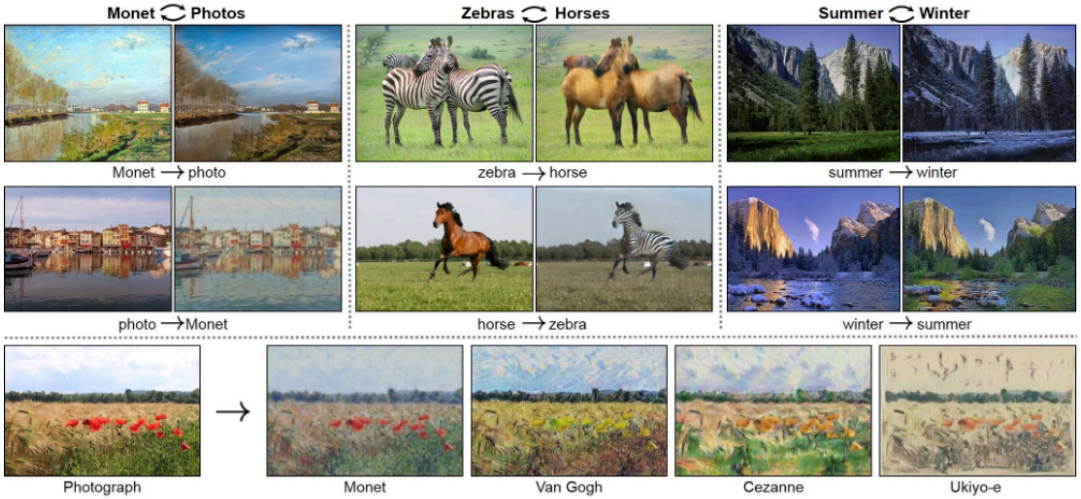


Generative Models

Jay Urbain, PhD

Credits:

- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Fei-Fei Li & Justin Johnson & Serena Yeung. (2018). Cs231n, Stanford University.
- Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

You might not think that programmers are artists, but programming is an extremely creative profession. It's logic-based creativity.
- John Romero

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.

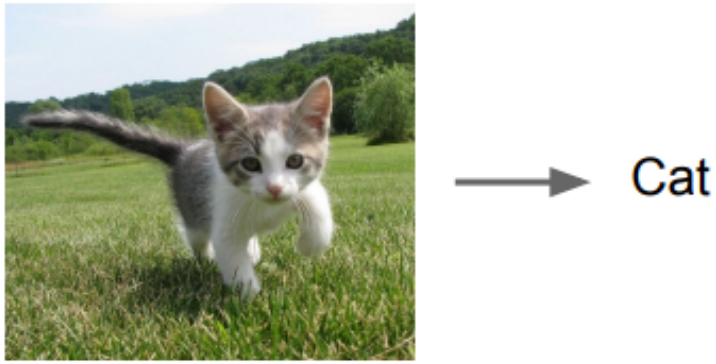
Supervised vs Unsupervised Learning

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.



Classification

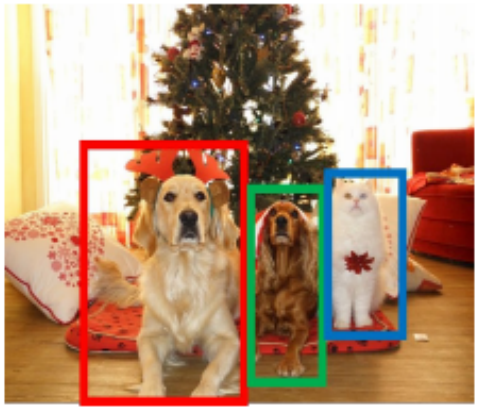
Supervised vs Unsupervised Learning

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

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

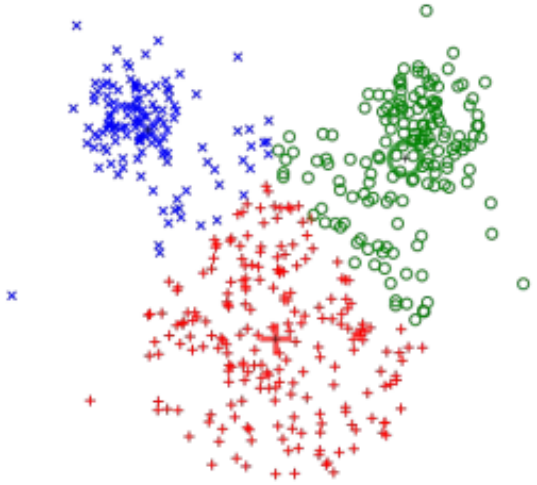
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

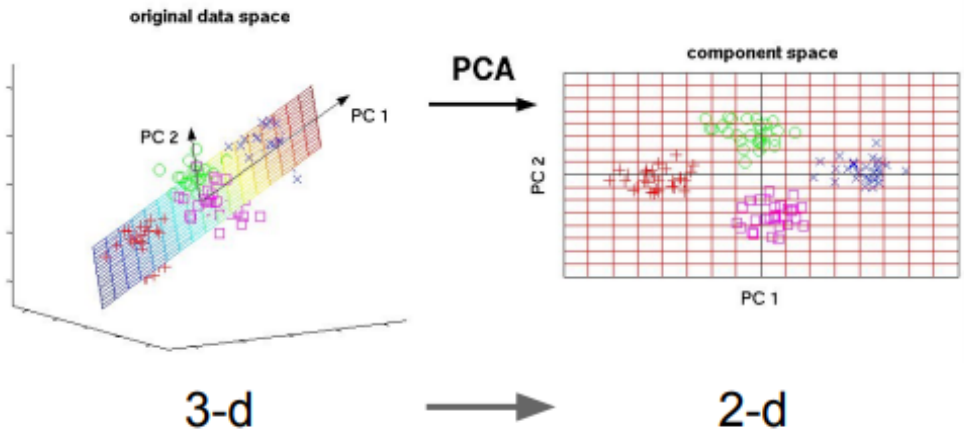
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Principal Component Analysis
(Dimensionality reduction)

Supervised vs Unsupervised Learning

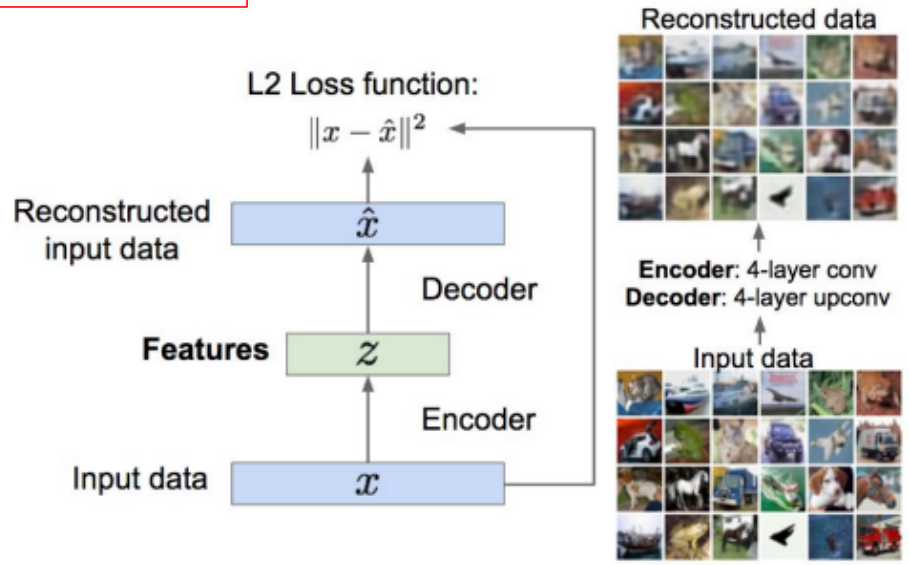
Use no external labels

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Autoencoders
(Feature learning)

Supervised vs Unsupervised Learning

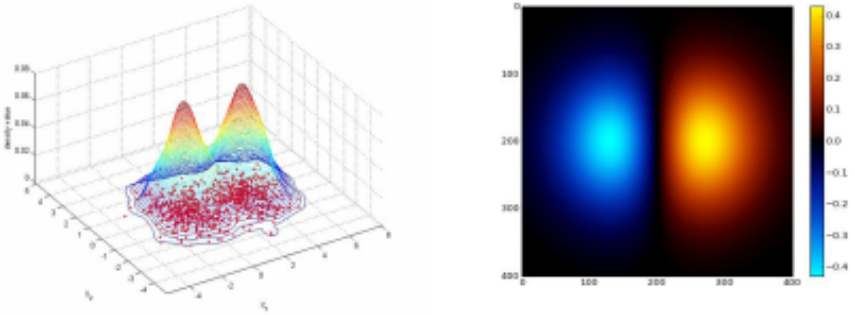
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

1-d density estimation



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
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Humans can learn from unlabeled data!

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

Training data is cheap

Data: x
Just data, no labels!

Holy grail: Solve unsupervised learning => understand structure of visual world

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Generative Models

Given training data, generate new samples from same distribution

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

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

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

Generative Models

Given training data, generate new samples from same distribution

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

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

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

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

- Generative models are one of the most promising approaches to understand the vast amount of data that surrounds us nowadays.
- OpenAI: "…algorithms which are able to create data might be substantially better at understanding intrinsically the world."

"What I cannot create, I do not understand."—Richard P. Feynman

Taxonomy of Generative Models

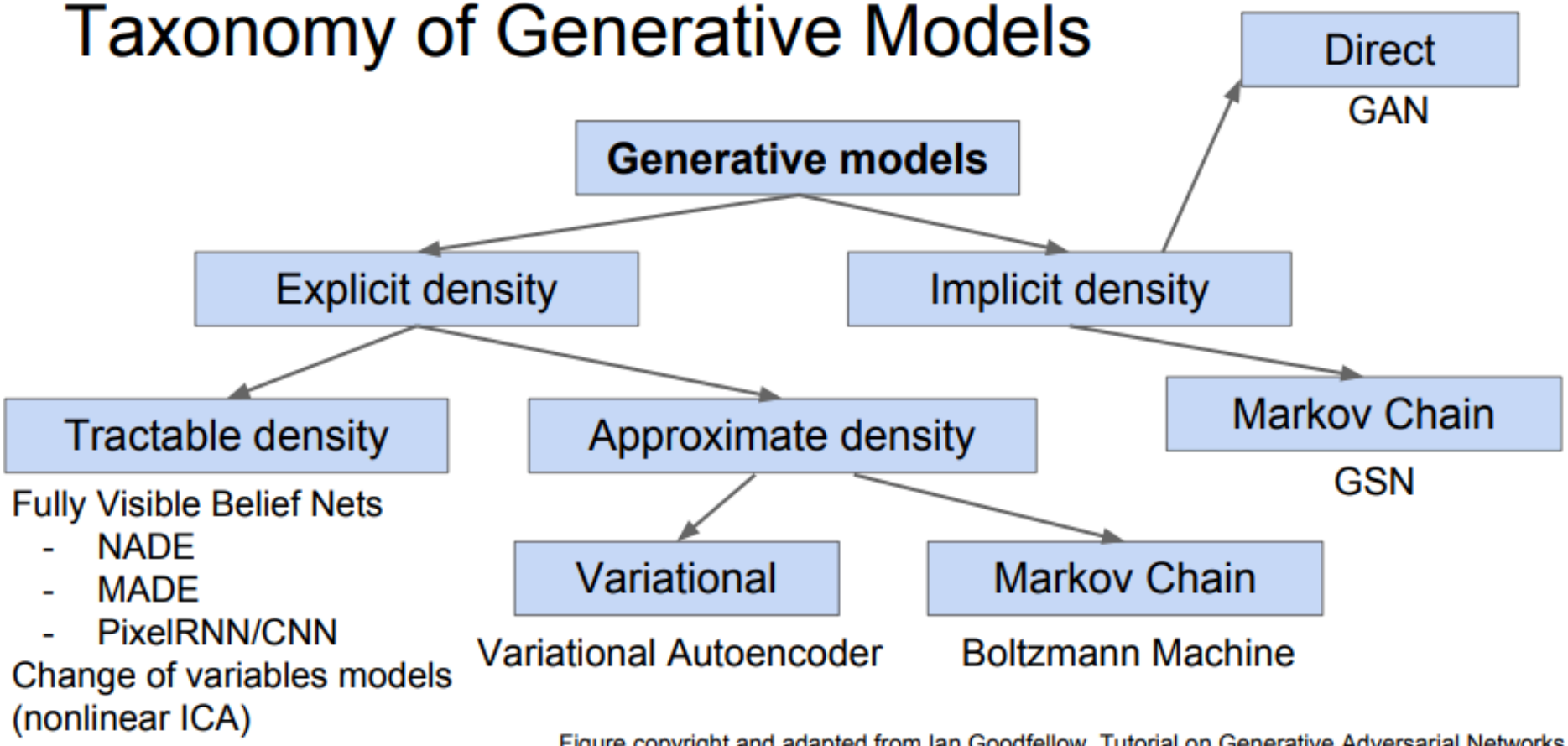
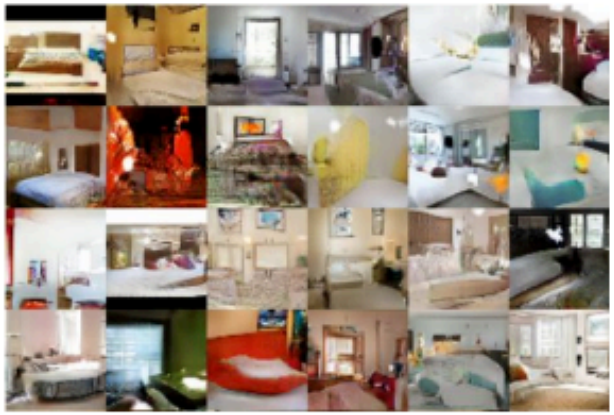
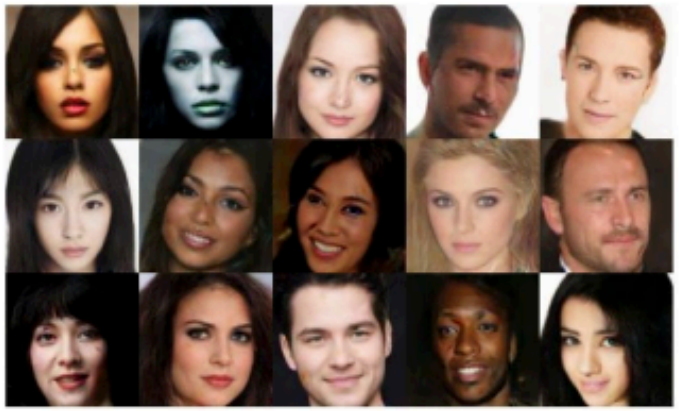


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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

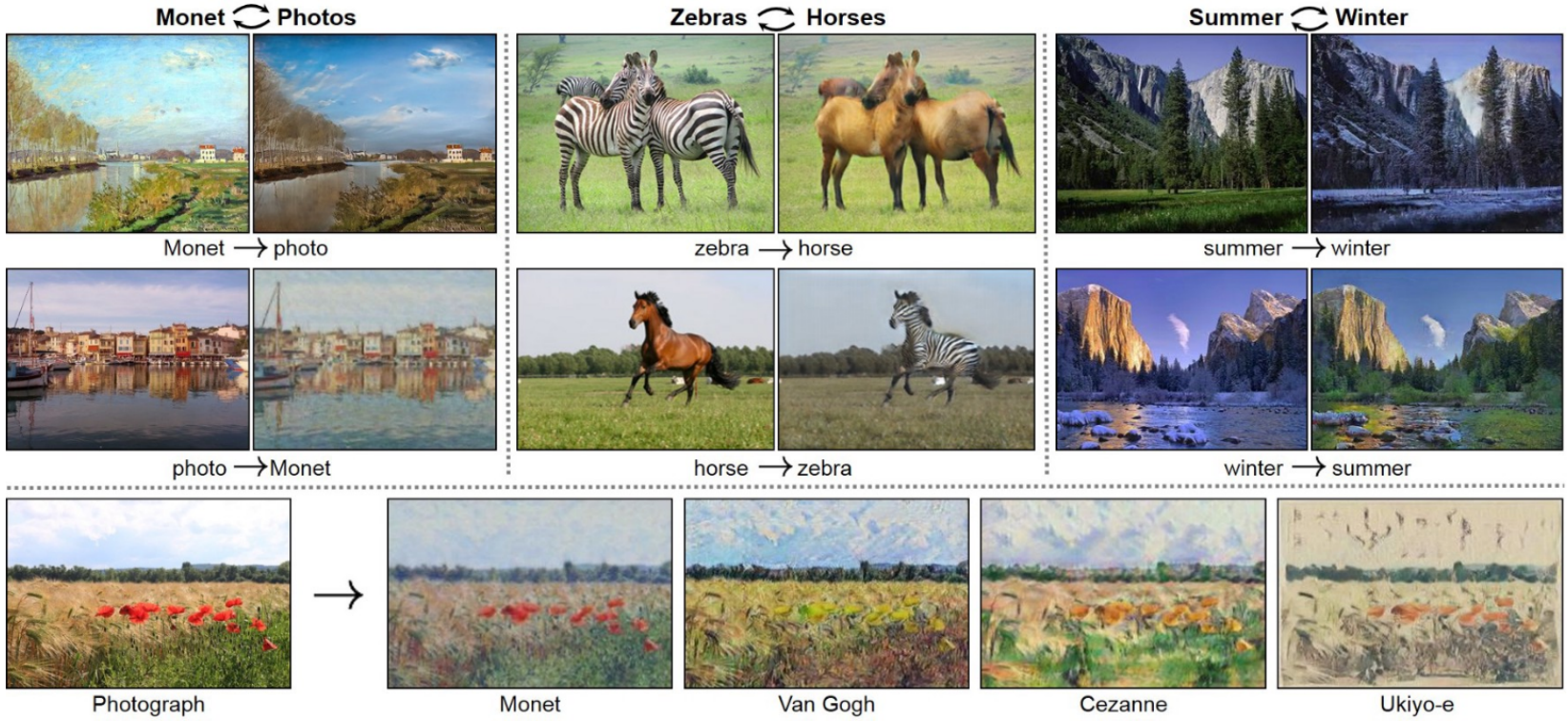


Image-to-Image Translation using GANs.

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Then maximize likelihood of training data

Fully visible belief network - PixelCNN

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

Variational Autoencoders (VAE)

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

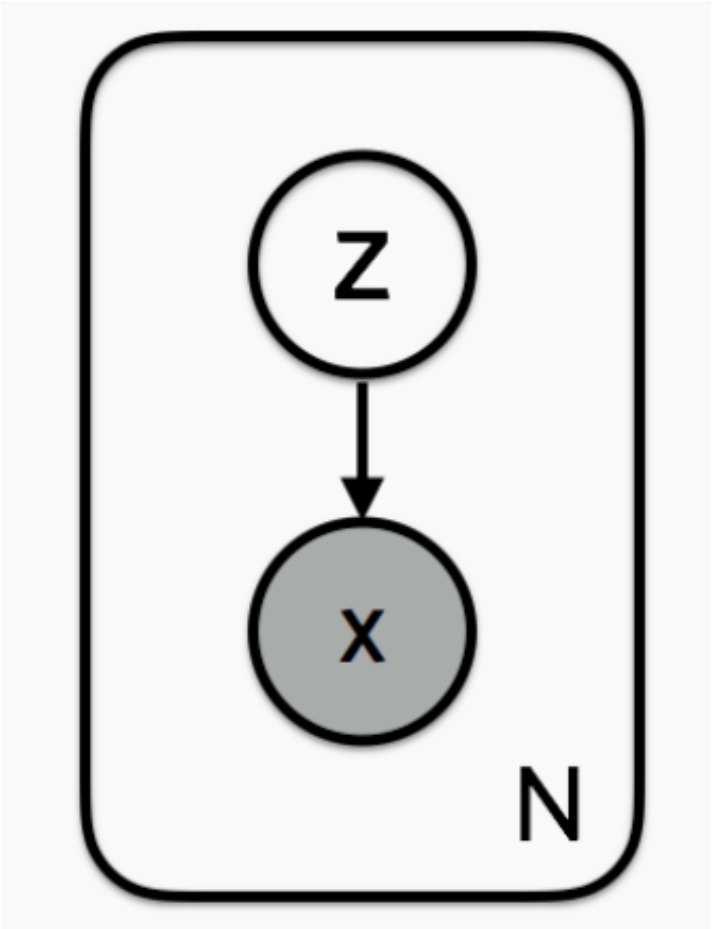
VAEs define intractable density function with latent **z**:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

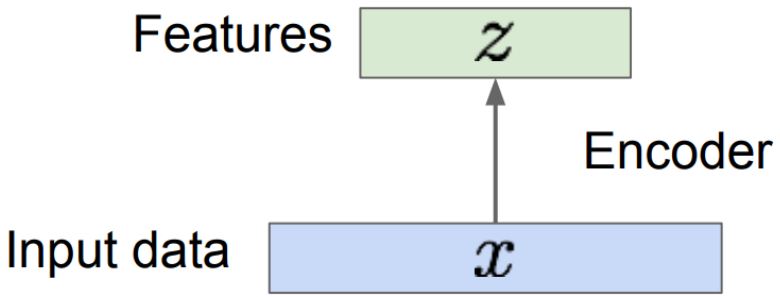
Variational Autoencoder

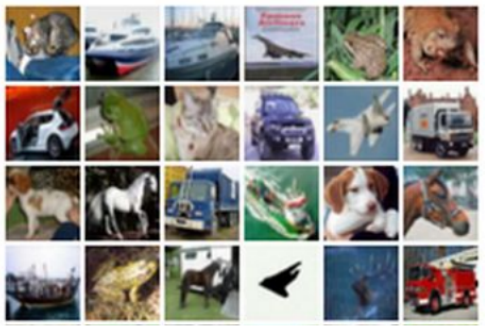
- Variational autoencoder contains a specific probability model of data *x* and latent variables *z*.
- The joint probability of the model: $p(x,z)=p(x|z)p(z)$.
- Follows generative process:
- For each datapoint *i*:
  - Draw latent variables $z_i \sim p(z)$
  - Draw datapoint $x_i \sim p(x \mid z)$
- The latent variables are drawn from a prior $p(z)$.
- The data *x* have a likelihood $p(x|z)$ that is conditioned on latent variables *z*.



Autoencoders

How to learn this feature representation?



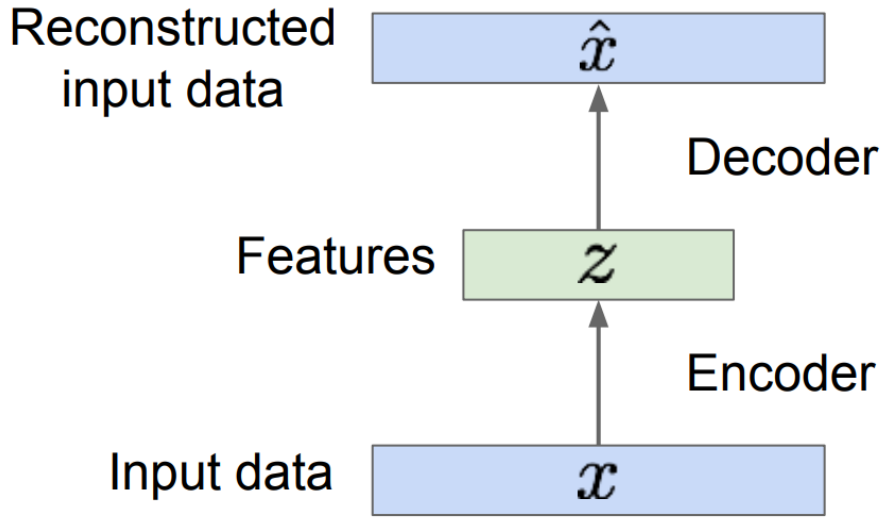


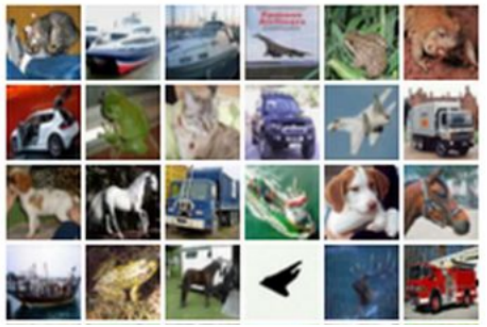
Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data

“Autoencoding” - encoding itself





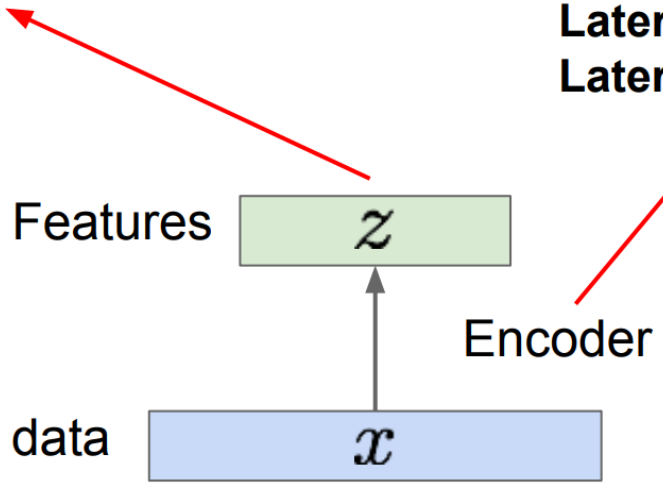
Autoencoders

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

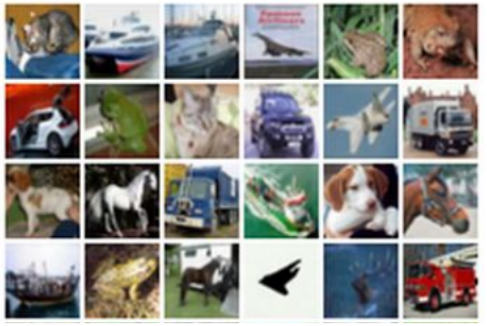
z usually smaller than **x** (dimensionality reduction)

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

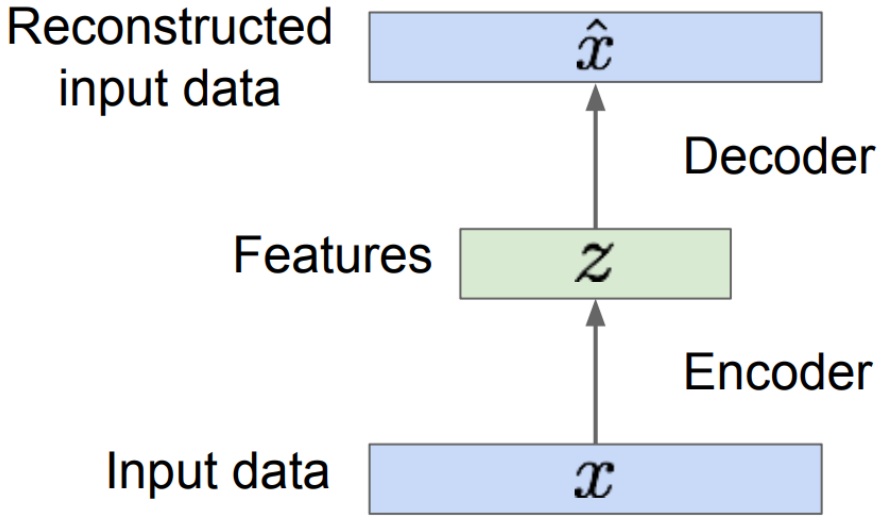


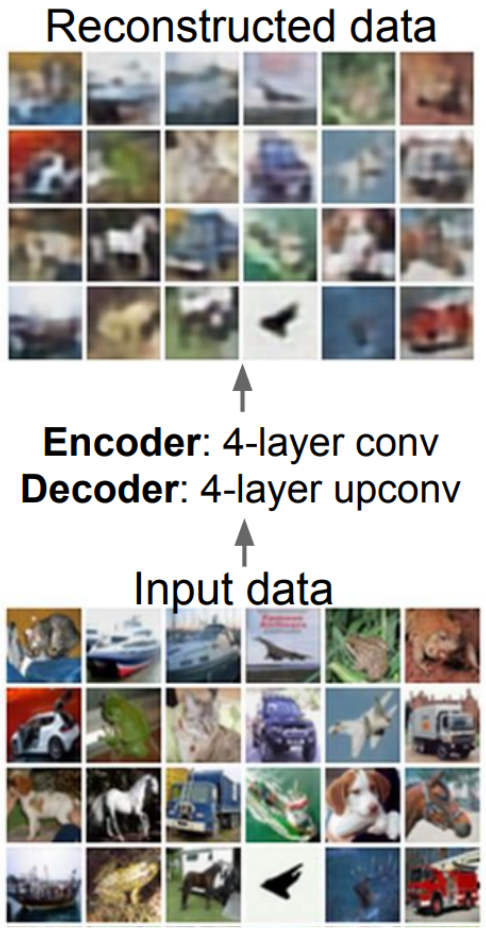
Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data

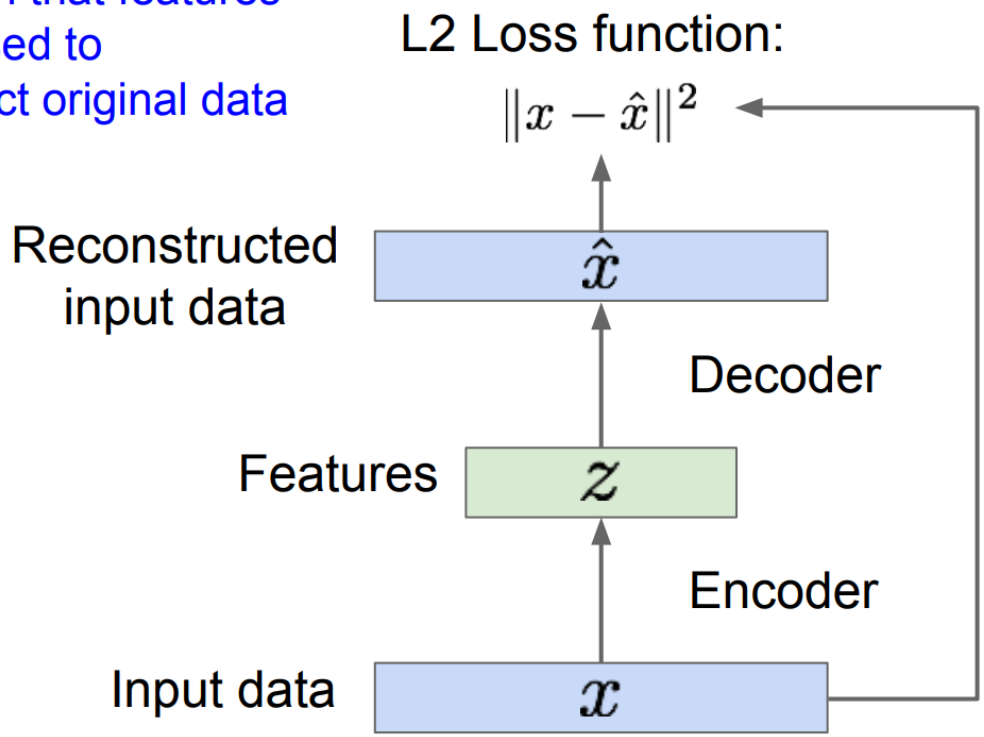
"Autoencoding" - encoding itself

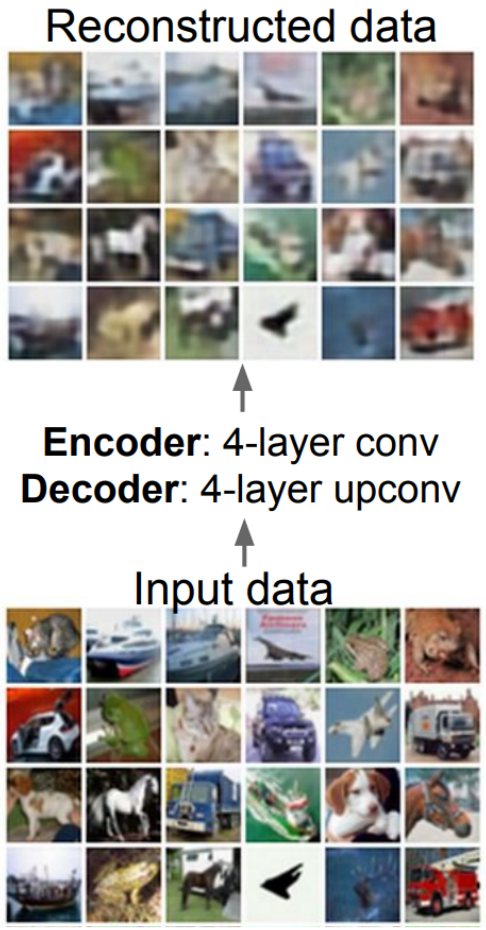




Autoencoders

Train such that features can be used to reconstruct original data

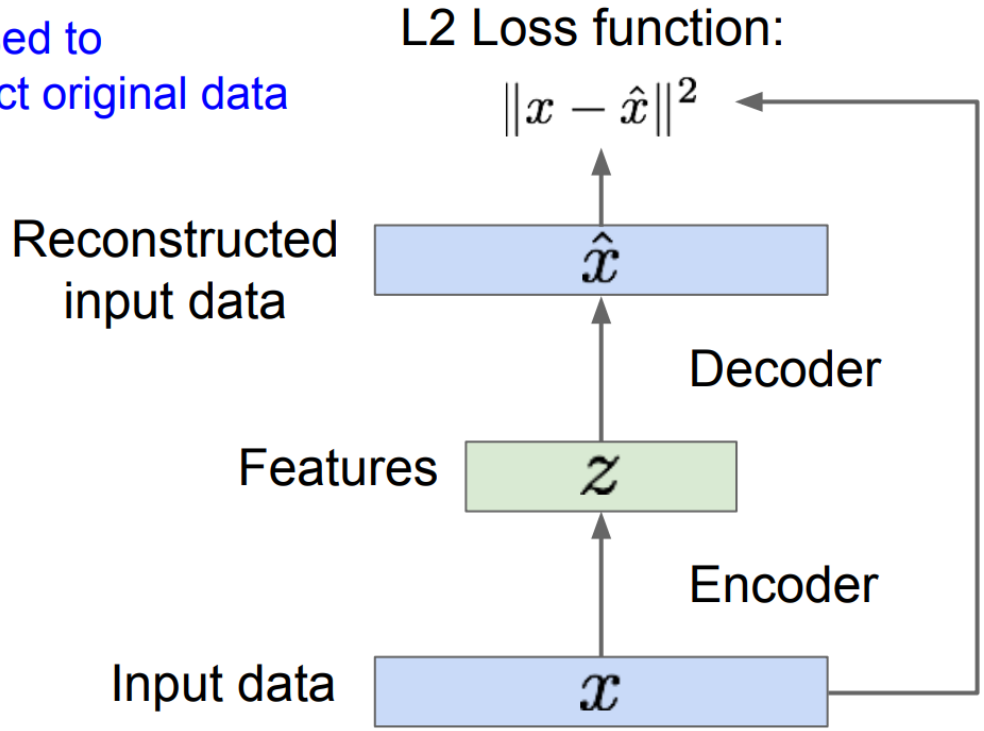


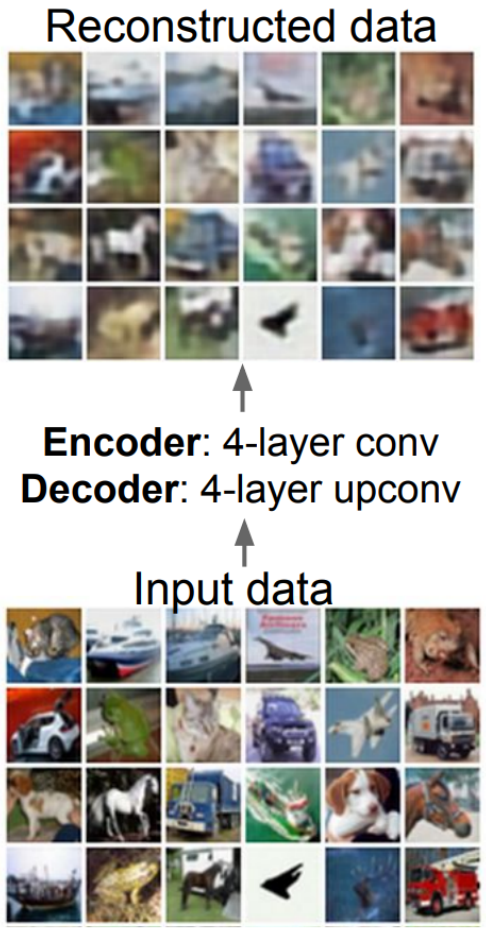


Autoencoders

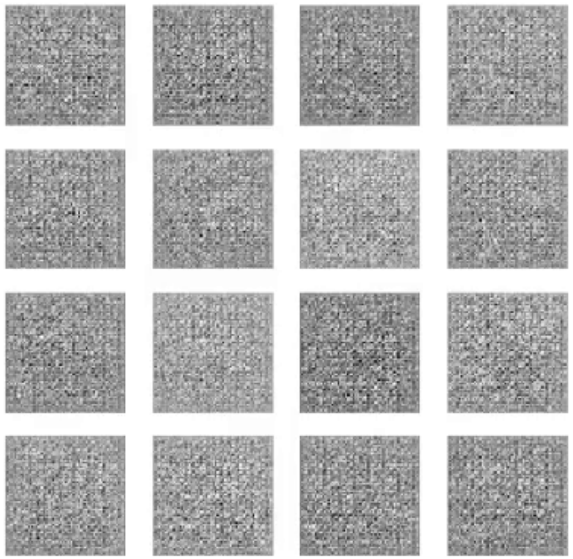
Train such that features can be used to reconstruct original data

Doesn't use labels!





GAN Models



Output of a GAN through time, learning to Create Hand-written digits.

GANs – Generative Adversarial Models

- Unsupervised – do not use labeled data
- Generative – learn joint probability *p(x, y)*
- Adversarial learning algorithm – minimax concept
- Generative models learn the intrinsic distribution function ***p(x,y)***, allowing them to generate both synthetic inputs ***x'*** and outputs/targets ***y'***, typically given some hidden parameters.

AI Generates Fake Celebrity Faces

- http://research.nvidia.com/publication/2017-10_Progressive-Growing-of

AI Learns Fashion Sense

- https://arxiv.org/pdf/1711.02231.pdf

Image to Image Translation using Cycle-Consistent Adversarial Neural Networks

- https://junyanz.github.io/CycleGAN/

AI Creates Modern Art

- https://www.technologyreview.com/s/608195/machine-creativity-beats-some-modern-art/

This Deep Learning AI Generated Thousands of Creepy Cat Pictures

- https://motherboard.vice.com/en_us/article/a3dn9j/this-deep-learning-ai-generated-thousands-of-creepy-cat-pictures

MIT is using AI to create pure horror

- https://qz.com/817604/mits-nightmare-machine-uses-artificial-intelligence-to-create-horrific-images-of-ghoulish-faces-and-scary-places/

Amazon's new algorithm designs clothing by analyzing a bunch of pictures

- https://www.theverge.com/2017/8/24/16195858/amazon-ai-fashion-designer

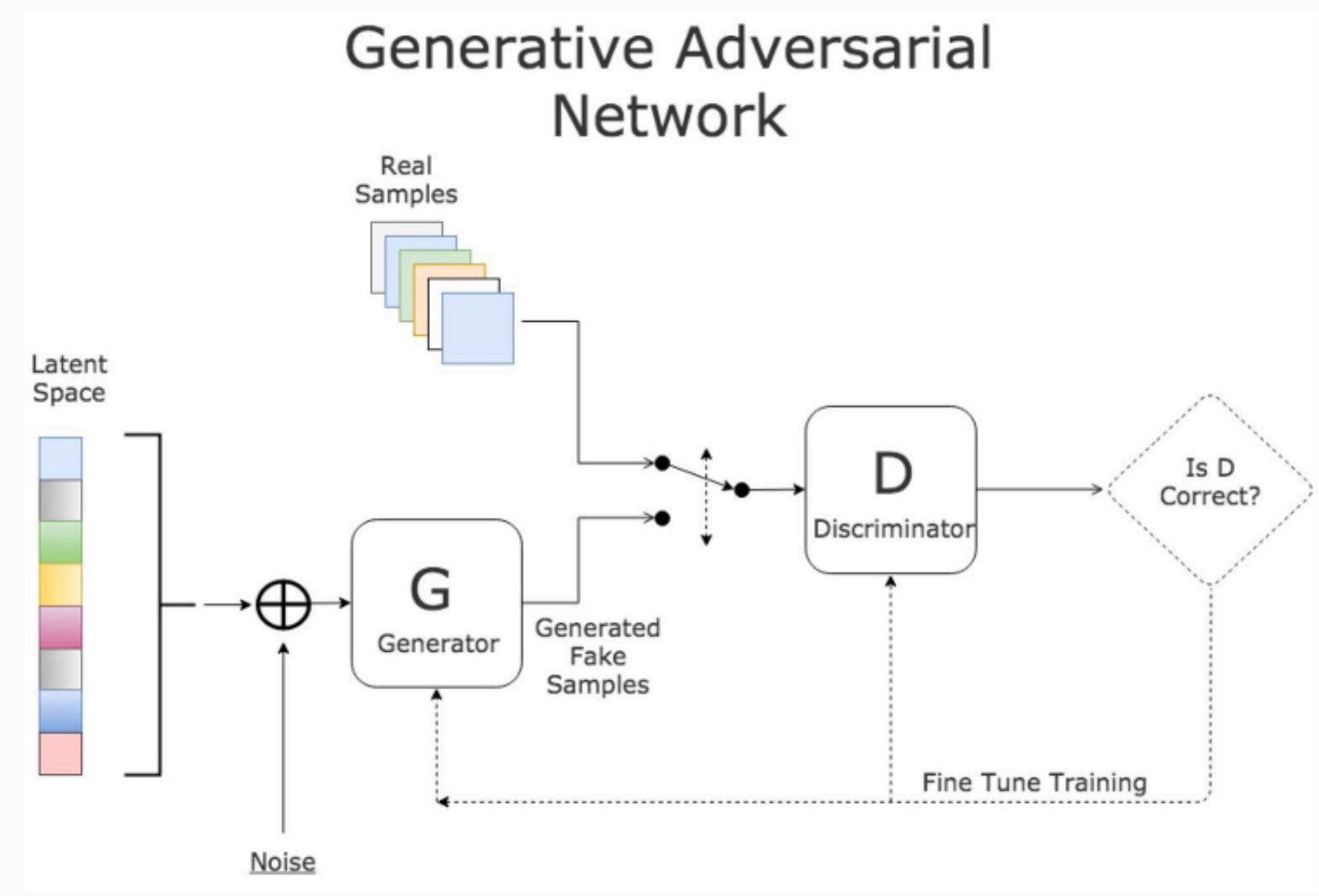
AI creates Photo-realistic Images

- https://www.inverse.com/article/39018-these-neutral-networks-can-generate-realistic-photos

Pros and Cons of GANs

- They currently generate the sharpest generated images
- They are easy to train (since no statistical inference is required), and only back-propagation is needed to obtain gradients

- GANs are difficult to optimize due to unstable training dynamics.
- No statistical inference can be done. GANs belong to the class of ***direct implicit*** density models; they model ***p(x)*** without explicitly defining the PDF.


Generative Adversarial Network
Real Samples
Latent Space
G
Generator
Generated Fake Samples
D
Discriminator
Is D Correct?
Noise
Fine Tune Training

Generative Adversarial Networks are composed of two models:

Generator

- Aims to generate new data similar to the expected one. The Generator could be thought of as forger, which creates fake works of art.

Discriminator

- Goal is to recognize if an input data is '*real*' — belongs to the original dataset — or if it is '*fake*' — generated by a forger. In this scenario, a Discriminator could be thought of as a person who identifies forgeries.

How do these models interact?

- The Generator and Discriminator are adversaries.
- The Generator (forger) needs to learn how to create data in such a way that the Discriminator isn’t able to distinguish it as fake anymore.
- The competition between the two is what improves their respective models.

Mathematical Model

- A neural network $G(z, \theta_1)$ is used to model the Generator.
- It's role is to map input noise variables **z** to the desired data space ***x*** (i.e., images).

- A second neural network $D(x, \theta_2)$ models the discriminator and outputs the probability that the data came from the real dataset**,** in the range (0,1).
- In both cases, θ_i represents the weights or parameters that define each neural network.

Mathematical Model

Generator

- Weight's are optimized to maximize the probability that any fake image is classified as belonging to the real dataset.
- **The loss/error function used for this network maximizes *D(G(z)).***

Discriminator

- Weight's are optimized to maximize the probability that any real data input *x* is classified as belonging to the real dataset, while minimizing the probability that any fake image is classified as belonging to the real dataset.
- **Maximizes the function *D(x)*, and it also minimizes *D(G(z)).***

MiniMax

- During training, the Discriminator and Generator are trying to optimize opposite loss functions.
- Can be thought of two agents playing a minimax game with value function **V(G,D).**
- In this minimax game, the generator is trying to maximize it's probability of having it's outputs recognized as real, while the discriminator is trying to minimize this same value.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Value Function of Minimax Game played by Generator and Discriminator

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  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 descending its stochastic gradient:

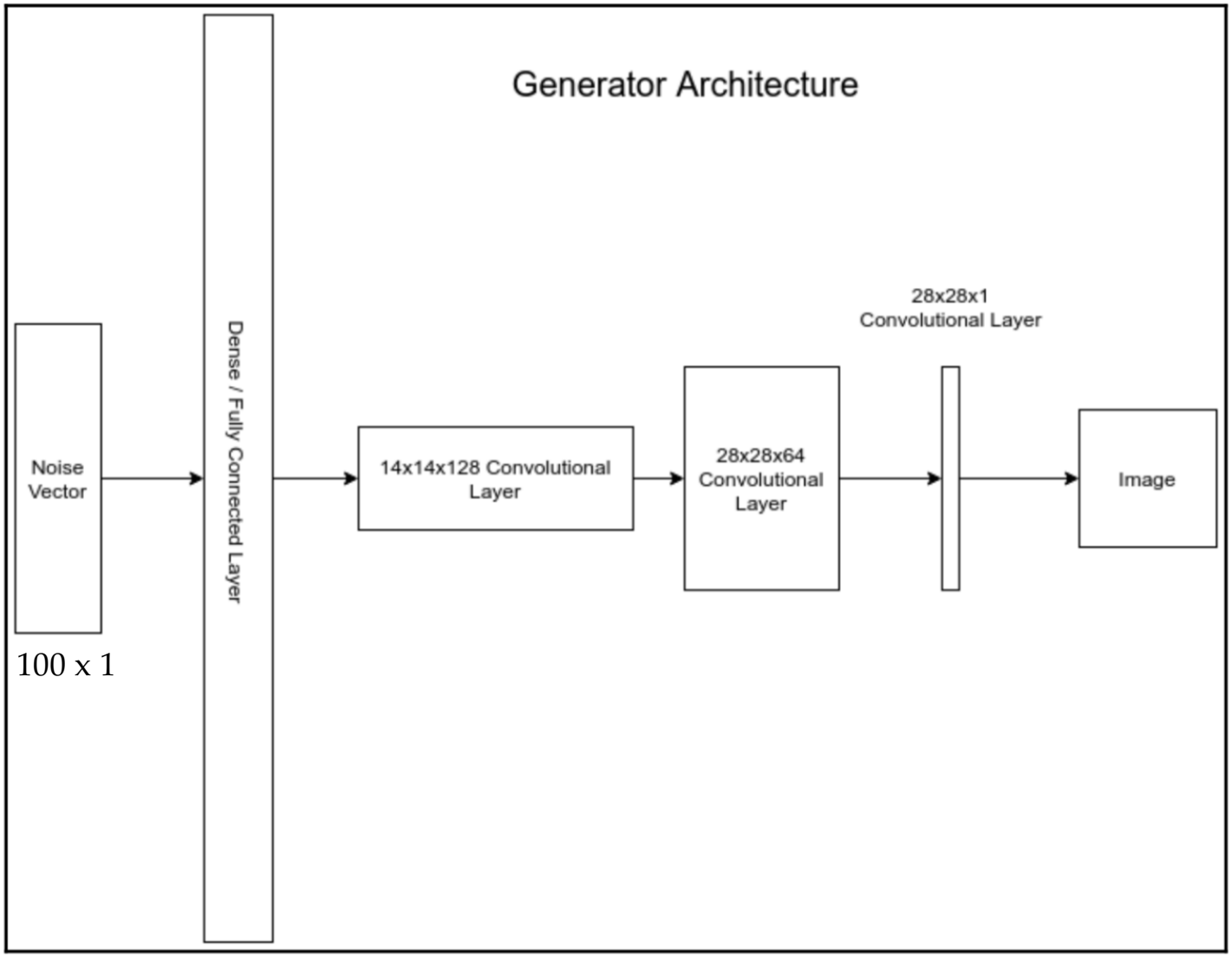
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

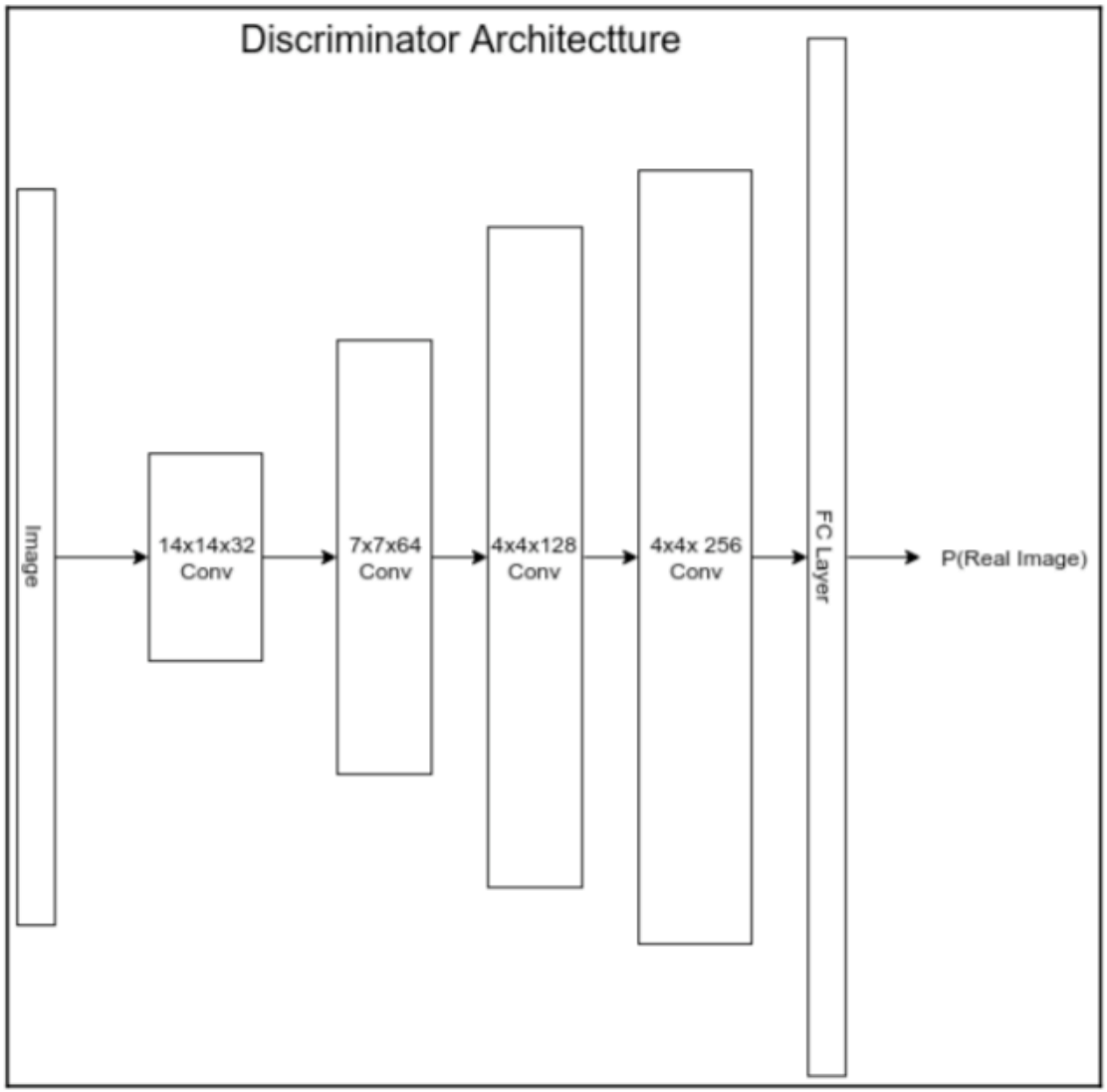
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

The fundamental steps to train a GAN can be described as following:

1. Sample a **noise** set and a **real-data** set, each with size *m.*
2. Train the **Discriminator** on this data.
3. Sample a different noise **subset** with size *m*.
4. Train the **Generator** on this data.
5. **Repeat** from Step 1.


Generator Architecture
Noise Vector
100 x 1
Dense / Fully Connected Layer
14x14x128 Convolutional Layer
28x28x64 Convolutional Layer
28x28x1 Convolutional Layer
Image


Discriminator Architectture
Image
14x14x32
Conv
7x7x64
Conv
4x4x128
Conv
4x4x 256
Conv
FC Layer
P(Real Image)

- https://blog.keras.io/building-autoencoders-in-keras.html

Go to demo