



Lightweight Deep Learning with Model Compression (Part 1)

**U Kang
Dept. of Computer Science and Eng.
Seoul National University**



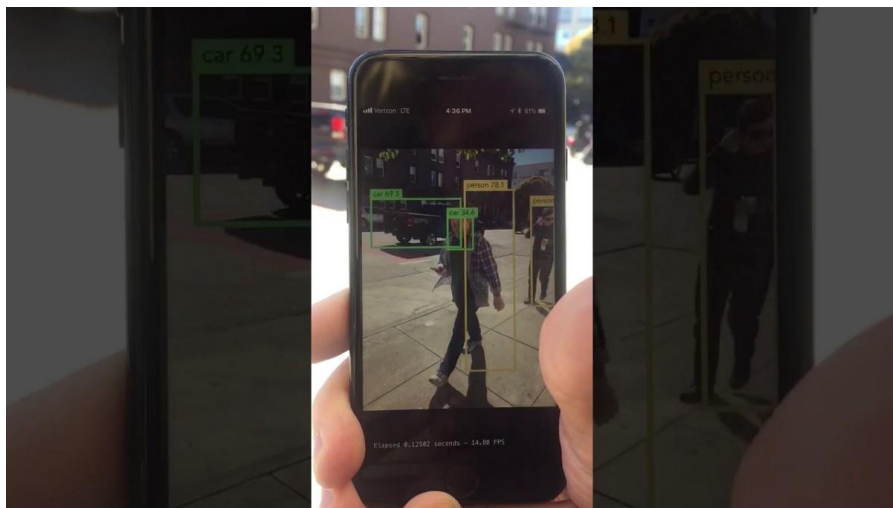
Why Model Compression?

- Recent deep learning models are becoming more complex
 - LeNet-5: 1M parameters
 - VGG-16: 133M parameters
- Problems of complex deep models
 - Huge storage(memory, disk) requirement
 - Computationally expensive
 - Uses lots of energy
 - Hard to deploy models on small devices (e.g. smart phones)



Model Compression

- Goal: make a lightweight model that is fast, memory-efficient, and energy-efficient
 - Especially useful for edge device





Outline

- (Part 1) Overview of Deep Learning
 - Just enough background to understand Part 2, to make this tutorial self-contained
 - Materials mainly based on Deep Learning Book (Goodfellow et al., 2016)
- (Part 2) Model Compression Techniques
 - Pruning
 - Weight Sharing
 - Quantization
 - Low-rank Approximation
 - Sparse Regularization
 - Distillation

<https://datalab.snu.ac.kr/~ukang/talks/19-BigComp19-tutorial/>



Outline

- ➔ ☐ **What is Deep Learning?**
- ☐ Feedforward Neural Network
- ☐ Convolutional Neural Network
- ☐ Recurrent Neural Network
- ☐ Conclusion

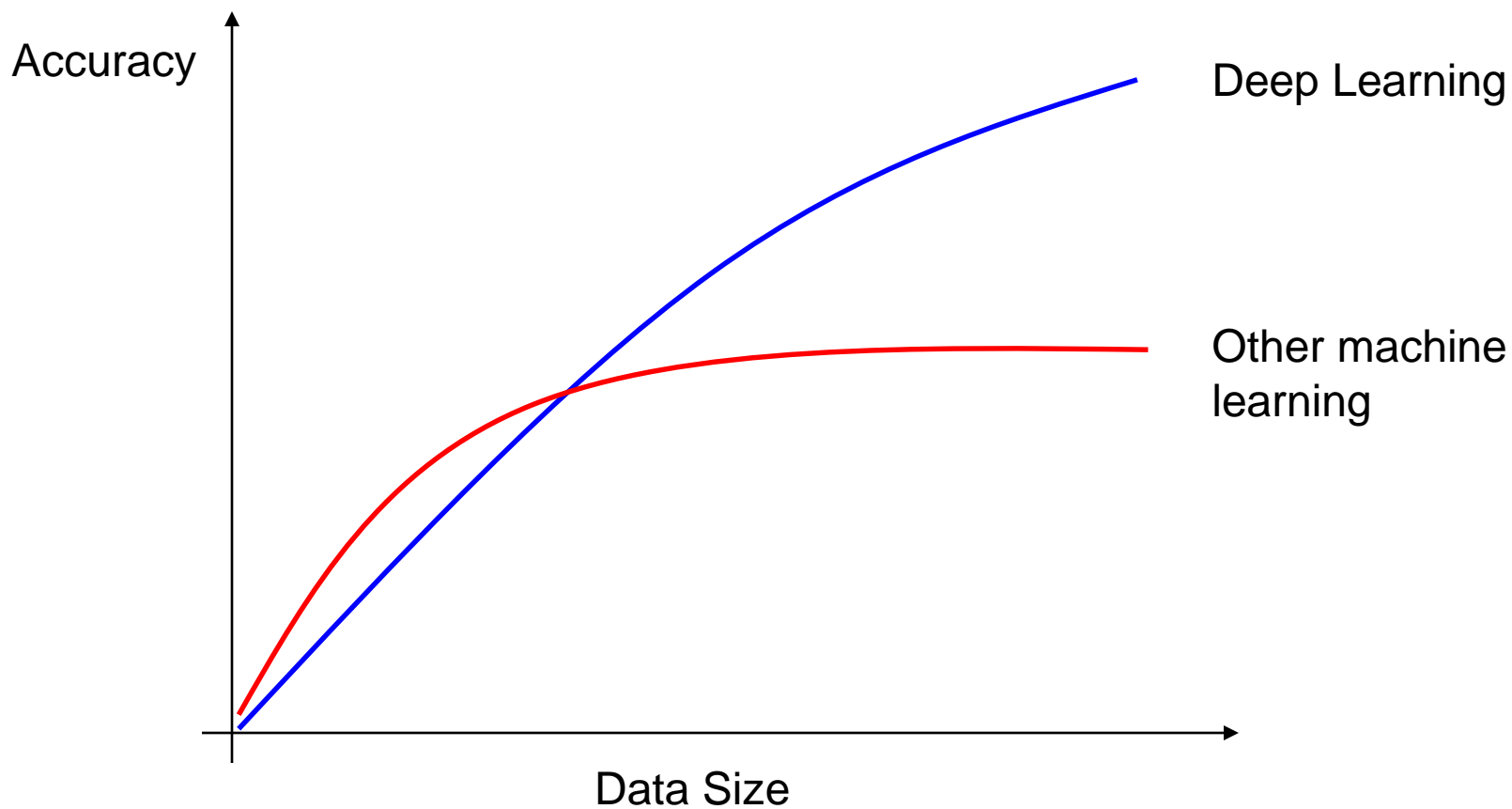


Deep Learning as a Machine Learning

- Machine Learning (ML)
 - Given x (predictor) and y (response), ML learns a function $f()$ from data, such that $y = f(x)$
 - E.g., x = image, y = category
 - This learned function $f()$ can be used to classify a new example x'
- This is different from a typical programming where you want to compute y , given x and $f()$
- Deep learning provides good performances in learning $f()$ for many problems
 - Learns non-linear functions



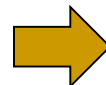
Deep Learning as a Machine Learning





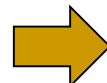
Learning Tasks

- Image classification



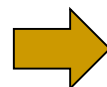
“Taxi”

- Speech recognition



Hello, dear

- Text classification



International politics

- ...



Main Idea

- Most perception (input processing) in the brain may use one learning algorithm
- Design learning methods that mimic the brain



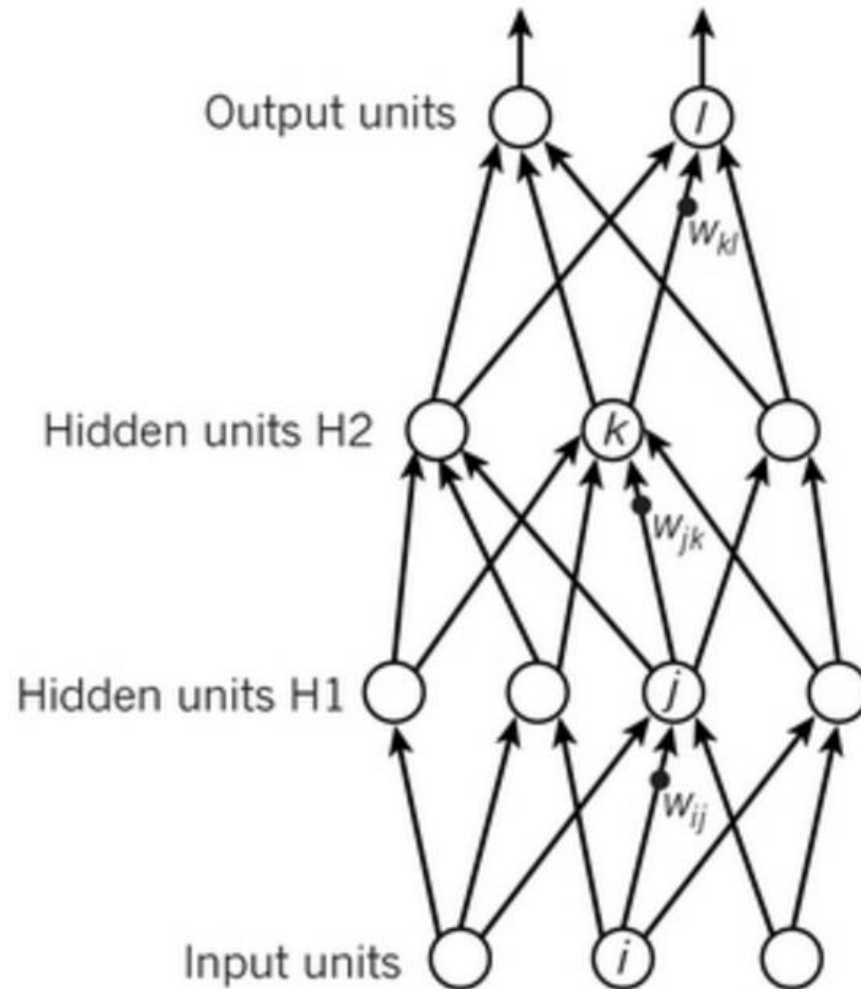


Neurons In the Brain





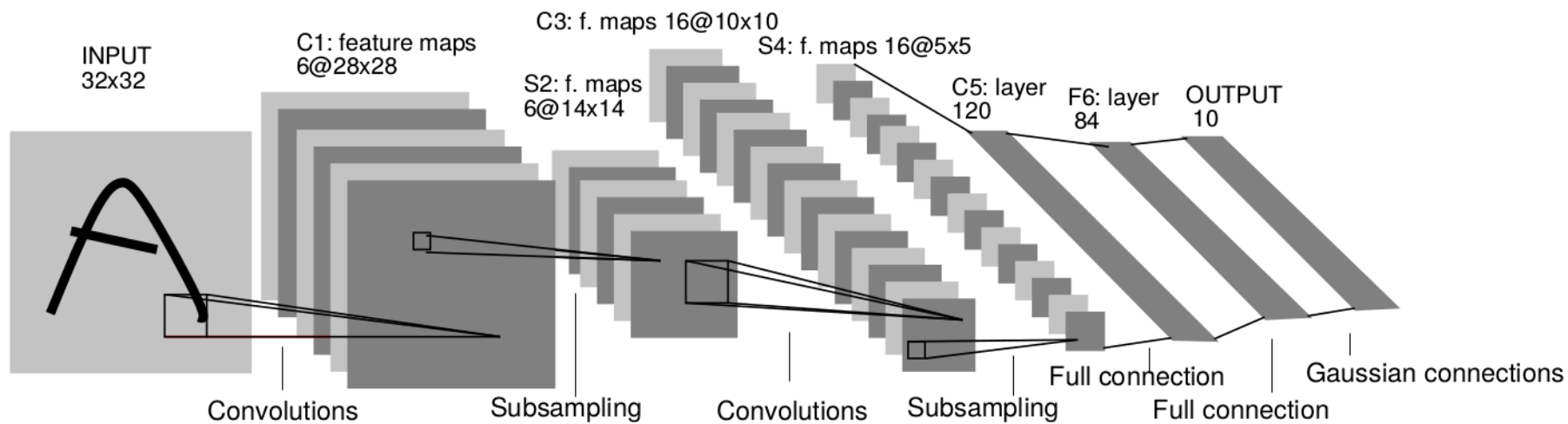
Neural Network



[LeCun et al.,
Nature 2015]



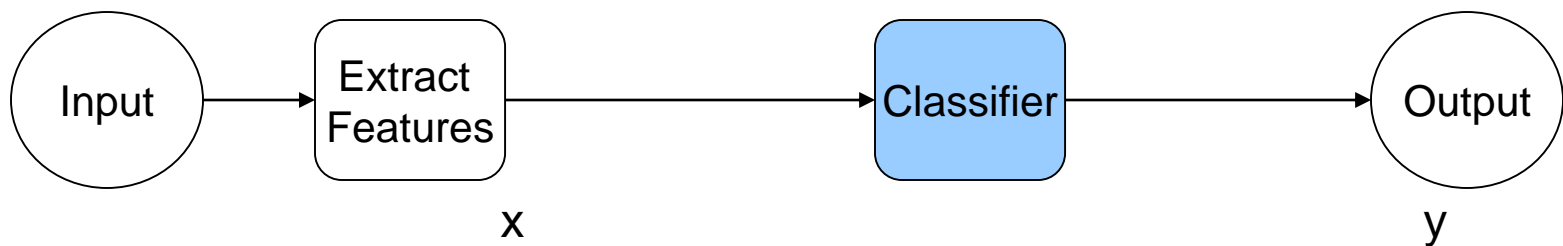
Convolutional Neural Net (CNN)



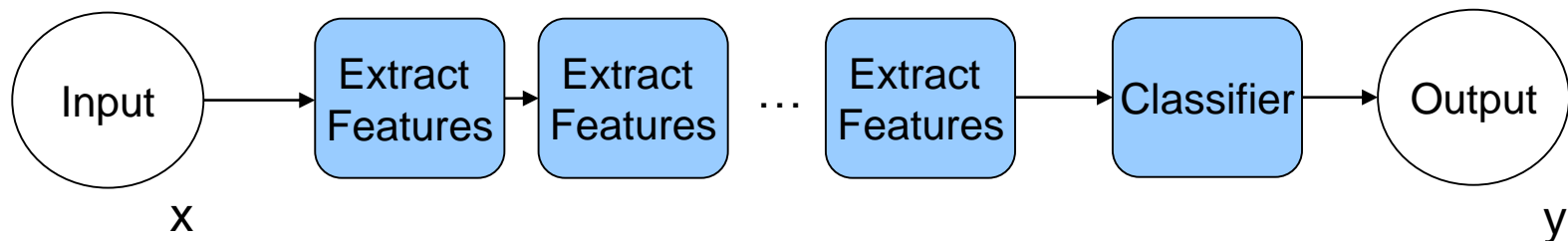


Representation Learning

■ Typical machine learning



■ Deep learning





Human Level Object Recognition

- ImageNet

- ~ 15M labeled images, ~ 22K categories



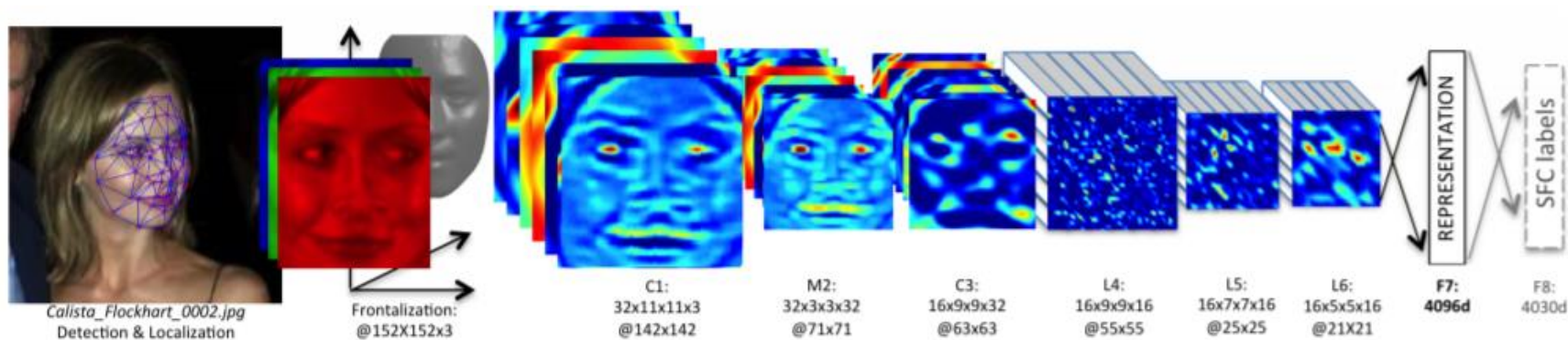
- Top-5 Error rates (1.2 million images, 1k categories)

- Non-CNN based method (~2012): 26.2 %
- Alexnet (2012): 15.3 %
- GoogLeNet (2014): 6.66 %
- Resnet (2015): 3.57%

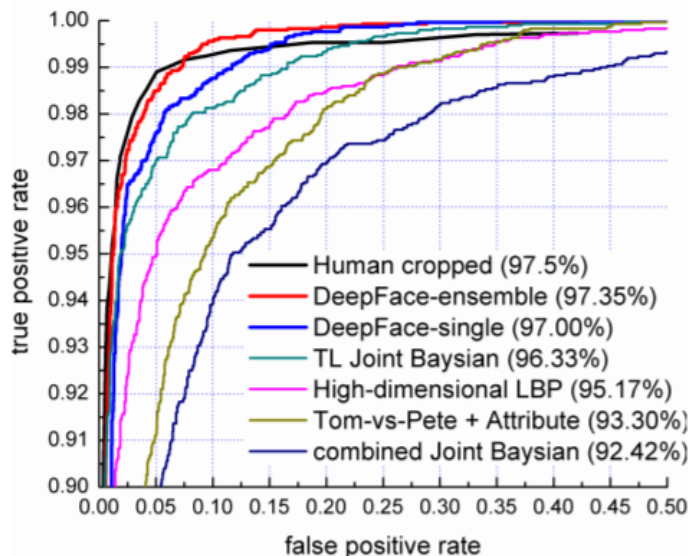
Human level:
5.1% error

Human-Level Face Recognition

■ DeepFace



97% accuracy
~ Human-level



[Taigman et al.
CVPR 2014]



Computer Game

■ Deepmind

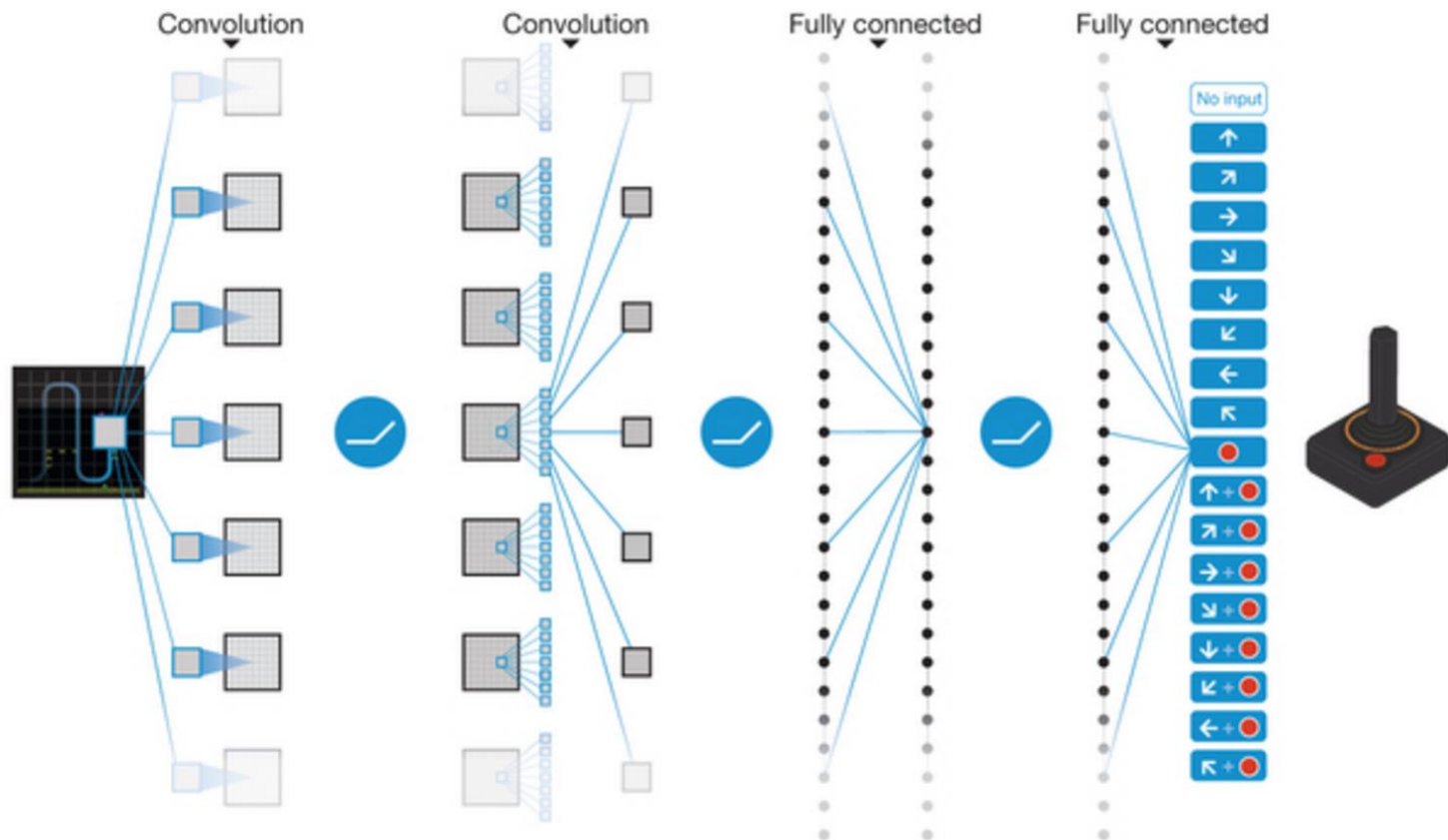




Computer Game

■ Deepmind

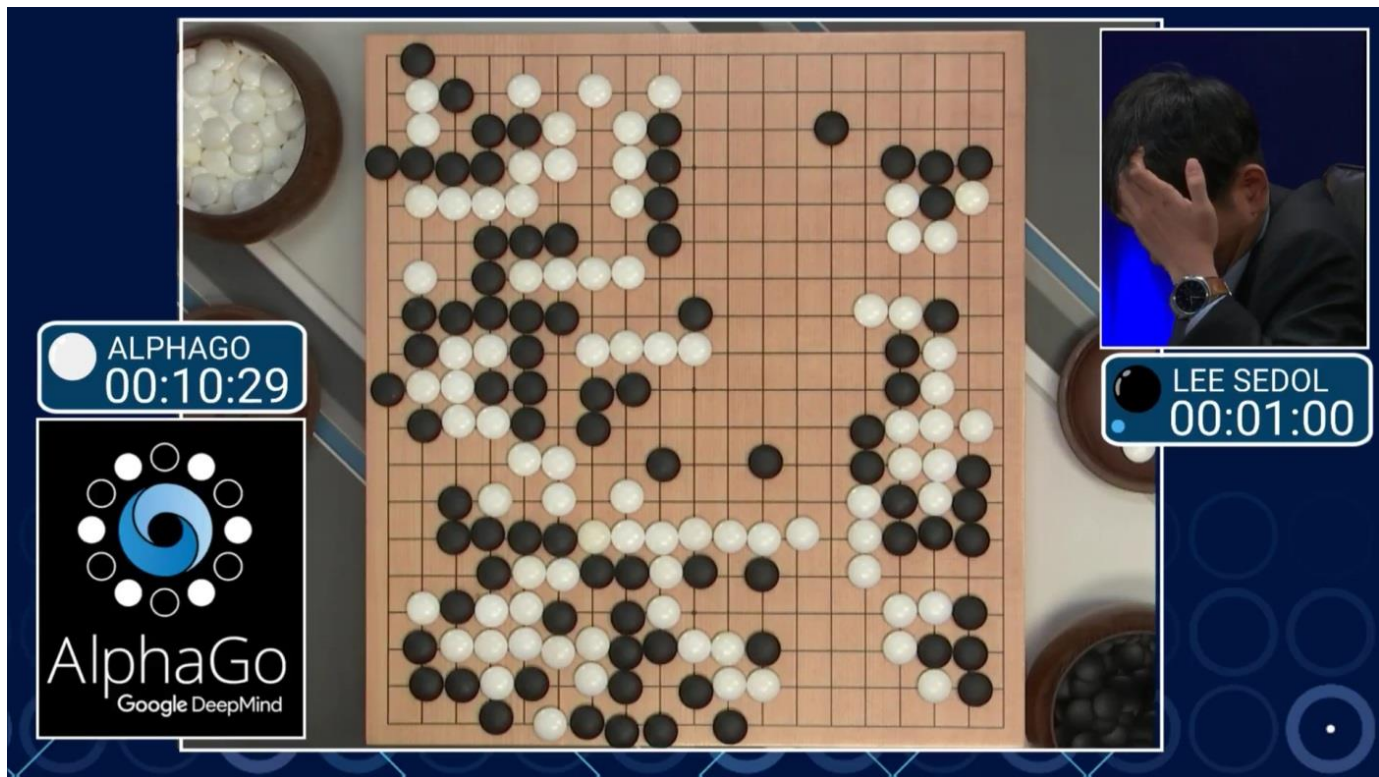
[Nature 2015]



<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

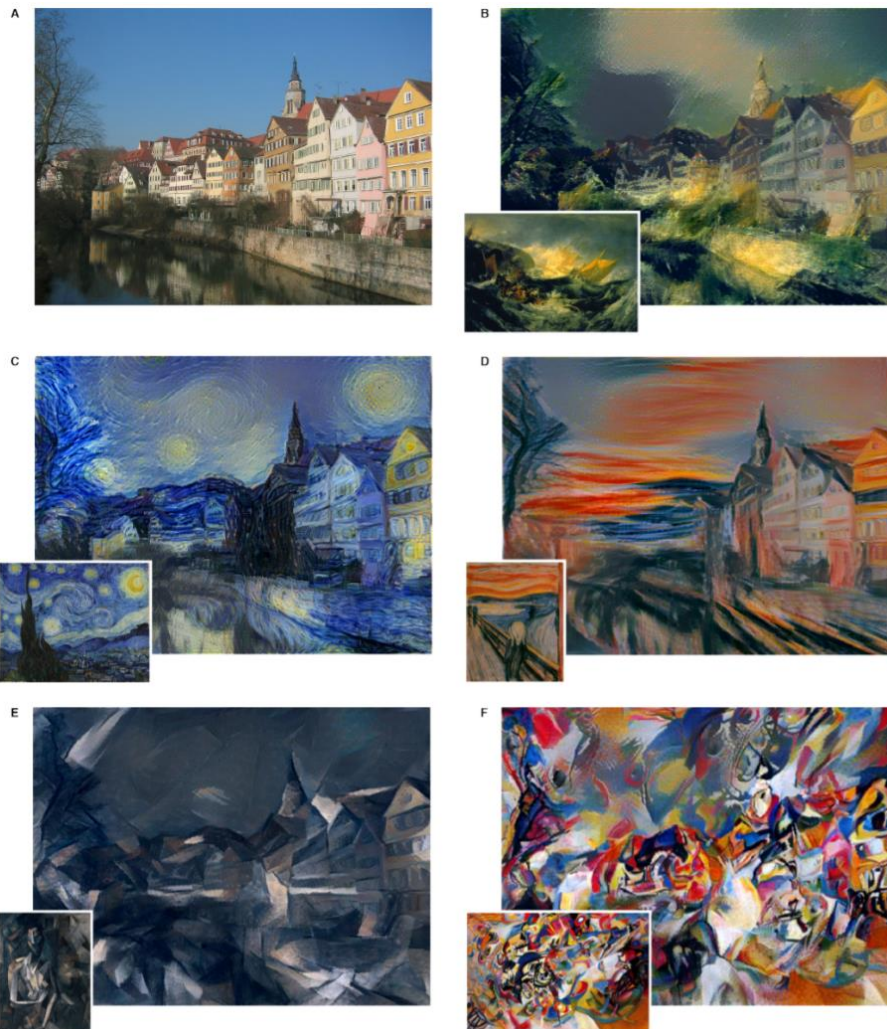


AlphaGo



[Silver et al., Mastering the game of Go with deep neural networks and tree search, Nature 2016]

Neural Artist



[Gatys et al., Image Style Transfer Using
Neural Networks, CVPR 2016]



Machine Translation

■ English to Korean

번역

즉석 번역 사용 안함

영어 한국어 독일어 언어 감지 ▼

한국어 영어 일본어 ▼

번역하기

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data. Deep learning is a driving force of the recent advances in AI. In this course, we study core techniques of deep learning to analyze large amount of data. Topics include machine learning basics, deep feedforward networks, regularization, optimization, convolutional networks, recurrent neural networks, etc.

딥 학습은 데이터에서 높은 수준의 추상화를 모델링하려는 일련의 알고리즘에 기반한 기계 학습의 한 부분입니다. 깊은 학습은 AI의 최근 발전의 원동력입니다. 본 과목에서는 다량의 데이터를 분석하기 위한 심화 학습의 핵심 기술을 학습한다. 주제는 기계 학습 기본, 깊은 피드 포워드 네트워크, 정규화, 최적화, 길쌈 네트워크, 반복적인 신경 네트워크 등을 포함합니다.

dib hagseub-eun deiteoeseo nop-eun sujun-ui chusanghwaleul modellinghalyeoneun illyeon-ui algolijeum-e gibanhan gigye hagseub-ui han bubun-ibnida. gip-eun hagseub-eun Alui choegeun baljeon-ui wondonglyeog-ibnida. bon gwamog-eseoneun dalyang-ui deiteoleul bunseoghagiwihan simhwa hagseub-ui haegsim gisul-eul hagseubhanda. jujeneun gigye hagseub gibon, gip-eun pideu powodeu neteuwokeu, jeong-gyuhwa, choejeoghwa, gilssam neteuwokeu, banbogjeog in singyeong neteuwokeu deung-eul pohamhabnida.



Outline

☒ What is Deep Learning?

➔ ☐ **Feedforward Neural Network**

☐ Convolutional Neural Network

☐ Recurrent Neural Network

☐ Conclusion



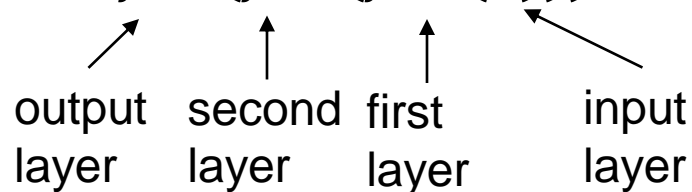
Deep FeedForward Networks

- Deep feedforward networks are the key deep learning models
 - Also called feedforward neural networks or multi-layer perceptrons (MLP)
 - Goal: approximate some function f^*
 - E.g., a classifier $y = f^*(x)$ maps an input x to a category y
 - A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of θ



Deep FeedForward Networks

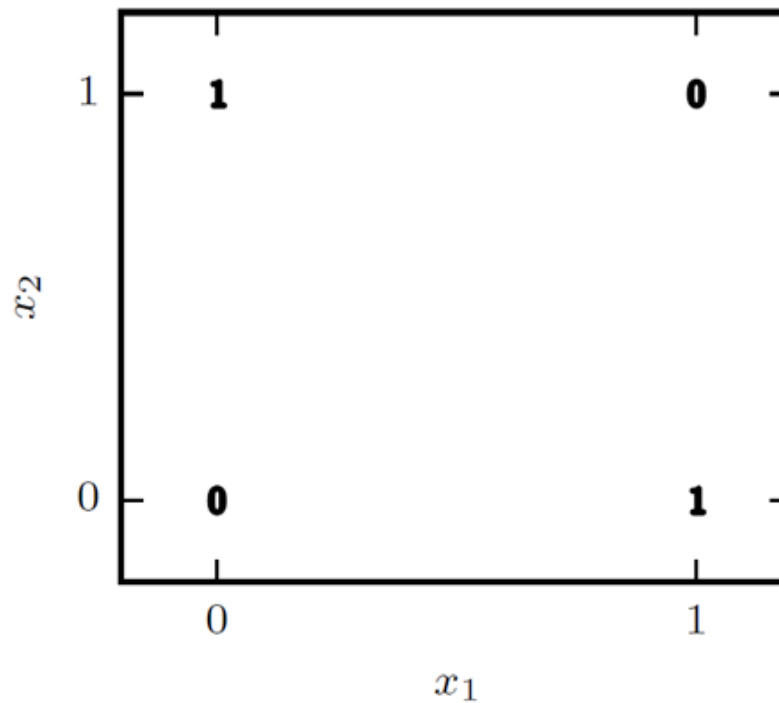
- Deep feedforward networks are the key deep learning models
 - These models are called feedforward because information flows through the function from x to f to output y
 - These models are called networks because they are typically represented by composing together many different functions
 - E.g., three functions $f^{(1)}, f^{(2)}, f^{(3)}$ connected in a chain to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$





Learning XOR

- XOR function: an operation on two binary values
 - XOR outputs 1 only when exactly one of the two values is 1





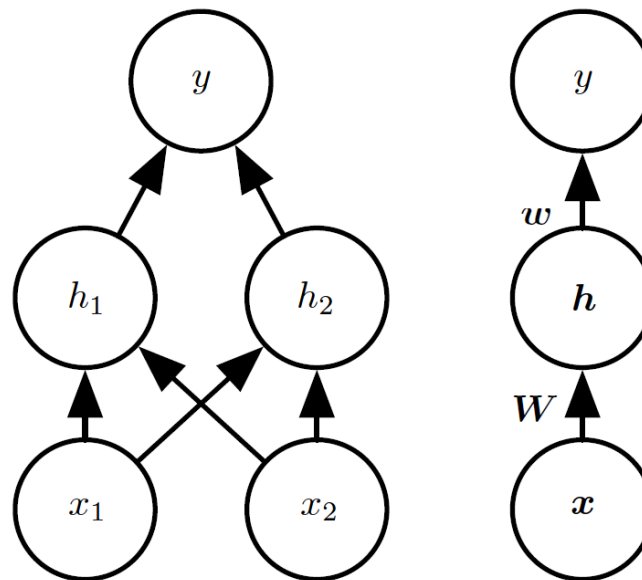
Learning XOR

- Our model provides $y = f(x; \theta)$, and our learning algorithm learns θ such that f outputs the same value as the target XOR function f^*
 - Evaluation will be performed on four points: $X = \{(0,0), (0,1), (1,0), (1,1)\}$
 - MSE loss function: $J(\theta) = \frac{1}{4} \sum_{x \in X} (f^*(x) - f(x; \theta))^2$
- First model: $f(x; w, b) = x^T w + b$
 - Solving the normal equation, we obtain $w = 0$ and $b = \frac{1}{2}$
 - That is, it outputs 0.5 everywhere
 - Linear models always fail for XOR!



Feedforward Network for XOR

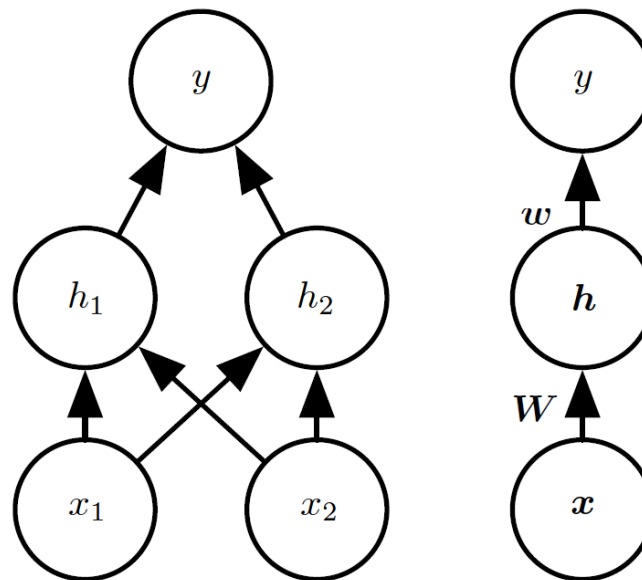
- Feedforward network with one hidden layer with two hidden units
 - The vector of hidden units are computed by $\mathbf{h} = f^{(1)}(\mathbf{x}; \mathbf{W}, \mathbf{c})$
 - The output unit is computed by $y = f^{(2)}(\mathbf{h}; \mathbf{w}, \mathbf{b})$
 - The complete model is $f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, \mathbf{b}) = f^{(2)}(f^{(1)}(\mathbf{x}))$





Feedforward Network for XOR

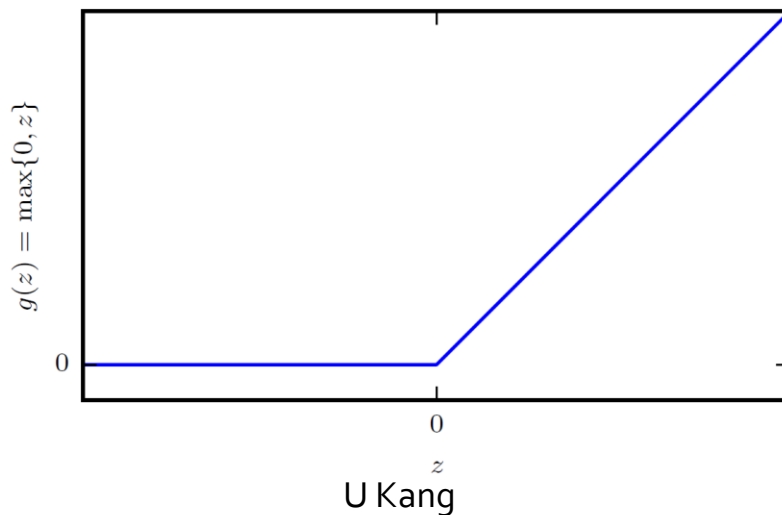
- Assume we use linear regression model for $f^{(2)}$
 - I.e., $f^{(2)}(\mathbf{h}) = \mathbf{h}^T \mathbf{w}$
- What function should $f^{(1)}$ compute?
 - What if $f^{(1)}$ is linear?





Feedforward Network for XOR

- We need a non-linear function to describe features
- Most neural networks do so using an affine transformation by a fixed, nonlinear function called an activation function
 - $\mathbf{h} = g(\mathbf{W}^T \mathbf{x} + \mathbf{c})$
 - g is typically chosen to be a function applied elementwise with $h_i = g(\mathbf{x}^T \mathbf{W}_{:,i} + c_i)$
 - The default activation function is rectified linear unit or ReLU: $g(z) = \max\{0, z\}$





Feedforward Network for XOR

- Feedforward network with ReLU

- $f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, \mathbf{b}) = \mathbf{w}^T \max\{0, \mathbf{W}^T \mathbf{x} + \mathbf{c}\} + b$

- Solution to the XOR problem

- $\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \mathbf{c} = [0 \ -1]^T, \mathbf{w} = [1 \ -2]^T, b = 0$

- From input to output

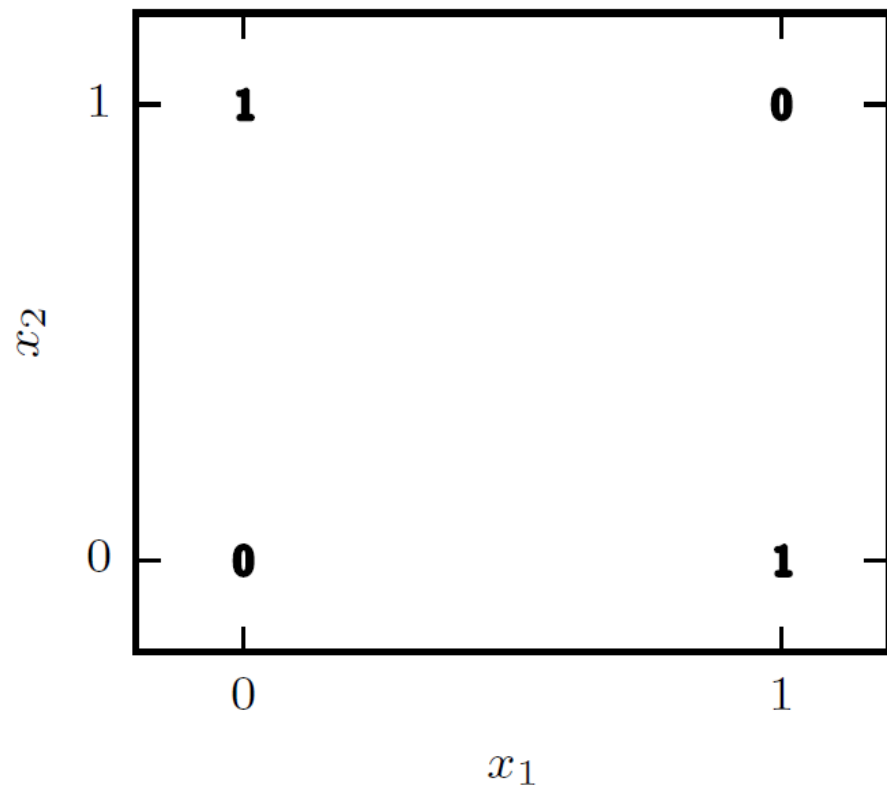
- $\mathbf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \mathbf{XW} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}, \text{adding } \mathbf{c} \rightarrow \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$

- Applying ReLU $\rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$, multiplying by the weight vector $\mathbf{w} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

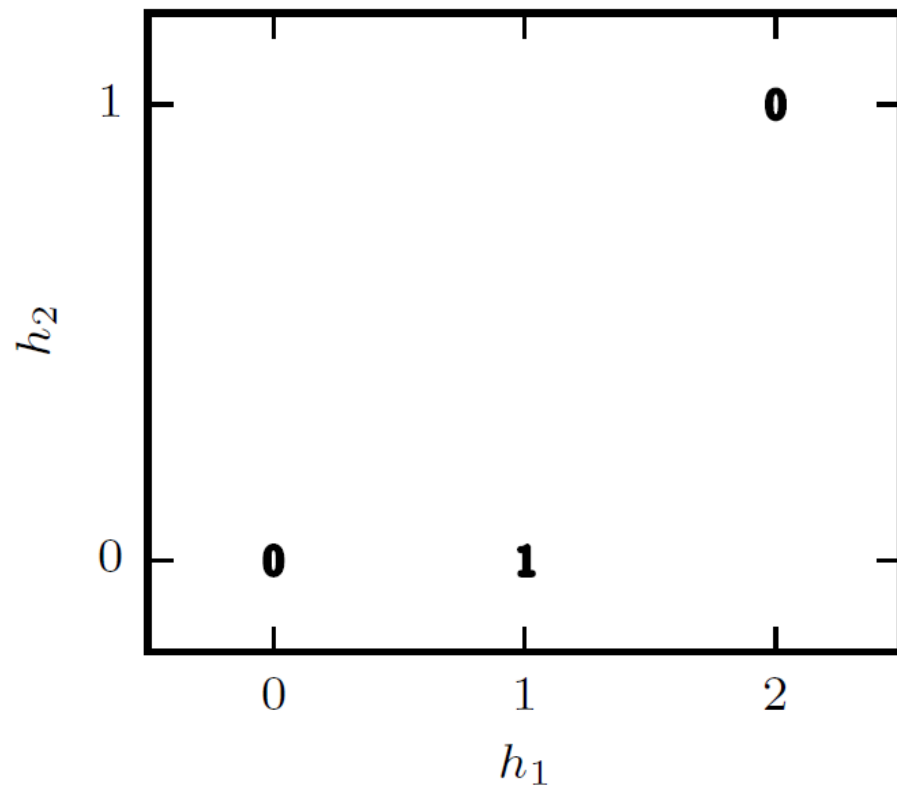


Solving XOR

Original \mathbf{x} space



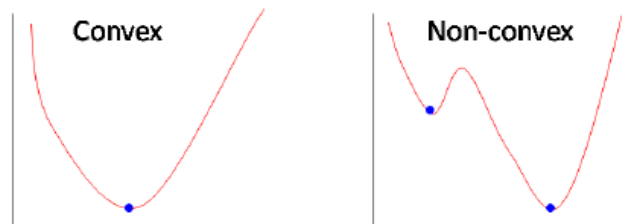
Learned \mathbf{h} space





Gradient-Based Learning

- Neural networks are trained by using iterative, gradient-based optimizers
 - The objective function is non-convex
 - These optimizers find a sufficiently low value, rather than global minimum



- Two important components in gradient-based learning
 - Cost functions
 - Output units



Cost Functions

- In most cases, our model defines $p(y|x; \theta)$ and we use the principle of maximum likelihood
 - I.e., minimize cross-entropy between the training data and the model's prediction
 - $J(\theta) = -E_{x,y \sim \hat{p}_{data}} \log p_{model}(y|x)$
 - If $p_{model}(y|x) = N(y; f(x; \theta), I)$, then we recover the mean squared error cost: $J(\theta) = \frac{1}{2} E_{x,y \sim \hat{p}_{data}} ||y - f(x; \theta)||^2$
 - The total cost function often is combined with a regularization term
 - E.g., weight decay parameter for linear regression



Softmax Units for Multinoulli Output Distributions



- Useful to represent a categorical distribution (= a probability distribution over a discrete variable with n possible values)
- The output vector \mathbf{y} contains n probabilities
- Softmax is a generalization of sigmoid for n possible values:
$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \text{ where } \mathbf{z} \in R^n \text{ and } i \in Z^n \text{ in } [0, n - 1]$$
- Given features \mathbf{h} , a layer of softmax output units produces a vector $\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}^T \mathbf{h} + b)$
- The loss function is $J(\theta) = -\log P(y|x) = -\log \text{softmax}(\mathbf{z})_y$ where $\mathbf{z} = \mathbf{W}^T \mathbf{h} + b$



Hidden Units

■ Rectified linear units (ReLU)

- $g(z) = \max\{0, z\}$
- Advantage: simple and effective (no vanishing gradient problem)
- Disadvantage: cannot learn via gradient-based methods on examples for which their activation is 0

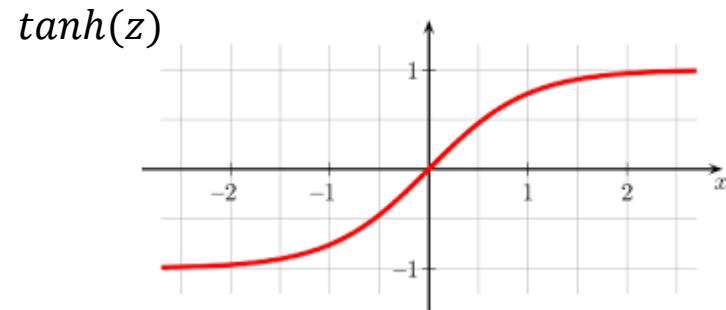
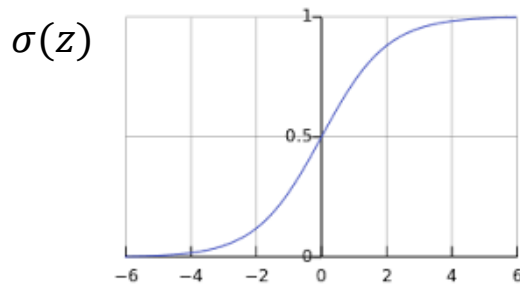
■ Generalizations of ReLU

- Generalizations using non-zero slope α_i when $z_i < 0$: $h_i = \max(0, z_i) + \alpha_i \min(0, z_i)$
 - Absolute value rectification: use $\alpha_i = -1$ to obtain $g(z) = |z|$
 - Leaky ReLU: fixes α_i to a small value like 0.01
 - PReLU (parametric ReLU) treats α_i as a learnable parameter



Hidden Units

- Logistic sigmoid and hyperbolic tangent
 - Famous hidden units before the introduction of ReLU
 - Sigmoid function $\sigma(z)$
 - Hyperbolic tangent function $\tanh(z) = 2\sigma(2z) - 1$



- Problems: saturate to a high value when z is very positive, and to a low value when z is very negative
 - Gradient is close to 0 when they saturate: only strongly sensitive to their input when z is near 0



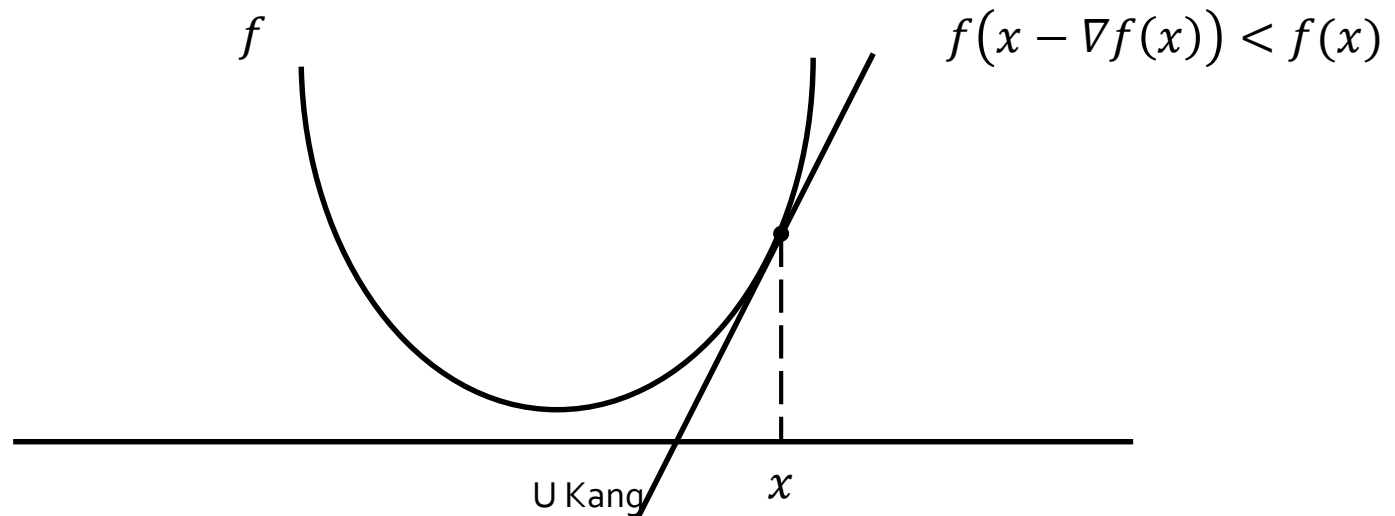
Training

- Training a neural network model means to learn parameters θ
- The goal is to find θ that minimizes the loss function $J(\theta; x; y)$
- $\theta^* = \operatorname{argmin}_{\theta} J(\theta; x; y)$
- How to learn such θ^* ?
 - Gradient Descent!



Gradient Descent

- A simple way to minimize a function $f()$:
 - Take a gradient ∇f
 - Start at some point x and evaluate $\nabla f(x)$
 - Make a step in the reverse direction of the gradient:
 $x \leftarrow x - \eta \nabla f(x)$. This is called *gradient descent*.
 - Repeat until converged





Backpropagation - Overview

- In gradient descent, it is essential to compute gradient, or partial derivative $\frac{\partial J}{\partial w}$ for each parameter w
- Backpropagation is an algorithm to compute partial derivatives $\frac{\partial J}{\partial w}$ of cost function J with regard to parameter w efficiently in neural networks
- The main idea is dynamic programming
 - Many computations share common computations
 - Store the common computations in memory, and read them from memory when needed, without re-computing them from scratch



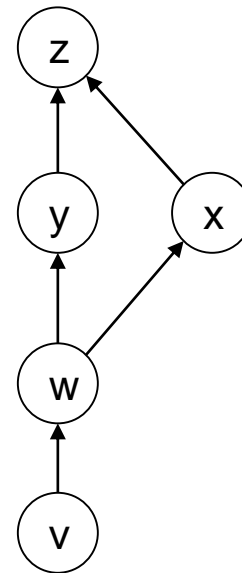
Back-Propagation Example

- Back-propagation procedure (re-use blue-colored computation)

- Compute $\frac{\partial z}{\partial y}$ and $\frac{\partial z}{\partial x}$


- $\frac{\partial z}{\partial w} \leftarrow \frac{\partial y}{\partial w} \frac{\partial z}{\partial y} + \frac{\partial x}{\partial w} \frac{\partial z}{\partial x}$

- $\frac{\partial z}{\partial v} \leftarrow \frac{\partial w}{\partial v} \frac{\partial z}{\partial w}$





Outline

- ☒ What is Deep Learning?
- ☒ Feedforward Neural Network
-  ☐ **Convolutional Neural Network**
- ☐ Recurrent Neural Network
- ☐ Conclusion



Convolutional Networks

- Scale up neural networks to process very large images/video sequences
 - Sparse connections
 - Parameter sharing
- Automatically generalize across spatial translations of inputs
- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, ...)

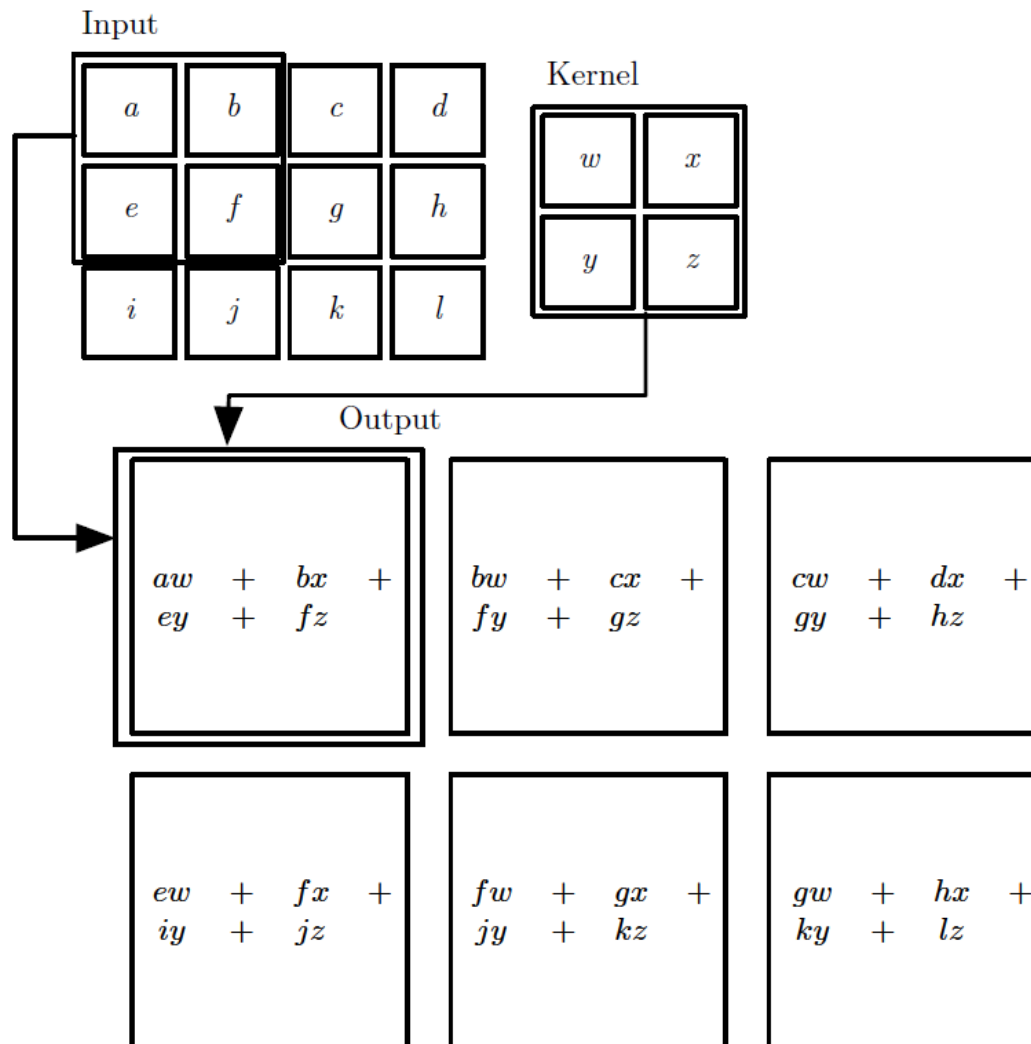


Key Idea

- Replace matrix multiplication in neural nets with convolution
- Everything else stays the same (with minor changes)
 - Maximum likelihood
 - Back-propagation
 - Etc.



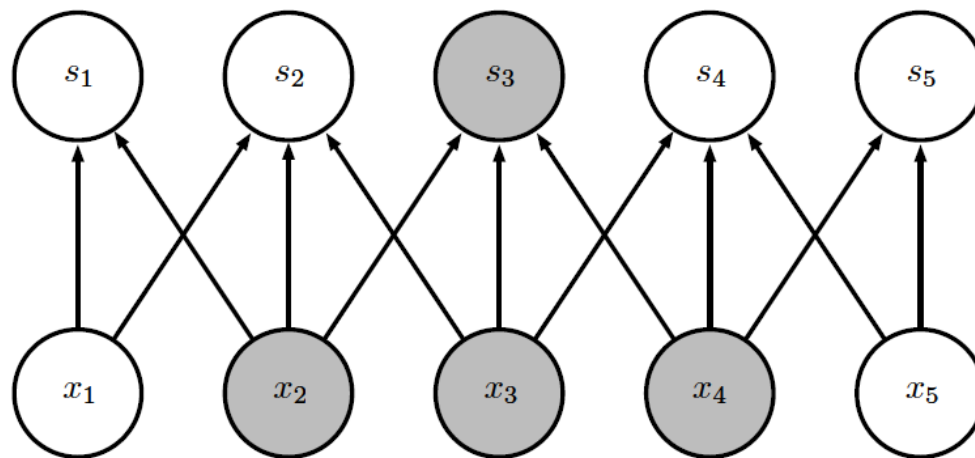
2D Convolution



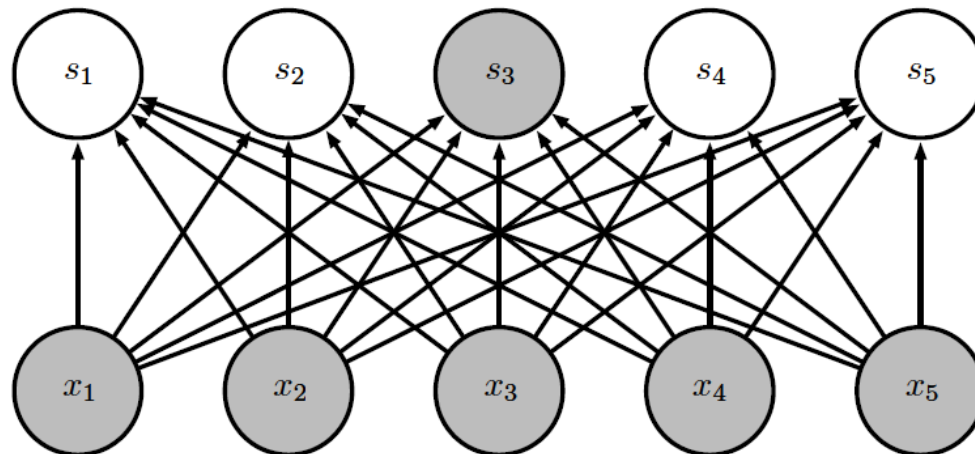


Sparse Connectivity

Sparse
connections
due to small
convolution
kernel



Dense
connections





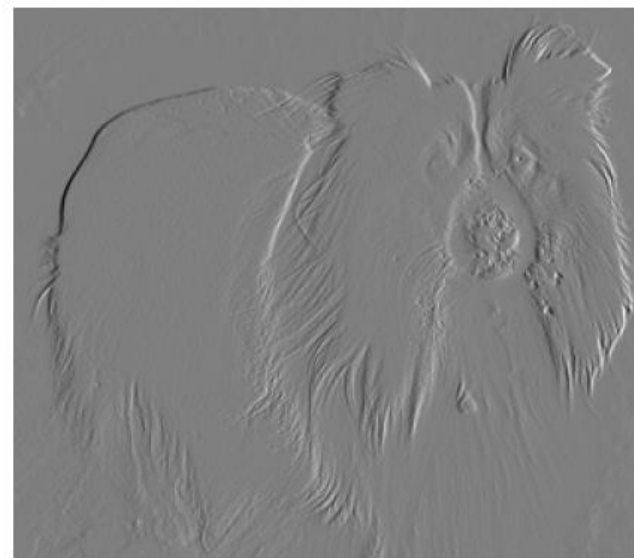
Edge Detection by Convolution



Input

1	-1
---	----

Kernel



Output



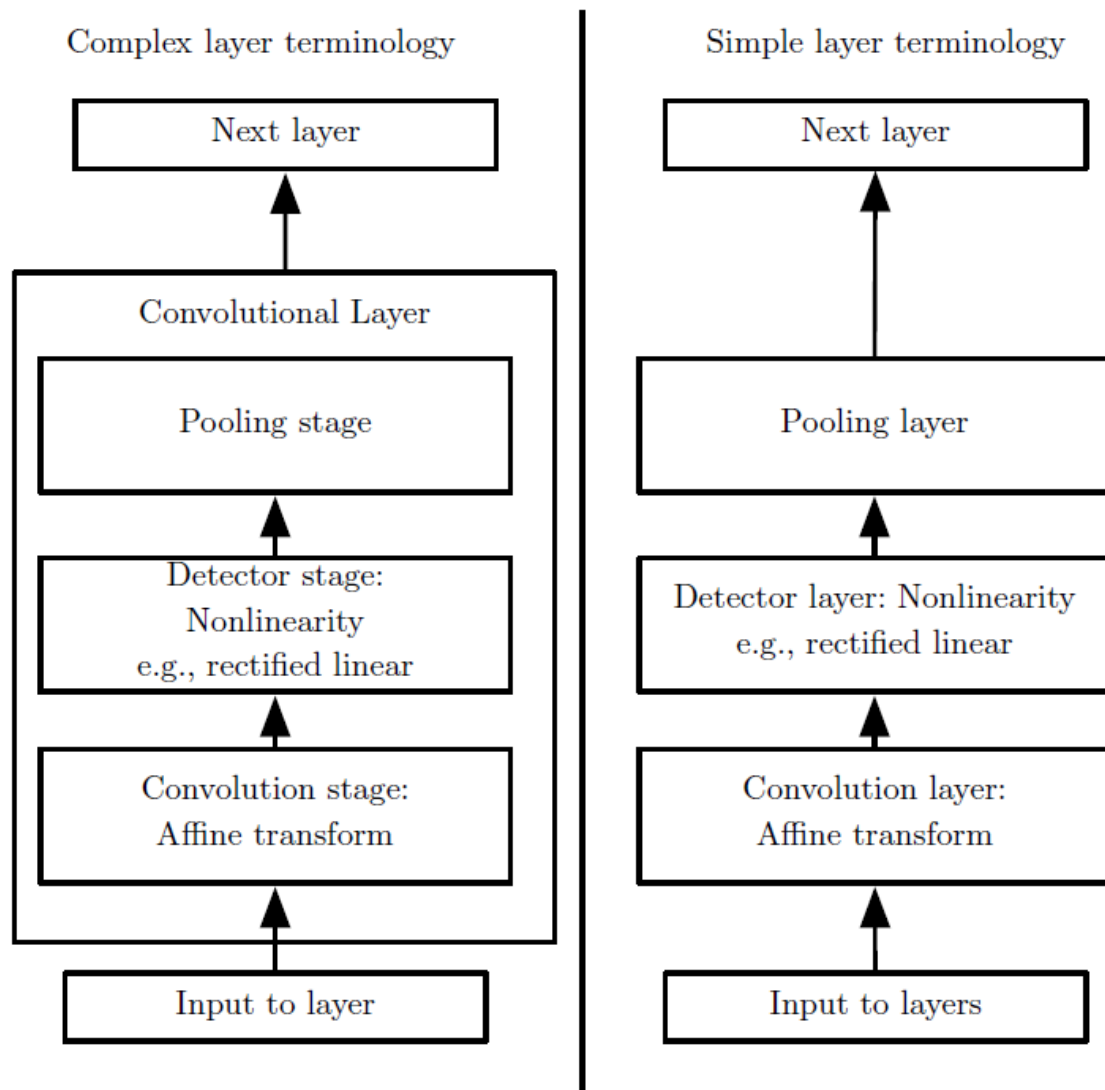
Efficiency of Convolution

- Input size: 320 by 280
- Kernel size: 2 by 1
- Output size: 319 by 280

	Convolution	Dense matrix	Sparse matrix
Stored floats	2	$319 \times 280 \times 320 \times 280$ $> 8e9$	$2 \times 319 \times 280 =$ 178,640
Float muls or adds	$319 \times 280 \times 3 =$ 267,960	$> 16e9$	Same as convolution (267,960)

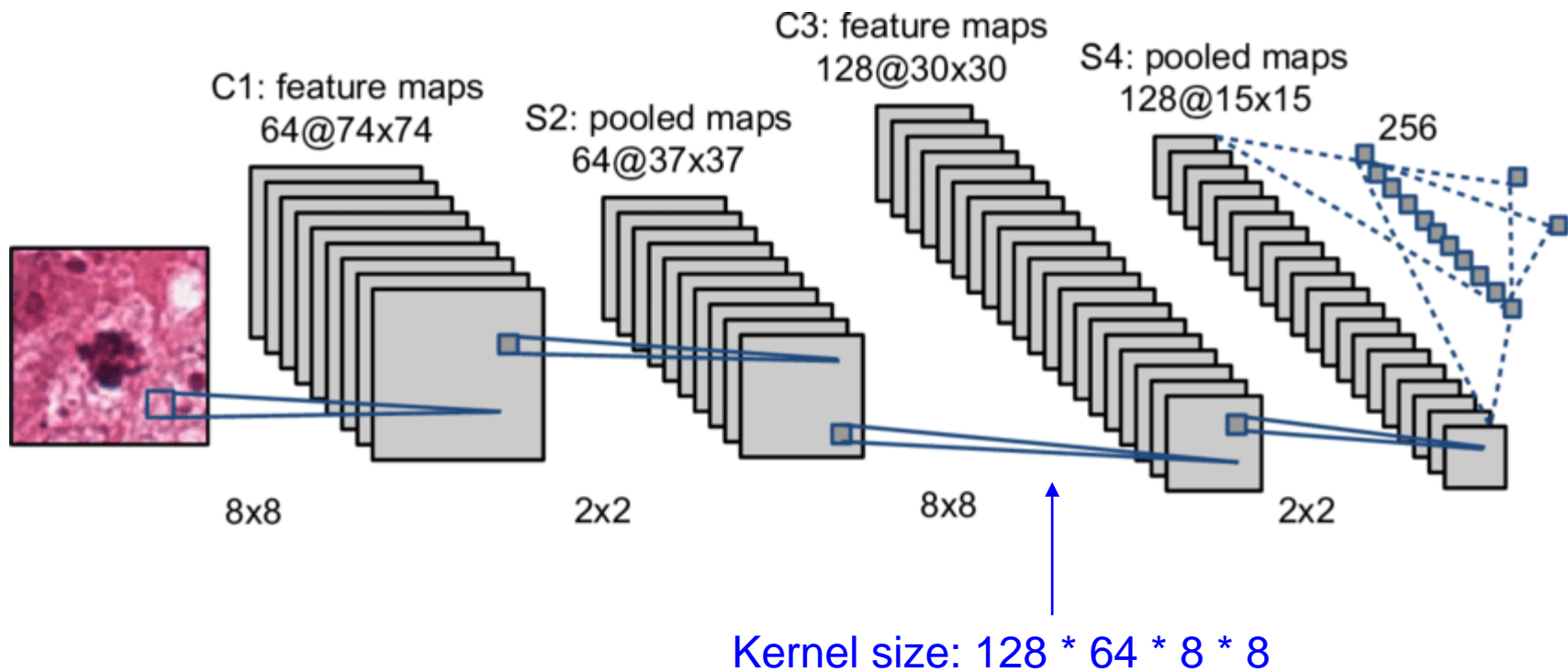


Convolutional Network Components





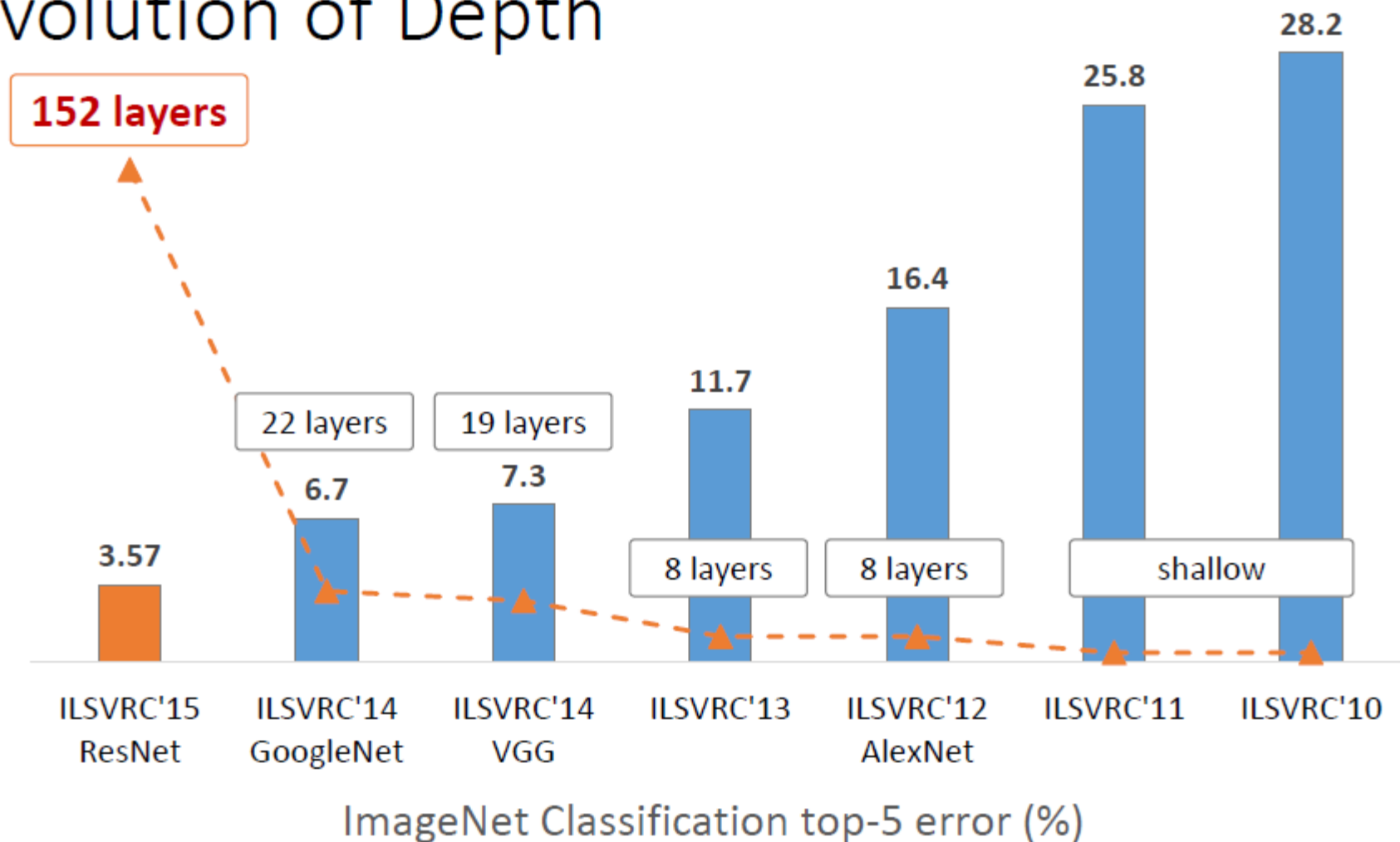
Feature Map





Major Architectures

Revolution of Depth



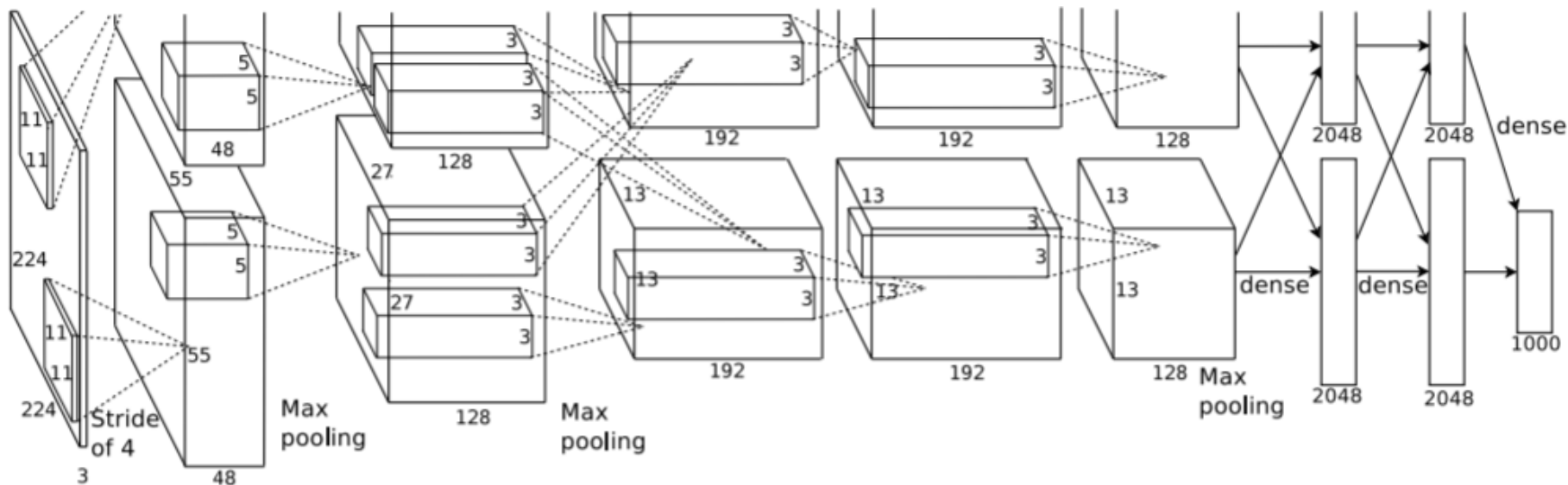
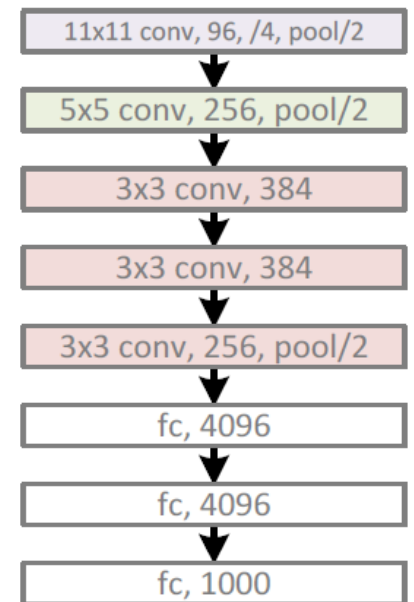
[Kaiming He]



Alexnet

■ 8 layers

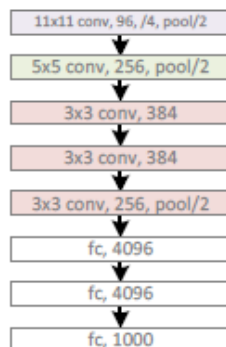
- ❑ 1st layer: filters 224 x 224 x 3 input image with 96 kernels of size 11 x 11 x 3 with a stride of 4 pixels (+max pooling)
- ❑ 2nd layer: filters the input with 256 kernels of 5 x 5 x 48 (+max pooling)
- ❑ 3rd layer: filters the input with 384 kernels of size 3 x 3 x 256
- ❑ ...
- ❑ 6, 7, 8th layers: fully connected layers



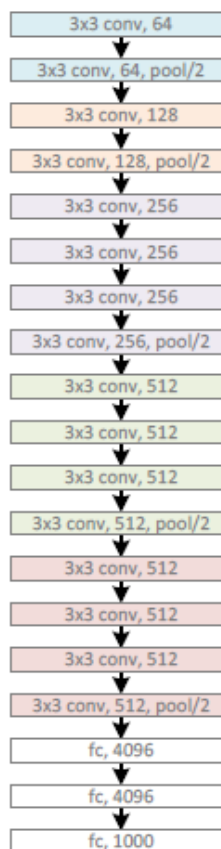


Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)





Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

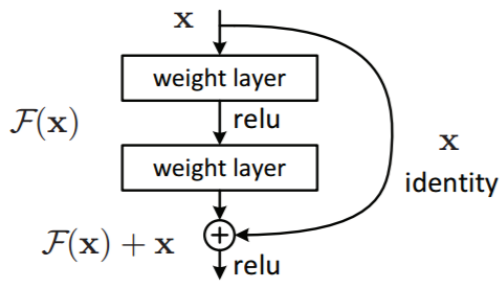


ResNet, 152 layers
(ILSVRC 2015)





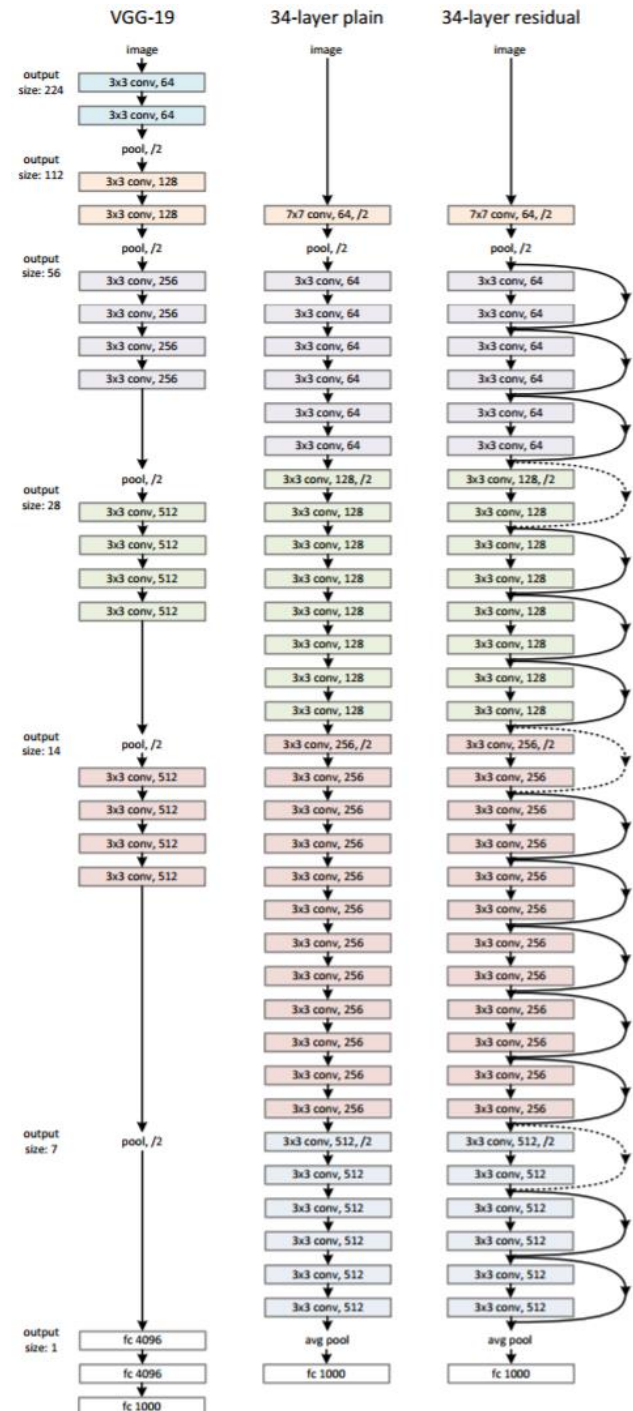
- Shortcut connection



Residual learning: a building block.


Let $H(x)$ be the desired underlying mapping.
ResNet lets the stacked nonlinear layers learn
 $F(x) = H(x) - x$.

It seems to be easier to optimize the residual mapping than to optimize the original mapping. If an identity mapping were optimal, then it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.





Outline

- ☒ What is Deep Learning?
- ☒ Feedforward Neural Network
- ☒ Convolutional Neural Network
-  ☒ **Recurrent Neural Network**
- ☐ Conclusion



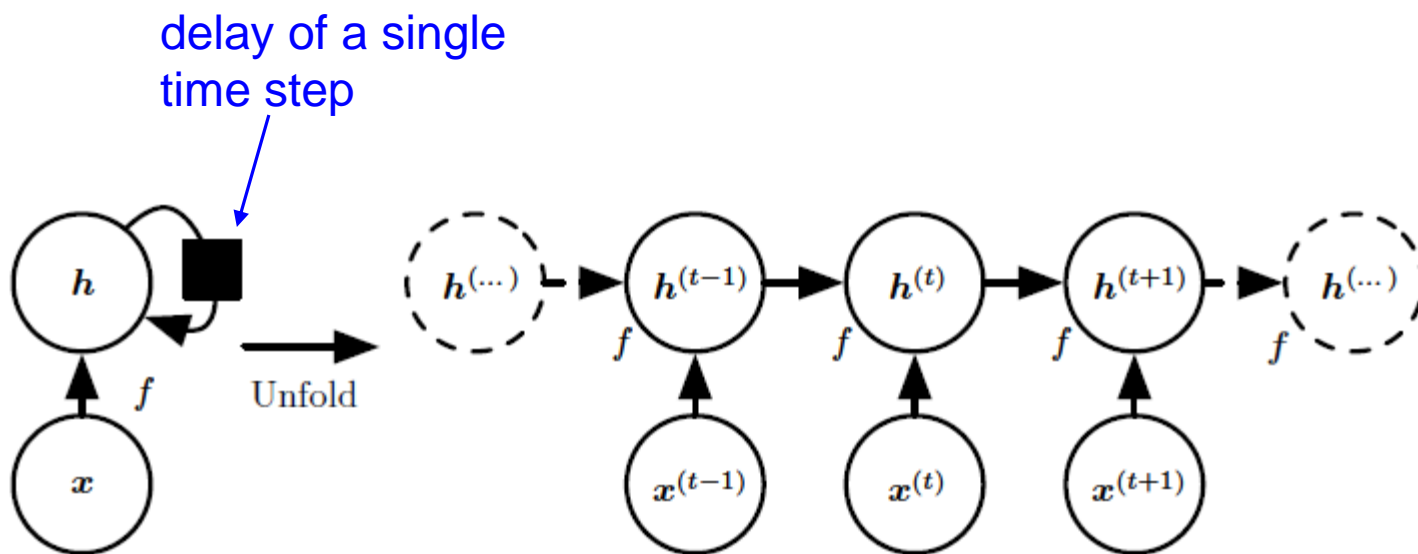
RNN

- Recurrent neural network (RNN)
 - A family of neural networks for processing sequential data
 - Can scale to much longer sequences than other networks do
 - Can process sequences of variable (or infinite) length



Unfolding Computation Graphs

- Consider a dynamical system driven by an external signal $x^{(t)}$
 - $h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$

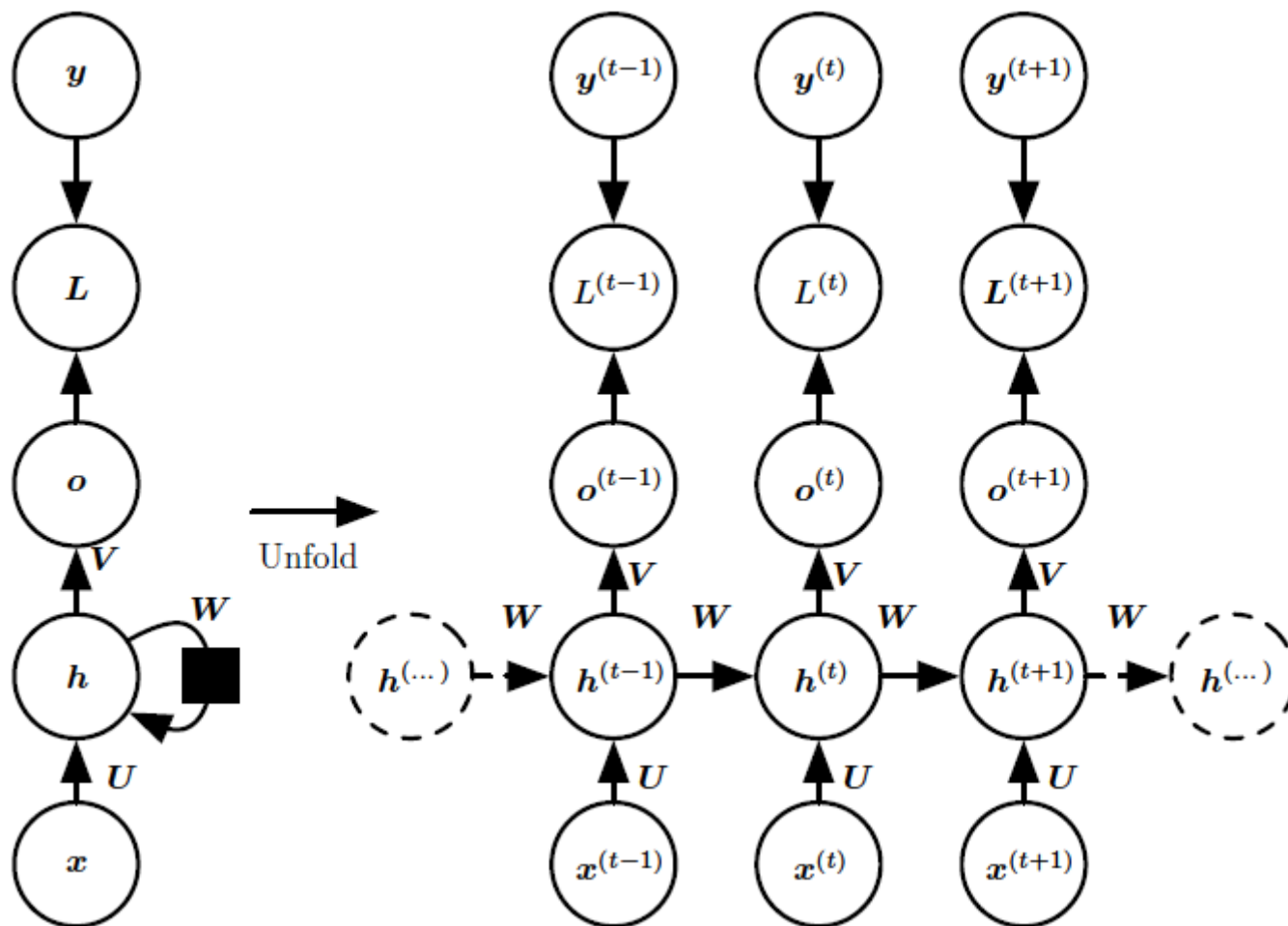


recurrent graph
or circuit diagram

unrolled graph



Recurrent Hidden Units

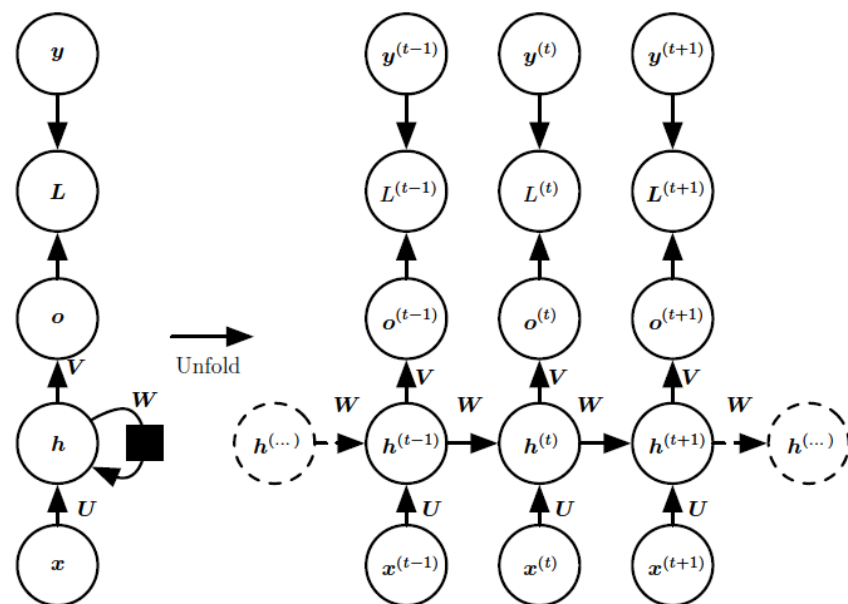




Equations for RNN

- $\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}$
- $\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$
- $\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$
- $\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$
- The total loss is the sum of the losses over all time steps:

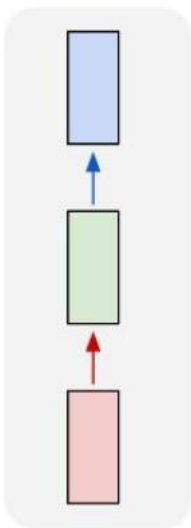
$$\begin{aligned} \square \quad & L(\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)}\}, \{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(\tau)}\}) \\ &= \sum_t L^{(t)} \\ &= -\sum_t \log p_{\text{model}}(\mathbf{y}^{(t)} | \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)}\}) \end{aligned}$$





Types of RNN

one to one



one to many

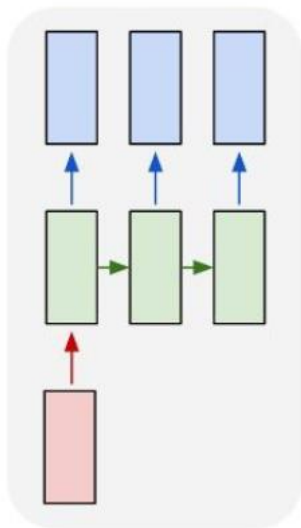
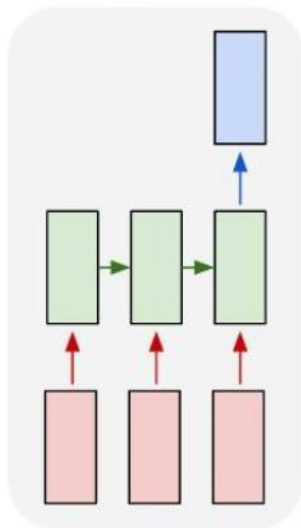


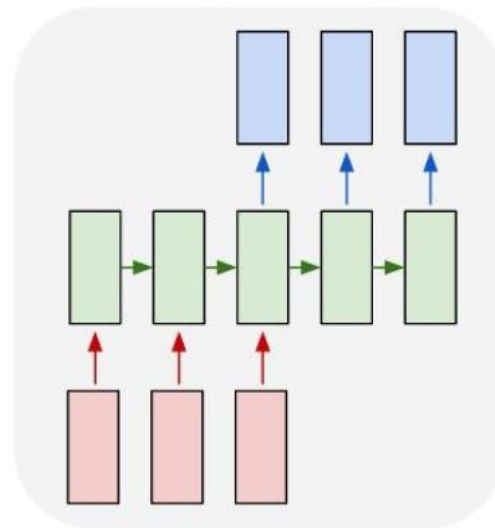
Image
Captioning

many to one



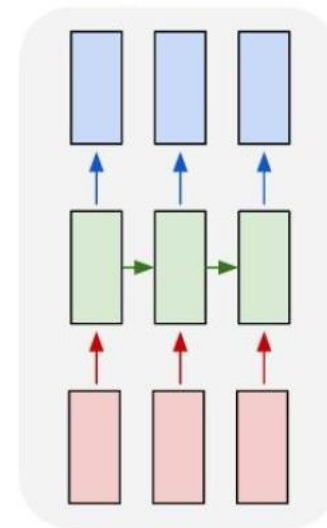
Sentiment
Classification

many to many



Machine
Translation

many to many

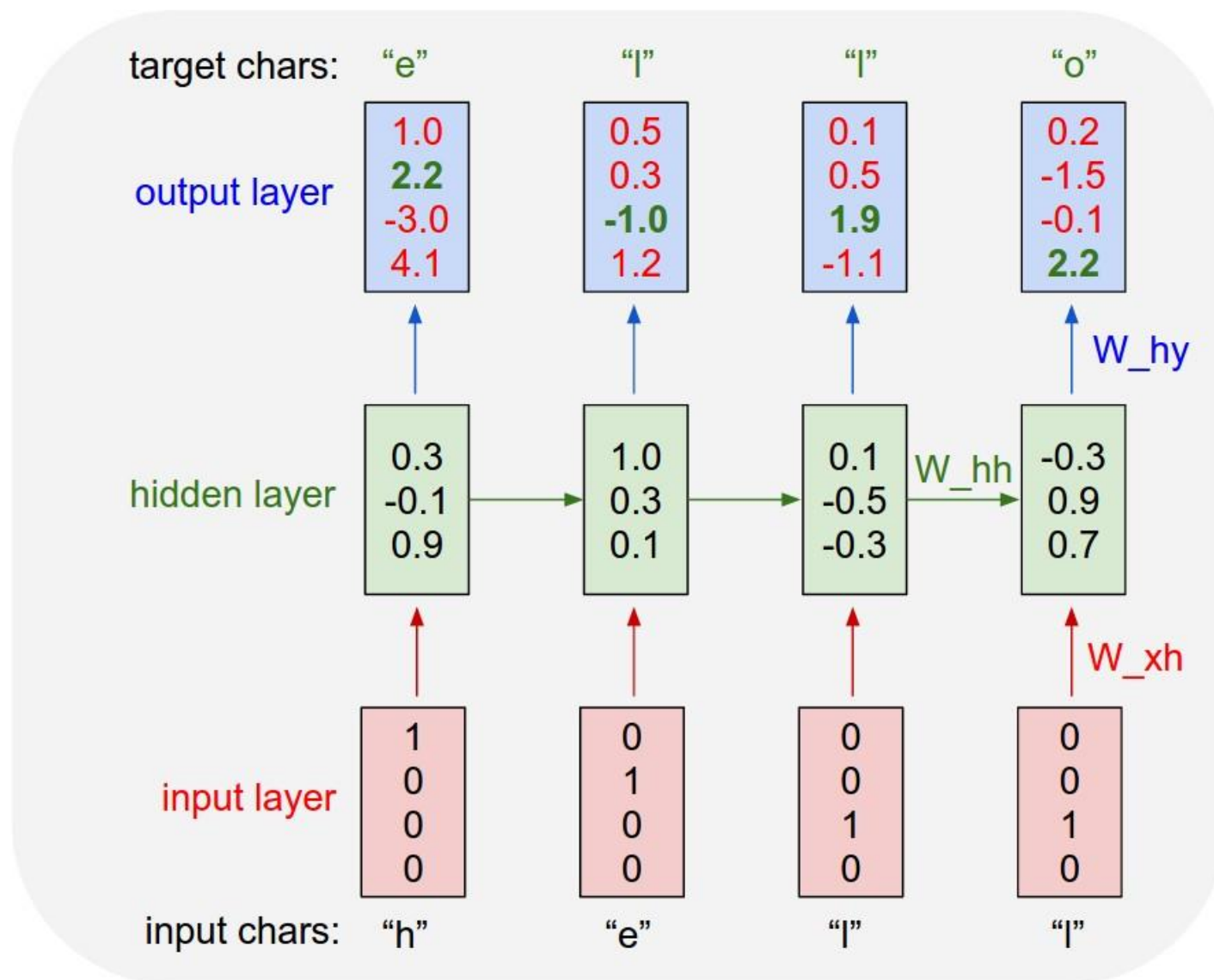


Video
Classification
on frame level

[Andrej Karpathy]




Character-Level Language Model



[Andrej
Karpathy]



Outline

- ☒ What is Deep Learning?
- ☒ Feedforward Neural Network
- ☒ Convolutional Neural Network
- ☒ Recurrent Neural Network
-  ☐ **Conclusion**



Conclusion

- Deep learning
 - A machine learning method motivated from the processing of signals in brain
- Major architecture
 - Feedforward Neural Network (FNN)
 - Convolutional Neural Network (CNN)
 - Recurrent Neural Network (RNN)



Thank You!