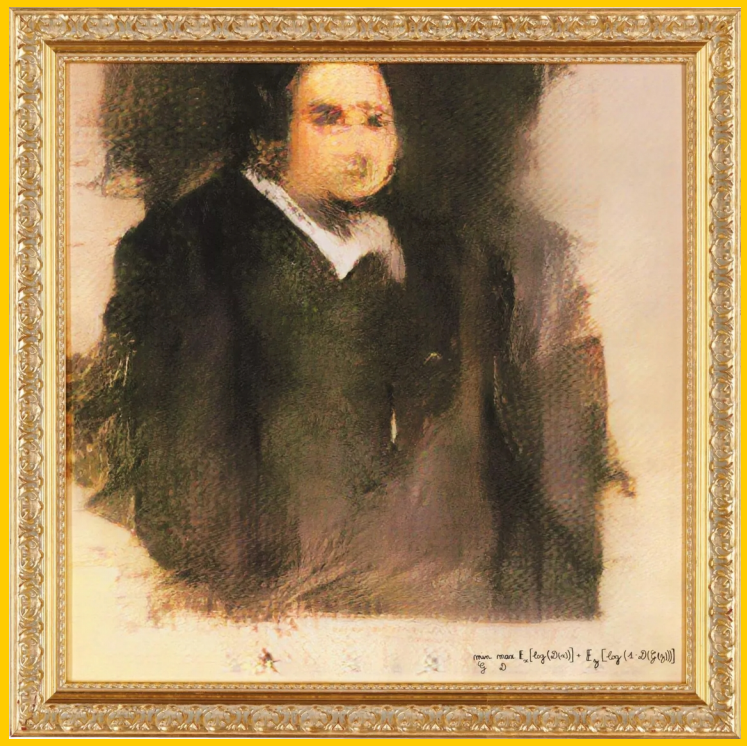
Statistical Learning Group 5

Generative Models



Anup Tuladhar

May 31st, 2019

UNIVERSITY OF CALGARY

HOTCHKISS BRAIN INSTITUTE

Which reality TV star is fake?

What are the interesting use cases for generative models

"Zoom and Enhance"

Dahl et al, 2017

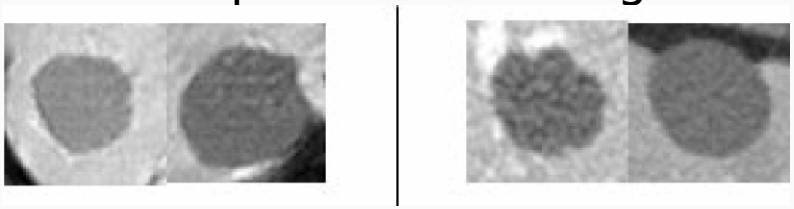
Data de-noising and completion

Li et al, 2017

Style transfer

Adrian Rosebrock

Data augmentation to aid in supervised learning



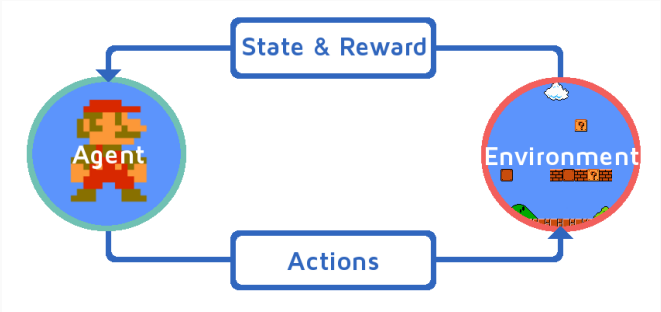
Maayan et al, 2018

Machine learning categories

Reinforcement Learning

Learn a **policy** for an agent to maximize its **reward** in an environment

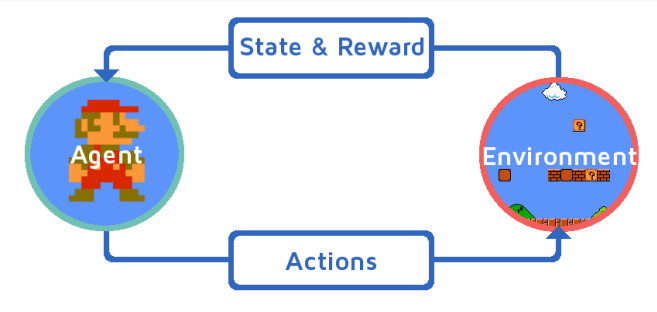
E.g. Q-learning, SARSA



Machine learning categories

Reinforcement Learning

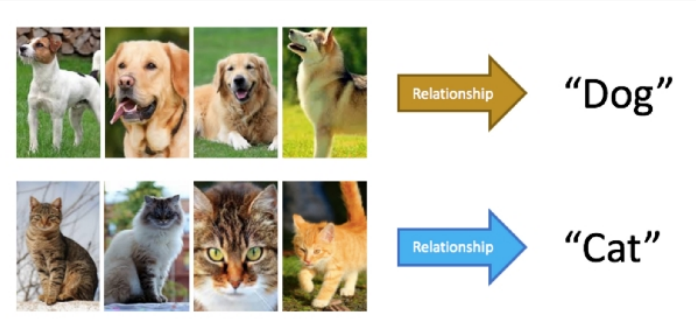
Learn a **policy** for an agent to maximize its **reward** in an environment



E.g. Q-learning, SARSA

Supervised Learning

Learn a model that knows how to map **data** to **label**

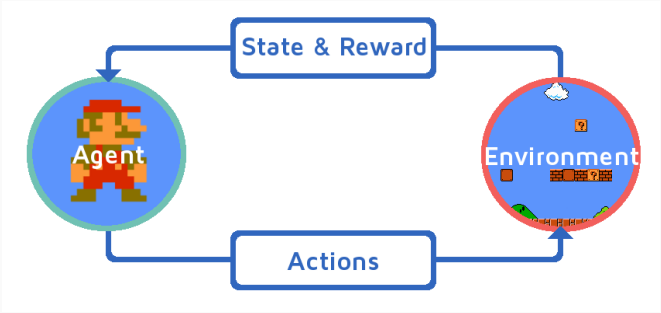


E.g. Decision trees, Random forests, Support vector machines

Machine learning categories

<u>Reinforcement Learning</u>

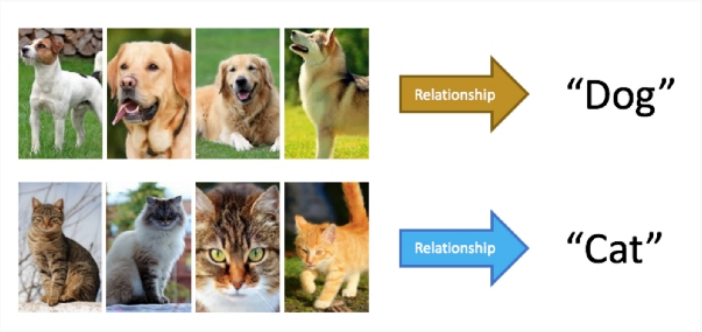
Learn a **policy** for an agent to maximize its **reward** in an environment



E.g. Q-learning, SARSA

<u>Supervised Learning</u>

Learn a model that knows how to map **data** to **label**



E.g. Decision trees, Random forests, Support vector machines

<u>Unsupervised Learning</u>

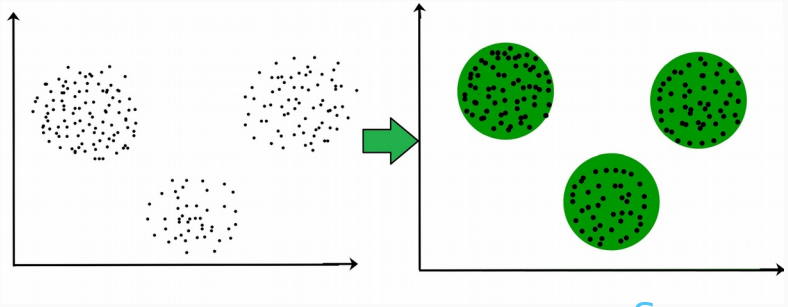
Learn the underlying structure of the **data** in the absence of labels

Types of Unsupervised Learning

Clustering

Group the data into based on the similarity, or dissimilarity, of their features

E.g. k-means clustering, hierarchical clustering



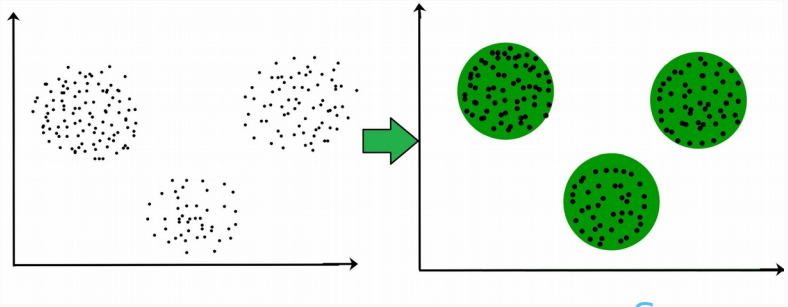
Source

Types of Unsupervised Learning

Clustering

Group the data into based on the similarity, or dissimilarity, of their features

E.g. k-means clustering, hierarchical clustering

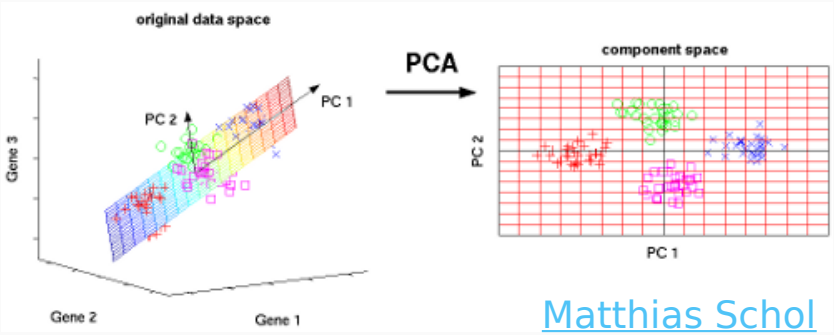


Source

Dimensionality reduction

Distill the feature space of the data down to a **smaller feature space** that still captures the **most important sources of variation**

E.g. Principal component analysis, autoencoders



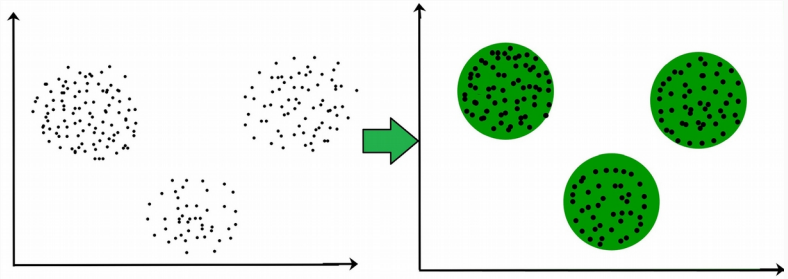
Matthias Scholz

Types of Unsupervised Learning

Clustering

Group the data into based on the similarity, or dissimilarity, of their features

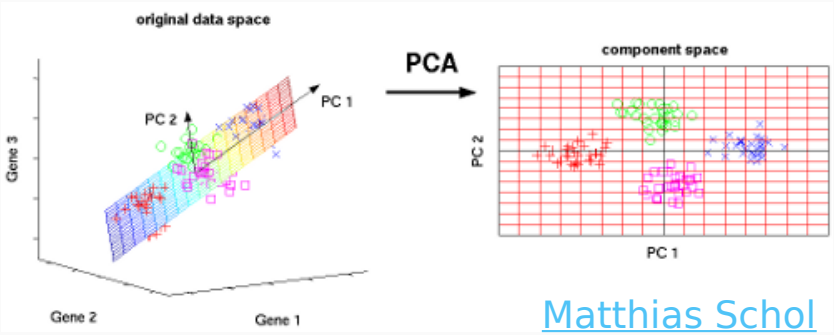
E.g. k-means clustering, hierarchical clustering



Source

Dimensionality reduction

Distill the feature space of the data down to a **smaller feature space** that still captures the **most important sources of variation**

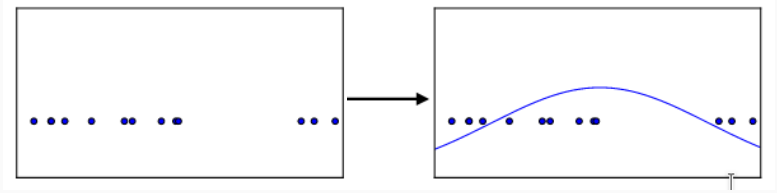


Matthias Scholz

E.g. Principal component analysis, autoencoders

Density estimation

Determine the underlying distribution of the data



Ian Goodfello

Generative Modeling

General Goal: Model the joint data distribution

p(observed, target)

Generative Modeling

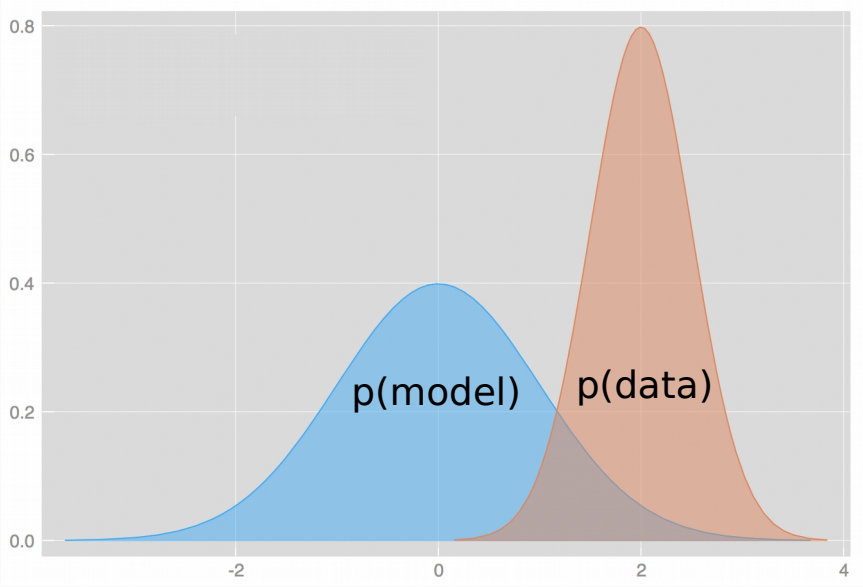
General Goal: Model the joint data distribution

p(observed, target)

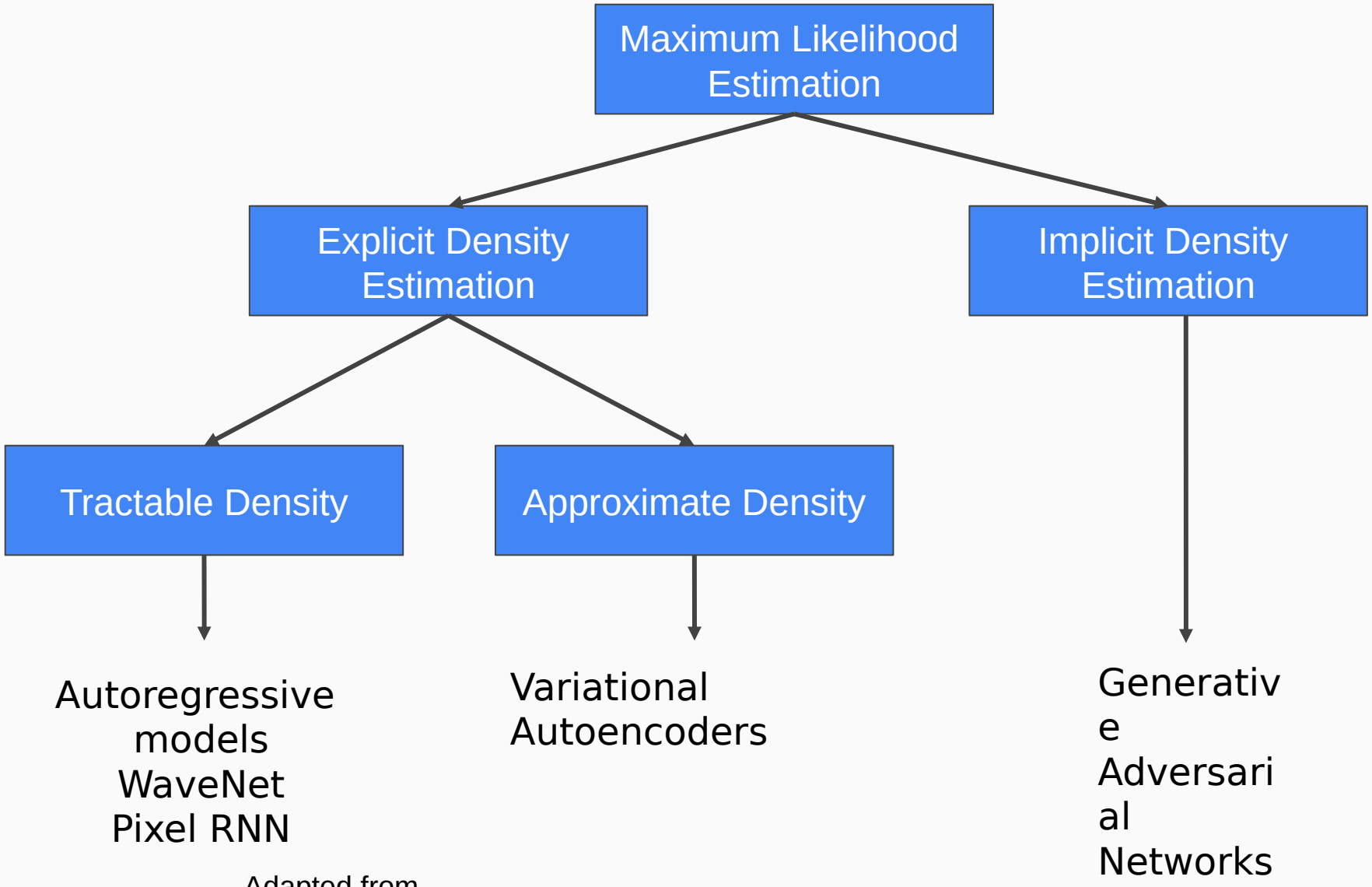
Our goal: Create a model, from which we can sample, that approximates the joint data distribution

I.e. Minimize the KL divergence between the data-generating distribution and the model distribution

$$\min_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} D_{\mathrm{KL}}\left(p_{\mathrm{data}}(\mathbf{x}) \,\|\, p(\mathbf{x})\right)$$

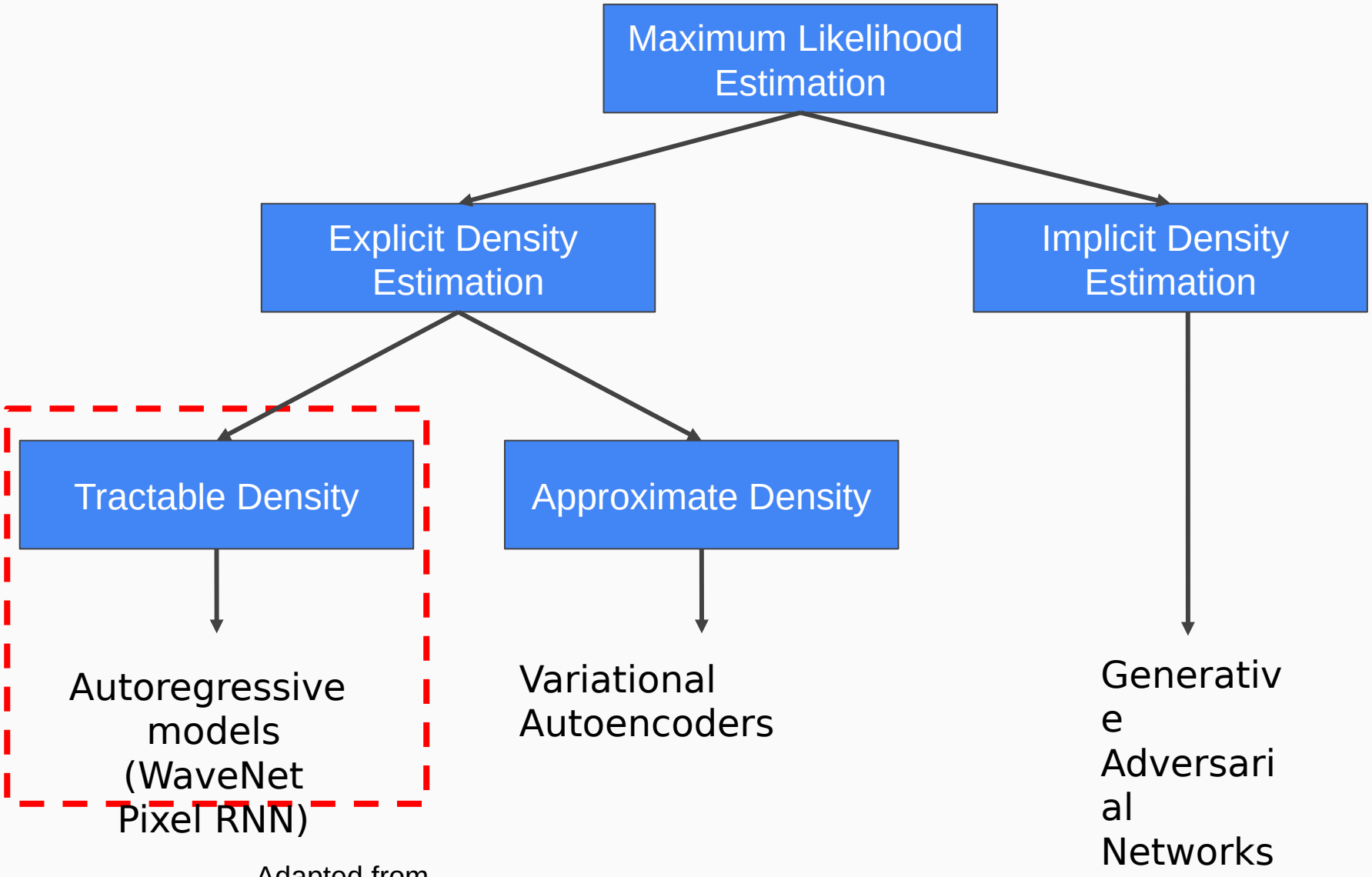


Taxonomy of Generative Models



Adapted from
Ian Goodfellow, Tutorial on Generative Adversarial Networks, NIPS 2016

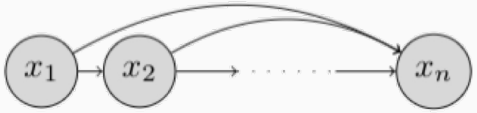
Taxonomy of Generative Models

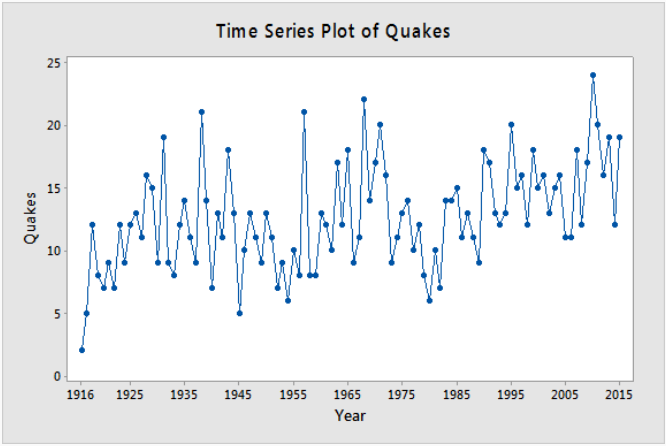


Adapted from
Ian Goodfellow, Tutorial on Generative Adversarial Networks, NIPS 2016

Tractable Explicit Density Estimation: Autoregressive Models

Auto-regression: Prediction based on past data

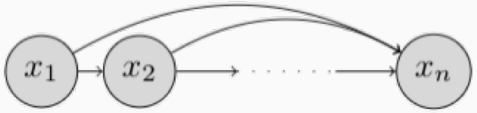




Source:
STAT 501, Penn State

Tractable Explicit Density Estimation: Autoregressive Models

<u>Auto-regression:</u> Prediction based on past data

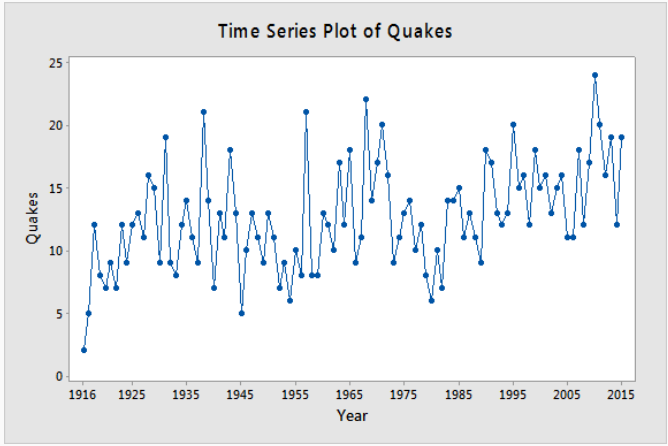


Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i^{th} value (e.g. pixel value, audio signal) given all previous values

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{<i})$$

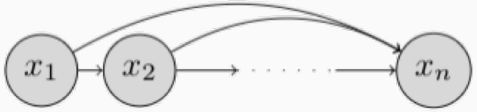
Training goal: Maximize the likelihood of training data

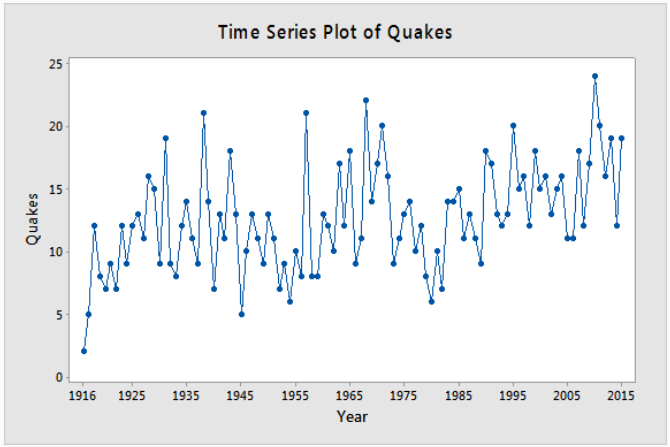


Source:
STAT 501, Penn State

Tractable Explicit Density Estimation: Autoregressive Models

Auto-regression: Prediction based on past data





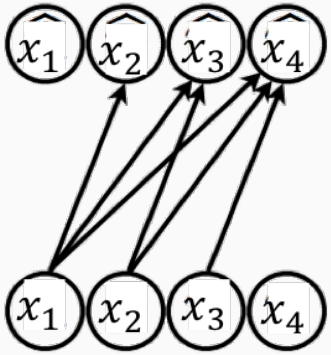
Source: STAT 501, Penn State

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i^{th} value (e.g. pixel value, audio signal) given all previous values

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{<i})$$

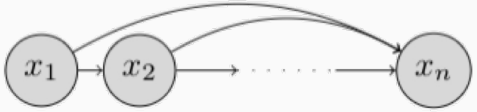
Training goal: Maximize the likelihood of training data

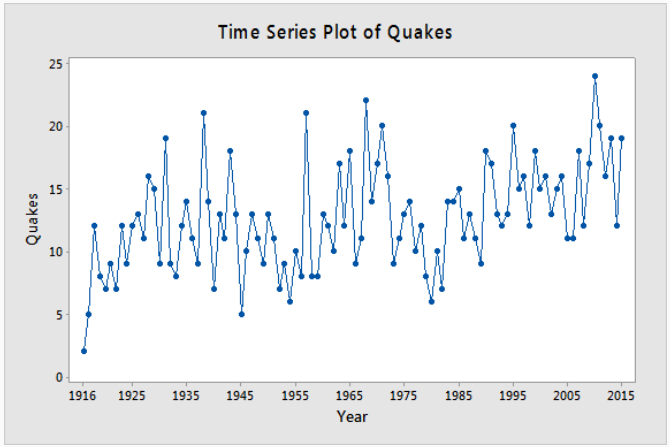


Fully Visible Belief Network
(Source: Stanford CS236)

Tractable Explicit Density Estimation: Autoregressive Models

Auto-regression: Prediction based on past data





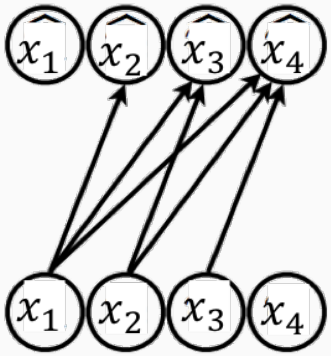
Source: STAT 501, Penn State

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

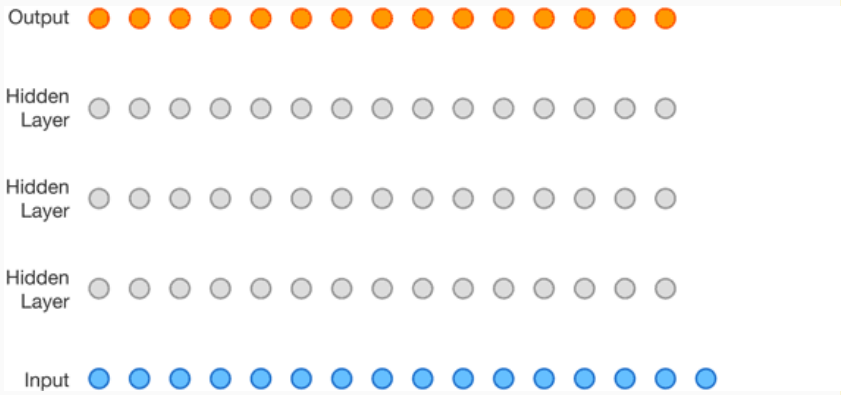
Probability of i^{th} value (e.g. pixel value, audio signal) given all previous values

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{<i})$$

Training goal: Maximize the likelihood of training data



Fully Visible Belief Network (Source: Stanford CS236)



WaveNet (Source: Google DeepMind

Autoregressive Models: PixelRNN

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i[th] value (e.g. pixel value, audio signal) given all previous values

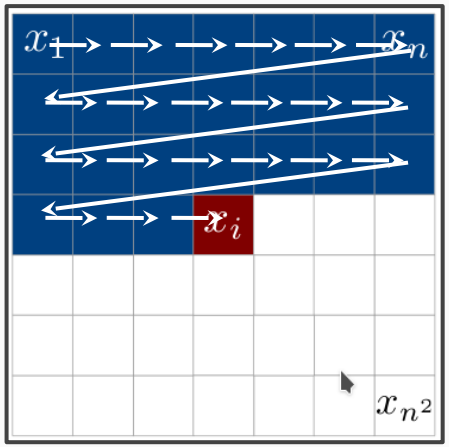
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n^2} p(x_i | \mathbf{x}_{<i})$$

Training goal: Maximize the likelihood of training data

Treat images as a sequence of pixels

Use Recurrent Neural Networks (RNN) model the distribution over pixel values
- LSTM, specifically

Generation begins at a single pixel (e.g. top left) and proceeds sequentially
- Either row by row



Source:
Oord et al (2016), Pixel Recurrent Neural Networks

Autoregressive Models: PixelRNN

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i[th] value (e.g. pixel value, audio signal) given all previous values

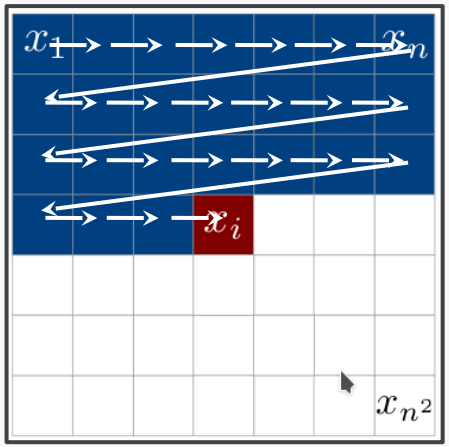
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n^2} p(x_i | \mathbf{x}_{<i})$$

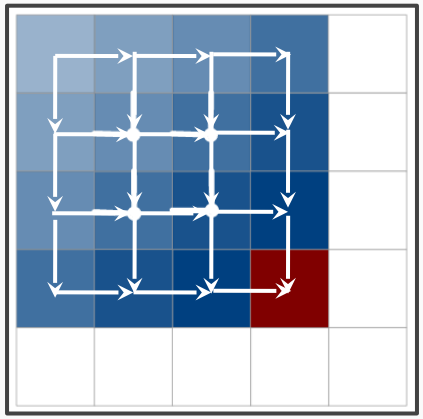
Training goal: Maximize the likelihood of training data

Treat images as a sequence of pixels

Use Recurrent Neural Networks (RNN) model the distribution over pixel values
- LSTM, specifically

Generation begins at a single pixel (e.g. top left) and proceeds sequentially
- Either row by row
- Or bi-directionally





Source:
Oord et al (2016), Pixel Recurrent Neural Networks

Autoregressive Models: PixelRNN

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i[th] value (e.g. pixel value, audio signal) given all previous values

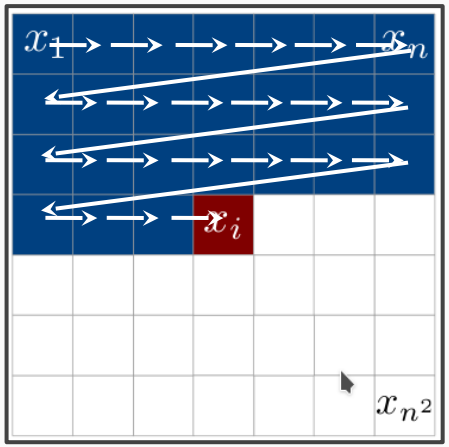
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n^2} p(x_i | \mathbf{x}_{<i})$$

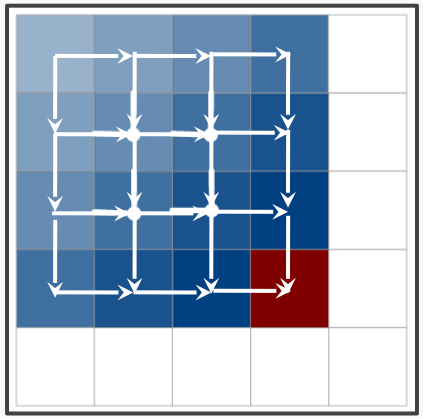
Training goal: Maximize the likelihood of training data

Treat images as a sequence of pixels

Use Recurrent Neural Networks (RNN) model the distribution over pixel values
- LSTM, specifically

Generation begins at a single pixel (e.g. top left) and proceeds sequentially
- Either row by row
- Or by diagonal





Source:
Oord et al (2016), Pixel Recurrent Neural Networks

PixelCNN: Uses a Convolutional Neural Network instead of RNN to model distribution, speeds up

Autoregressive Models: PixelRNN

Likelihood of data x (e.g. image, song) is defined as a product of conditional probabilities

Probability of i^{th} value (e.g. pixel value, audio signal) given all previous values

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n^2} p(x_i | \mathbf{x}_{<i})$$

Training goal: Maximize the likelihood of training data

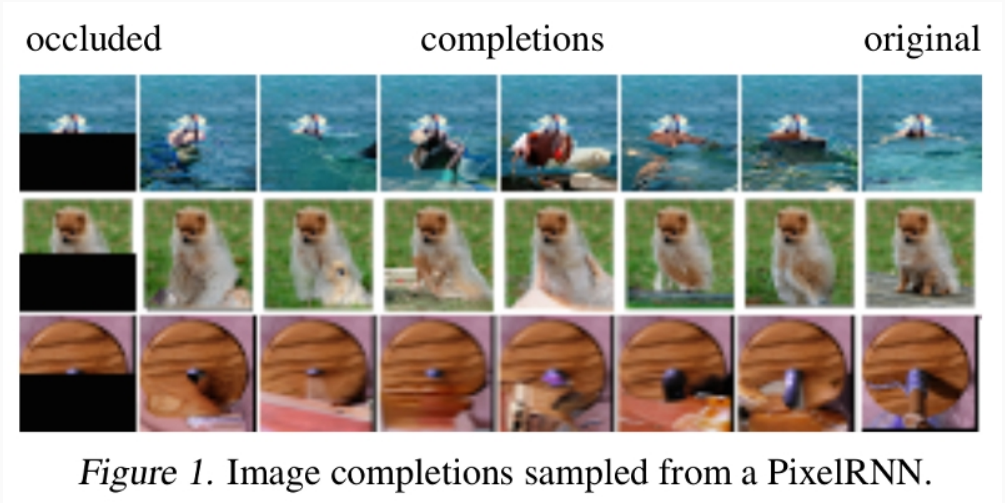
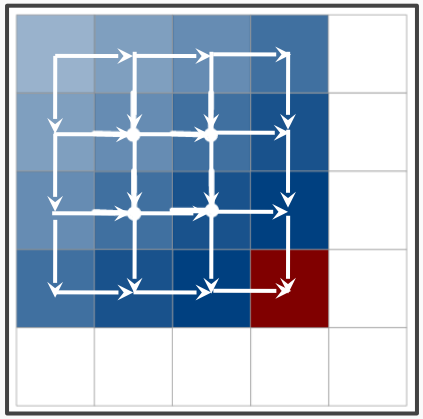
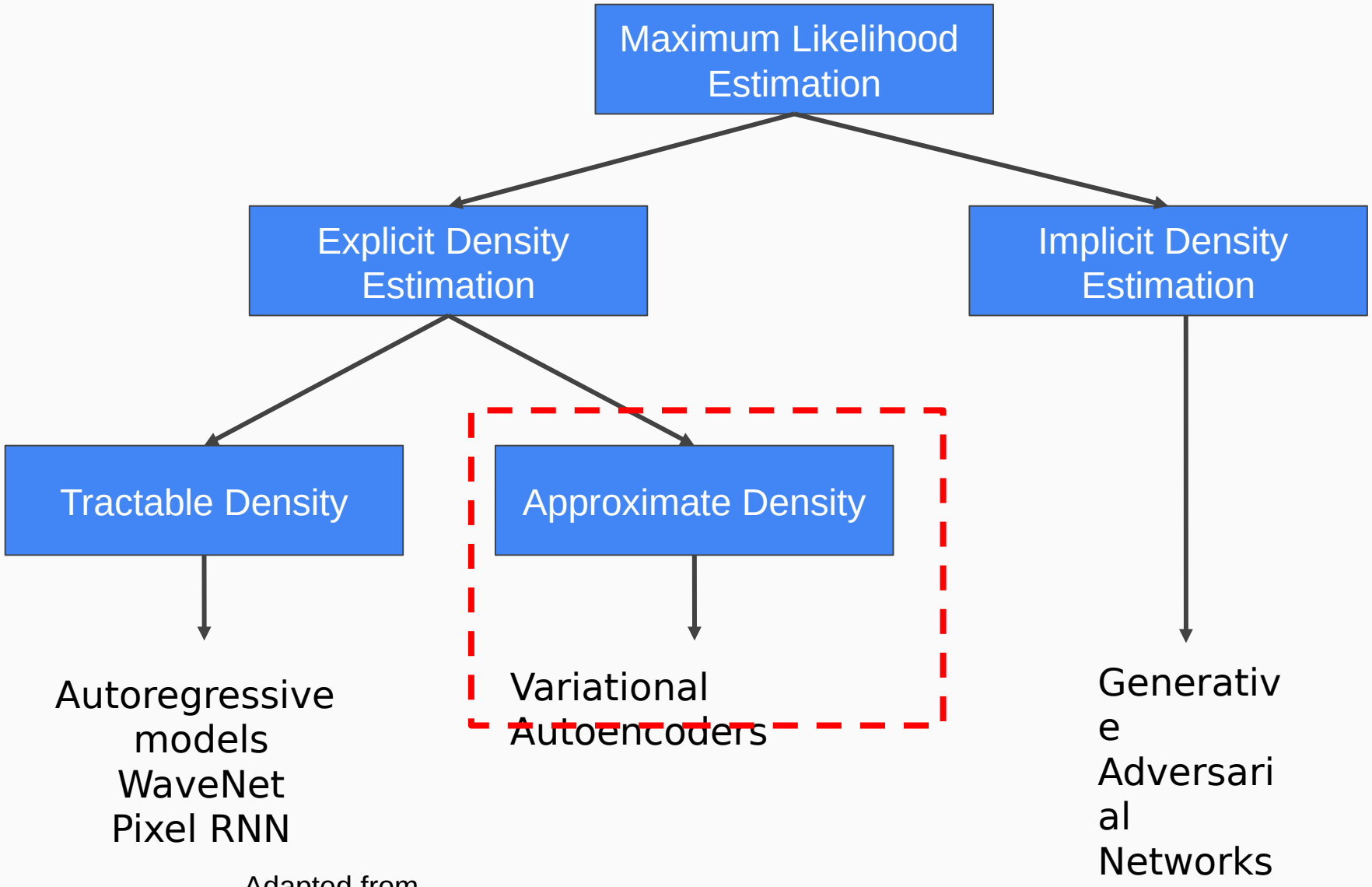


Figure 1. Image completions sampled from a PixelRNN.



Source:
Oord et al (2016), Pixel Recurrent Neural Networks

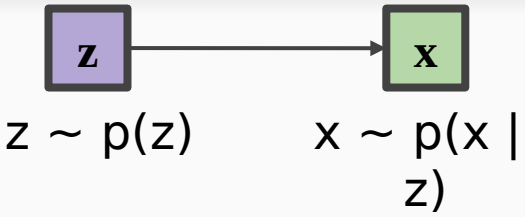
Taxonomy of Generative Models



Adapted from
Ian Goodfellow, Tutorial on Generative Adversarial Networks, NIPS 2016

Latent space

Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)



Latent space

Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)

z ~ p(z)

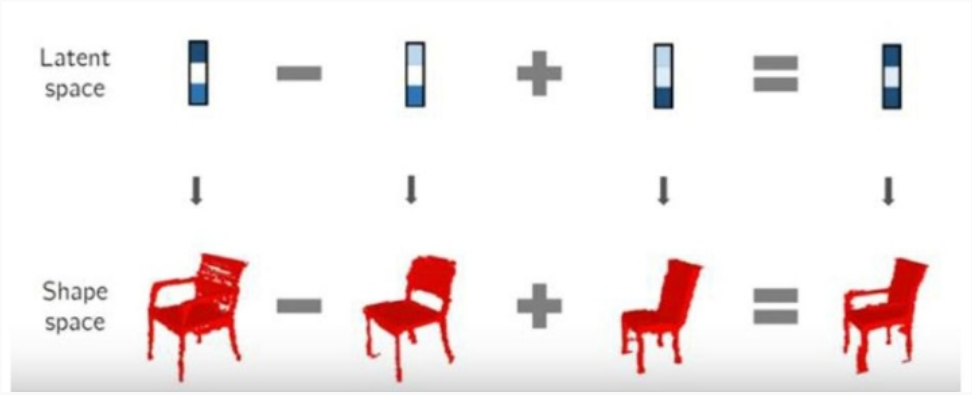
x ~ p(x | z)

Latent space

Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)

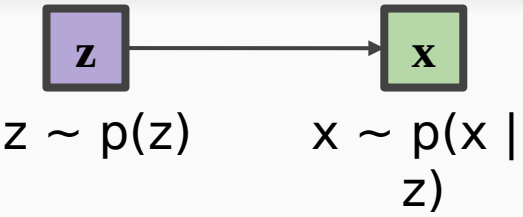
z ~ p(z)

x ~ p(x | z)

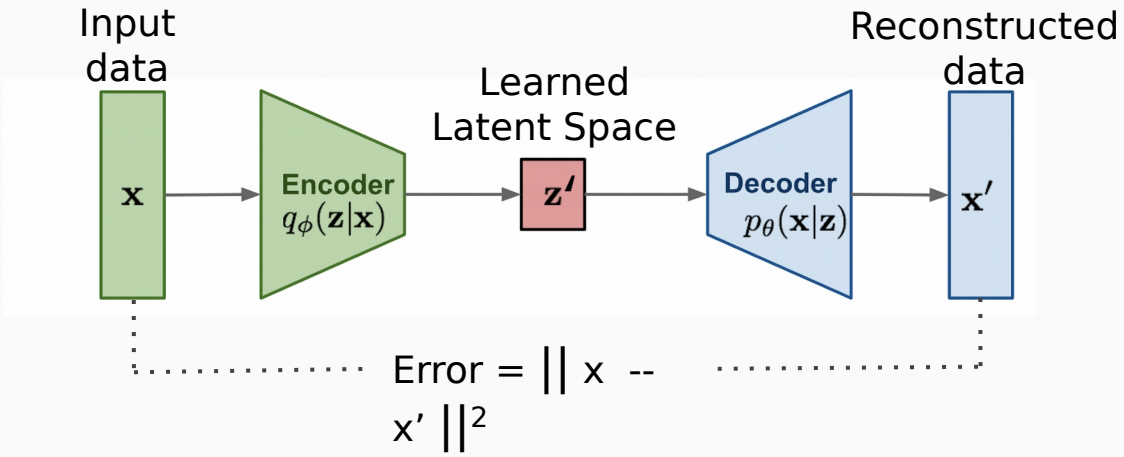


Approximate Explicit Density Estimation: Traditional Autoencoder

Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)



Traditional Autoencoder: Unsupervised learning of the latent space of the training data



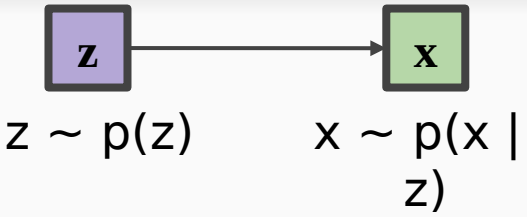
Adapted from: Lilian Weng

Training goal: Minimize the loss between the training data and the reconstructed data

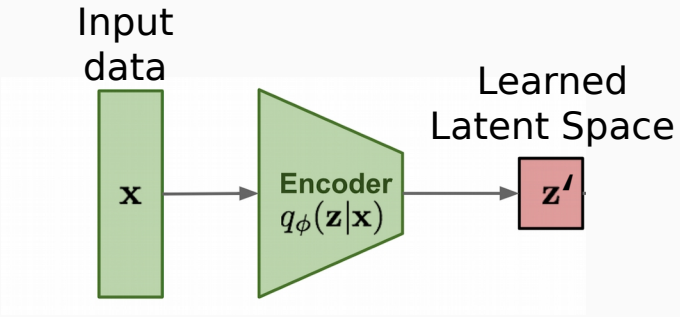
I.e. Train an encoder that creates a latent space which can be used to mimic the

Approximate Explicit Density Estimation: Traditional Autoencoder

Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)



Traditional Autoencoder: Unsupervised learning of the latent space of the training data



* Latent space has less dimensions (less features) than the input data

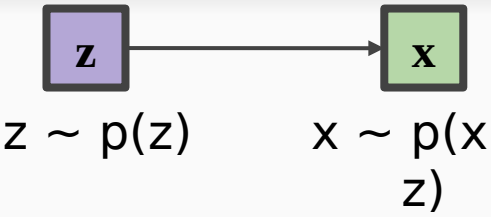
→ Data compression

→ Dimensionality reduction

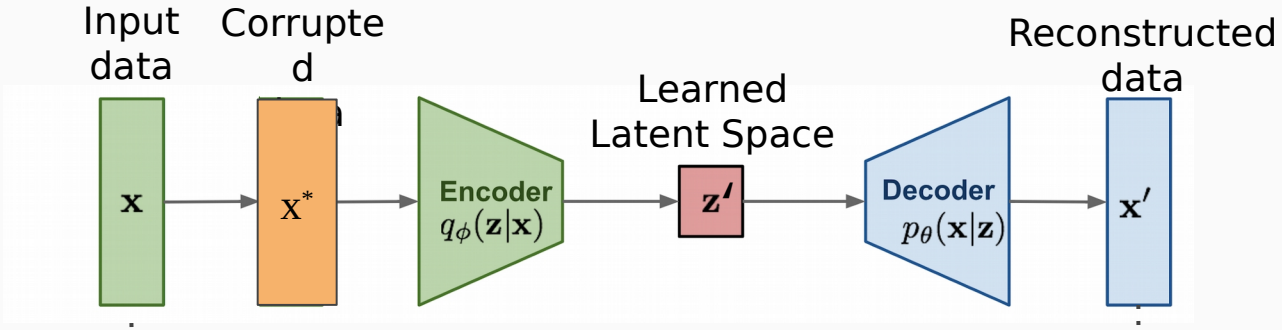
→ Feature extraction for later supervised learning

Approximate Explicit Density Estimation: Denoising Autoencoder

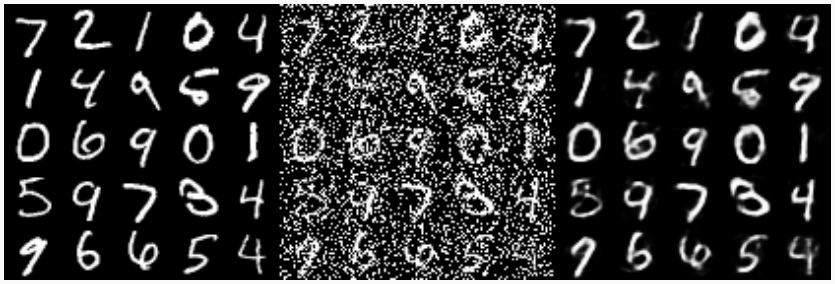
Latent space: An underlying and unobserved distribution (z) that is assumed to be a generative source of data (x)



Denoising Autoencoder: Reconstruct corrupted images to their original form



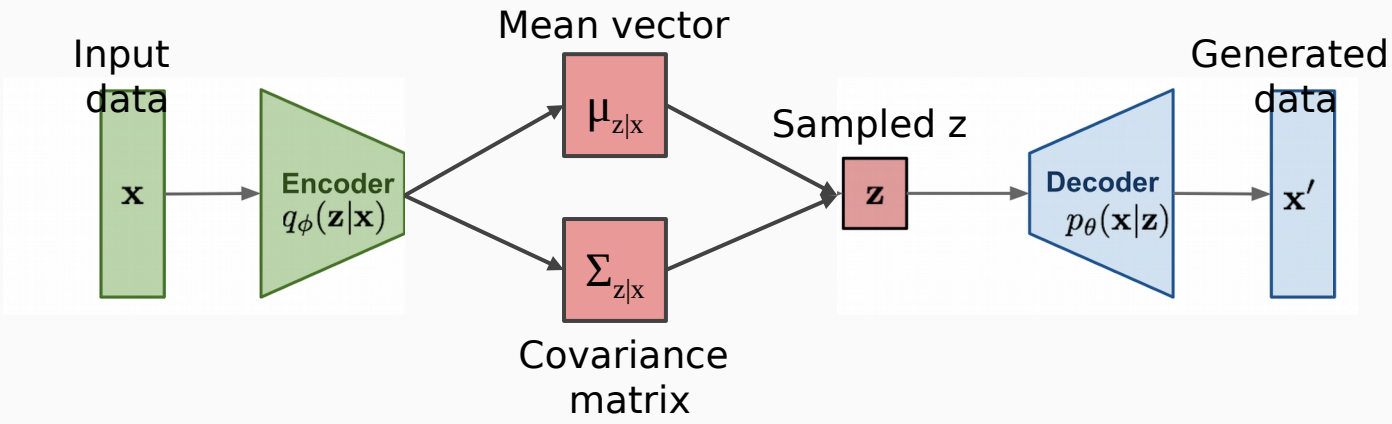
Adapted from:
Lilian Weng



Approximate Explicit Density Estimation: Variational Autoencoder

Traditional Autoencoder: Map input to **fixed** latent vector z

Variational Autoencoder: Map input to a **distribution** that generates variations in z



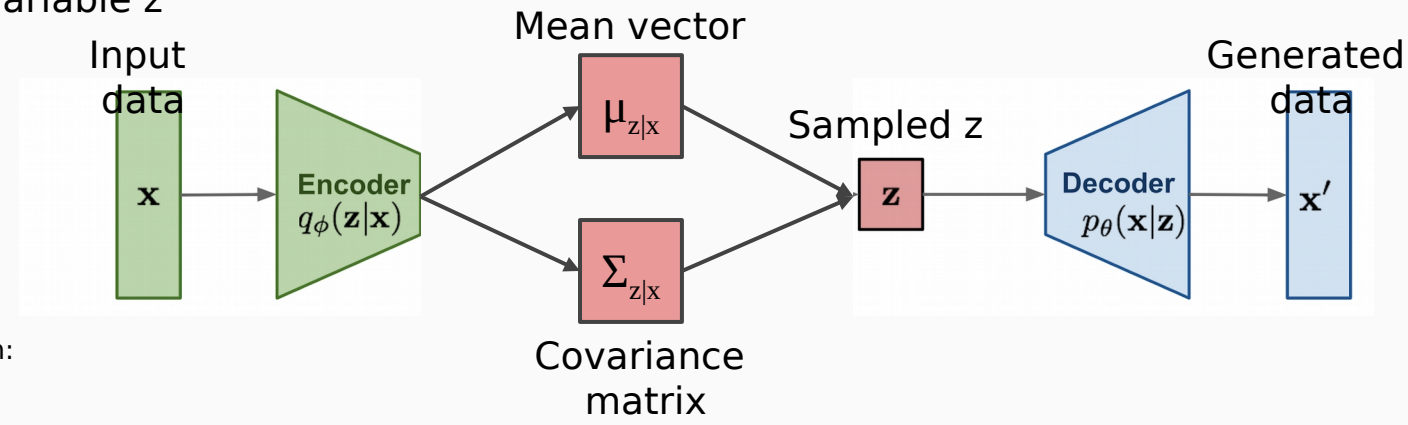
Adapted from:
Lilian Weng

Approximate Explicit Density Estimation: Variational Autoencoder

Traditional Autoencoder: Map input to **fixed** latent vector z

Variational Autoencoder: Map input to a **distribution** that generates variations in z

- Choose a prior probability for p(z) → Often set to a simple distribution i.e. Gaussian
- Need to learn parameters of Decoder Network $p_\theta(x \mid z)$ that can generate data from a variable z



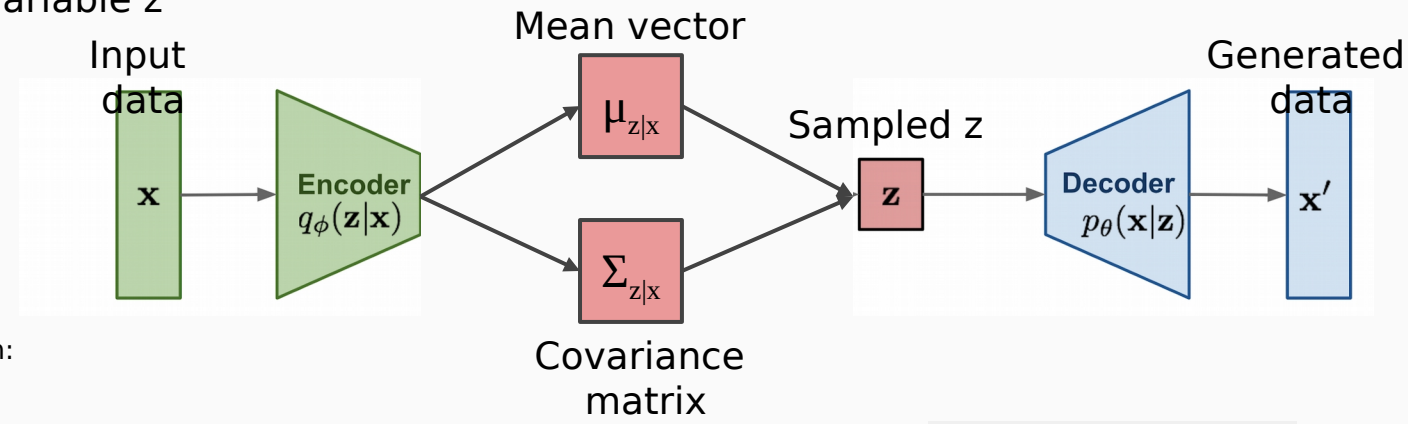
Adapted from:
Lilian Weng

Approximate Explicit Density Estimation: Variational Autoencoder

Traditional Autoencoder: Map input to **fixed** latent vector z

Variational Autoencoder: Map input to a **distribution** that generates variations in z

- Choose a prior probability for p(z) → Often set to a simple distribution i.e. Gaussian
- Need to learn parameters of Decoder Network $p_\theta(x \mid z)$ that can generate data from a variable z



Adapted from:
Lilian Weng

$$\min_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} D_{\mathrm{KL}}\left(p_{\mathrm{data}}(\mathbf{x}) \,\|\, p(\mathbf{x})\right) = \max_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

Data likelihood:

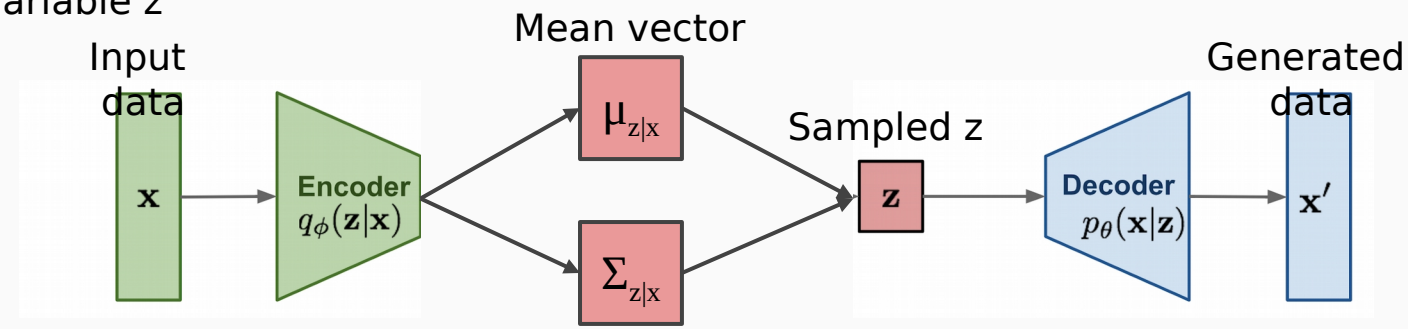
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Approximate Explicit Density Estimation: Variational Autoencoder

Traditional Autoencoder: Map input to **fixed** latent vector z

Variational Autoencoder: Map input to a **distribution** that generates variations in z

- Choose a prior probability for p(z) → Often set to a simple distribution i.e. Gaussian
- Need to learn parameters of Decoder Network $p_\theta(x \mid z)$ that can generate data from a variable z



Adapted from: Lilian Weng

$$\min_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} D_{\mathrm{KL}}\left(p_{\mathrm{data}}(\mathbf{x}) \,\|\, p(\mathbf{x})\right) = \max_{p \in \mathcal{P}_{\mathbf{x},\mathbf{z}}} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

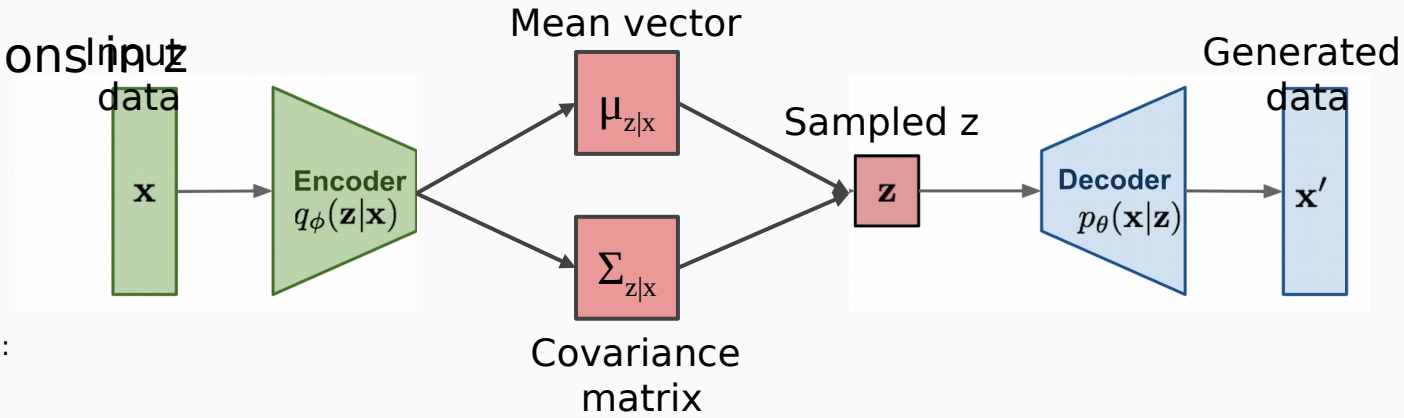
Data likelihood:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Intractable (computationally expensive) for high dimensional z (which is the case for most problems of interest)

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations in z



Adapted from:
Lilian Weng

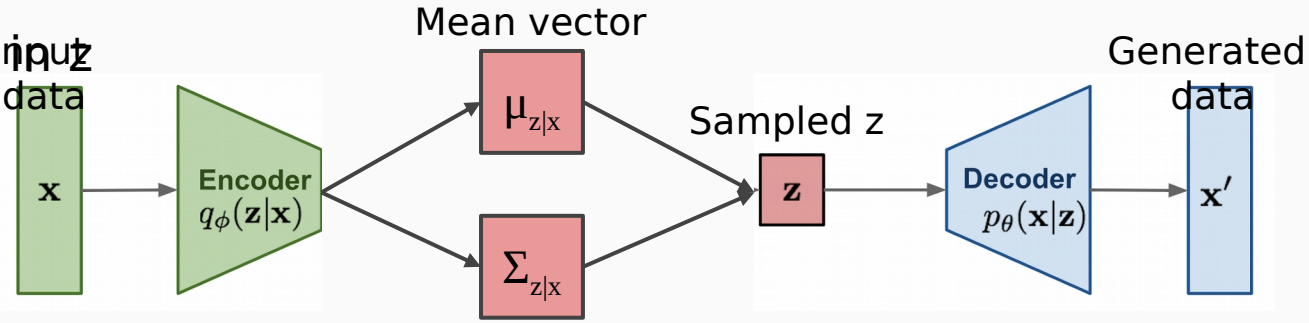
$$\log p_\theta(x^{(i)}) = \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Solvable

Intractable, but know it is ≥ 0

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations in z



Adapted from: Lilian Weng

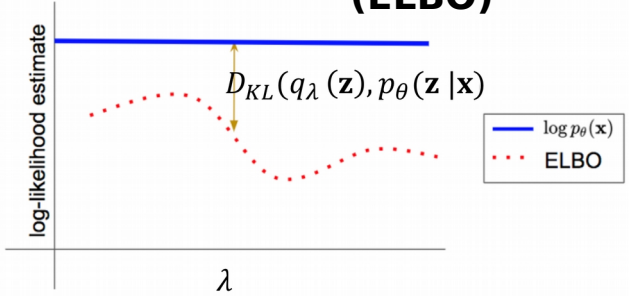
$$\log p_\theta(x^{(i)}) = \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

Solvable

Intractable, but know it is ≥ 0

Variational Lower Bound (ELBO)

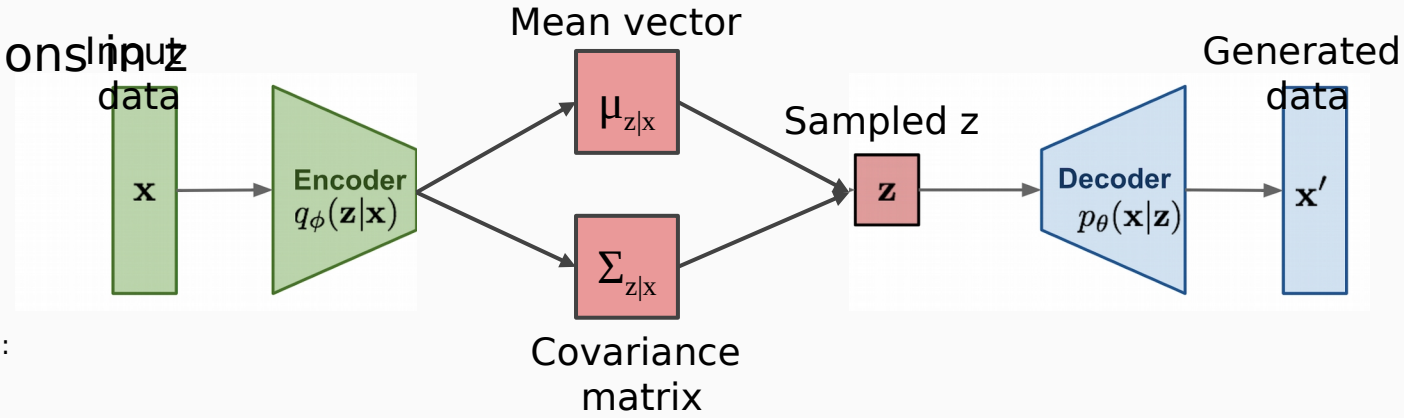
$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$



Training goal: Maximize the Variational Lower Bound

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations of input



Adapted from:
Lilian Weng

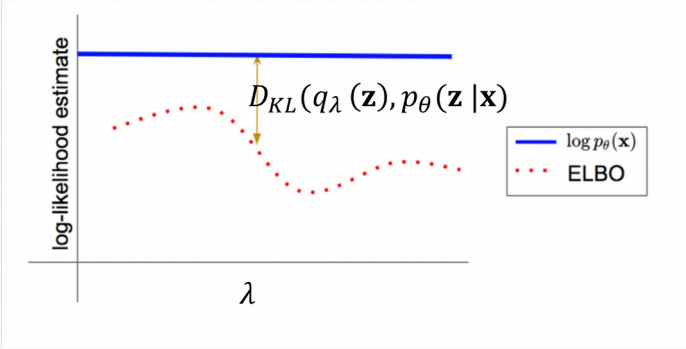
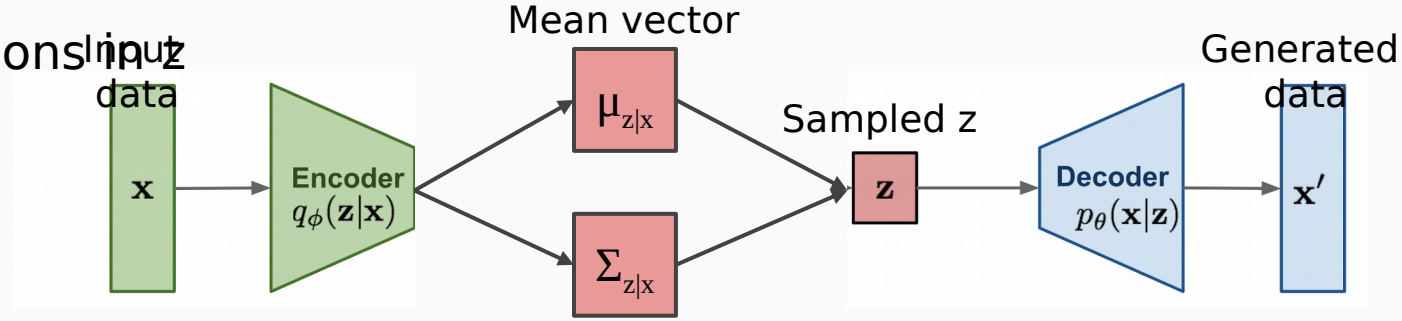


Image Source

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations in z



Adapted from: Lilian Weng

Smooth interpolations in latent space

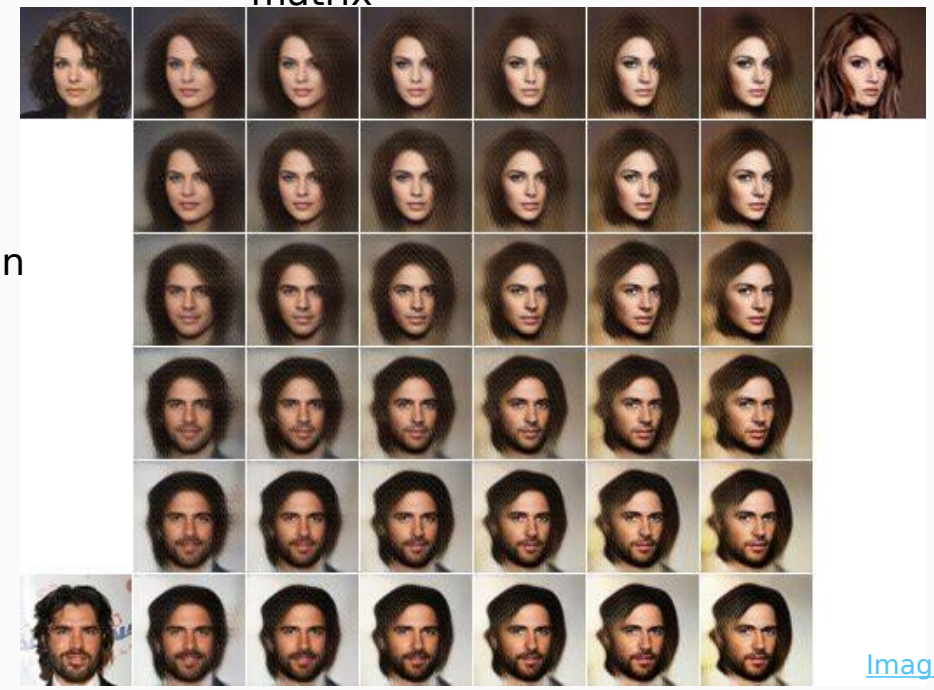
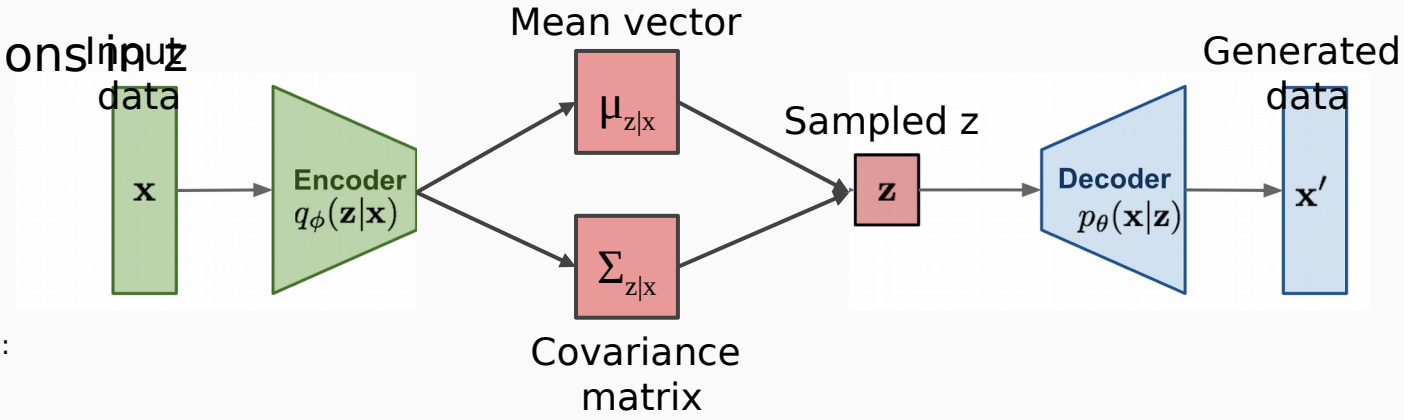


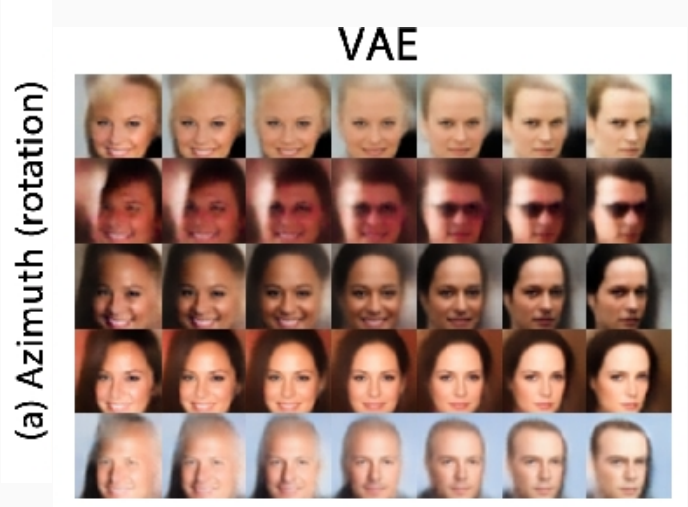
Image Source

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations in z



Adapted from:
Lilian Weng

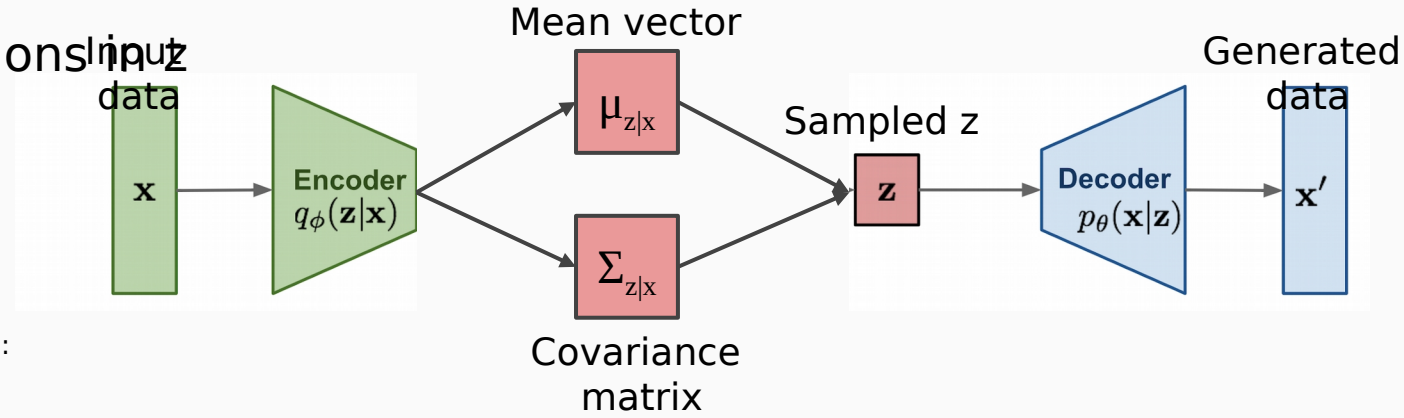


Entangled latent space

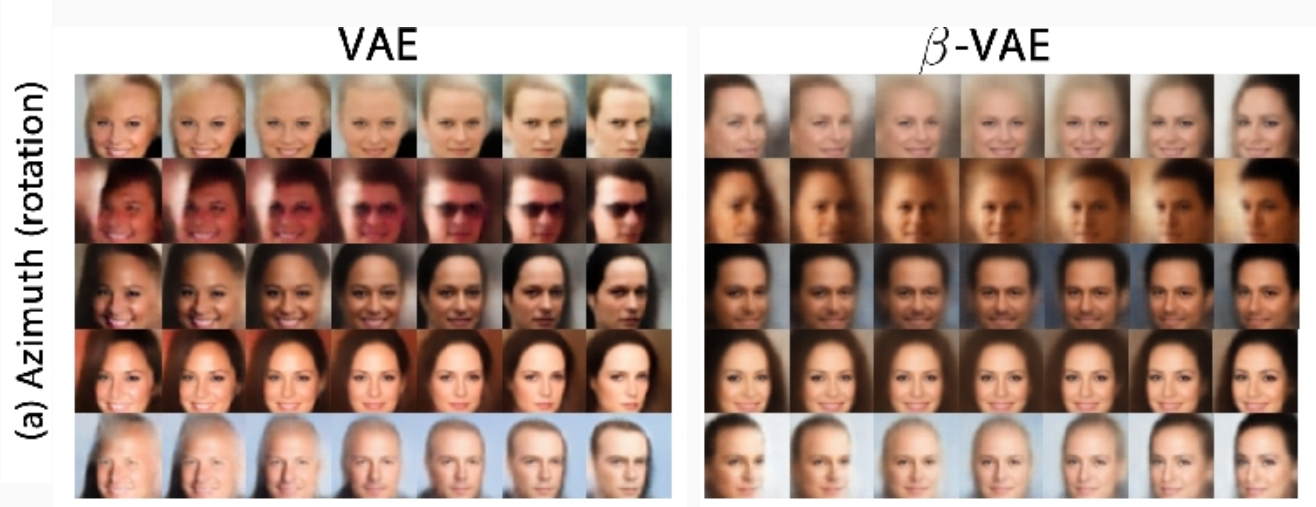
Higgins et al, beta-VAE, ICLR 2017

Approximate Explicit Density Estimation: Variational Autoencoder

Variational Autoencoder: Map input to a **distribution** that generates variations in z



Adapted from:
Lilian Weng

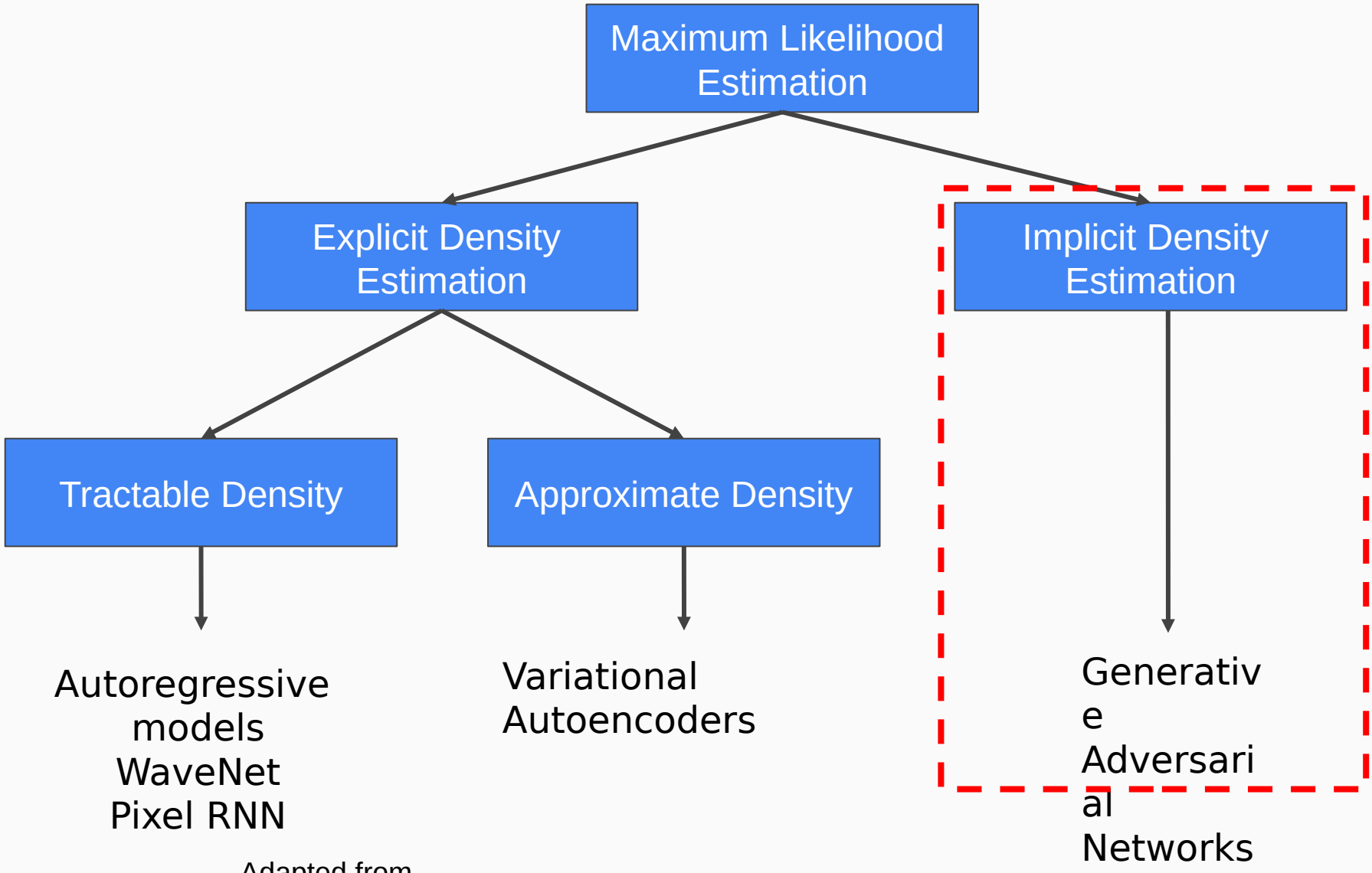


Entangled latent space

Reduced entanglement

Higgins et al, beta-VAE, ICLR 2017

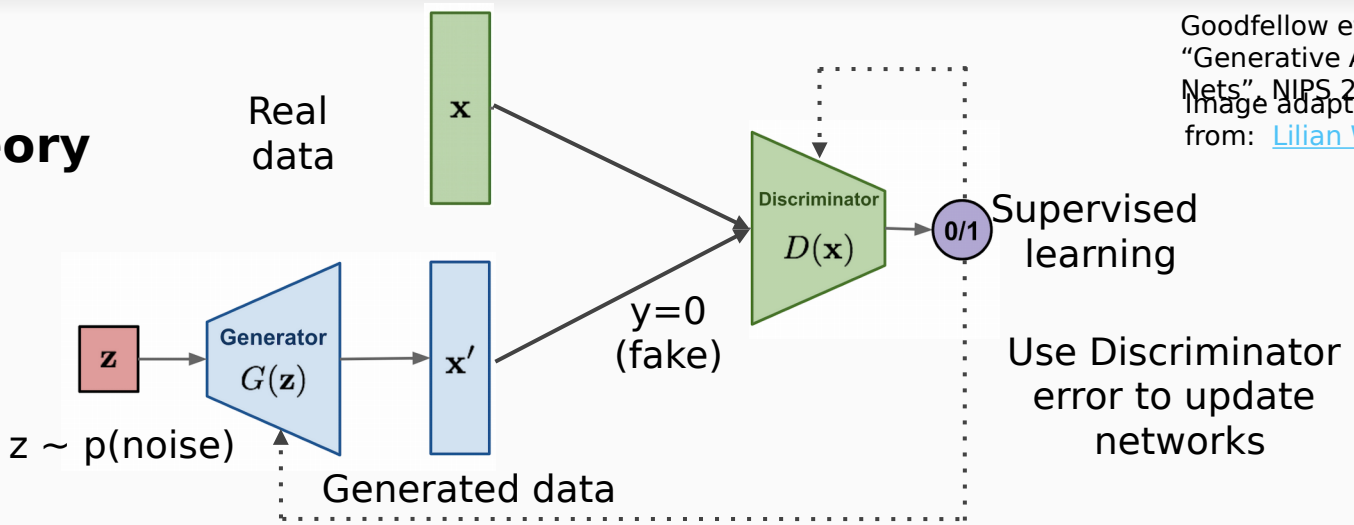
Taxonomy of Generative Models



Adapted from
Ian Goodfellow, Tutorial on Generative Adversarial Networks, NIPS 2016

Implicit Density Estimation: Generative Adversarial Networks

Game theory

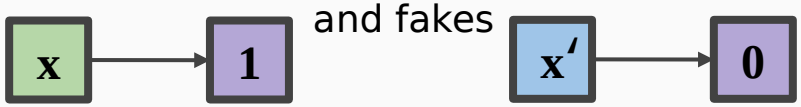


Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator
Learns to transform noise into fakes in order to fool the Discriminator

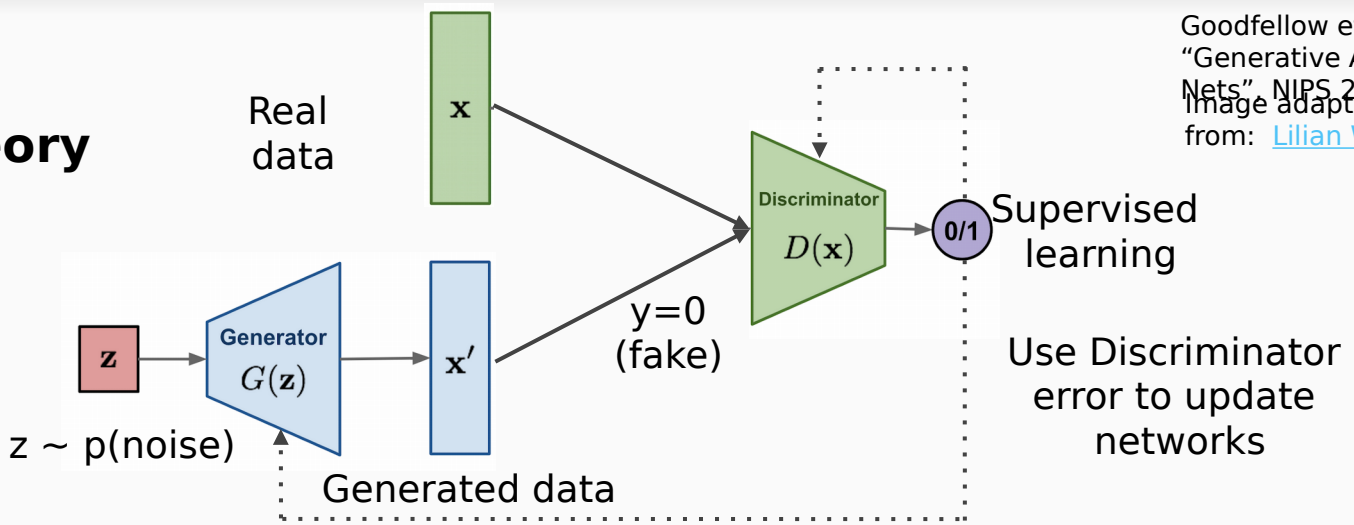
vs

Discriminator
Tries to tell the difference between real examples (data) and fakes



Implicit Density Estimation: Generative Adversarial Networks

Game theory



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator

Learns to transform noise into fakes in order to fool the Discriminator

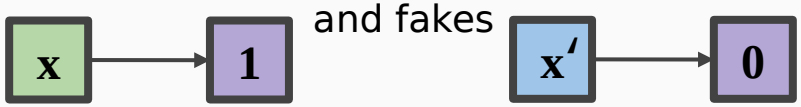
$$\min_{\theta} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Generator tries to minimize the Discriminator's accuracy on generated data

vs

Discriminator

Tries to tell the difference between real examples (data) and fakes

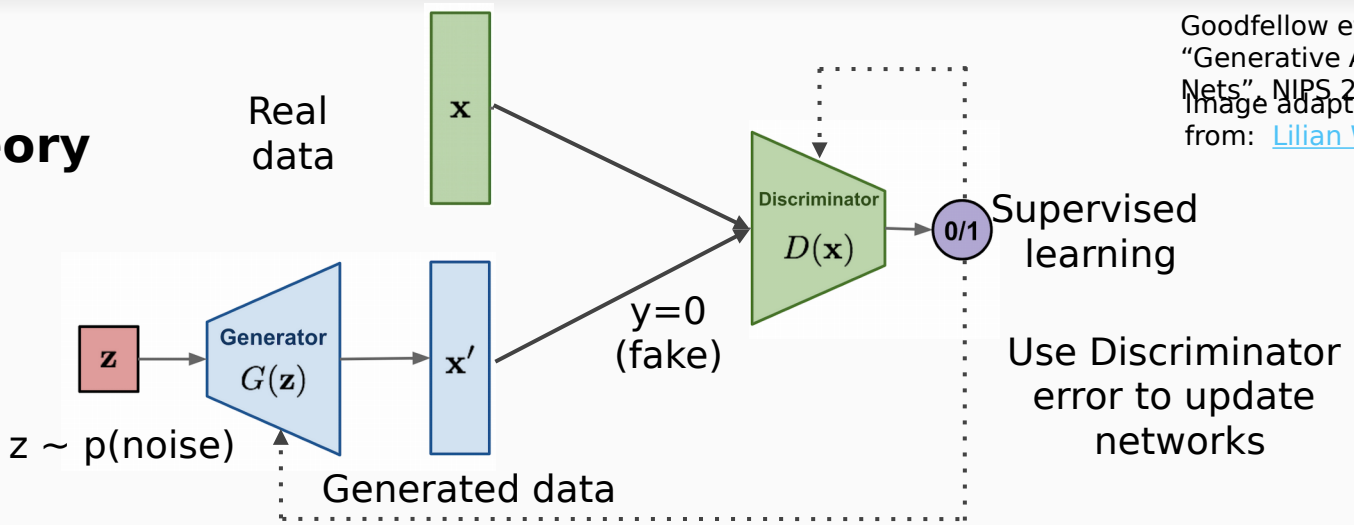


$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Discriminator tries to maximize its accuracy on identifying real as real and fake as fake

Implicit Density Estimation: Generative Adversarial Networks

Game theory



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator

Learns to transform noise into fakes in order to fool the Discriminator

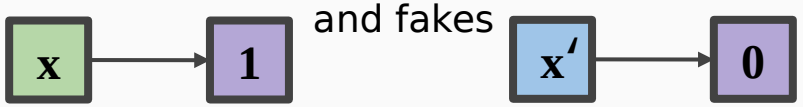
$$\min_{\theta} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Generator tries to minimize the Discriminator's accuracy on generated data

vs

Discriminator

Tries to tell the difference between real examples (data) and fakes



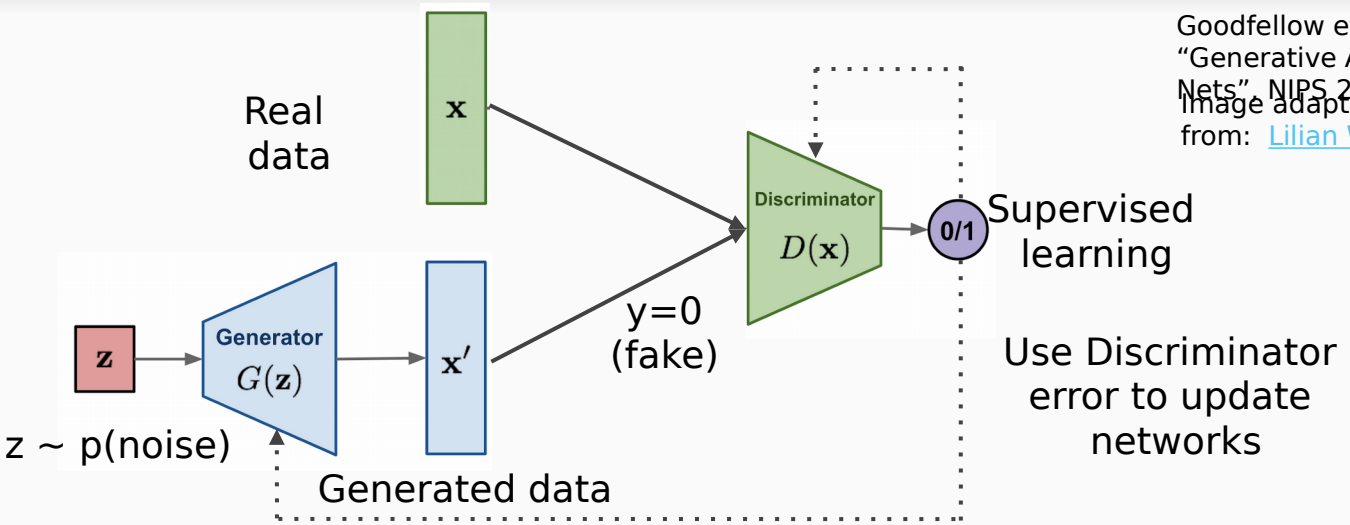
$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Discriminator tries to maximize its accuracy on identifying real as real and fake as fake

AN objectio

$$\min_{\theta} \max_{\phi} V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

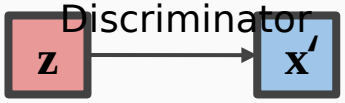
Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

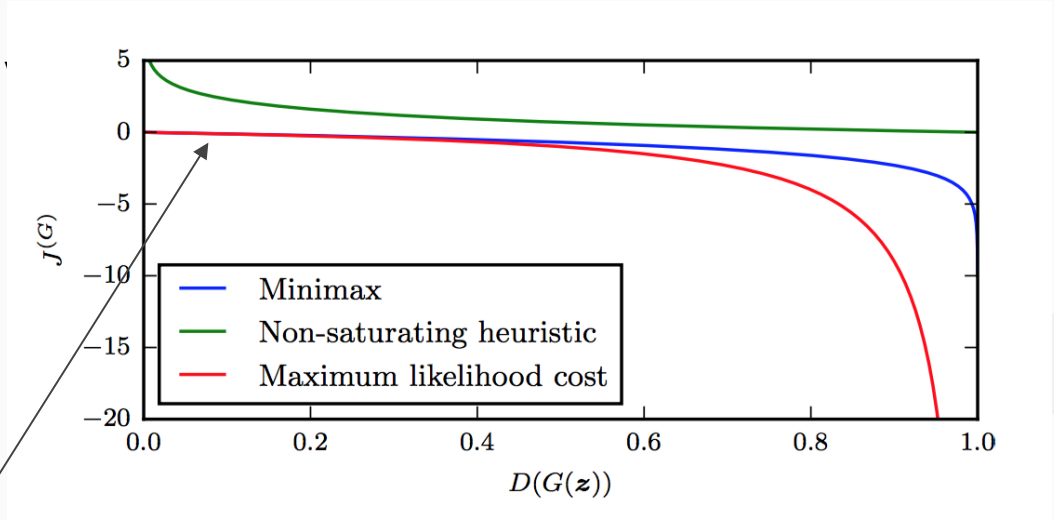
Generator

Learns to transform noise into fakes in order to fool the Discriminator



$$\min_\theta \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Generator tries to minimize the Discriminator's accuracy on generated data



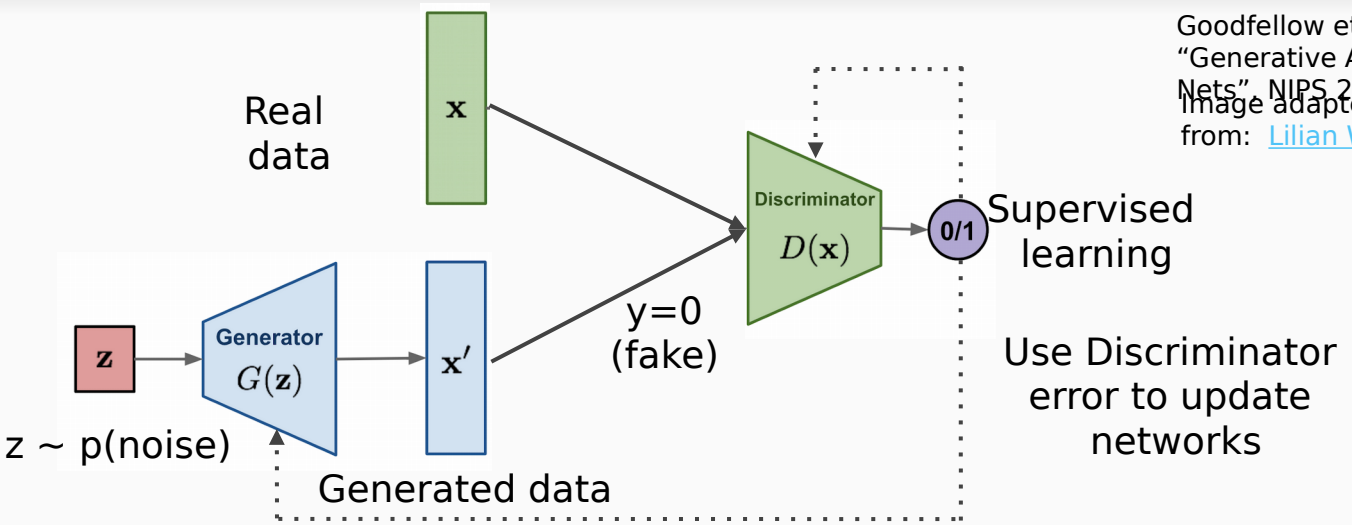
take as fake

** Difficult to train in this set-up because the gradients for Generator are very close to 0 at the start **

AN objectio

$$\min_\theta \max_\phi V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

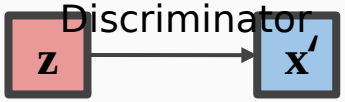
Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

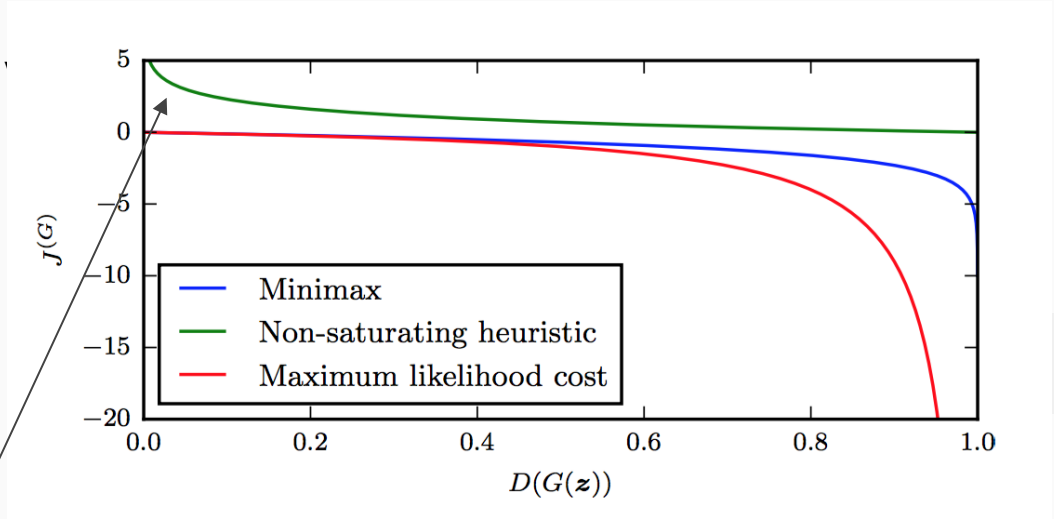
Generator

Learns to transform noise into fakes in order to fool the Discriminator



$$\max_{\phi} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Generator tries to maximize the Discriminator's error on generated data



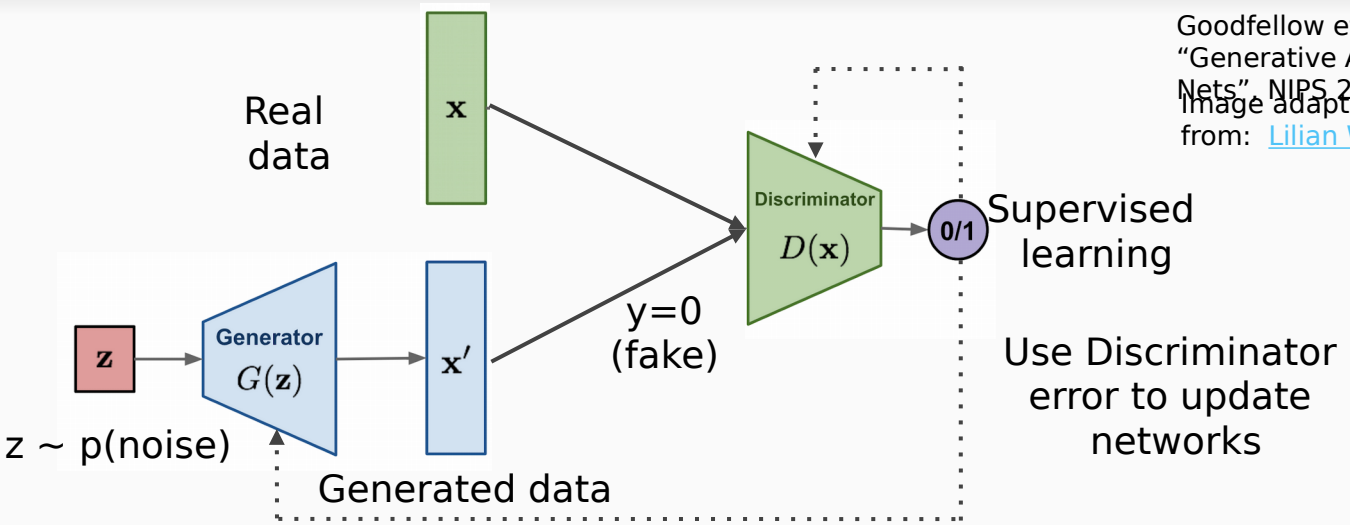
take as take

**** Higher gradient signals for bad examples, works better in practice ****

AN objectio

$$\min_{\theta}\max_{\phi} V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

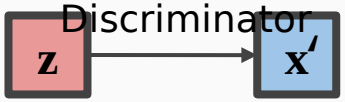
Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator

Learns to transform noise into fakes in order to fool the Discriminator



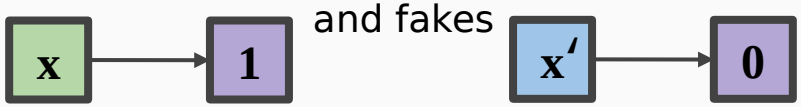
$$\max_{\phi} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Generator tries to maximize the Discriminator's error on generated data

vs

Discriminator

Tries to tell the difference between real examples (data) and fakes



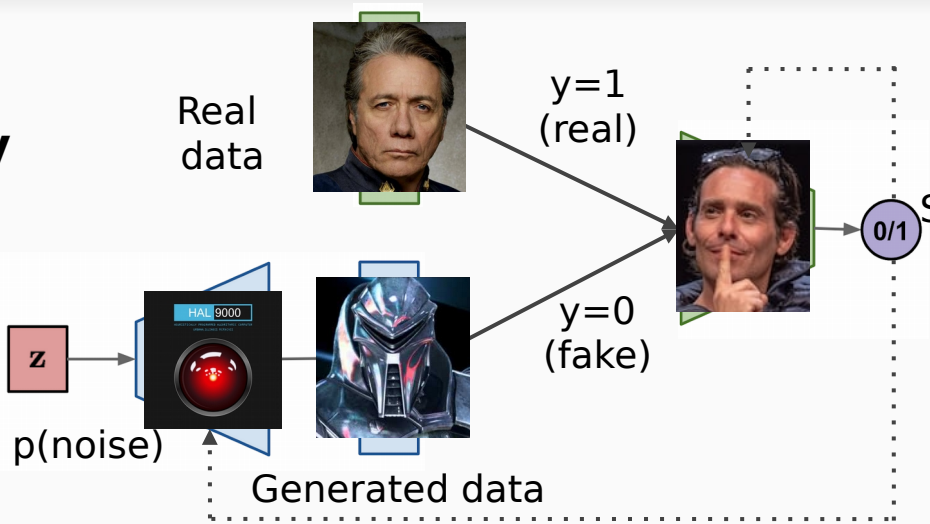
$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

I.e. Discriminator tries to maximize its accuracy on identifying real as real and fake as fake

AN objective function: Alternate between maximizing Generator and Discriminator objective functions

Implicit Density Estimation: Generative Adversarial Networks

Game theory approach



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

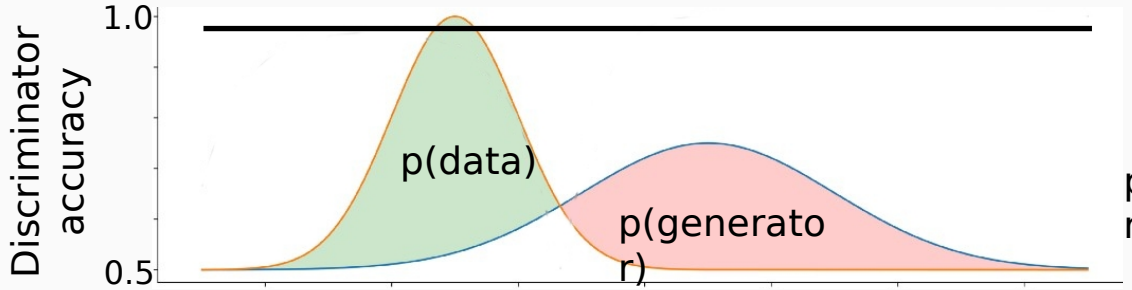
Generator
Learns to transform noise into fakes in order to fool the Discriminator

$$\max_{\phi} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

vs

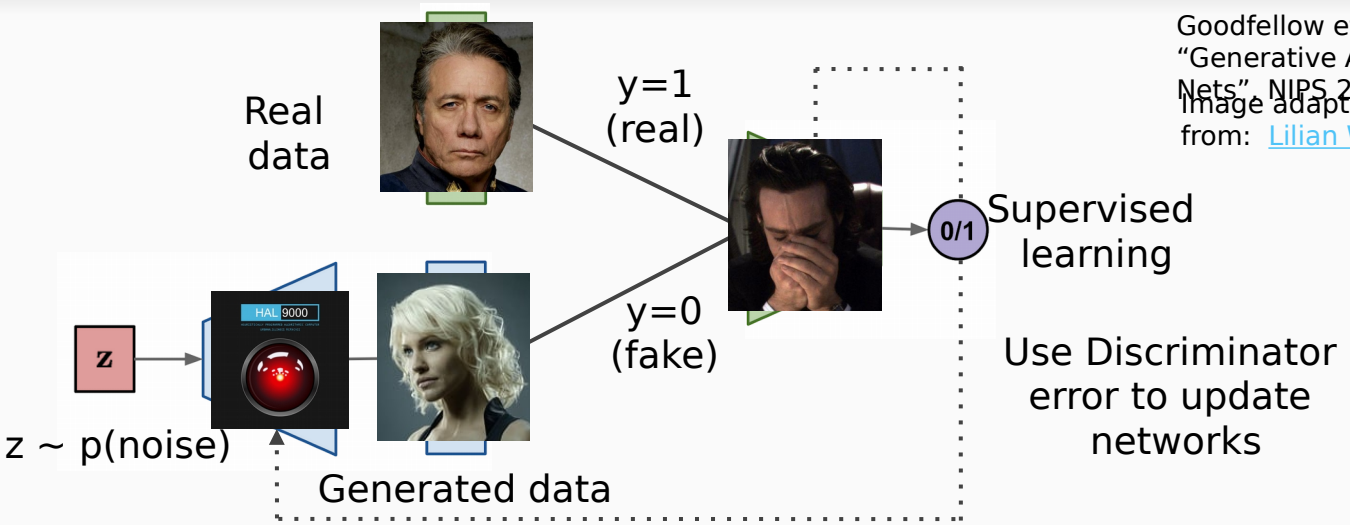
Discriminator
Tries to tell the difference between real examples (data) and fakes

$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$



p(generator) ≠ p(data)

Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

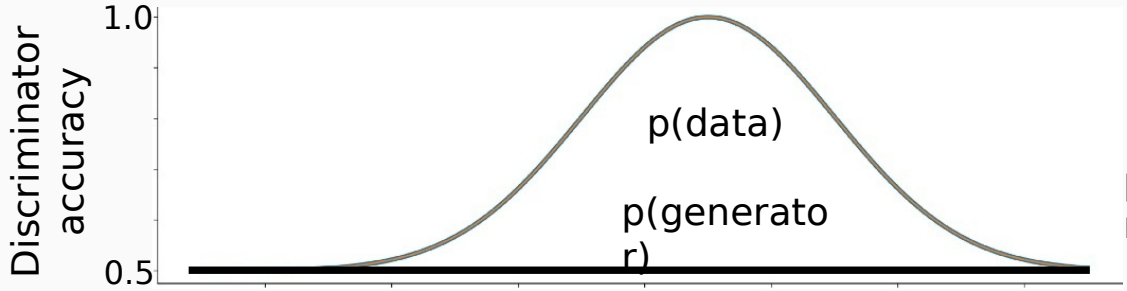
Generator
Learns to transform noise into fakes in order to fool the Discriminator

$$\max_{\phi} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

vs

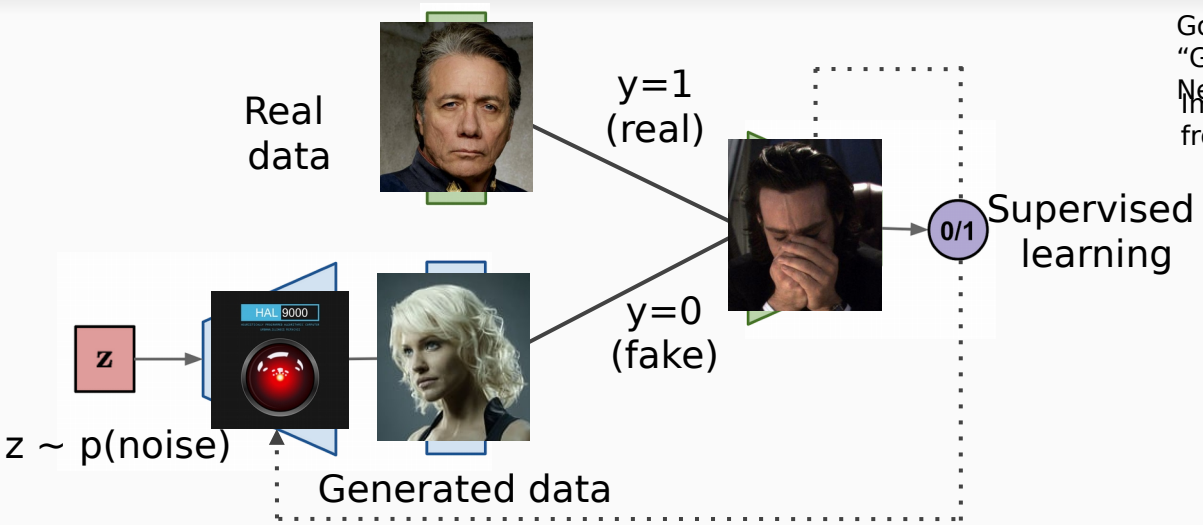
Discriminator
Tries to tell the difference between real examples (data) and fakes

$$\max_{\phi} \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$



Nash Equilibrium (Game theory)
p(generator) ≈ p(data)

Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator
Learns to transform noise into fakes in order to fool the Discriminator

$$\max_{\phi} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

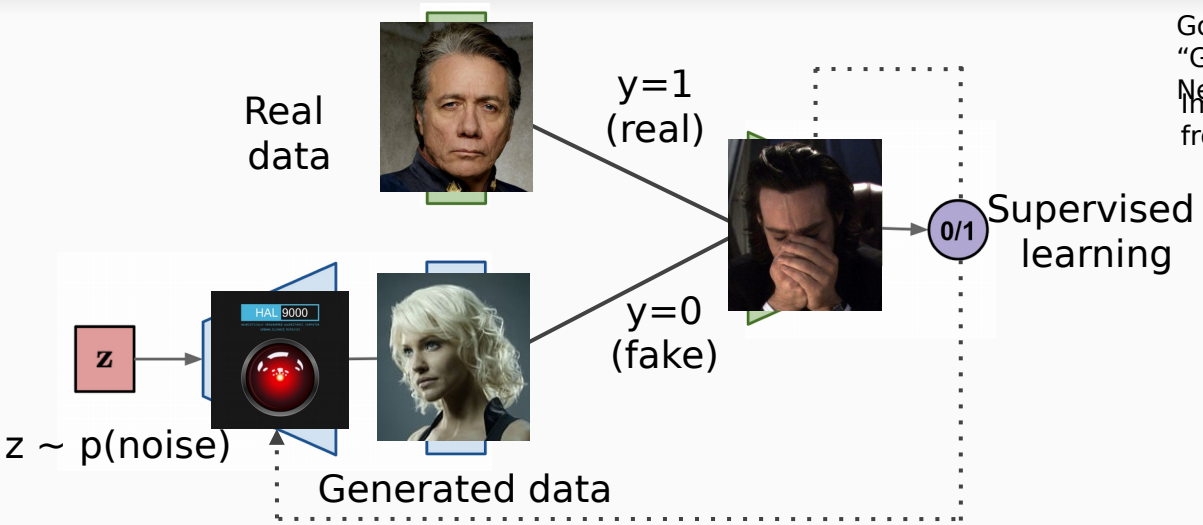
vs

Discriminator
Tries to tell the difference between real examples (data) and fakes

$$\max_{\phi} \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

Rapid improvement in image quality over time

Implicit Density Estimation: Generative Adversarial Networks



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014
Image adapted from: Lilian Weng

Generator
Learns to transform noise into fakes in order to fool the Discriminator

$$\max_{\phi} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(D_\phi(G_\theta(\mathbf{z})))]$$

vs

Discriminator
Tries to tell the difference between real examples (data) and fakes

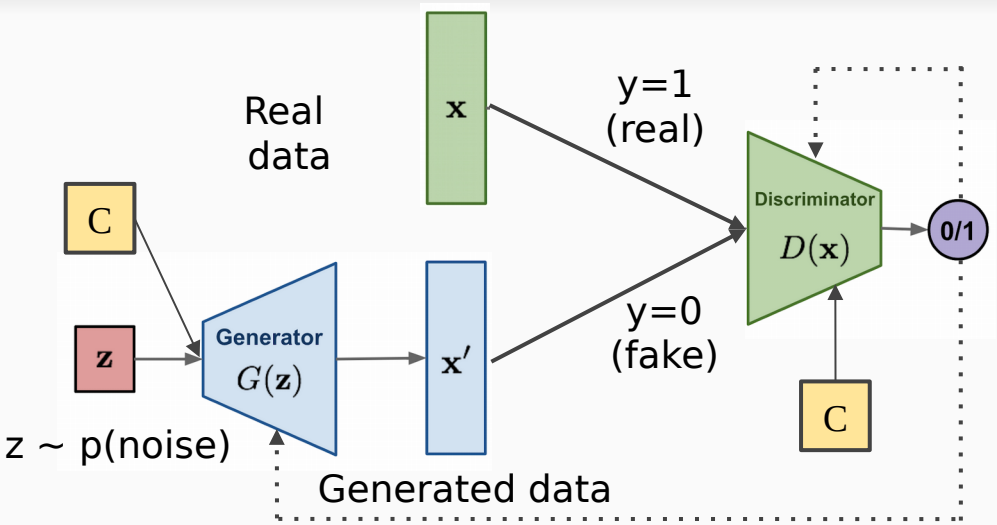
$$\max_{\phi} \mathbb{E}_{\mathbf{x}\sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

Rapid improvement in image quality over time

Many variants of GANs:
E.g. BiGAN, CycleGAN, BigGAN, InfoGAN, etc

See: GAN Zoo and

Conditional GANs



Goodfellow et al, "Generative Adversarial Nets", NIPS 2014

Image adapted from: Lilian Weng

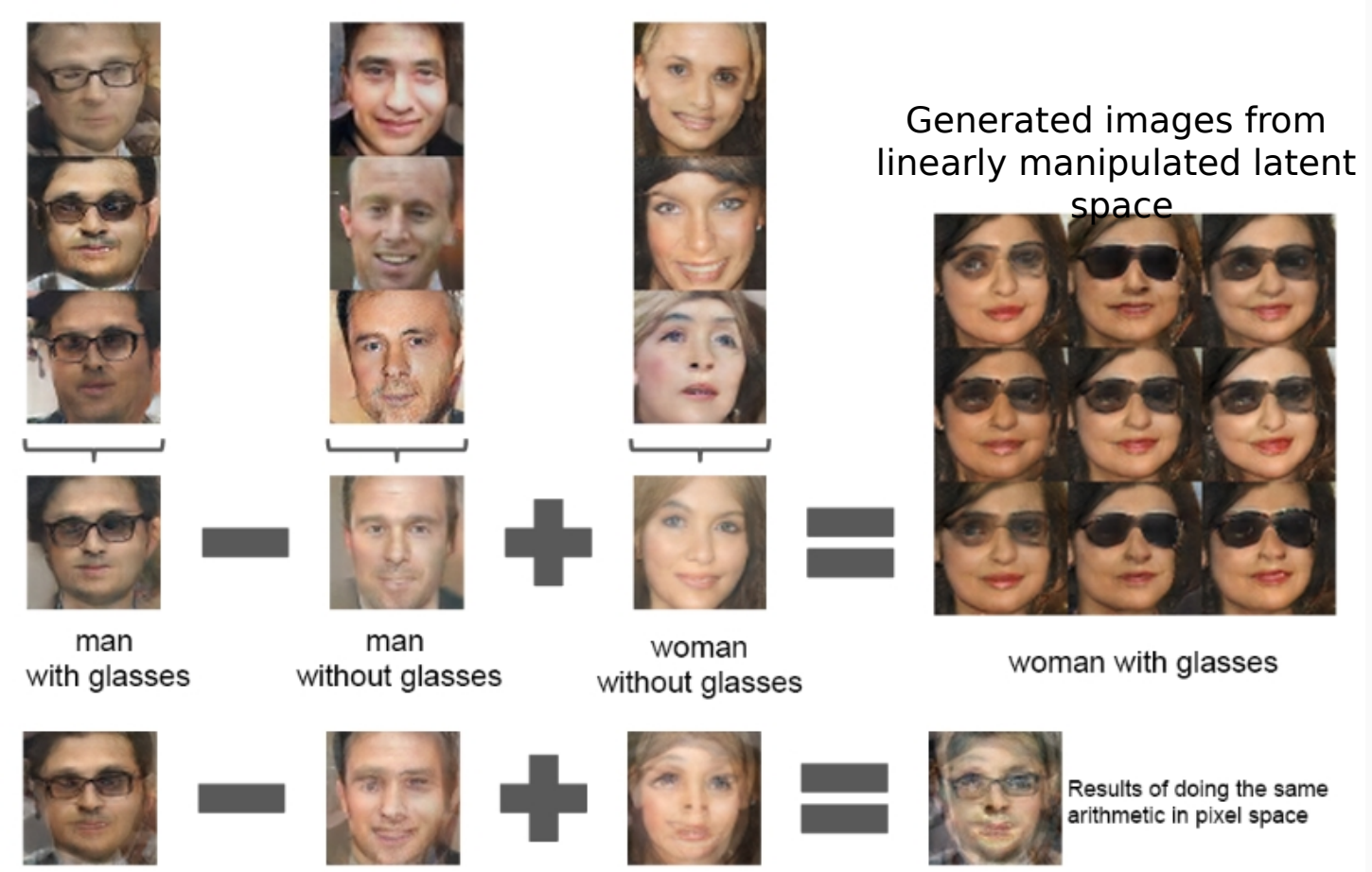
E.g. text-image pairs  ,
human)

Supervised Generative Model

Latent space manipulations with GANs

Generated samples from latent vectors conditioned on text

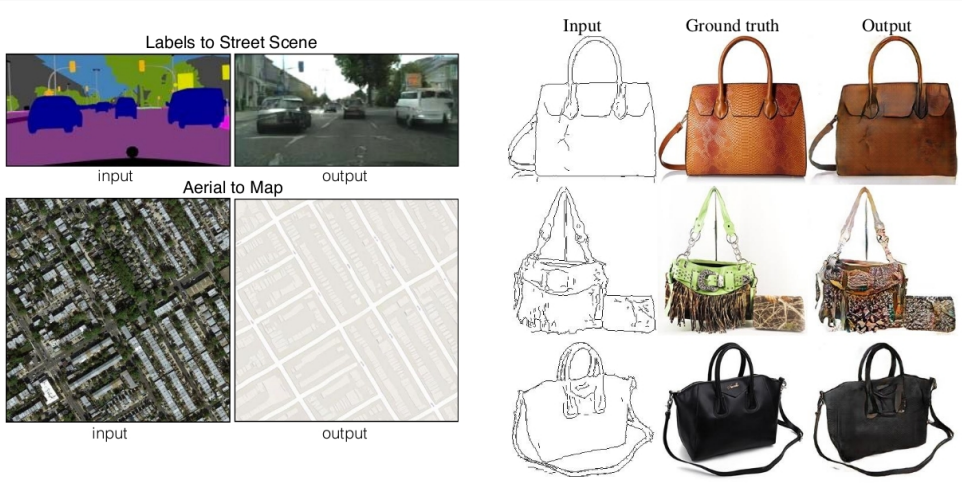
Example from averaged latent vector



Radford et al, ICLR 2016

Fun GAN examples

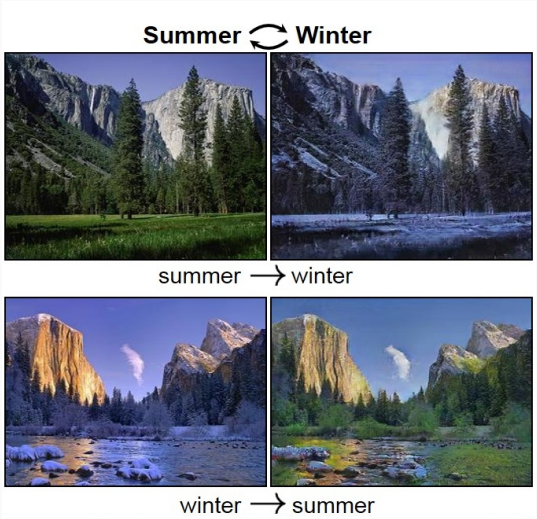
Image to image translation



Pix2Pix using Conditional GANs (Isola et al, 2016)

Combine latent spaces to create novel images

GANBreeder using BigGAN (Brock et al, 20)

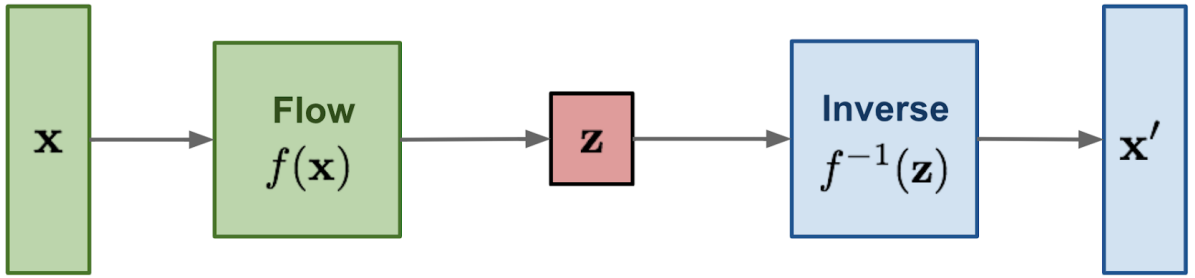


Style Transfer, Image to image translation

Pix2Pix using CycleGAN

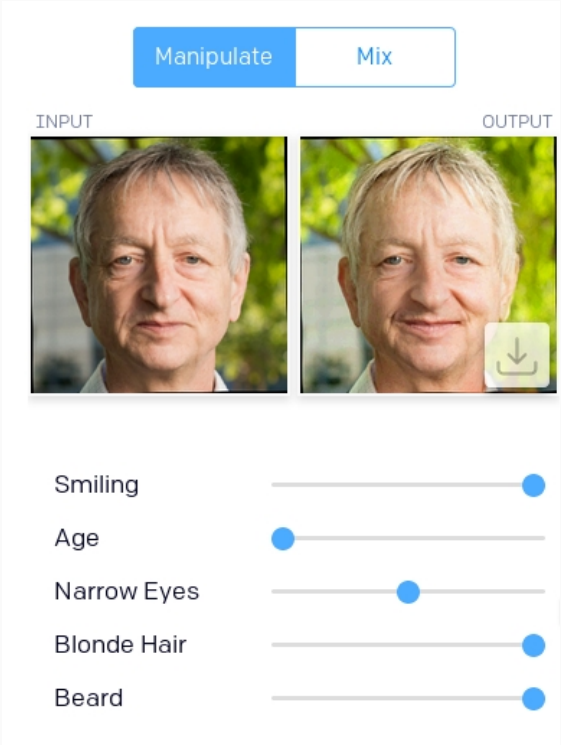
Flow-Based Models (Brief)

Flow-based generative models: minimize the negative log-likelihood



Explicitly learns probability density

Works in latent space



Kingma, Dhariwal (2018) Glow: Generative Flow with Invertible 1x1 Convolutions

Which reality TV star is ~~fake~~ real?

Which celebrity is ~~fake~~ real?