Deep Learning for Medical Image Analysis

COMP5423

Hao CHEN

Dept. of CSE,CBE&LIFS, HKUST

jhc@cse.ust.hk

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING 計算機科學及工程學系

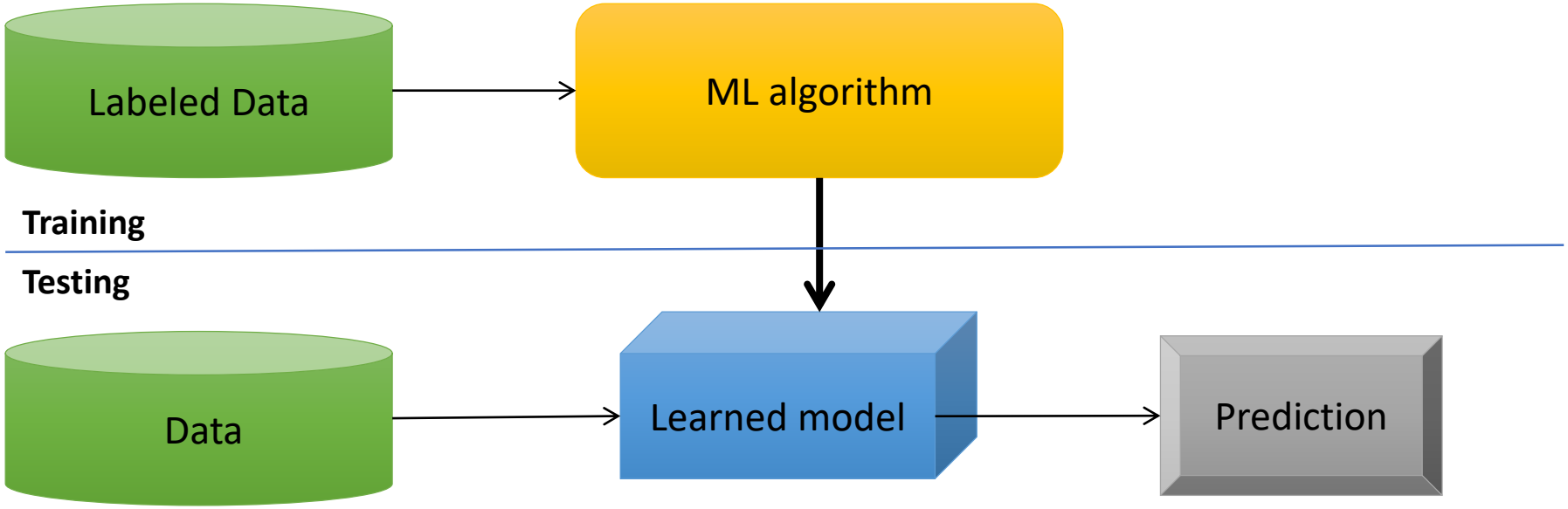
香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Fundamentals of Deep Learning

- Machine learning basics
- Deep learning
- Regularization for deep learning
- Optimization for deep learning
- Advanced deep learning models

Machine Learning Basics

- Machine learning (ML) is a field of computer science that gives computers the ability to learn without being explicitly programmed.
- Methods that can learn from and make predictions on data.
- It usually has some data pre-processing and feature design work.



Supervised Learning

- Consider an unknown joint probability distribution $p_{X,Y}$ and assume training data $(x_i, y_i) \sim p_{X,Y}$, with $x_i \in X,\ y_i \in Y,\ i = 1, \dots, N$.
- In most cases, x_i is a vector of features, and y_i is a scalar (e.g., a category or a real value).
- The training data is generated i.i.d.
- For any new (x, y), inference is to estimate the conditional: $p(Y = y | X = x)$
- Target tasks can be classification, regression, 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.

Unsupervised Learning

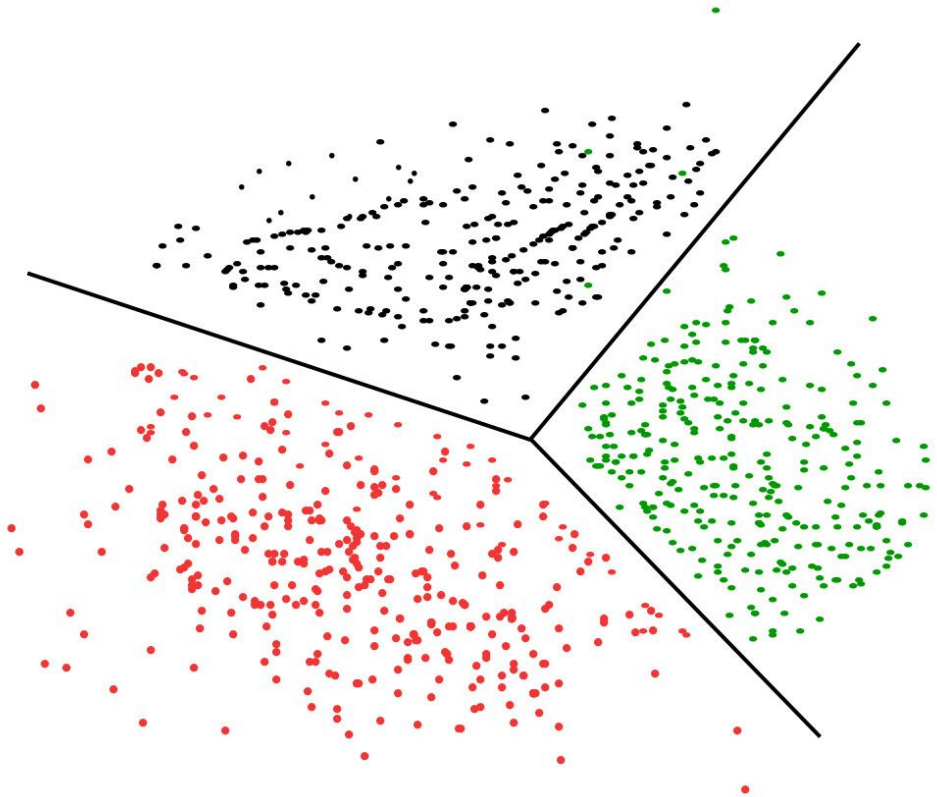
Data: x

Just data, no labels!

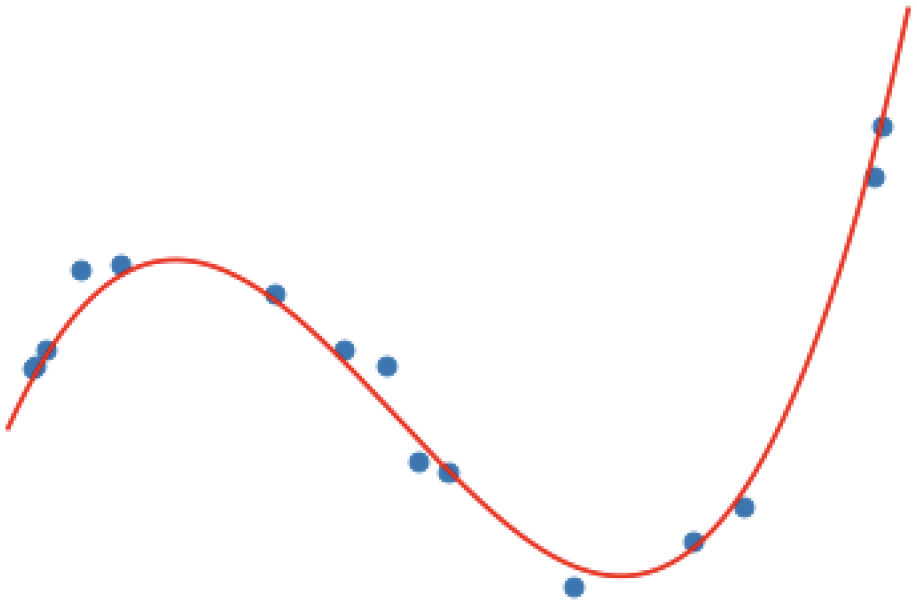
Goal: Learn some underlying hidden structure of the data

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

Intuitive Understanding



Classification and clustering consist in identifying a decision boundary between samples of distinct classes/clusters.



In regression, the computer program is asked to predict a numerical value given some input.

Empirical Risk Minimization

- Consider a function $f: X \rightarrow Y$ generated by a machine learning algorithm.
- The predictions of this function can be evaluated via a **loss function** $\mathcal{L}\big(y, f(x)\big) \geq 0$, such that $\mathcal{L}\big(y, f(x)\big)$ measures how close the prediction $f(x)$ from y is.

- Examples of loss functions:

  Classification: $\mathcal{L}\big(y, f(x)\big) = -\sum y log f(x)$

  Regression: $\mathcal{L}\big(y, f(x)\big) = \sum \|y - f(x)\|^2$

Empirical Risk Minimization

- Let $\mathcal{F}$ denote the set of all functions f (**hypothesis space**) that can be produced by the chosen learning algorithm.
- We are looking for a function $f \in \mathcal{F}$ with a small expected risk:

$$R(f) = \mathbb{E}_{(x,y)\sim p_{X,Y}}[\mathcal{L}(y, f(x))]$$

$$f_* = \arg\min_{f\in\mathcal{F}} R(f)$$

Empirical Risk Minimization

- If we have i.i.d. training data $d = \{(x_i, y_i) | i = 1, \dots, N\}$, we can compute an estimate, the empirical risk (or **training error**)

$$\hat{R}(f, d) = \frac{1}{N} \sum_{(x_i, y_i) \in d} \mathcal{L}\big(y_i, f(x_i)\big)$$

- This results into the empirical risk minimization principle:

$$f_*^d = \arg\min_{f \in \mathcal{F}} \hat{R}(f, d)$$

- Most machine learning algorithms, including neural networks, implement empirical risk minimization.

Capacity, Underfitting and Overfitting

- The **capacity** of a hypothesis space induced by a learning algorithm intuitively represents the ability to find a good model $f \in \mathcal{F}$ for any functions.
- ML algorithm must perform well on **new, previously unseen** inputs. The ability to perform well on previously unobserved inputs is called **generalization**.
- The factors determining how well a ML algorithm will perform are its ability to:

1. Make the training error small and
2. Make the gap between training and test error small.

Deep Learning. MIT Press.

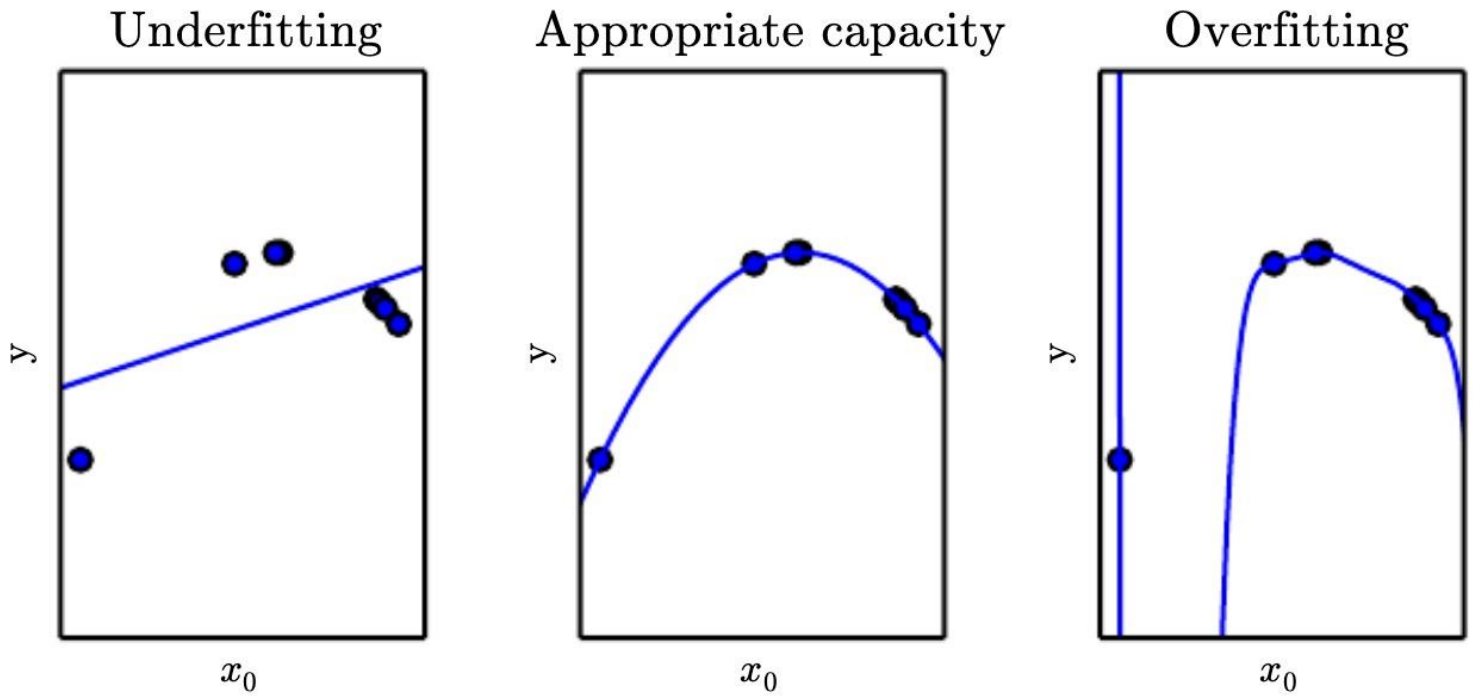
Capacity, Underfitting and Overfitting

- **Underfitting** occurs when the model is not able to obtain a sufficiently low error value on the training set.
- **Overfitting** occurs when the gap between the training error and test error is too large.

- We can control whether a model is more likely to overfit or underfit by altering its capacity.
- One way to control the capacity of a learning algorithm is by choosing its hypothesis space.

Capacity, Underfitting and Overfitting

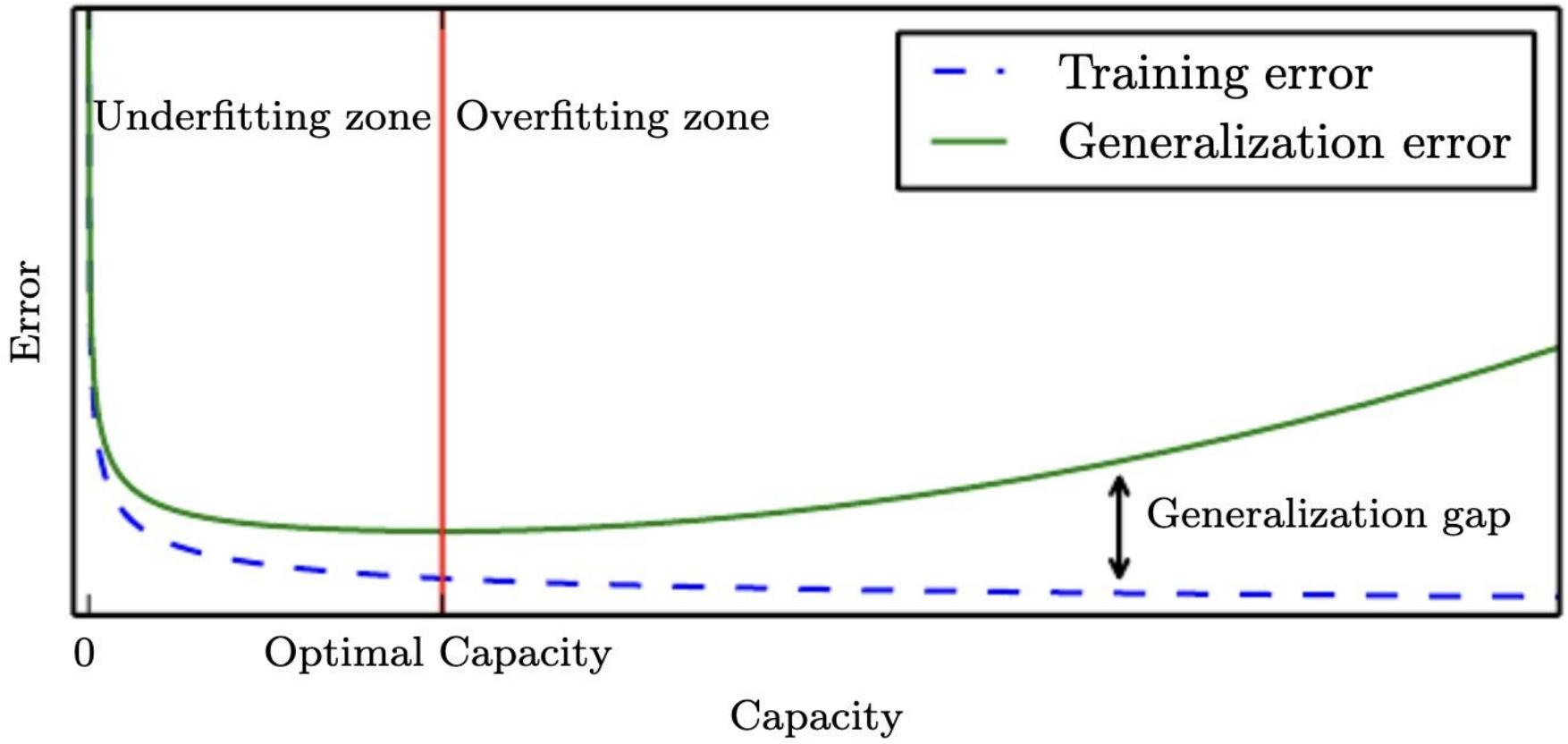
- In this figure, we fit three models to an example training set:



Deep Learning. MIT Press.

Capacity, Underfitting and Overfitting

- Typical relationship between capacity and error. Training and test errors behave differently.



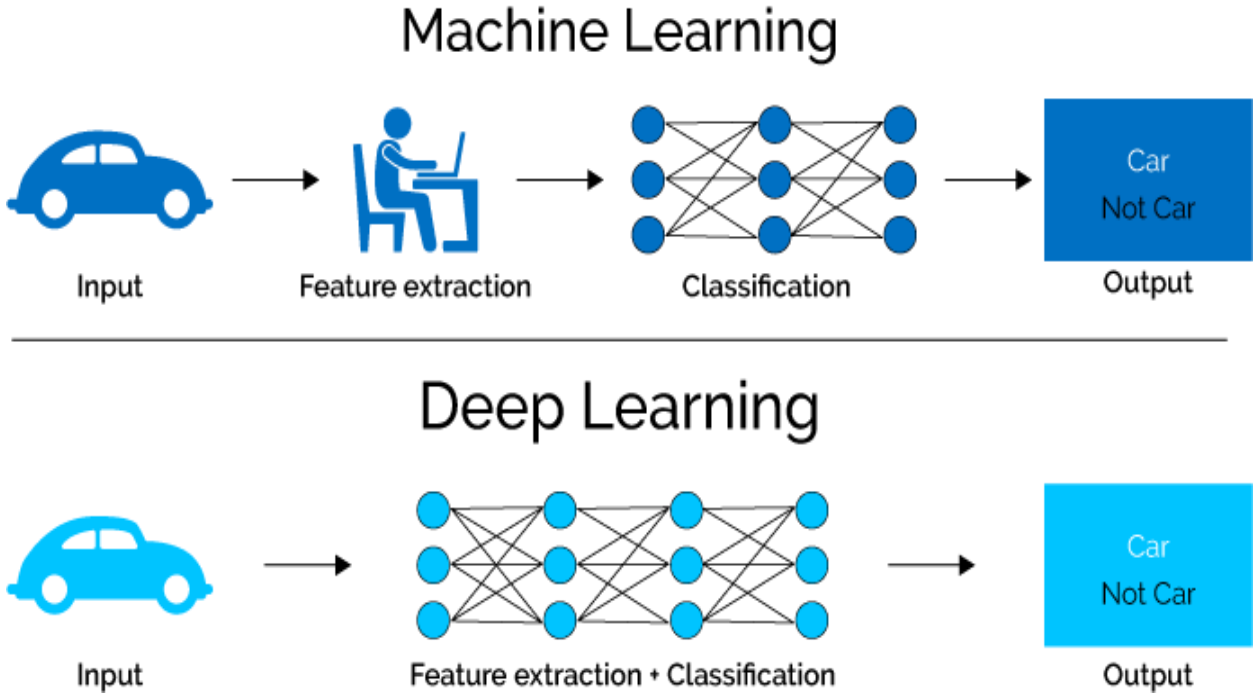
Deep Learning. MIT Press.

Regularization

- **Regularization** is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.
- It discourages learning a more complex model, so as to avoid the risk of overfitting.
- Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization.
- Typical regularization methods include weight decay, early stopping, etc.

Deep Learning. MIT Press.

What is Deep Learning (DL) ?

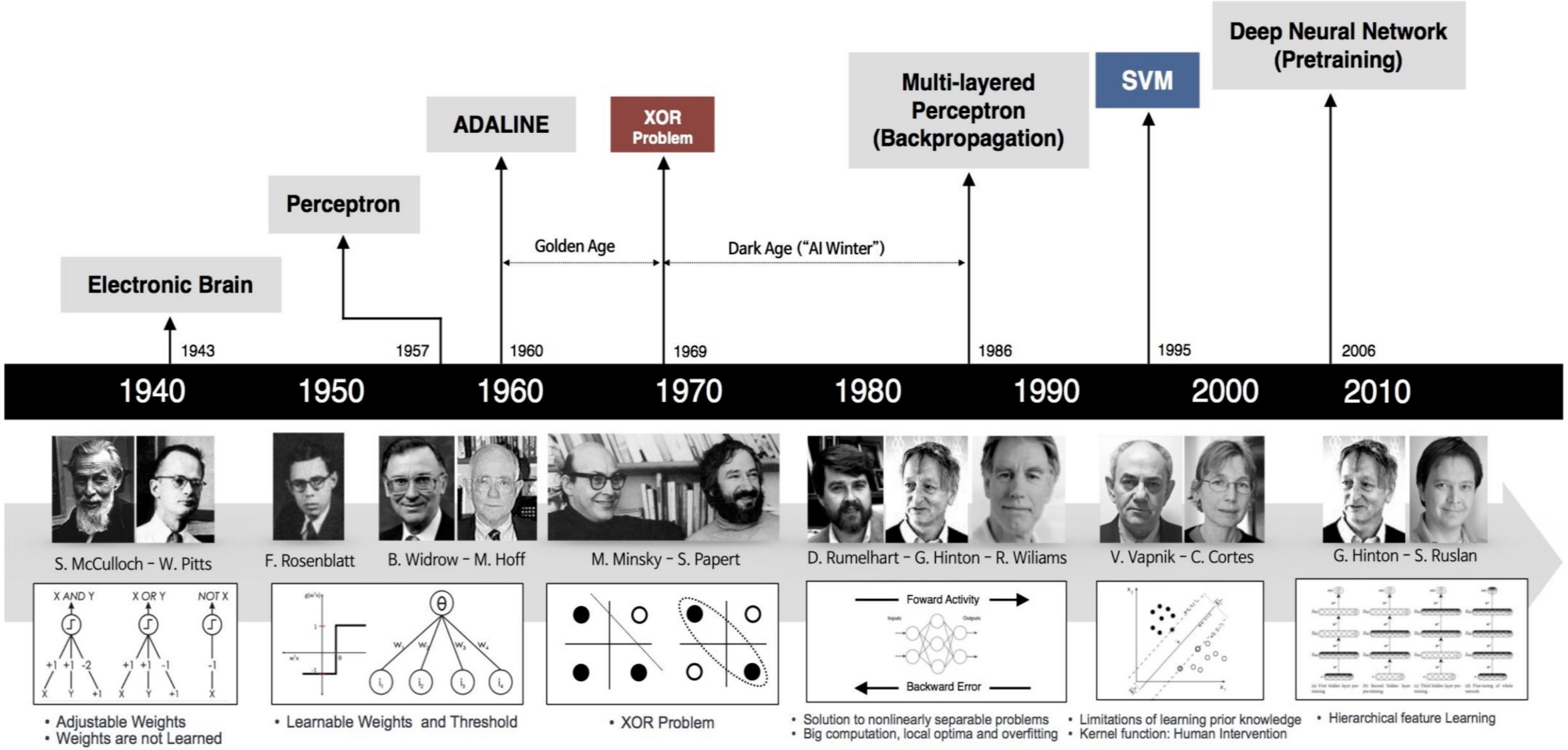


- In ML, manually designed features are often over-specified, incomplete and take a long time to design as well as validate.
- DL attempts to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data.

Neural network is an old idea

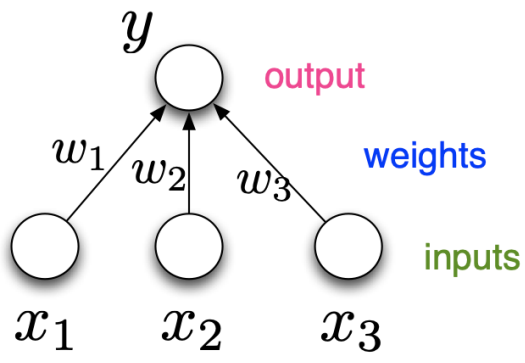
- One of the very first ideas in machine learning and artificial intelligence
- Date back to 1940s
- Many cycles of boom and bust
- Repeated promises of "true AI" that were unfulfilled followed by "AI winters"

Historical Milestones



Picture credit: Barbara Fusinska.

Multilayer Perceptron



$$y = g\left(b + \sum_i x_i w_i\right)$$

output, bias, i'th weight, nonlinearity, i'th input

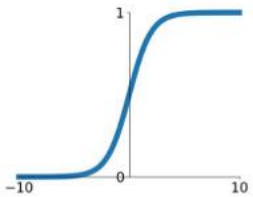
- The artificial neuron receives one or more inputs and sums them to produce an output.
- Usually each input is separately weighted, and the sum is passed through a non-linear function known as an **activation function**.
- These units are much more powerful if connecting many of them into a neural network.

Multilayer Perceptron

- Some activation functions:

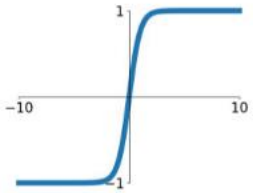
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



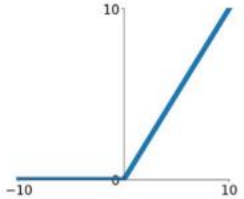
tanh

$\tanh(x)$



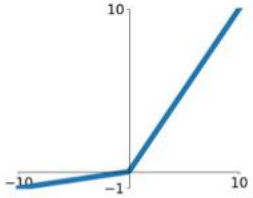
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

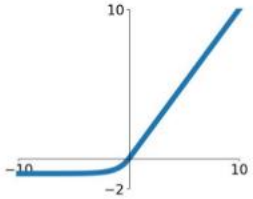


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

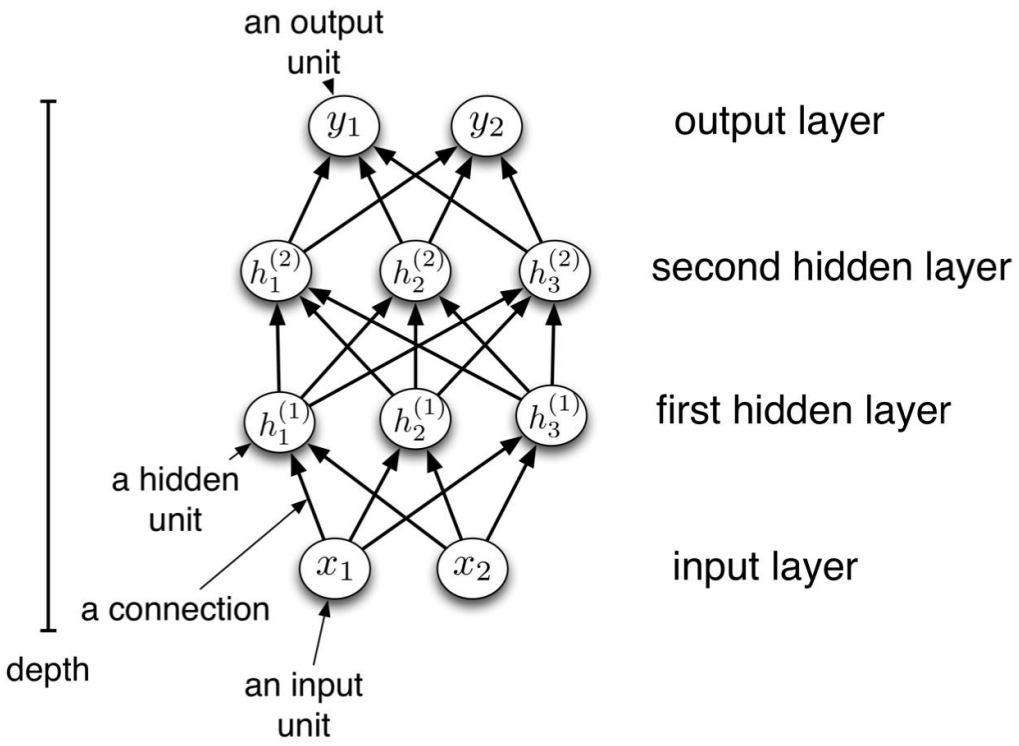
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Multilayer Perceptron

- We can connect lots of units together into a **directed acyclic graph**.
- This gives a feed-forward neural network, also called **multilayer perceptron** (MLP).
- Typically, units are grouped together into layers.
- A MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer.

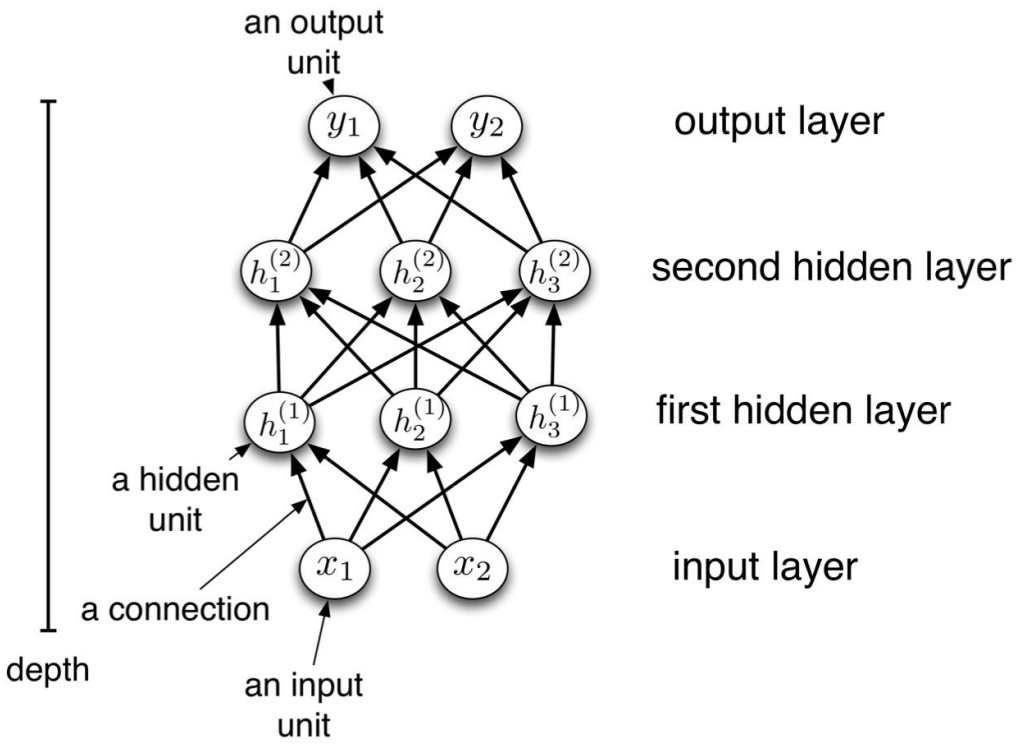


Multilayer Perceptron

- Each layer connects N input units to M output units.
- In the simplest case, all input units are connected to all output units. We call this a **fully connected layer**.

- The output units are a function of the input units:

$$y = f(x) = \phi(Wx + b)$$



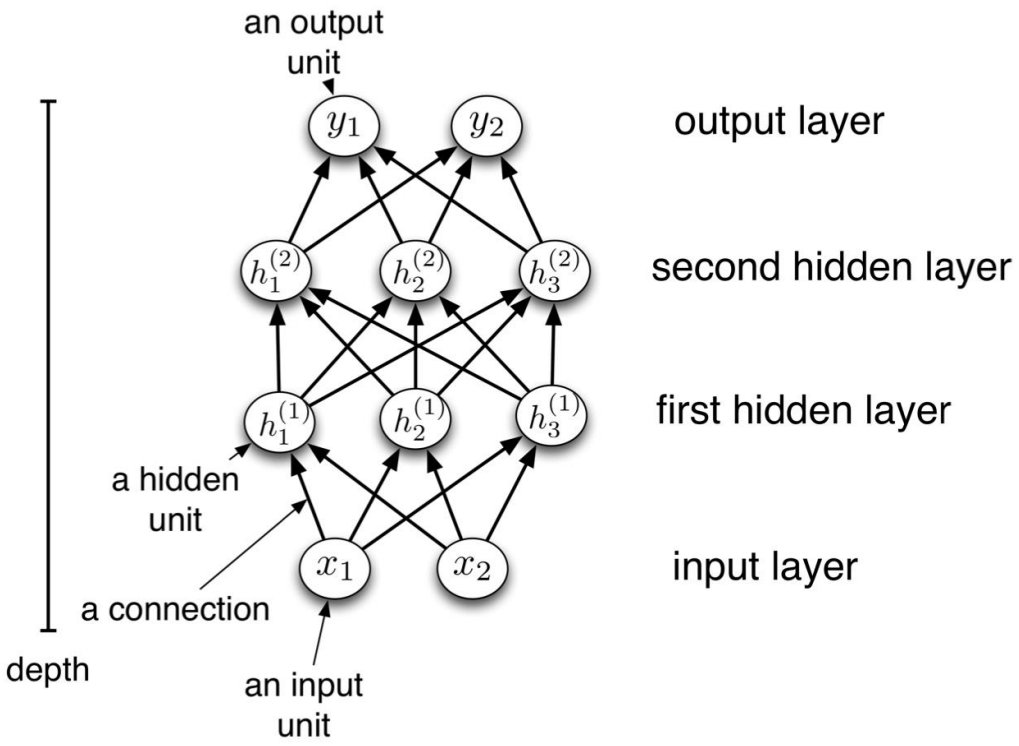
Multilayer Perceptron

- Each layer computes a function, so the network computes a composition of functions:

$$h^{(1)} = f^{(1)}(x)$$
$$h^{(2)} = f^{(2)}\left(h^{(1)}\right)$$
$$h^{(3)} = f^{(3)}\left(h^{(2)}\right)$$
$$y = f^{(L)} \circ \cdots \circ f^{(1)}(x)$$

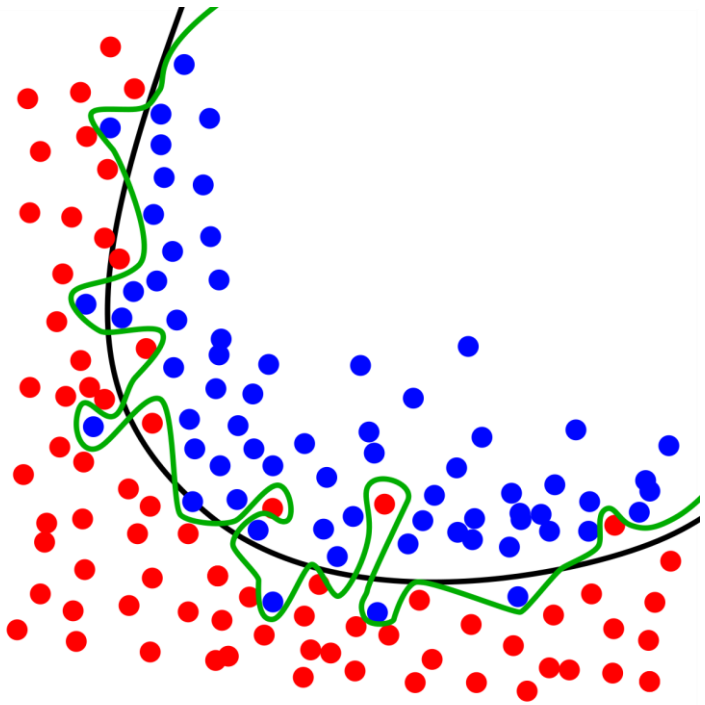
- Finally, the loss function will be:

$$\mathcal{L}\left(y_i, f^{(L)} \circ \cdots \circ f^{(1)}(x_i)\right)$$



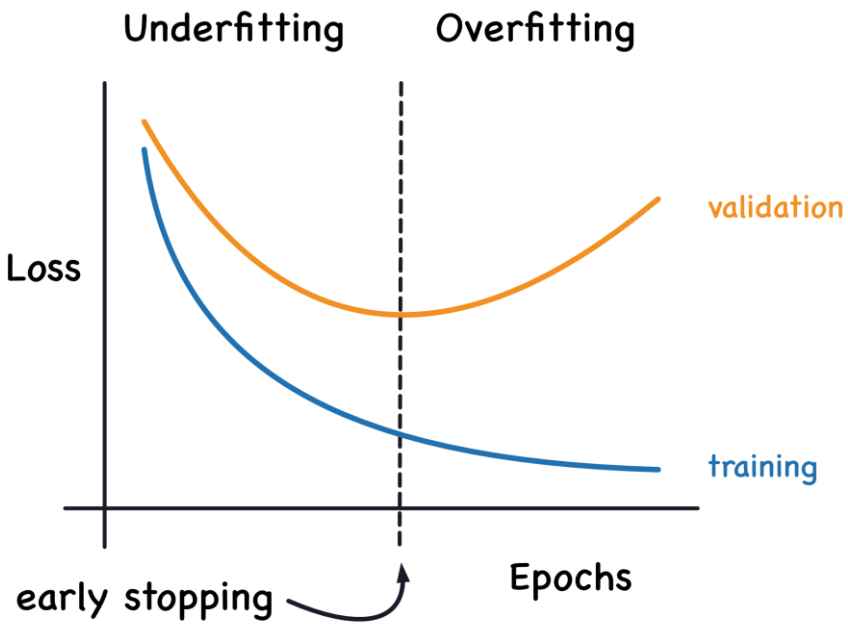
Regularization for DL

- For underfitting, it could be solved by increasing the model complexity.
- For overfitting, we could prevent it via following:
  - Larger data set;
  - Parameter norm penalty (e.g., L2 and L1 norm)
  - Others: data augmentation, noise robustness, model ensemble, early stopping, dropout, adversarial training, etc.



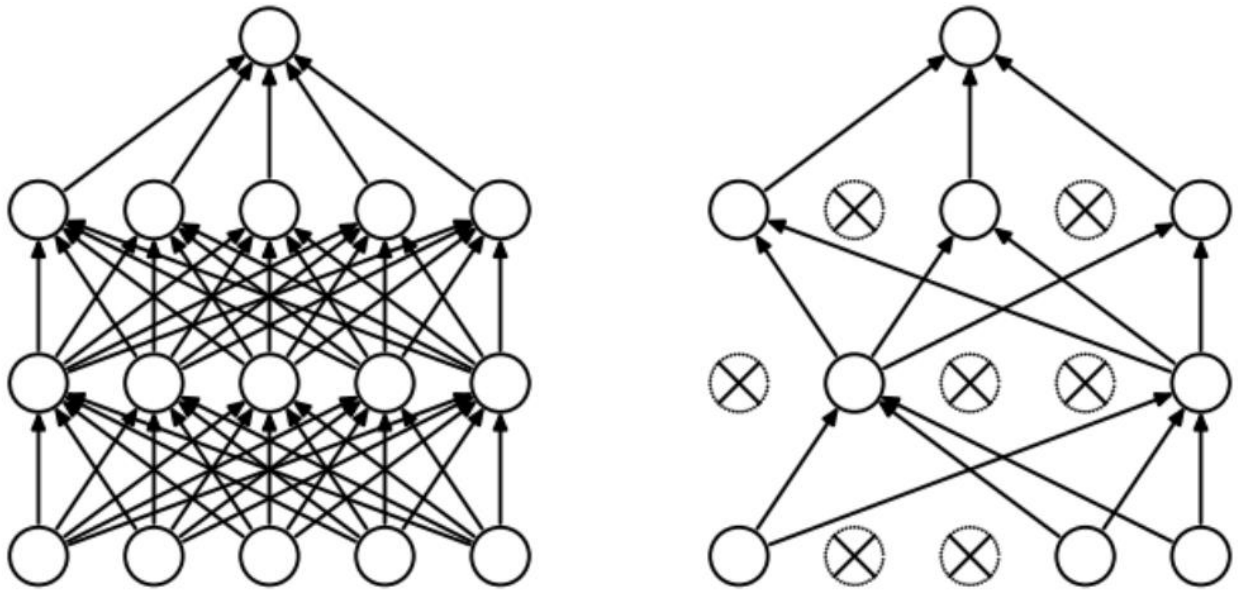
Regularization for DL

- Early Stopping: A method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold-out validation dataset.



Regularization for DL

- Dropout: The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much.



(a) Standard Neural Net (b) After applying dropout.

Srivastava, et al. Dropout: a simple way to prevent neural networks from overfitting. JMLR 2014.

Regularization for DL

- Adversarial training: training on adversarially perturbed examples from the training set. Human cannot tell the difference between the original example and the **adversarial example**, but the network can make highly different predictions.

$$\boldsymbol{x} \quad + .007 \times \quad \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \quad = \quad \boldsymbol{x} + \epsilon\,\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

$y =$"panda" w/ 57.7% confidence	"nematode" w/ 8.2% confidence	"gibbon" w/ 99.3% confidence

A demonstration of adversarial example generation applied to GoogLeNet on ImageNet.

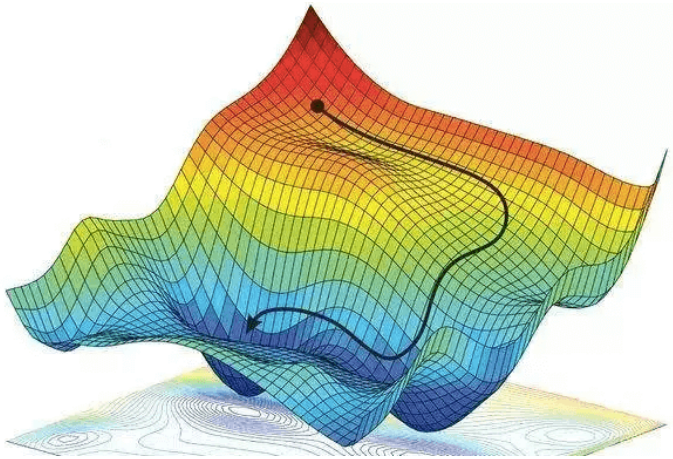
Deep Learning. MIT Press.

Optimization for DL

- **Stochastic gradient descent** (SGD) is the most used algorithm when training deep models, which update the parameters based on the mini-batch samples.
- **Backpropagation** is the central algorithm for deep learning training, which is an algorithm for computing gradients iteratively.
- Other optimization tricks include momentum, adaptive learning rates, batch normalization, etc.

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

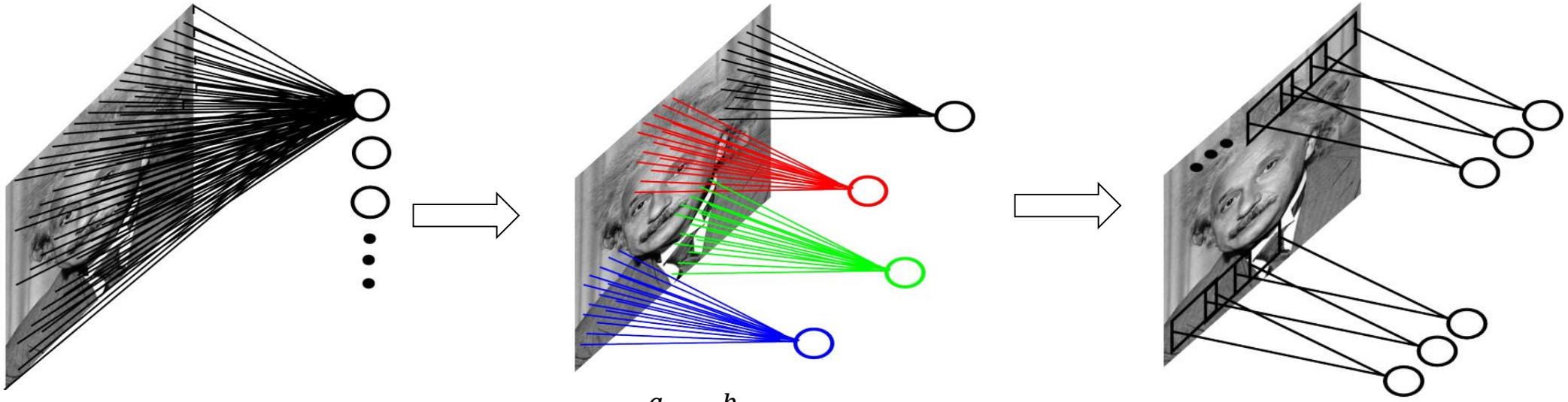
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$



Deep Learning. MIT Press.

Convolution Neural Network

- **Convolutional filters** to capture patterns in the input space instead of fully connection.
- Shared convolution parameters across different locations instead of only focusing on local dependencies.
- Convolution filters will be learned during training.

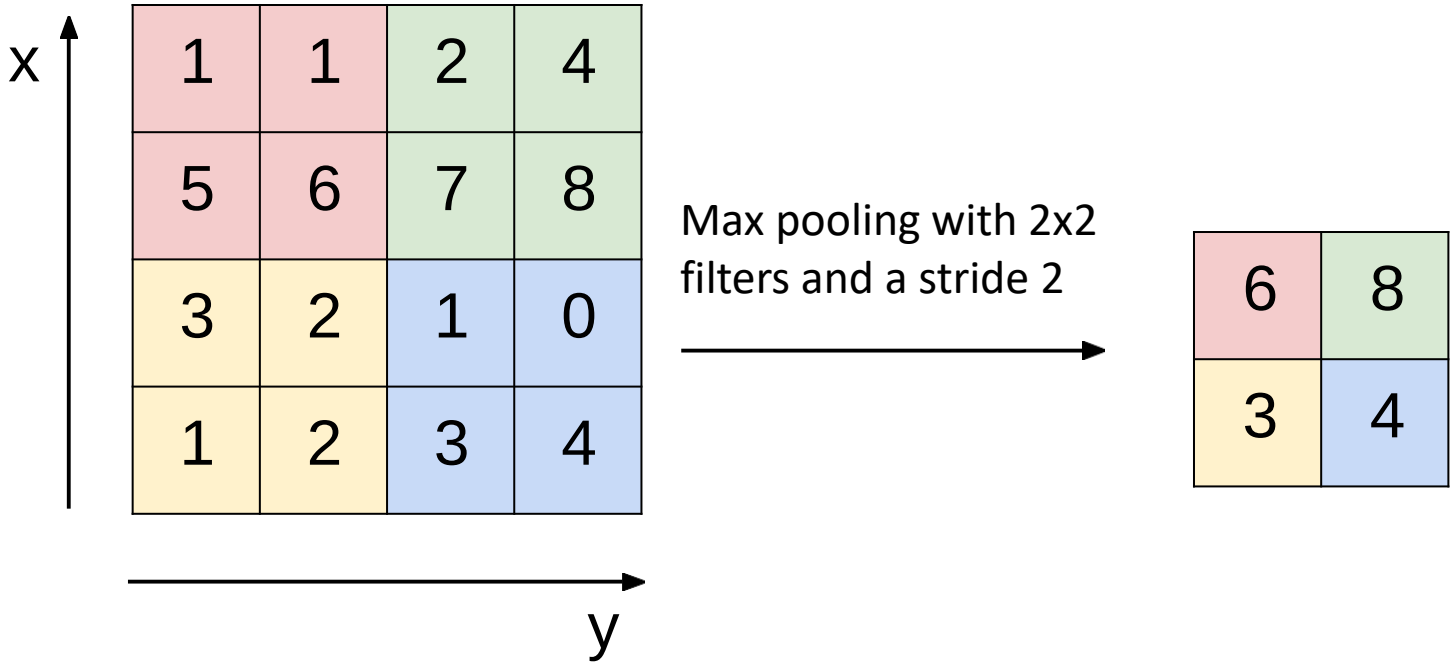


$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Convolution Neural Network

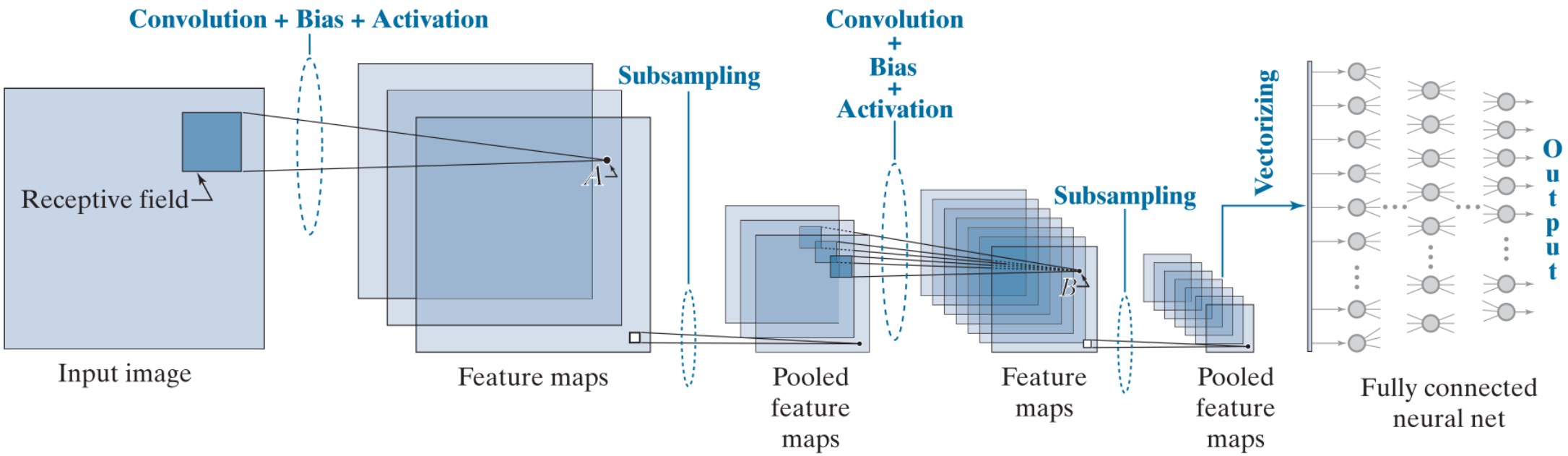
- Subsampling (or pooling): a reduction in spatial resolution for achieving translational invariance.
- Common pooling methods:

Average pooling, max-pooling, and L_2 pooling.



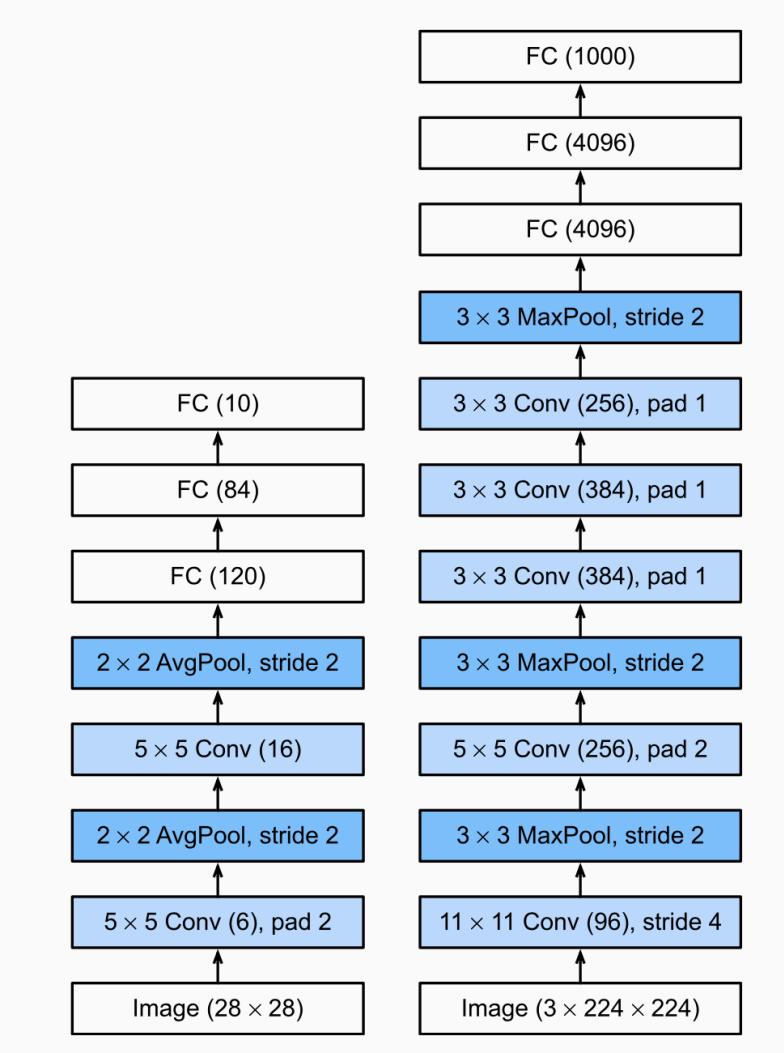
Convolution Neural Network

- LeNet Architecture



Digital Image Processing, by Gonzalez and Woods, Pearson, 2018.

Deep CNN (AlexNet)

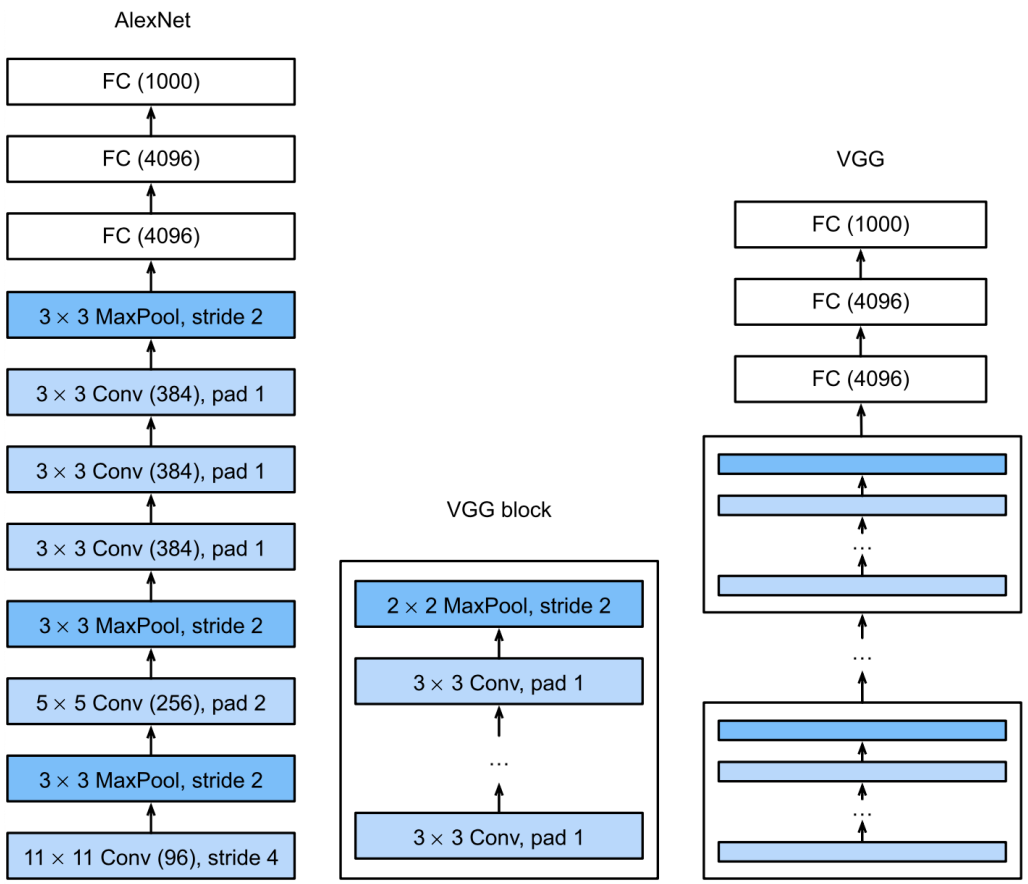


From LeNet (left) to AlexNet (right).

- AlexNet has a similar structure to that of LeNet, but uses more convolutional layers and a larger parameter space to fit the large-scale ImageNet dataset.
- Today AlexNet has been surpassed by much more effective architectures but it is a key step from **shallow** to **deep** networks that are used nowadays.

https://d2l.ai/chapter_convolutional-modern/

Networks Using Blocks (VGG)

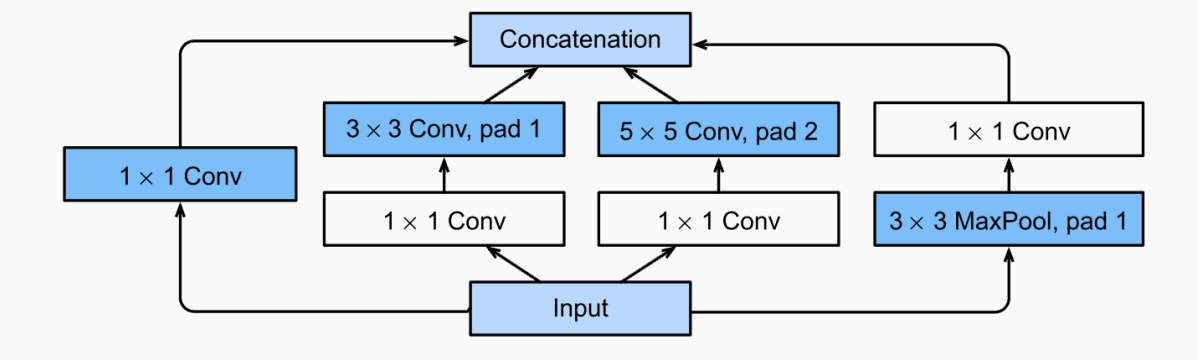


From AlexNet to VGG that is designed from building blocks.

- VGG constructs a network using reusable convolutional blocks.
- The use of blocks leads to very compact representations of the network definition. It allows for efficient design of complex networks.
- VGG also leveraged layers of **deep** and **small** convolutions (i.e., 3×3) for more effective feature representation.

https://d2l.ai/chapter_convolutional-modern/

Networks with Parallel Concatenations (GoogLeNet)

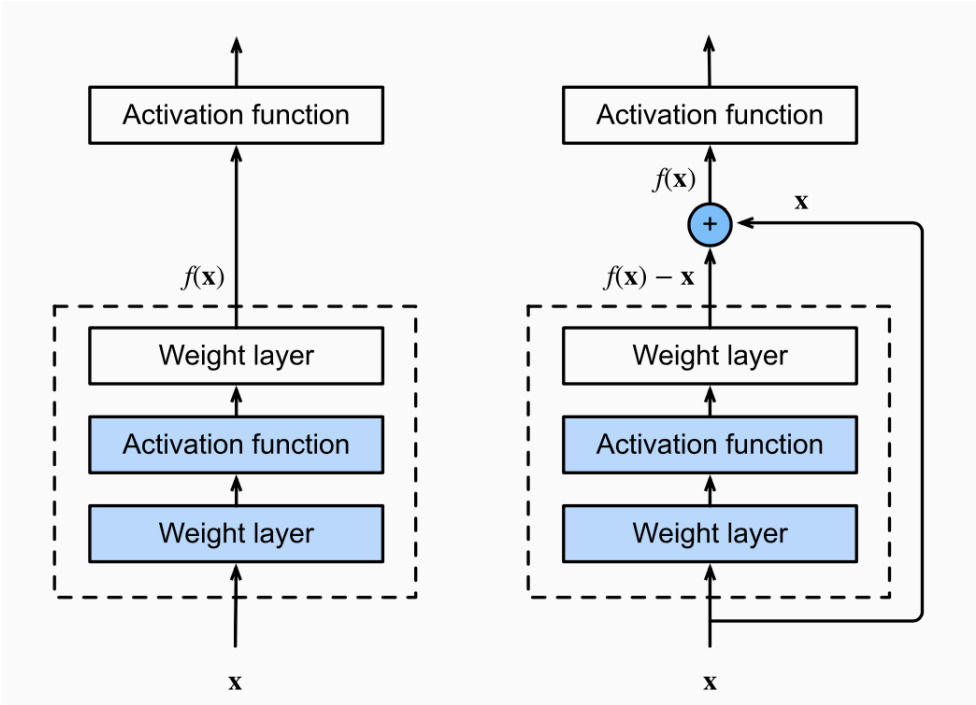


Structure of the Inception block.

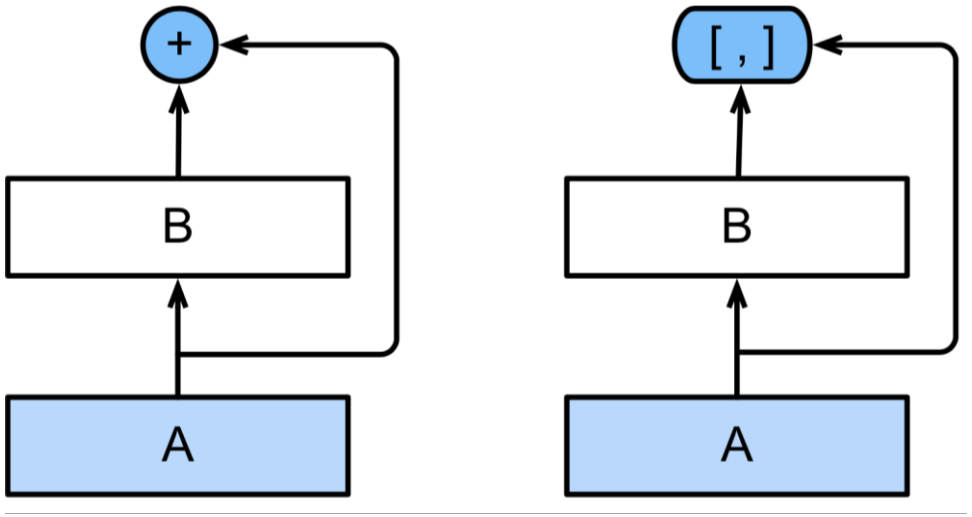
- The Inception block is equivalent to a subnetwork with four paths. It extracts information in parallel through convolutional layers of different window shapes and max-pooling layers.
- GoogLeNet, as well as its succeeding versions, was one of the most **efficient** models on ImageNet, providing similar test accuracy with lower computational complexity.

https://d2l.ai/chapter_convolutional-modern/

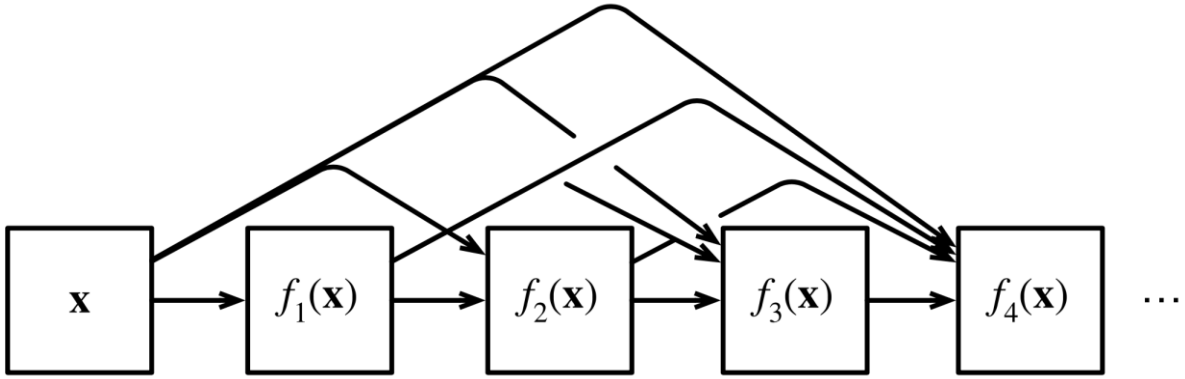
Residual Networks (ResNet) and Densely Connected Networks (DenseNet)



A regular block (left) and a residual block (right).



The main difference between ResNet (left) and DenseNet (right) in cross-layer connections: use of addition and use of concatenation.

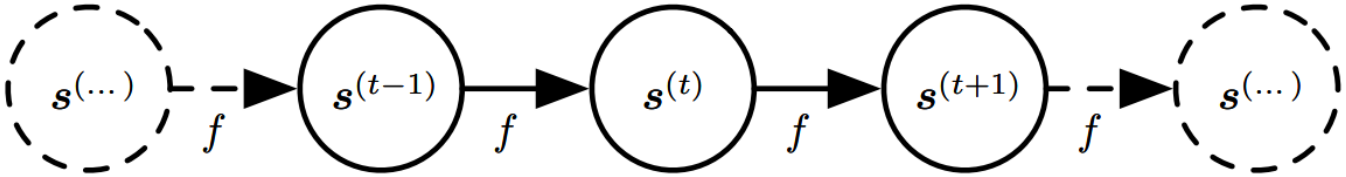


Densely connected Networks

https://d2l.ai/chapter_convolutional-modern/

Recurrent Neural Networks

- A classical dynamical system: $s^{(t)} = f(s^{(t-1)}; \theta)$
- This system is recurrent because the definition s at time t refers back to the same definition at time $t-1$. And we could unfold it as:



- Now, let us consider a dynamical system driven by external signal x:

$$s^{(t)} = f(s^{(t-1)}; x^{(t)}; \theta)$$

where the state now contains information about the whole past sequence.

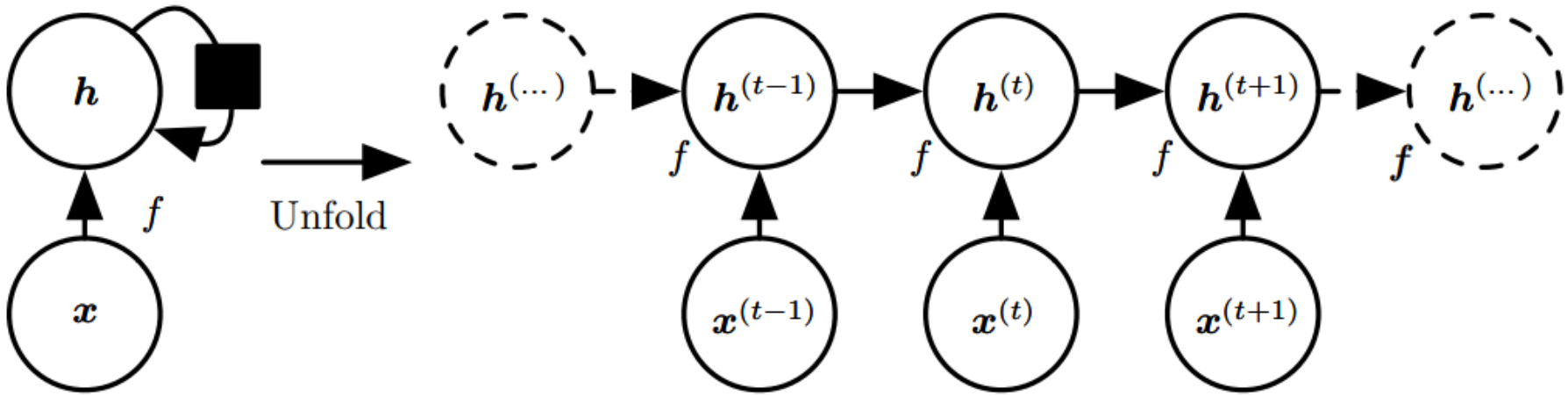
Deep Learning. MIT Press.

Recurrent Neural Networks

- RNN could be built as many ways, one way to define the hidden state is via:

$$h^{(t)} = f\left(h^{(t-1)}; x^{(t)}; \theta\right)$$

And we could unfold it as:



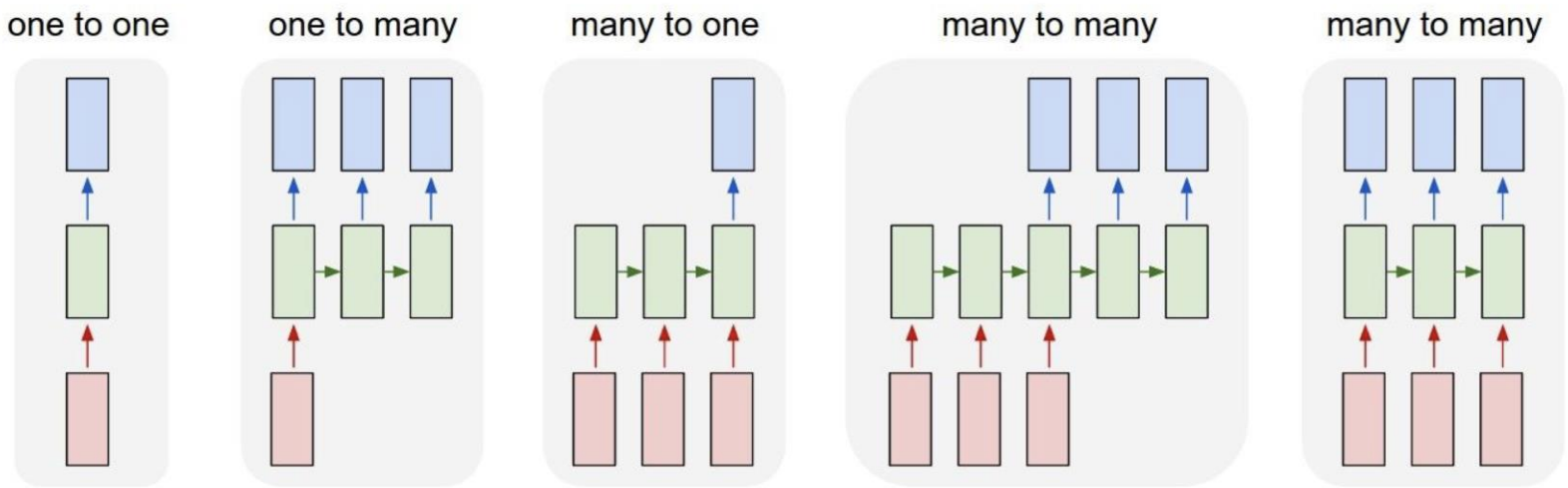
Deep Learning. MIT Press.

Recurrent Neural Networks

- Mathematical formalization :

$$h_t = f(W_h h_{t-1} + W_i x_t)$$

  where f is a nonlinear and differentiable function.
- The outputs are depending on problem and computational resource.
- Example for outputs - from left to right: vanilia NN, image caption, sentiment classification, machine translation, surgical video classification on frame level.



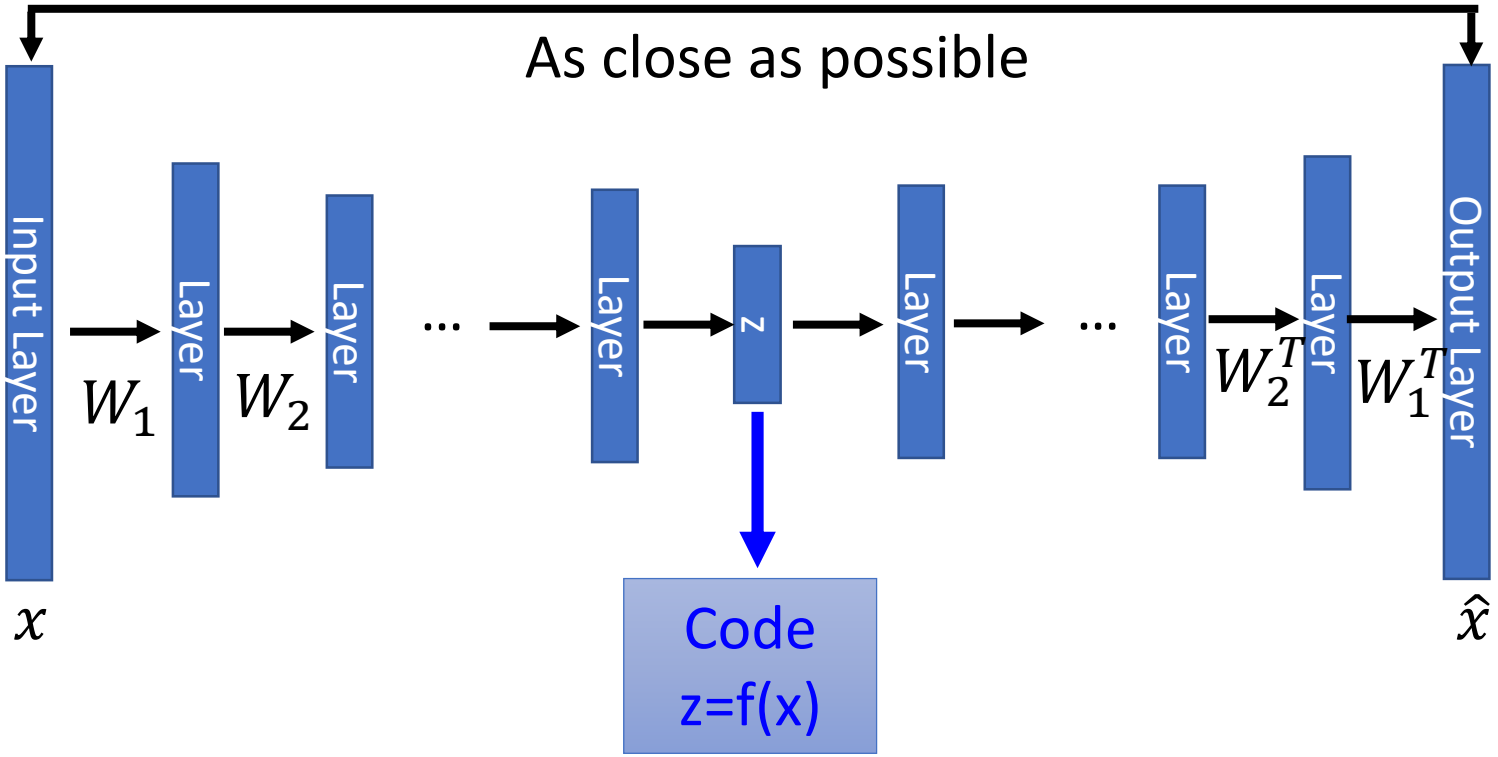
Autoencoder

- Supervised learning uses paired data and labels in order to train a network, e.g., disease classification.
- Unsupervised learning relies on data only. Although no explicit labels are required, it still needs to define a loss – this is an implicit supervision.

- The primary objective of Autoencoders is data compression.
- Unsupervised learning approach for learning a lower-dimensional feature representation from unlabeled training data.

Autoencoder

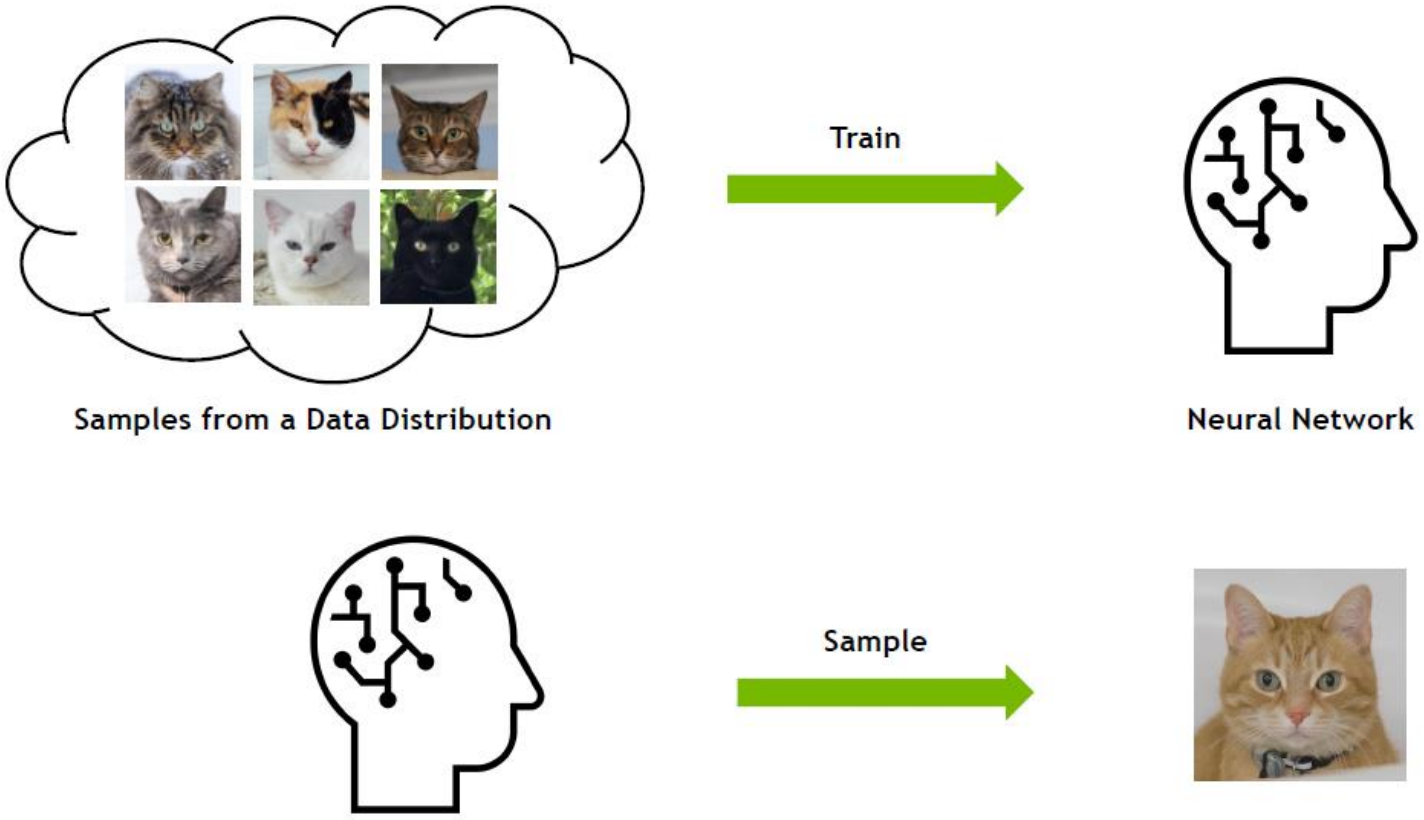
- Encoder: compress input into a latent-space of usually smaller dimension.
- A bottleneck (responsible for compression).
- Decoder: reconstruct input from the latent space.



Hinton, et al. Reducing the dimensionality of data with neural networks. Science 2006.

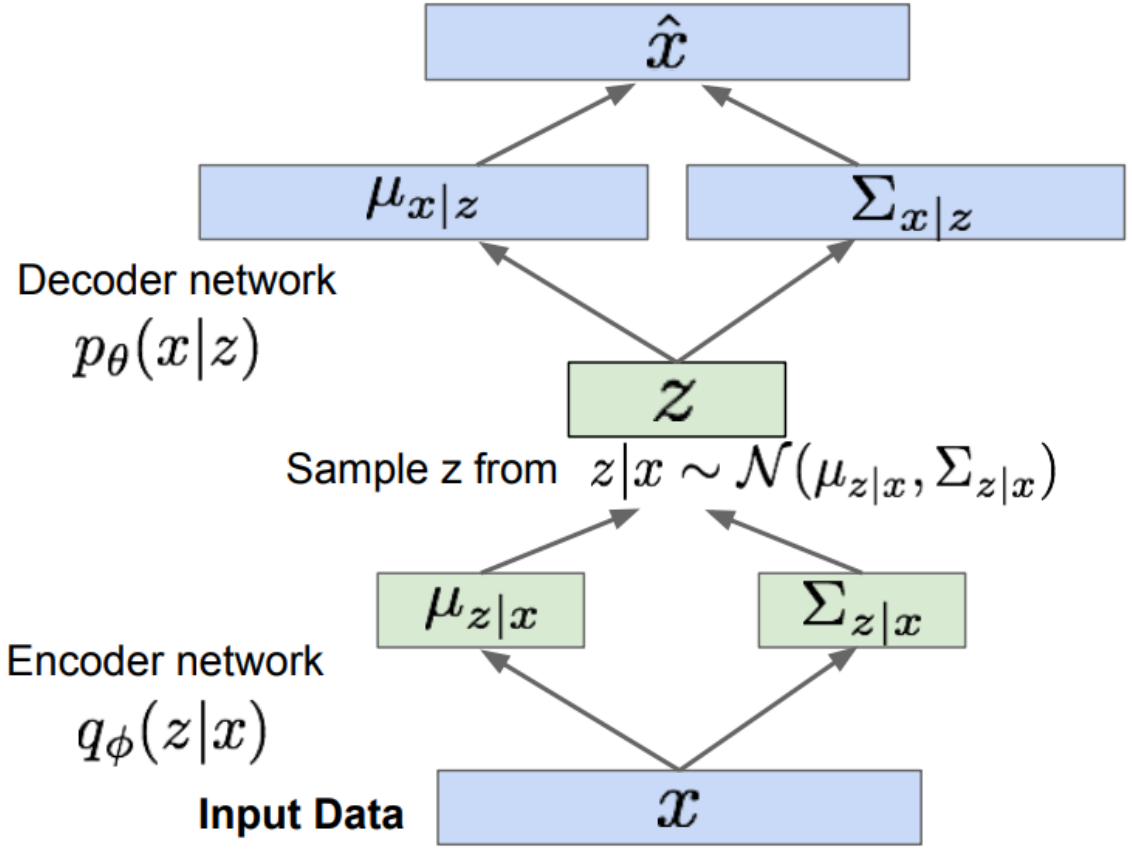
Deep Generative Learning

- Learning to generate data



Variational Autoencoder (VAE)

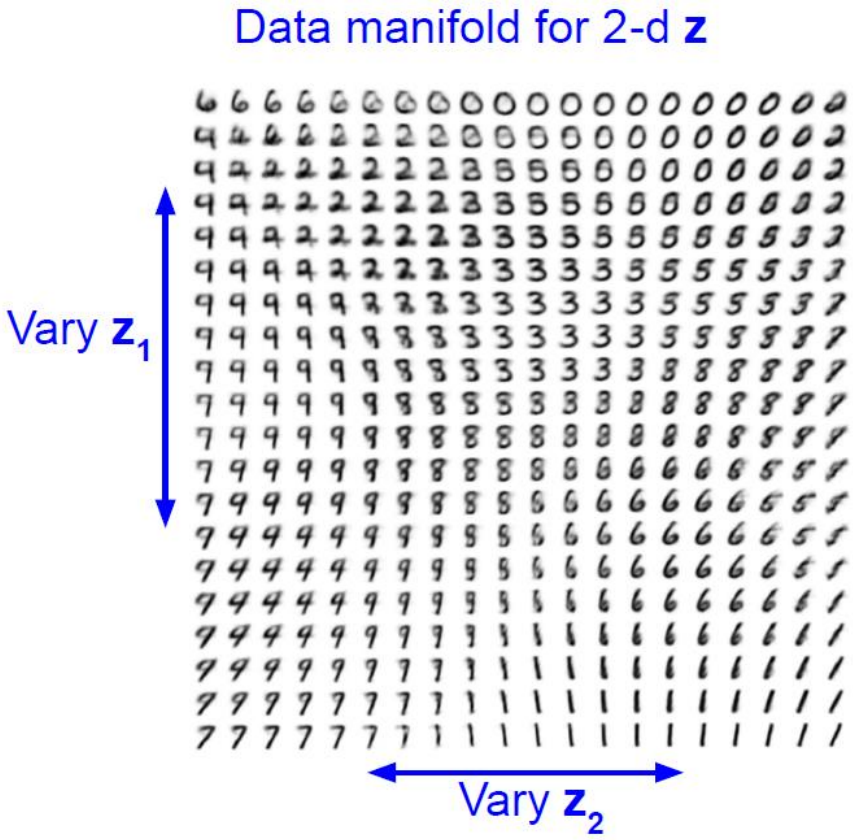
- Latent variable models form a rich class of probabilistic models that can infer hidden structure in the underlying data.
- By forcing latent variables to become normally distributed, VAEs gain control over the latent space.

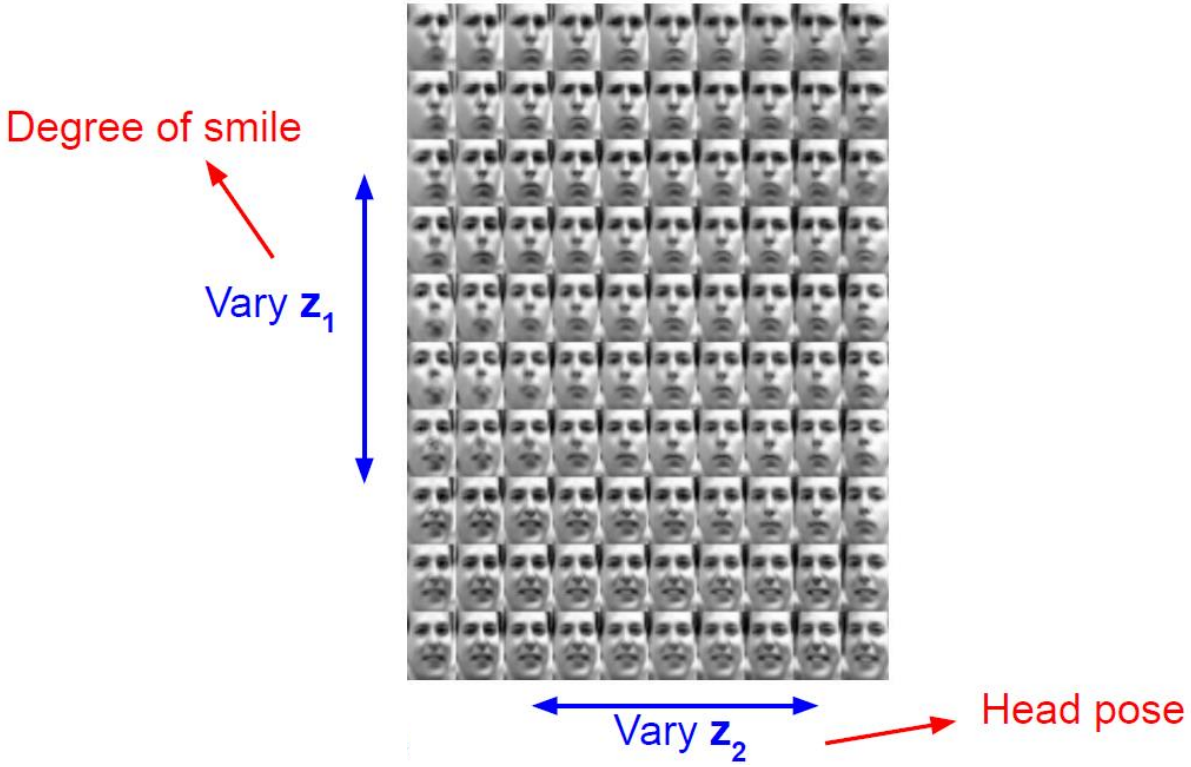


Stanford CS231n, Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014.

Variational Autoencoder (VAE)

- Given a trained VAE, use decoder network and sample z from prior.

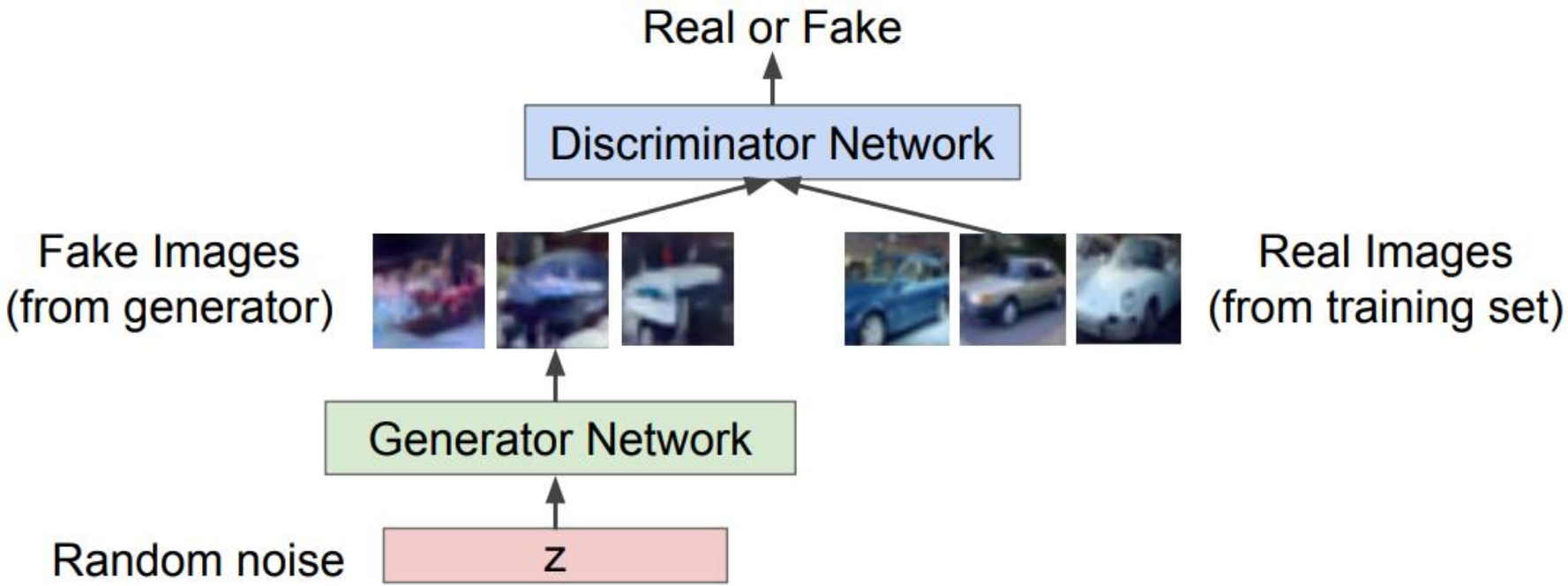




Stanford CS231n, Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014.

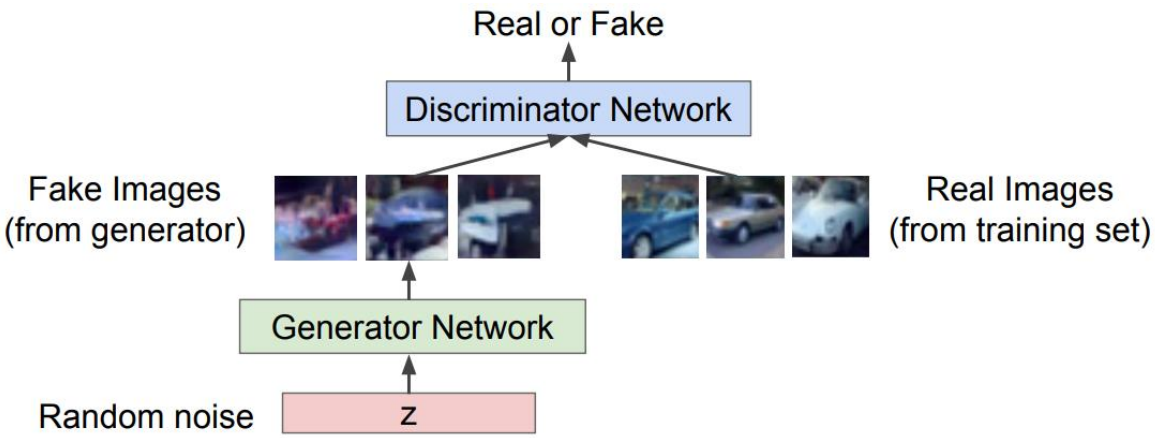
Generative Adversarial Network (GAN)

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



Generative Adversarial Network (GAN)

- **Training GANs: Two-player game**
- 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).



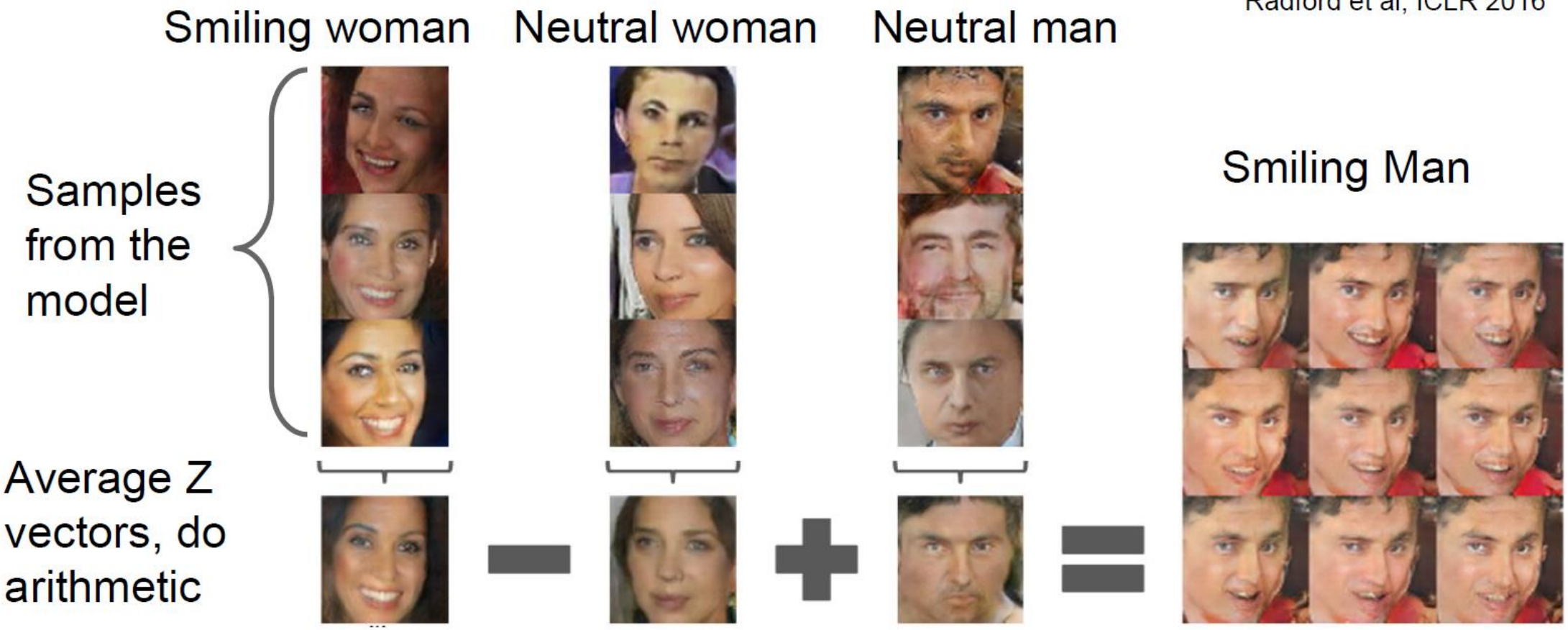
Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Generative Adversarial Network (GAN)

Radford et al, ICLR 2016



Denoising Diffusion Probabilistic Model

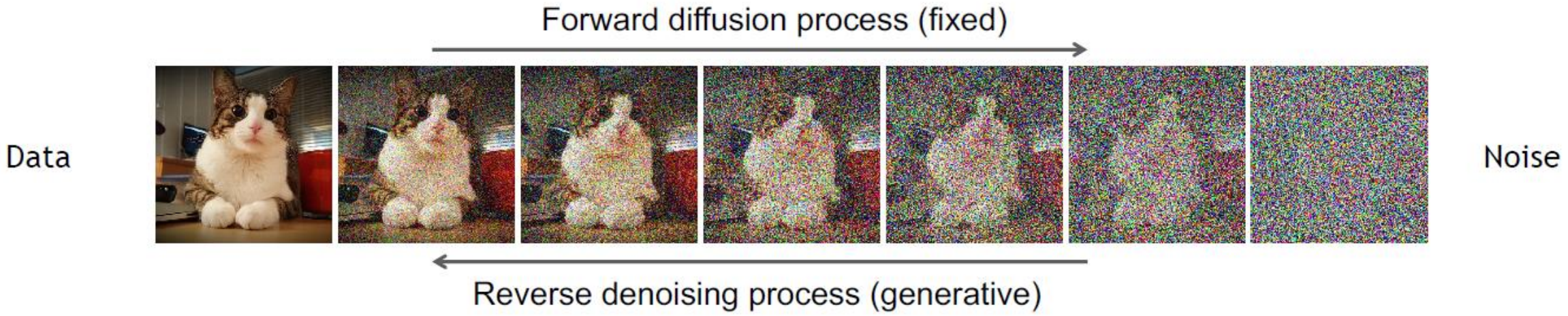
- Learning to generate by denoising.
- Emerging as powerful generative models, outperforming GANs.

Dhariwal , et al. Diffusion models beat gans on image synthesis. NeuIPS 2021.
Ho, et al. Cascaded Diffusion Models for High Fidelity Image Generation. JMLR 2022.

Denoising Diffusion Probabilistic Model

Denoising diffusion models consist of two processes:

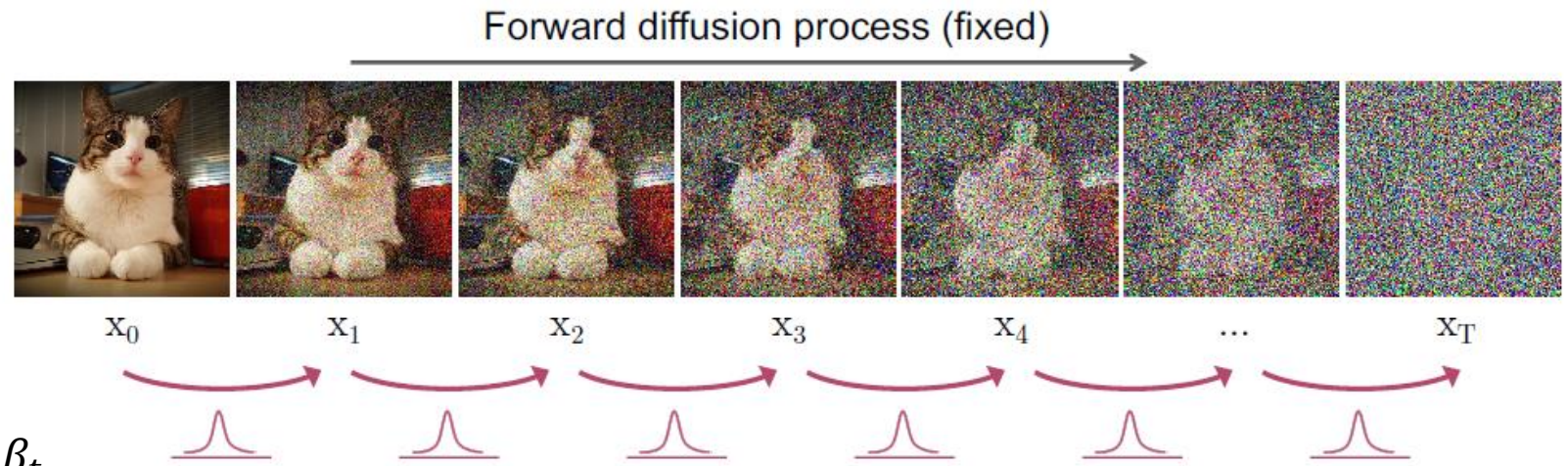
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Denoising Diffusion Probabilistic Model

Forward Diffusion Process

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



Noise Scheduler: β_t

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \Rightarrow q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

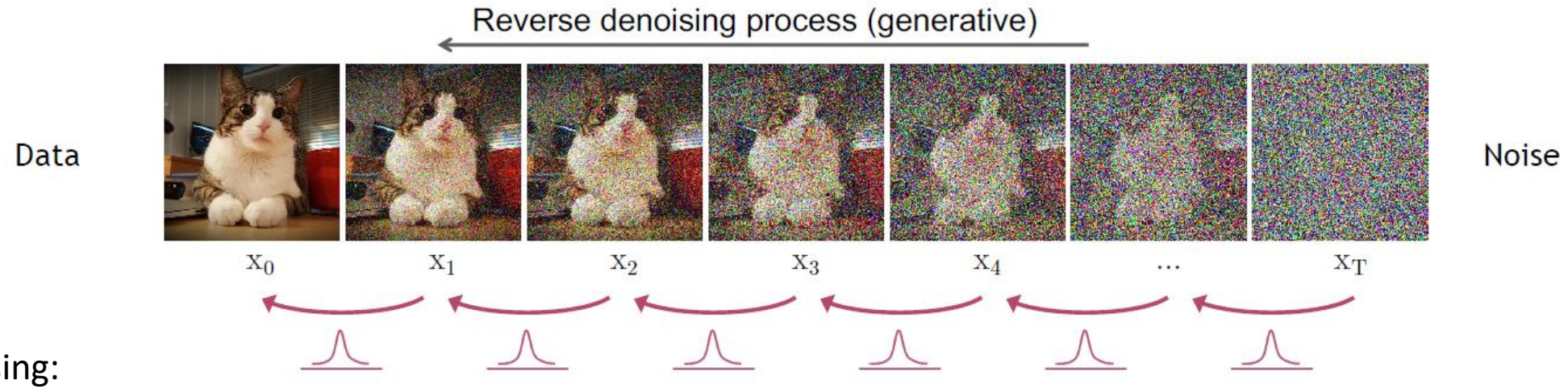
(joint)

Output Mean Variance (fixed)

Denoising Diffusion Probabilistic Model

Reverse Denoising Process (parametrized backward process)

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



Denoising:
$x_{t-1} = x_t - noise$

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

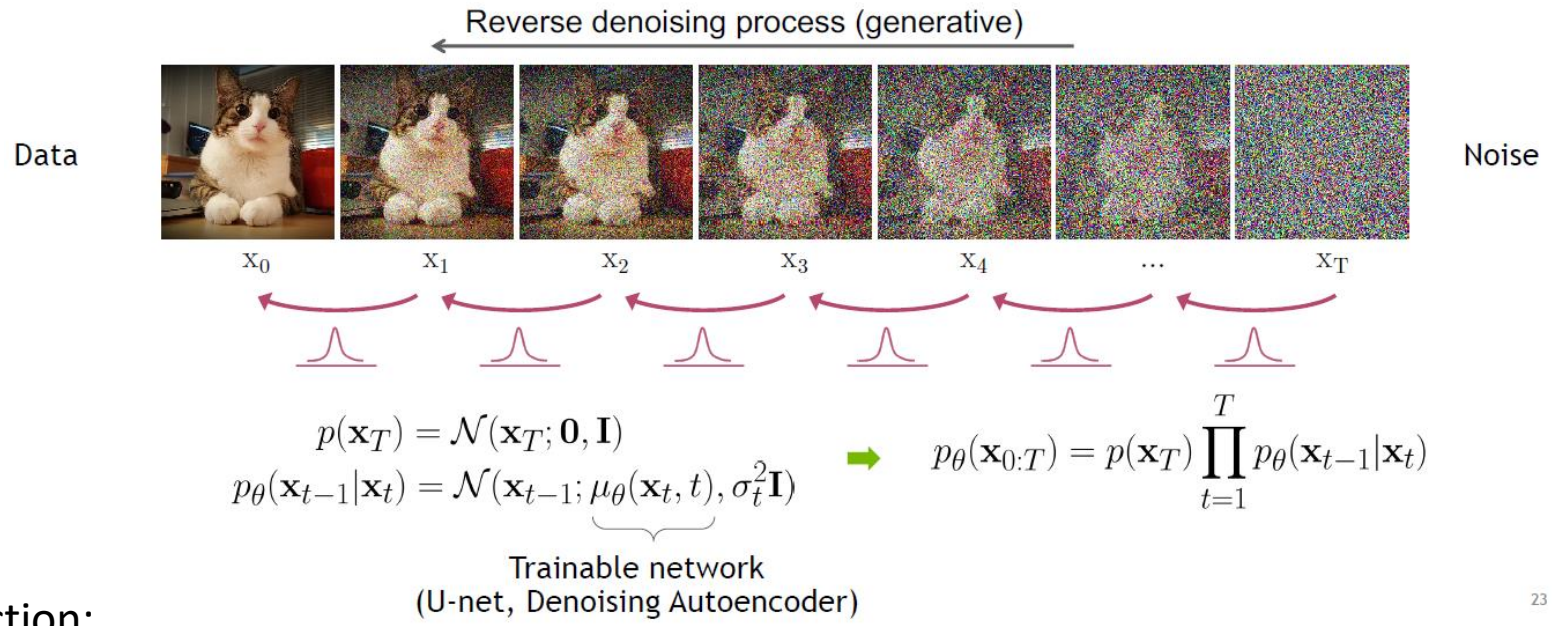
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \sigma_t^2 \mathbf{I})$$

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Denoising Diffusion Probabilistic Model

Reverse Denoising Process (parametrized backword process)

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



Loss function:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Added noise

Predicted noise

Denoising Diffusion Probabilistic Model

Training and Sampling

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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

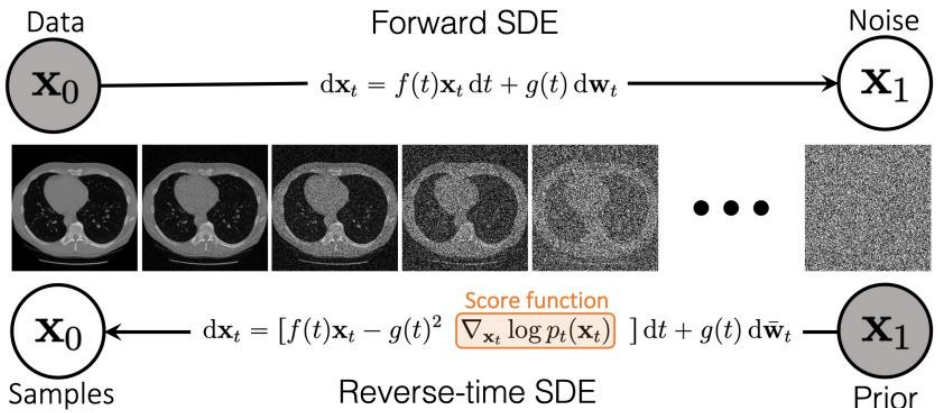
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

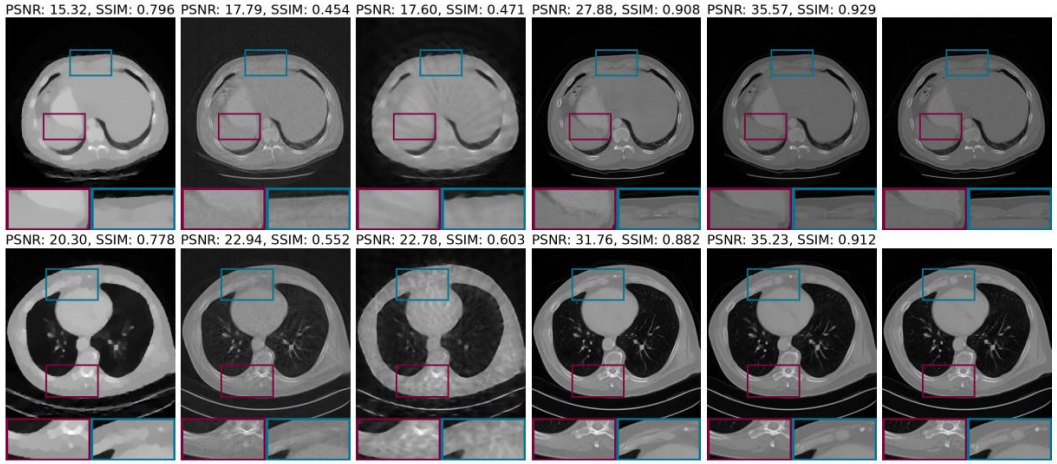
Ho, et al. Cascaded Diffusion Models for High Fidelity Image Generation. . JMLR 2022.

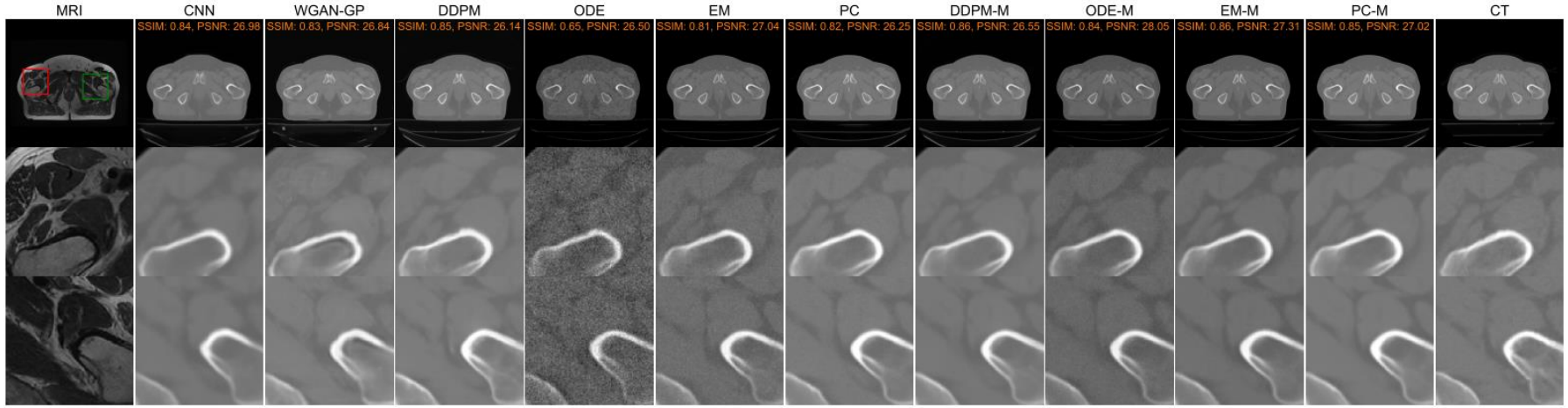
Denoising Diffusion Probabilistic Model

- Recent applications in MIA.

Song, et al. ICLR 2022. Unsupervised learning for inverse medical imaging.







Lyu, et al. Conversion between ct and mri images using diffusion and score-matching models. Arxiv 2022.

Summary

- Machine learning basics
- Deep learning
- Regularization for deep learning
- Optimization for deep learning
- Advanced deep learning models