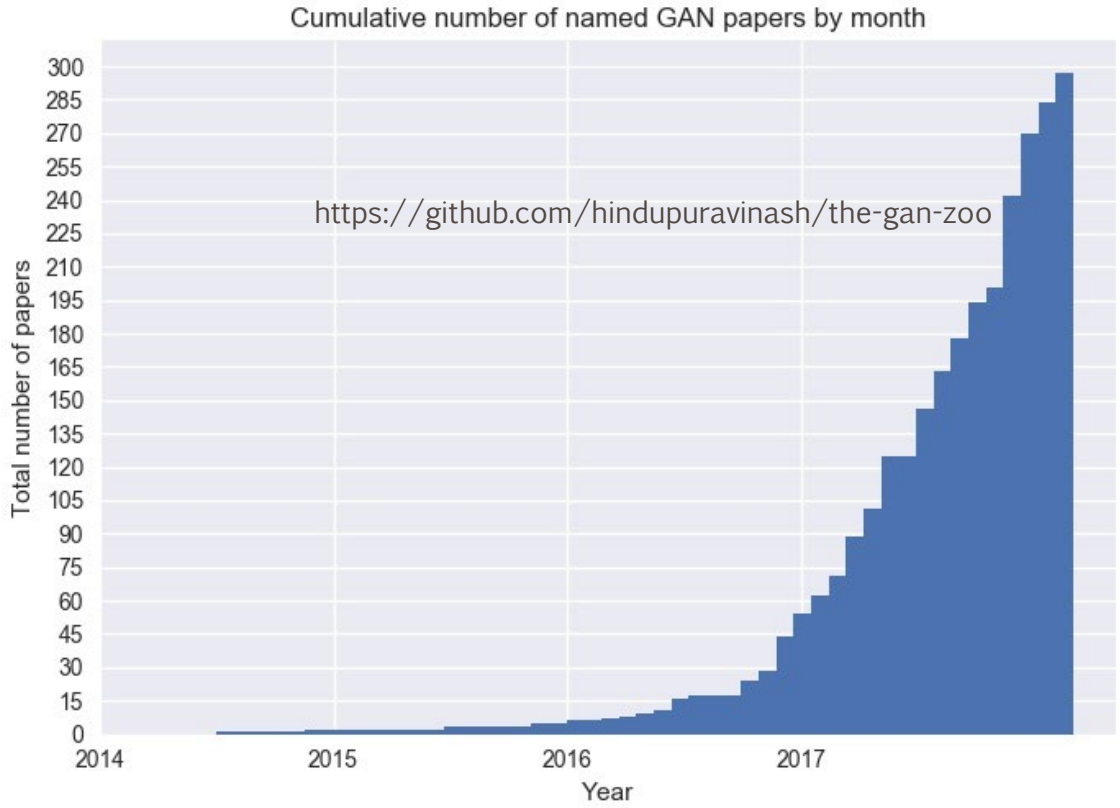
GENERATIVE MODELS

Chih-Chung Hsu (許志仲)
Institute of Data Science
National Cheng Kung University
https://cchsu.info

All Kinds of GAN …

GAN
ACGAN
BGAN
CGAN
DCGAN
EBGAN
fGAN
GoGAN
⋮



Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017

[2]We use the Greek α prefix for α-GAN, as AEGAN and most other Latin prefixes seem to have been taken

https://deephunt.in/the-gan-zoo-79597dc8c347

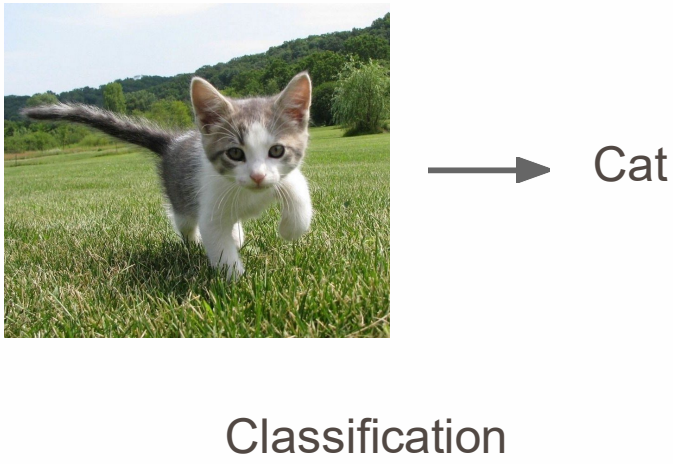
Supervised vs. Unsupervised

- Supervised learning:
  - We have correct answer to learn!!
    - Exactly learning the proper parameters!!
  - There are several state-of-the-art methods!!
- Unsupervised learning:
  - We don’t have the correct answer
    - Only “guess”
  - Need to have a lot of strategies to fine-tune the estimated answer
  - Not good enough algorithm so far

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

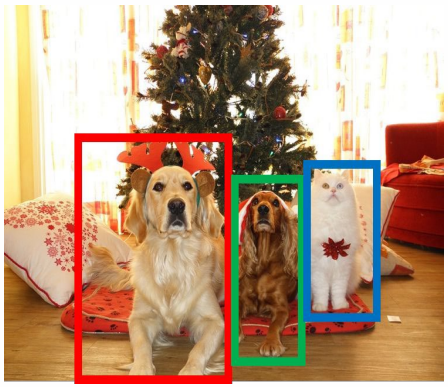
- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



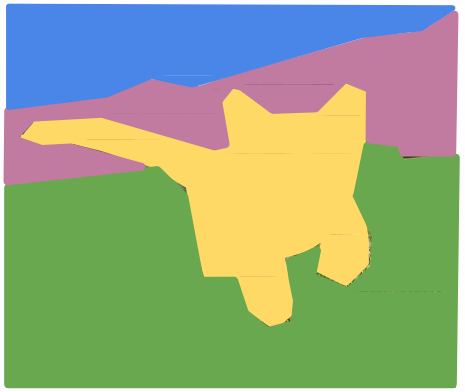
DOG, DOG, CAT

Object Detection

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



GRASS, CAT, TREE, SKY

Semantic Segmentation

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

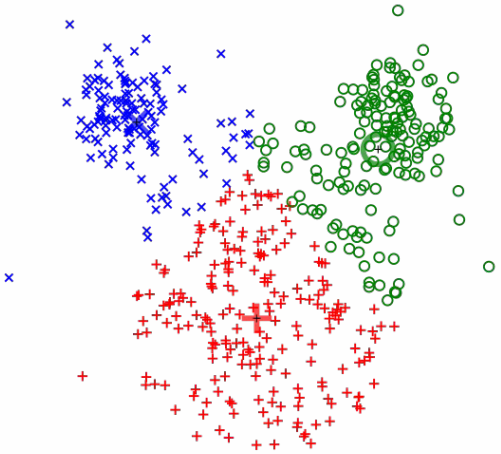
A cat sitting on a suitcase on the floor

Image captioning

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

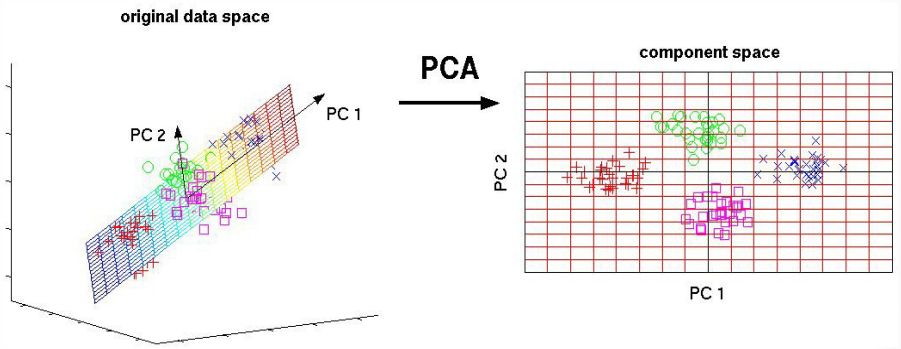


K-means clustering

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



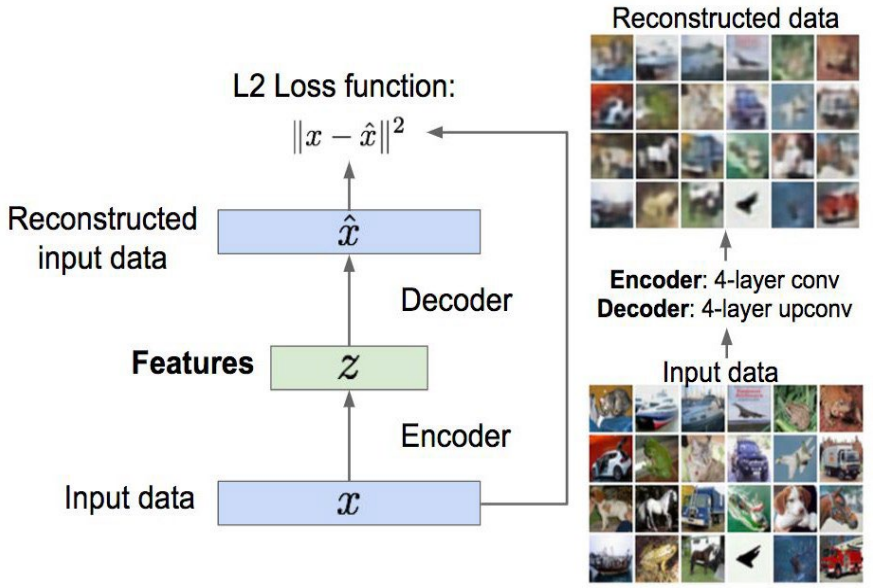
3-d 2-d

Principal Component Analysis (Dimensionality reduction)

Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



Autoencoders (Feature learning)

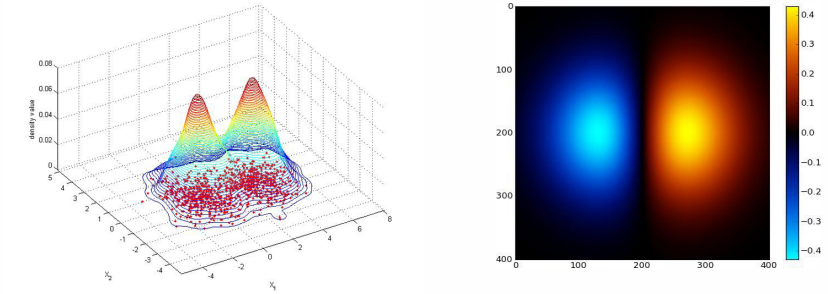
Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label

Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation

- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.



2-d density estimation

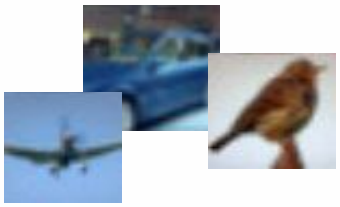
Supervised vs Unsupervised Learning

- Supervised Learning
- Data: (x, y)
- x is data, y is label
- Goal: Learn a function to map x -> y
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

- Unsupervised Learning
- Data: x ← Training data is cheap
- Just data, no labels!
- Goal: Learn some underlying hidden structure of the data
- Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Solve unsupervised learning
=> understand structure of visual world

Generative Models



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Given training data, generate new samples from same distribution

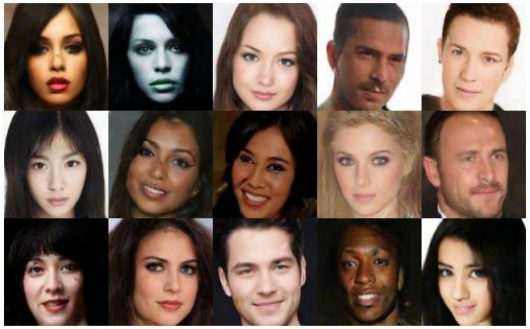
Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.

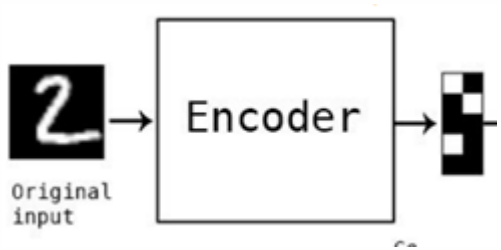

- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
- Training generative models can also enable inference of latent representations that can be useful as general features

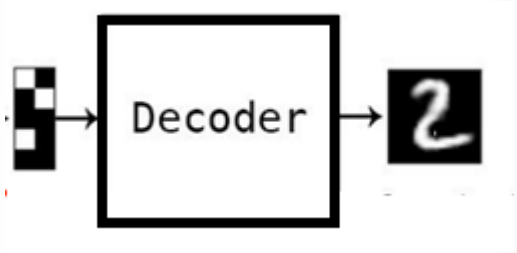
Unsupervised Learning

- More challenging than supervised learning :
  - No label or curriculum → self learning
- Traditional solutions:
  - Clustering
  - Linear / nonlinear dimensionality reduction
    - PCA vs. Manifold learning
- Some NN solutions :
  - Boltzmann machine
  - Auto-encoder or Variational Inference
  - Generative Adversarial Network

Unsupervised learning vs. Generative model

- Unsupervised learning
  - z=f(x)
- Generative model
  - x=g(z)
- It is …
  - P(z|x) vs. P(x|z)
  - An encoder vs. a decoder
    - Encoder: Feature extraction / Dimensionality reduction
    - Decoder: Generator / Upsampling
    - P(z|x) = P(x, z) / P(x) ➔ P(x) Intractable (ELBO)
    - P(x|z) = P(x, z) / P(z) ➔ P(z) is prior





P(x,z) is necessary!!

Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

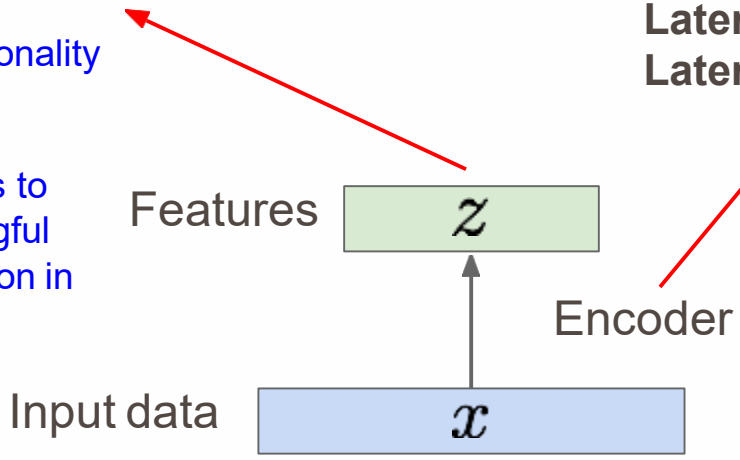
Q: Why dimensionality reduction?

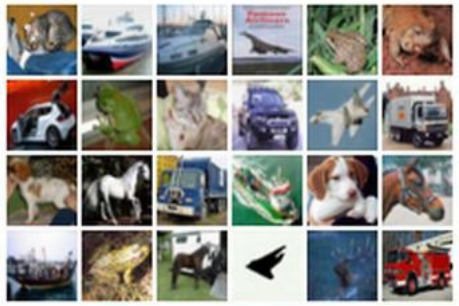
A: Want features to capture meaningful factors of variation in data

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN

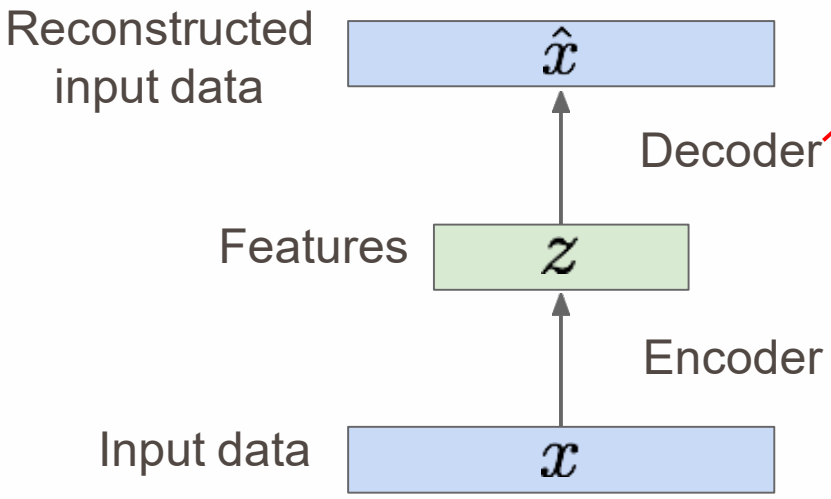




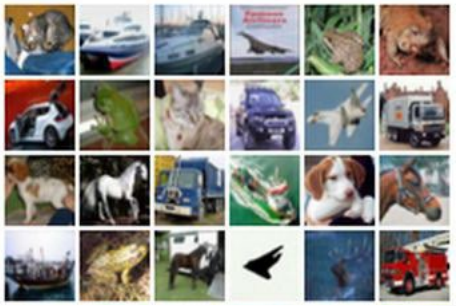
Some background first: Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data "Autoencoding" - encoding itself



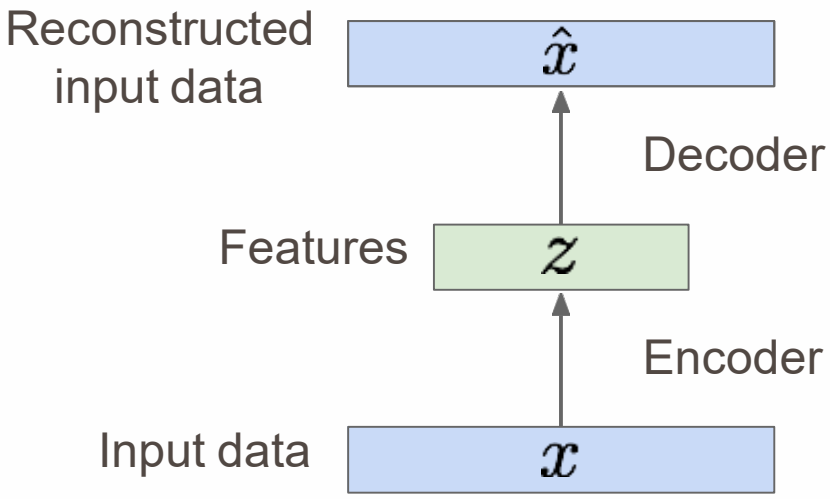
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)



Some background first: Autoencoders

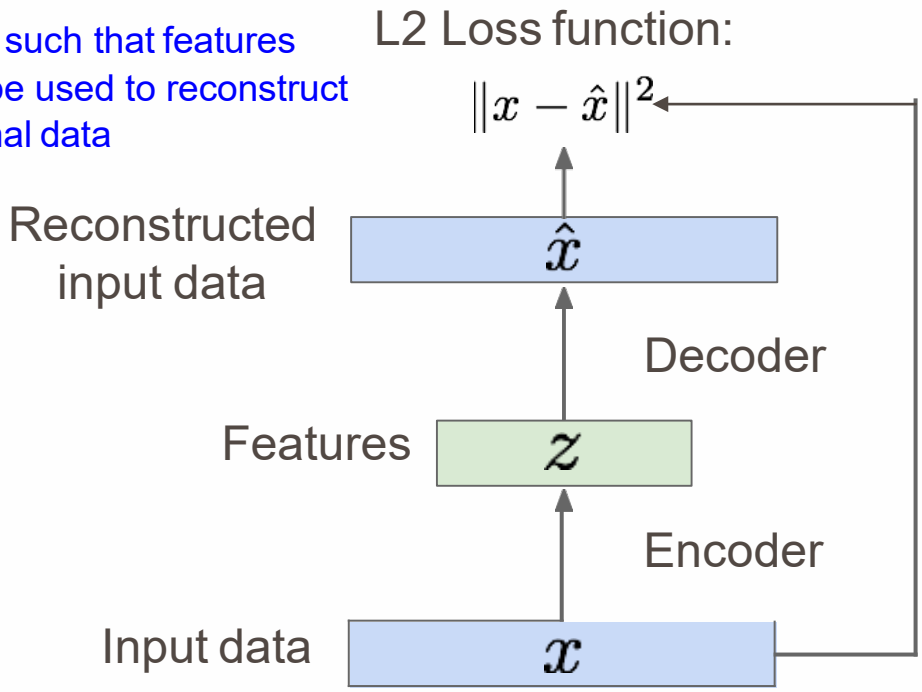
How to learn this feature representation?

Train such that features can be used to reconstruct original data "Autoencoding" - encoding itself

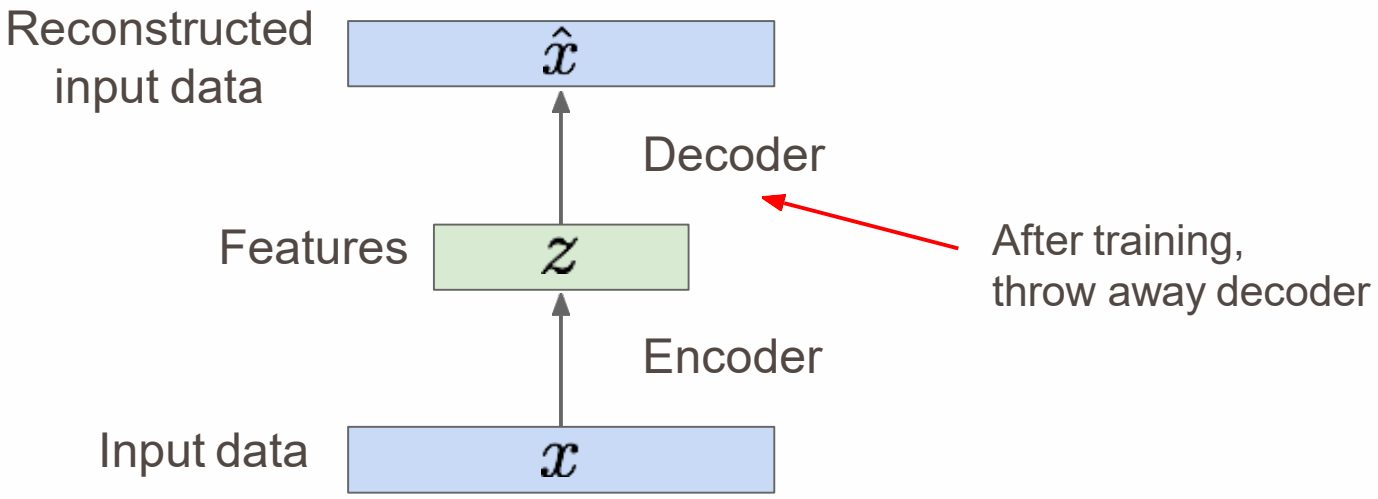


Some background first: Autoencoders

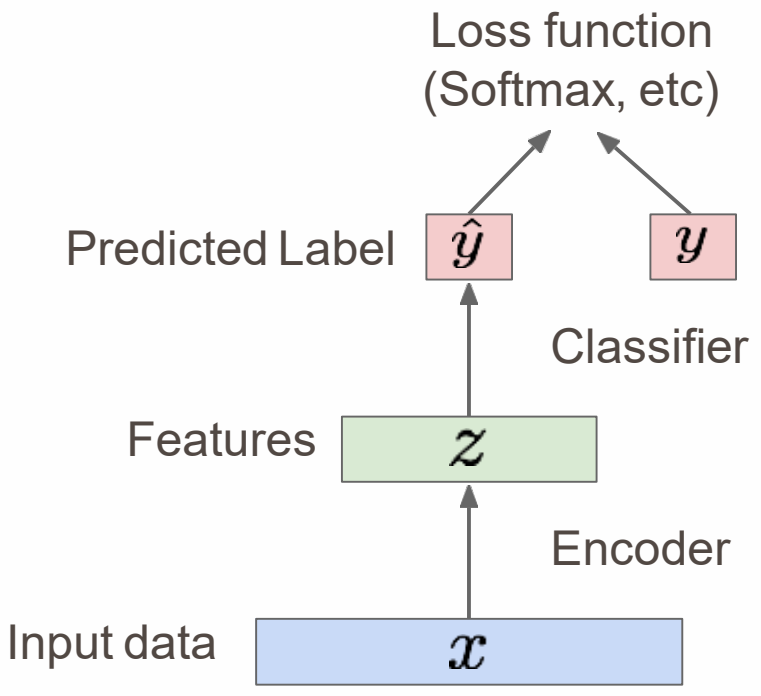
Train such that features can be used to reconstruct original data



Some background first: Autoencoders



Some background first: Autoencoders



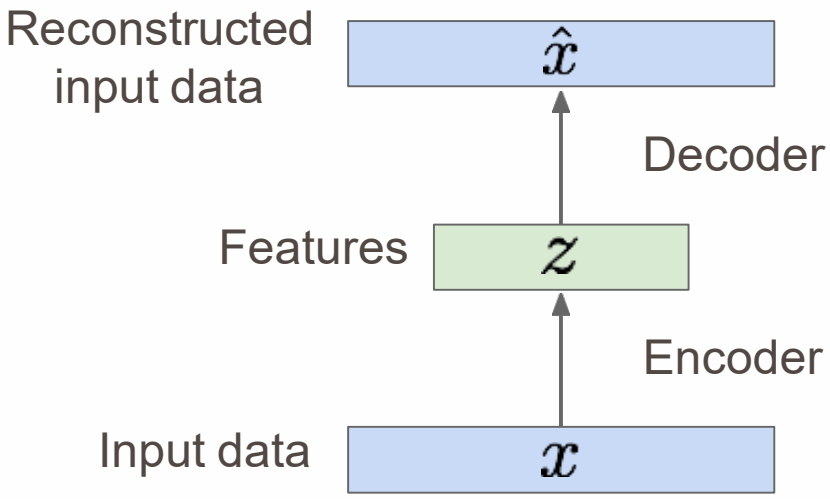
Encoder can be used to initialize a **supervised** model

Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)

bird plane
dog deer truck

Some background first: Autoencoders



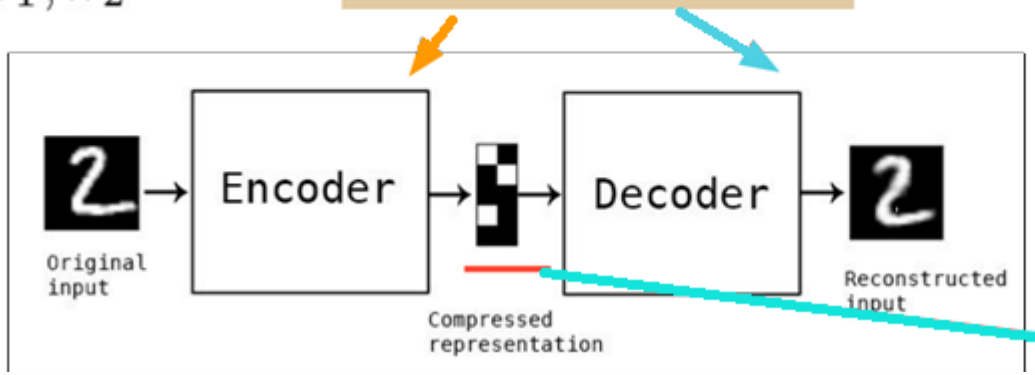
Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data. Can we generate new images from an autoencoder?

Unsupervised Deep Learning: AutoEncoder

- With no answer “data”
  - Use “Reconstruction” to learn!!
  - A good representation should keep the information well (reconstruction error)
  - Deep + nonlinearity might help enhance the representation power

$$\min_{\mathbf{w}_1,\mathbf{w}_2} \left\| \mathbf{x}_i - g\left(f\left(\mathbf{x}_i; \mathbf{w}_1\right); \mathbf{w}_2\right) \right\|_2^2$$

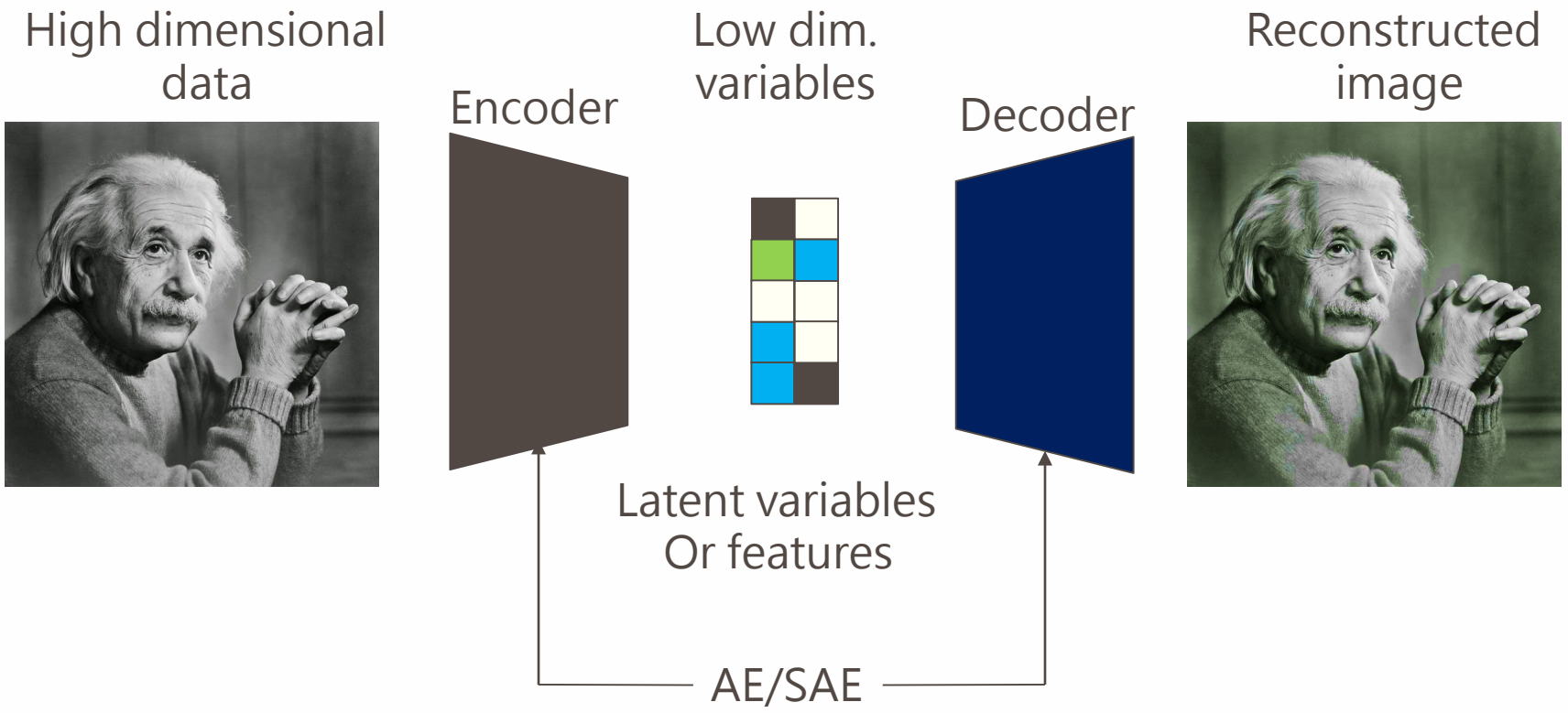


Credit: LeCun

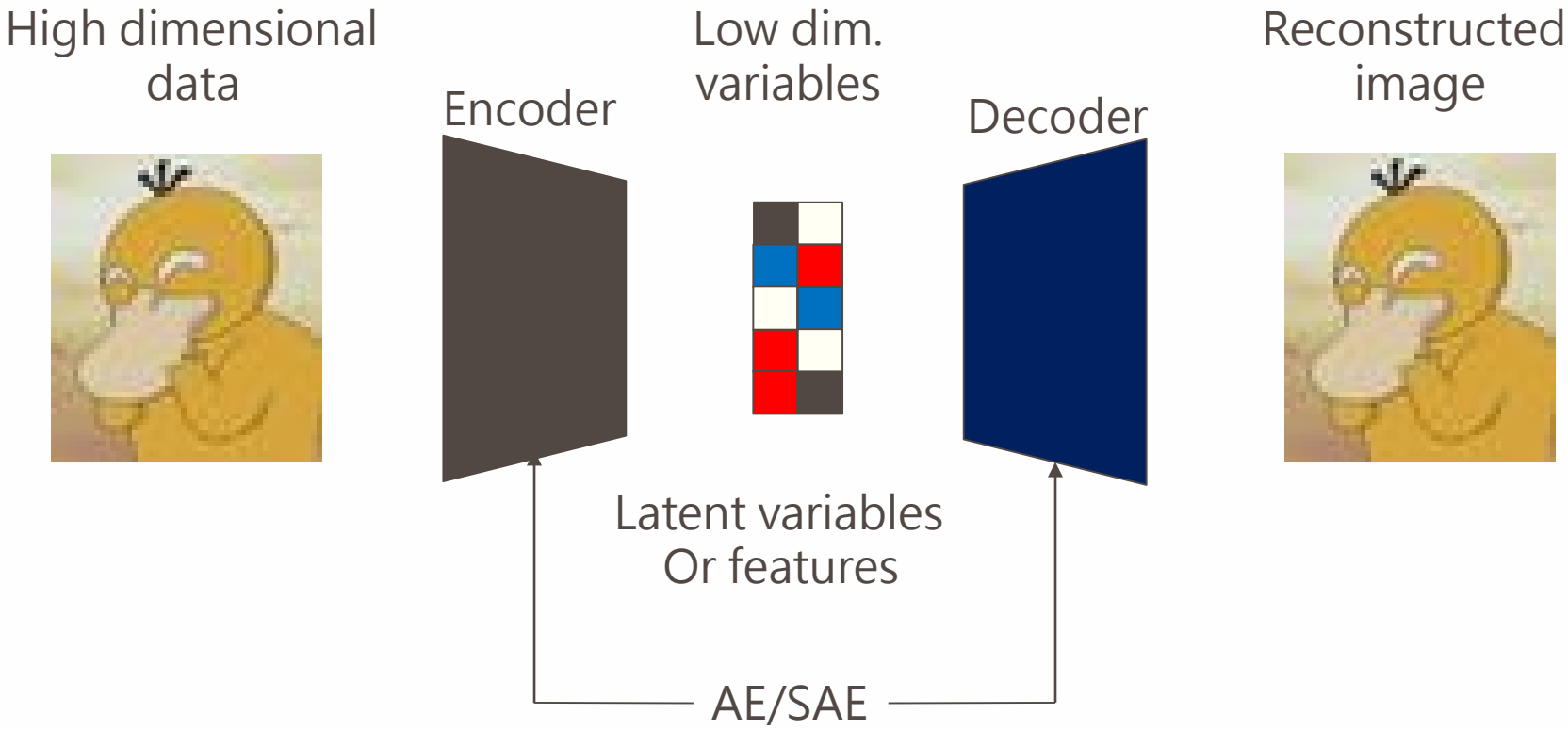
Deep Version of AutoEncoder

- Stacked autoencoder (SAE)
- Similar to AE but deeper
- Use CNN/Fully connected layers

What Exactly AE is?



What Exactly AE is? (cont.)



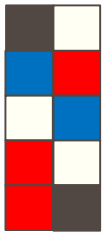
What Exactly AE is? (cont.)

Low dim. variables 1

Decoder

Reconstructed image 1

For example
0.666 = 可達鴨

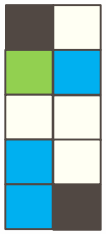


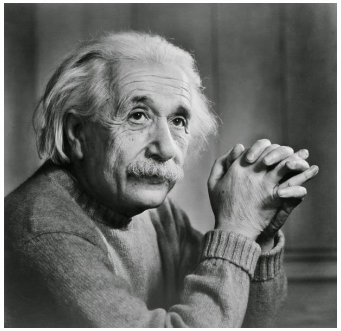
Low dim. Variables 2

Decoder

Reconstructed image 2

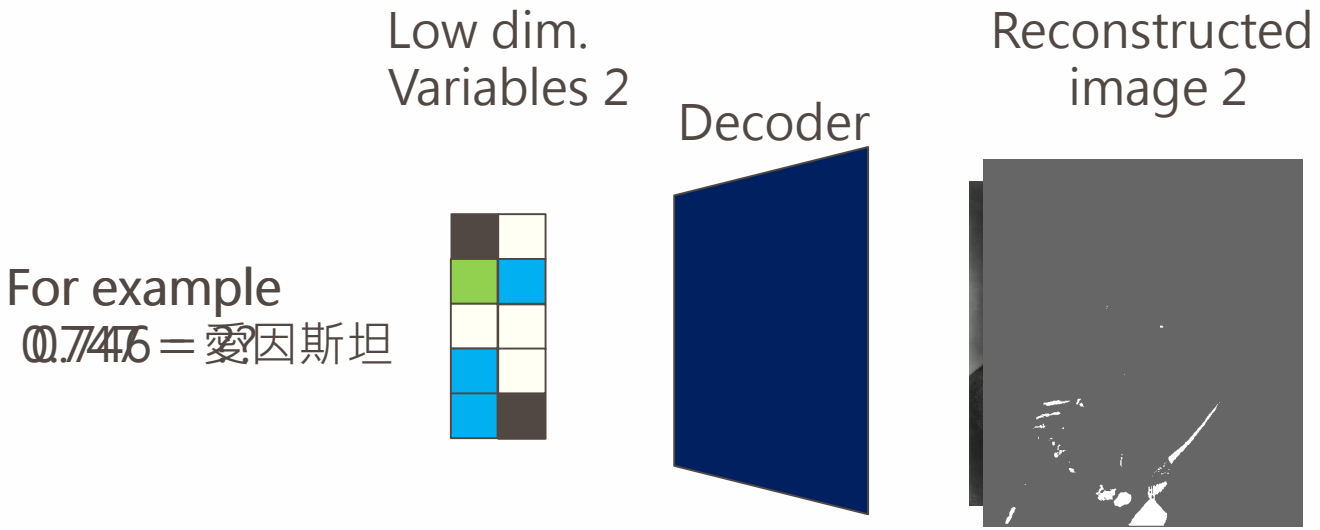
For example
0.747 = 愛因斯坦





Problem in SAE/AE

- One feature corresponds to one reconstructed image!
  - Feature is generated from Encoder….
    - Such AE/SAE cannot be used to generate arbitrary images

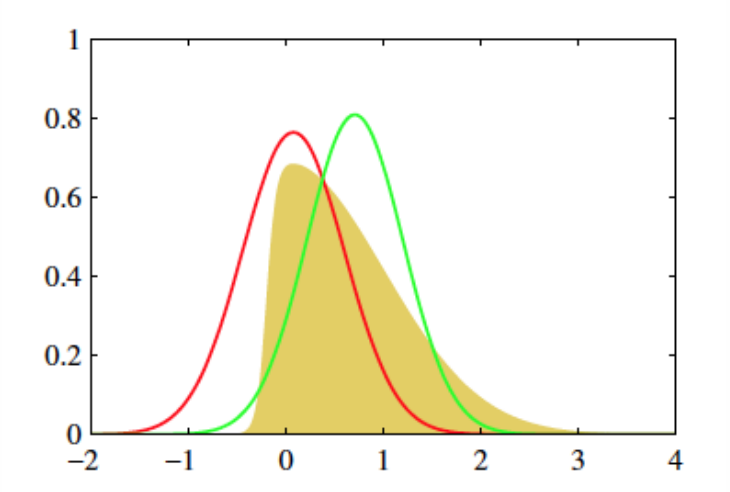


Improved AutoEncoder

- Variational autoencoder - VAE
- Kingma et al, “Auto-Encoding Variational Bayes”, 2013.
  - Generative Model + Stacked Autoencoder
  - Based on Variational approximation
- From AE to VAE
  - Since the feature (latent variable) is not continuous
    - Explicit feature is required for generating an image
  - MODELING feature instead!!

Variational Inference

- Target: p
  - Hard to find their distribution
- We assumed that it likes to Gaussian distributions, green and red lines *q*
  - Which one closes to the real distribution p, then we choose that q as the solution
    - Minimize distance between distributions!!



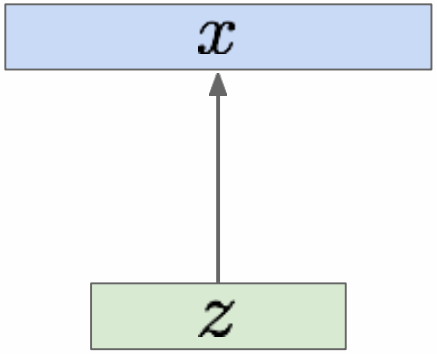
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



Intuition (remember from autoencoders!): **x** is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

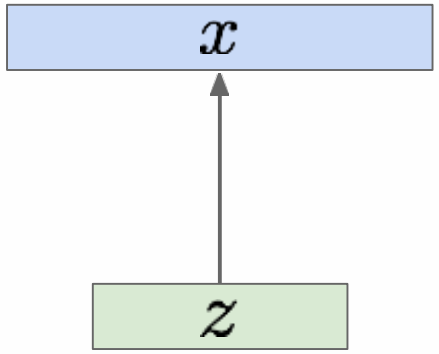
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

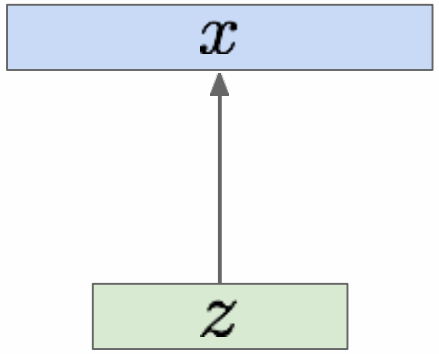
Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$

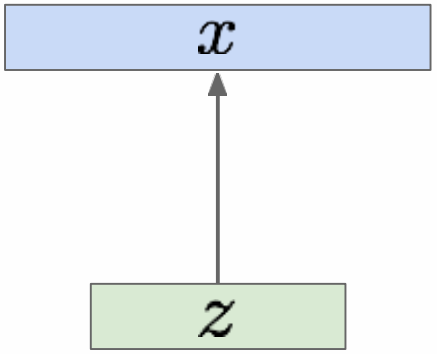


Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

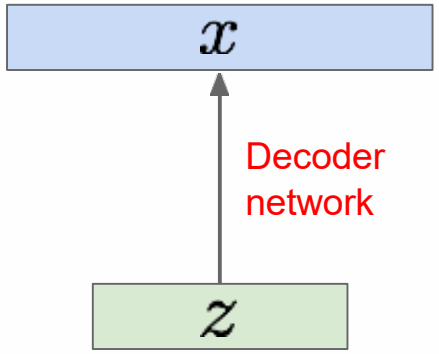
Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian.

Conditional p(x|z) is complex (generates image) => represent with neural network

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

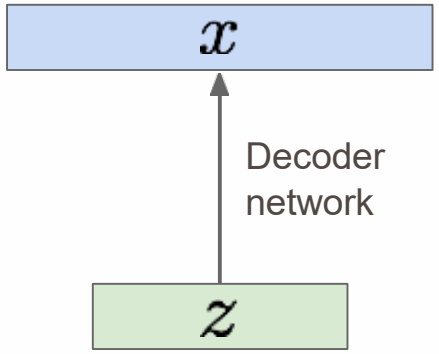
Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$

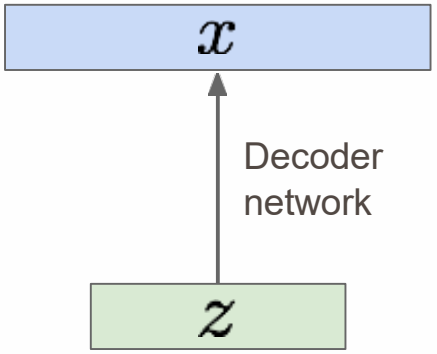


Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Strategy for training generative models from FVBNs (fully visible belief networks, Deep Belief nets). Learn model parameters to maximize likelihood of training data

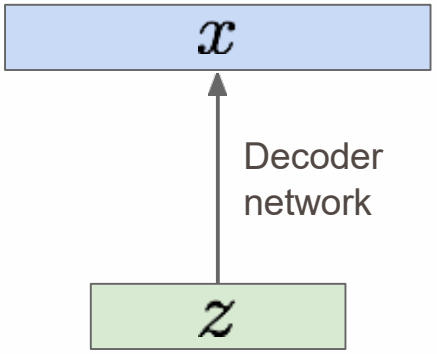
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

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

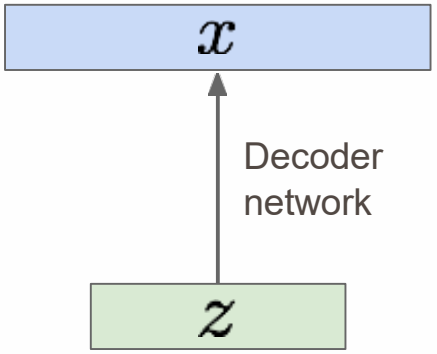
Now with latent z

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Strategy for training generative models from FVBNs, Learn model parameters to maximize likelihood of training data

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

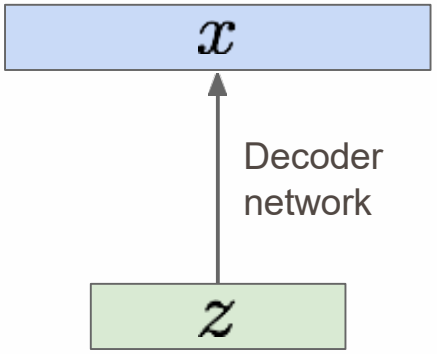
Q: What is the problem with this?

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

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

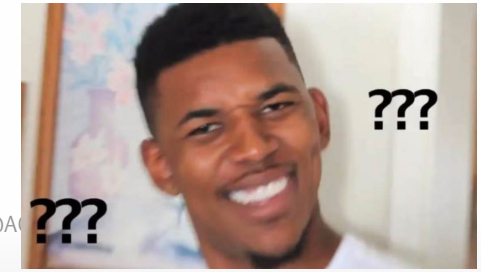
Q: What is the problem with this?

Intractable!

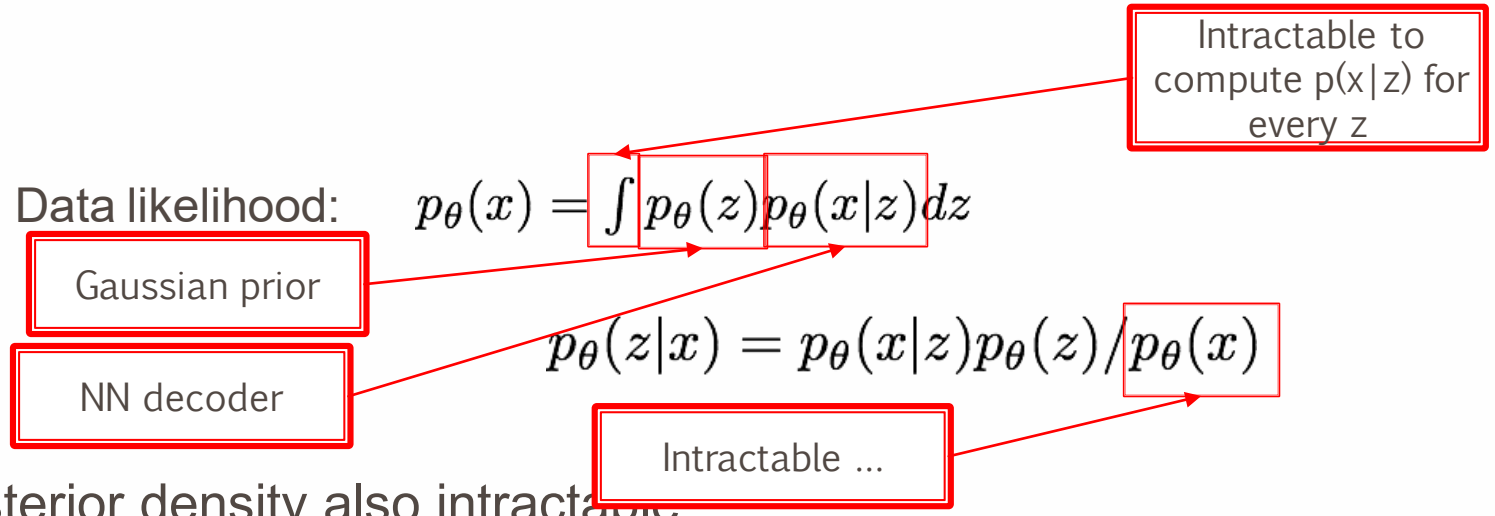
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

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

Given p(x)



Variational Autoencoders: Intractability



- Posterior density also intractable:

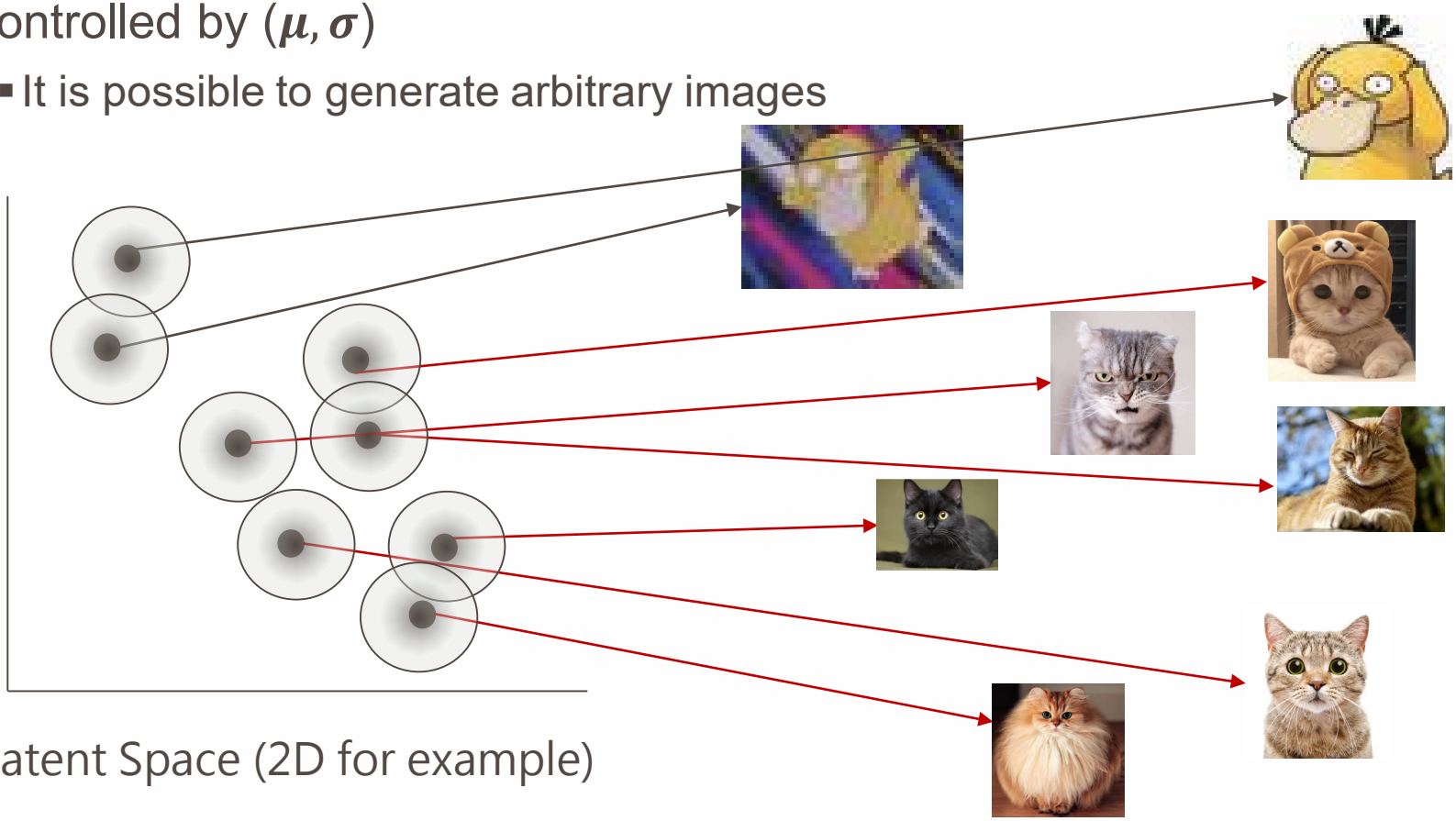
Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(x|z)$,

- Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

From AE to VAE

- Modeling: Assume the feature is sampled from Gaussian controlled by (μ, σ)
  - It is possible to generate arbitrary images



Latent Space (2D for example)

Reconstructed images

From AE to VAE

- In this way, loss function can be defined as
  - $L_{data} = \|X - \bar{X}\|_2^2$, where $\bar{X}$ is the reconstructed image
  - $L_{latent} = \mathrm{KL}(\mathrm{P}|\mathrm{Q})$➔ KL(Latent variables, Gaussian)
  - $L = L_{data} + L_{latent}$

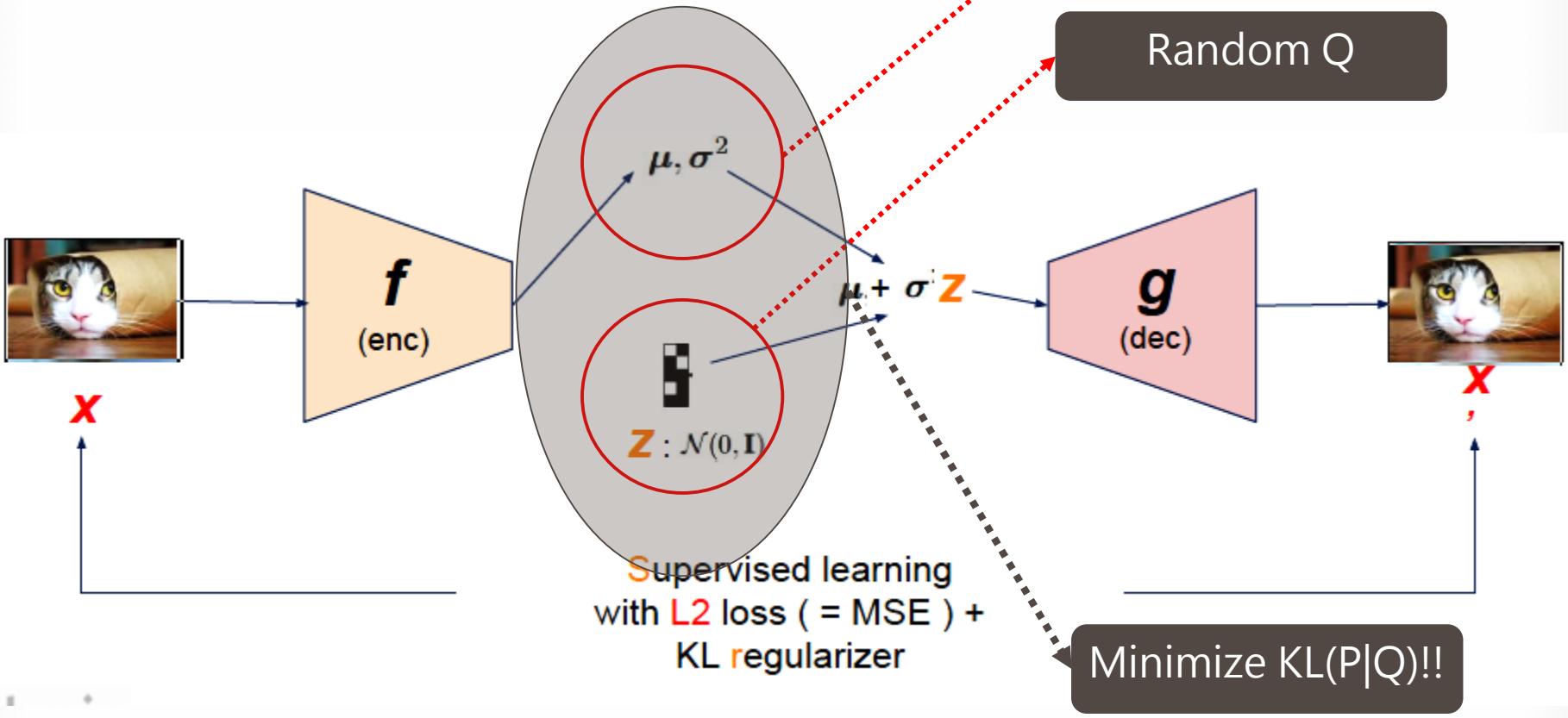
- Difficult to optimize L
  - The distribution of latent variables is unknown & uncontrollable.
- Solution:
  - Force latent variable to be a parameters of a specified distribution: Encoder ➔ $(\boldsymbol{\mu}, \boldsymbol{\sigma})$

Variational AE (VAE)

- Minimize KL(P|Q)!!
  - ➔ Variational inference!!
- Recall that
  - P(z|x) = P(x, z) / P(x) ➔ P(x) Intractable (ELBO)
  - Approximation solution
    - Use q(z|θ) to approximate P(z|x)
    - Variational inference!
- Shortcoming
  - Blurred images will be generated (no guarantee its quality)

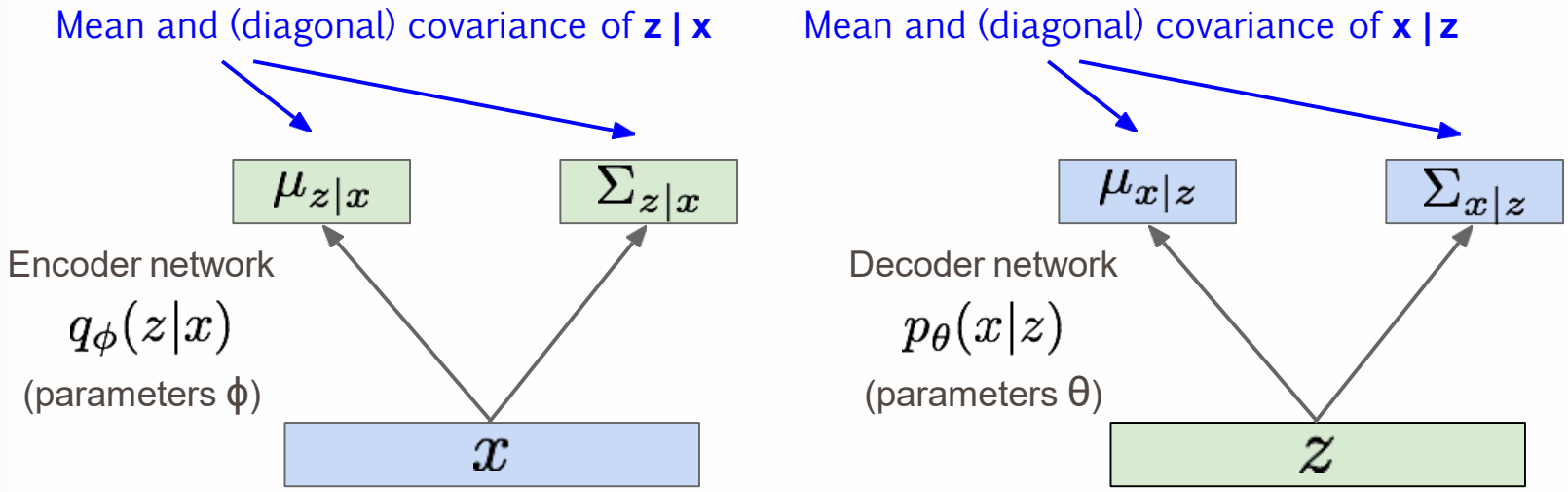
From AE to VAE

- Explicit feature is required for generating an image
  - MODELING feature instead!!



Variational Autoencoders

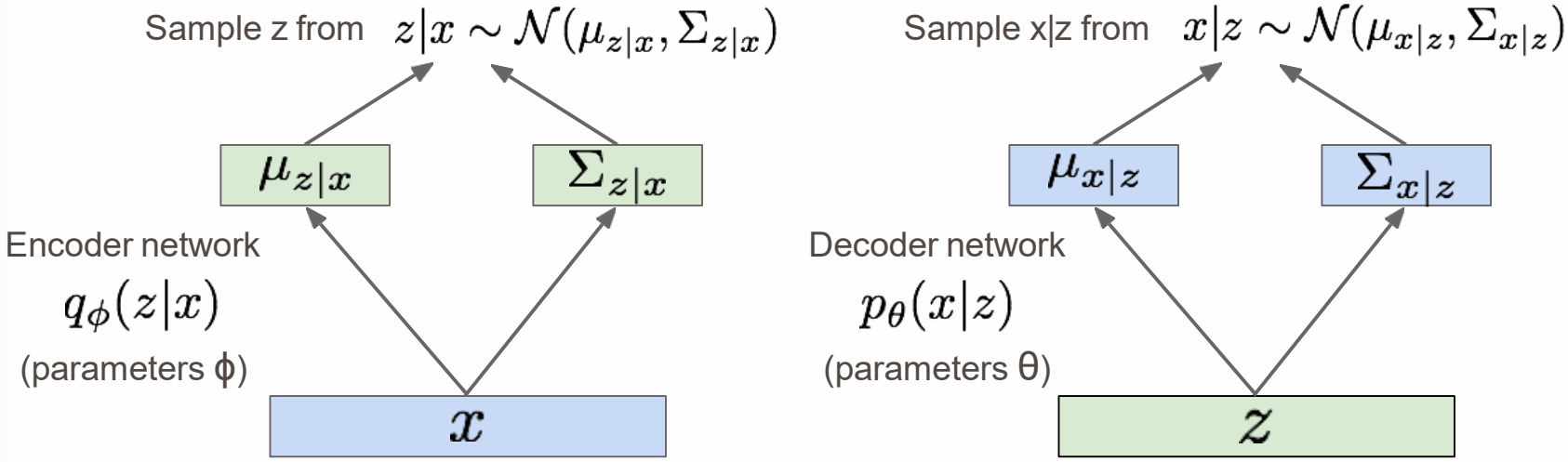
Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

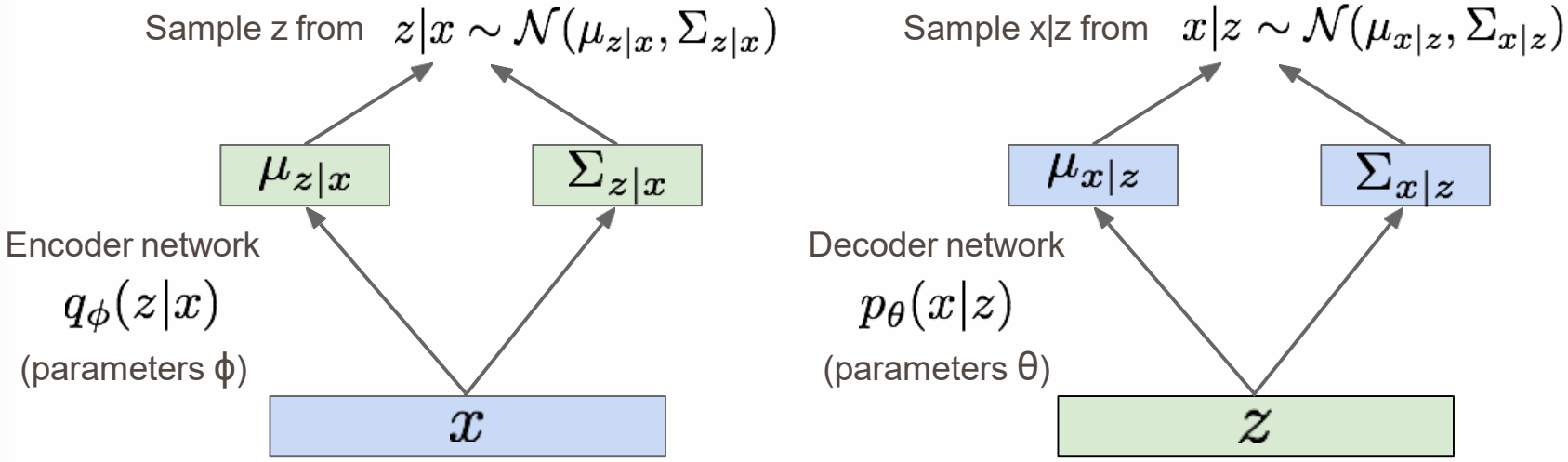
Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

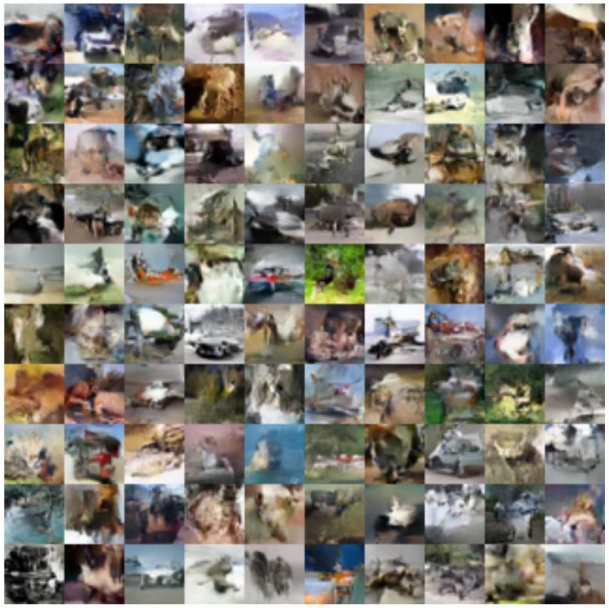
Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Encoder and decoder networks also called "recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders: Generating Data!



32x32 CIFAR-10

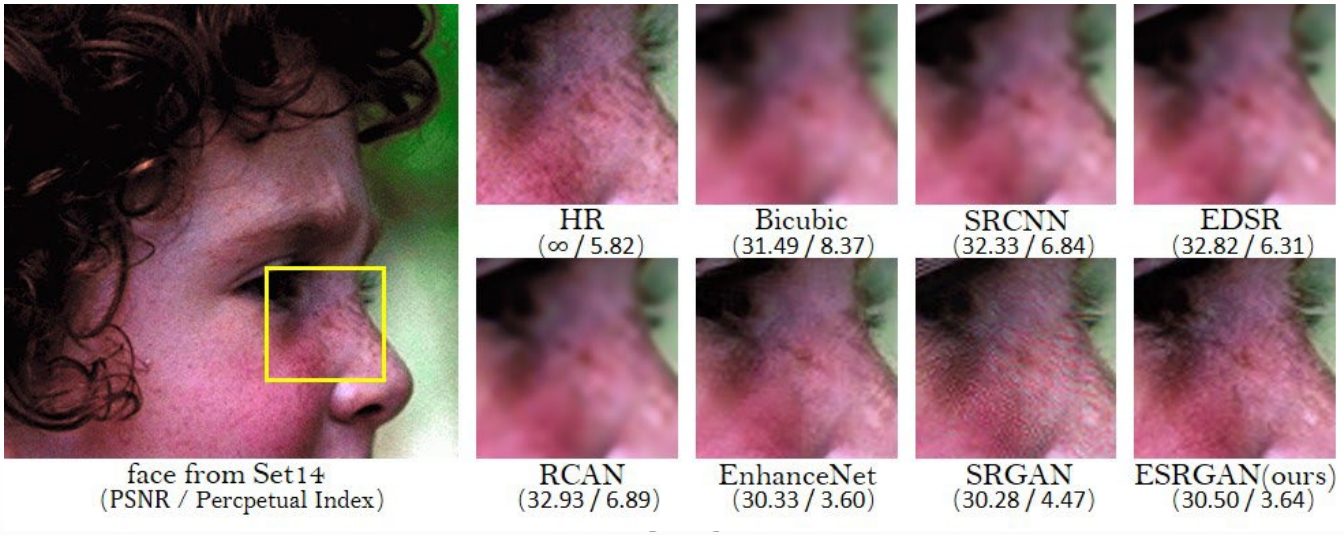
Labeled Faces in the Wild

Variational Autoencoders

- Probabilistic spin to traditional autoencoders => allows generating data
- Defines an intractable density => derive and optimize a (variational) lower bound
- Pros:
  - Principled approach to generative models
  - Allows inference of q(z|x), can be useful feature representation for other tasks
- Cons:
  - Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
  - Samples blurrier and lower quality compared to state-of-the-art (GANs)
- Active areas of research:
  - More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs)
  - Incorporating structure in latent variables, e.g., Categorical Distributions

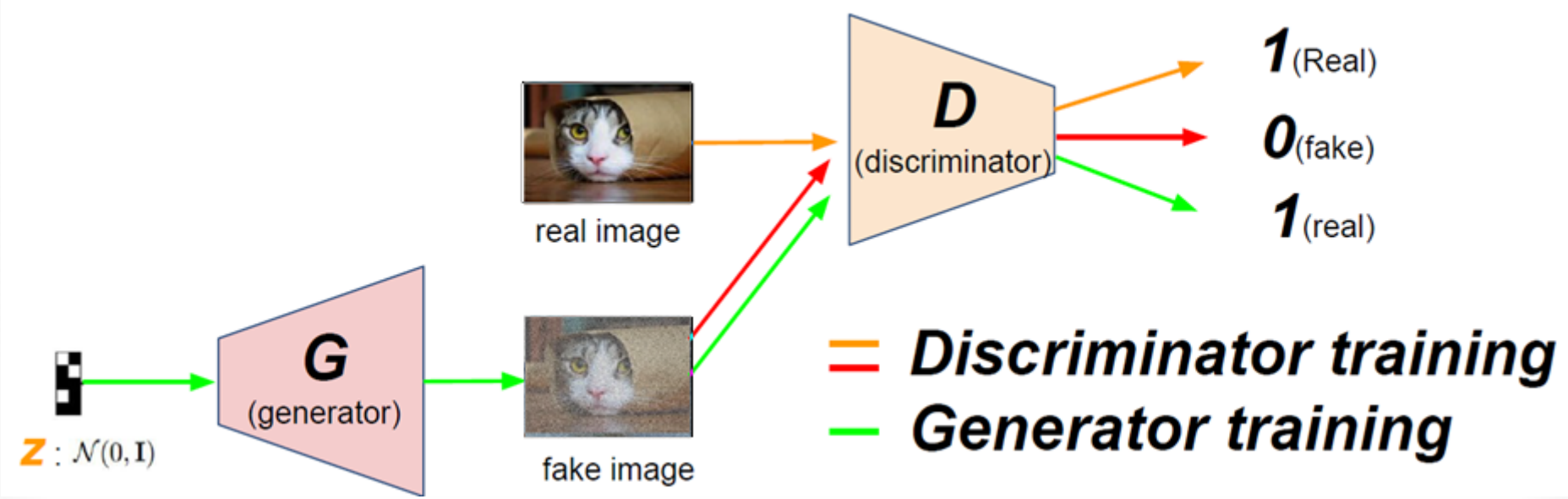
But Why?

- Recall that we use “pixel” to measure the quality
  - So what?
  - It is well-known that there is no promising metric that can reflect the truly perceptual quality (visual quality)
- Example
  - PSNR
  - Some perceptual index were proposed to resolve this issue.



Unsupervised Deep Learning

- How to generate an image with good quality?
  - Generative adversarial network (GAN)



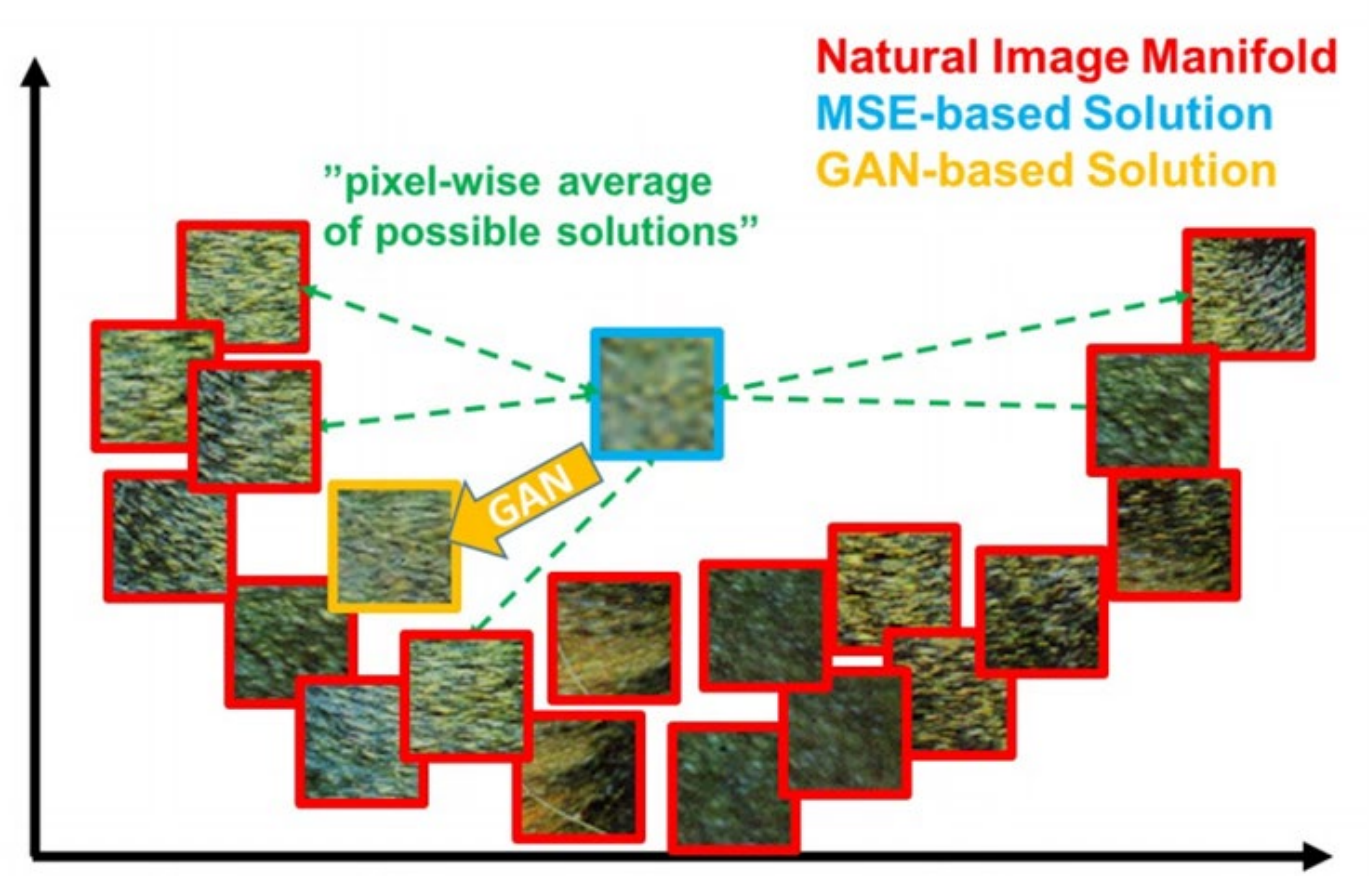
Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Why Generative Models?

- Excellent test of our ability to use high-dimensional, complicated probability distributions
- Simulate possible futures for planning or simulated RL
- Missing data
  - Semi-supervised learning
- Multi-modal outputs
- Realistic generation tasks

Generating an Image using GAN

- Learn and predict P(x|z)



Ledig et al. CVPR 2017

Training Procedure

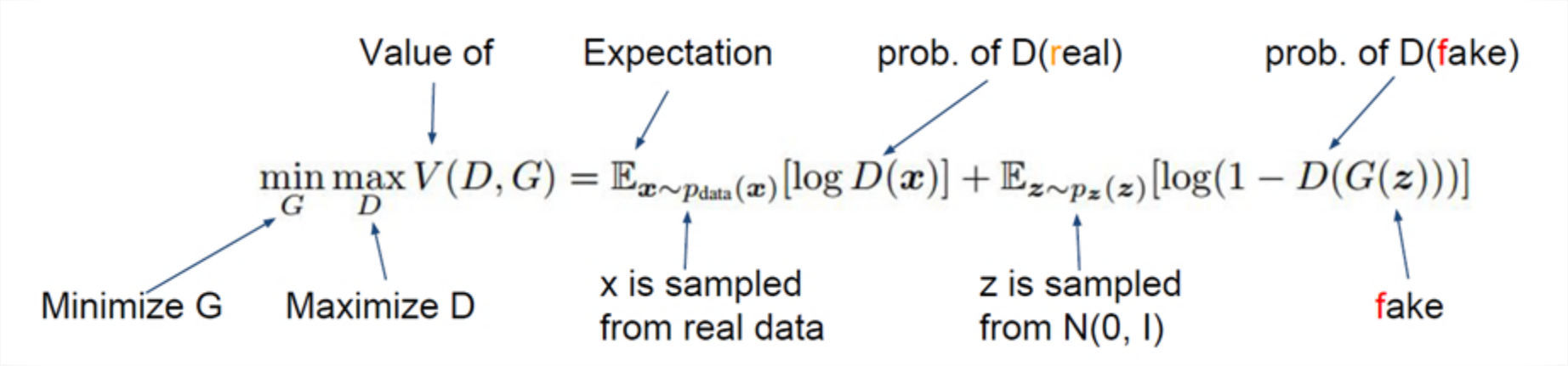
- Use SGD-like algorithm of choice (Adam) on two mini-batches simultaneously:
  - A mini-batch of training examples
  - A mini-batch of generated samples
- Optional: run k steps of one player for every step of the other player.

(Goodfellow 2016)

partial credit by CS311n

The Cost Function of GAN

- Notation



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

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images
- Train jointly in minimax game
- Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

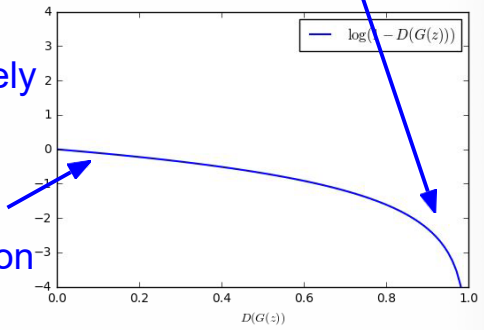
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

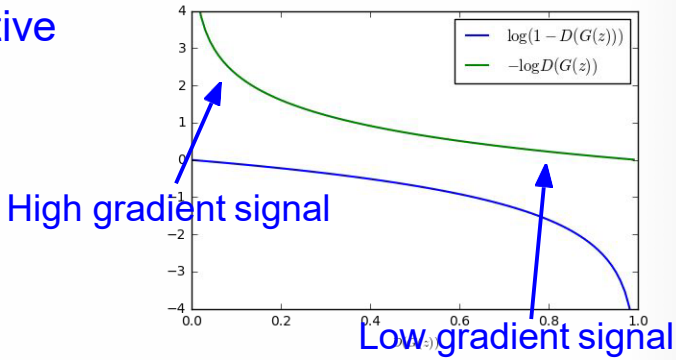
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

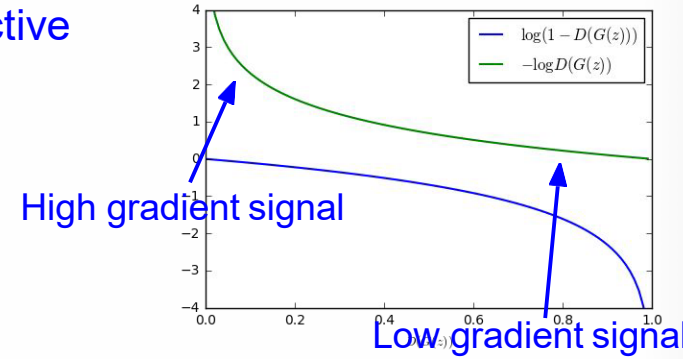
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.



Training GANs: Two-player game

Putting it together: GAN training algorithm

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Training GANs: Two-player game

Putting it together: GAN training algorithm

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

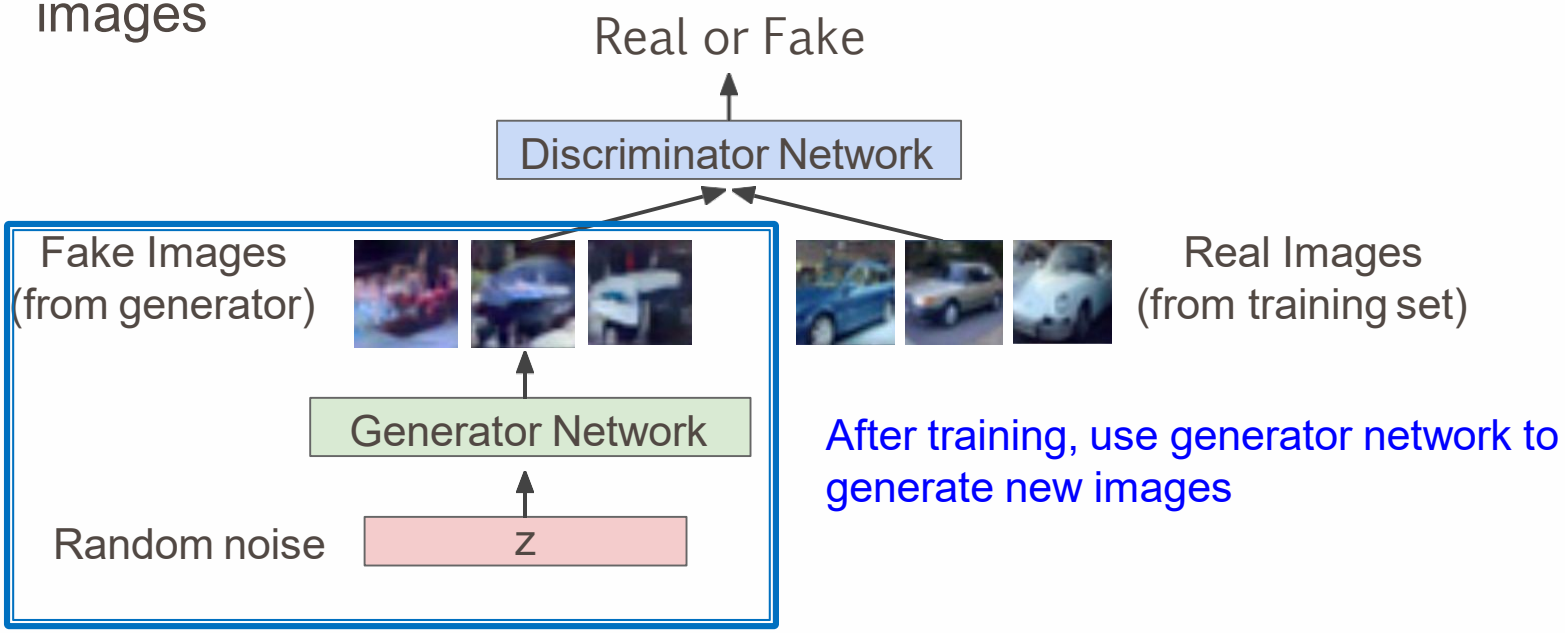
end for

Some find k=1 more stable, others use k > 1, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

Problems in GANs

- No guarantee to equilibrium
  - Mode collapsing
    - All smoothing results
  - Oscillation
    - May never converge
  - No indicator when to finish
- All generative models
  - Evaluation metrics (predefined)
  - Robust but difficult to train
  - Diversity testing is required

GAN's Ways

- Theatrical analysis of the nature of the GANs
  - WGAN
    - Wasserstein GAN (Replace KL with Wasserstein)
    - Solved the issue when there is no overlapping between distributions of generated & ground truth samples
  - BEGAN
  - WGAN-GP
  - RAGAN
  - …etc
- Applications
  - Based on a state-of-the-art GAN and fine-tune it.

Improved GAN: DCGAN

- Radford et al, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015
  - Tricks for gradient flow
    - Max pooling → Strided convolution or average pooling
    - Use LeakyReLU instead of ReLU
  - Other tricks
    - Use batch normal both generator and discriminator
    - Use Adam optimizer (lr = 0.0002, a = 0.9, b=0.5)

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

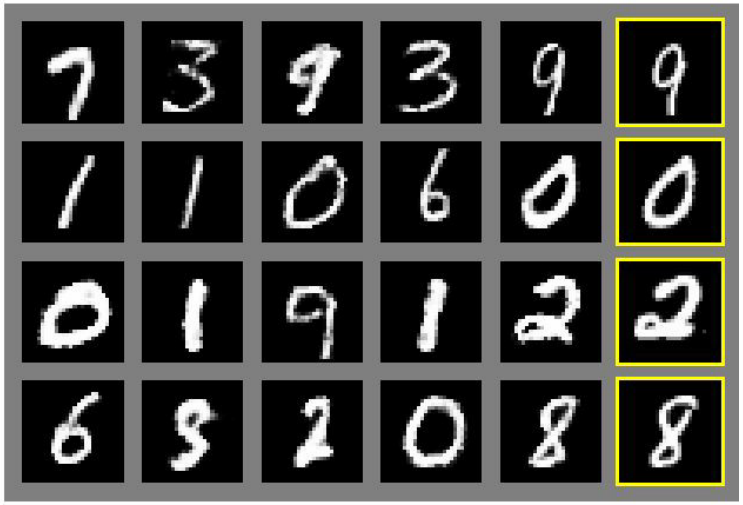
DCGAN

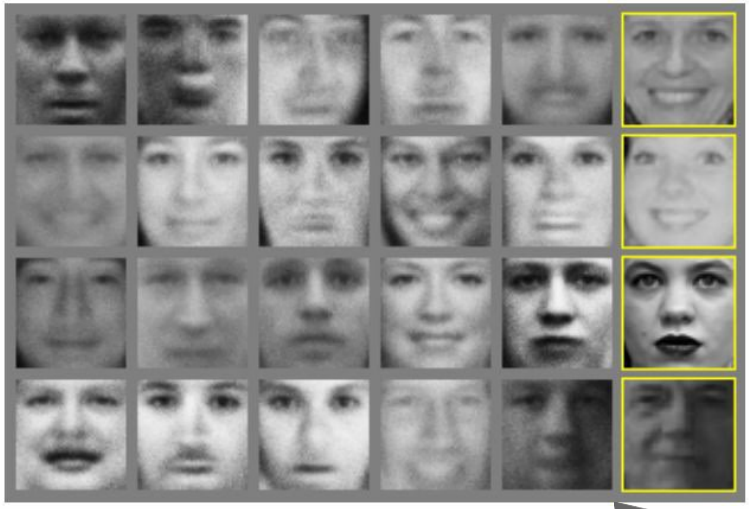
- Convert max-pooling layers to convolution layers
- Convert fully connected layers to global average pooling layers in the discriminator
- Use batch normalization layers in the generator and the discriminator
- Use leaky-ReLU activation functions in the discriminator
- Other tricks
  - Use Adam optimizer (lr = 0.0002, a = 0.9, b=0.5)

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

Generative Adversarial Nets

Generated samples

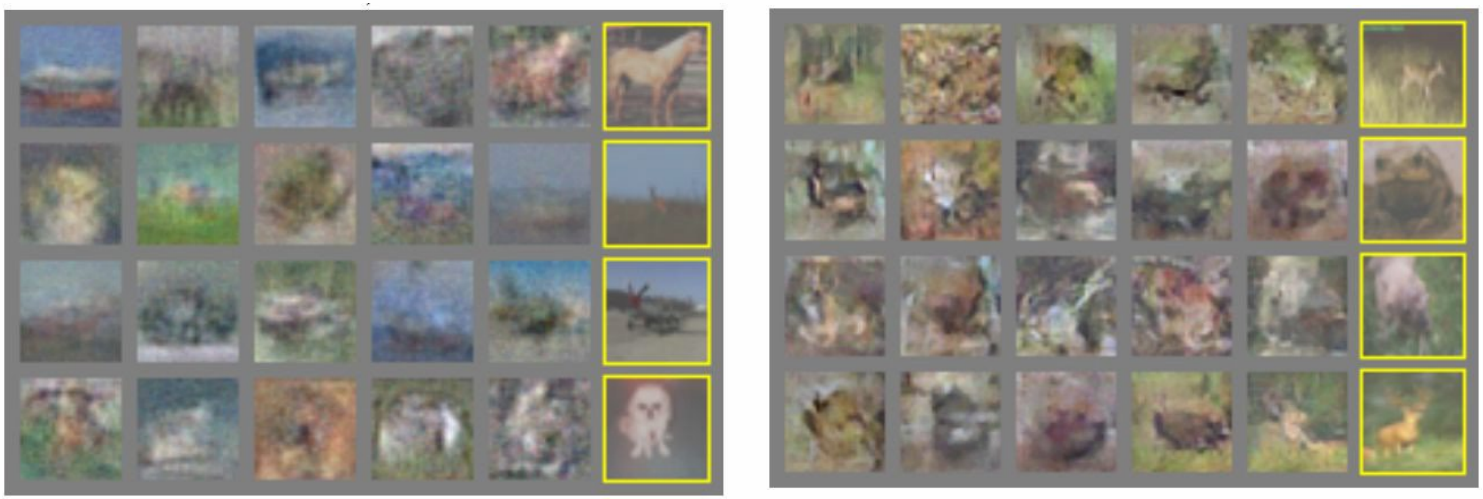




Nearest neighbor from training set

Generative Adversarial Nets

Generated samples (CIFAR-10)



Nearest neighbor from training set

Generative Adversarial Nets: Convolutional Architectures

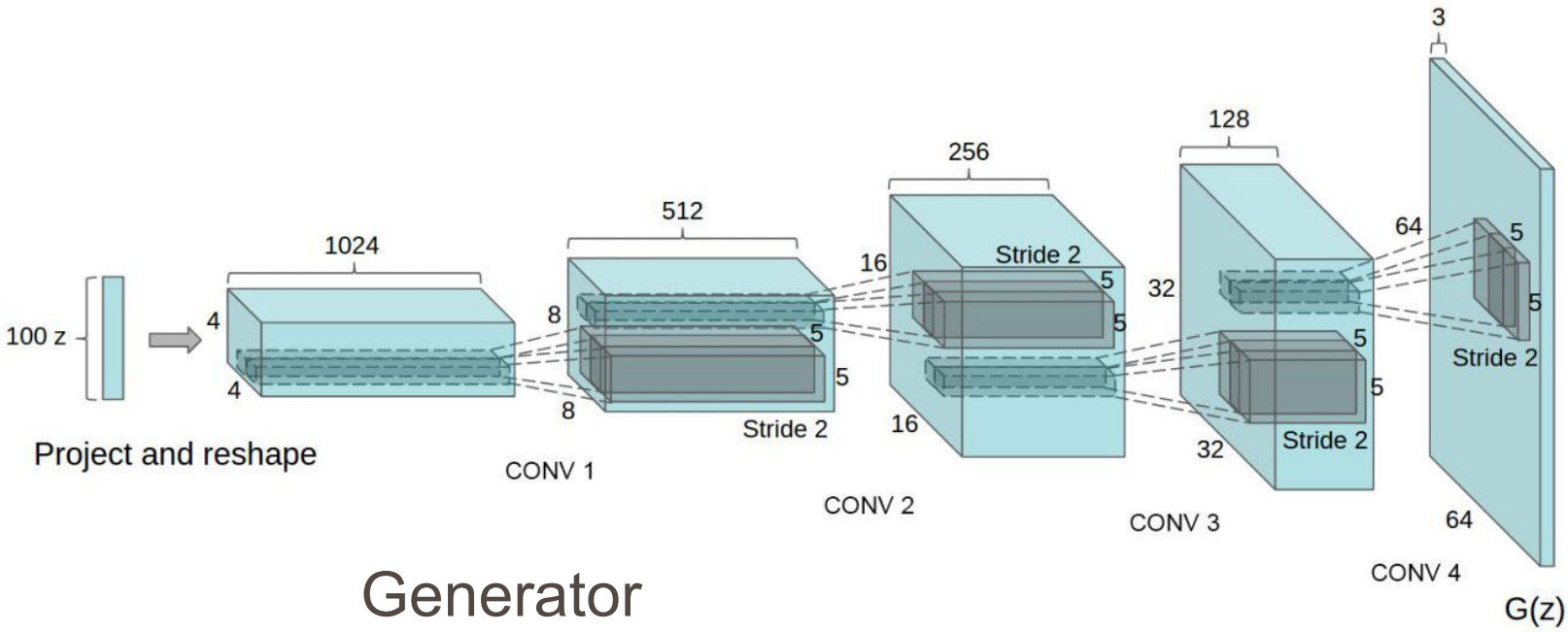
Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

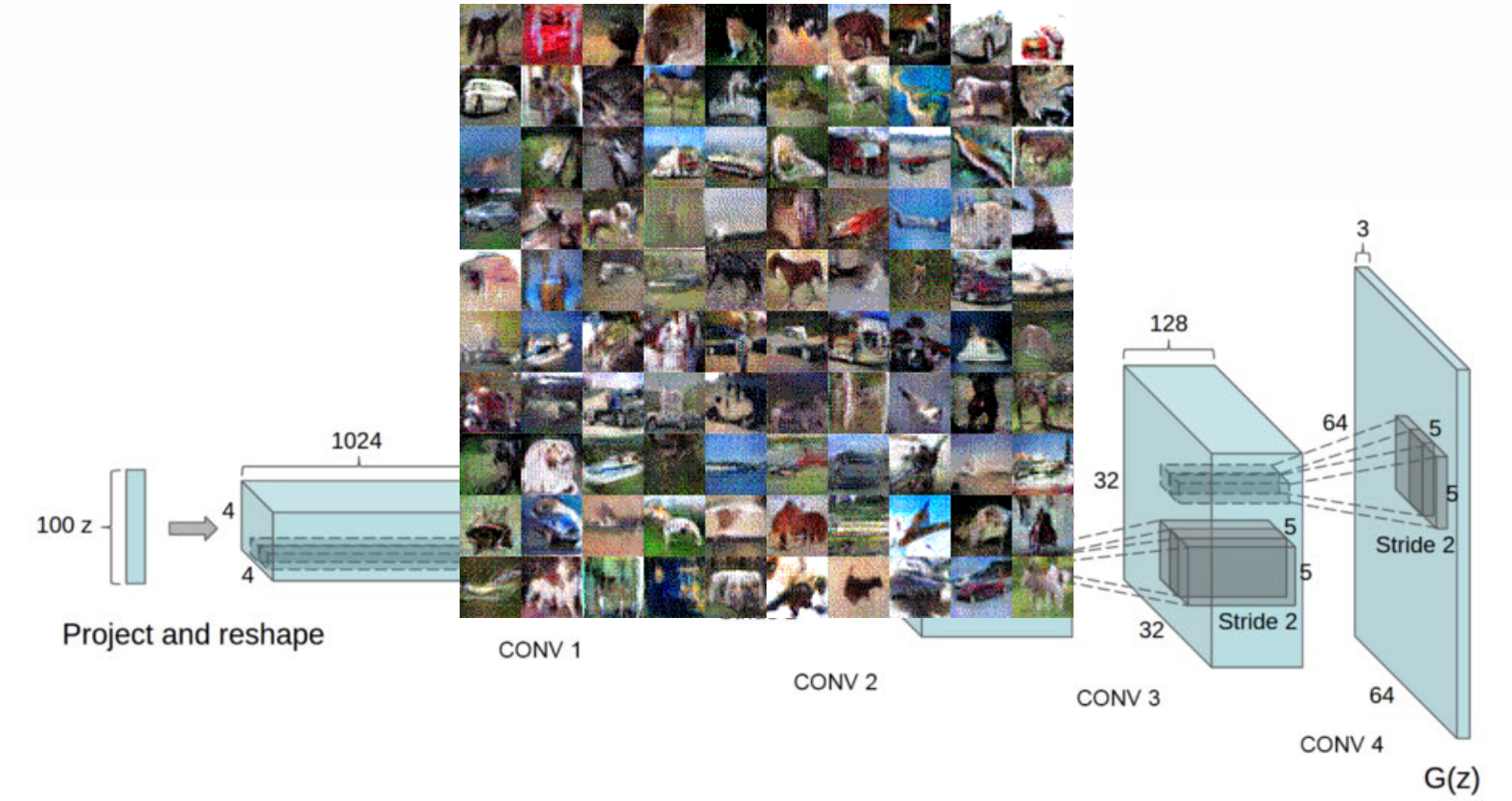
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures



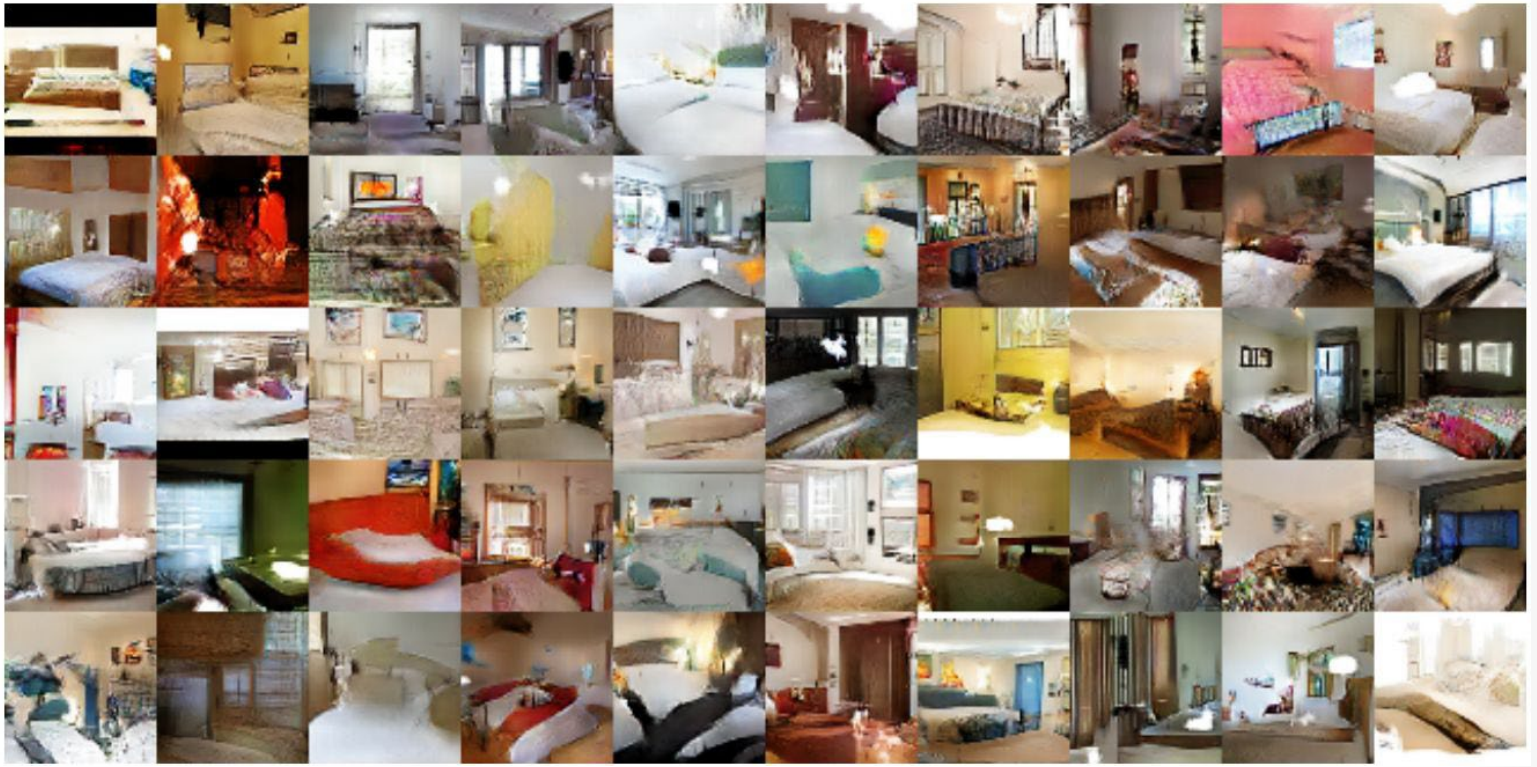
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN: Generate the images with Deep Convolutional GAN



Generative Adversarial Nets: Convolutional Architectures

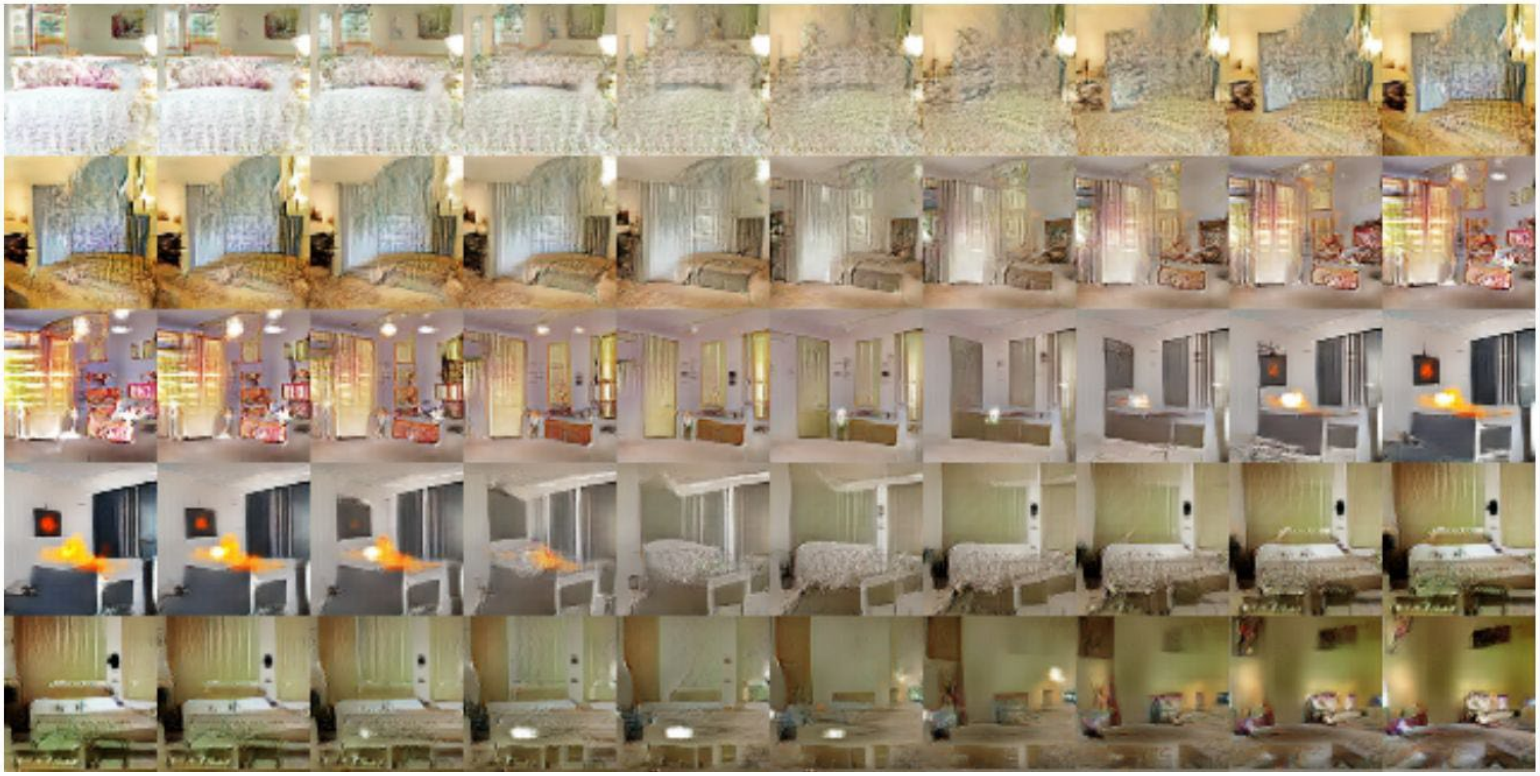
Samples from the model look much better!



Radford et al, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

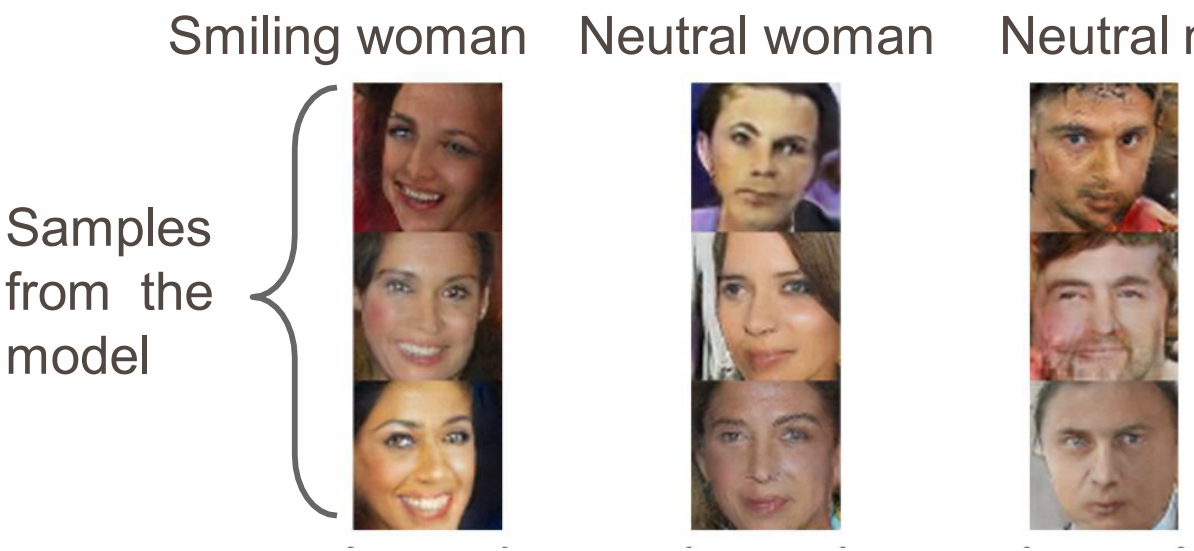
Interpolating between random points in latent space



Radford et al, ICLR 2016

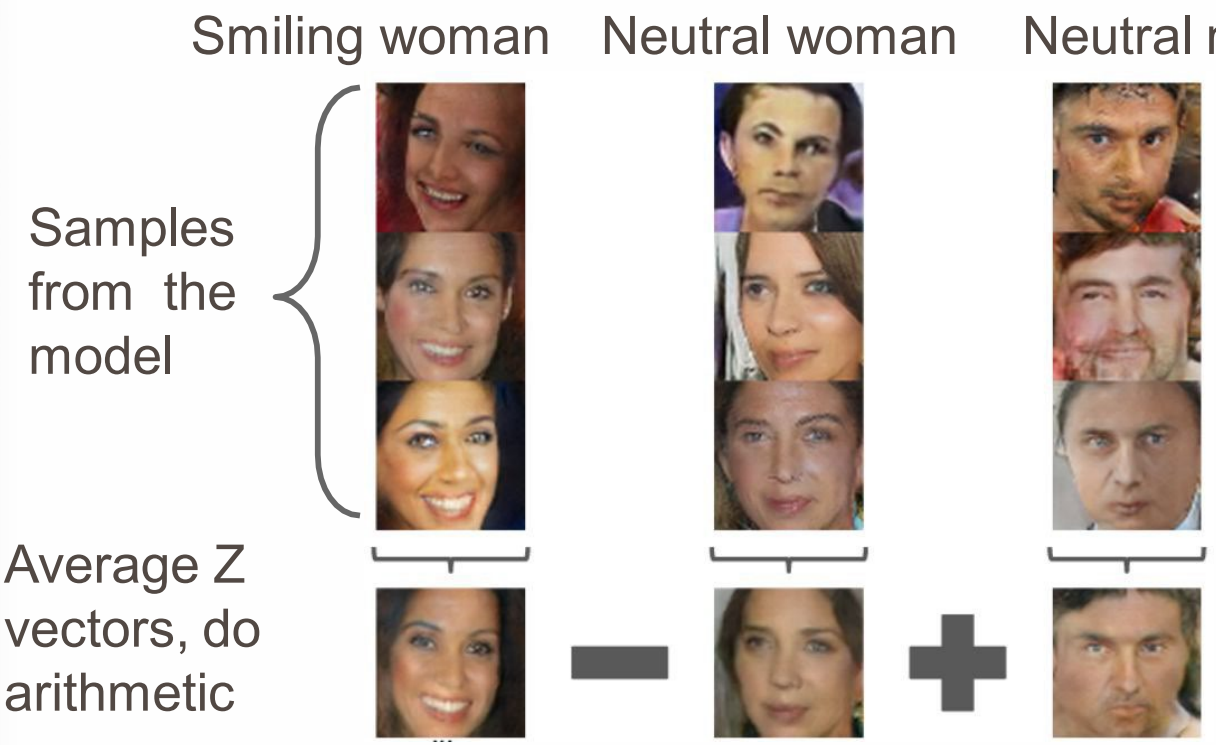
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



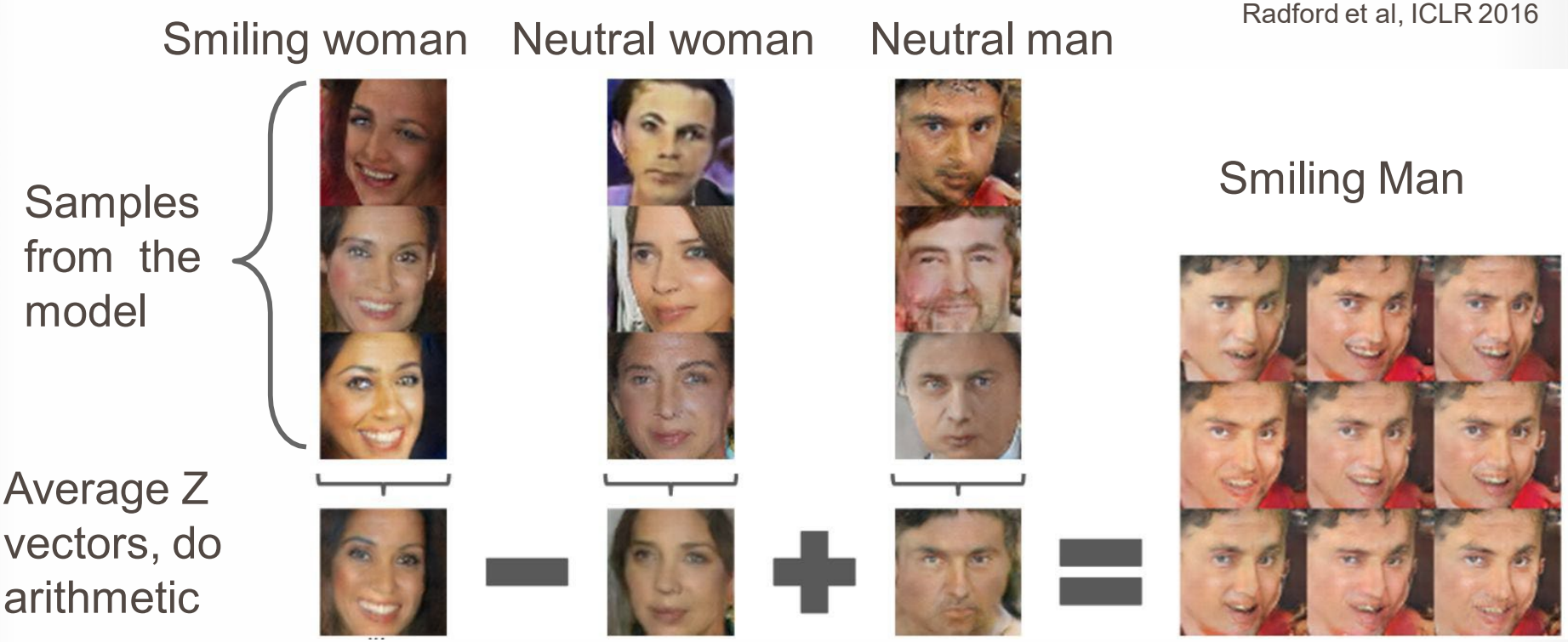
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



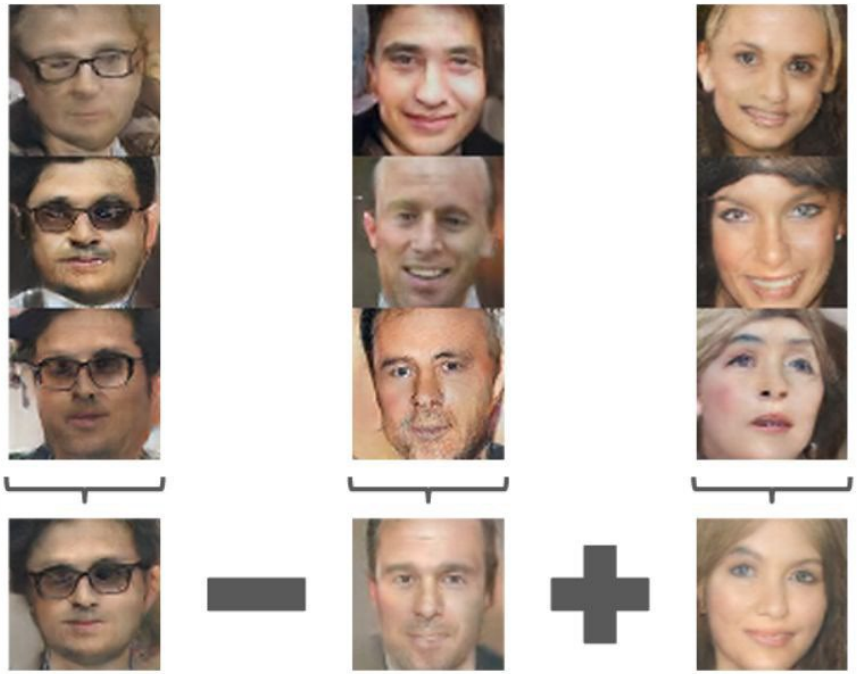
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



Generative Adversarial Nets: Interpretable Vector Math

Glasses man — No glasses man + No glasses woman



Radford et al, ICLR 2016

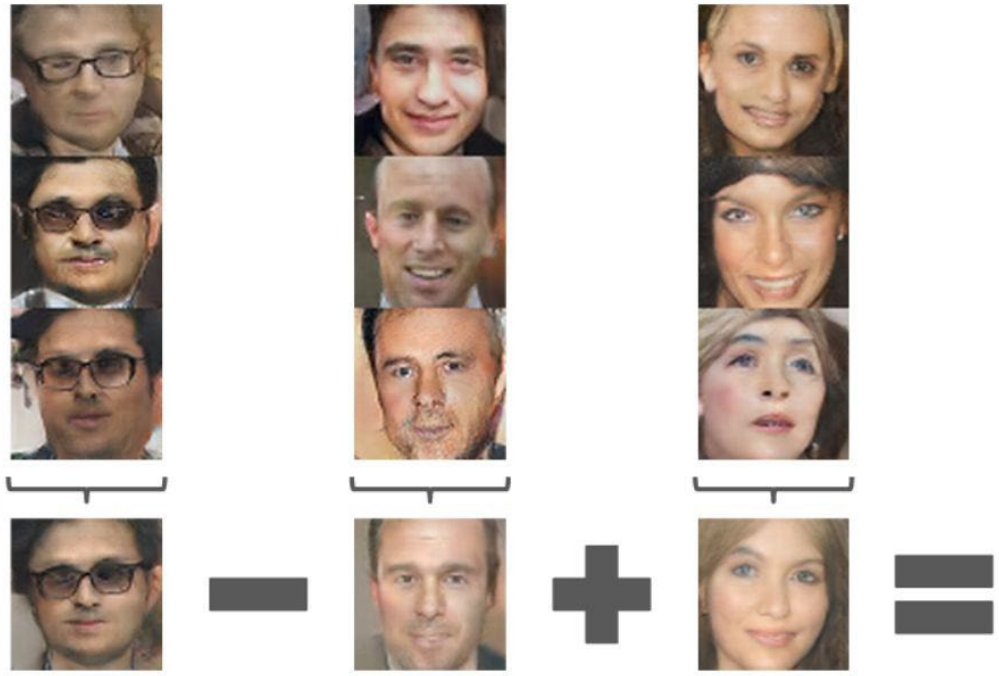
Generative Adversarial Nets: Interpretable Vector Math

Glasses man

No glasses man

No glasses woman

Radford et al, ICLR 2016



Woman with glasses

https://github.com/soumith/ganhacks

and tricks for trainings GANs

“The GAN Zoo”

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
acGAN - Face Aging With Conditional Generative Adversarial Networks
ACGAN - Coverless Information Hiding Based on Generative adversarial networks
acGAN - On-line Adaptative Curriculum Learning for GANs
ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
AdaGAN - AdaGAN: Boosting Generative Models
Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
AdvGAN - Generating adversarial examples with adversarial networks
AE-GAN - AE-GAN: adversarial eliminating with GAN
AE-OT - Latent Space Optimal Transport for Generative Models
AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
AffGAN - Amortised MAP Inference for Image Super-resolution
AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization
AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
ALI - Adversarially Learned Inference (github)
AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
AlphaGAN - AlphaGAN: Generative adversarial networks for natural image matting
AM-GAN - Activation Maximization Generative Adversarial Nets
AmbientGAN - AmbientGAN: Generative models from lossy measurements (github)
AMC-GAN - Video Prediction with Appearance and Motion Conditions
AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
APD - Adversarial Distillation of Bayesian Neural Network Posteriors
APE-GAN - APE-GAN: Adversarial Perturbation Elimination with GAN
ARAE - Adversarially Regularized Autoencoders for Generating Discrete Structures (github)
ARDA - Adversarial Representation Learning for Domain Adaptation
ARIGAN - ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network
ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
ASDL-GAN - Automatic Steganographic Distortion Learning Using a Generative Adversarial Network
ATA-GAN - Attention-Aware Generative Adversarial Networks (ATA-GANs)
Attention-GAN - Attention-GAN for Object Transfiguration in Wild Images
AttGAN - Arbitrary Facial Attribute Editing: Only Change What You Want (github)
AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks (github)
AVID - AVID: Adversarial Visual Irregularity Detection
B-DCGAN - B-DCGAN:Evaluation of Binarized DCGAN for FPGA
B-GAN - Generative Adversarial Nets from a Density Ratio Estimation Perspective
BAGAN - BAGAN: Data Augmentation with Balancing GAN
Bayesian GAN - Deep and Hierarchical Implicit Models
Bayesian GAN - Bayesian GAN (github)
BCGAN - Bayesian Conditional Generative Adverserial Networks
BCGAN - Bidirectional Conditional Generative Adversarial networks
BEAM - Boltzmann Encoded Adversarial Machines
BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
BEGAN-CS - Escaping from Collapsing Modes in a Constrained Space
Bellman GAN - Distributional Multivariate Policy Evaluation and Exploration with the Bellman GAN
BGAN - Binary Generative Adversarial Networks for Image Retrieval (github)
Bi-GAN - Autonomously and Simultaneously Refining Deep Neural Network Parameters by a Bi-Generative Adversarial Network Aided Genetic Algorithm
BicycleGAN - Toward Multimodal Image-to-Image Translation (github)
BiGAN - Adversarial Feature Learning
BinGAN - BinGAN: Learning Compact Binary Descriptors with a Regularized GAN
BourGAN - BourGAN: Generative Networks with Metric Embeddings
BranchGAN - Branched Generative Adversarial Networks for Multi-Scale Image Manifold Learning
BRE - Improving GAN Training via Binarized Representation Entropy (BRE) Regularization (github)
BridgeGAN - Generative Adversarial Frontal View to Bird View Synthesis
BS-GAN - Boundary-Seeking Generative Adversarial Networks
BubGAN - BubGAN: Bubble Generative Adversarial Networks for Synthesizing Realistic Bubbly Flow Images
BWGAN - Banach Wasserstein GAN
C-GAN - Face Aging with Contextual Generative Adversarial Nets
C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training (github)
CA-GAN - Composition-aided Sketch-realistic Portrait Generation
CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks (github)
CAN - CAN: Creative Adversarial Networks, Generating Art by Learning About Styles and Deviating from Style Norms
CapsGAN - CapsGAN: Using Dynamic Routing for Generative Adversarial Networks
CapsuleGAN - CapsuleGAN: Generative Adversarial Capsule Network
CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
CatGAN - CatGAN: Coupled Adversarial Transfer for Domain Generation
CausalGAN - CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training
CC-GAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks (github)
CC-GAN - Conditional Image-to-Image Translation
CDcGAN - Simultaneously Color-Depth Super-Resolution with Conditional Generative Adversarial Network
CE-GAN - Deep Learning for Imbalance Data Classification using Class Expert Generative Adversarial Network
CFG-GAN - Composite Functional Gradient Learning of Generative Adversarial Models
CGAN - Conditional Generative Adversarial Nets
CGAN - Controllable Generative Adversarial Network
Chekhov GAN - An Online Learning Approach to Generative Adversarial Networks
ciGAN - Conditional Infilling GANs for Data Augmentation in Mammogram Classification
CinCGAN - Unsupervised Image Super-Resolution using Cycle-in-Cycle Generative Adversarial Networks
CipherGAN - Unsupervised Cipher Cracking Using Discrete GANs
ClusterGAN - ClusterGAN : Latent Space Clustering in Generative Adversarial Networks
CM-GAN - CM-GANs: Cross-modal Generative Adversarial Networks for Common Representation Learning
CoAtt-GAN - Are You Talking to Me? Reasoned Visual Dialog Generation through Adversarial Learning
CoGAN - Coupled Generative Adversarial Networks
ComboGAN - ComboGAN: Unrestrained Scalability for Image Domain Translation (github)
ConceptGAN - Learning Compositional Visual Concepts with Mutual Consistency
Conditional cycleGAN - Conditional CycleGAN for Attribute Guided Face Image Generation
constrast-GAN - Generative Semantic Manipulation with Contrasting GAN
Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
CorrGAN - Correlated discrete data generation using adversarial training
Coulomb GAN - Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields
Cover-GAN - Generative Steganography with Kerckhoffs' Principle based on Generative Adversarial Networks
cowboy - Defending Against Adversarial Attacks by Leveraging an Entire GAN
CR-GAN - CR-GAN: Learning Complete Representations for Multi-view Generation
Cramer GAN - The Cramer Distance as a Solution to Biased Wasserstein Gradients
Cross-GAN - Crossing Generative Adversarial Networks for Cross-View Person Re-identification
crVAE-GAN - Channel-Recurrent Variational Autoencoders
CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
CSG - Speech-Driven Expressive Talking Lips with Conditional Sequential Generative Adversarial Networks
CT-GAN - CT-GAN: Conditional Transformation Generative Adversarial Network for Image Attribute Modification
CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (github)
D-GAN - Differential Generative Adversarial Networks: Synthesizing Non-linear Facial Variations with Limited Number of Training Data
D-WCGAN - I-vector Transformation Using Conditional Generative Adversarial Networks for Short Utterance Speaker Verification
D2GAN - Dual Discriminator Generative Adversarial Nets
D2IA-GAN - Tagging like Humans: Diverse and Distinct Image Annotation
DA-GAN - DA-GAN: Instance-level Image Translation by Deep Attention Generative Adversarial Networks (with Supplementary Materials)
DADA - DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification
DAGAN - Data Augmentation Generative Adversarial Networks
DAN - Distributional Adversarial Networks
DBLRGAN - Adversarial Spatio-Temporal Learning for Video Deblurring
DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (github)
DE-GAN - Generative Adversarial Networks with Decoder-Encoder Output Noise
DeblurGAN - DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks (github)
DeepFD - Learning to Detect Fake Face Images in the Wild
Defense-GAN - Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models (github)
Defo-Net - Defo-Net: Learning Body Deformation using Generative Adversarial Networks
DeliGAN - DeLiGAN : Generative Adversarial Networks for Diverse and Limited Data (github)
DF-GAN - Learning Disentangling and Fusing Networks for Face Completion Under Structured Occlusions
DialogWAE - DialogWAE: Multimodal Response Generation with Conditional Wasserstein Auto-Encoder
DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
DistanceGAN - One-Sided Unsupervised Domain Mapping
DM-GAN - Dual Motion GAN for Future-Flow Embedded Video Prediction
DMGAN - Disconnected Manifold Learning for Generative Adversarial Networks
DNA-GAN - DNA-GAN: Learning Disentangled Representations from Multi-Attribute Images
DOPING - DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN
dp-GAN - Differentially Private Releasing via Deep Generative Model
DP-GAN - DP-GAN: Diversity-Promoting Generative Adversarial Network for Generating Informative and Diversified Text
DPGAN - Differentially Private Generative Adversarial Network
DR-GAN - Representation Learning by Rotating Your Faces
DRAGAN - How to Train Your DRAGAN (github)
Dropout-GAN - Dropout-GAN: Learning from a Dynamic Ensemble of Discriminators
DRPAN - Discriminative Region Proposal Adversarial Networks for High-Quality Image-to-Image Translation
DSH-GAN - Deep Semantic Hashing with Generative Adversarial Networks
DSP-GAN - Depth Structure Preserving Scene Image Generation
DTLC-GAN - Generative Adversarial Image Synthesis with Decision Tree Latent Controller
DTN - Unsupervised Cross-Domain Image Generation
DTR-GAN - DTR-GAN: Dilated Temporal Relational Adversarial Network for Video Summarization
DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
Dualing GAN - Dualing GANs
DVGAN - Human Motion Modeling using DVGANs
Dynamics Transfer GAN - Dynamics Transfer GAN: Generating Video by Transferring Arbitrary Temporal Dynamics from a Source Video to a Single Target Image
E-GAN - Evolutionary Generative Adversarial Networks
EAR - Generative Model for Heterogeneous Inference
EBGAN - Energy-based Generative Adversarial Network
ecGAN - eCommerceGAN : A Generative Adversarial Network for E-commerce
ED//GAN - Stabilizing Training of Generative Adversarial Networks through Regularization
Editable GAN - Editable Generative Adversarial Networks: Generating and Editing Faces Simultaneously
EGAN - Enhanced Experience Replay Generation for Efficient Reinforcement Learning
EL-GAN - EL-GAN: Embedding Loss Driven Generative Adversarial Networks for Lane Detection
ELEGANT - ELEGANT: Exchanging Latent Encodings with GAN for Transferring Multiple Face Attributes
EnergyWGAN - Energy-relaxed Wassertein GANs (EnergyWGAN): Towards More Stable and High Resolution Image Generation
ESRGAN - ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks
ExGAN - Eye In-Painting with Exemplar Generative Adversarial Networks
ExposureGAN - Exposure: A White-Box Photo Post-Processing Framework (github)
ExprGAN - ExprGAN: Facial Expression Editing with Controllable Expression Intensity
f-CLSWGAN - Feature Generating Networks for Zero-Shot Learning
f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
FairGAN - FairGAN: Fairness-aware Generative Adversarial Networks
Fairness GAN - Fairness GAN
FakeGAN - Detecting Deceptive Reviews using Generative Adversarial Networks
FBGAN - Feedback GAN (FBGAN) for DNA: a Novel Feedback-Loop Architecture for Optimizing Protein Functions
FBGAN - Featurized Bidirectional GAN: Adversarial Defense via Adversarially Learned Semantic Inference
FC-GAN - Fast-converging Conditional Generative Adversarial Networks for Image Synthesis
FF-GAN - Towards Large-Pose Face Frontalization in the Wild
FGGAN - Adversarial Learning for Fine-grained Image Search
Fictitious GAN - Fictitious GAN: Training GANs with Historical Models
FIGAN - Frame Interpolation with Multi-Scale Deep Loss Functions and Generative Adversarial Networks
Fila-GAN - Synthesizing Filamentary Structured Images with GANs
First Order GAN - First Order Generative Adversarial Networks (github)
Fisher GAN - Fisher GAN
Flow-GAN - Flow-GAN: Bridging implicit and prescribed learning in generative models
FrankenGAN - FrankenGAN: Guided Detail Synthesis for Building Mass-Models Using Style-Synchonized GANs

2017: Explosion of GANs

Better training and generation

LSGAN, Zhu 2017.

Wasserstein GAN, Arjovsky 2017. Improved Wasserstein GAN, Gulrajani 2017.

Progressive GAN, Karras 2018.

Improved Versions of GAN

- There are more than 100 improved GANs/Applications since 2014!!
  - A hot topic in deep learning

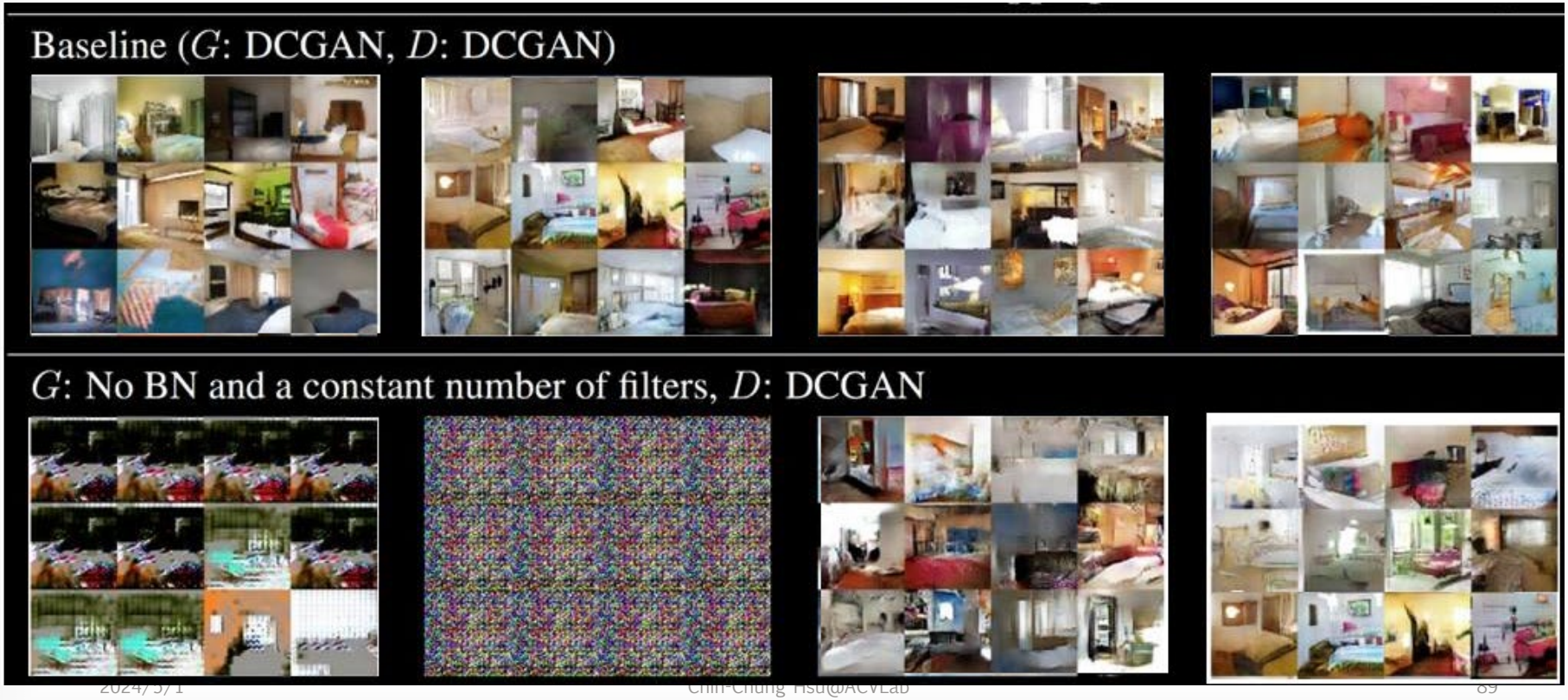
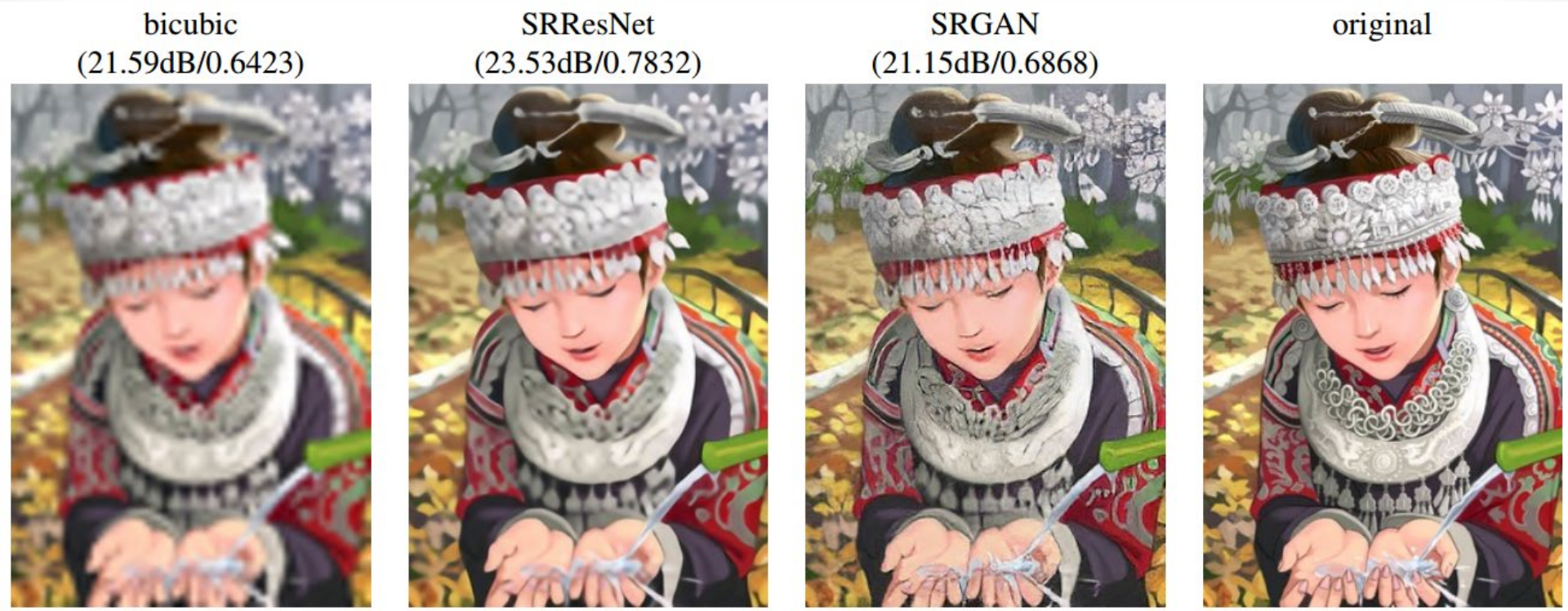
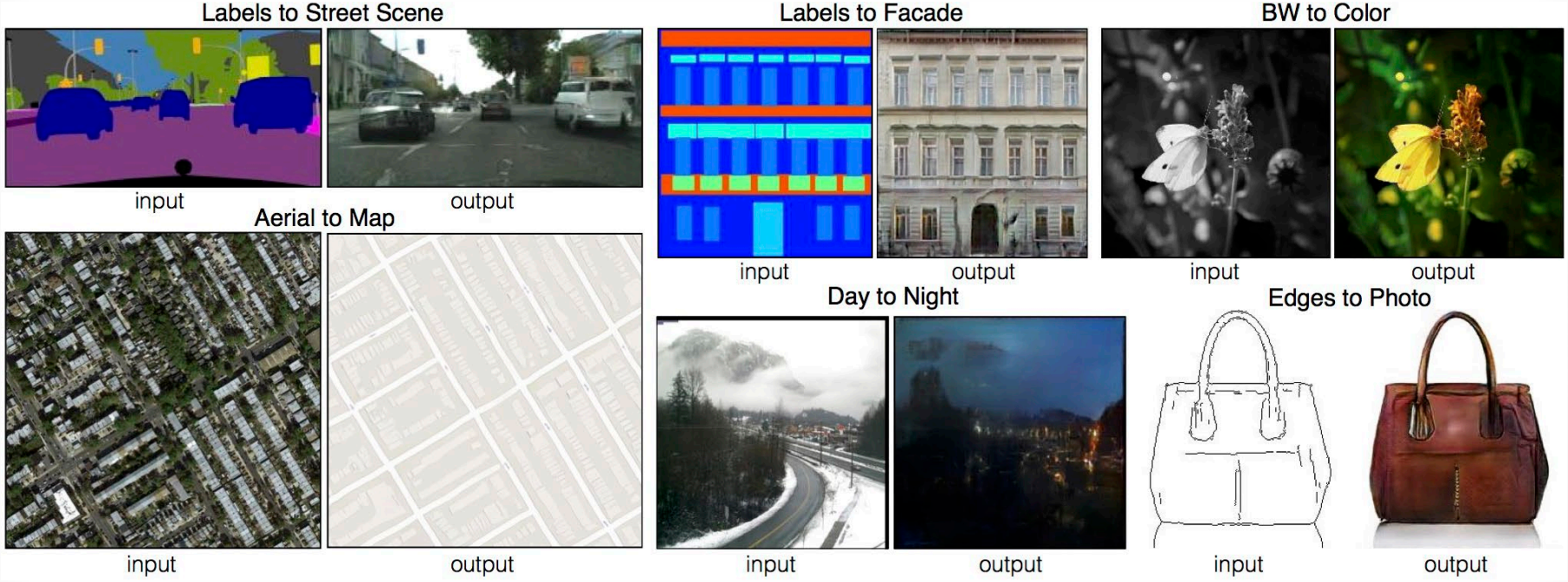


Image Super-Resolution



Ledig et al. CVPR 2017

Image-to-Image Translation



Isola et al. CVPR 2017

Text2Image

this small bird has a pink breast and crown, and black primaries and secondaries.

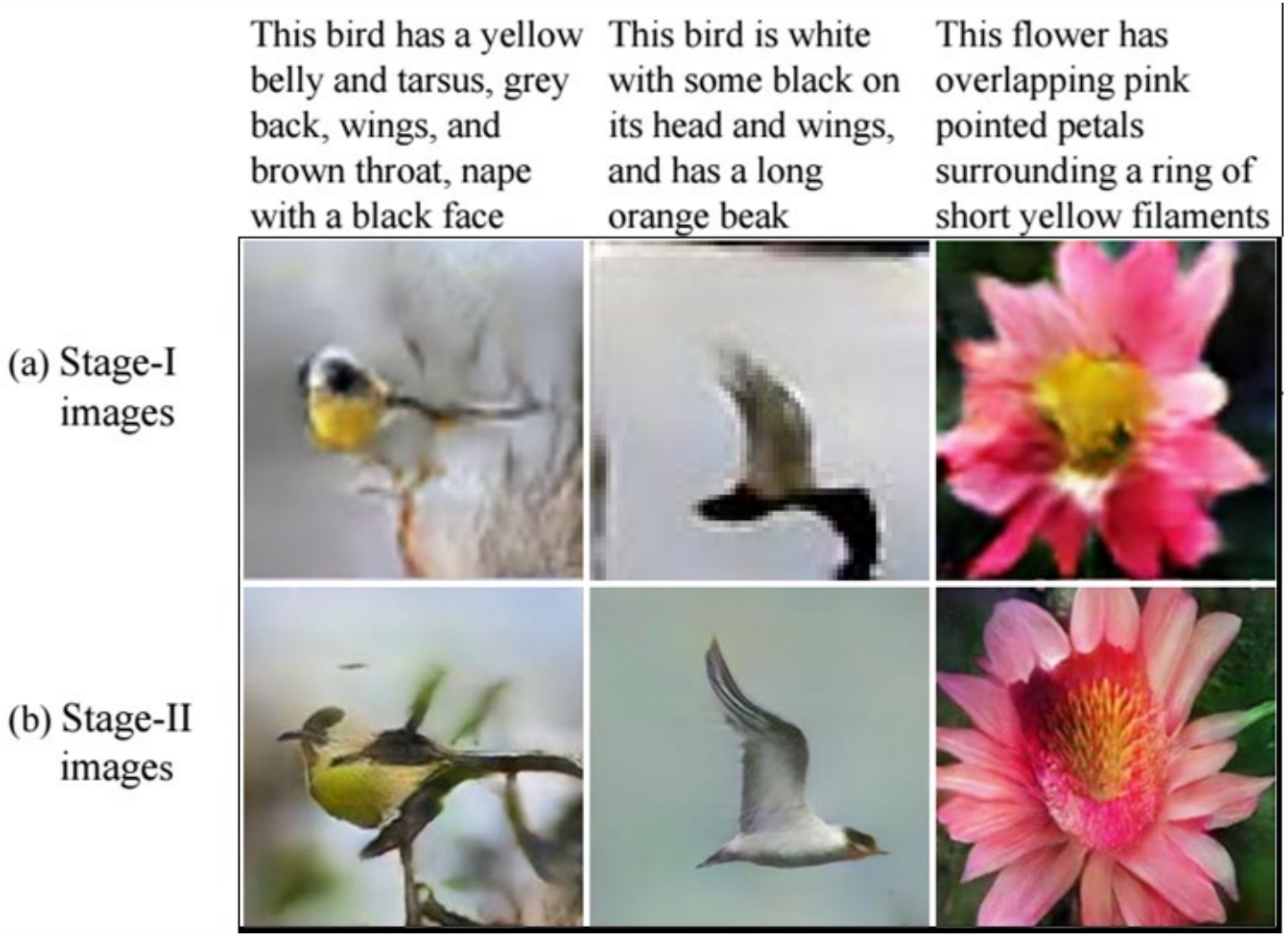
this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

Reed et al. ICML 2016

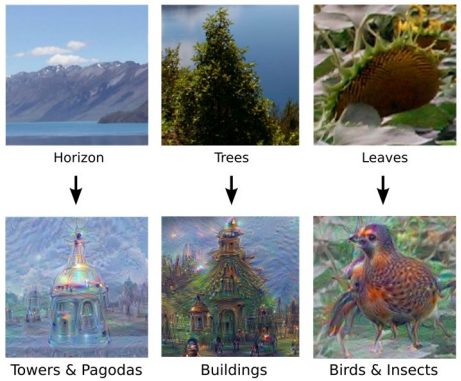
StackGAN



Zhang et al. 2016

Style Transfer

Style Transfer



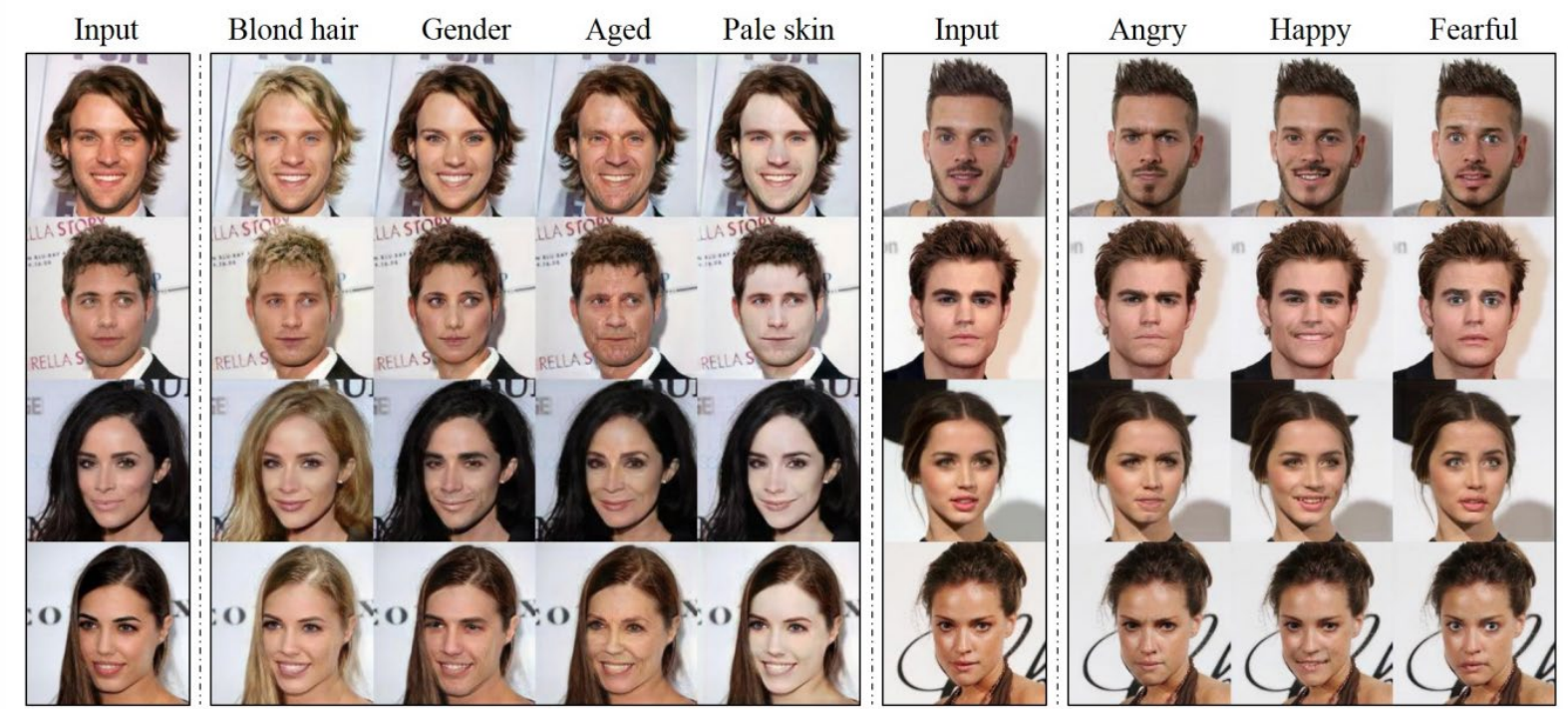
Pose Generation

Ground truth

Generated

Source: https://papers.nips.cc/paper/6644-pose-guided-person-image-generation.pdf

StarGAN



Source: https://arxiv.org/pdf/1711.09020.pdf

Change the Cloth

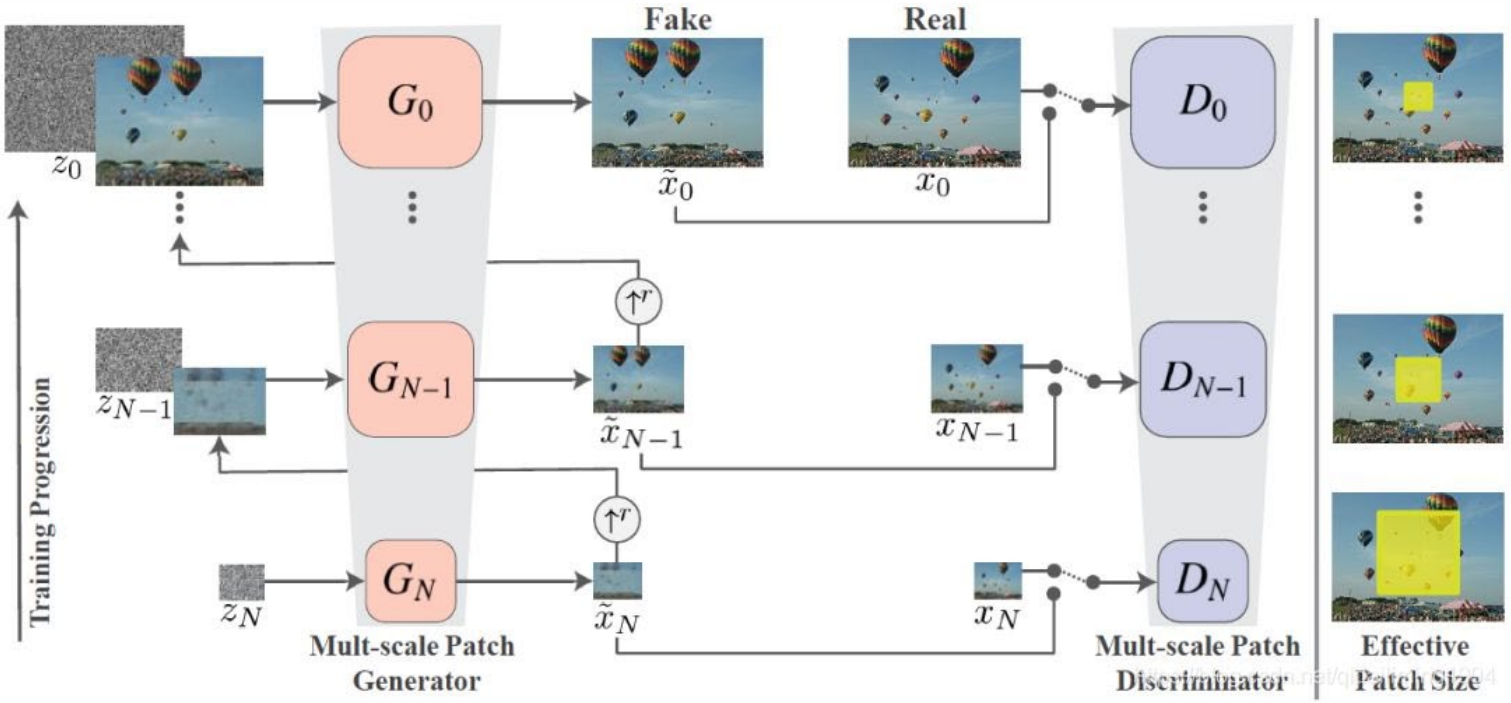
Example results on LOOKBOOK dataset(top), left is input, right is generated clothes. Results on a similar dataset (bottom).
More results will be added soon.

Source: https://github.com/fxia22/PixelDTGAN

2019: BigGAN

Brock et al., 2019

SinGAN, ICCV 2019.

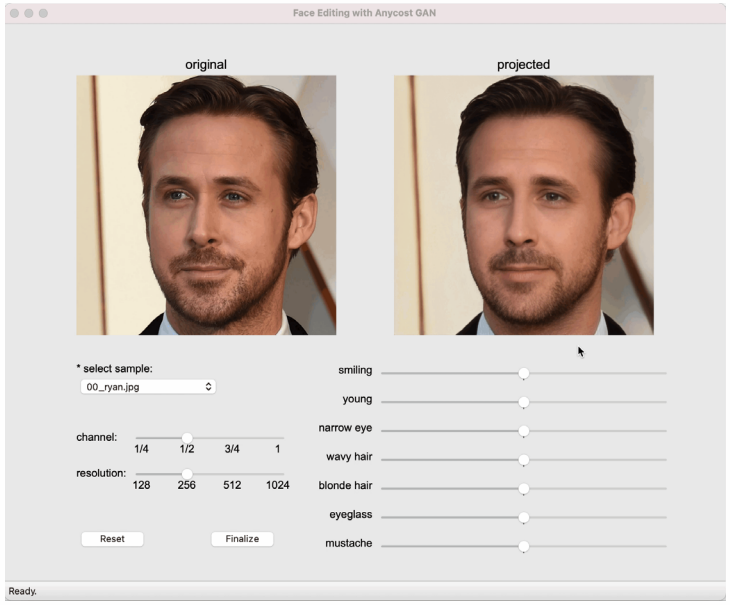


Shaham, Tamar Rott, Tali Dekel, and Tomer Michaeli. "Singan: Learning a generative model from a single natural image." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

DeepFake for Faces

- StyleGANv2 [CVPR2020]

- **Anycost GAN [CVPR2021]**

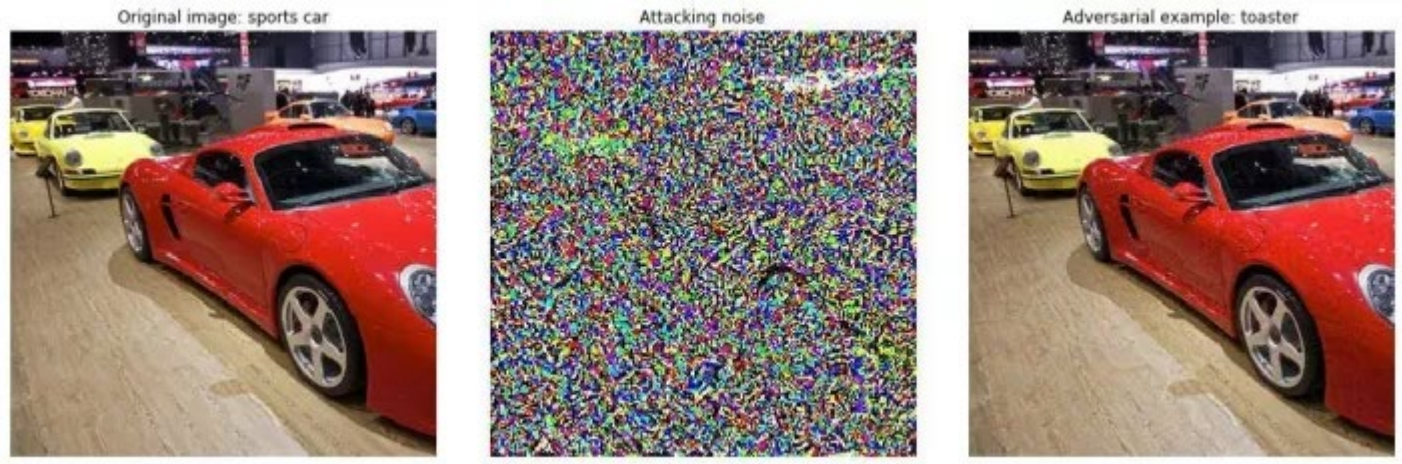


GAN-based Applications

- What application should GAN be used
  - Any task related to image synthesis (合成影像任務包含)
    - Image super-resolution
      - Discriminator can be used to judge its fidelity and resolution
    - Image translation
      - Discriminator can be used to identify its quality
    - Image segmentation
      - Discriminator can be used to tune generated segmentation map
    - Data argumentation
      - Discriminator can be used to check the fidelity of the simulated image
    - etc...

Rethinking GANs

- Is possible to fool a DNN by adding specified noises?
  - Adversarial attack



(a) Car (b) Noise map (c) Toaster

Adversarial Attack

- Cause model uninterpretable
  - Goodfellow given some reasons
    - Assume that a linear operation in NN is $\mathbf{z} = \mathbf{w}(\mathbf{x} + \mathbf{b})$
    - Consider that
      - $\mathbf{z} = \mathbf{w}(\mathbf{x} + \mathbf{b}) * \mathbf{d}$, where d is the attach signal
      - When $\mathbf{wd}$ is large
        - Attacking will be very effective
  - How to find $\mathbf{d}$?
    - Yes, we can train it by supervised way
    - Similar to “Discriminator” does

GANs

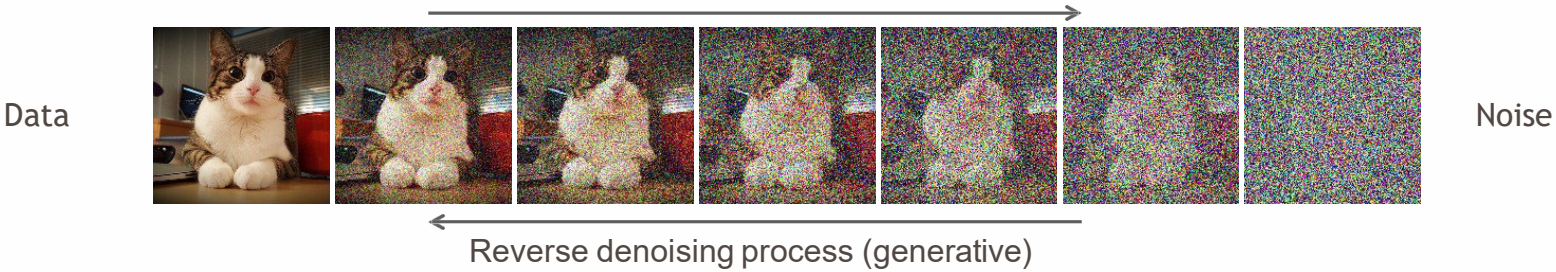
- Don’t work with an explicit density function
  - Take game-theoretic approach: learn to generate from training distribution through 2-player game
- Pros:
  - Beautiful, state-of-the-art samples!
- Cons:
  - Trickier / more unstable to train
  - Can’t solve inference queries such as p(x), p(z|x)
- Active areas of research:
  - Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
  - Conditional GANs, GANs for all kinds of applications

DENOISED DIFFUSION PROBABILISTIC MODELS

NPIS 2021

Denoising Diffusion Models

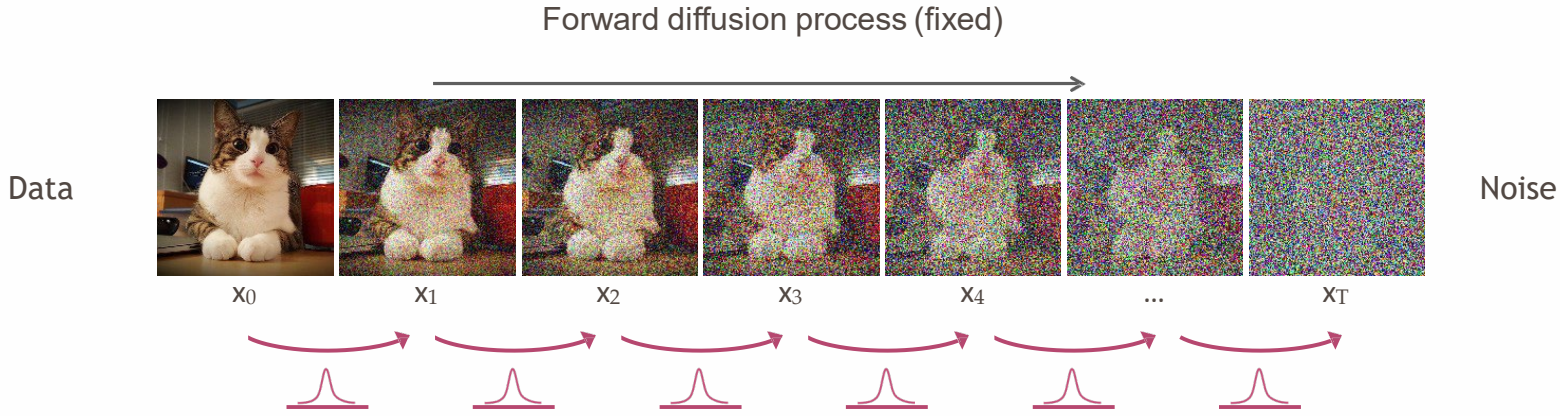
- Learning to generate by denoising
  - Denoising diffusion models consist of two processes:
  - Forward diffusion process that gradually adds noise to input
  - Reverse denoising process that learns to generate data by denoising
  - Forward diffusion process (fixed)



Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015 Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Forward Diffusion Process

- The formal definition of the forward process in T steps:



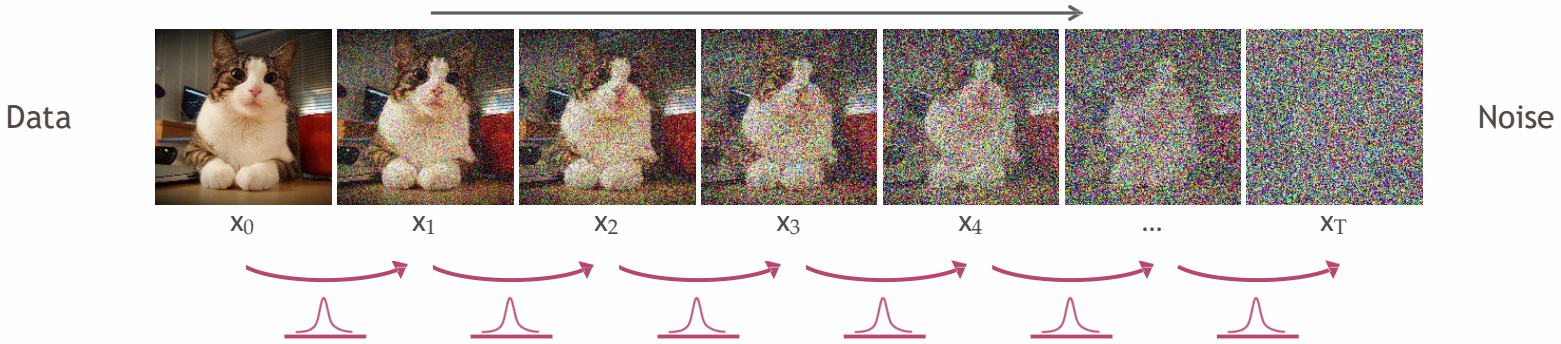
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

mean variance

➡ Sample:

$$\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$$

Diffusion Kernel



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

mean

variance

➡ Sample:

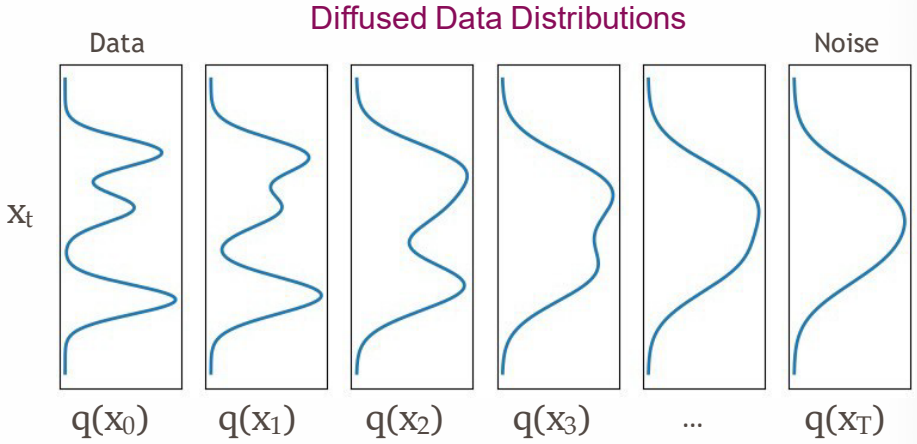
$$\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$$

What happens to a distribution in the forward diffusion?

- So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



- We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

Generative Learning by Denoising

- Recall, that the diffusion parameters are designed such that

$$q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$$

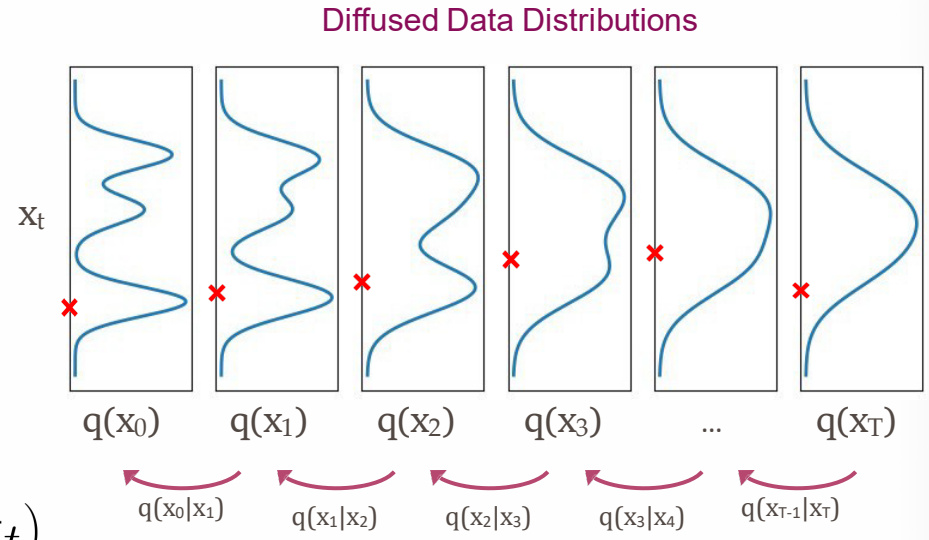
- Generation:
  - Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$
- Iteratively sample

$$\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$$

- In general,

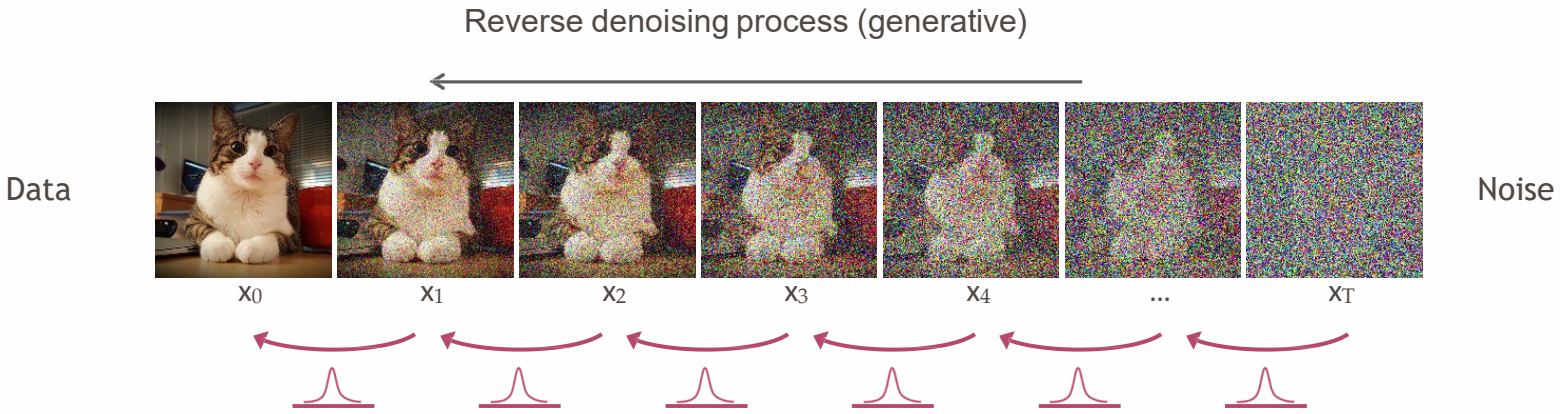
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

  - is intractable.
- Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$
  - Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.



Reverse Denoising Process

- Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

Autoencoder predicts the mean of the denoised image x(t-1) given x(t).

How do we train? (summary version)

What is the loss function? (Ho et al. NeurIPS 2020)

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\mathbf{x}_t}, t)||^2 \right]$$

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

U-Net autoencoder takes x(t) as input and simply predict a noise. The goal of the training is to generate a noise pattern that is unit normal. Very similar to VAE, right?

Summary

- Training and Sample Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

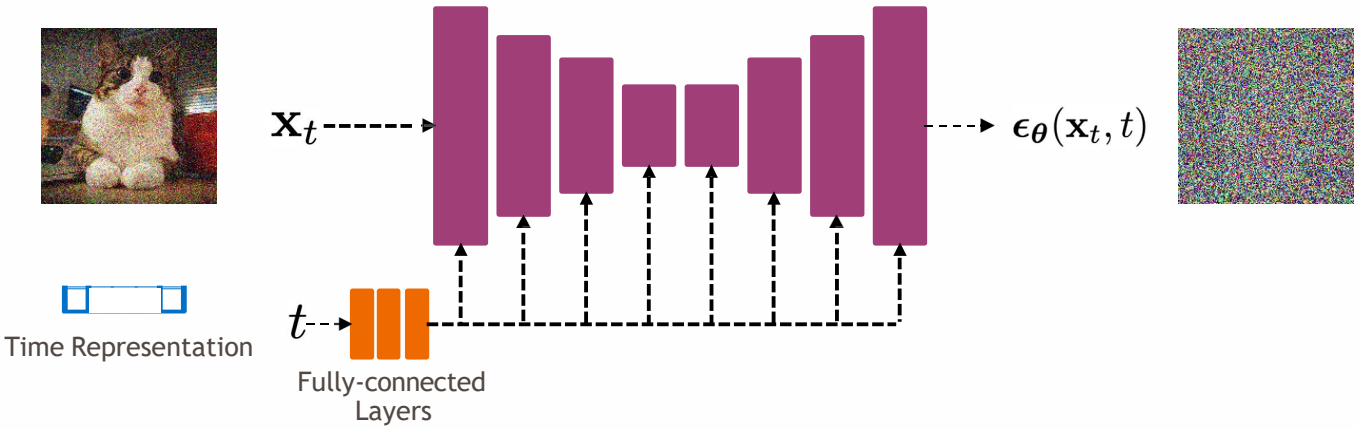
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Intuitively: During forward process we add noise to image. During reverse process we try to predict that noise with a U-Net and then subtract it from the image to denoise it.

Implementation Considerations

- Network Architectures
  - Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



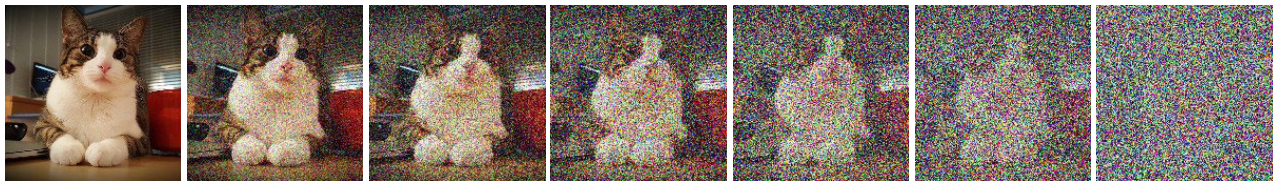
Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

Diffusion Parameters

- Noise Schedule

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

Data  Noise

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Above, β_t and σ_t^2 control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for β_t, and σ_t^2 is set equal to β_t. Slowly increase the amount of added noise.

Kingma et al. NeurIPS 2022 introduce a new parameterization of diffusion models using signal-to-noise ratio (SNR), and show how to learn the noise schedule by minimizing the variance of the training objective.

We can also train σ_t^2 while training the diffusion model by minimizing the variational bound (Improved DPM by Nichol and Dhariwal ICML 2021) or after training the diffusion model (Analytic-DPM by Bao et al. ICLR 2022).

What happens to an image in the forward diffusion process?

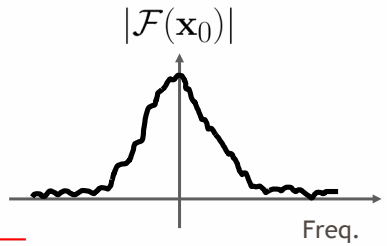
- Recall that sampling from $q(\mathbf{x}_t|\mathbf{x}_0)$ is done using

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

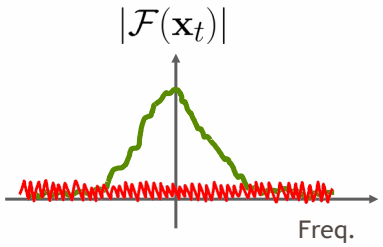
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$$

Fourier Transform

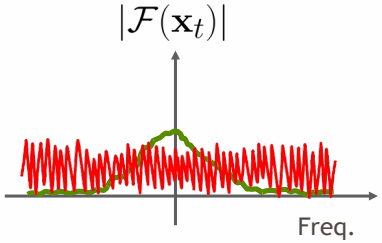
$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\,\mathcal{F}(\mathbf{x}_0) + \sqrt{(1-\bar{\alpha}_t)}\,\mathcal{F}(\epsilon)$$



Small t
$\bar{\alpha}_t \sim 1$



Large t
$\bar{\alpha}_t \sim 0$



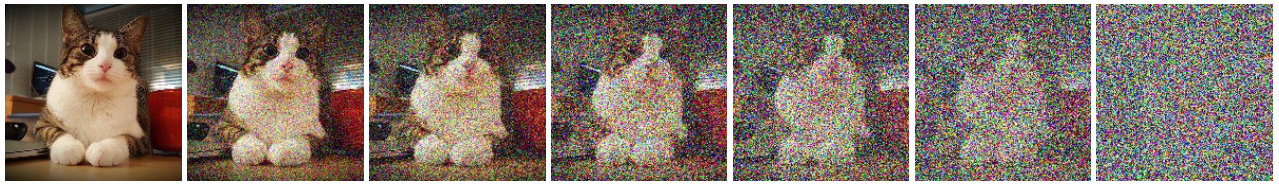
- **In the forward diffusion, the high frequency content is perturbed faster.**

Content-Detail Tradeoff

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\mathbf{x}_t}, t)||^2 \right]$$

Reverse denoising process (generative)

Data



Noise

x_0 x_1 x_2 x_3 x_4 ... x_T

The denoising model is specialized for generating the high-frequency content (i.e., low-level details)

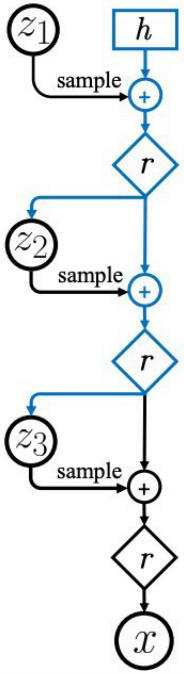
The denoising model is specialized for generating the low-frequency content (i.e., coarse content)

The weighting of the training objective for different timesteps is important!

Connection to VAEs

- Diffusion models can be considered as a special form of hierarchical VAEs.

- However, in diffusion models:
  - The encoder is fixed
  - The latent variables have the same dimension as the data
  - The denoising model is shared across different timestep
  - The model is trained with some reweighting of the variationa bound.



Vahdat and Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, NeurIPS 2020 Sønderby, et al.. Ladder variational autoencoders, NeurIPS 2016.

Summary: Denoising Diffusion Probabilistic Models

- Diffusion process can be reversed if the variance of the Gaussian noise added at each step of the diffusion is small enough.
- To reverse the process we train a U-Net that takes input: current noisy image and timestamp, and predicts the noise map..
- Training goal is to make sure that the predicted noise map at each step is unit gaussian (Note that in VAE we also required the latent space to be unit gaussian).
- During sampling/generation, subtract the predicted noise from the noisy image at time t to generate the image at time t-1 (with some weighting).
- The devil is in the details:
  - Network architectures
  - Objective weighting
  - Diffusion parameters (i.e., noise schedule)

APPLICATIONS
IMAGE SYNTHESIS, CONTROLLABLE GENERATION, TEXT-TO-IMAGE

GLIDE
OpenAI

- A 64x64 base model + a 64x64 → 256x256 super-resolution model.

- Tried classifier-free and CLIP guidance. Classifier-free guidance

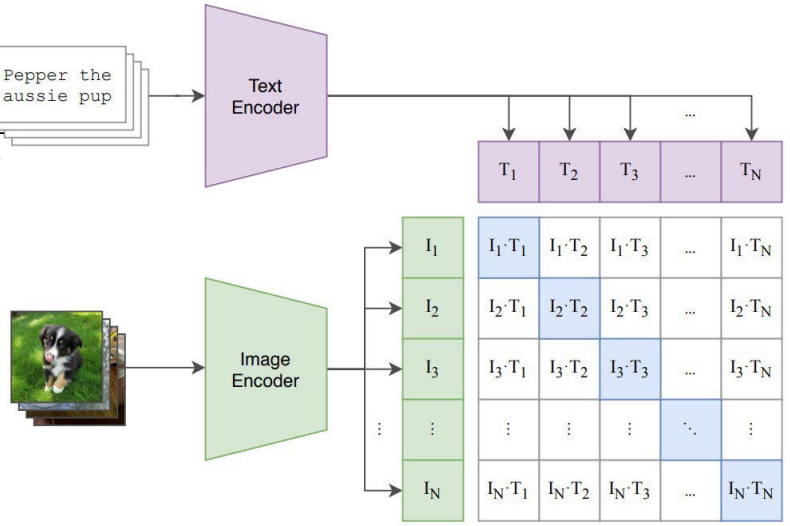
Samples generated with classifier-free guidance (256x256)

Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

CLIP guidance

- What is a CLIP model?
- Trained by contrastive cross-entropy loss:

$$-\log \frac{\exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_k)/\tau)} - \log \frac{\exp(f(\mathbf{x}_i)\cdot g(\mathbf{c}_j)/\tau)}{\sum_k \exp(f(\mathbf{x}_k)\cdot g(\mathbf{c}_j)/\tau)}$$



- The optimal value of $f(\mathbf{x})\cdot g(\mathbf{c})$ is

$$\log \frac{p(\mathbf{x},\mathbf{c})}{p(\mathbf{x})p(\mathbf{c})} = \log p(\mathbf{c}|\mathbf{x}) - \log p(\mathbf{c})$$

Radford et al., "Learning Transferable Visual Models From Natural Language Supervision", 2021.
Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

CLIP guidance

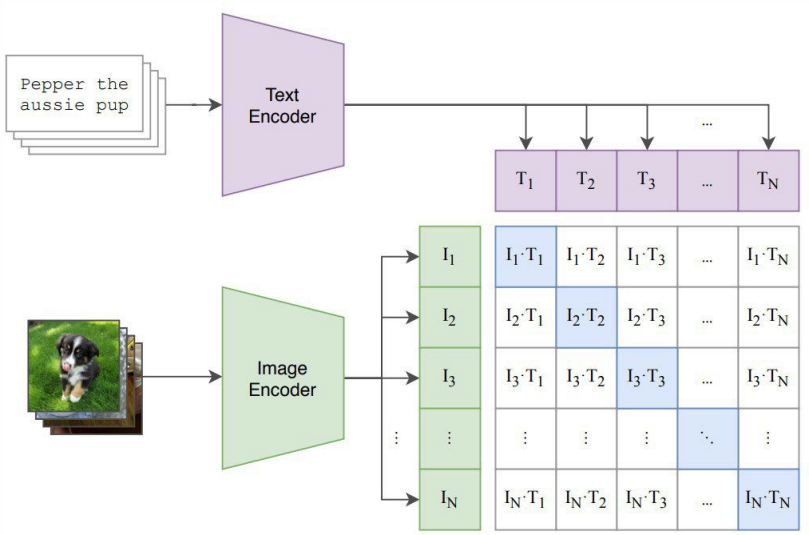
- Replace the classifier in classifier guidance with a CLIP model

- Sample with a modified score:

$$\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)]$$

$$= \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega(\underbrace{\log p(\mathbf{c}|\mathbf{x}_t) - \log p(\mathbf{c})}_{\text{CLIP model}})]$$

$$= \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega(f(\mathbf{x}_t) \cdot g(\mathbf{c}))]$$

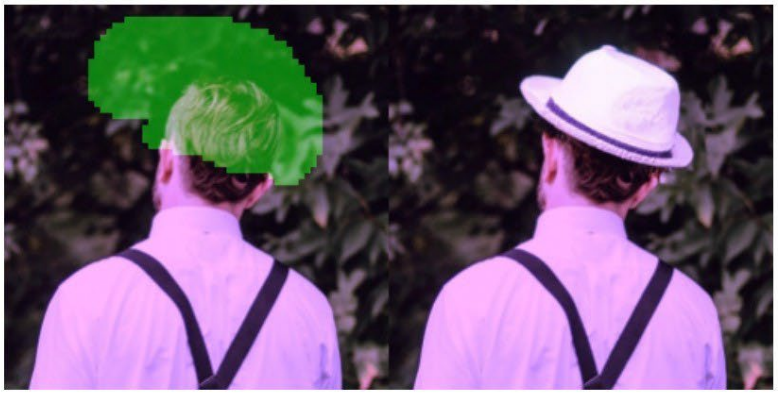


Radford et al., "Learning Transferable Visual Models From Natural Language Supervision", 2021.
Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

GLIDE
OpenAI

- Fine-tune the model especially for inpainting: feed randomly occluded images with an additional mask channel as the input.

"an old car in a snowy forest"



"a man wearing a white hat"

Text-conditional image inpainting examples

Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021.

DALL·E 2

- OpenAI

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

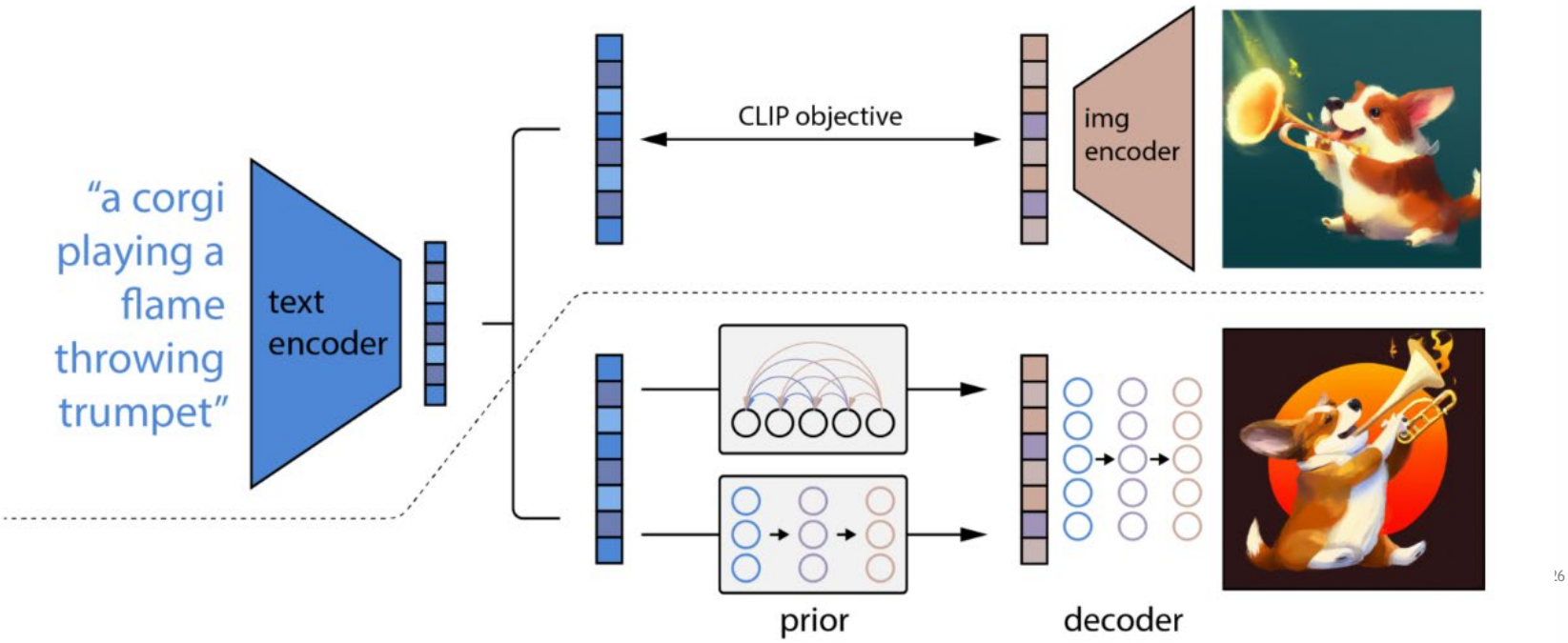
1kx1k Text-to-image generation.

Outperform DALL-E (autoregressive transformer).

Ramesh et al., “Hierarchical Text-Conditional Image Generation with CLIP Latents”, arXiv 2022.

DALL·E 2

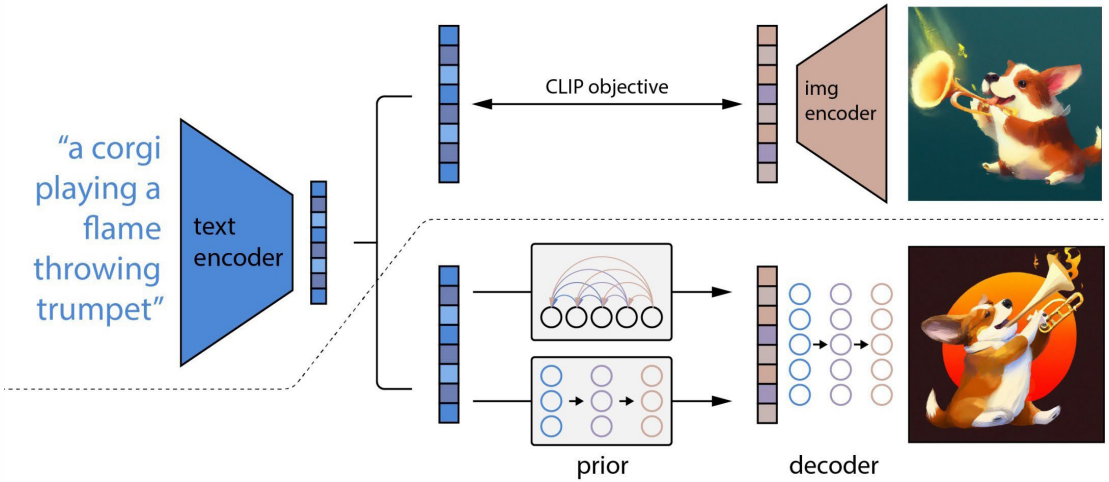
- Model components
  - Prior: produces CLIP image embeddings conditioned on the caption.
  - Decoder: produces images conditioned on CLIP image embeddings and text.



Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

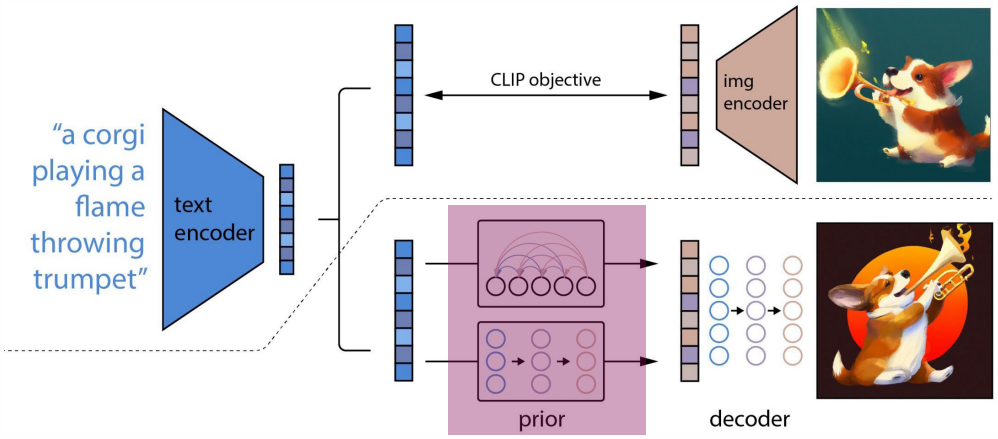
DALL·E 2
Model components

- Why conditional on CLIP image embeddings?
  - CLIP image embeddings capture high-level semantic meaning.
  - Latents in the decoder model take care of the rest.
  - The bipartite latent representation enables several text-guided image manipulation tasks.



DALL·E 2
Model components (1/2): prior model

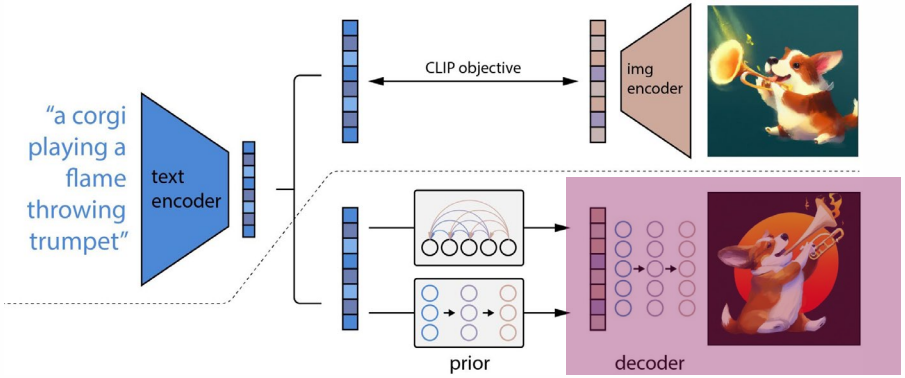
- Prior: produces CLIP image embeddings conditioned on the caption.
  - Option 1. autoregressive prior: quantize image embedding to a seq. of discrete codes and predict them autoregressively.
  - Option 2. diffusion prior: model the continuous image embedding by diffusion models conditioned on caption.



Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2
Model components (2/2): decoder model

- Decoder: produces images conditioned on CLIP image embeddings (and text).
  - Cascaded diffusion models: 1 base model (64x64), 2 super-resolution models (64x64 → 256x256, 256x256 → 1024x1024).
  - Largest super-resolution model is trained on patches and takes full-res inputs at inference time.
  - Classifier-free guidance & noise conditioning augmentation are important.



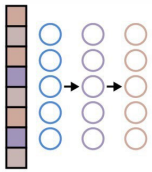
Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2
Bipartite latent representations

Bipartite latent representations $(\mathbf{z}, \mathbf{x}_T)$

$\mathbf{z}$: CLIP image embeddings

$\mathbf{x}_T$: inversion of DDIM sampler
(latents in the decoder model)



decoder

Near exact
reconstruction

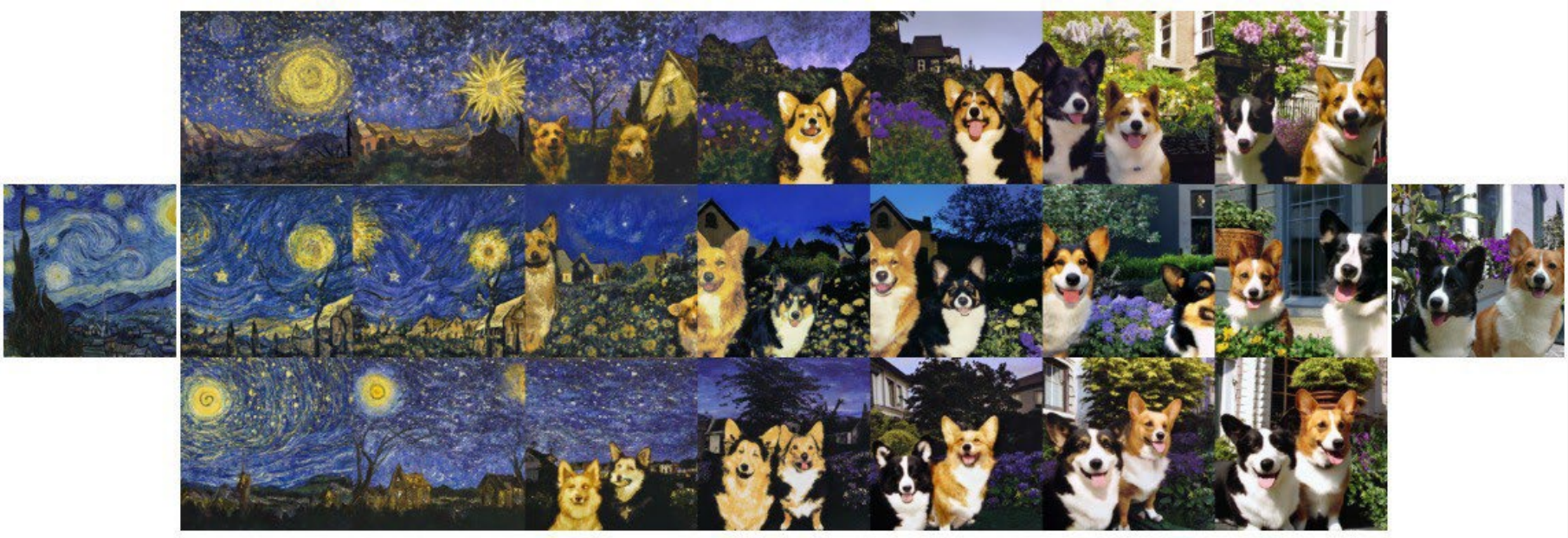
Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2
Image variations

Fix the CLIP embedding $\mathbf{z}$.

Decode using different decoder latents $\mathbf{x}_T$

DALL·E 2
Image interpolation



Interpolate image CLIP embeddings $\mathbf{z}$.

Use different $\mathbf{x}_T$ to get different interpolation trajectories.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

DALL·E 2
Text Diffs

a photo of a cat → an anime drawing of a super saiyan cat, artstation

a photo of a victorian house → a photo of a modern house

a photo of an adult lion → a photo of lion cub

Change the image CLIP embedding towards the difference of the text CLIP embeddings of two prompts.

Decoder latent is kept as a constant.

Imagen
Google Research, Brain team

- Input: text; Output: 1kx1k images
  - An unprecedented degree of photorealism
  - SOTA automatic scores & human ratings
  - A deep level of language understanding
  - Extremely simple
  - no latent space, no quantization

A brain riding a rocketship heading towards the moon.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Imagen (Google Research, Brain team…被解散了…)

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen
Google Research, Brain team

A dragon fruit wearing karate belt in the snow.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Imagen
Google Research, Brain team

A relaxed garlic with a blindfold reading a newspaper while floating in a pool of tomato soup.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

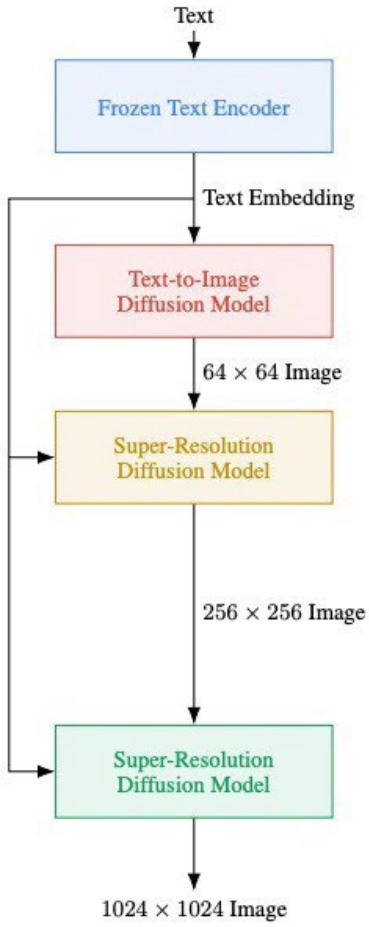
Imagen
Google Research, Brain team

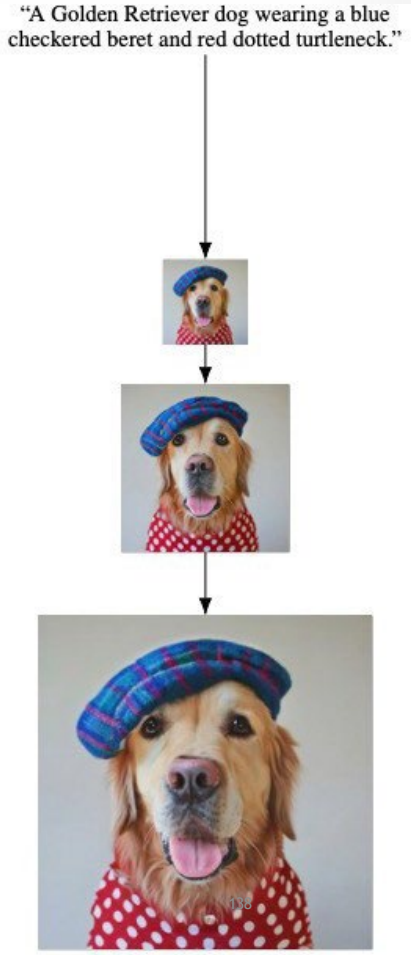
A cute hand-knitted koala wearing a sweater with 'CVPR' written on it.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Imagen

- Key modeling components:
  - Cascaded diffusion models
  - Classifier-free guidance and dynamic thresholding.
  - Frozen large pretrained language models as text encoders. (T5-XXL)





Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

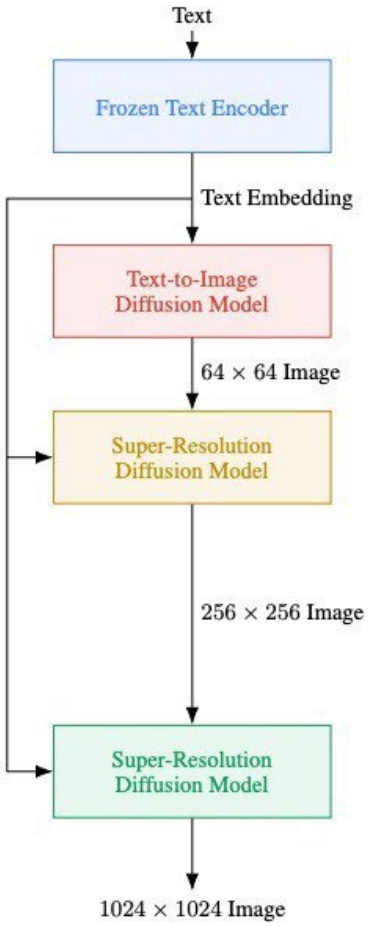
Imagen

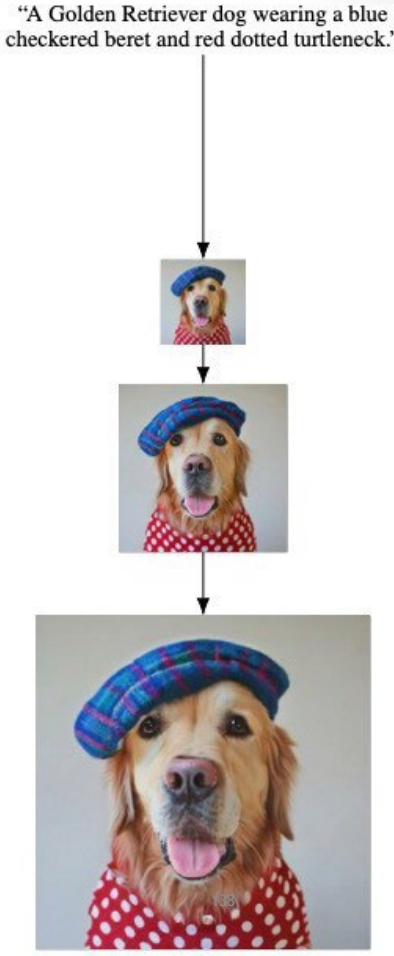
- Key observations:

- Beneficial to use text conditioning for all super-res models.
  - Noise conditioning augmentation weakens
    - information from low-res models, thus
    - needs text conditioning as extra information input.

- Scaling text encoder is extremely efficient.
  - More important than scaling diffusion model size.

- Human raters prefer T5-XXL as the text encoder over CLIP encoder on DrawBench.

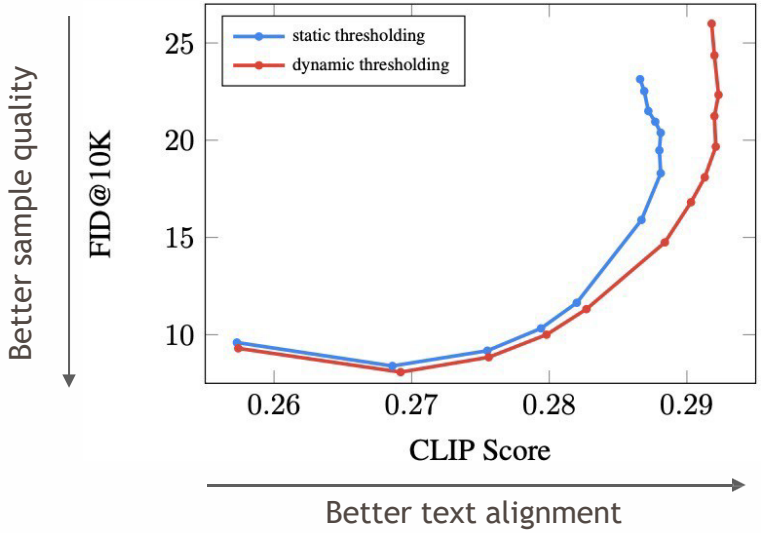




Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen

- Dynamic thresholding
  - Large classifier-free guidance weights → better text alignment, worse image quality



Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen
Dynamic thresholding

- Large classifier-free guidance weights → better text alignment, worse image quality
- Hypothesis : at large guidance weight, the generated images are saturated due to the very large gradient updates during sampling
- Solution – dynamic thresholding: adjusts the pixel values of samples at each sampling step to be within a dynamic range computed over the statistics of the current samples.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen
Dynamic thresholding

Static thresholding

Dynamic thresholding

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

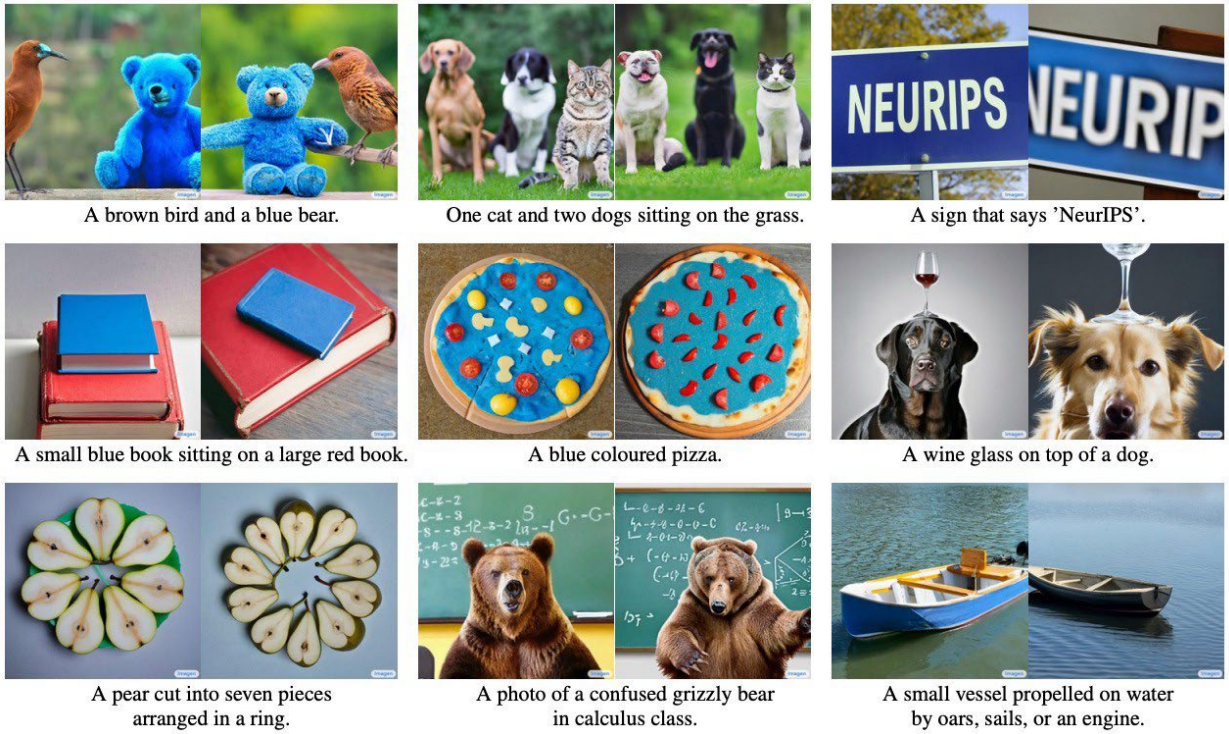
Imagen

- DrawBench: new benchmark for text-to-image evaluations
  - A set of 200 prompts to evaluate text-to-image models across multiple dimensions.
  - E.g., the ability to faithfully render different colors, numbers of objects, spatial relations, text in the scene, unusual interactions between objects.
  - Contains complex prompts, e.g, long and intricate descriptions, rare words, misspelled prompts.

Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen

- DrawBench: new benchmark for text-to-image evaluations



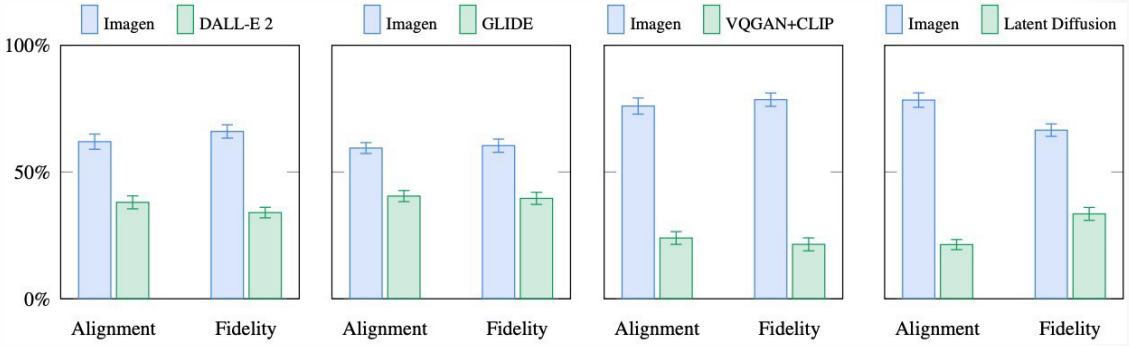
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Imagen Evaluations

Imagen got SOTA automatic evaluation scores on COCO dataset

Model	FID-30K	Zero-shot FID-30K
AttnGAN [76]	35.49	
DM-GAN [83]	32.64	
DF-GAN [69]	21.42	
DM-GAN + CL [78]	20.79	
XMC-GAN [81]	9.33	
LAFITE [82]	8.12	
Make-A-Scene [22]	7.55	
DALL-E [53]		17.89
LAFITE [82]		26.94
GLIDE [41]		12.24
DALL-E 2 [54]		10.39
Imagen (Our Work)		**7.27**

Imagen is preferred over recent work by human raters in sample quality & image-text alignment on DrawBench.



Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Stable Diffusion

- Latest & Publicly available text-to-image generation

High-Resolution Image Synthesis with Latent Diffusion Models

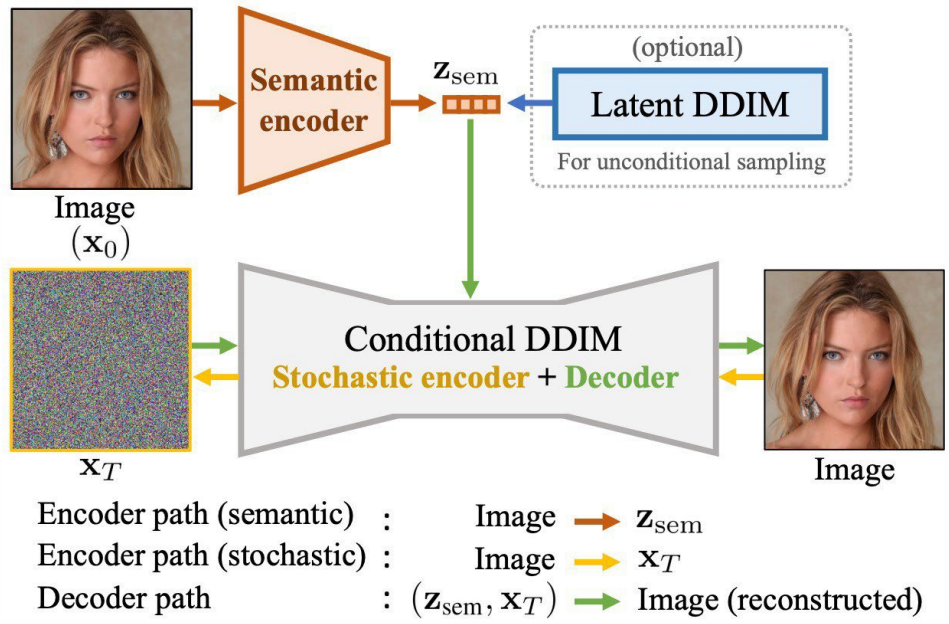
Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer

CVPR '22 Oral | GitHub | arXiv | Project page

Stable Diffusion is a latent text-to-image diffusion model. Thanks to a generous compute donation from Stability AI and support from LAION, we were able to train a Latent Diffusion Model on 512x512 images from a subset of the LAION-5B database. Similar to Google's Imagen, this model uses a frozen CLIP ViT-L/14 text encoder to condition the model on text prompts. With its 860M UNet and 123M text encoder, the model is relatively lightweight and runs on a GPU with at least 10GB VRAM. See this section below and the model card.

Diffusion Autoencoders

- Learning semantic meaningful latent representations in diffusion models

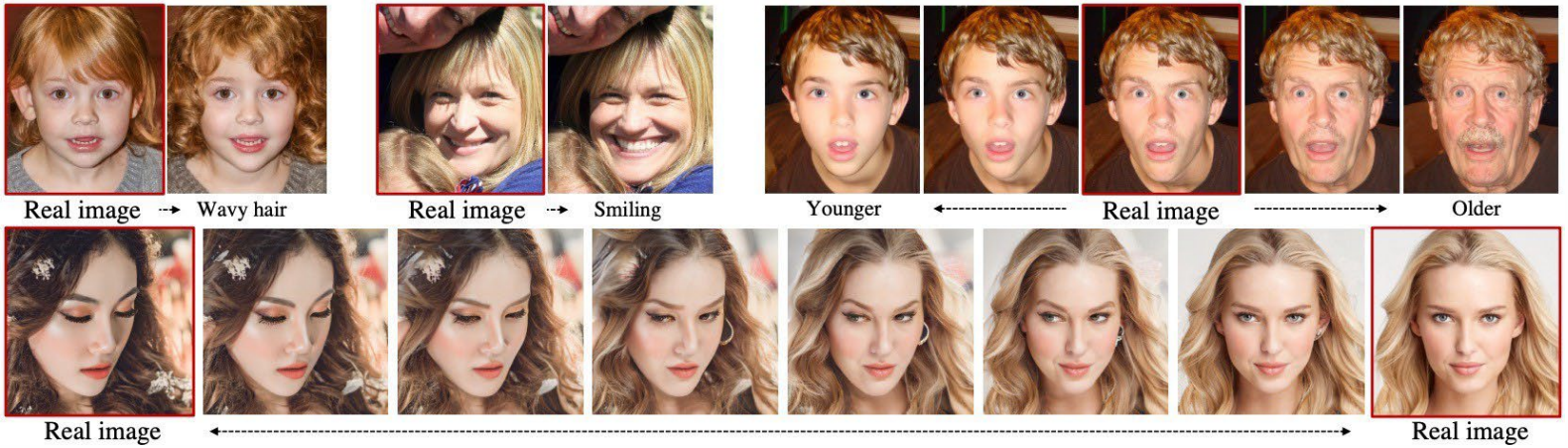


To be discussed in detail in paper presentation

Preechakul et al., "Diffusion Autoencoders: Toward a Meaningful and Decodable Representation", CVPR 2022.

Diffusion Autoencoders

- Learning semantic meaningful latent representations in diffusion models



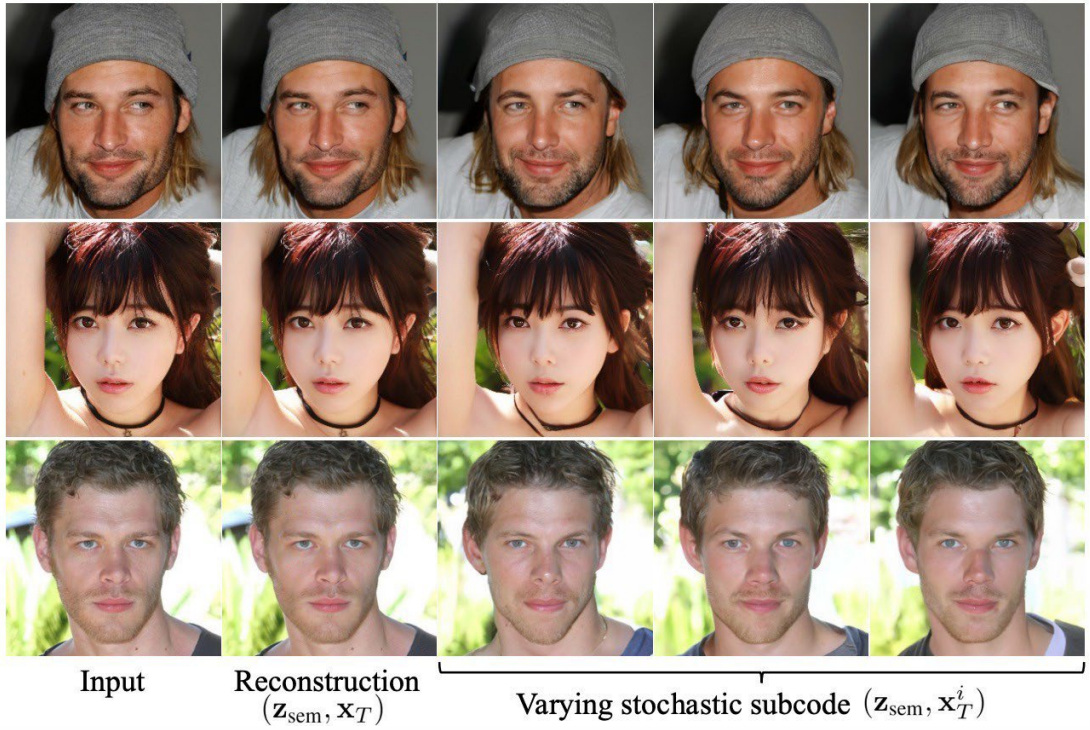
Changing the semantic latent $\mathbf{z}_{\text{sem}}$

Very similar to StyleGAN based editing. Zsem is the latent representation similar to the W/W+ space of StyleGAN

Preechakul et al., "Diffusion Autoencoders: Toward a Meaningful and Decodable Representation", CVPR 2022.

Diffusion Autoencoders

- Learning semantic meaningful latent representations in diffusion models



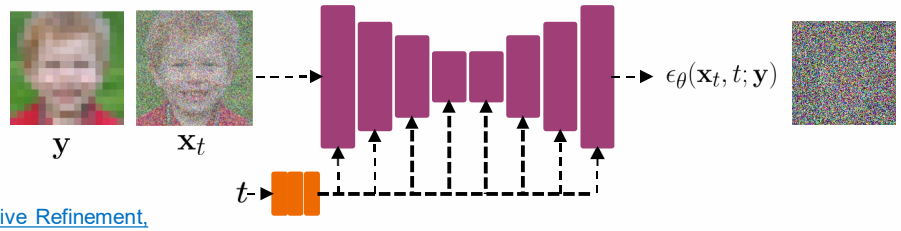
Preechakul et al., "Diffusion Autoencoders: Toward a Meaningful and Decodable Representation", CVPR 2022.

Super-Resolution

- Super-Resolution via Repeated Refinement (SR3)
  - Image super-resolution can be considered as training high-resolution image $p(\mathbf{x}|\mathbf{y})$ where $\mathbf{y}$ is a low-resolution image and $\mathbf{x}$ is the corresponding
  - Train a score model for $\mathbf{x}$ conditioned on $\mathbf{y}$ using:

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}\, \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\, \mathbb{E}_t\, ||\epsilon_\theta(\mathbf{x}_t, t; \mathbf{y}) - \epsilon||_p^p$$

  - The conditional score is simply a U-Net with $\mathbf{x_t}$ and $\mathbf{y}$ (resolution image) concatenated.

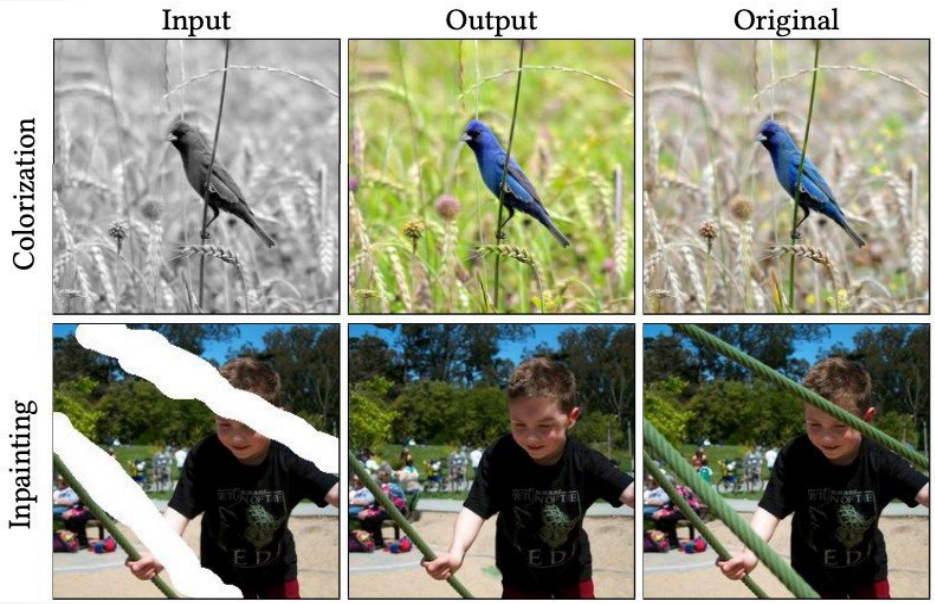


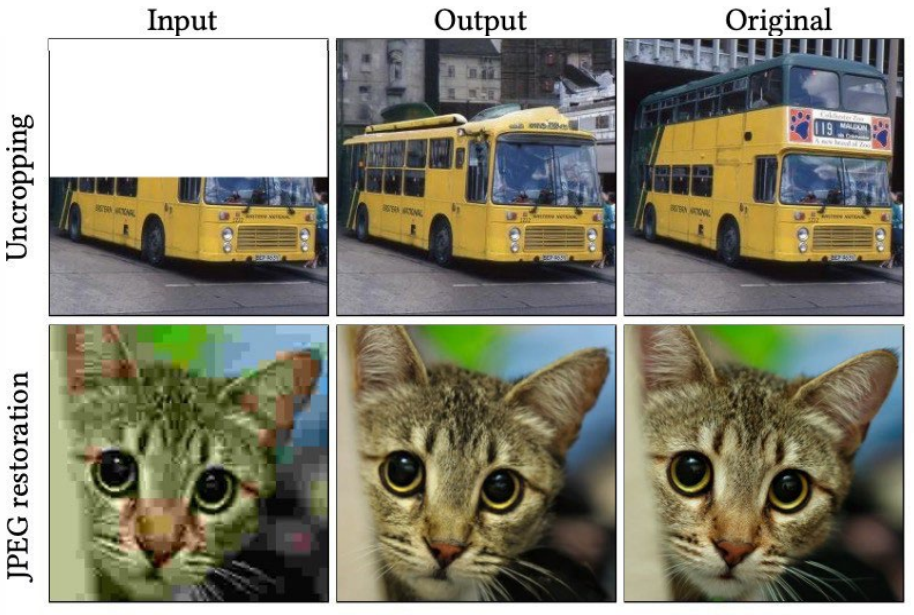
Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

Super-Resolution
Super-Resolution via Repeated Refinement (SR3)

Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

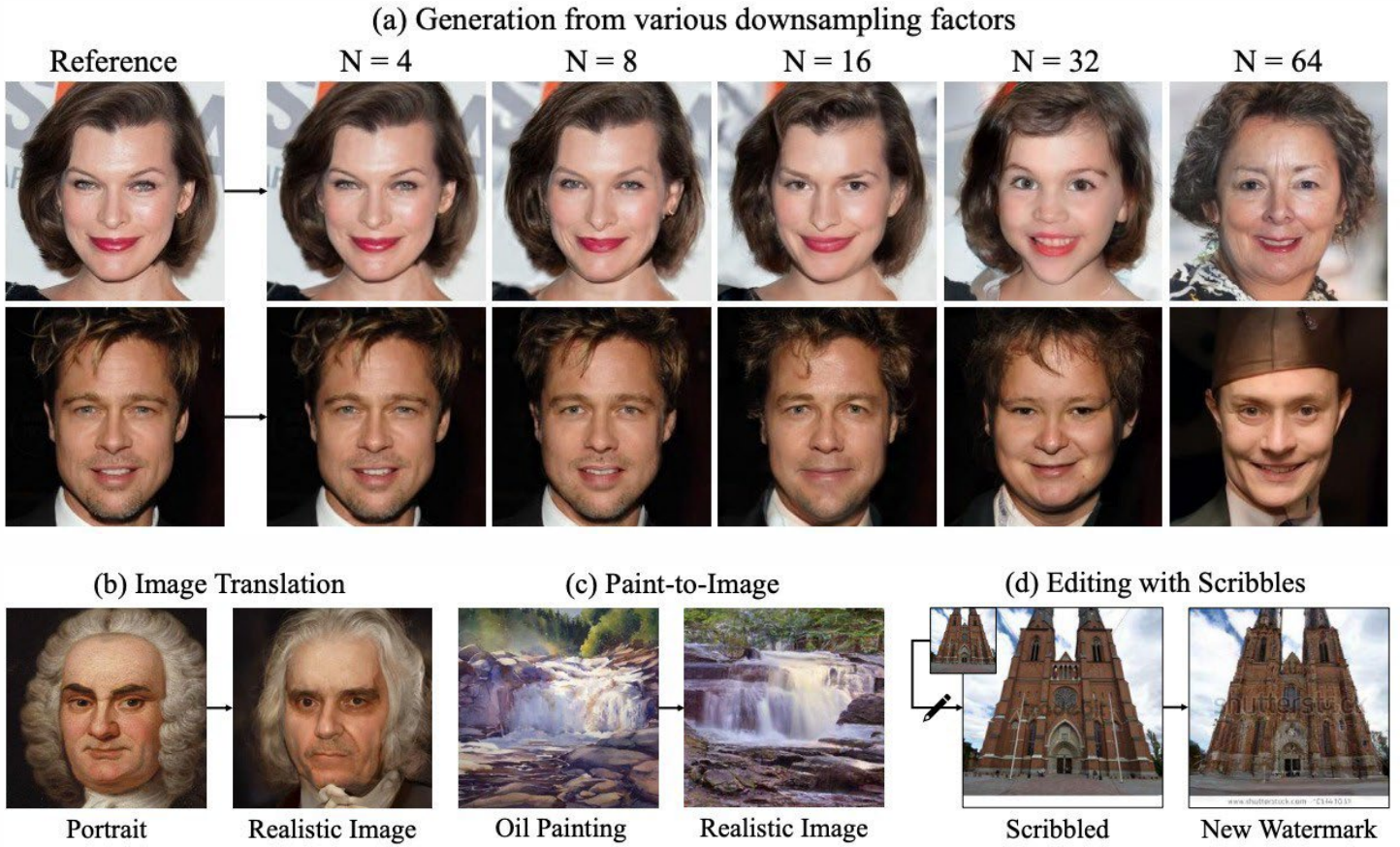
Image-to-Image Translation
Palette: Image-to-Image Diffusion Models





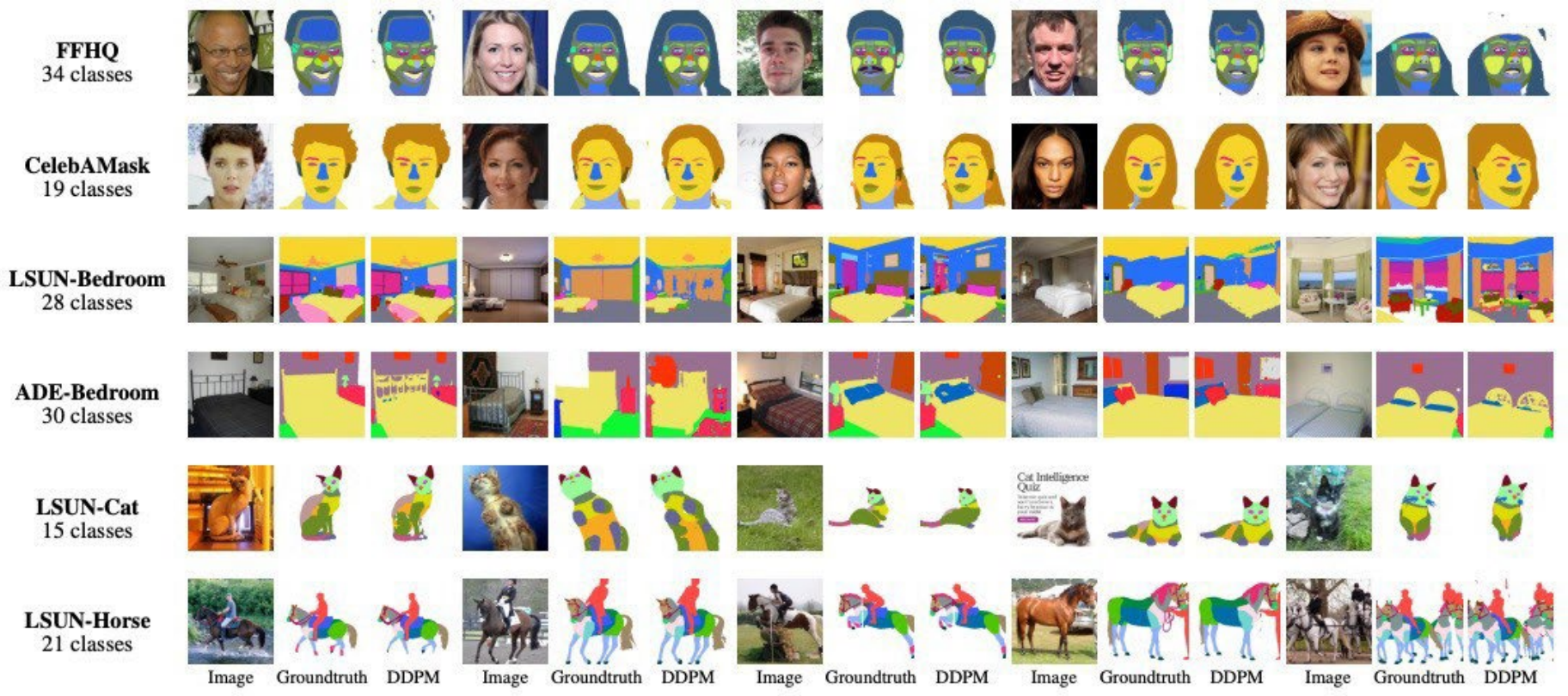
Saharia et al., Palette: Image-to-Image Diffusion Models, 2022

Conditional Generation
Iterative Latent Variable Refinement (ILVR)



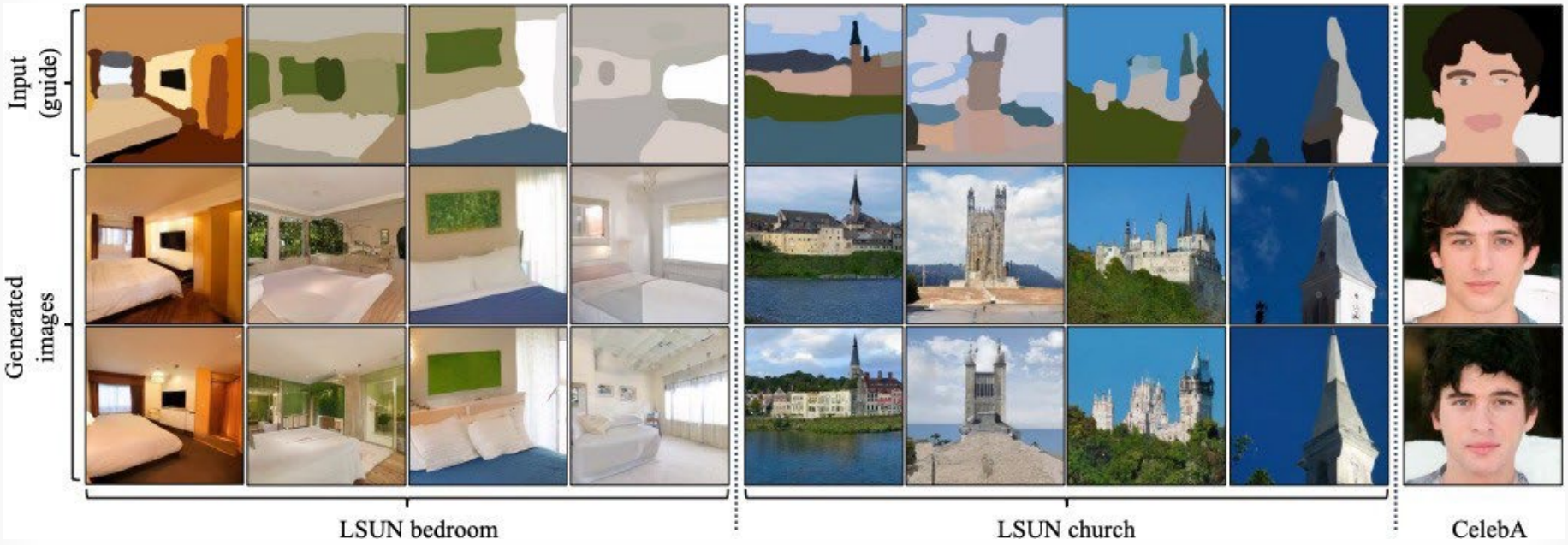
[Choi et al., ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models, ICCV 2021]

Semantic Segmentation



Baranchuk et al., Label-Efficient Semantic Segmentation with Diffusion Models, ICLR 2022

Image Editing (SDEdit)



Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022

APPENDIX

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[\log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[\log p_\theta(x^{(i)}) \right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Taking expectation wrt. z (using encoder network) will come in handy later

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)} \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}
\end{aligned}
$$

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && (\text{Logarithms})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

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

The expectation wrt. z (using encoder network) let us write nice KL terms

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

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

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

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

We want to maximize the data likelihood

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

We want to maximize the data likelihood

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

We want to maximize the data likelihood

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational Autoencoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

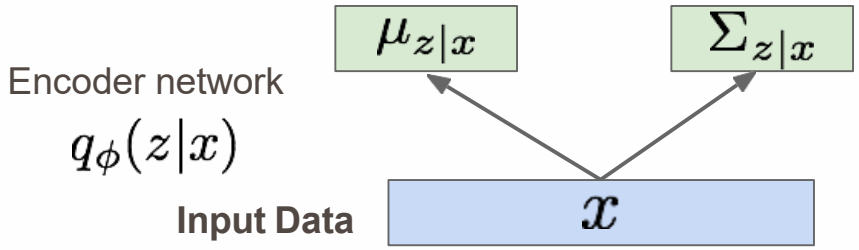
Let's look at computing the bound (forward pass) for a given minibatch of input data

Input Data x

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

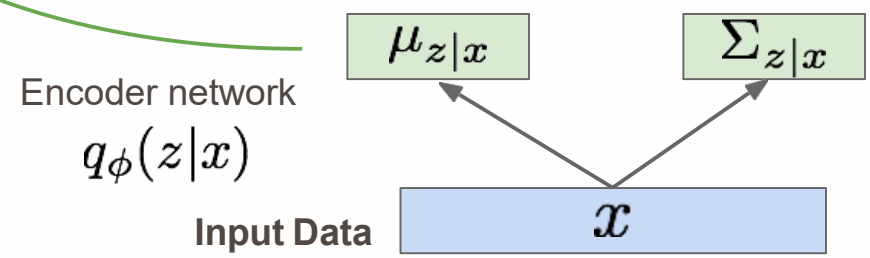


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

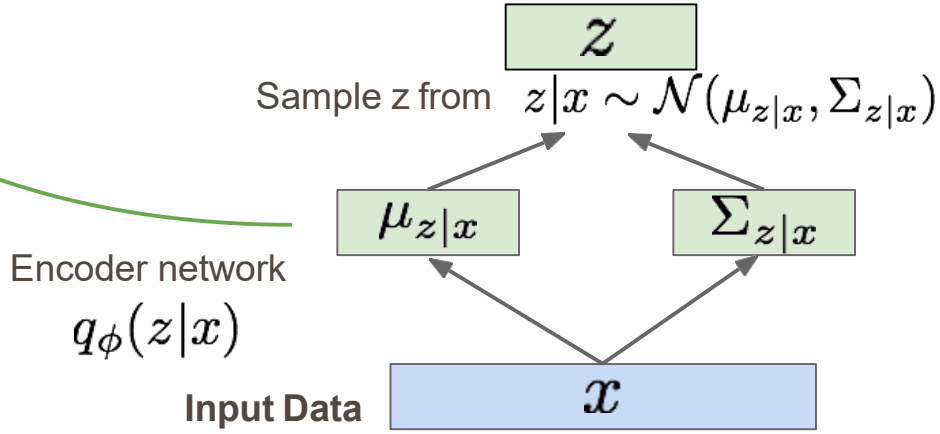


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

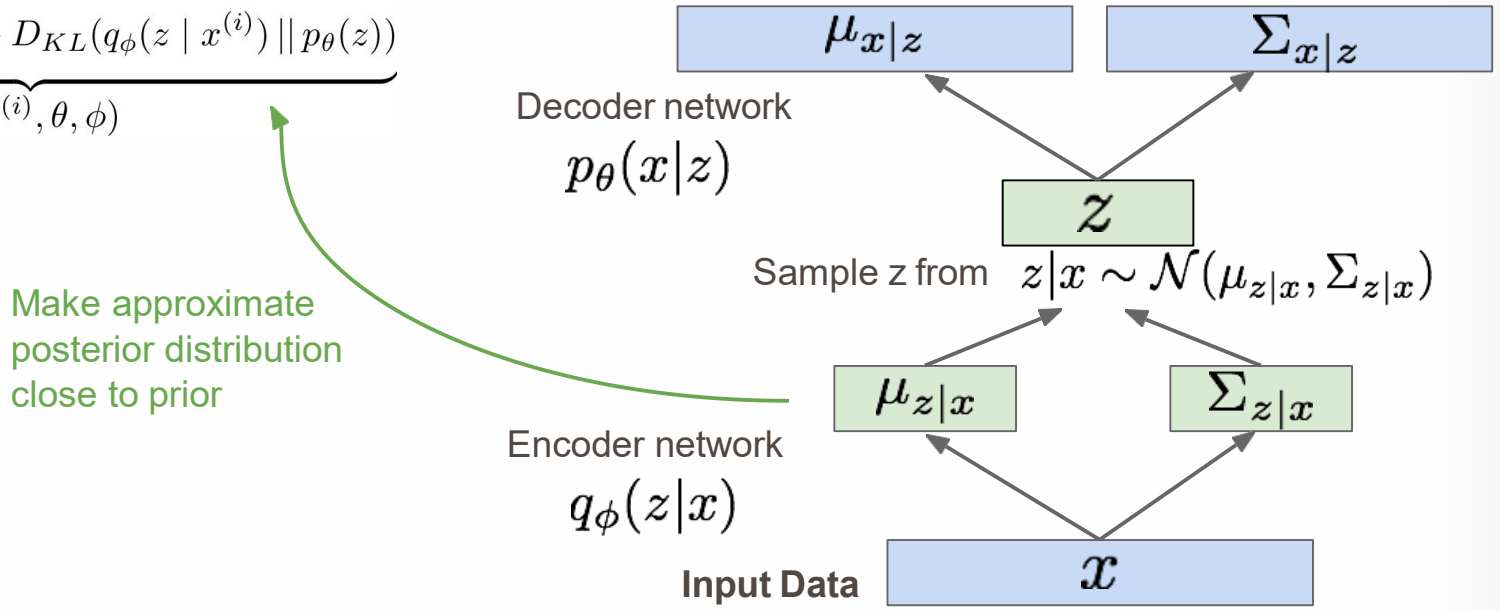
Make approximate posterior distribution close to prior



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

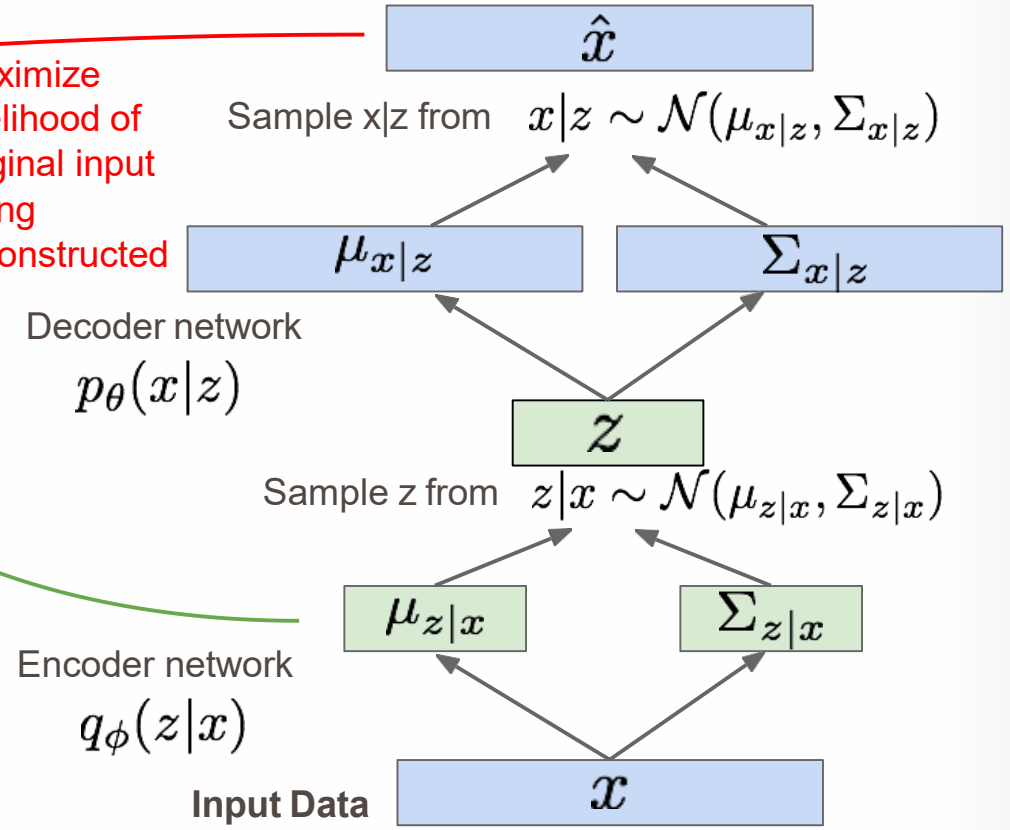


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed



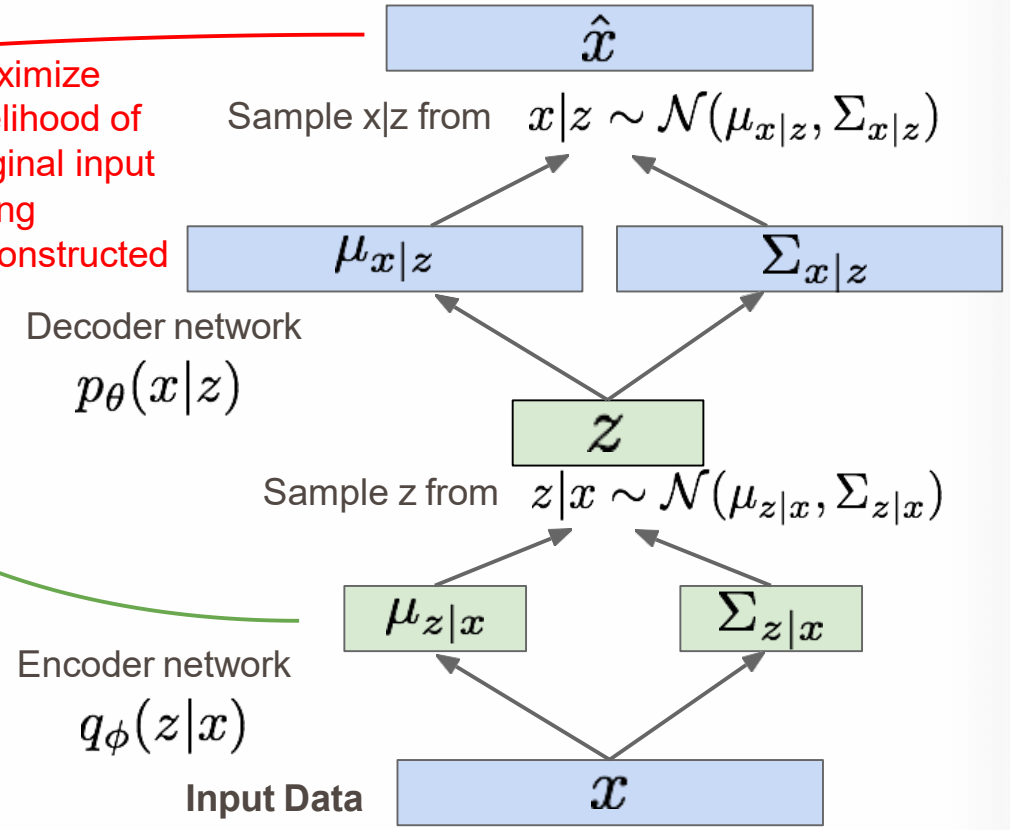
Make approximate posterior distribution close to prior

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

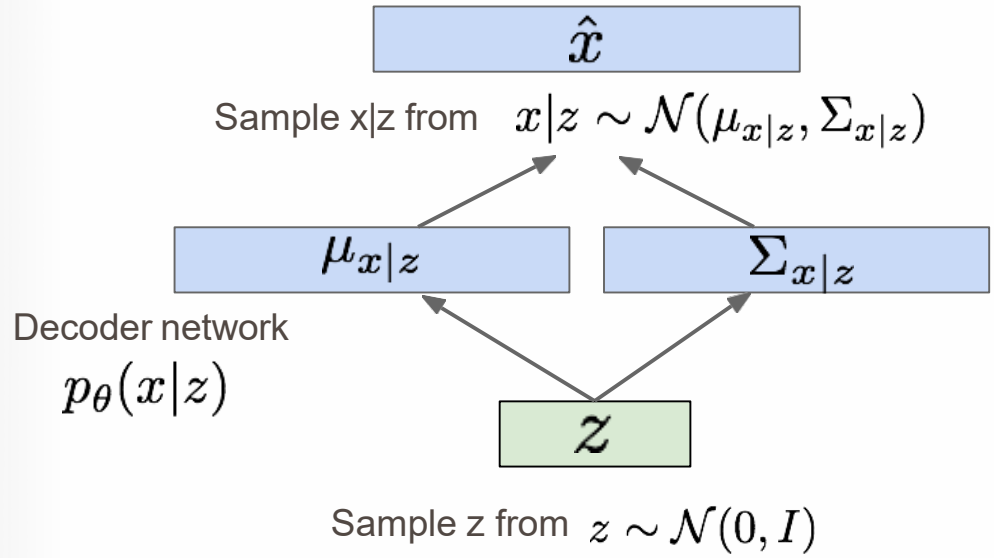


Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!

Variational Autoencoders: Generating Data!

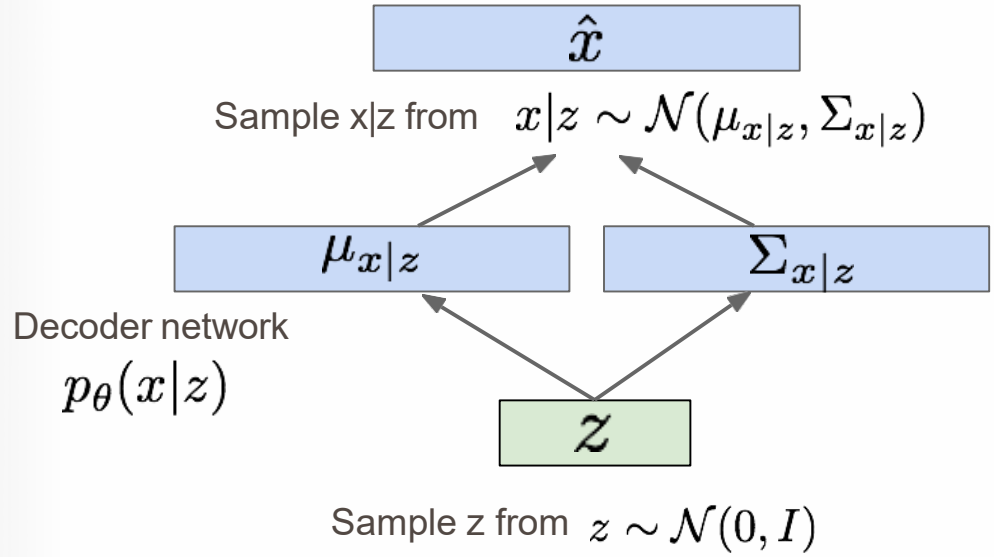
Use decoder network. Now sample z from prior!

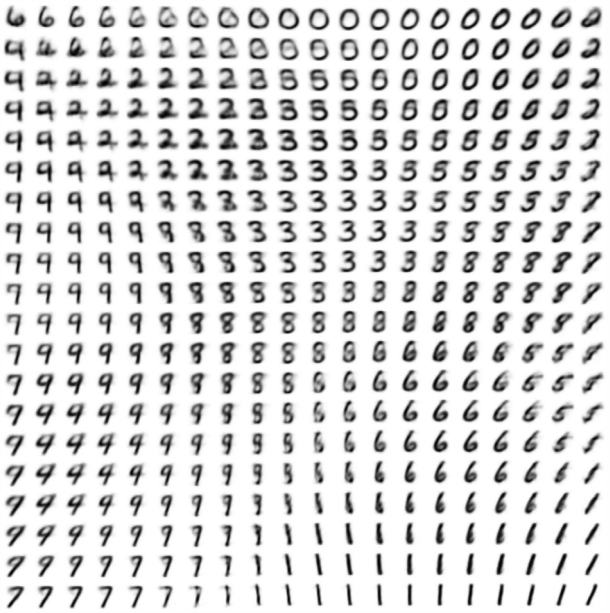


Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!

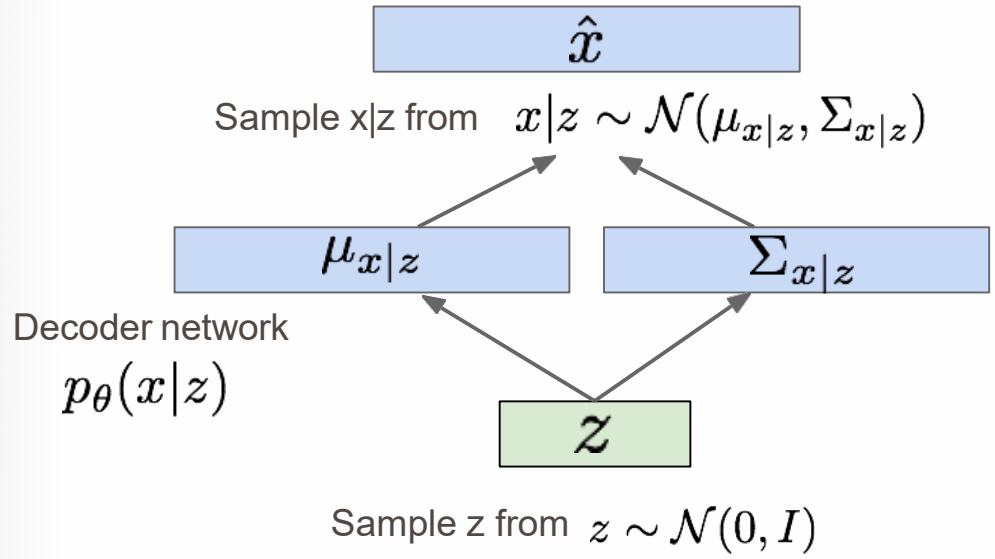




Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

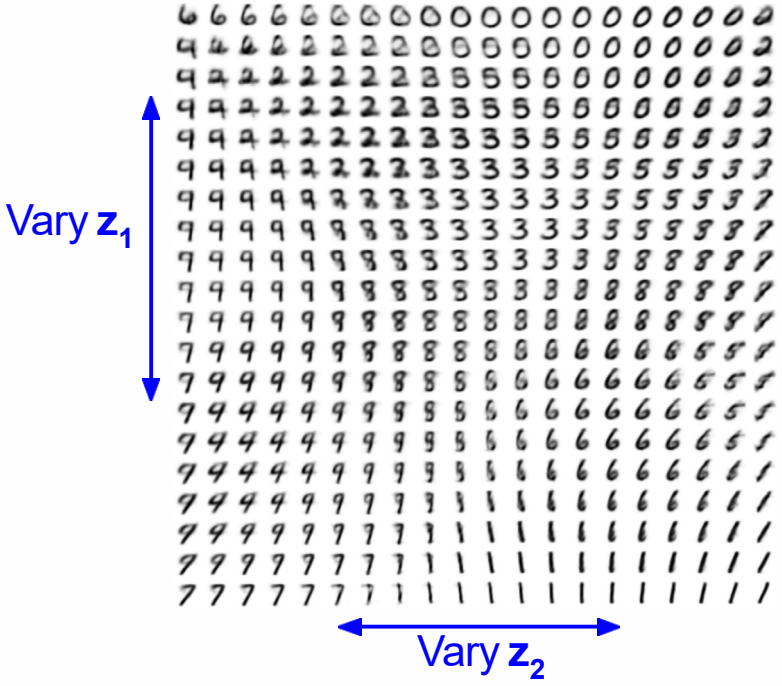
Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

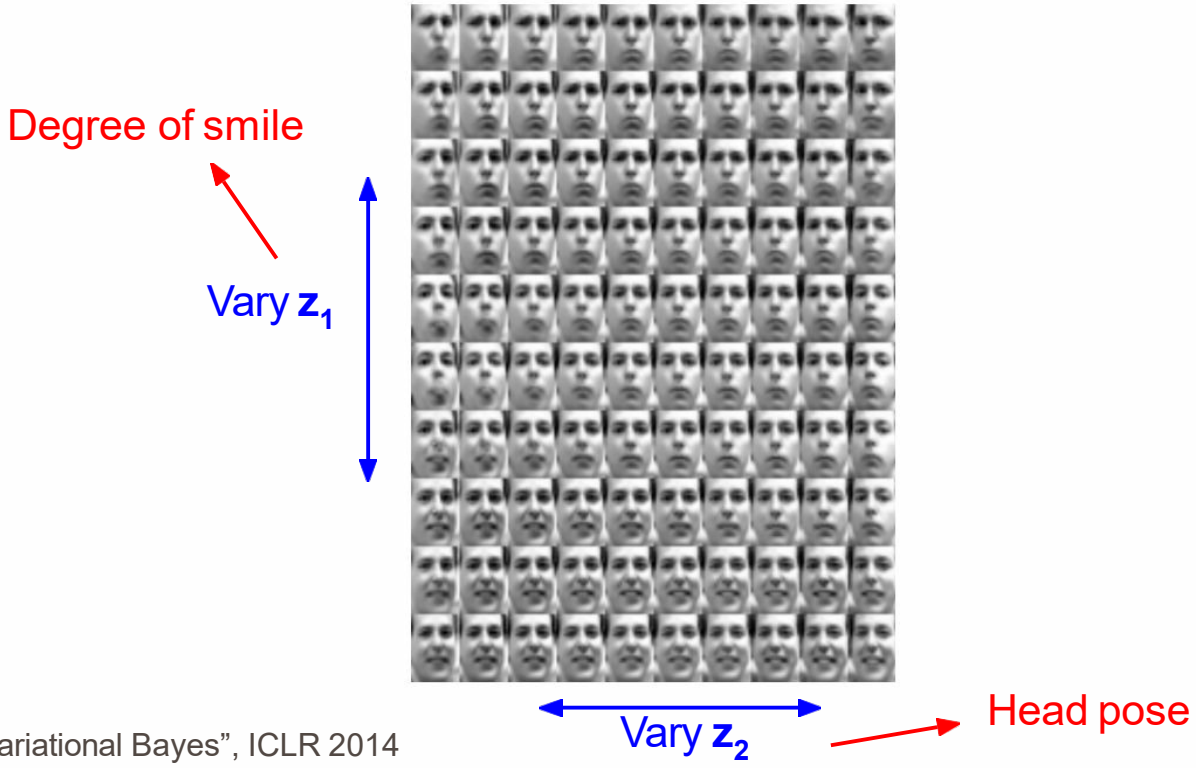
Data manifold for 2-d **z**



Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation



Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Can we use other divergence?

Name	$D_f(P\|\|Q)$	Generator $f(u)$
Total variation	$\frac{1}{2}\int \lvert p(x)-q(x)\rvert \,\mathrm{d}x$	$\frac{1}{2}\lvert u-1\rvert$
Kullback-Leibler	$\int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x$	$u\log u$
Reverse Kullback-Leibler	$\int q(x)\log\frac{q(x)}{p(x)}\,\mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)}\,\mathrm{d}x$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)}\,\mathrm{d}x$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)}-\sqrt{q(x)}\right)^2\,\mathrm{d}x$	$\left(\sqrt{u}-1\right)^2$
Jeffrey	$\int (p(x)-q(x))\log\left(\frac{p(x)}{q(x)}\right)\,\mathrm{d}x$	$(u-1)\log u$
Jensen-Shannon	$\frac{1}{2}\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x$	$-(u+1)\log\frac{1+u}{2}+u\log u$
Jensen-Shannon-weighted	$\int p(x)\pi\log\frac{p(x)}{\pi p(x)+(1-\pi)q(x)}+(1-\pi)q(x)\log\frac{q(x)}{\pi p(x)+(1-\pi)q(x)}\,\mathrm{d}x$	$\pi u\log u-(1-\pi+\pi u)\log(1-\pi+\pi u)$
GAN	$\int p(x)\log\frac{2p(x)}{p(x)+q(x)}+q(x)\log\frac{2q(x)}{p(x)+q(x)}\,\mathrm{d}x-\log(4)$	$u\log u-(u+1)\log(u+1)$

Using the divergence
you want...and take care of it...

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1-\log(-t)$
Pearson χ^2	$\frac{1}{4}t^2+t$
Neyman χ^2	$2-2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$
Jensen-Shannon	$-\log(2-\exp(t))$
Jensen-Shannon-weighted	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1-\exp(t))$