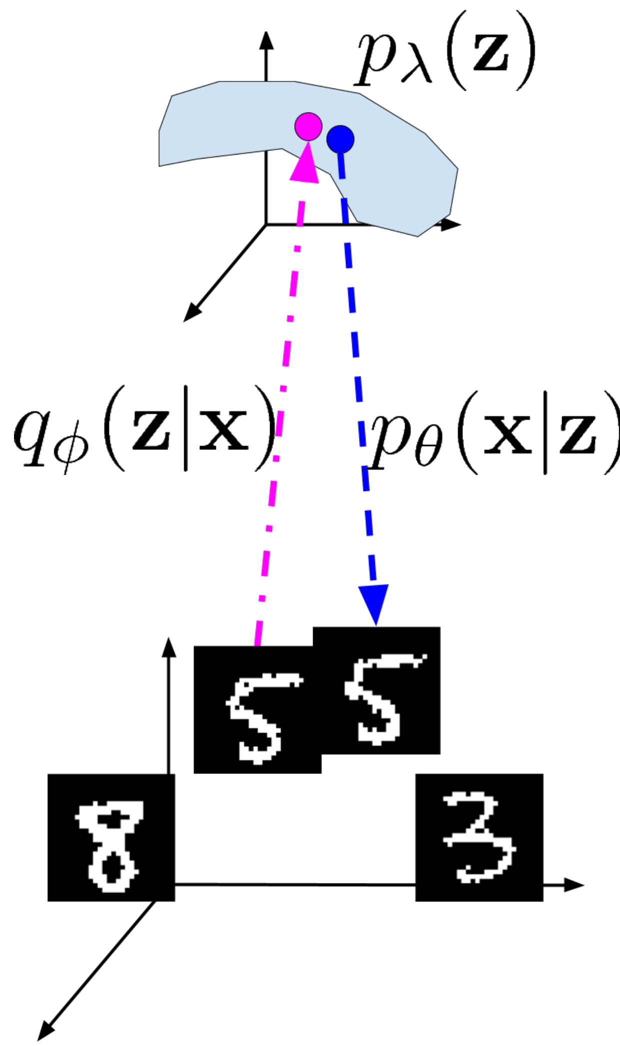




A **deep neural net** that outputs parameters of the generator (**decoder**), *e.g.*, a normal distribution or Bernoulli distribution.

DGM: Variational Auto-Encoder



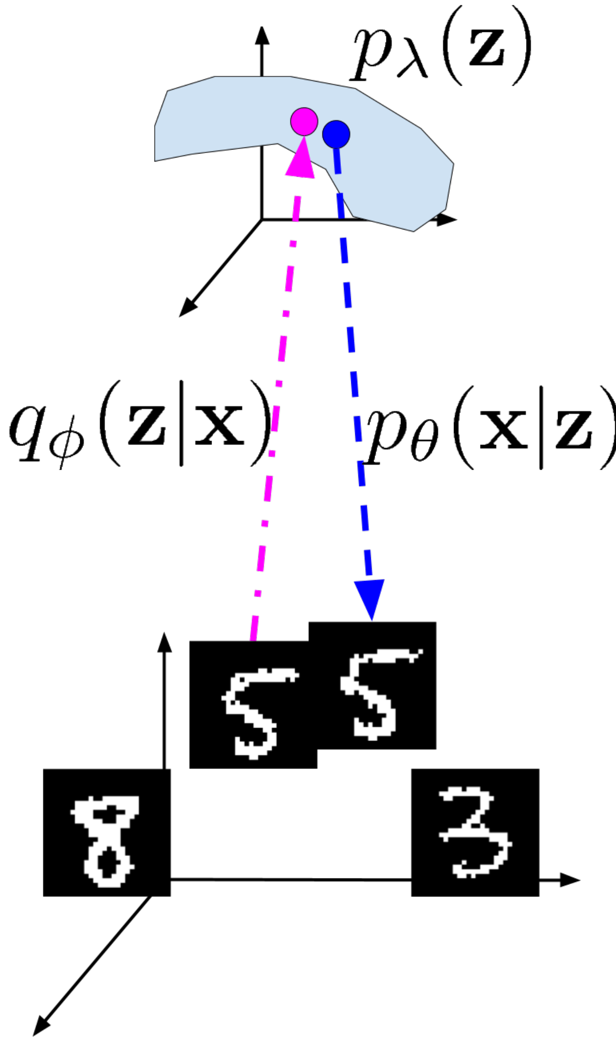


A **prior** that **regularizes** the encoder and takes part in the **generative process**.

$$p_\lambda(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$$

DGM: Variational Auto-Encoder

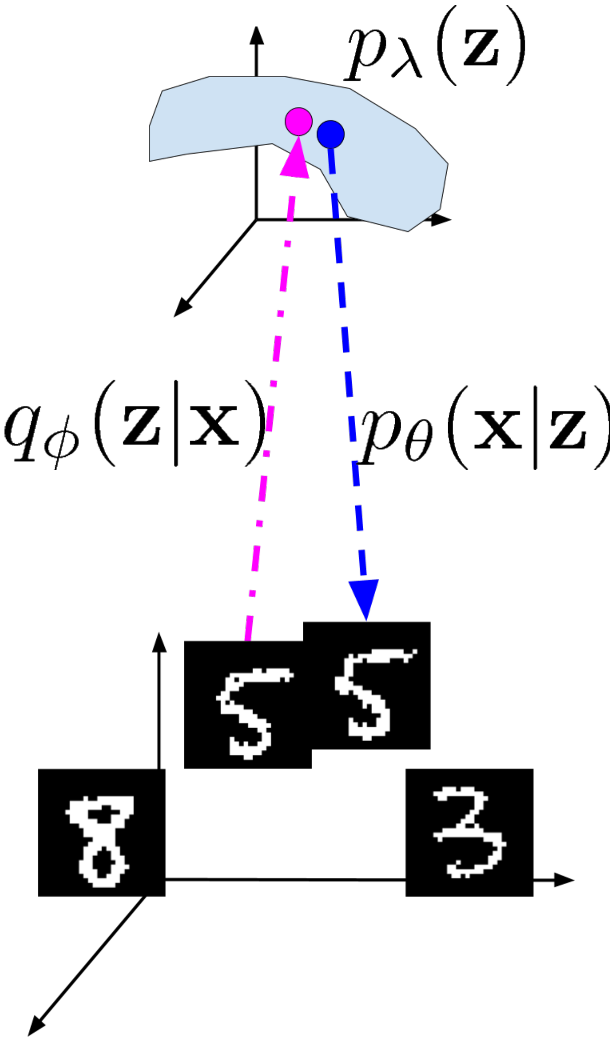
$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$



DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

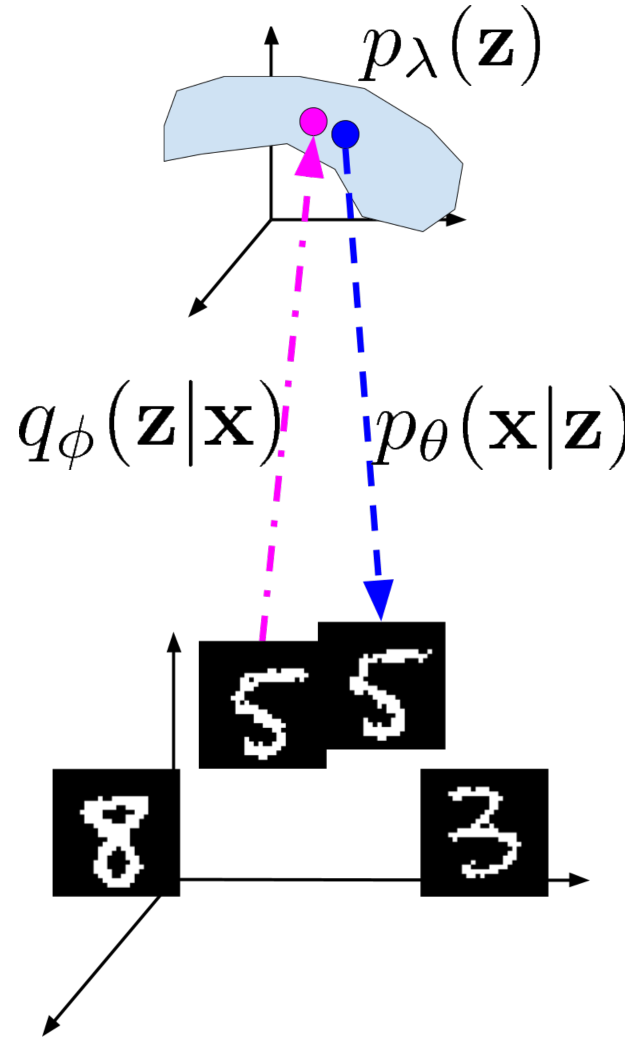


DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN



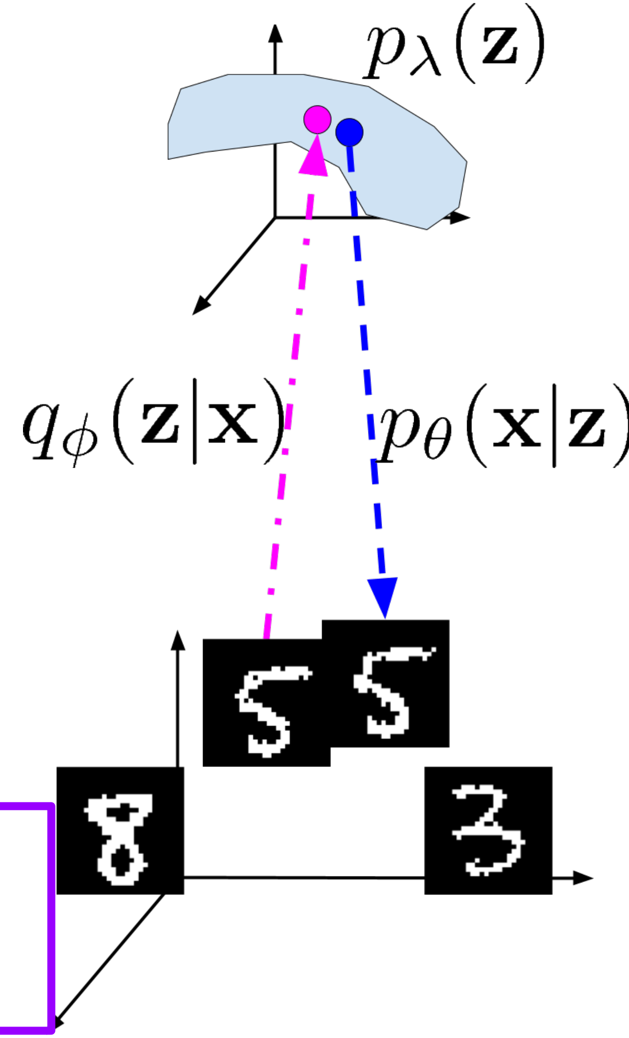
DGM: Variational Auto-Encoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

Auto-regressive Prior
Objective Prior
Stick-Breaking Prior
VampPrior



DGM: Variational Auto-Encoder

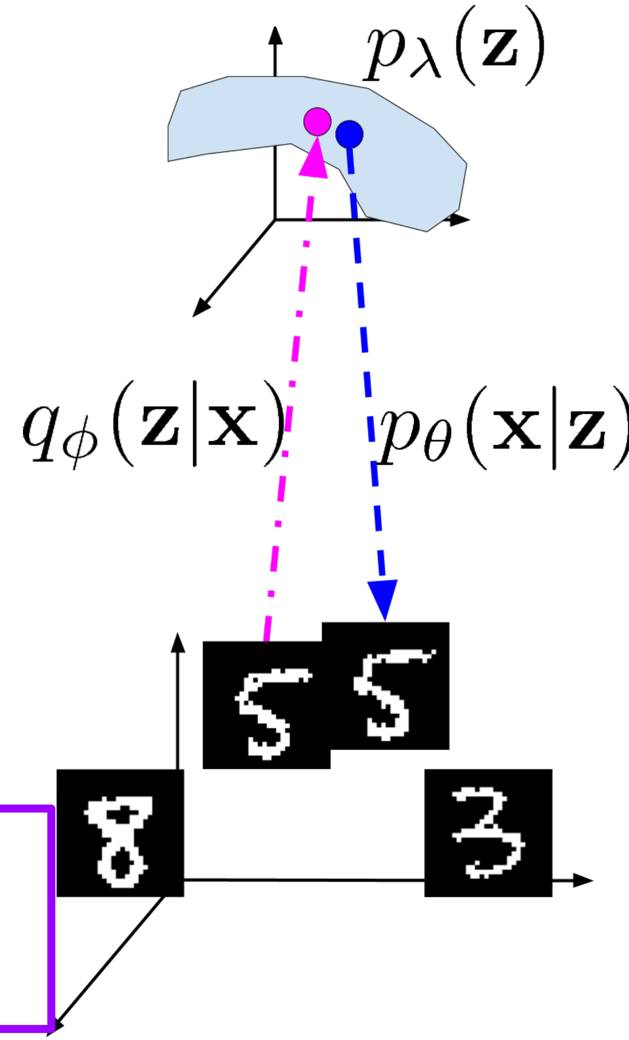
$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})$$

Normalizing flows
Volume-preserving flows
Gaussian processes
Stein Particle Descent
Operator VI

Feedforward nets
Convolutional nets
PixelCNN
Gated PixelCNN

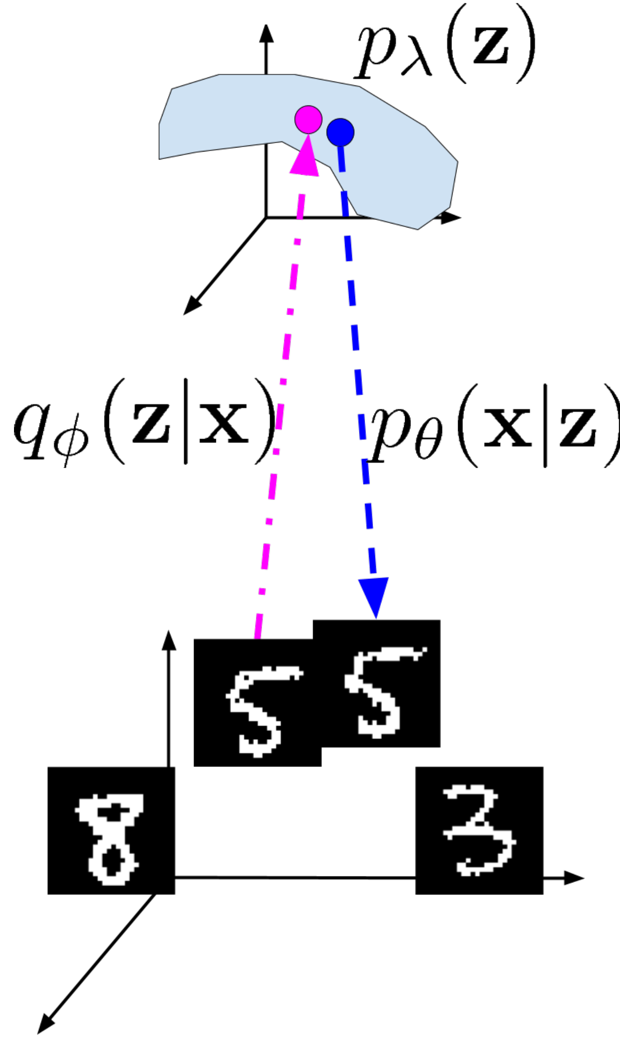
Auto-regressive Prior
Objective Prior
Stick-Breaking Prior
VampPrior

Importance Weighted AE
Renyi Divergence
Stein Divergence



Improving the posterior

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$



Normalizing flows

- Diagonal posterior – **insufficient** and **inflexible**.

- How to get more flexible posterior?
  → apply a series of T **invertible transformations**
  $\mathbf{f}^{(t)}$ to $\mathbf{z}^{(0)} \sim q(\mathbf{z}|\mathbf{x})$.

- **New objective:**

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}\Big[\ln p(\mathbf{x}|\mathbf{z}^{(T)}) + \sum_{t=1}^{T} \ln \Big|\det \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{z}^{(t-1)}}\Big|\Big] - \mathrm{KL}\big(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(T)})\big).$$

Normalizing flows

- Diagonal posterior – **insufficient** and **inflexible**.

- How to get more flexible posterior?
  $\rightarrow$ apply a series of T **invertible transformations** $\mathbf{f}^{(t)}$ to $\mathbf{z}^{(0)} \sim q(\mathbf{z}|\mathbf{x})$.

- New objective:

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}\Big[\ln p(\mathbf{x}|\mathbf{z}^{(T)}$$

$$q(\mathbf{z}^{(T)}|\mathbf{x}) = q(\mathbf{z}^{(0)}|\mathbf{x}) \prod_{t=1}^{T} \left| \det \frac{\partial f^{(t)}}{\partial \mathbf{z}^{(t-1)}} \right|^{-1} .$$

Normalizing flows

- Diagonal posterior – **insufficient** and **inflexible**.

- How to get more flexible posterior?
  $\rightarrow$ apply a series of T **invertible transformations** $\mathbf{f}^{(t)}$ to $\mathbf{z}^{(0)} \sim q(\mathbf{z}|\mathbf{x})$.

- **New objective**:

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}\left[\ln p(\mathbf{x}|\mathbf{z}^{(T)}) + \sum_{t=1}^{T} \ln \left|\det \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{z}^{(t-1)}}\right|\right] - \mathrm{KL}\big(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(T)})\big).$$

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770. ICML 2015

Normalizing flows

- Diagonal posterior – **insufficient** and **inflexible**.

- How to get more flexible posterior?

  $\rightarrow$ apply a series of T **invertible transformations**

  $\mathbf{f}^{(t)}$ to $\mathbf{z}^{(0)} \sim q(\mathbf{z}|\mathbf{x})$.

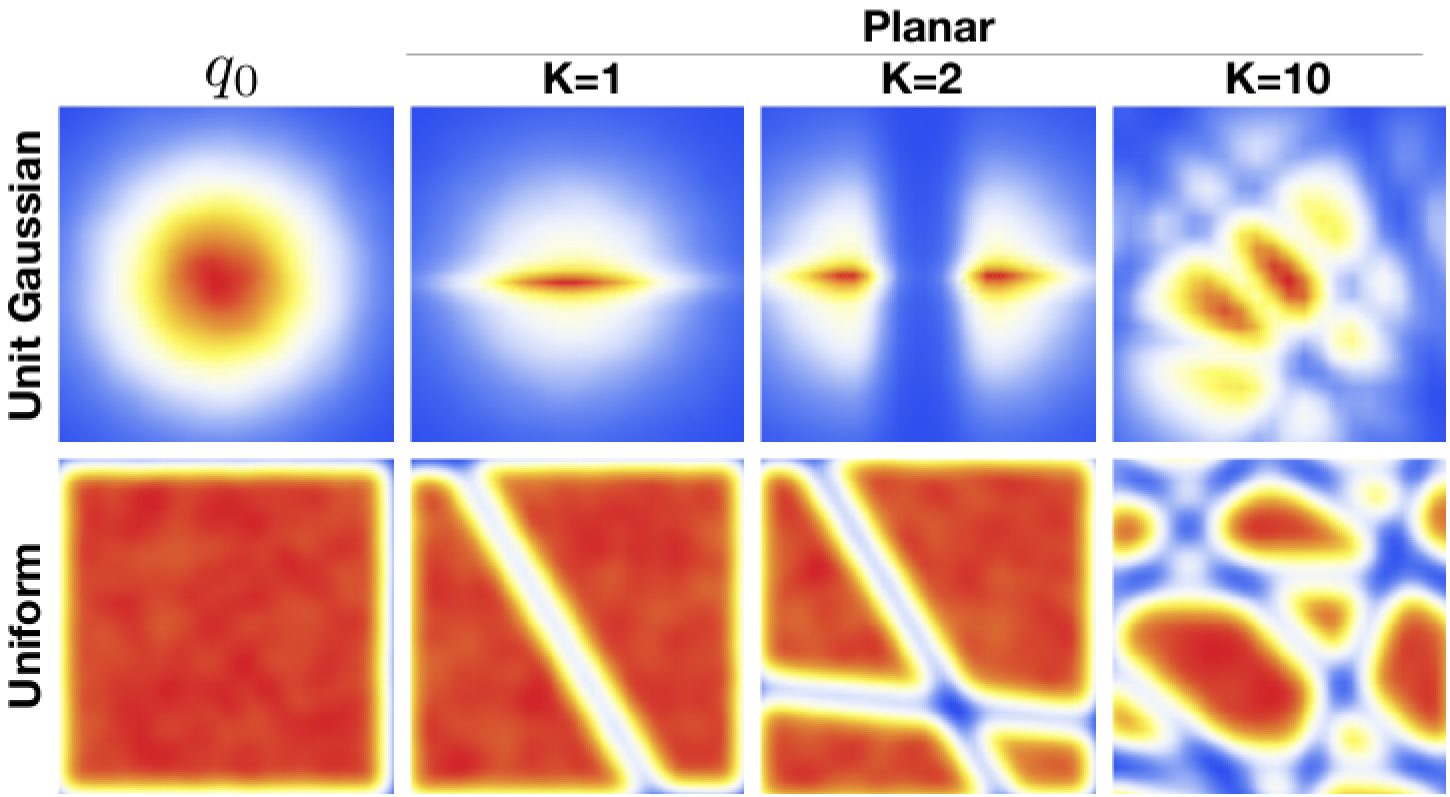
- **New objective**:

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}^{(0)}|\mathbf{x})}\left[\ln p(\mathbf{x}|\mathbf{z}^{(T)}) + \sum_{t=1}^{T} \ln \left|\det \frac{\partial \mathbf{f}^{(t)}}{\partial \mathbf{z}^{(t-1)}}\right|\right] - \mathrm{KL}\big(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(T)})\big).$$

Jacobian-determinant: (i) **general normalizing flow** (|det J| is ***easy*** to compute)

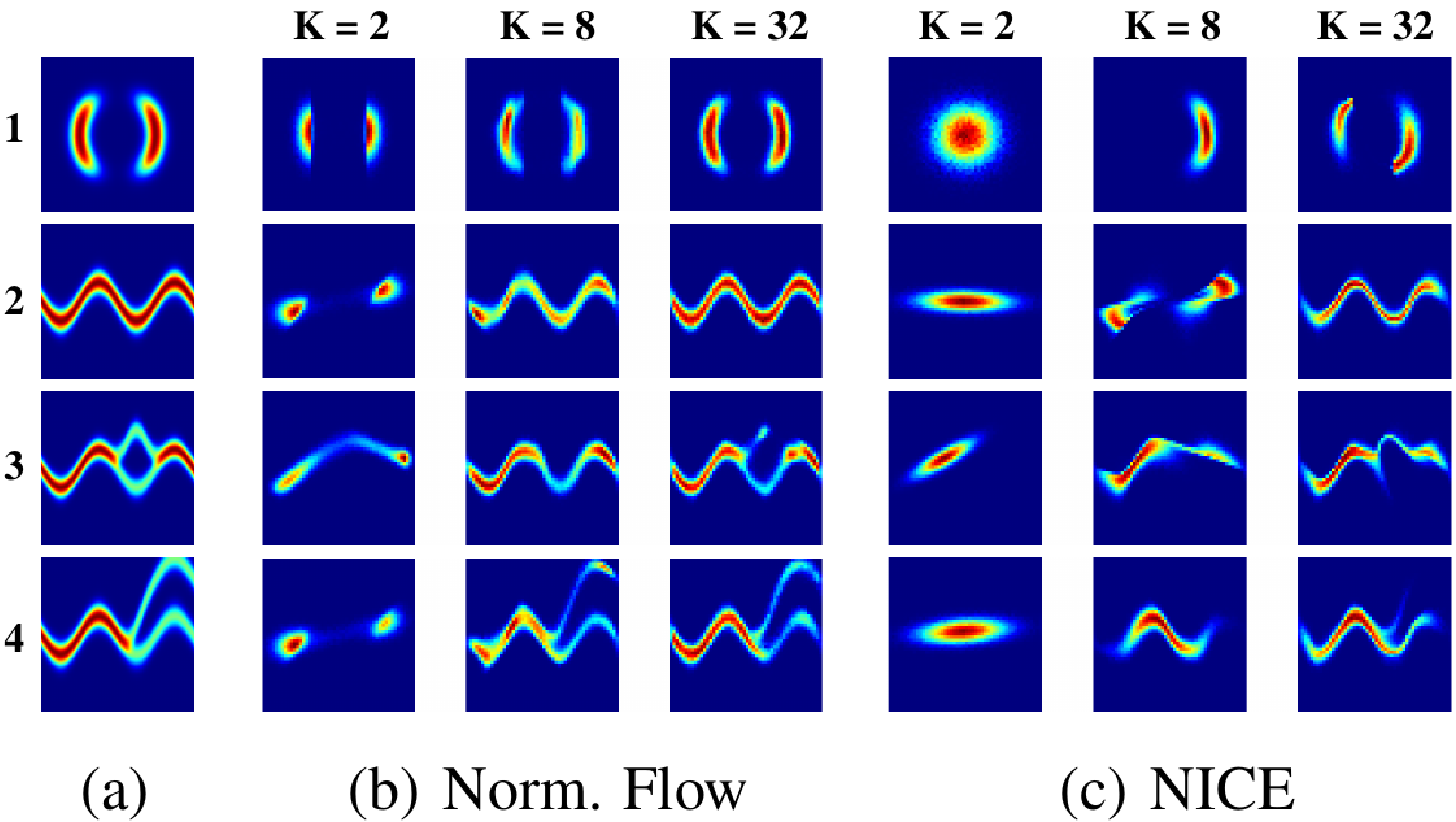
(ii) **volume-preserving flow**, *i.e.*, |det J|=1

Normalizing Flow



Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770. ICML 2015

Normalizing Flow



Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770. ICML 2015

Extensions of normalizing flows

- How to obtain more **flexible** posterior and preserve **$|\det \mathrm{J}|=1$**?
  - using ***orthogonal matrices*** → **Householder flow**

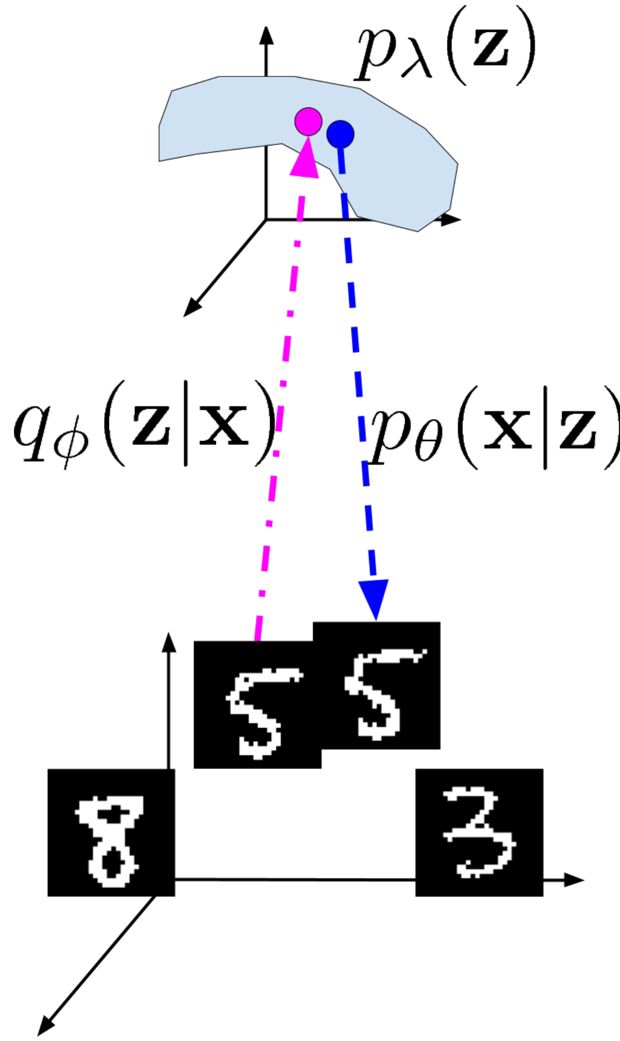
    Tomczak, J. M., & Welling, M. (2016). Improving Variational Inference with Householder Flow. arXiv preprint arXiv:1611.09630. NIPS Workshop on Bayesian Deep Learning 2016

- **General** normalizing flow:
  - using ***autoregressive model*** → **Inverse Autoregressive Flow**

    Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improving Variational Inference with Inverse Autoregressive Flow. NIPS 2016

Improving the decoder

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z}) p_\lambda(\mathbf{z})$$

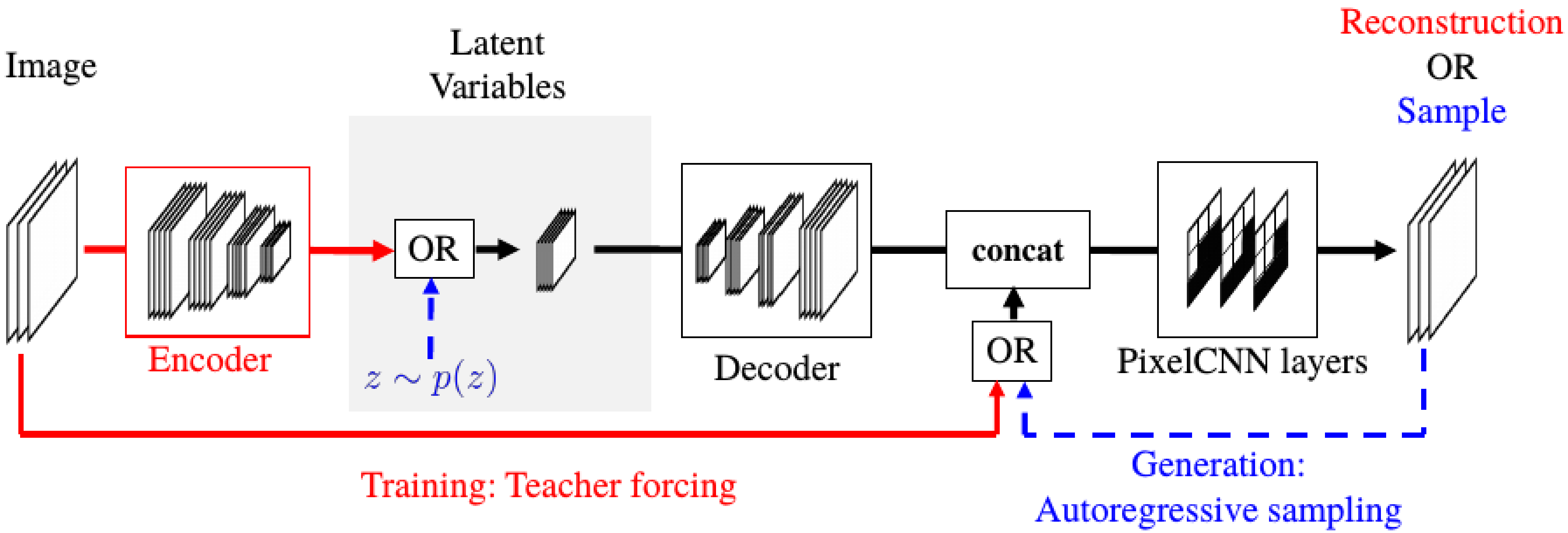


Improving the decoder

- Dependency only on $\mathbf{z}$ – **missing correlations**.
- How to get more flexible decoderposterior?
  $\rightarrow$ apply **autoregressive model**

$$p(\mathbf{x}|\mathbf{z}) = p(x_1|\mathbf{z}) \prod_{d=2}^{D} p(x_d|x_1, \ldots, x_{d-1}, \mathbf{z})$$

PixelVAE (PixelCNN + VAE)



Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., & Courville, A. (2016). PixelVAE: A latent variable model for natural images. arXiv preprint arXiv:1611.05013.

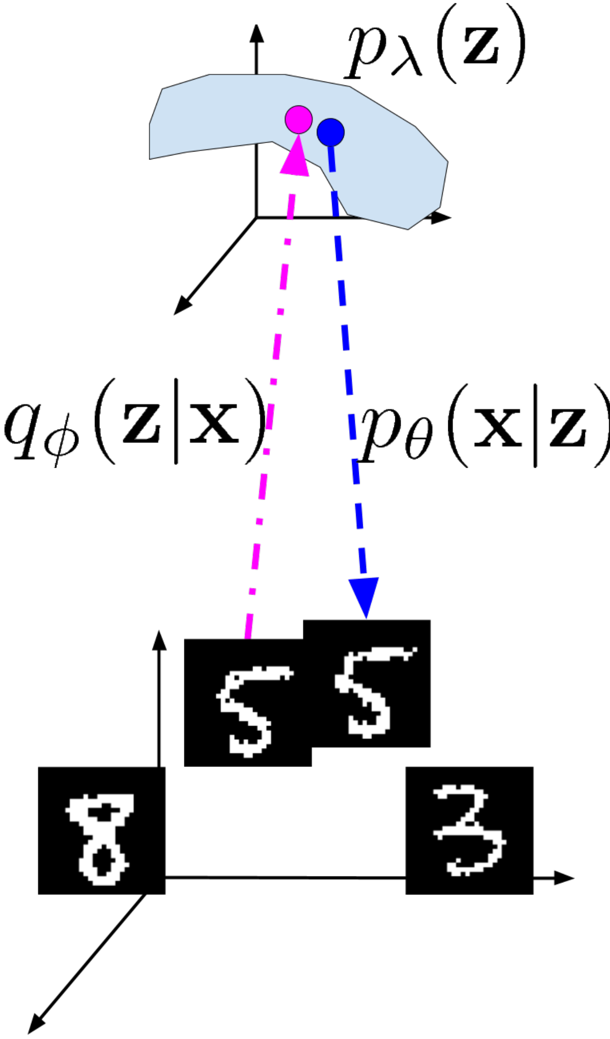
PixelVAE (PixelCNN + VAE)

Figure 6: Samples from hierarchical PixelVAE on the 64x64 ImageNet dataset.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., & Courville, A. (2016). PixelVAE: A latent variable model for natural images. arXiv preprint arXiv:1611.05013.

Improving the prior

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$



Improving the prior

- Standard normal prior – **unimodal, too restrictive**.

- How to get more flexible prior?

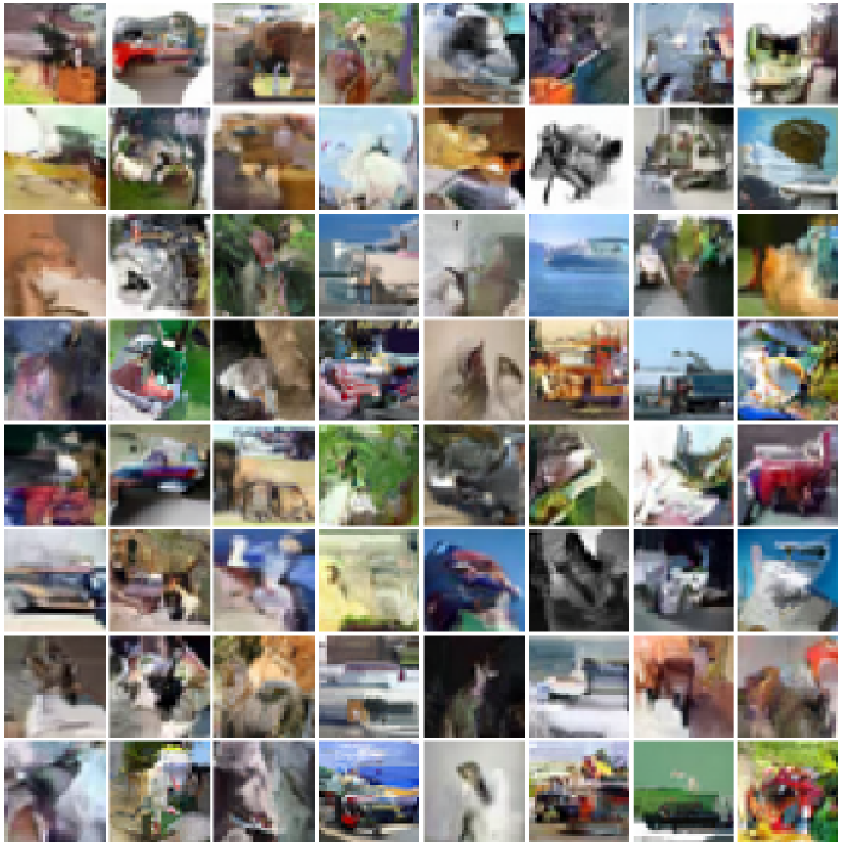
  $\rightarrow$ apply **autoregressive prior**

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., ... & Abbeel, P. (2016). Variational lossy autoencoder. arXiv preprint arXiv:1611.02731.

  $\rightarrow$ apply **variational mixture of posteriors (VampPrior)**

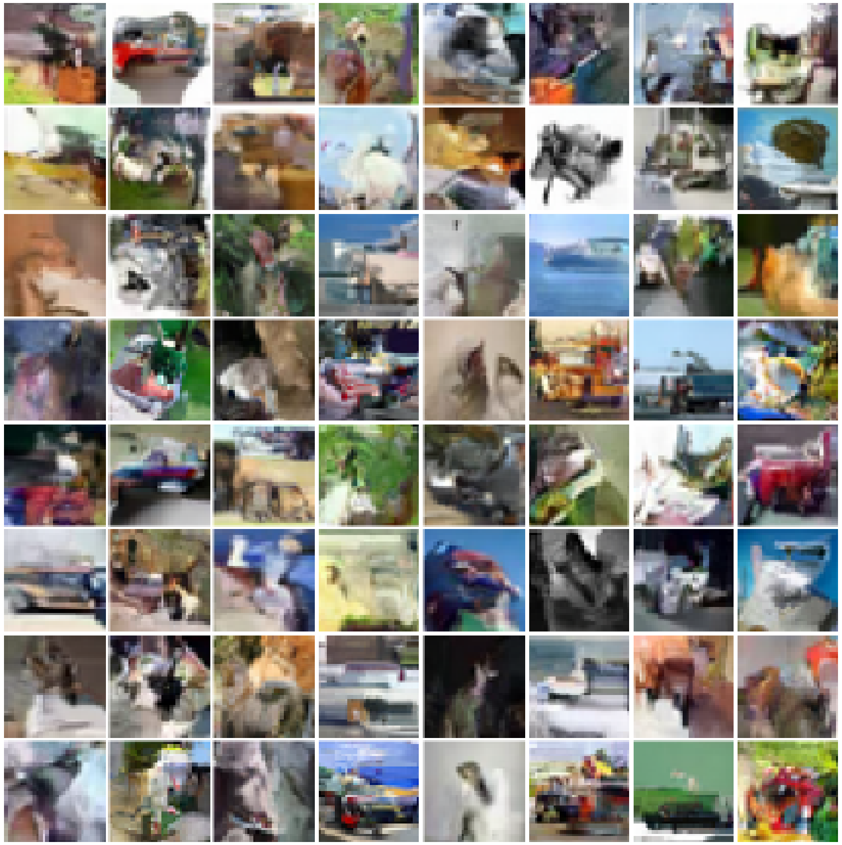
Tomczak, J. M., & Welling, M. (2017). VAE with a VampPrior. arXiv preprint arXiv:1705.07120.

Autoregressive prior



Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., ... & Abbeel, P. (2016). Variational lossy autoencoder. arXiv preprint arXiv:1611.02731.

Autoregressive prior



Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., ... & Abbeel, P. (2016). Variational lossy autoencoder. arXiv preprint arXiv:1611.02731.

VampPrior

(a) real data (b) VAE (c) HVAE + VampPrior (d) convHVAE + VampPrior

Tomczak, J. M., & Welling, M. (2017). VAE with a VampPrior. arXiv preprint arXiv:1705.07120.

Some extensions and applications of VAE

- Semi-supervised learning with VAE.

  Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). *Semi-supervised learning with deep generative models*. NIPS

- VAE for sequences.

  Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). *Generating sentences from a continuous space*. arXiv preprint arXiv:1511.06349.

  Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). *A recurrent latent variable model for sequential data*. NIPS

- More powerful decoders (using PixelCNN).

  Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., & Courville, A. (2016). *PixelVAE: A latent variable model for natural images*. arXiv preprint arXiv:1611.05013.

  Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., ... & Abbeel, P. (2016). *Variational lossy autoencoder*. arXiv preprint arXiv:1611.02731.

Some extensions and applications of VAE

- Applications: graph data

Kipf, T. N., & Welling, M. (2016). *Variational Graph Auto-Encoders*. arXiv preprint arXiv:1611.07308. NIPS Workshop

Berg, R. V. D., Kipf, T. N., & Welling, M. (2017). *Graph Convolutional Matrix Completion*. arXiv preprint arXiv:1706.02263.

- Applications: drug response prediction

Rampasek, L., & Goldenberg, A. (2017). *Dr.VAE: Drug Response Variational Autoencoder*. arXiv preprint arXiv:1706.08203.

- Applications: text generation

Yang, Z., Hu, Z., Salakhutdinov, R., & Berg-Kirkpatrick, T. (2017). *Improved Variational Autoencoders for Text Modeling using Dilated Convolutions*. arXiv preprint arXiv:1702.08139.

DGM: VAE

PROS

Log-likelihood framework

Easy **sampling**

Training using **gradient-based methods**

Stable training

Discovers **latent representation**

Could be easily combined with other probabilistic frameworks

CONS

Only **explicit** models

Produces **blurry images(?)**

$$1283 + 1146$$

Number of citations of seminal papers on GANs and VAE.*

**According to GoogleScholar, 26.09.2017*

In order to *make better* ***decisions***, we need a *better understanding* of ***reality***.

=

generative modeling

Web-page:
https://jmtomczak.github.io

Code on github:
https://github.com/jmtomczak

Contact:
J.M.Tomczak@uva.nl
jakubmkt@gmail.com

Part of the presented research was funded by the European Commission within the Marie Skłodowska-Curie Individual Fellowship (Grant No. 702666, ''*Deep learning and Bayesian inference for medical imaging*'').

RESEARCH & INNOVATION
Marie Skłodowska-Curie actions