

Deep Learning & Convolutional neural networks (CNN)

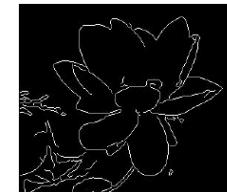
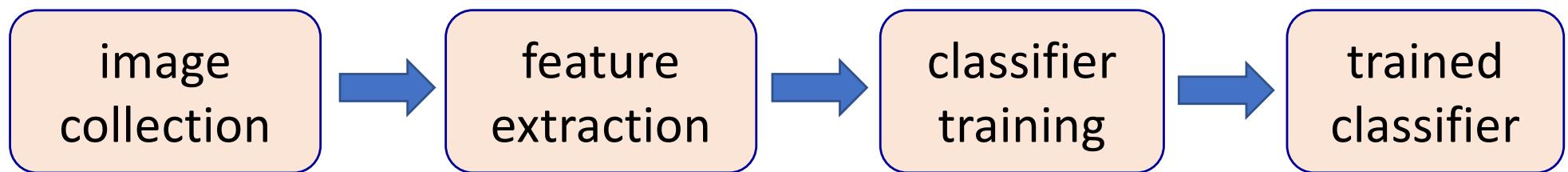
Slide from:

1. Y. Y. Lin

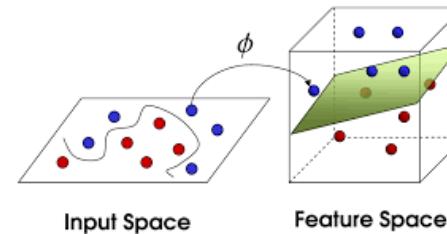
2. H. Y. Lee <http://speech.ee.ntu.edu.tw/~tlkagk/courses.html>

Conventional approach to object recognition

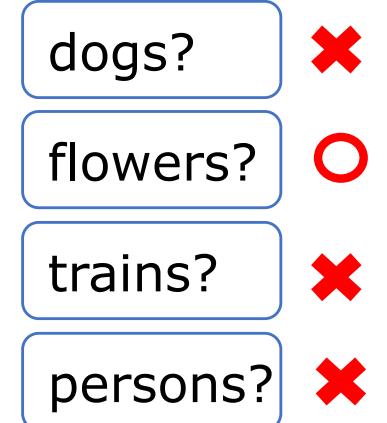
- Training phase



histogram of oriented
gradients (HoG)

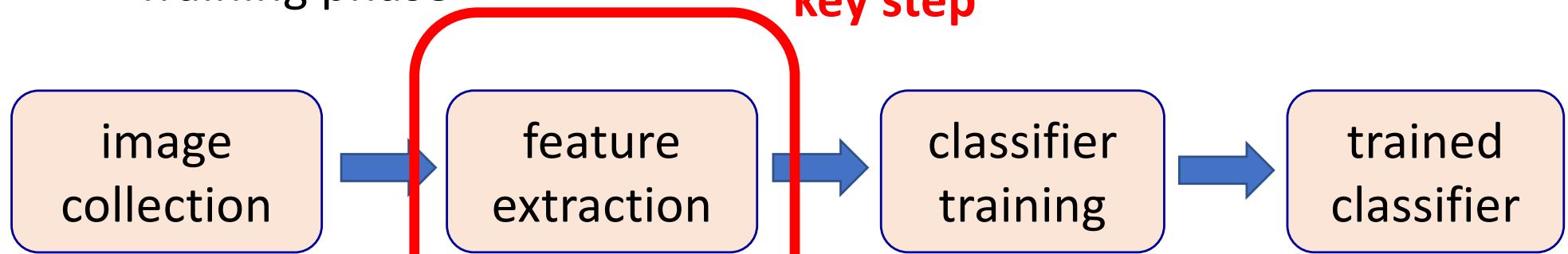


support vector
machines (SVMs)

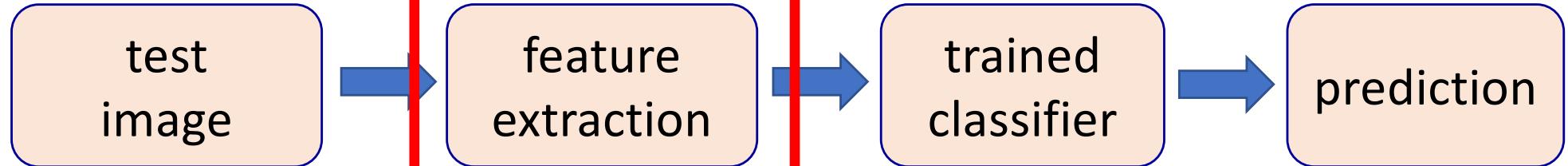


Conventional approach to object recognition

- Training phase



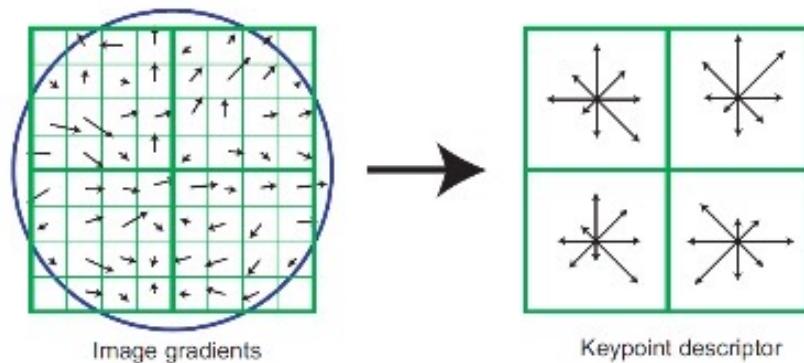
- Testing phase



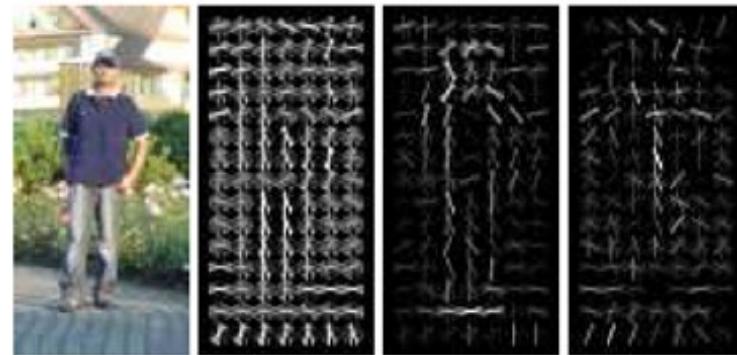
abrechne

Features are the keys

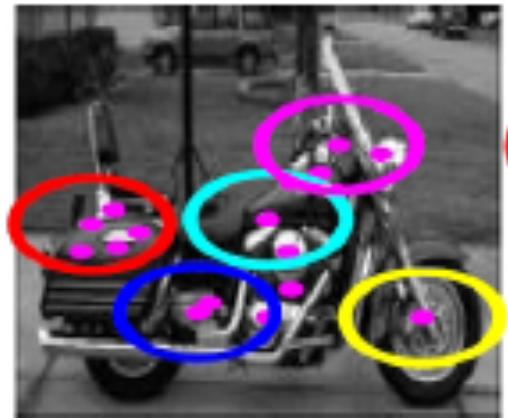
- Off-the-shelf visual features



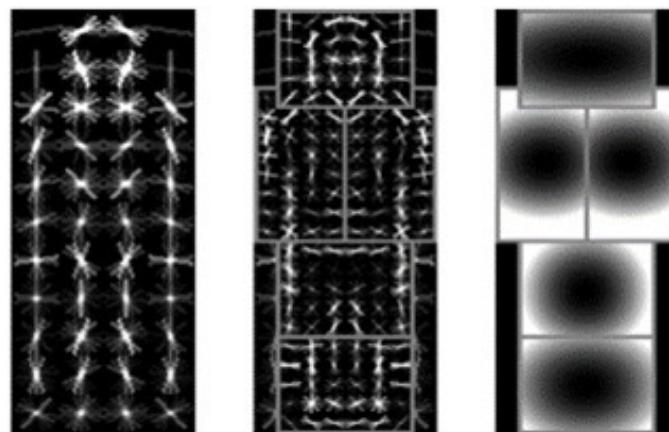
SIFT [Lowe, IJCV'04]



HoG [Dalal & Triggs, CVPR'05]



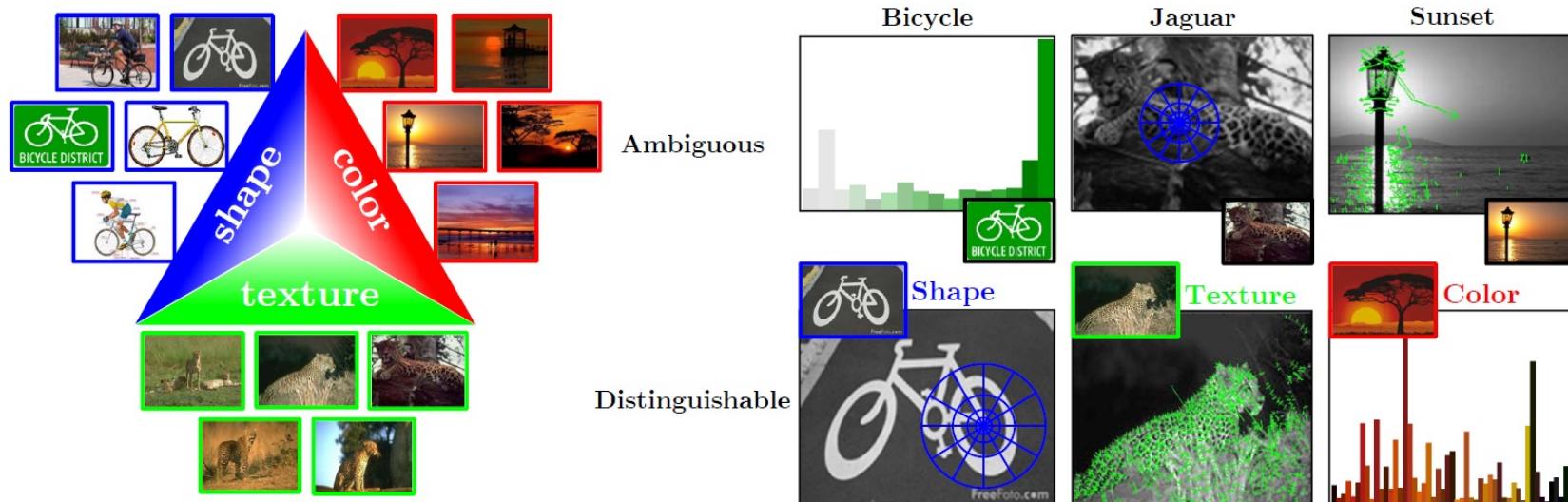
Constellation model [Fergus et al., CVPR'03]



DPM [Felzenszwalb et al., PAMI'10]

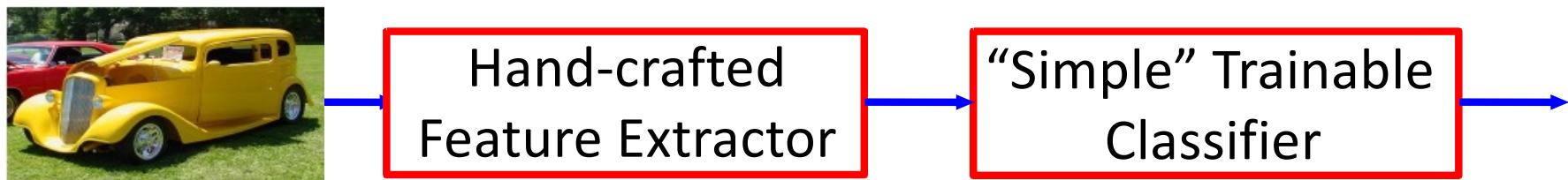
Features are the keys

- Features are the keys to recent progress in classification
- Are handcrafted features optimal?
- The optimal features for classification in general vary from task to task, even from category to category

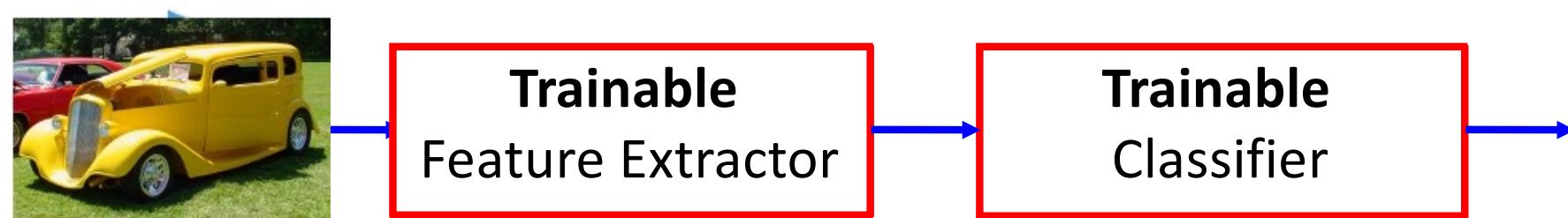


Conventional approaches vs. Deep learning

- Conventional approaches
 - **Fixed/engineered** features + trainable classifier

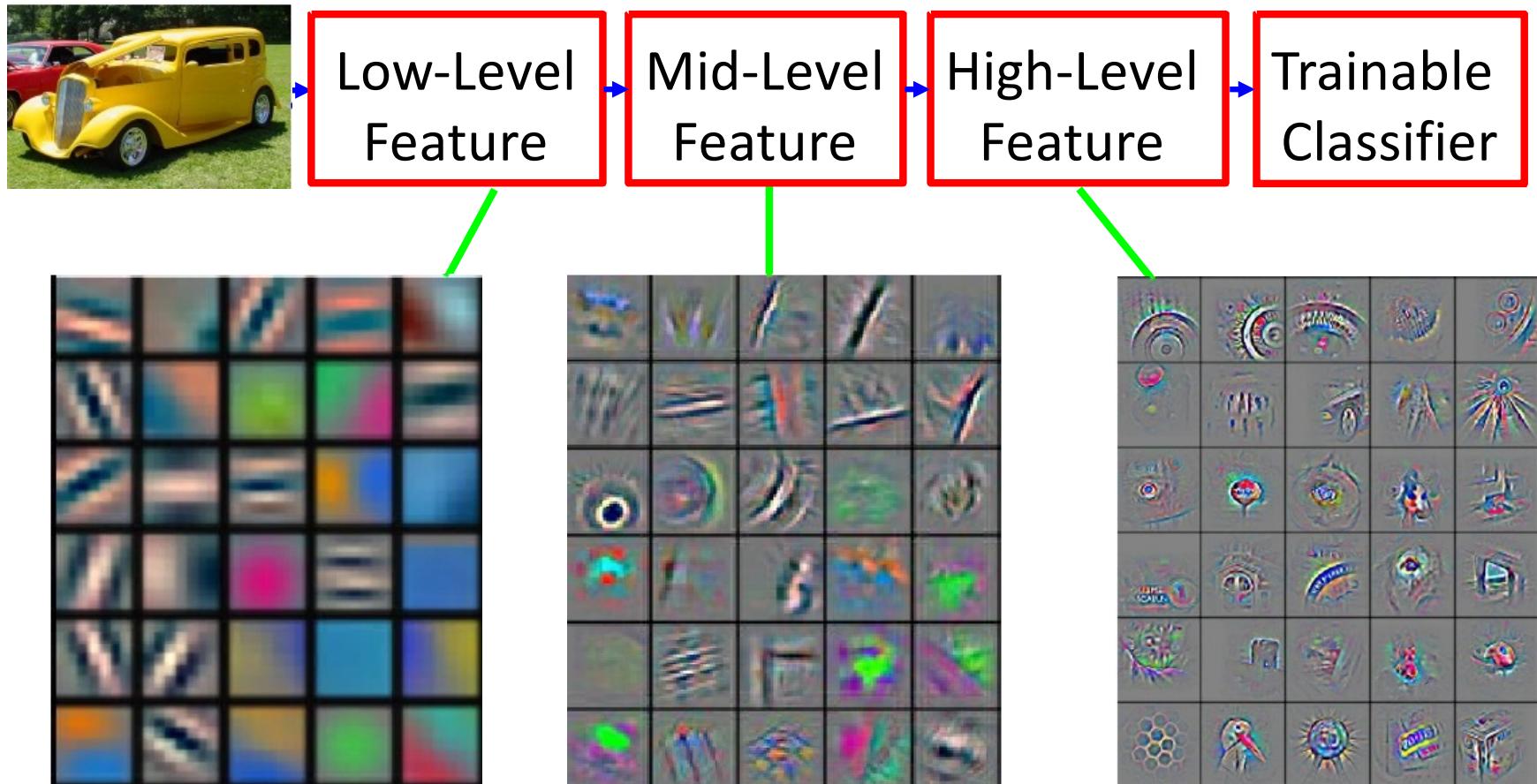


- Deep learning / End-to-end learning / Feature learning
 - **Trainable** features + trainable classifier



slide: Y LeCun & MA Ranzato

Deep learning = Learning hierarchical representations

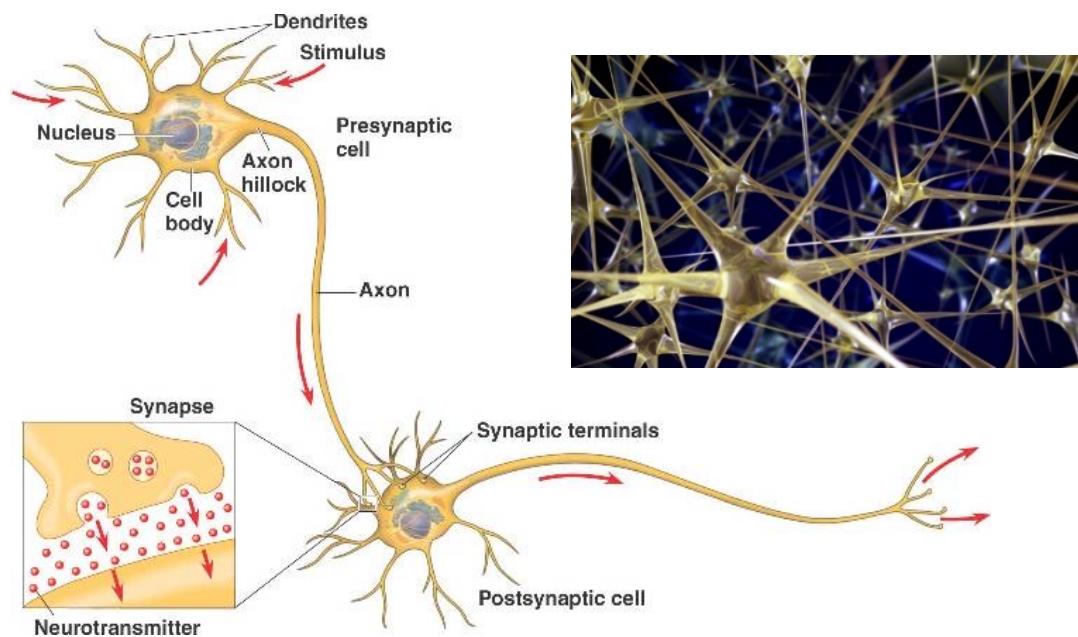


slide: Y LeCun & MA Ranzato

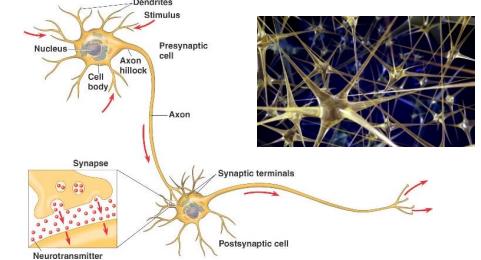
Deep Neural Networks (DNN)

Neural networks and neurons

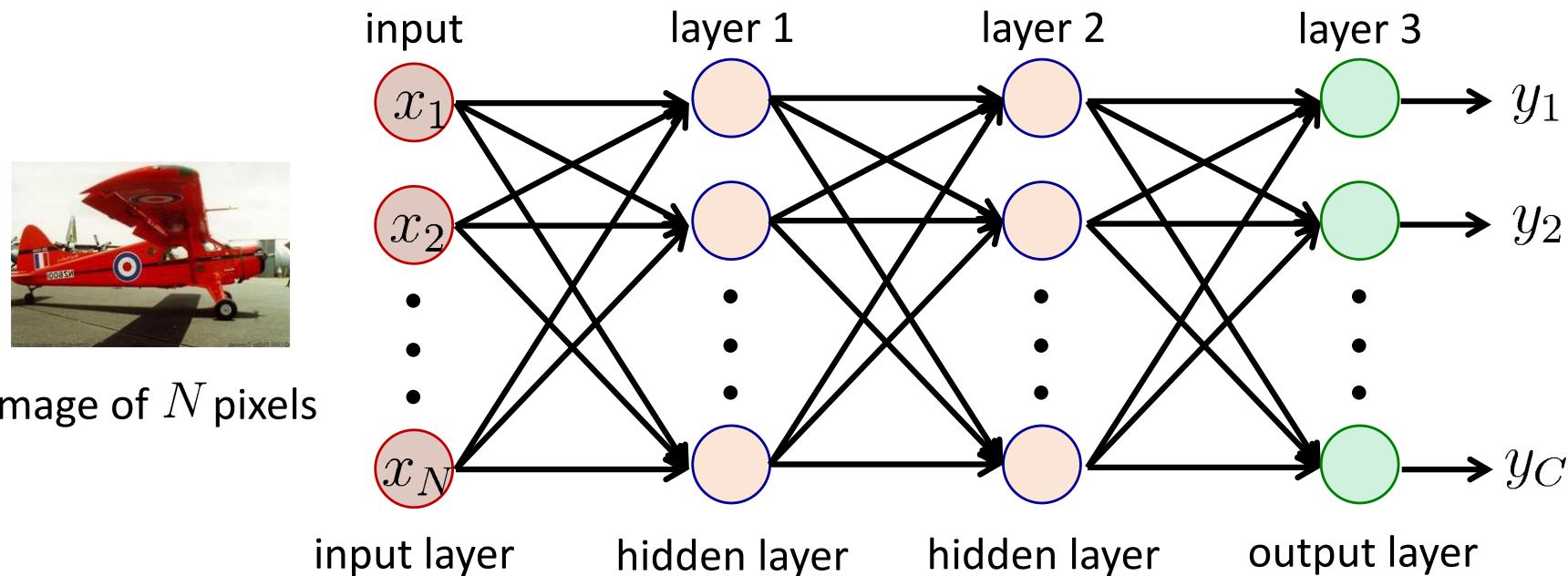
- Neural networks are presented as layers of interconnected neurons



Neural networks and neurons

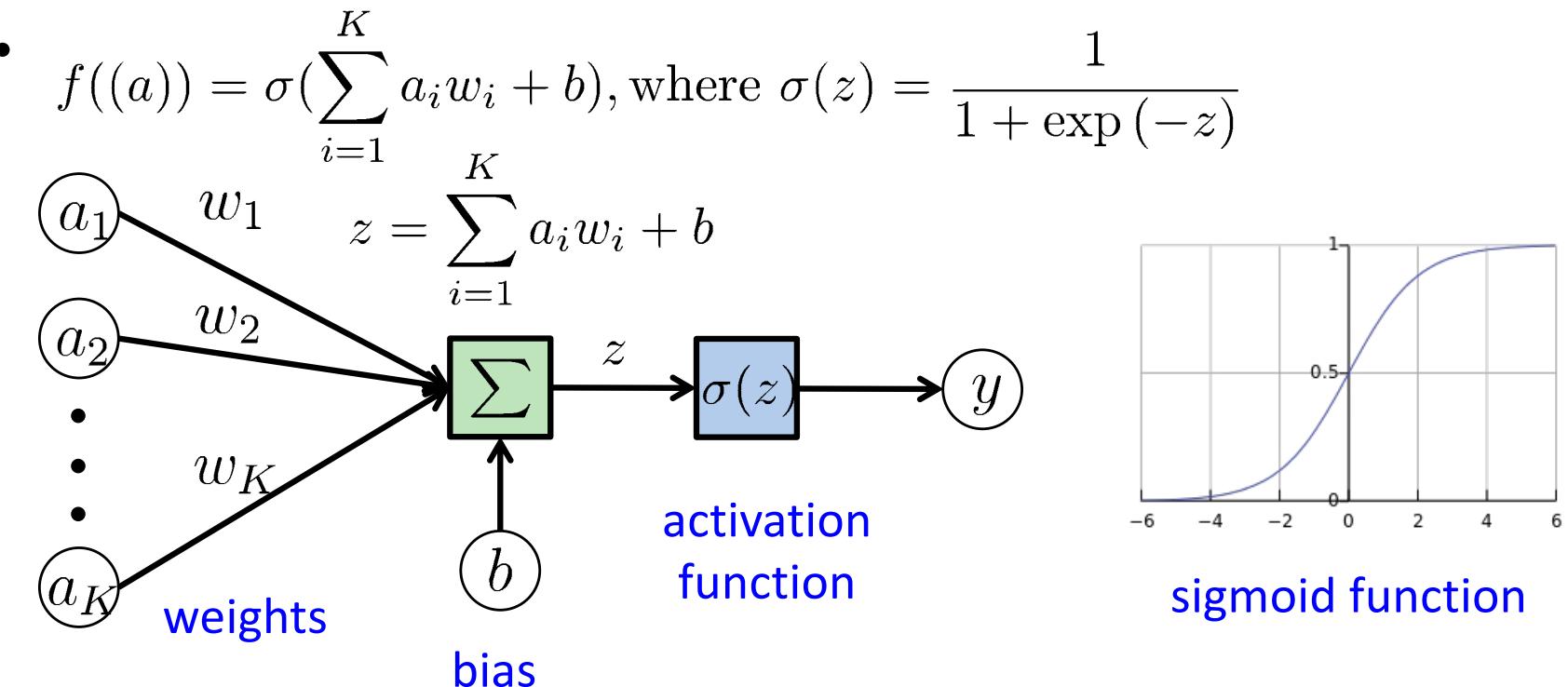


- Neural networks are presented as layers of interconnected neurons
 - Each layer of neurons takes messages from output of previous layer



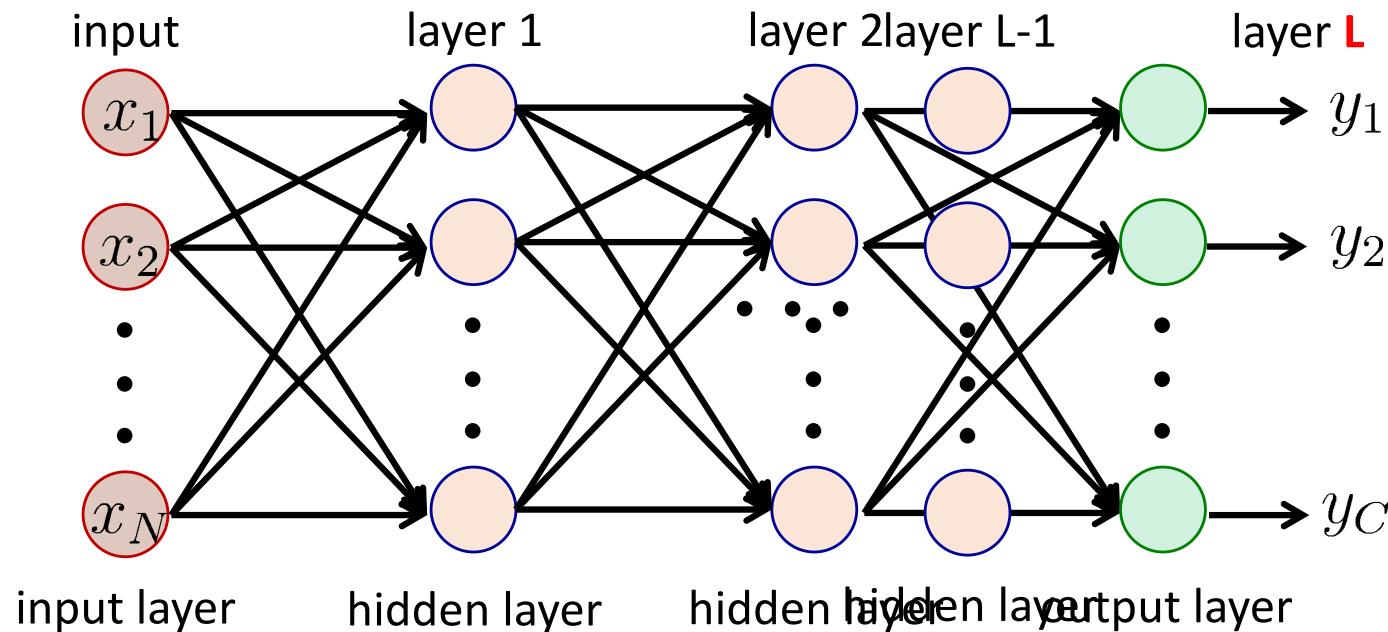
A single neuron

- A function $f : R^K \mapsto R$
 - Map K inputs to 1 output
 - Compute the biased weighted sum
 - Apply a non-linear mapping function (activation function)



What is deep neural networks (DNN)

- DNN is neural networks with many hidden layers



Ups and downs of Deep Learning

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
 - Do not have significant difference from DNN today
- 1986: Backpropagation
 - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM (Restricted Boltzmann Machine) initialization
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC image competition
- 2015.2: Image recognition surpassing human-level performance

Ups and downs of Deep Learning

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
 - Do not have significant difference from DNN today

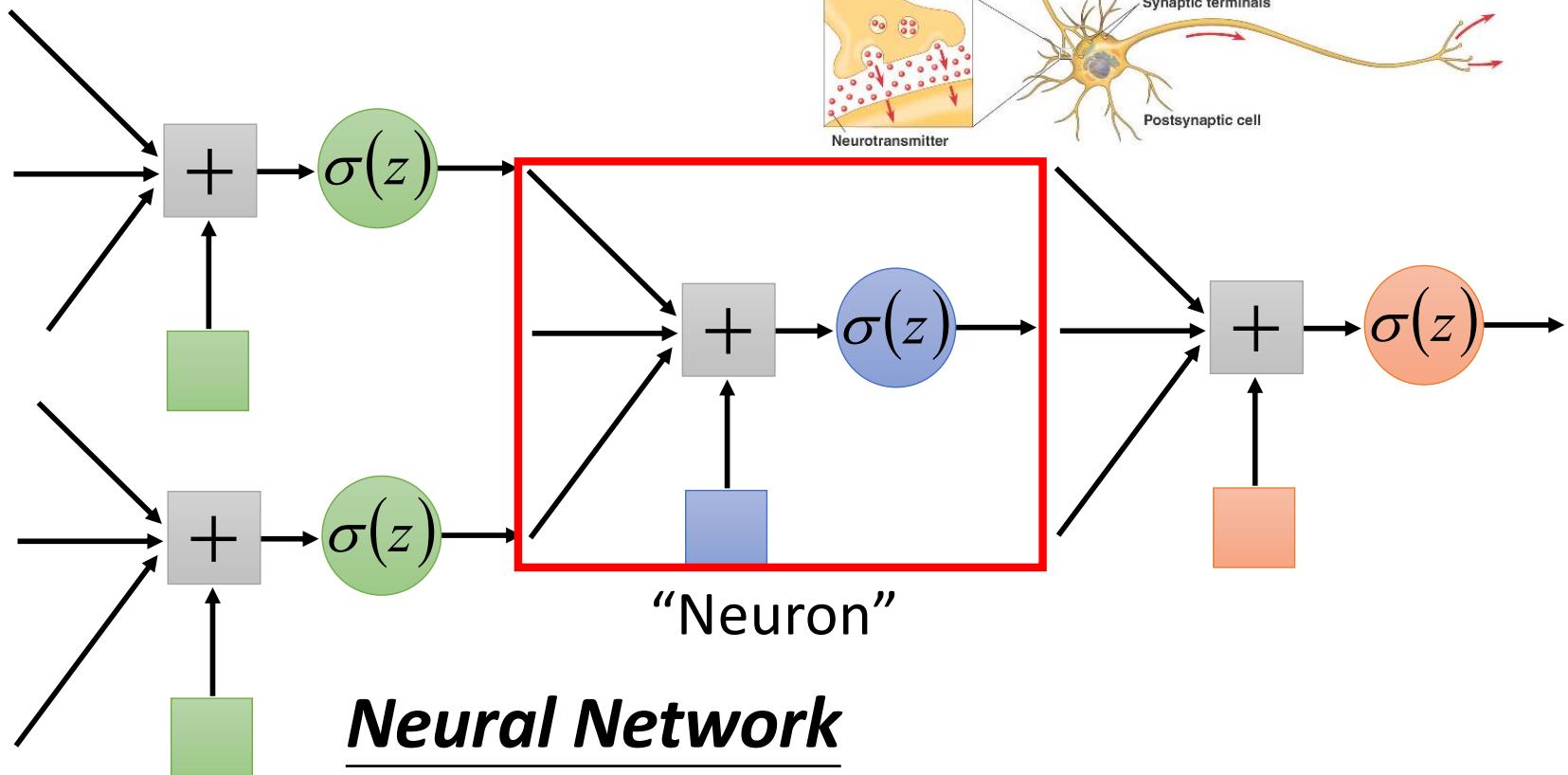
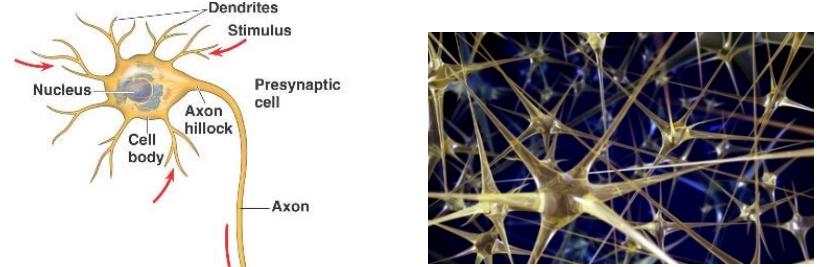


A classification problem

- 2011: Start to be popular in speech recognition

- 2012: win ILSVRC image competition
- 2015.2: Image recognition surpassing human-level performance
- 2016.3: Alpha GO beats Lee Sedol
- 2016.10: Speech recognition system as good as humans

Neural Network

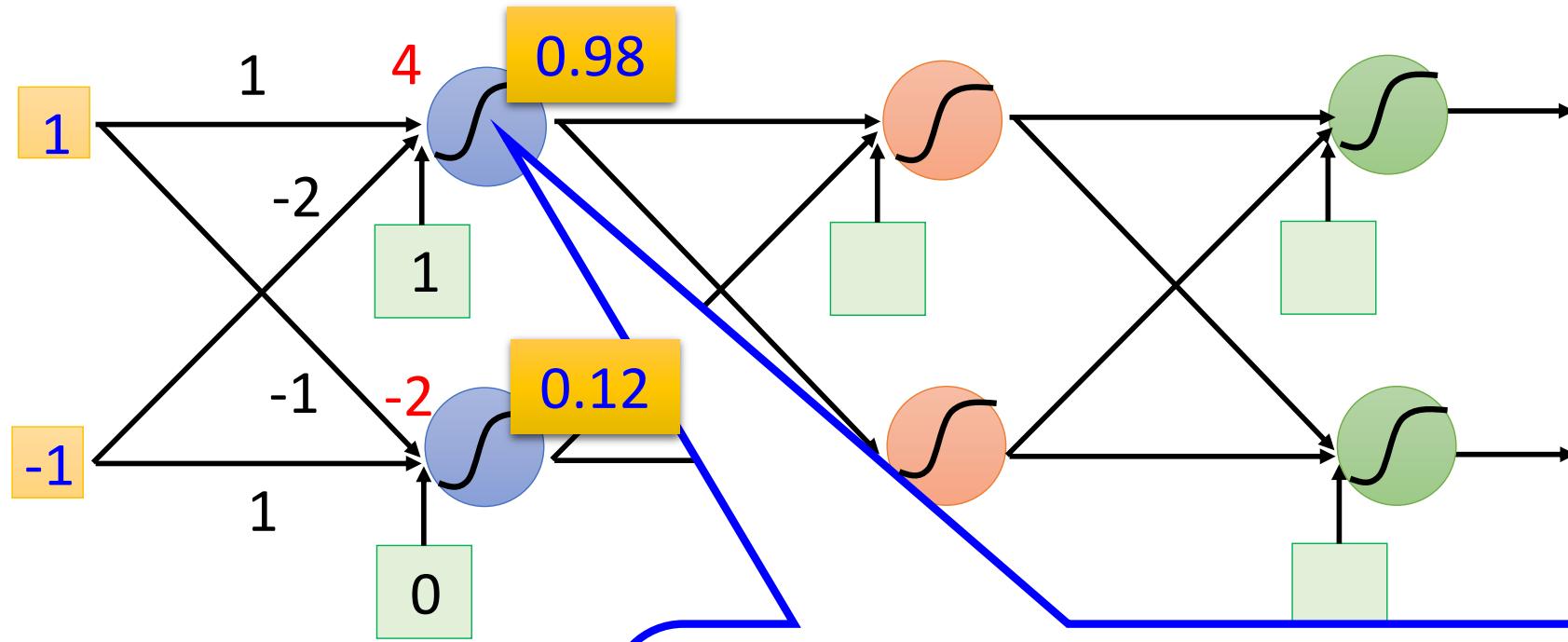


Neural Network

Different connection leads to different network structures

Network parameter θ : all the weights and biases in the “neurons”

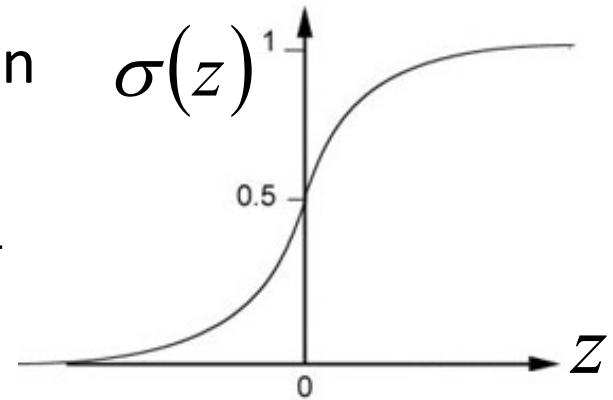
Fully Connect Feedforward Network



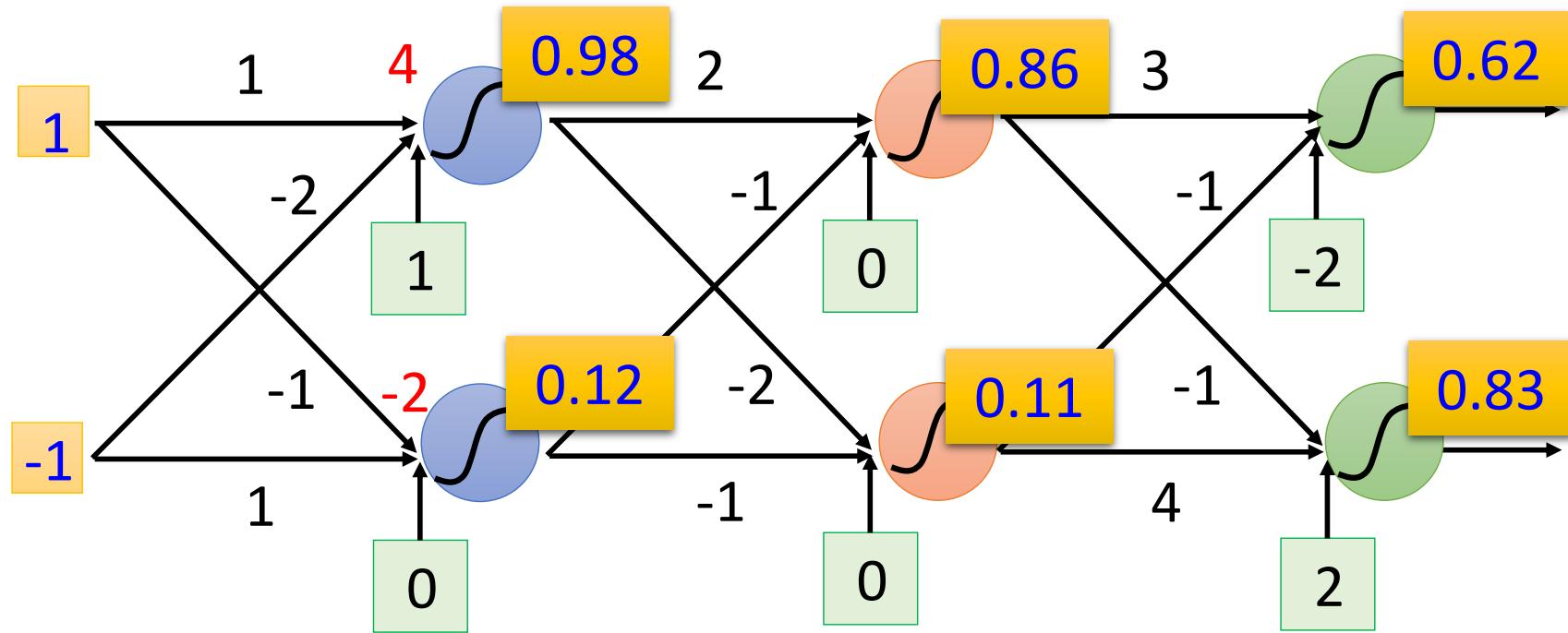
$$z = \sum_{i=1}^K a_i w_i + b$$

Sigmoid Function

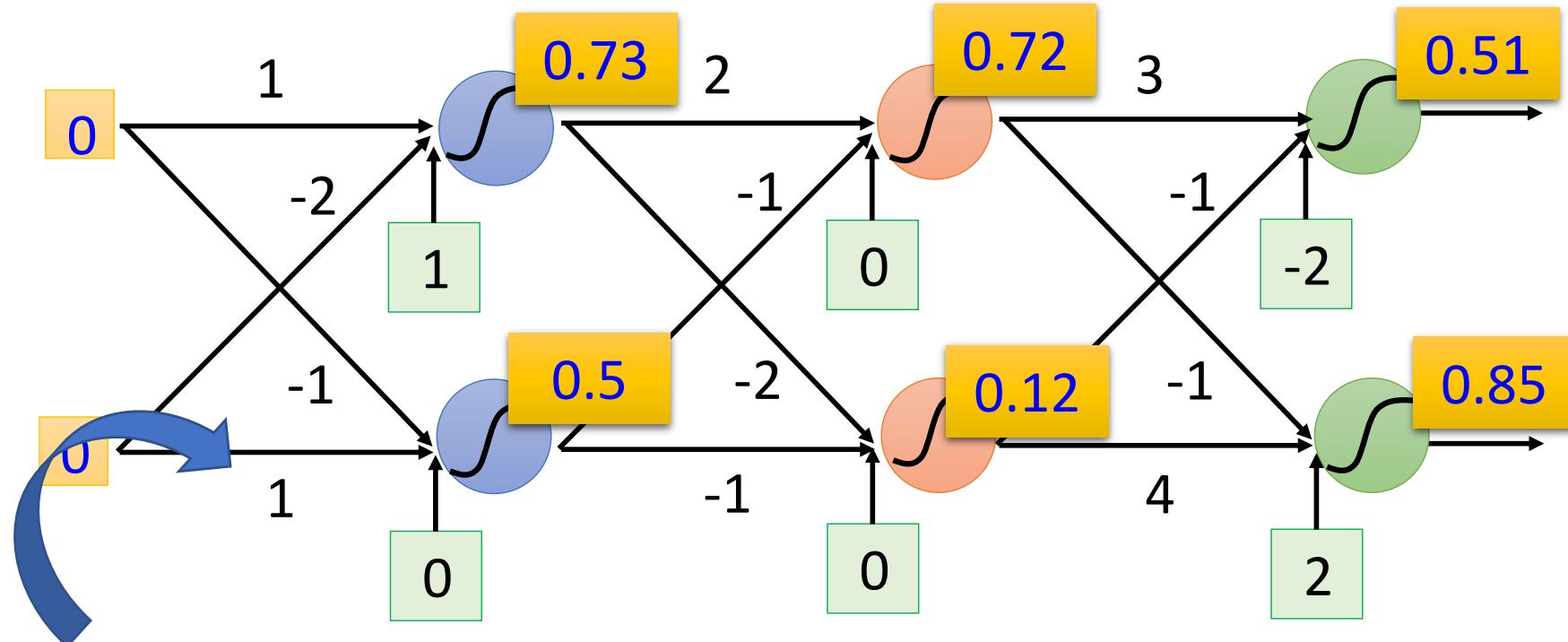
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Fully Connect Feedforward Network



Fully Connect Feedforward Network

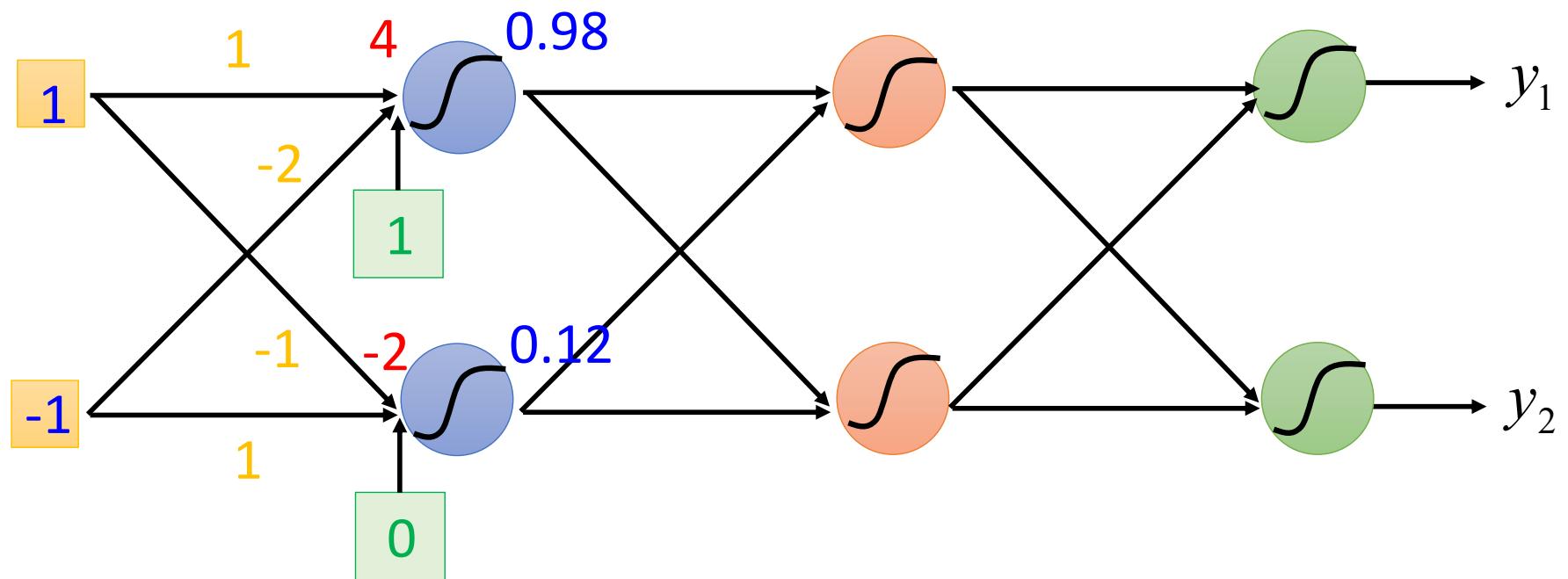


This is a function.
Input vector, output vector

$$f \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix}$$

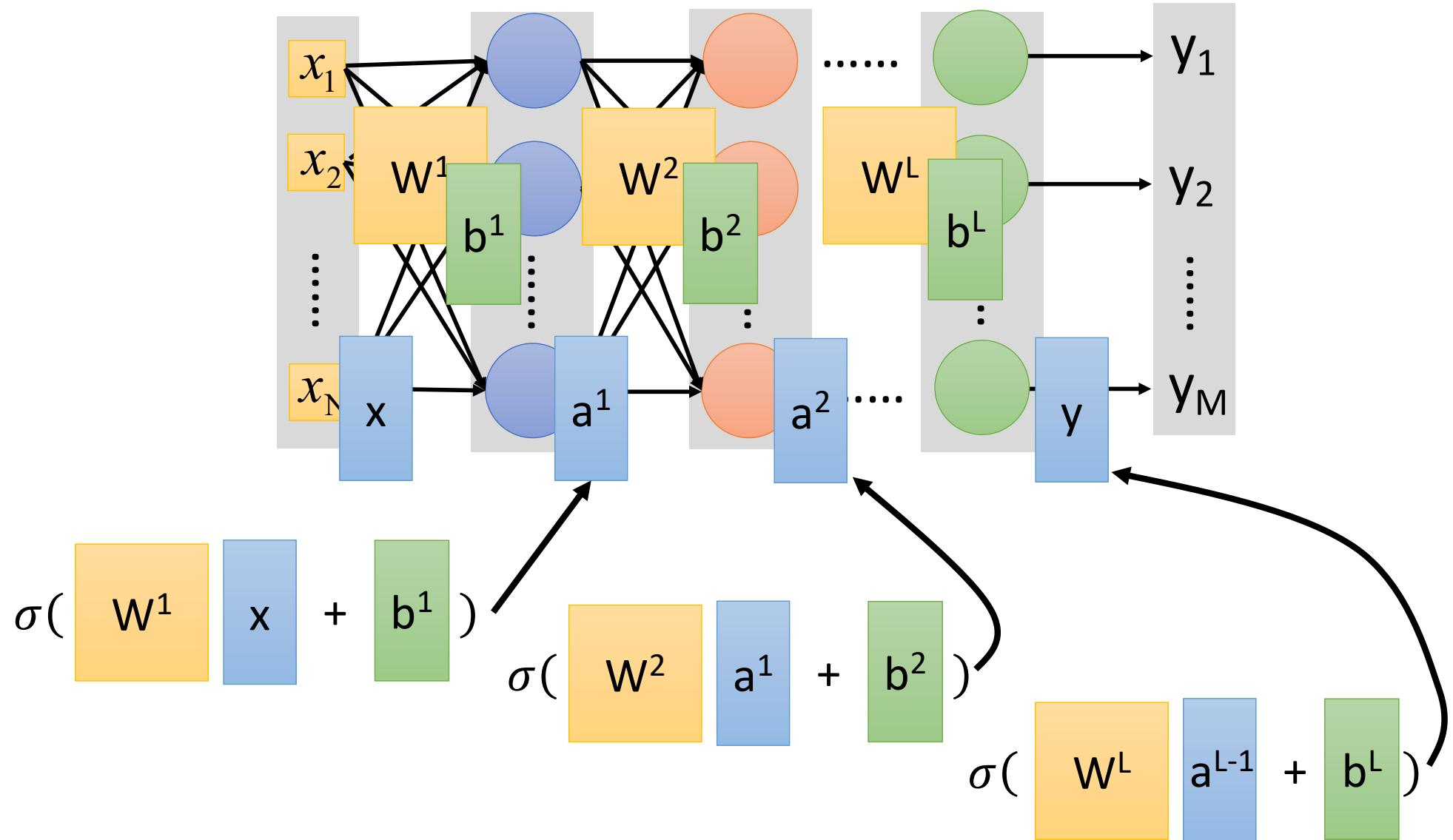
Given network structure, define a function set

Matrix Operation

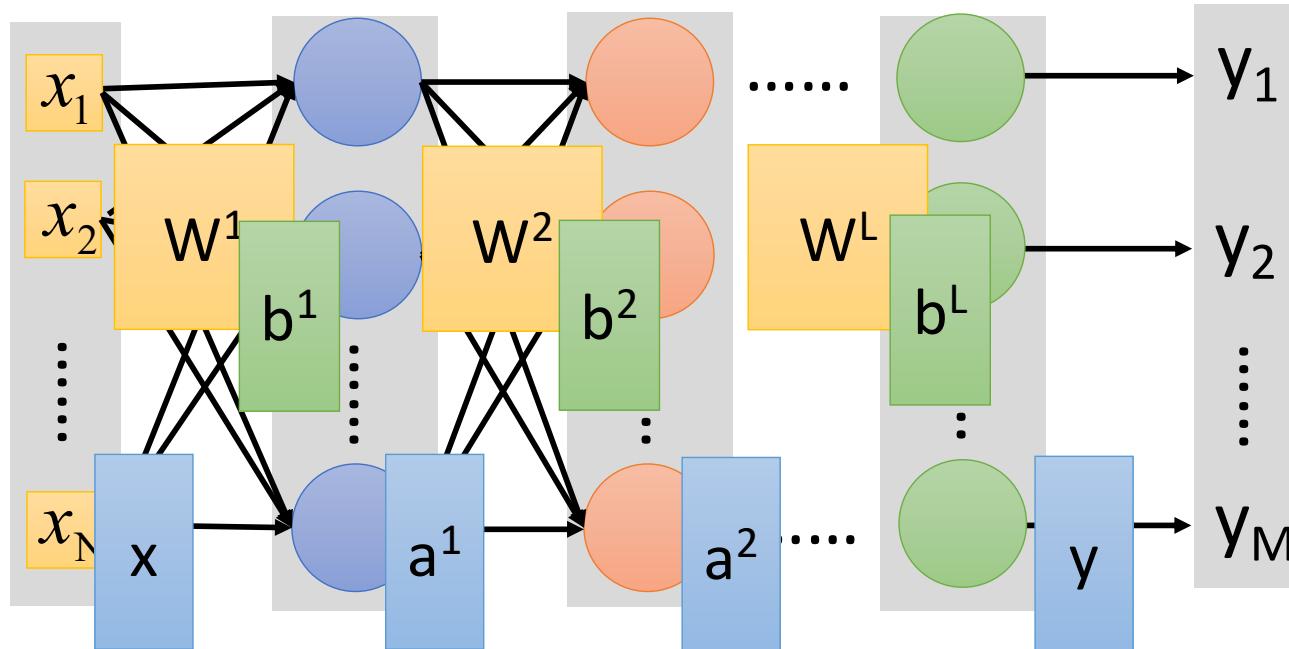


$$\sigma \left(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}} \right) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Neural Network



Neural Network



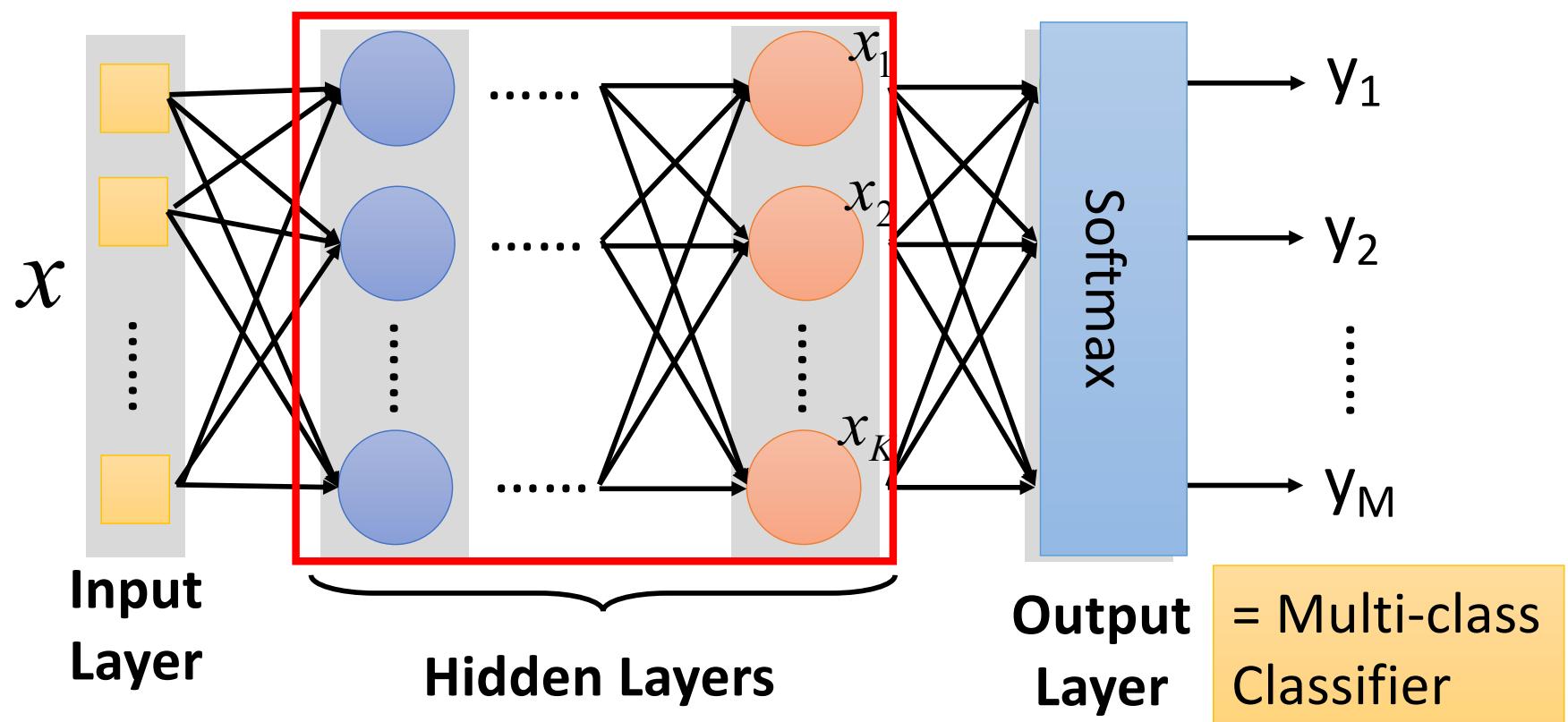
$$y = f(x)$$

Using parallel computing techniques
to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Output Layer as Multi-Class Classifier

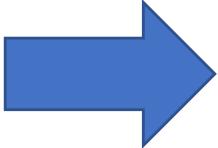
Feature extractor replacing
feature engineering



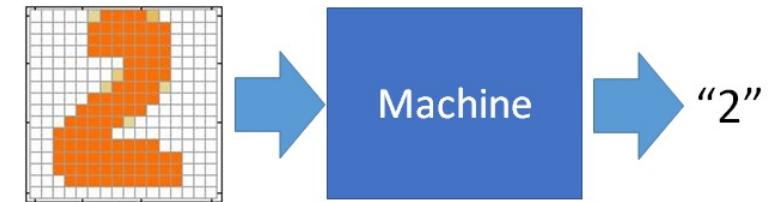
Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K.$$

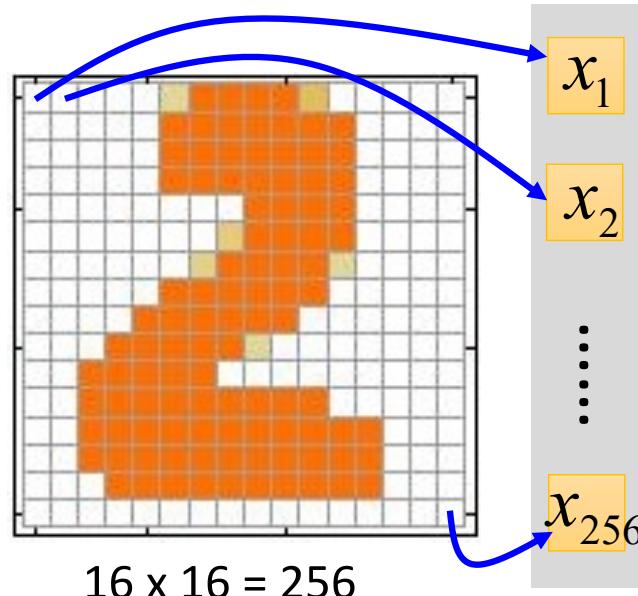
EX.

1.0		0.0236405
2.0		0.0642617
3.0		0.174681
4.0		0.474833
1.0		0.0236405
2.0		0.0642617
3.0		0.174681

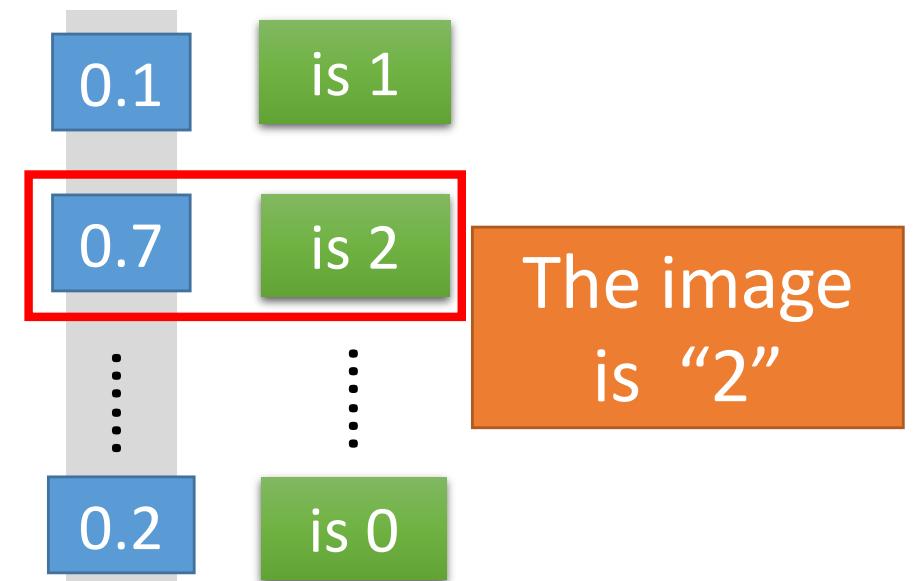
Example Application



Input



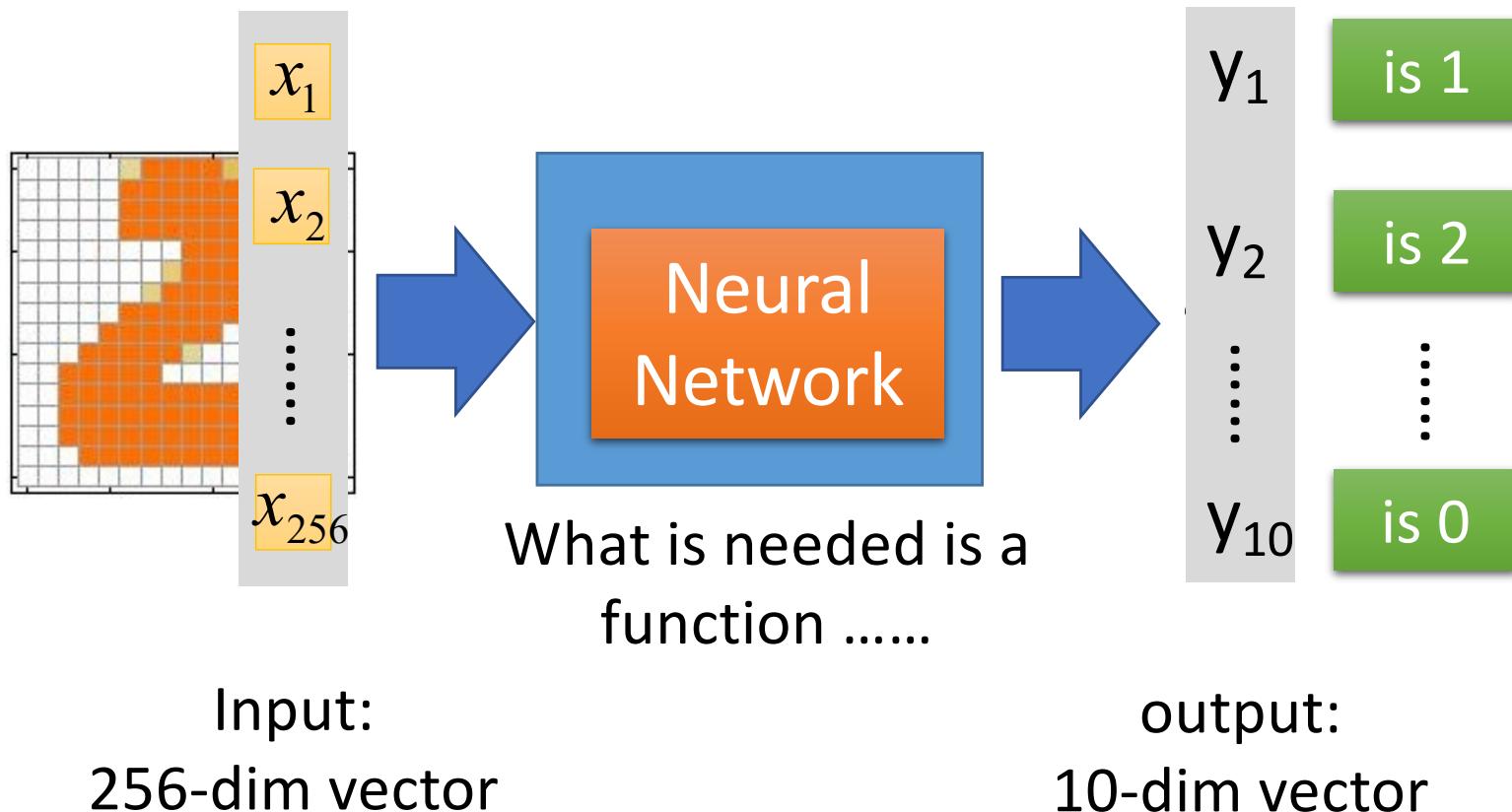
Output



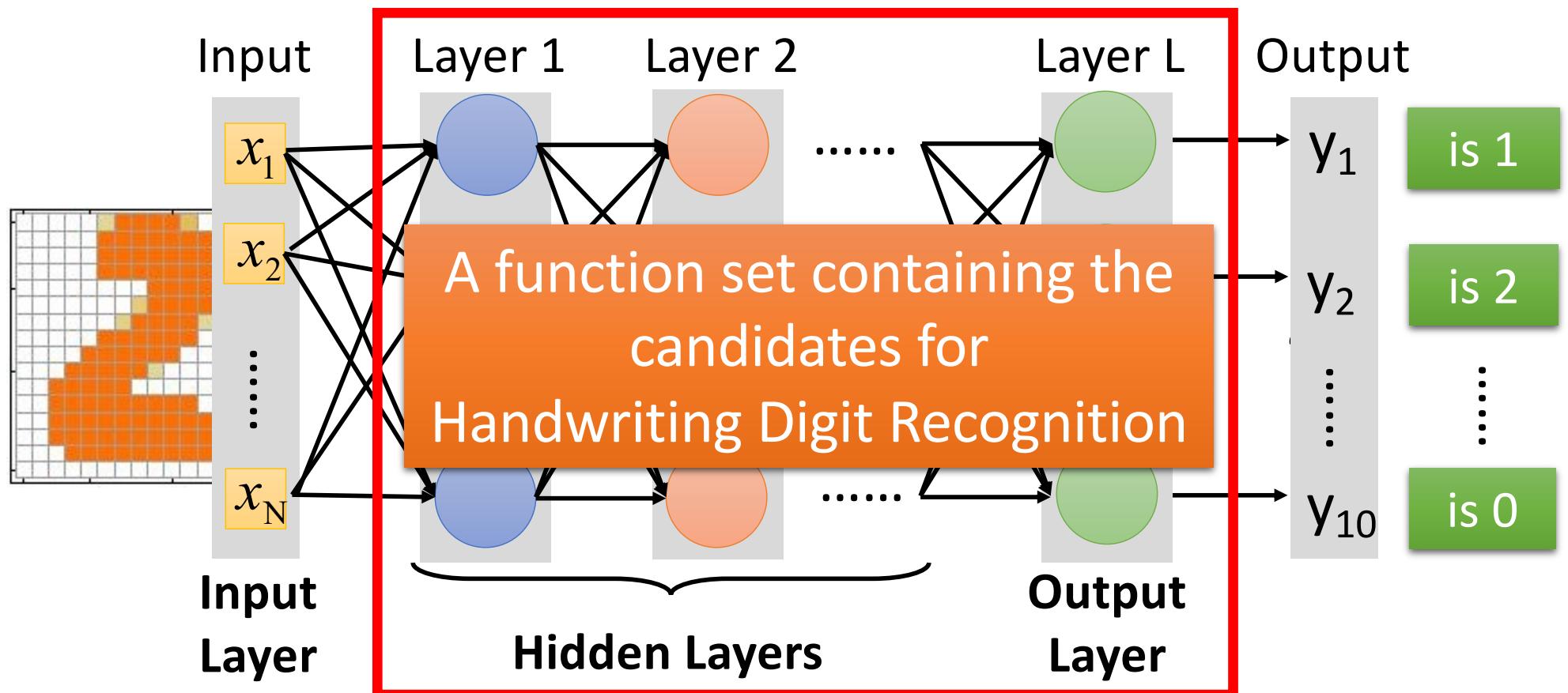
Each dimension represents
the confidence of a digit.

Example Application

- Handwriting Digit Recognition

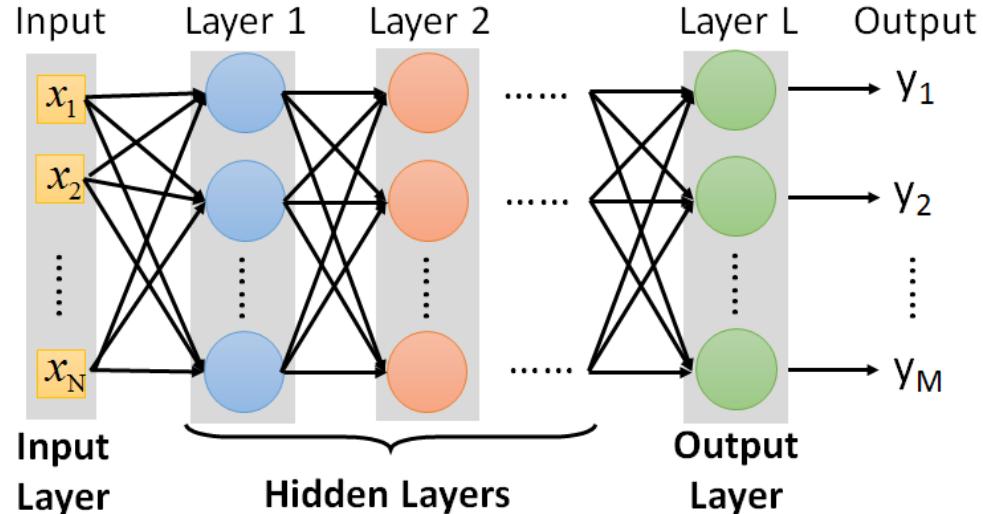


Example Application



You need to decide the network structure to let a good function in your function set.

FAQ



- Q: How many layers? How many neurons for each layer?

Trial and Error

+

Intuition

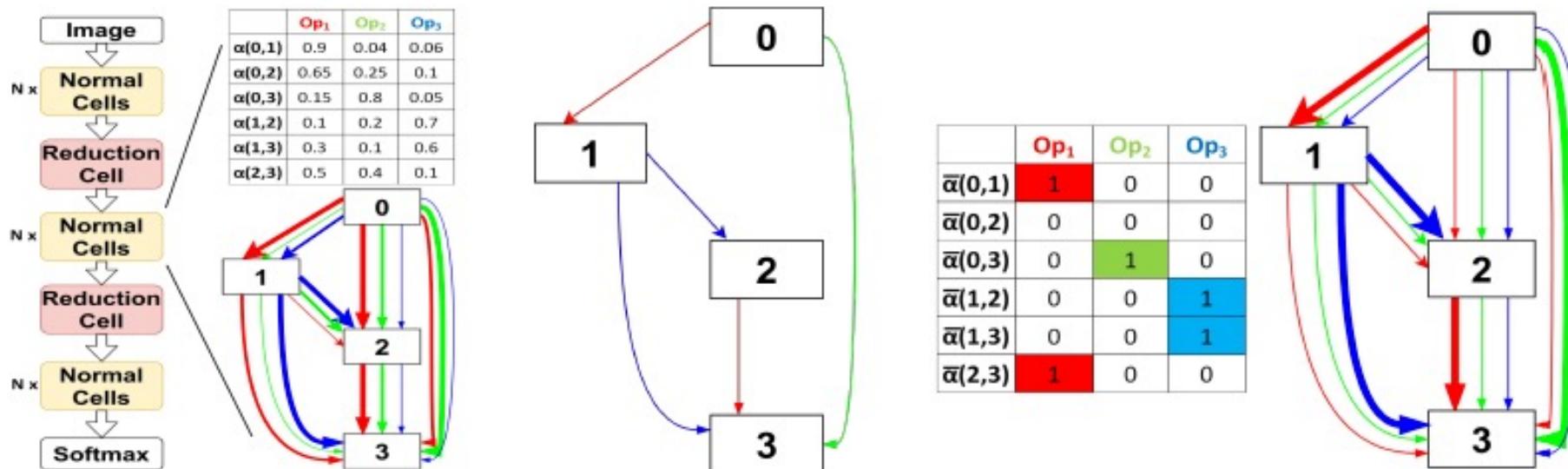
- Q: Can the structure be automatically determined?
 - E.g. Evolutionary Artificial Neural Networks
- Q: Can we design the network structure?

Convolutional Neural Network (CNN)

FAQ

- Q: Can the structure be automatically determined?
- E.g. Evolutionary Artificial Neural Networks

N. Sinha and K.W. Chen, “Evolving Neural Architecture Using One Shot Model,” GECCO 2021.



FAQ

- Q: Can the structure be automatically determined?
 - E.g. Evolutionary Artificial Neural Networks

N. Sinha and K.W. Chen, “Evolving Neural Architecture Using One Shot Model,” GECCO 2021.

Architecture	Test Error (%) C 10	Test Error (%) C100	Params (M)	Search Time (GPU Days)	Search Method
DenseNet-BC [14]	3.46	17.18	25.6	-	manual
PNAS [18]	3.41	-	3.2	225	SMBO
NASNet-A [37]	2.65	-	3.3	1800	RL
ENAS [26]	2.86	-	4.6	0.45	RL
DARTS [21]	2.76 ± 0.09	17.54	3.3	4	gradient-based
GDAS [8]	2.93	18.38	3.4	0.83	gradient-based
SNAS [33]	2.85 ± 0.02	-	2.8	1.5	gradient-based
SETN [7]	2.69	17.25	4.6	1.8	gradient-based
PDARTS [2]	2.50	16.55	3.4	0.3	gradient-based
AmoebaNet-A [27]	3.34 ± 0.06	18.93	3.2	3150	evolution
Large-scale Evolution [†] [28]	5.4	-	5.4	2750	evolution
Hierarchical Evolution [†] [19]	3.75	-	15.7	300	evolution
NSGANetV1-A2 [24]	2.65	17.42	0.9	27	evolution
NSGA-NET [23]	2.75	-	3.3	4	evolution
RSPS[17]	2.86 ± 0.08	-	4.3	2.7	random
EvNAS-A (Ours)	2.47 ± 0.06	16.37	3.6	3.83	evolution
EvNAS-B (Ours)	2.62 ± 0.06	16.51	3.8	3.83	evolution
EvNAS-C (Ours)	2.63 ± 0.05	16.86	3.4	3.83	evolution
EvNAS-Rand (Ours)	2.84 ± 0.08	-	2.7	0.62	random
EvNAS-ND (Ours)	2.78 ± 0.1	-	3.8	3.83	evolution
EvNAS-NDF (Ours)	2.75 ± 0.09	-	3.1	3.83	evolution
EvNAS-NDT (Ours)	2.67 ± 0.06	-	3.5	3.83	evolution
EvNAS-Mut (Ours)	2.79 ± 0.06	-	3.4	3.83	evolution
EvNAS-Cross (Ours)	2.81 ± 0.08	-	3.2	3.83	evolution

Training neural networks

- Collect a set of labeled training data $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$
- Training neural networks: Finding network parameters $\theta = \{\mathbf{w}, \mathbf{b}\}$ to minimize the loss between true training label \mathbf{y}_i and the estimated label, e.g.,

$$L(\theta) = \sum_{i=1}^N \|\mathbf{y}_i - g_{\mathbf{w}}(\mathbf{x}_i)\|^2$$

- Minimization can be done by **gradient descent** if $L(\cdot)$ is differentiable with respect to θ
- **Back-propagation**: a widely used method for optimizing multi-layer neural networks

Gradient Descent

Network parameters $\theta = \{w_1, w_2, \dots, b_1, b_2, \dots\}$

Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow \dots$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \vdots \\ \partial L(\theta)/\partial b_1 \\ \partial L(\theta)/\partial b_2 \\ \vdots \end{bmatrix}$$

Compute $\nabla L(\theta^0)$ $\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$

Compute $\nabla L(\theta^1)$ $\theta^2 = \theta^1 - \eta \nabla L(\theta^1)$

Millions of parameters

To compute the gradients efficiently,
we use **backpropagation**.

Chain Rule

Case 1

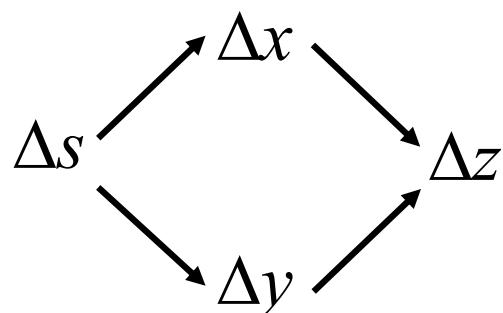
$$y = g(x) \quad z = h(y)$$

$$\Delta x \rightarrow \Delta y \rightarrow \Delta z$$

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$$

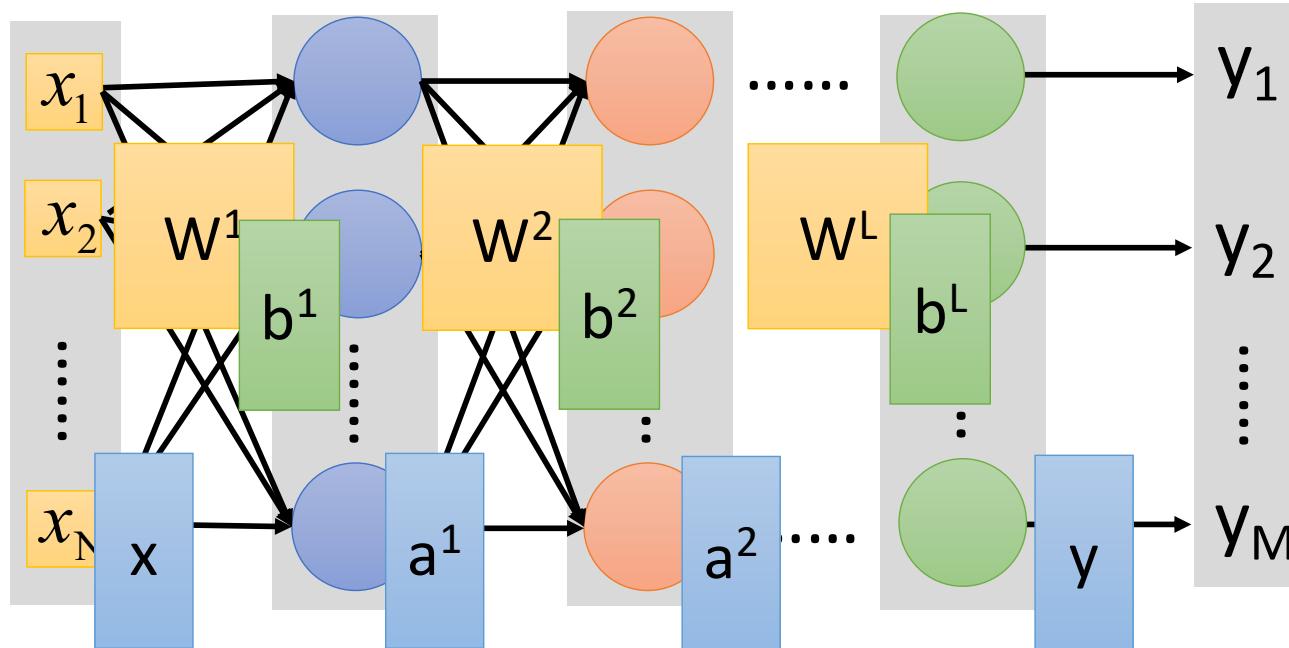
Case 2

$$x = g(s) \quad y = h(s) \quad z = k(x, y)$$



$$\frac{dz}{ds} = \frac{\partial z}{\partial x} \frac{dx}{ds} + \frac{\partial z}{\partial y} \frac{dy}{ds}$$

Neural Network



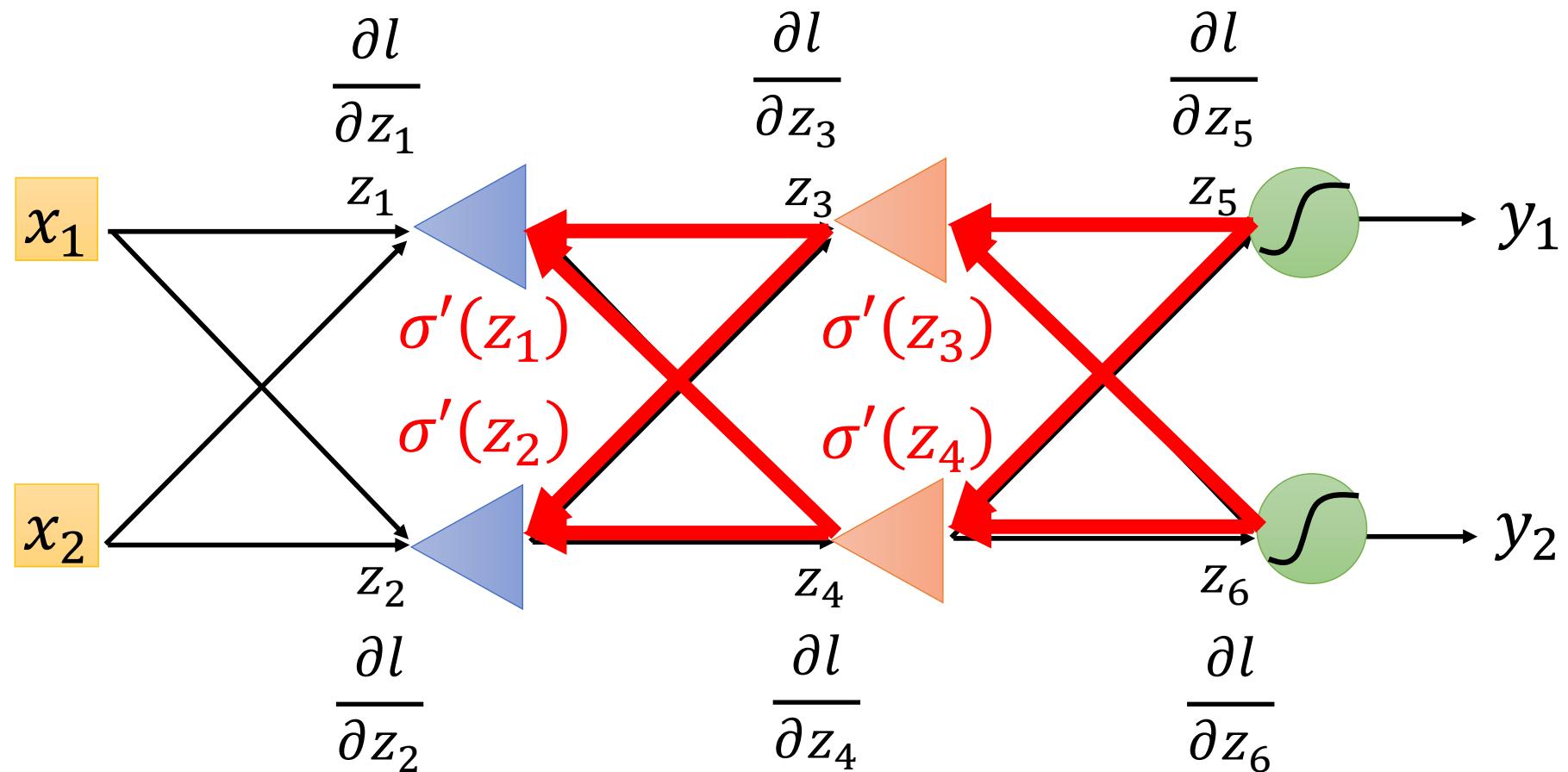
$$y = f(x)$$

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Backpropagation – Backward Pass

Compute $\partial l / \partial z$ for all activation function inputs z

Compute $\partial l / \partial z$ from the output layer



Backpropagation

- Backpropagation: an efficient way to compute $\partial L / \partial w$ in neural network



Caffe



Ref:

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html



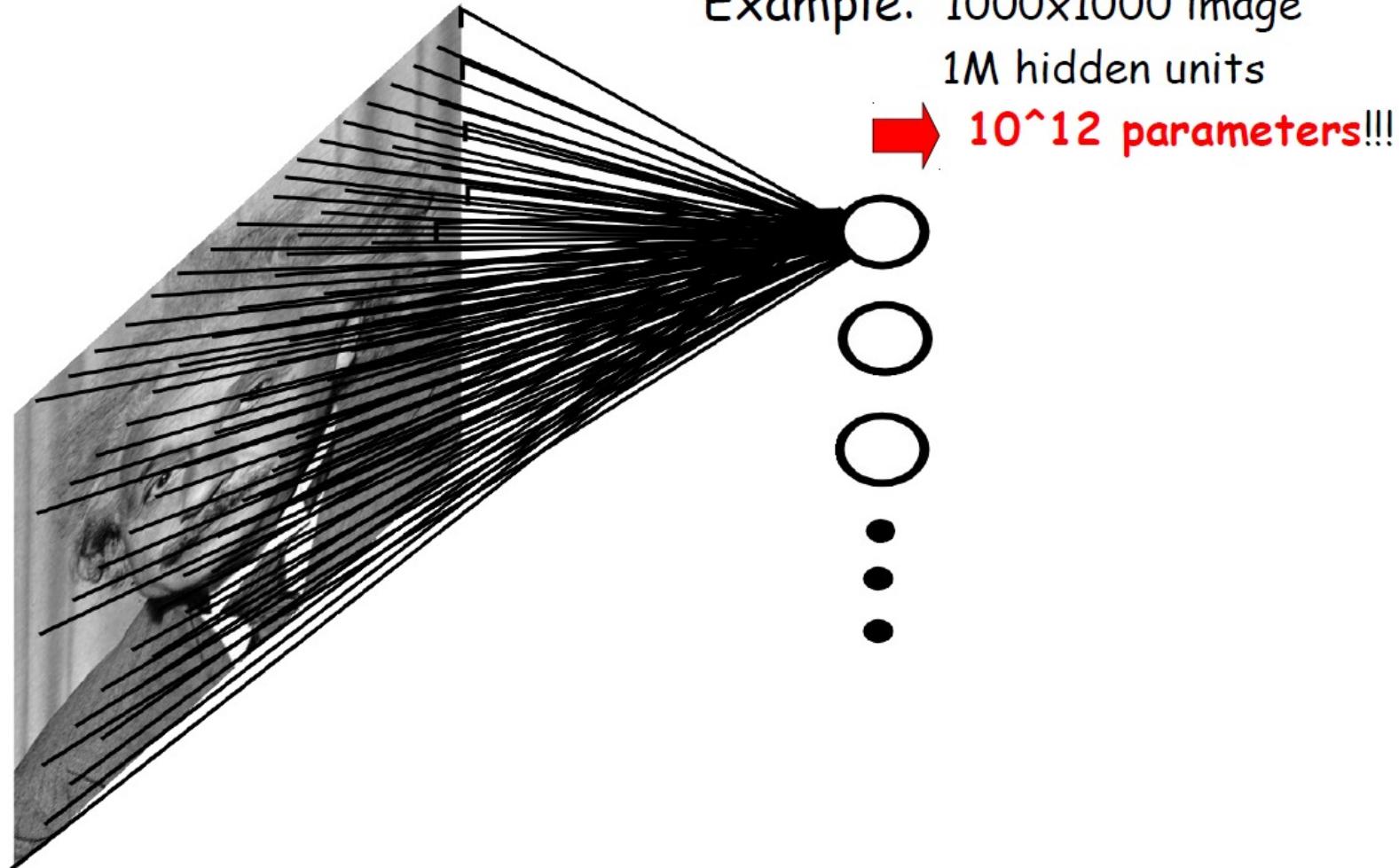
theano



libdnn
台大周伯威
同學開發

Convolutional Neural Networks (CNN)

of parameters in fully connected NN



slide: MA Ranzato

What makes deep learning successful in computer vision?

Li Fei-Fei



IM³GENET

Geoffrey Hinton



Data collection

One million images
with labels

Evaluation task

Predict 1,000 image
categories

Deep learning

CNN is not new
Design network structure
New training strategies

Convolutional neural networks (CNN)

- CNN: a multi-layer neural network with
 1. Local connectivity
 2. Weight sharing
- Why local connectivity?
 - Spatial correlation is local (**locality of spatial dependencies**)
 - Reduce # of parameters
- Why weight sharing?
- Statistics is at different locations (**stationarity of statistics**)
 - Reduce # of parameters

Why Local Connectivity

- Some patterns are much smaller than the whole image

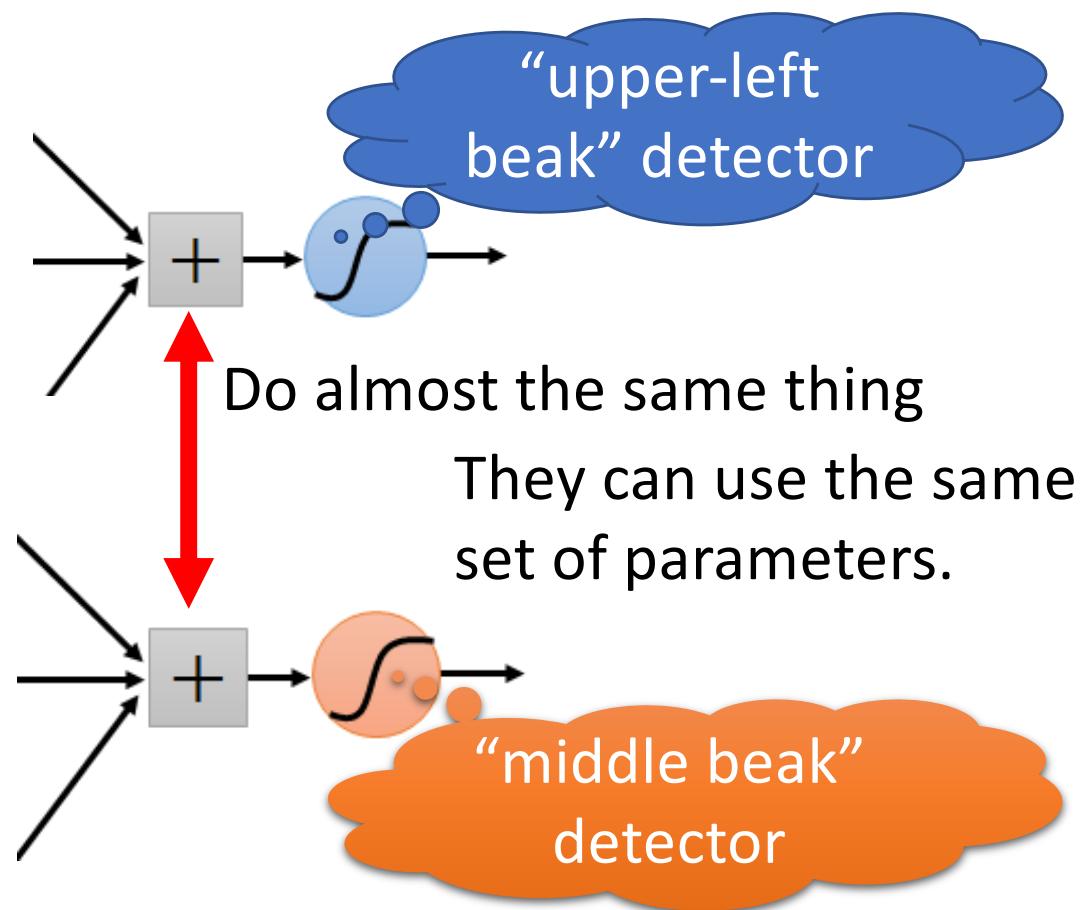
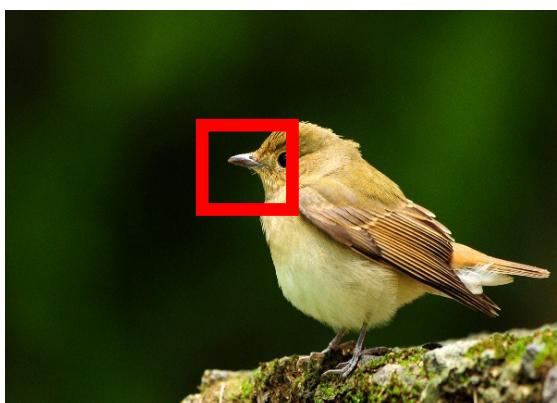
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

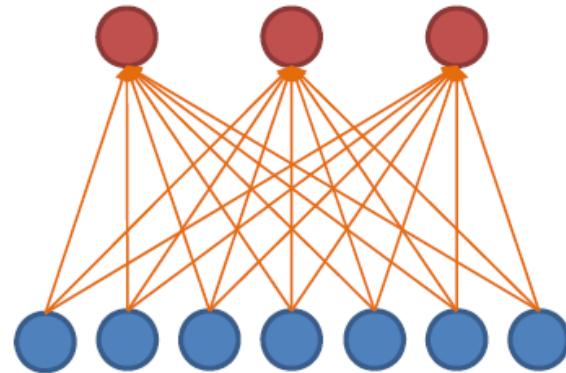


Why Weight Sharing

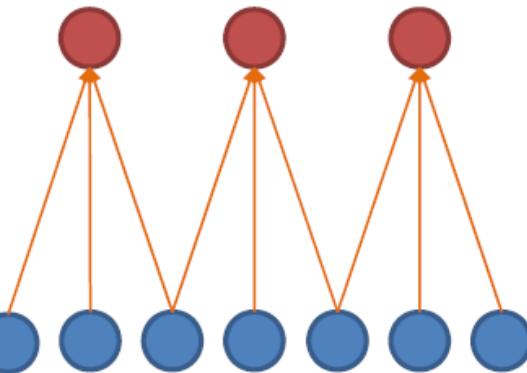
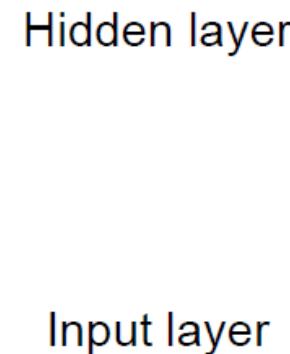
- The same patterns appear in different regions.



CNN: Local connectivity



Global connectivity

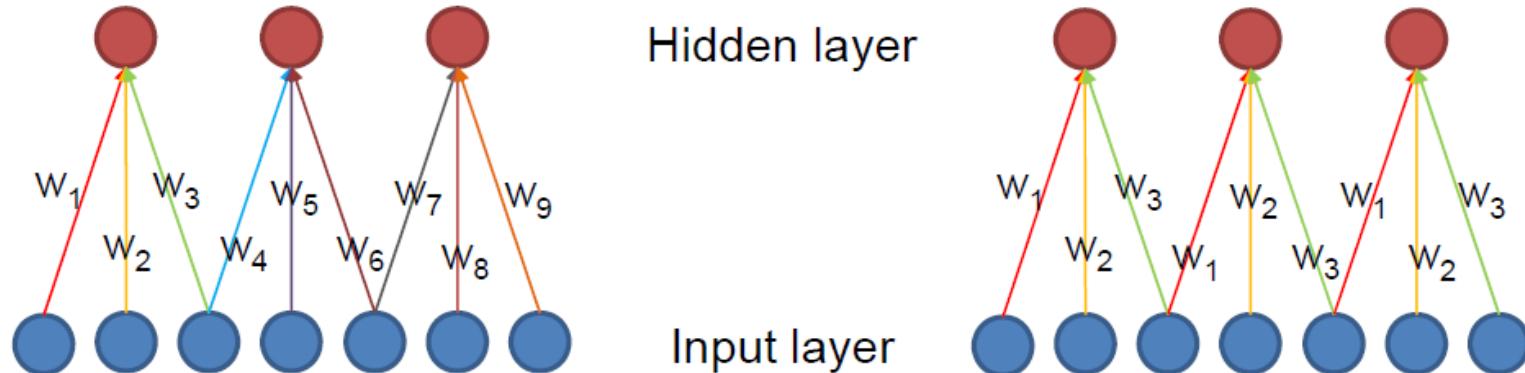


Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
 - Global connectivity: $3 \times 7 = 21$
 - Local connectivity: $3 \times 3 = 9$

slide: J.-B. Huang

CNN: Weight sharing



Without weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
 - Without weight sharing: $3 \times 3 = 9$
 - With weight sharing : $3 \times 1 = 3$

With weight sharing

slide: J.-B. Huang

Why CNN for Image

- Subsampling the pixels will not change the object

bird



subsampling

bird

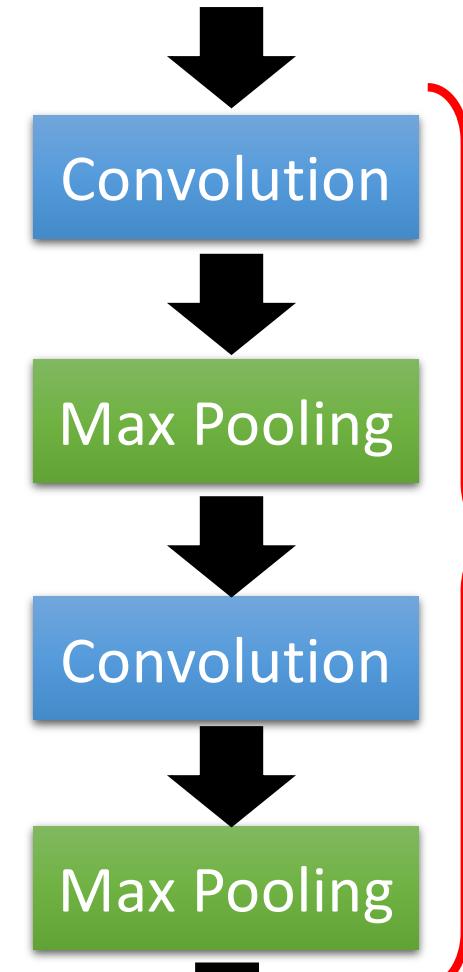
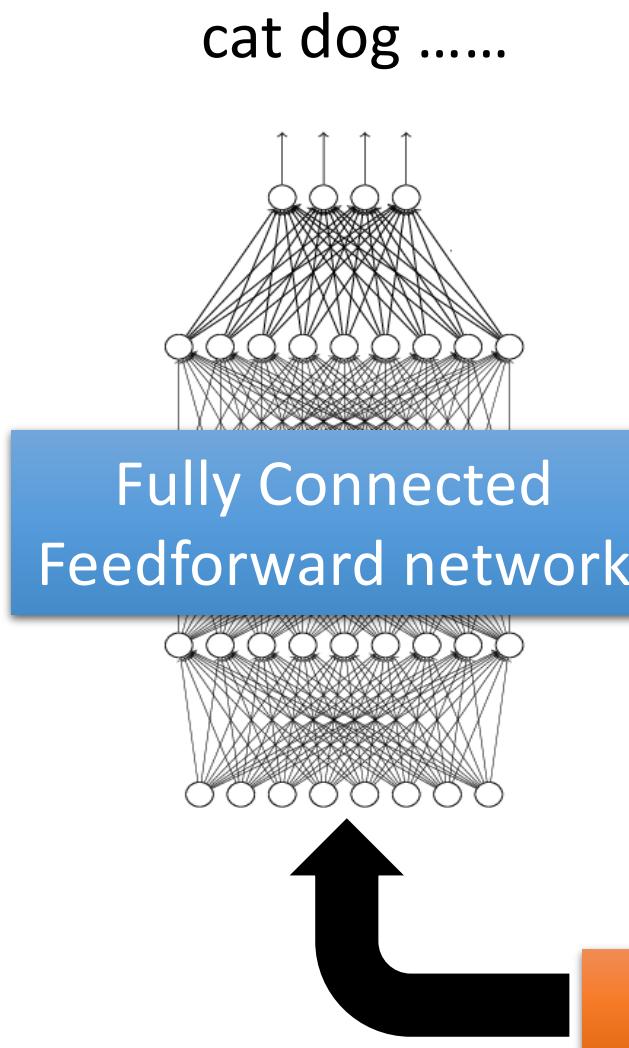


We can subsample the pixels to make image smaller



Less parameters for the network to process the image

The whole CNN

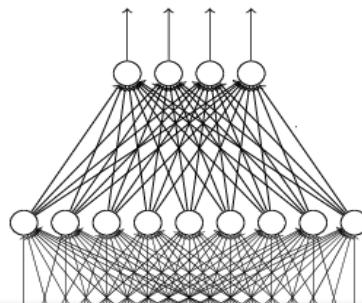


Can repeat
many times

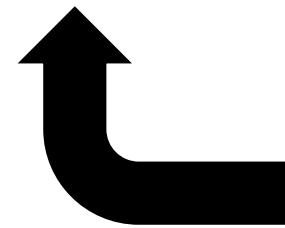
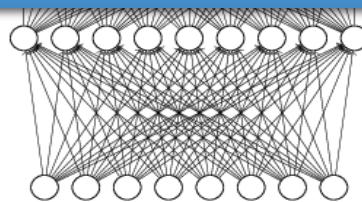
Flatten

The whole CNN

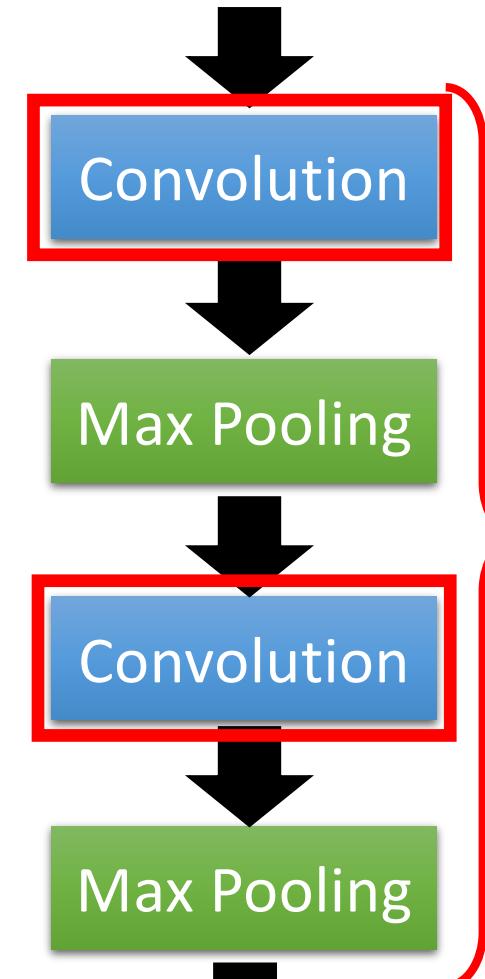
cat dog



Fully Connected
Feedforward network



Flatten



Can repeat
many times

CNN – Convolution

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1
Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2
Matrix

⋮

Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

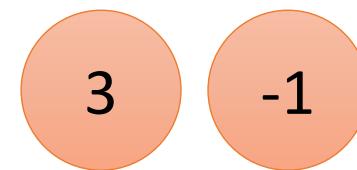
stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



CNN – Convolution

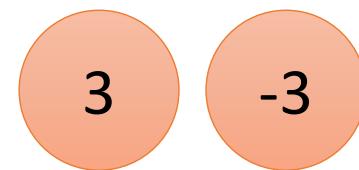
If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

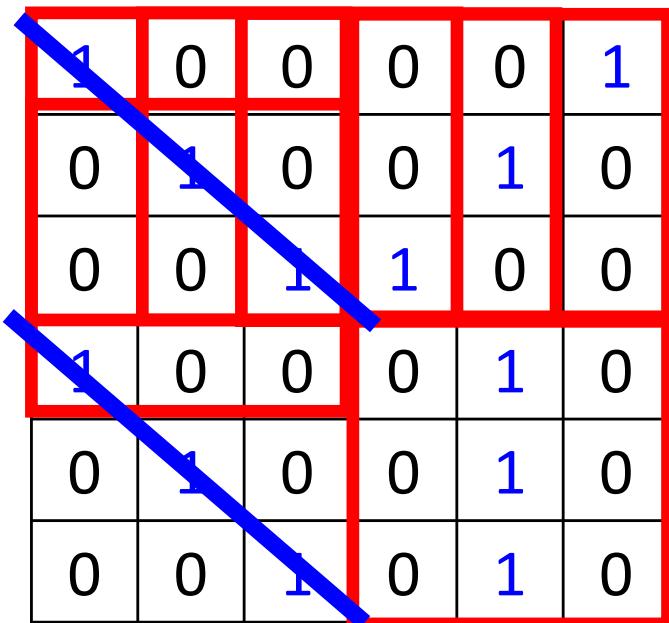
Filter 1



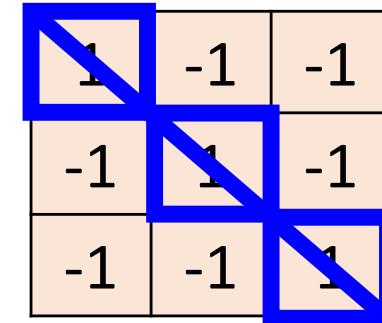
We set stride=1 below

CNN – Convolution

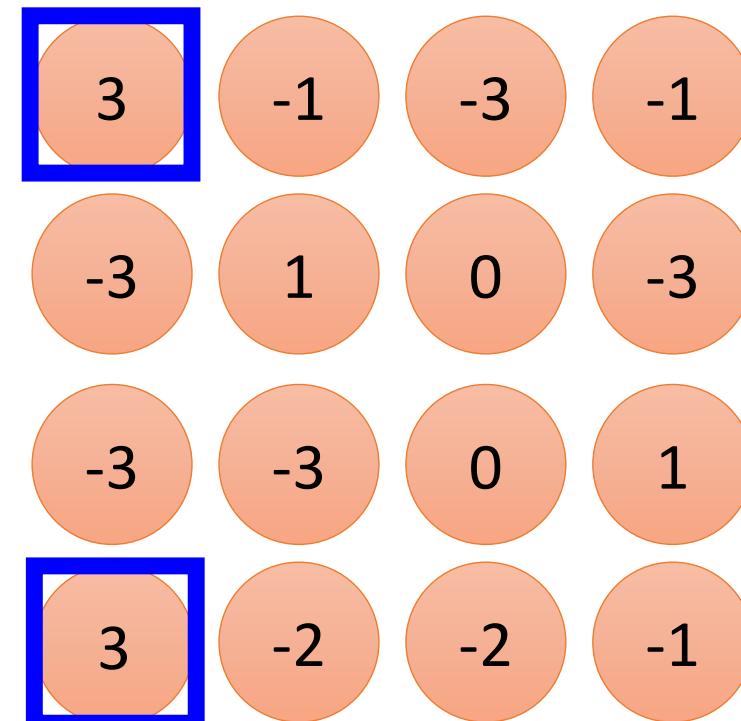
stride=1



6 x 6 image



Filter 1



Property 2

CNN – Convolution

stride=1

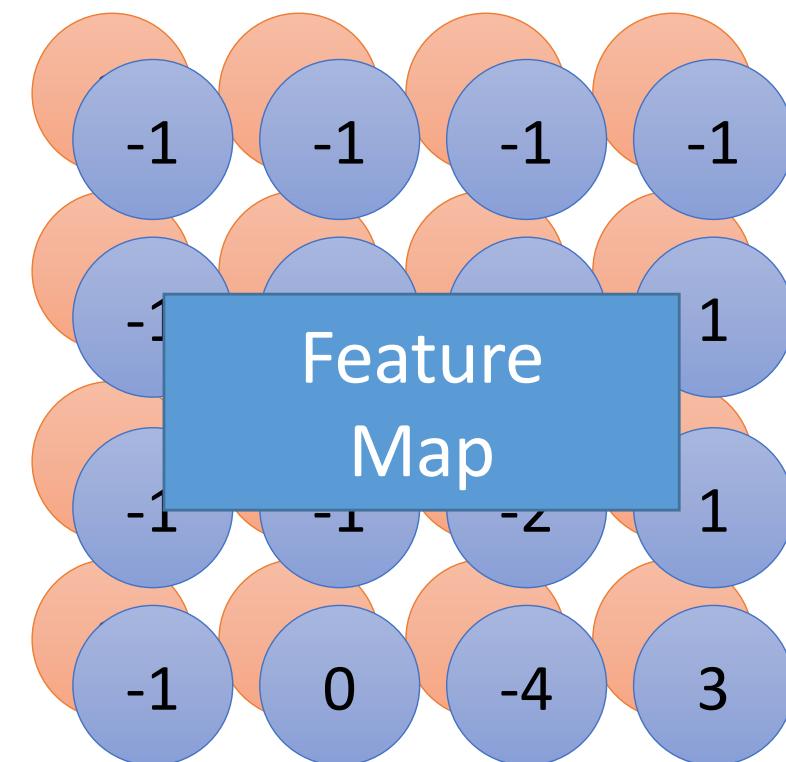
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

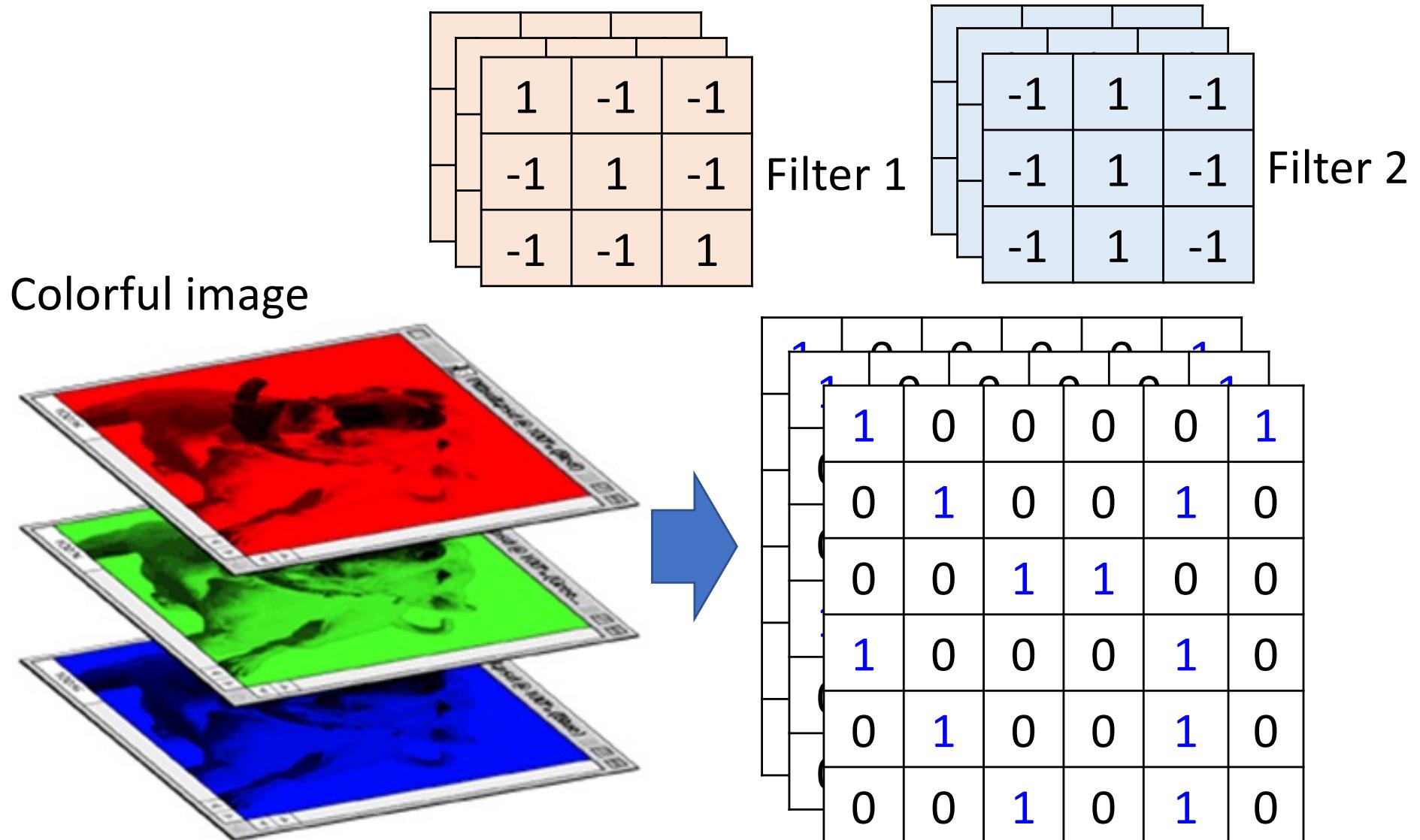
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Do the same process for
every filter

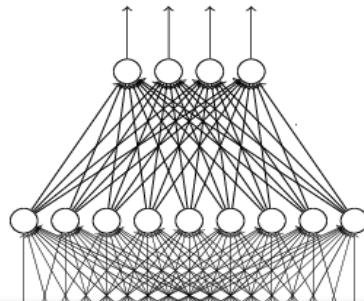


CNN – Colorful image

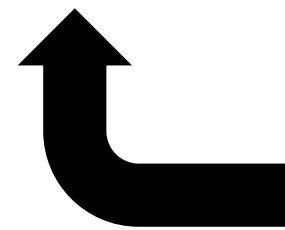
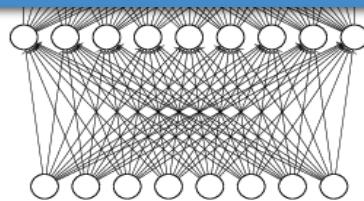


The whole CNN

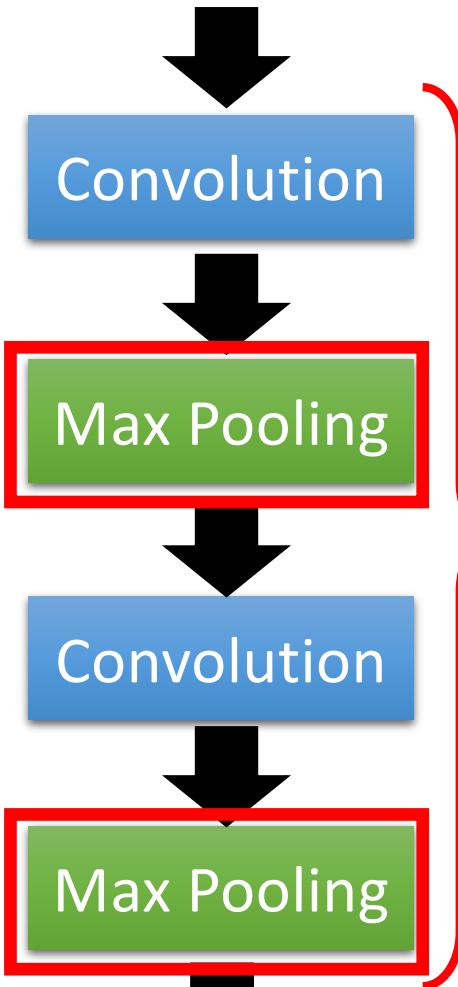
cat dog



Fully Connected
Feedforward network



Flatten



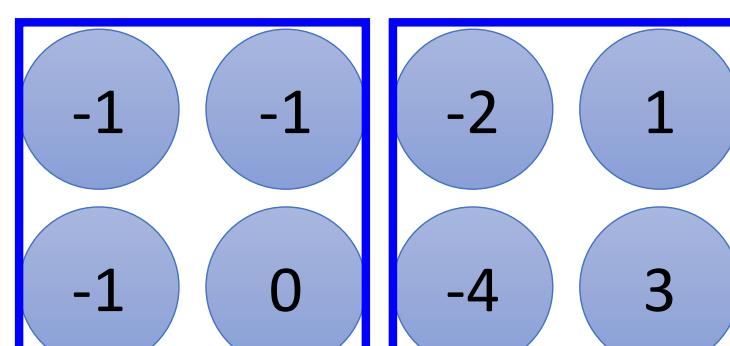
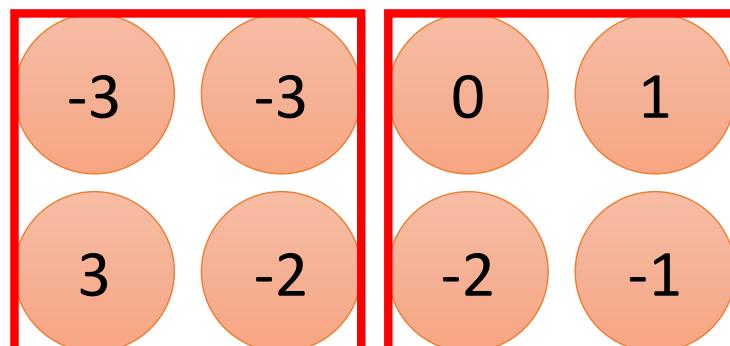
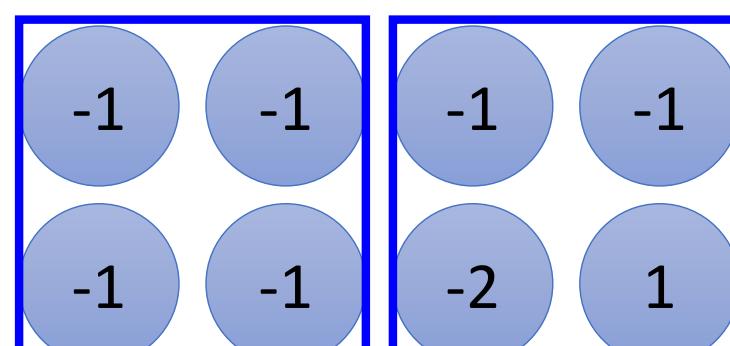
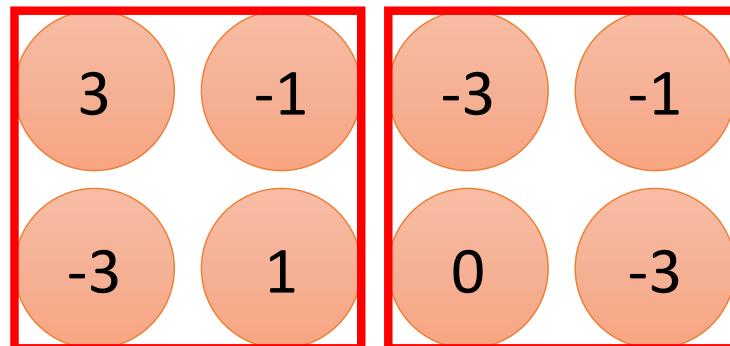
CNN – Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

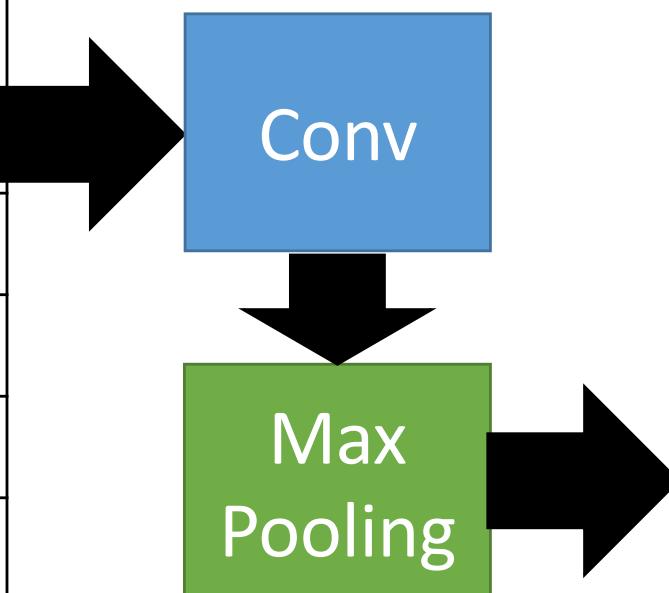
Filter 2



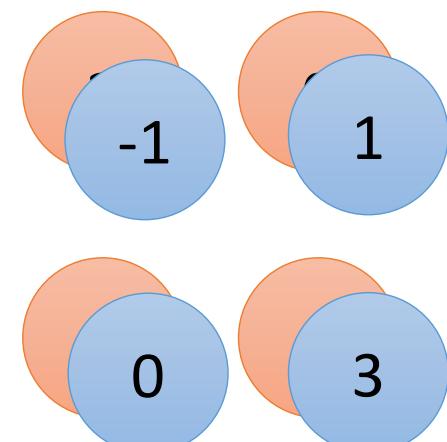
CNN – Max Pooling

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



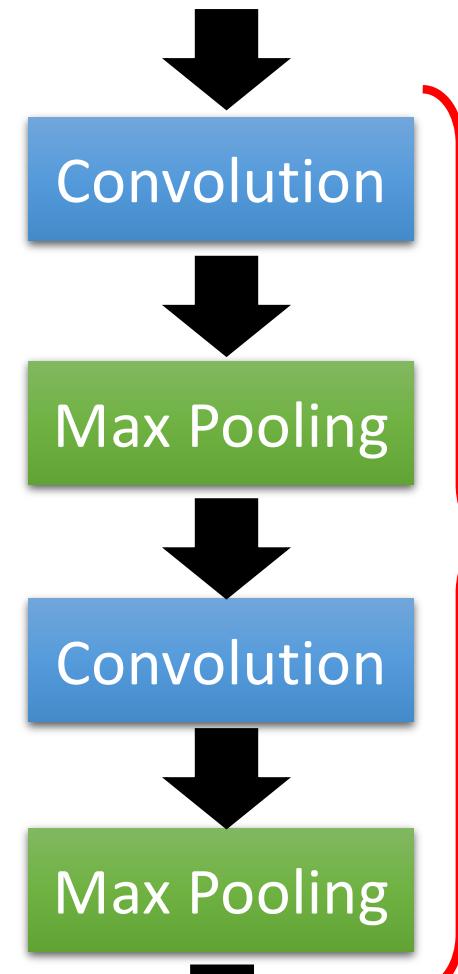
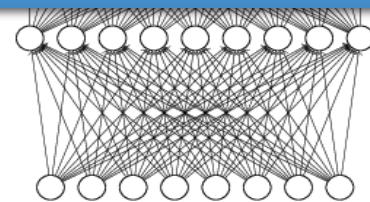
New image
but smaller



2 x 2 image

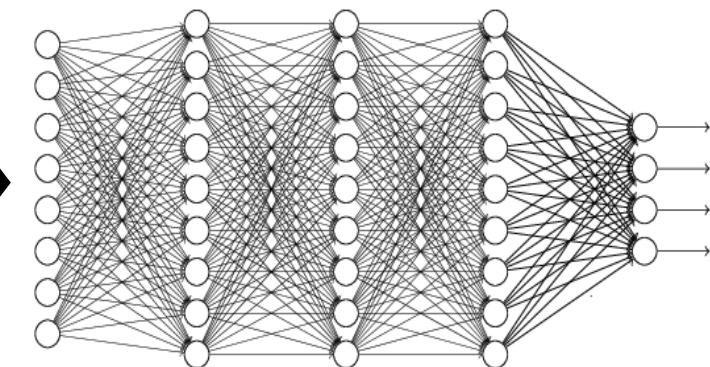
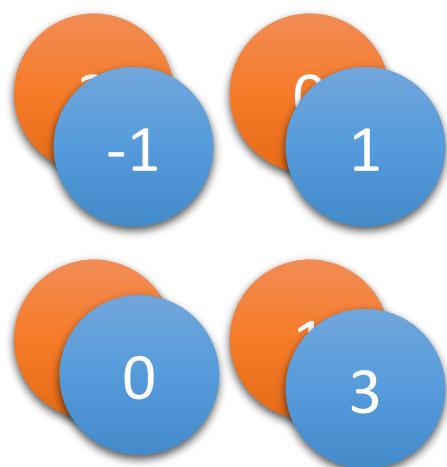
Each filter
is a channel

The whole CNN



Flatten

Flatten



Fully Connected
Feedforward network

What makes deep learning successful in computer vision?

Li Fei-Fei



IM³GENET

Geoffrey Hinton



Data collection

One million images
with labels

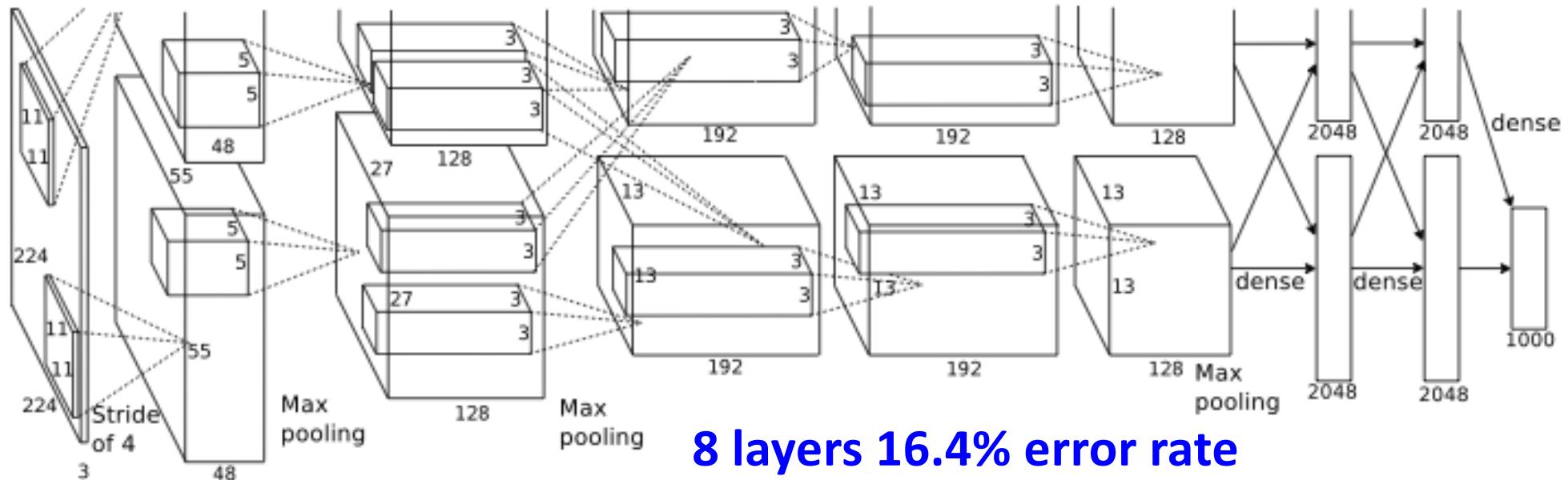
Evaluation task

Predict 1,000 image
categories

Deep learning

CNN is not new
Design network structure
New training strategies

Modern CNN: AlexNet



Input: $224 \times 224 \times 3 = 150K$

Neurons: $290400 + 186624 + 64896 + 64896 + 43264 + 4096 + 4096 + 1000 = 650K$

Weights: $11 \times 11 \times 3 \times 48 \times 2(35K) + 5 \times 5 \times 48 \times 128 \times 2(307K) + 128 \times 3 \times 3 \times 192 \times 4(884K) + 192 \times 3 \times 3 \times 192 \times 2(663K) + 192 \times 3 \times 3 \times 128 \times 2(442K) + 6 \times 6 \times 128 \times 2048 \times 4(38M) + 4096 \times 4096(17M) + 4096 \times 1000(4M) = 60M$

- **More data (1.2M)**
- **Trained on two GPUs for a week**
- **Dropout**

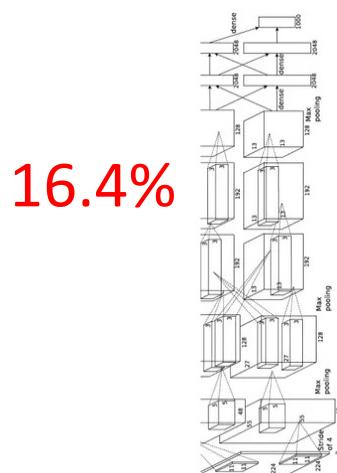
slide: M. Sun

ImageNet ILSVRC 2012-2014

Best non-convnet in 2012: 26.2%

Team	Year	Place	Error (top-5)	External data
SuperVision – Toronto (7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
<u>Human expert*</u>			5.1%	

Deep = Many hidden layers

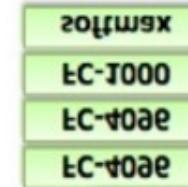


16.4%

AlexNet (2012)

7.3%

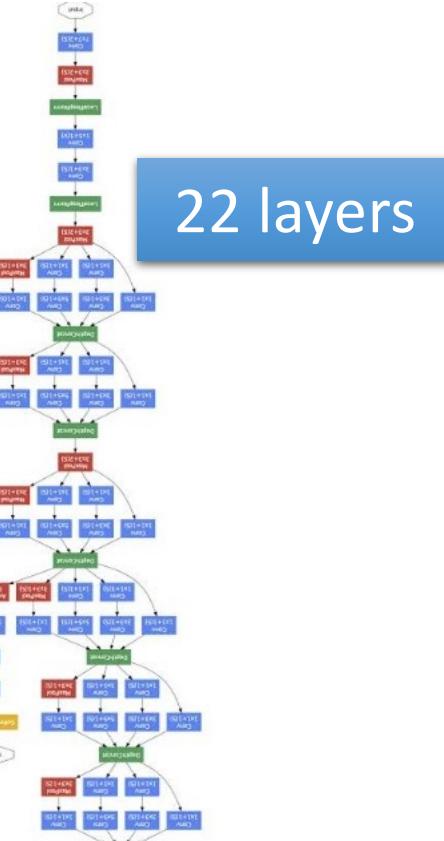
VGG (2014)



19 layers

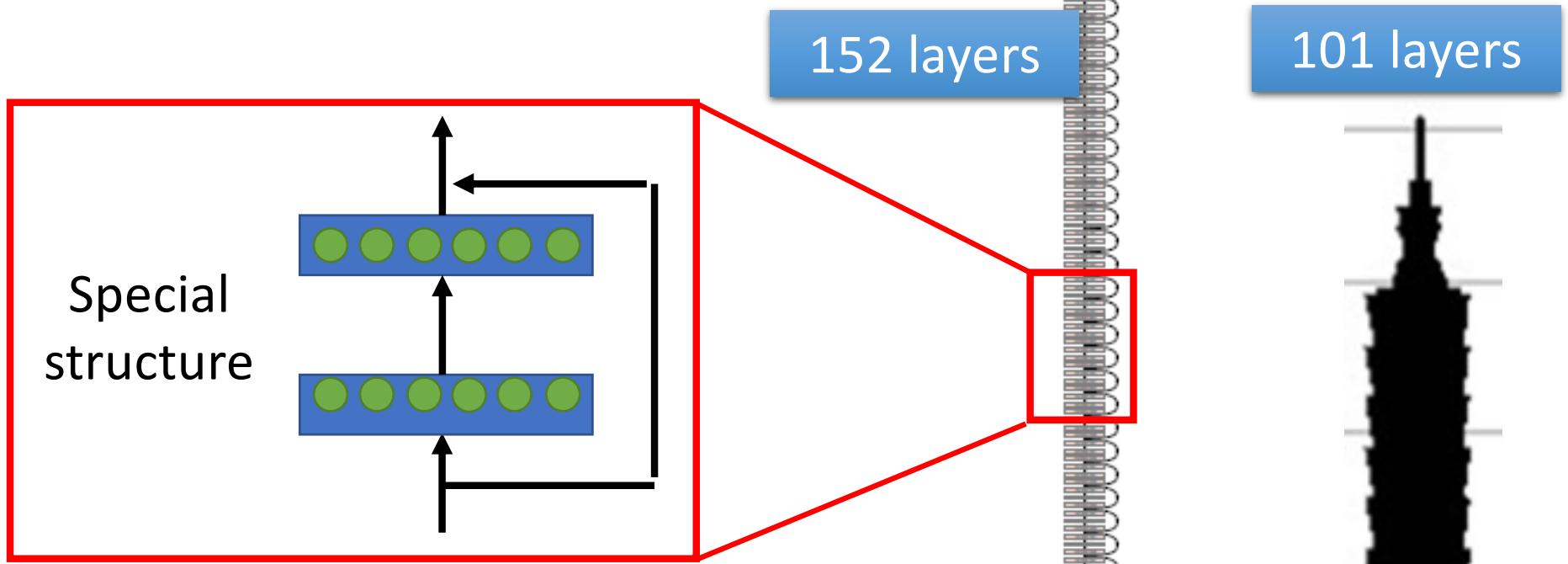
6.7%

GoogleNet (2014)



22 layers

Deep = Many hidden layers



Ref:

<https://www.youtube.com/watch?v=dxB6299gpvl>

3.57%

16.4%

7.3%

6.7%

AlexNet
(2012)

VGG
(2014)

GoogleNet
(2014)

Residual Net
(2015)

Taipei
101

ImageNet is an image database organized according to the [WordNet](#) hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.



What do these images have in common? *Find out!*

[Research updates on improving ImageNet data](#)



Fei-Fei Li, Ph.D.
(publishes under L. Fei-Fei)

Professor
Director, Stanford AI Lab
Computer Science Department

Office	Room 246 Gates Bldg
Phone	(650) 725-3860
Email	feifeili [at] cs [dot] stanford [dot] edu
Twitter	@drfeifei
Address	353 Serra Mall, Gates Building, Stanfor

How we teach computers to understand pictures – Fei Fei Li



Appendix: Vision Transformer

Slide from:
1. H. Y. Lee
2. J. Johnson

Sequence Labeling

Is it possible to consider the context?

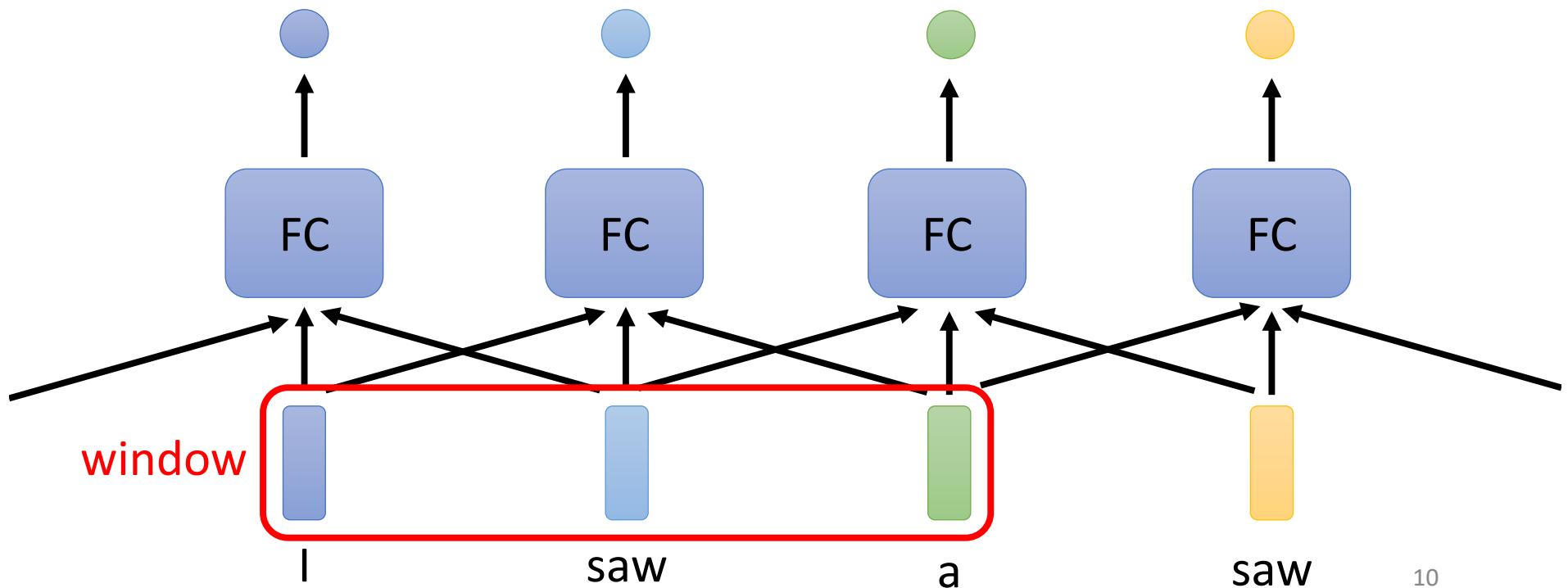


Fully-connected

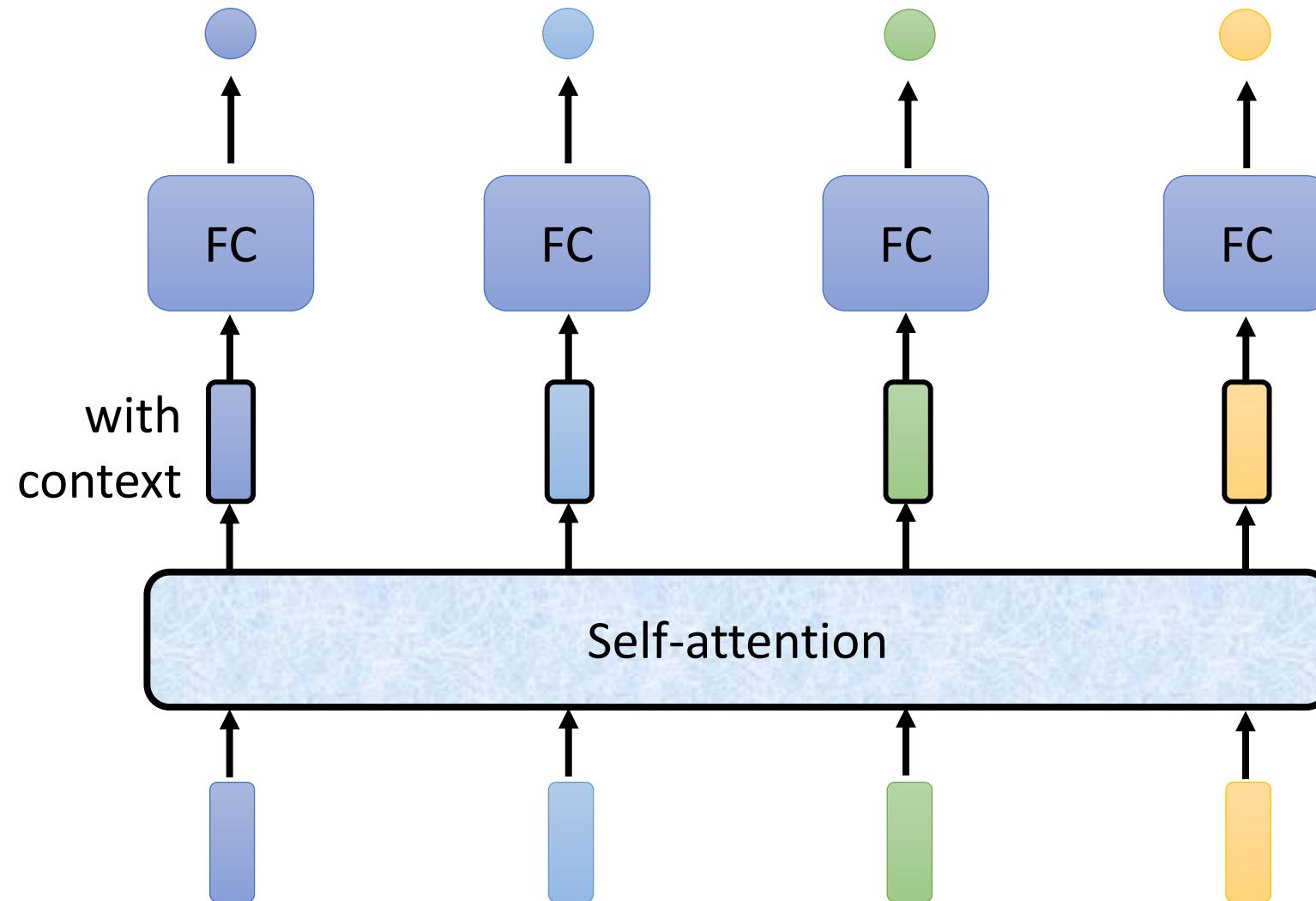
FC can consider the neighbor

How to consider the whole sequence?

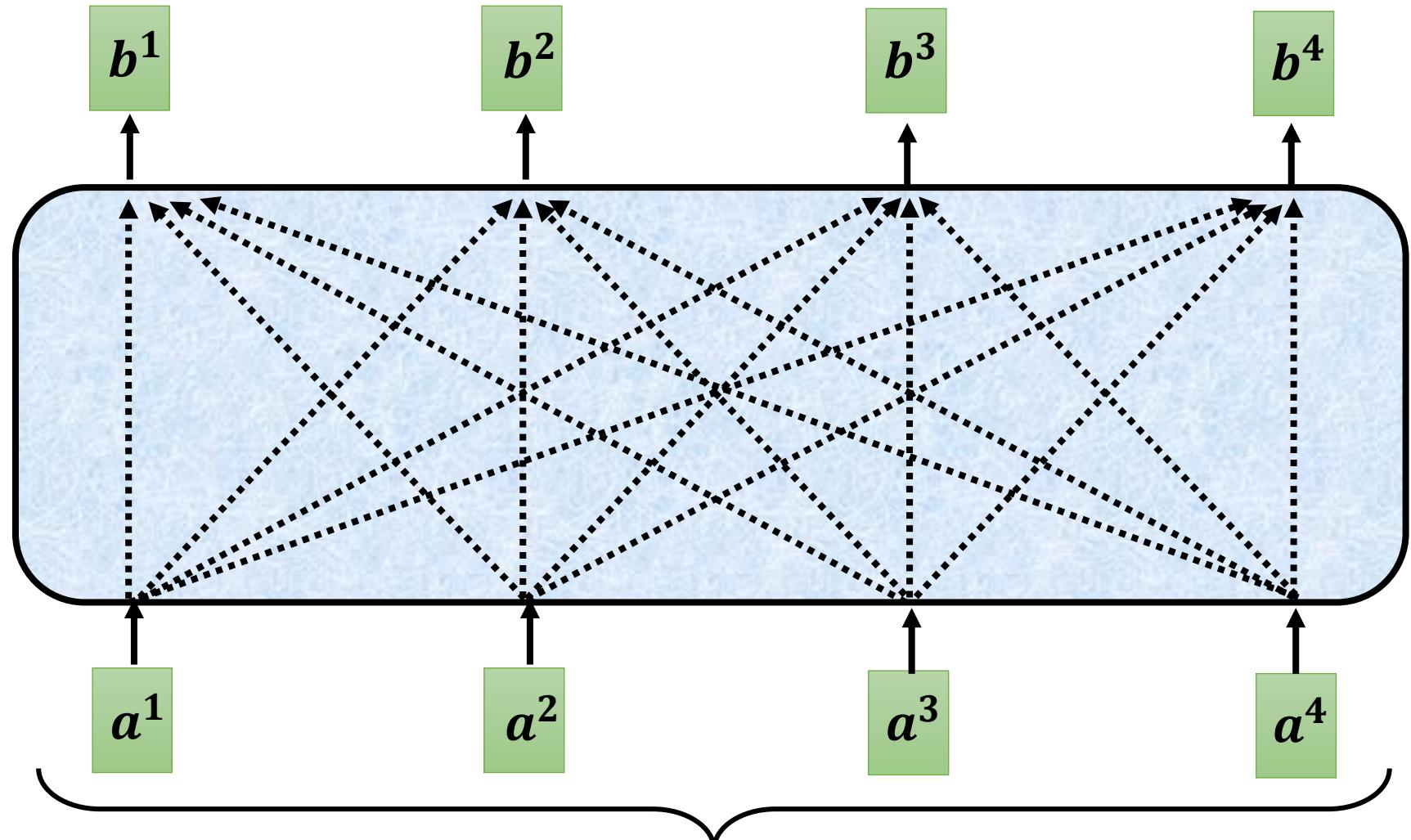
a window covers the whole sequence?



Self-attention



Self-attention

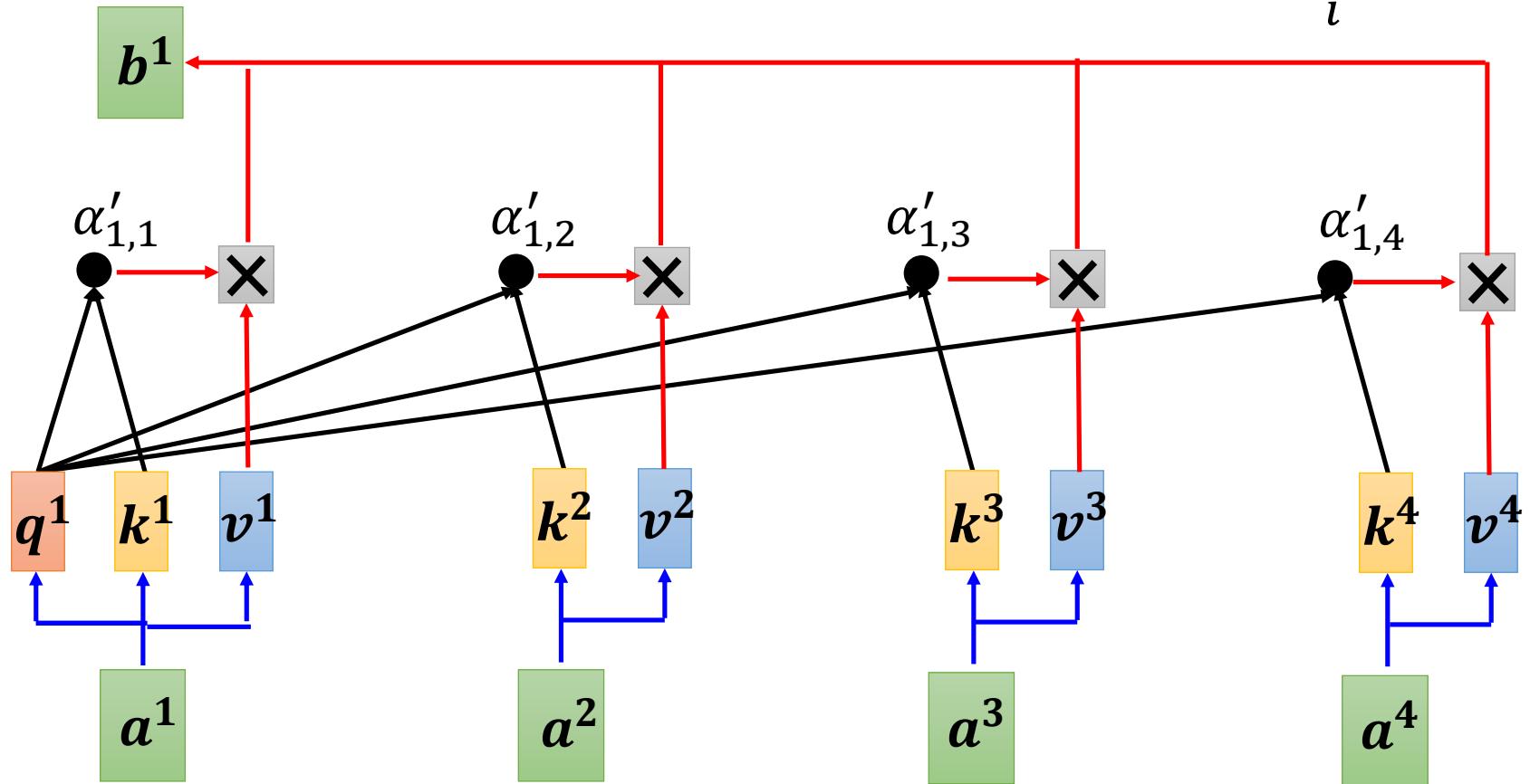


Can be either **input** or a **hidden layer**

Self-attention

Extract information based
on attention scores

$$b^1 = \sum_i \alpha'_{1,i} v^i$$



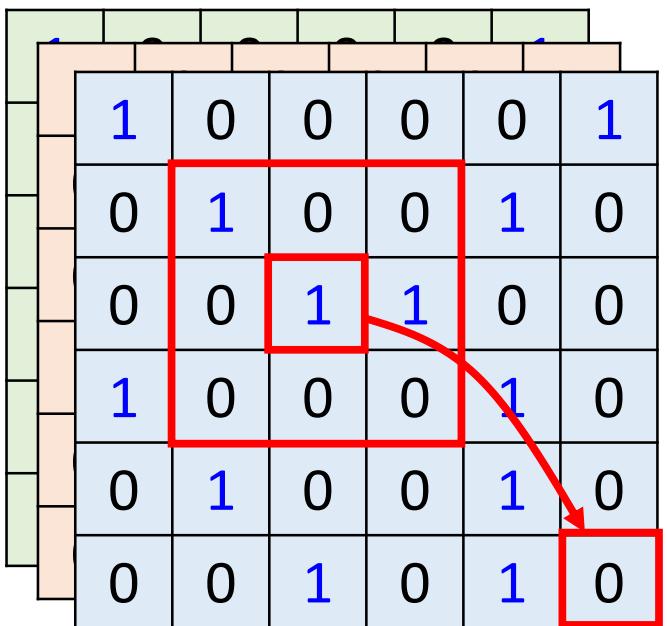
$$v^1 = W^v a^1$$

$$v^2 = W^v a^2$$

$$v^3 = W^v a^3$$

$$v^4 = W^v a^4$$

Self-attention v.s. CNN



CNN: self-attention that can only attends in a receptive field

- CNN is simplified self-attention.

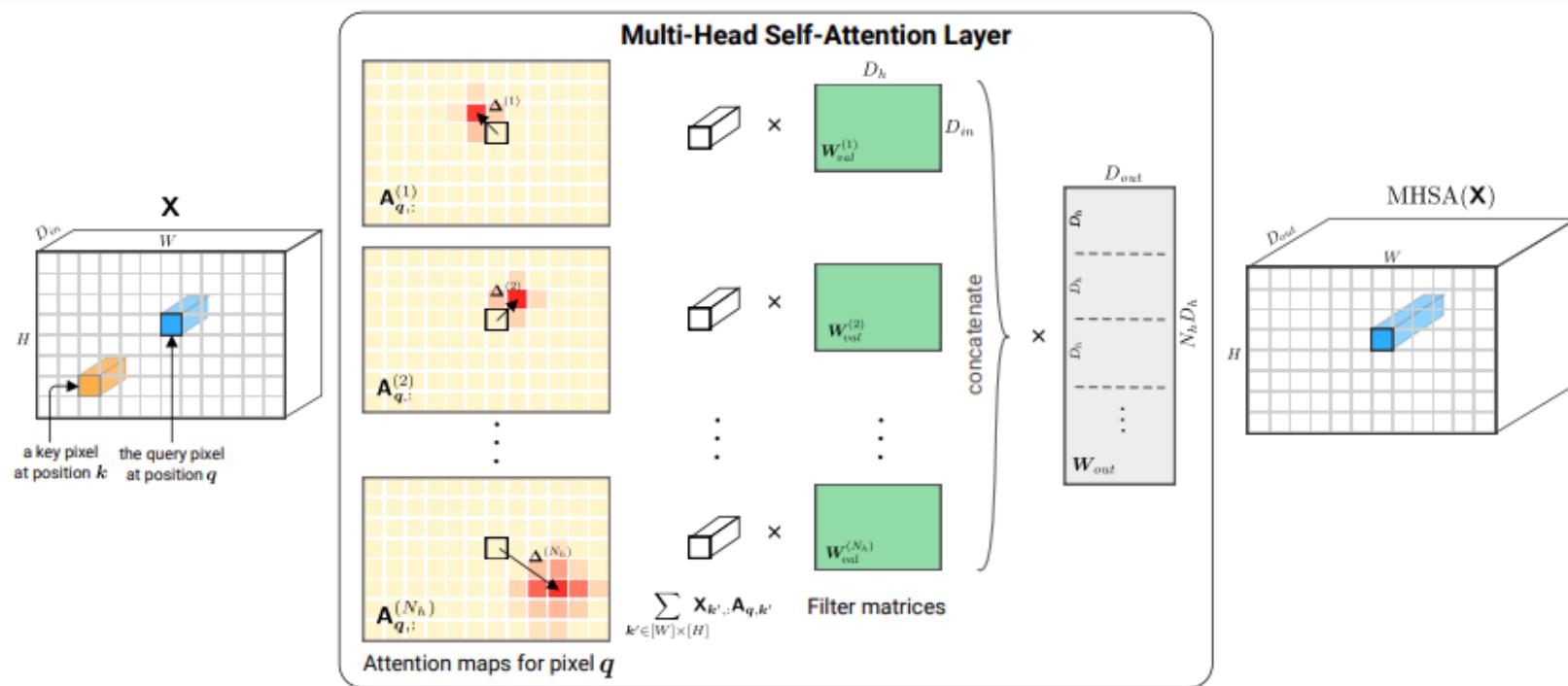
Self-attention: CNN with learnable receptive field

- Self-attention is the complex version of CNN.

Self-attention v.s. CNN

Self-attention

CNN



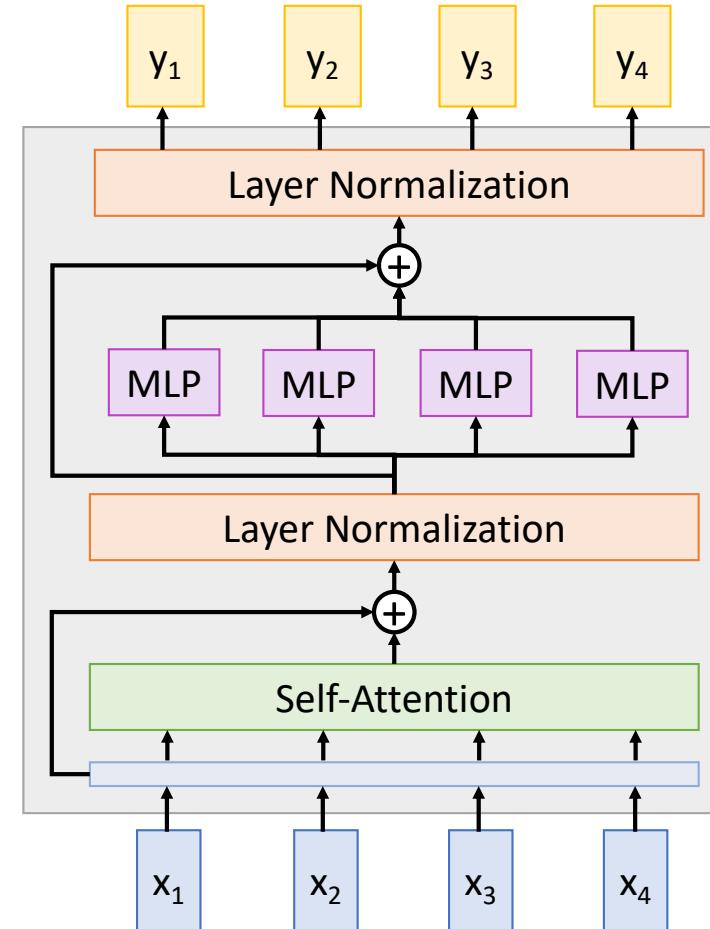
On the Relationship between Self-Attention and Convolutional Layers

<https://arxiv.org/abs/1911.03584>

Last Time: Transformer

Transformer block inputs a set of vectors, outputs a set of vectors.

Vectors only communicate via (multiheaded) self-attention



Vaswani et al, "Attention is all you need", NeurIPS 2017

Last Time: Transformer

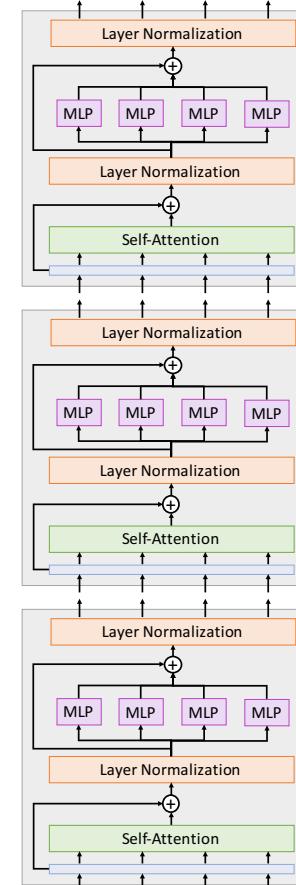
Transformer Block:

Input: Set of vectors x

Output: Set of vectors y

Hyperparameters:

- Number of blocks
- Number of heads per block
- Width (channels per head, FFN width)

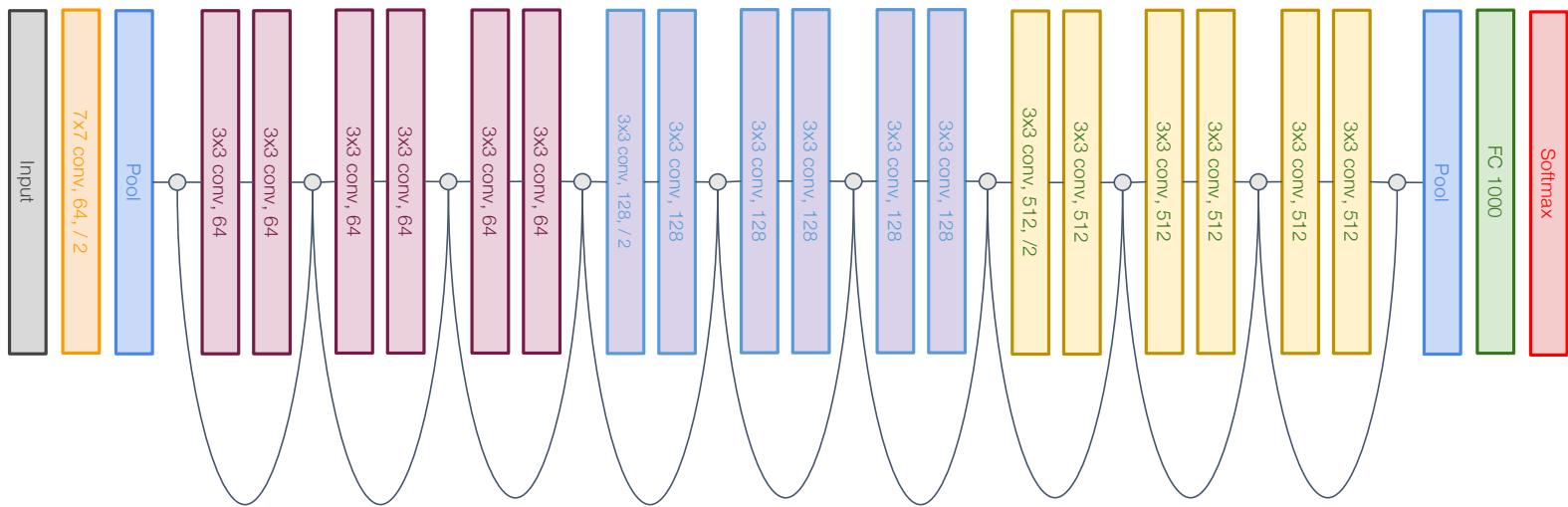


Vaswani et al, "Attention is all you need", NeurIPS 2017

Today: How to use Attention / Transformers for Vision?

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)



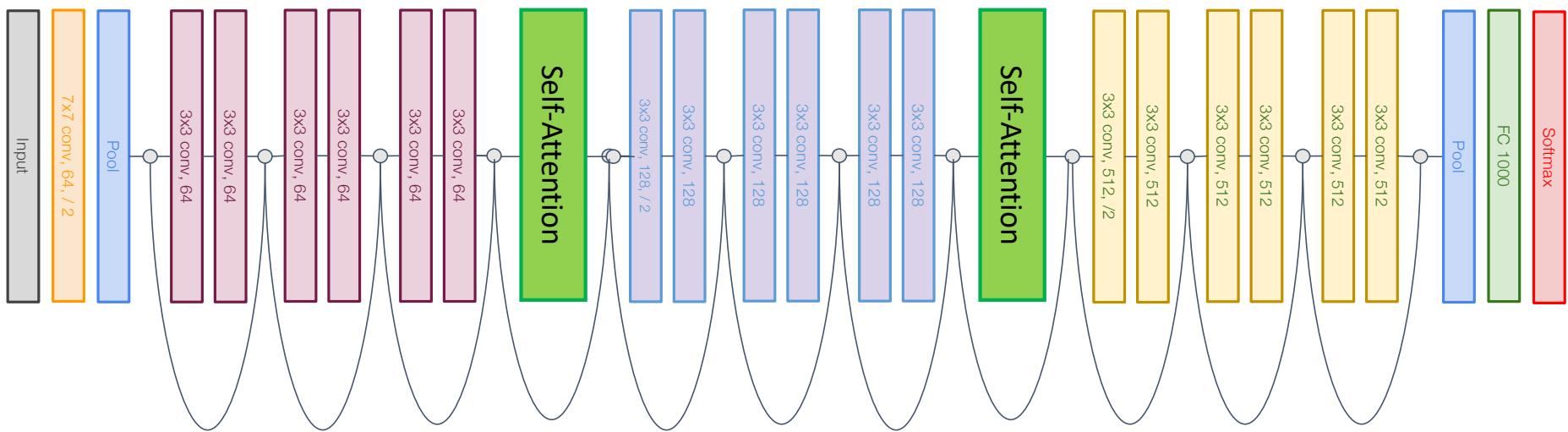
Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Start from standard CNN architecture (e.g. ResNet)

Add Self-Attention blocks between existing ResNet blocks



Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

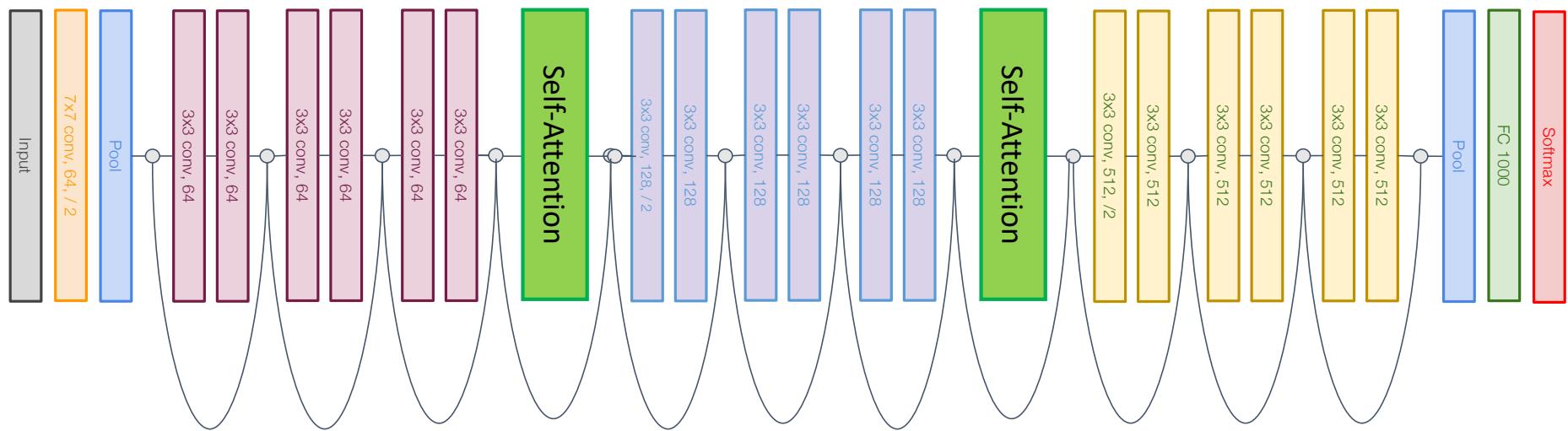
Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #1: Add attention to existing CNNs

Model is still a CNN! Start from standard CNN architecture (e.g. ResNet)

Can we replace

convolution entirely? Add Self-Attention blocks between existing ResNet blocks

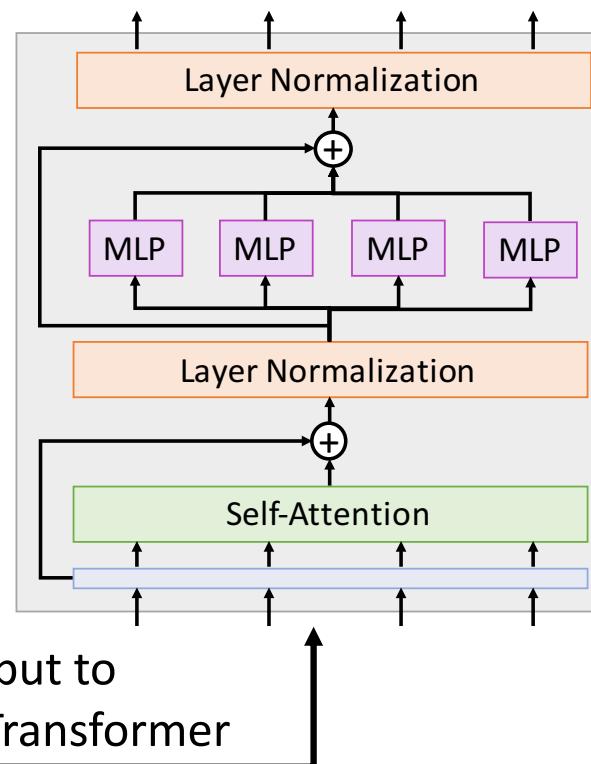
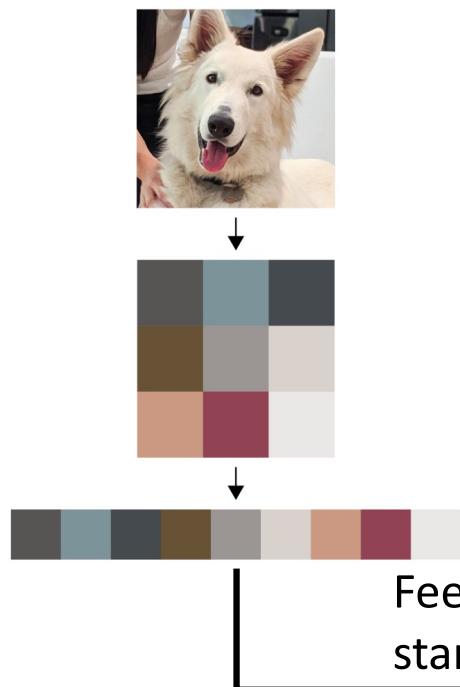


Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2018

Wang et al, "Non-local Neural Networks", CVPR 2018

Idea #3: Standard Transformer on Pixels

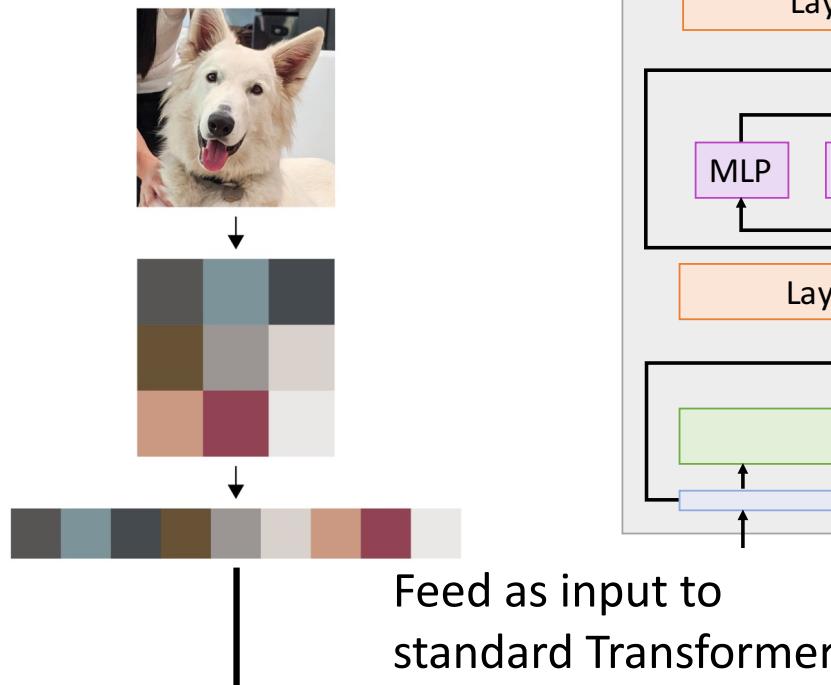
Treat an image as a set of pixel values



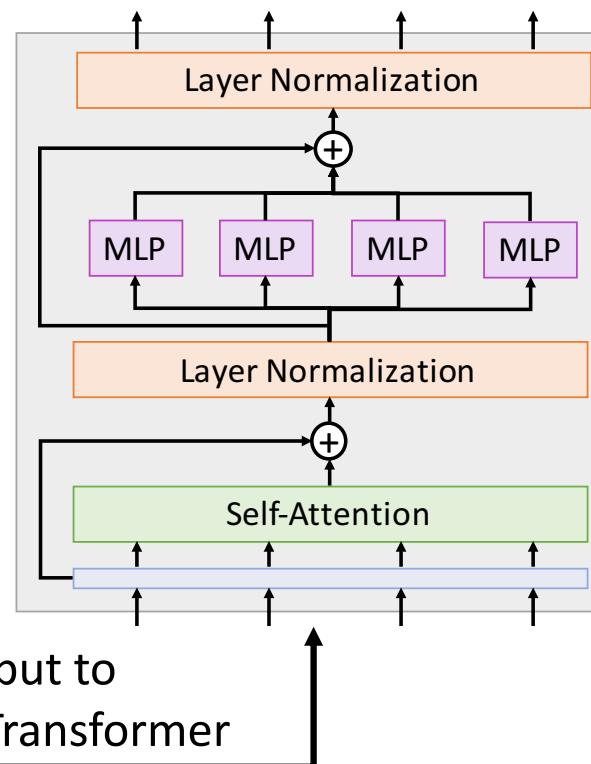
Chen et al, "Generative Pretraining from Pixels", ICML 2020

Idea #3: Standard Transformer on Pixels

Treat an image as a set of pixel values



Chen et al, "Generative Pretraining from Pixels", ICML 2020



Problem: Memory use!

R x R image needs R^4 elements per attention matrix

R=128, 48 layers, 16 heads per layer takes 768GB of memory for attention matrices for a single example...

Idea #4: Standard Transformer on Patches



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

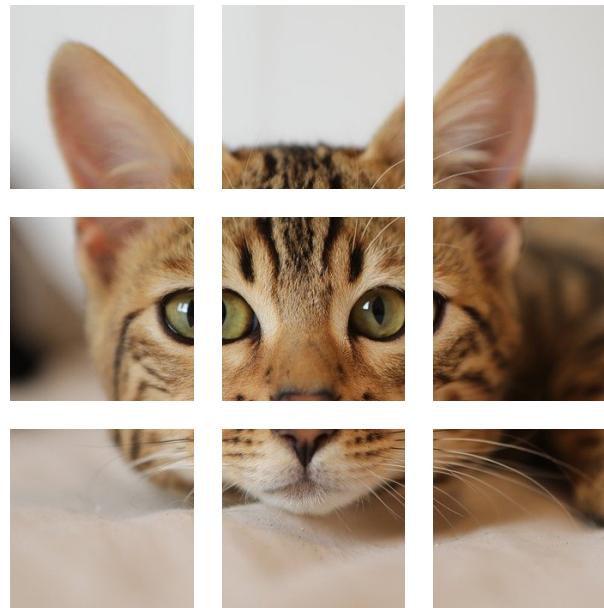
[Cat image](#) is free for commercial
use under a [Pixabay license](#)

Justin Johnson

Lecture 18 - 51

March 23, 2022

Idea #4: Standard Transformer on Patches



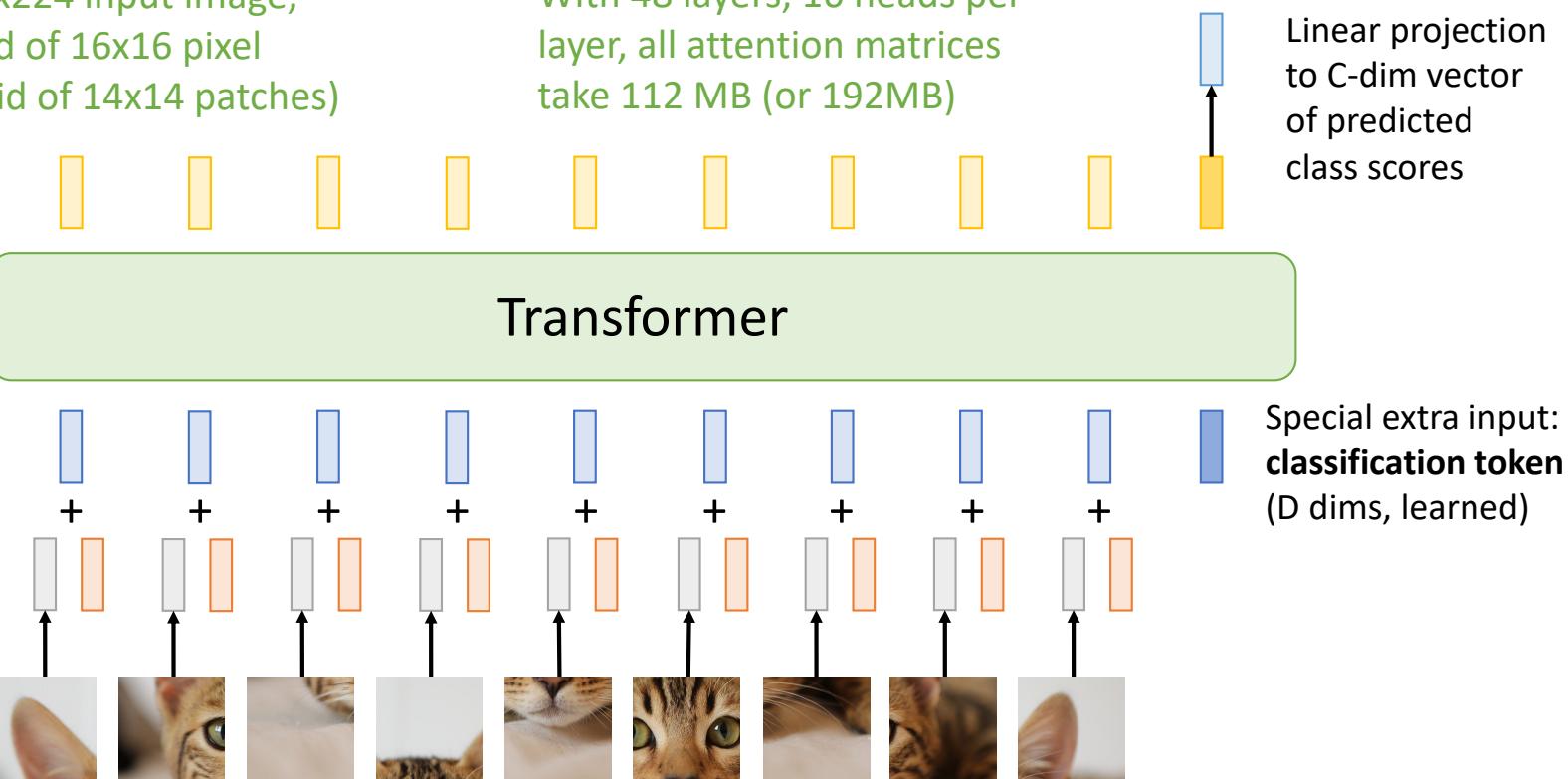
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

[Cat image](#) is free for commercial
use under a [Pixabay license](#)

Vision Transformer (ViT)

In practice: take 224x224 input image,
divide into 14x14 grid of 16x16 pixel
patches (or 16x16 grid of 14x14 patches)

Output vectors

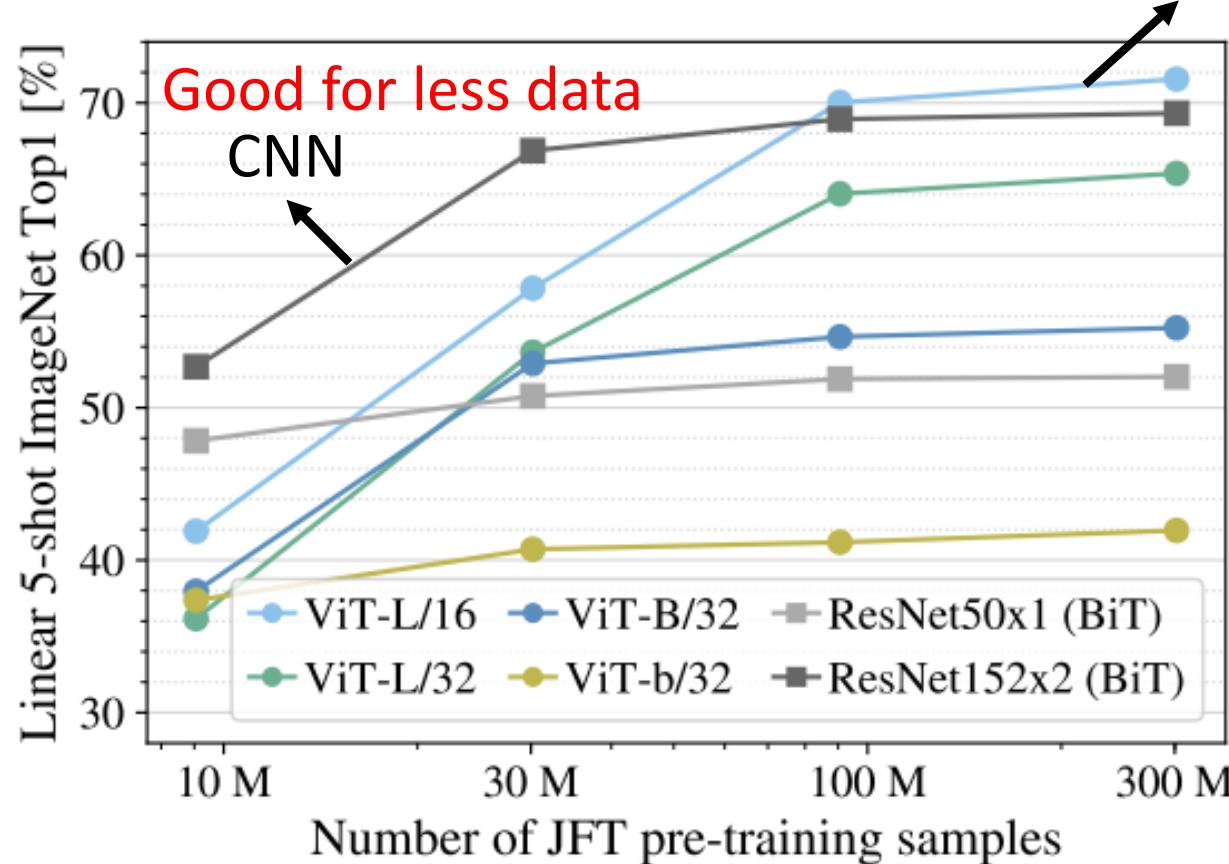


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Cat image is free for commercial use under a [Pixabay license](#)

Self-attention v.s. CNN

Good for more data
Self-attention



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

<https://arxiv.org/pdf/2010.11929.pdf>