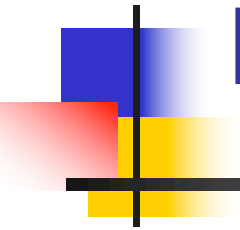


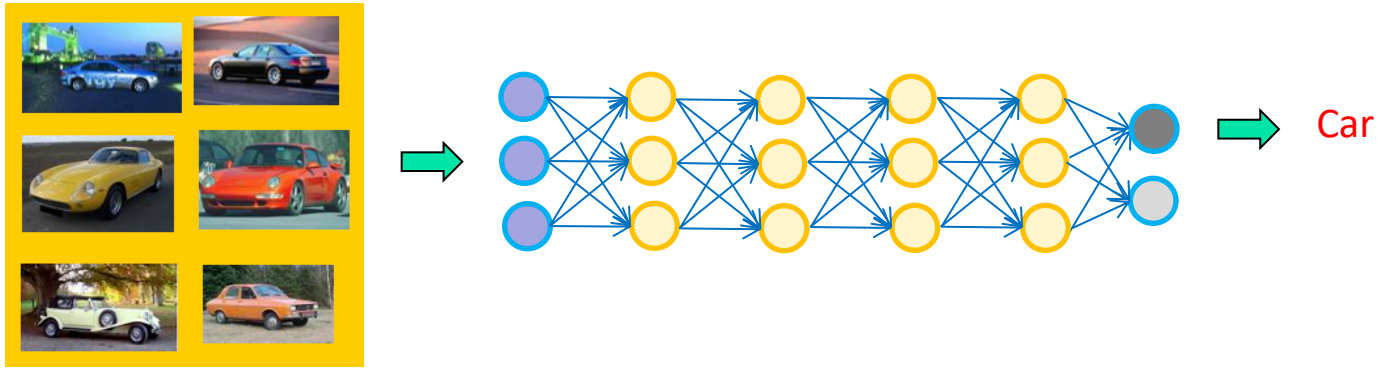
Deep Learning

Russell Chap. 18.7

Luger Chap. 11

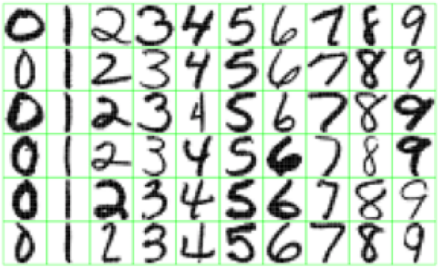
Deep Learning

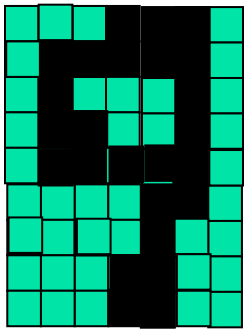
- Concept
  - Deep learning means using a neural network with several layers of nodes between input and output
  - The series of layers between input & output do feature identification

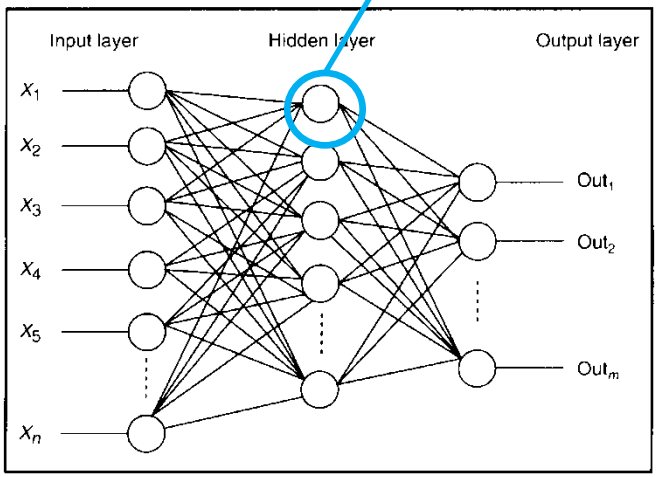


Deep Learning

- What does multilayer network do?
  - Character recogition



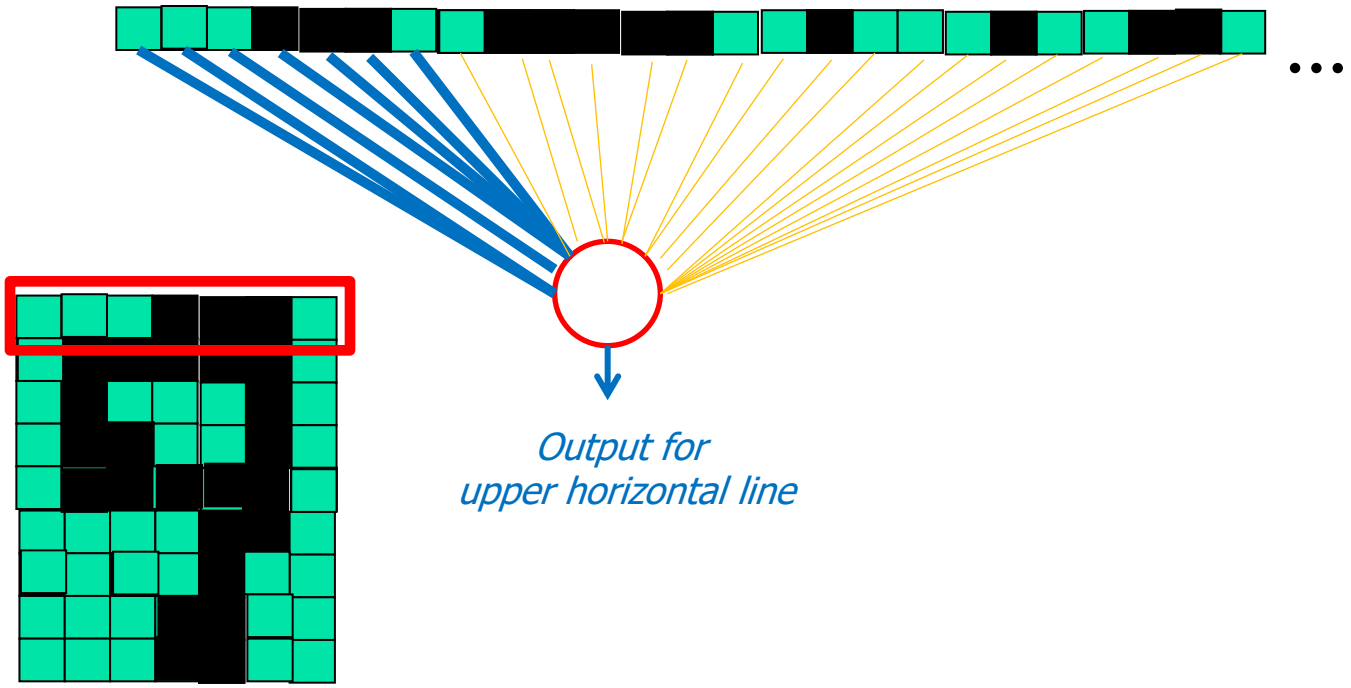




'9'

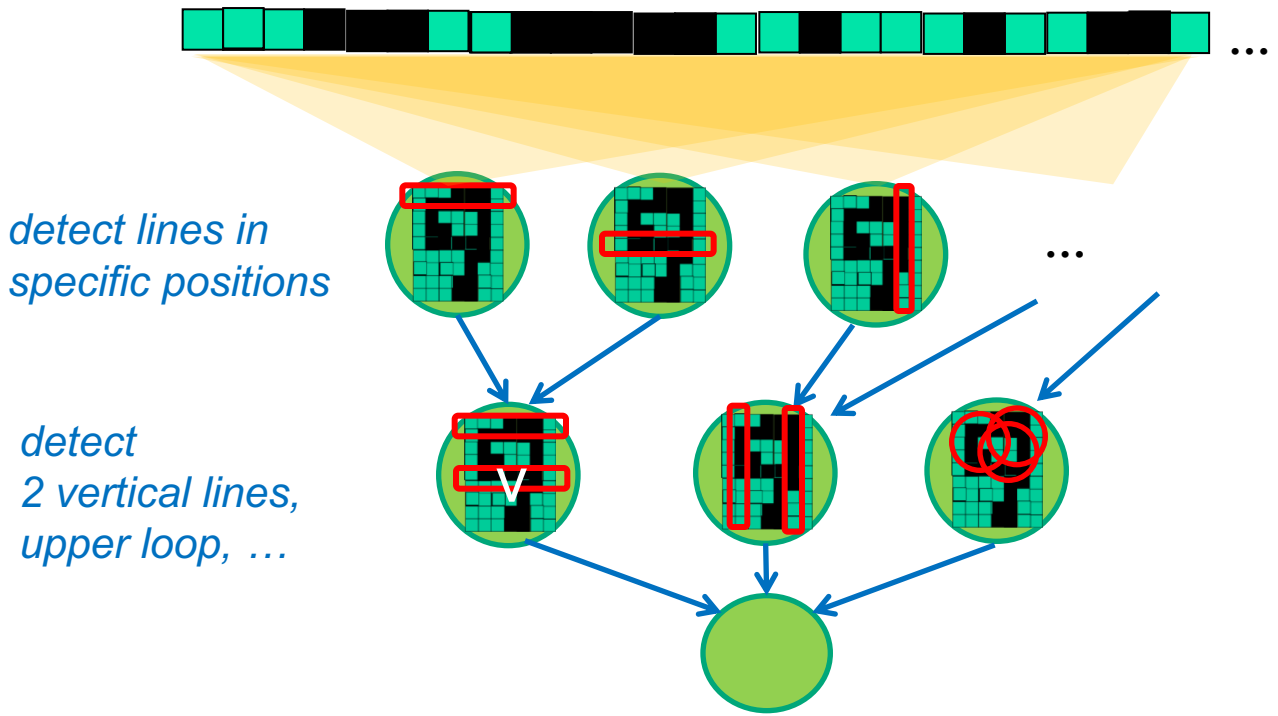
Deep Learning

- Feature detector
  - Hidden units work as feature detector



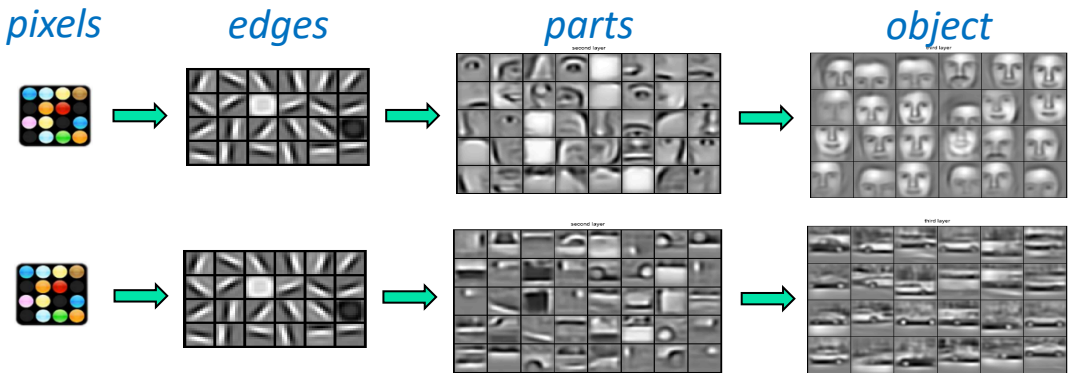
Deep Learning

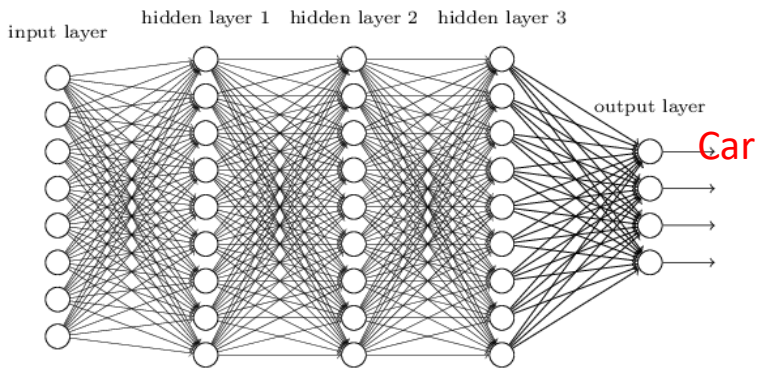
- Feature detector
  - Successive layers learn higher-level features

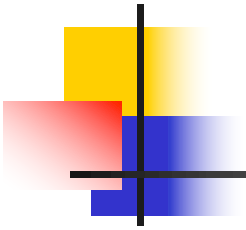


Deep Learning

- More complex problems → more features
  → more layers (deep structure) in the network







Deep Learning

- Various models
  - Convolution Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
  - Long-Short Term Memory (SLTM)
  - Autoencoders
  - Restricted Boltzman Machines (RBM)

  …

CNN (Convolutional Neural Net)

- A special kind of multilayer neural networks
  - Convolution
  - ReLU activation
  - Pooling (Subsampling)
- Learning by backpropagation
  - Implicitly extract relevant features
  - Topological properties from an image

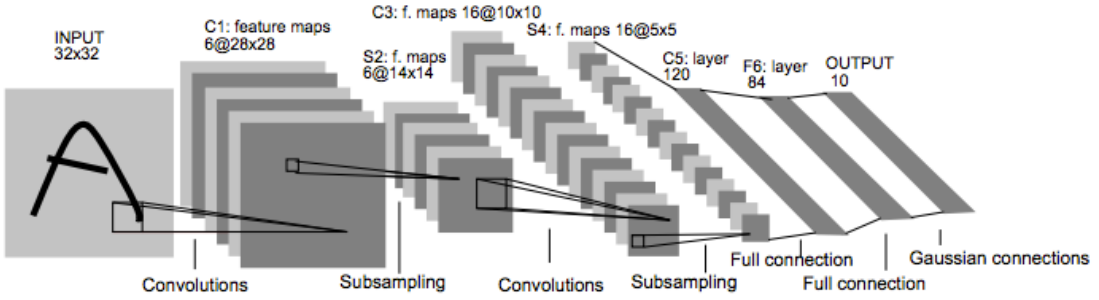
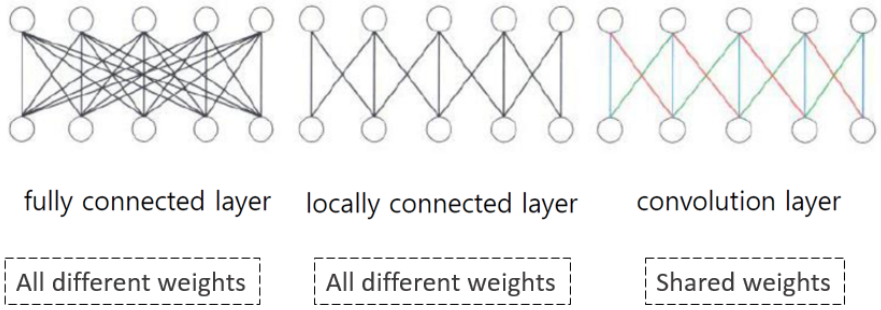
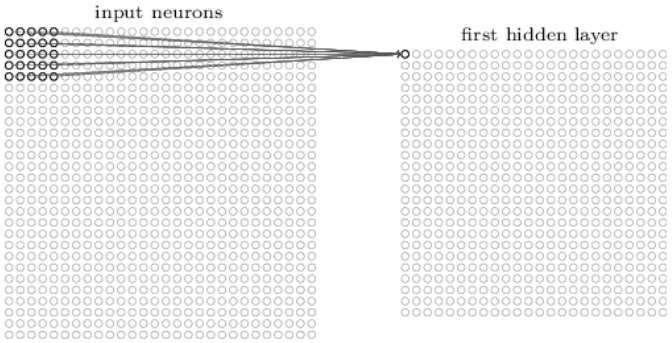


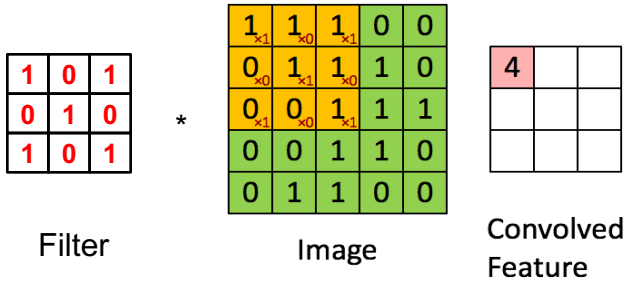
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

CNN (Convolutional Neural Net)

- Convolution in neural network
  - Smaller number of of connection
  - Weight sharing
  - Detect features at different positions





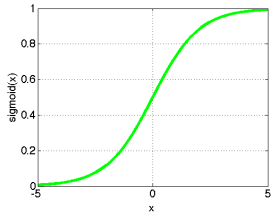


CNN (Convolutional Neural Net)

- ReLU(Rectified Linear Unit) for output function
  - Helps vanishing gradient problem
  - Makes learning faster

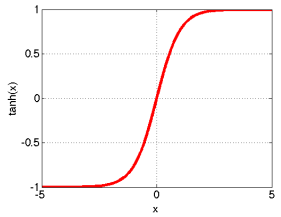
    - Sigmoid

$$y = \frac{1}{1 + e^{-x}}$$



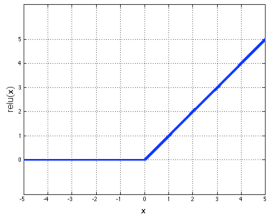
    - Tanh

$$y = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



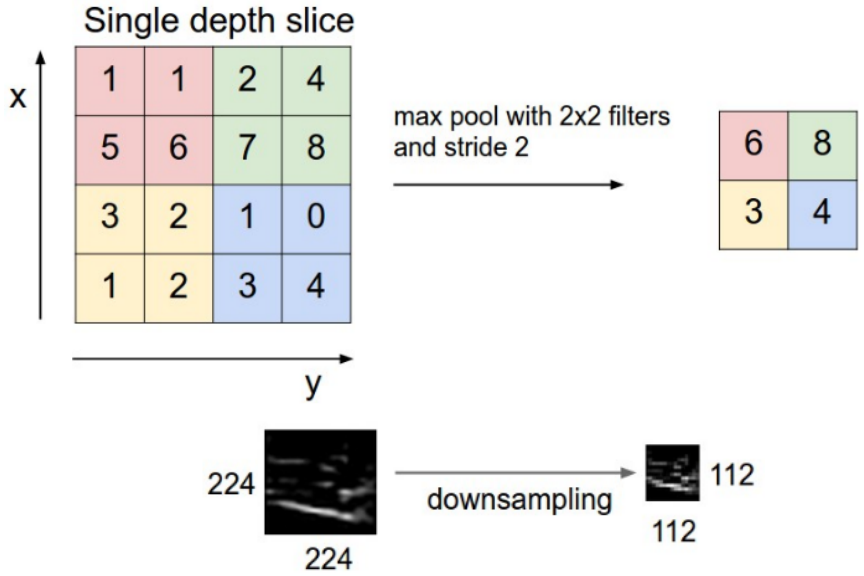
    - ReLU

$$y = \max(0, x)$$



CNN (Convolutional Neural Net)

- Pooling
  - Spatial Subsampling
  - Sum or max
  - Invariance to small transformations
  - Reduce the effect of noises/distortions



http://cs231n.github.io/convolutional-networks/

CNN Example

- A toy ConvNet example
  - From “How Convolutional Neural Networks work” by Brandon Rohrer
- The problem: image classification

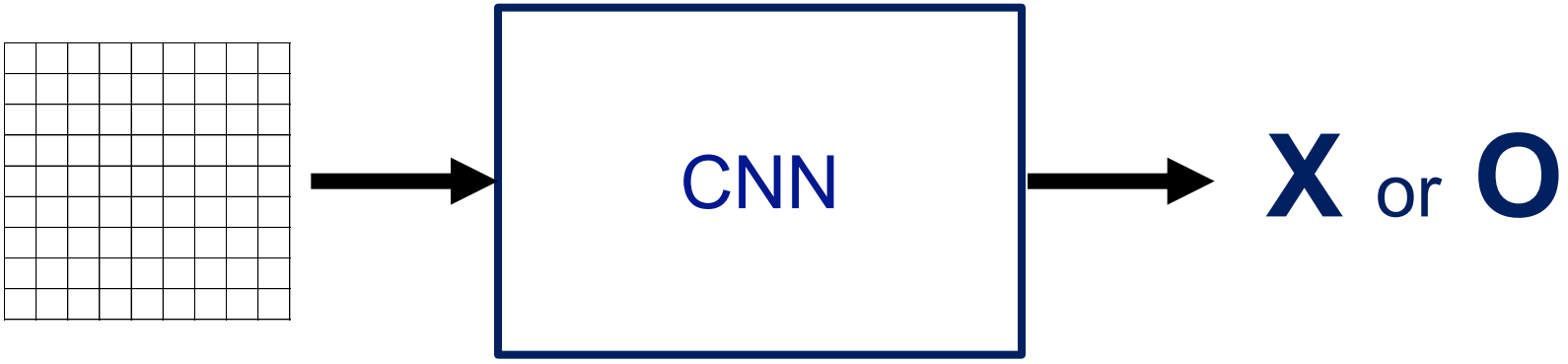
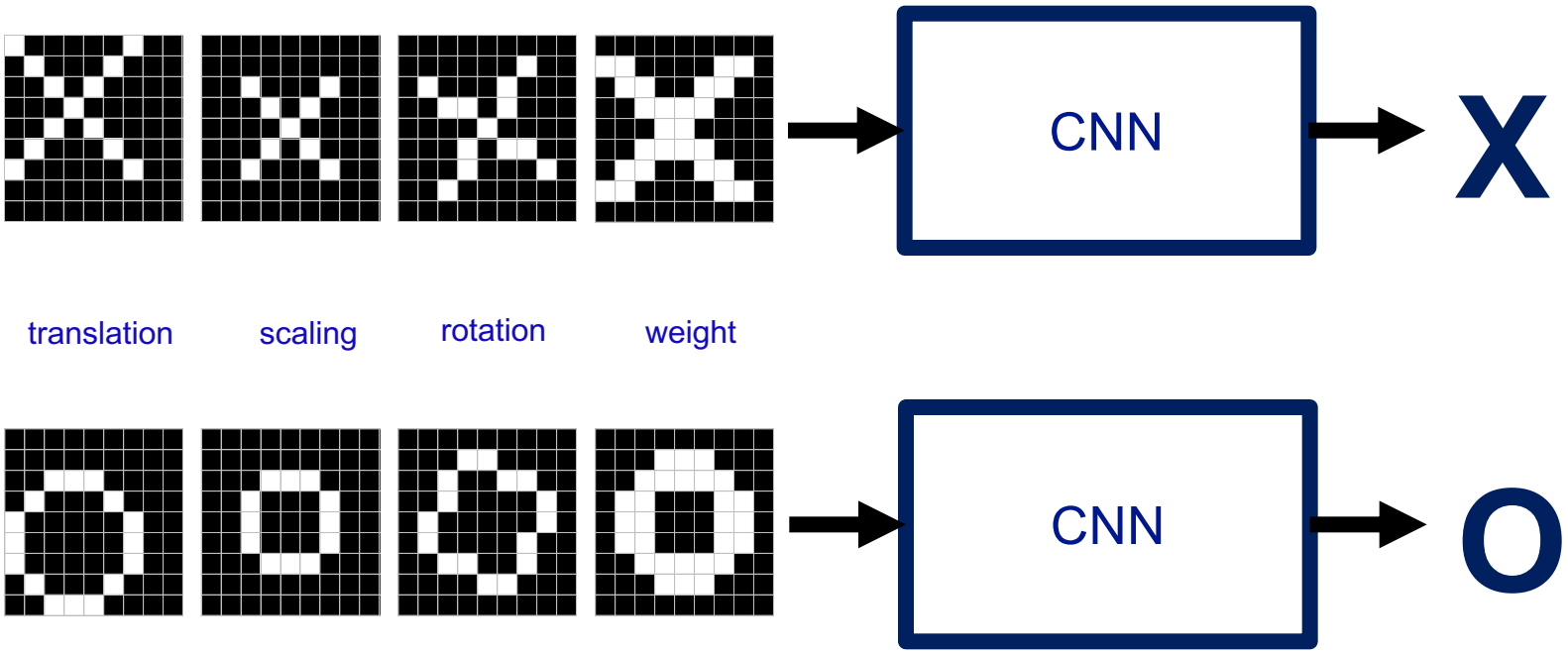


Image
(2D array of pixels)

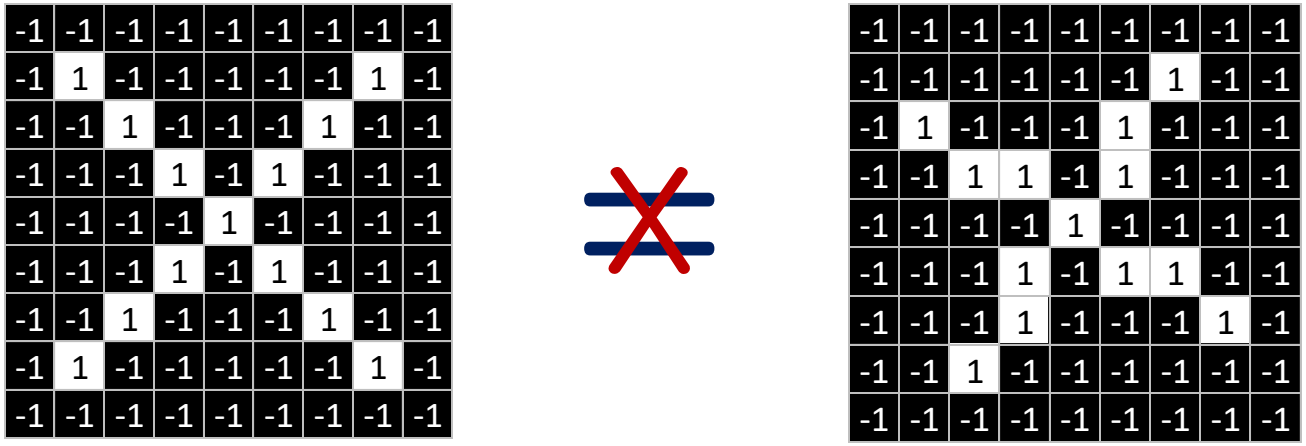
CNN Example

- For example,



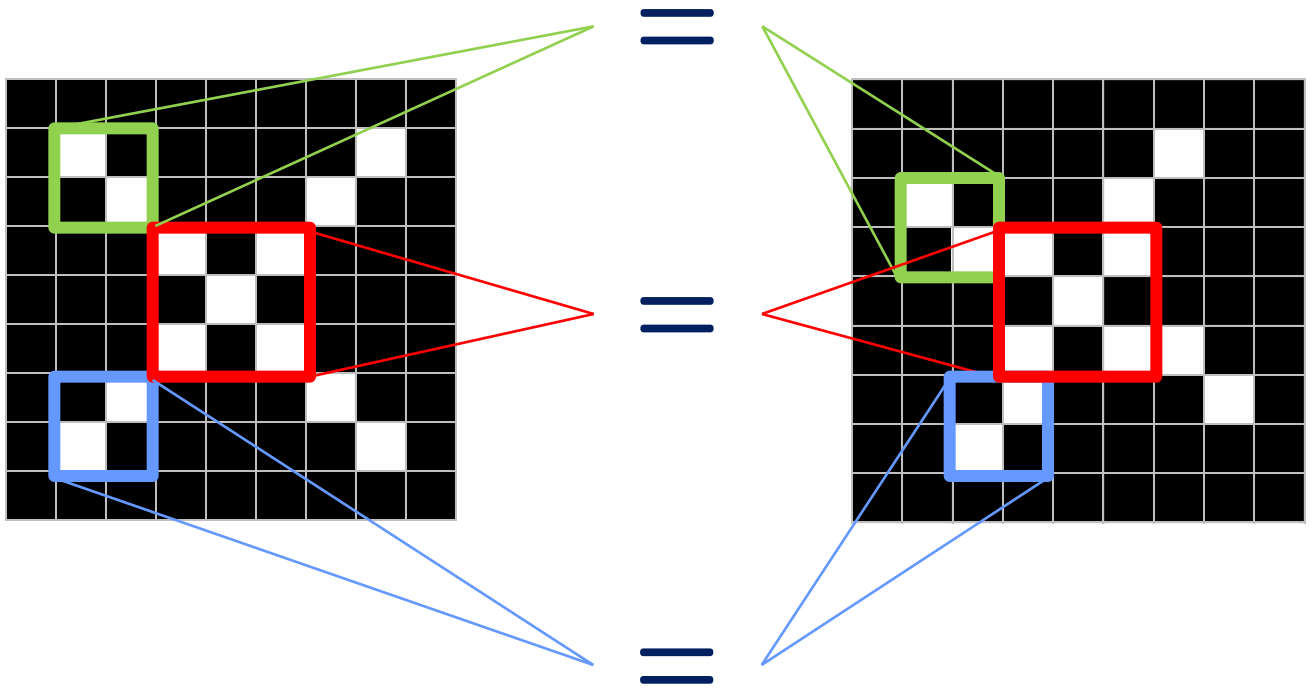
CNN Example

- What computers see



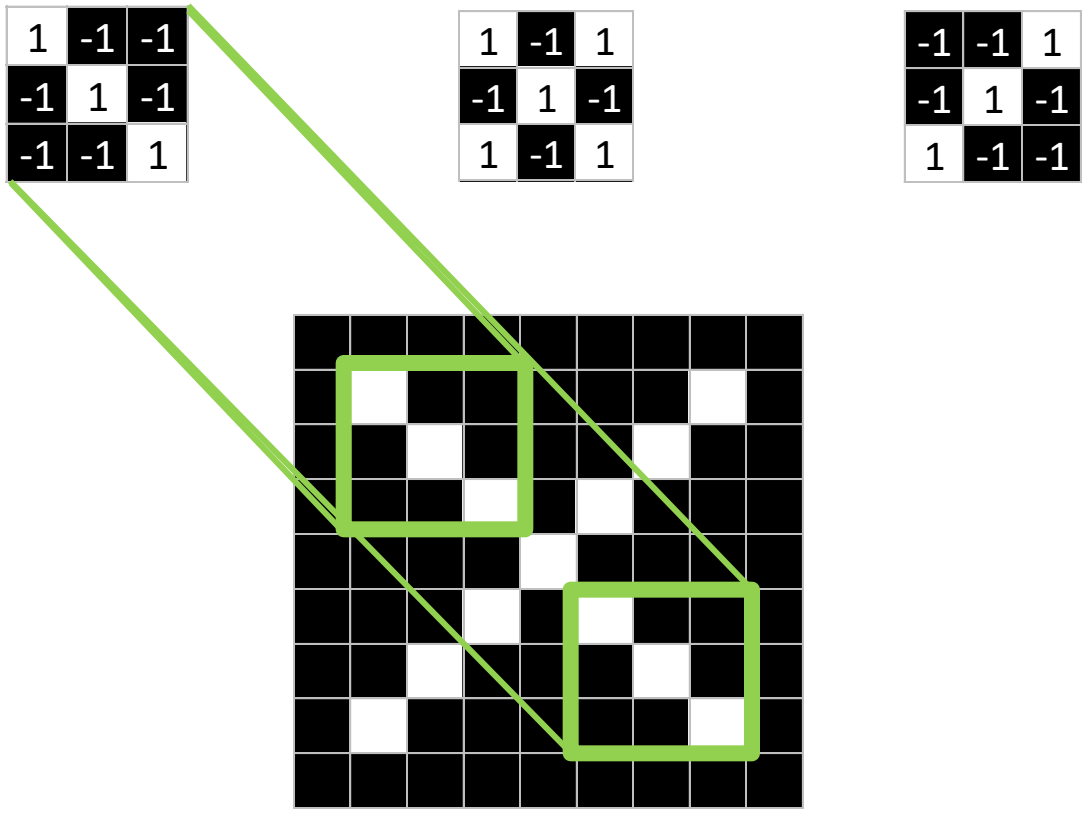
CNN Example

- ConvNets match pieces of the image



CNN Example

- Detecting features using filters



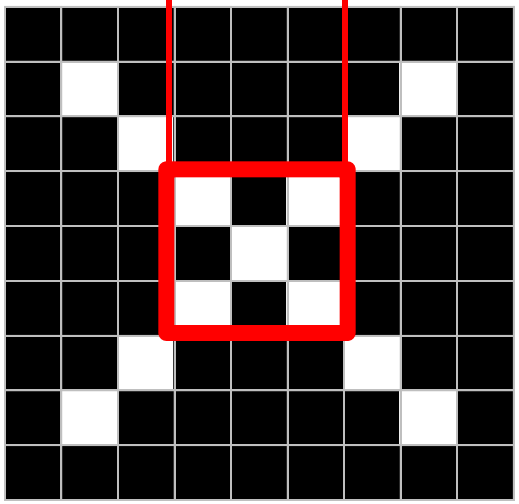
CNN Example

- Detecting features using filters

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

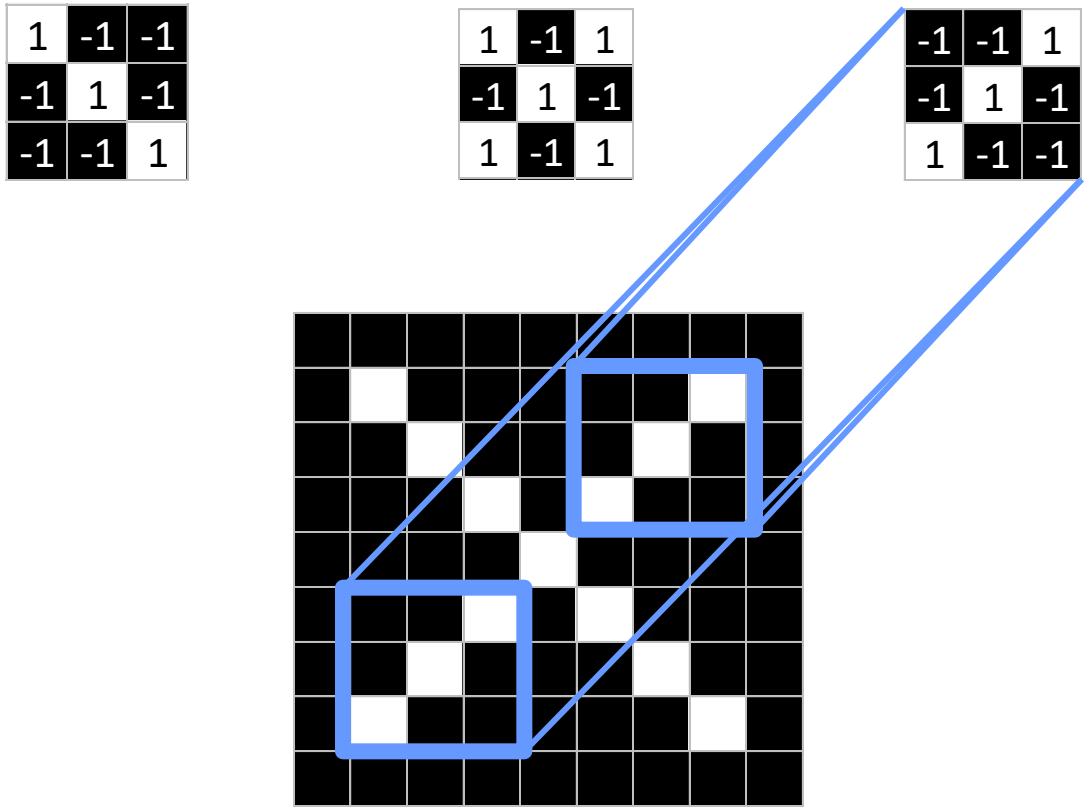
1	-1	1
-1	1	-1
1	-1	1

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



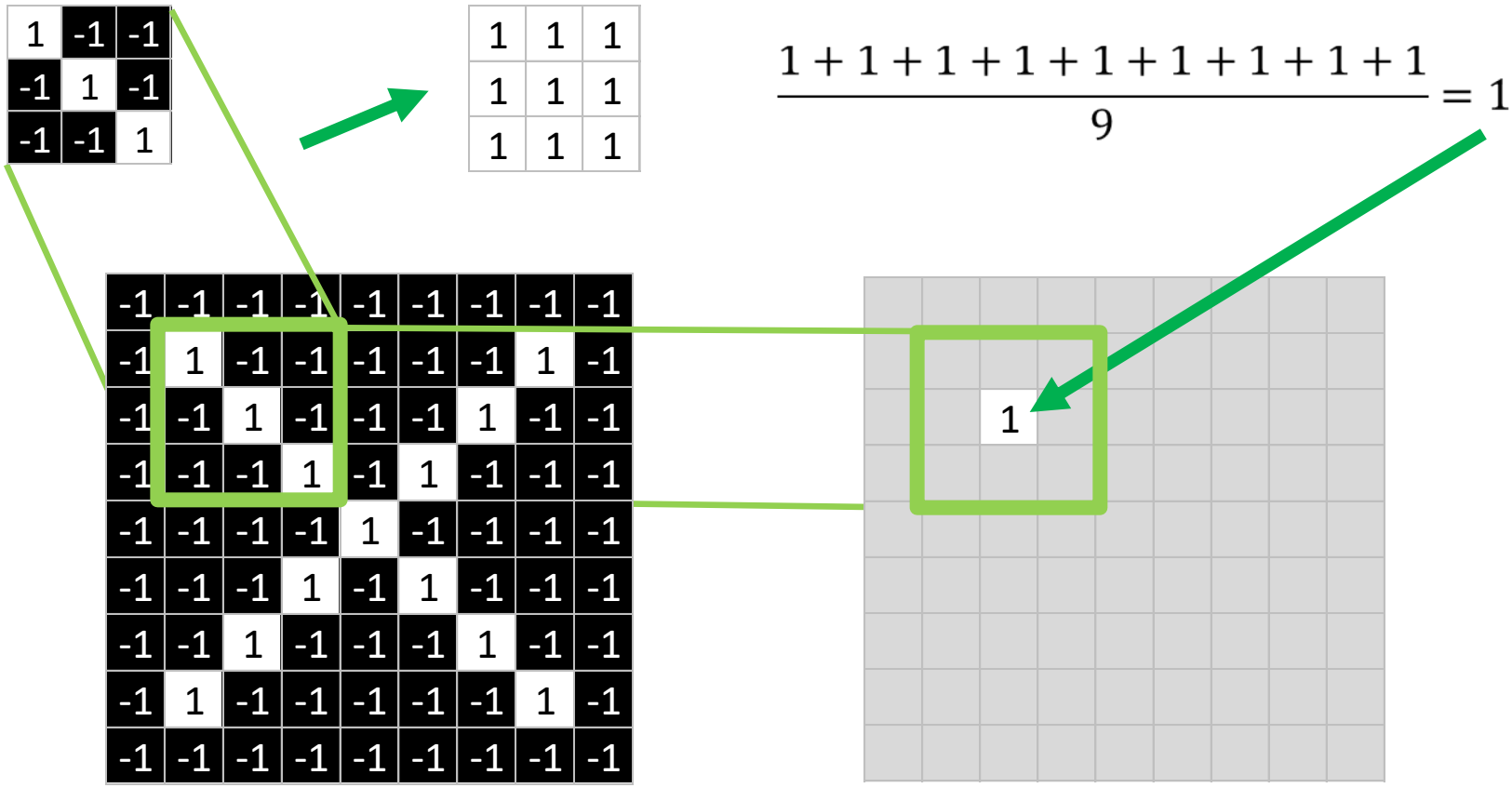
CNN Example

- Detecting features using filters



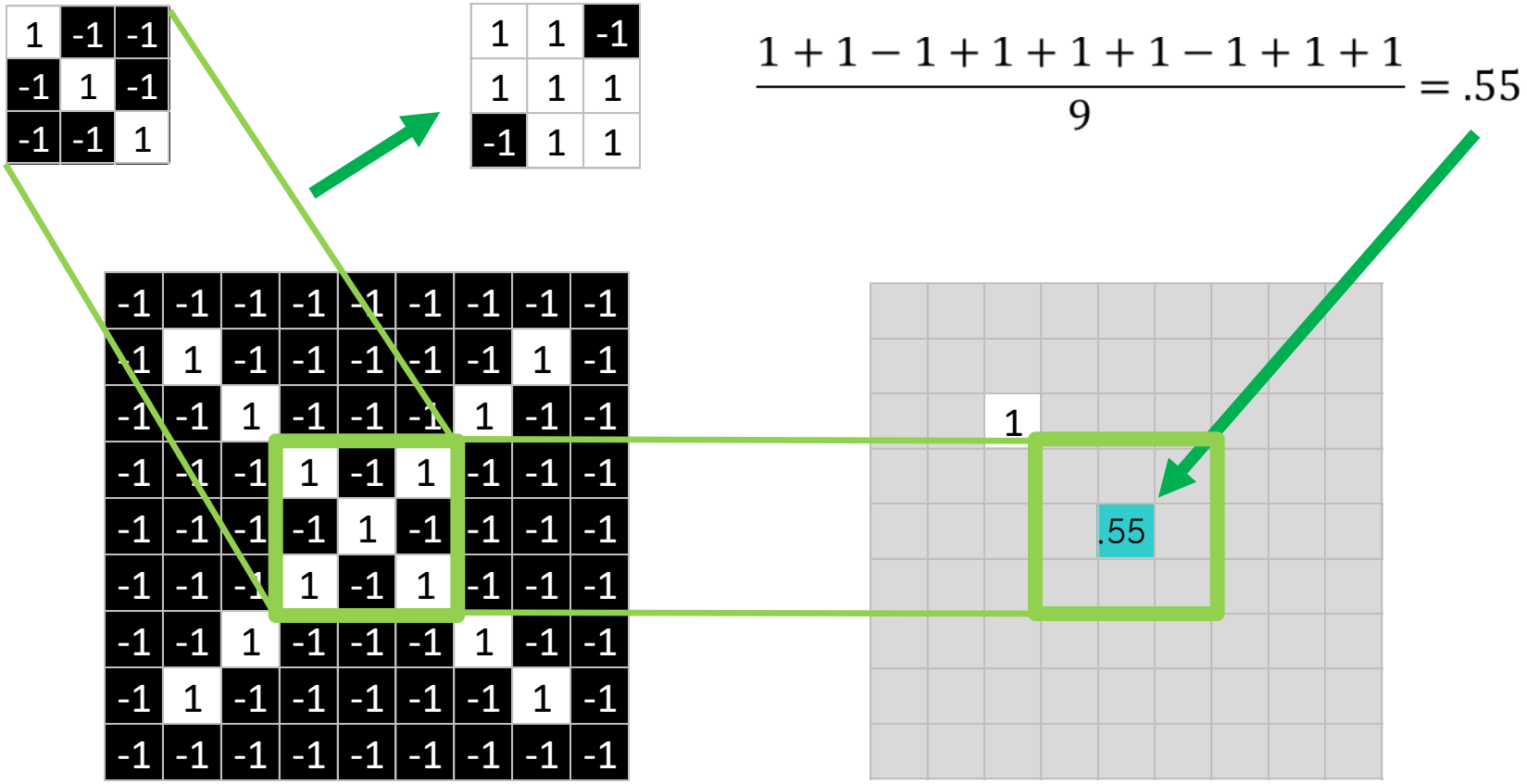
CNN Example

- Convolution: Filtering computation



CNN Example

- Convolution: Filtering computation



CNN Example

- Convolution: Filtering computation

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

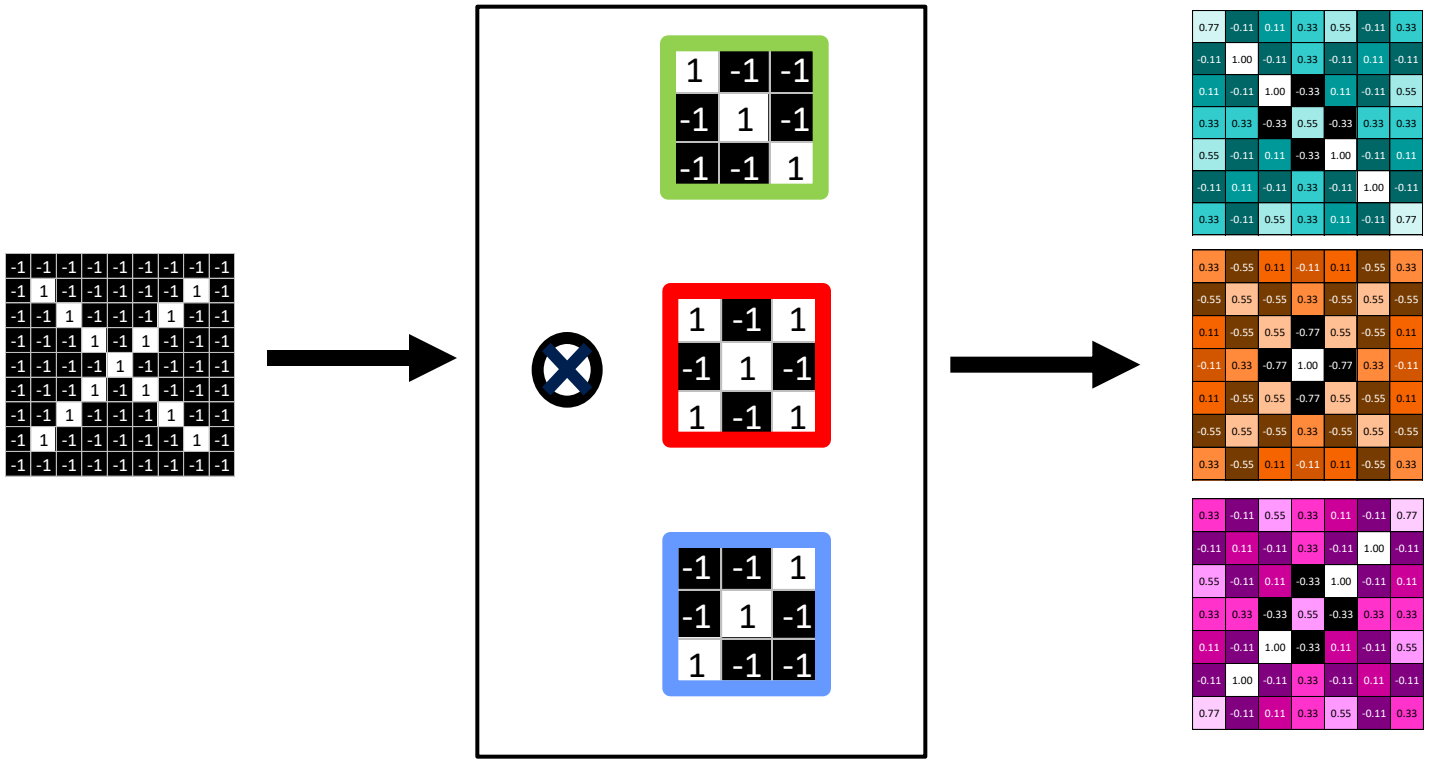
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Convolution

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

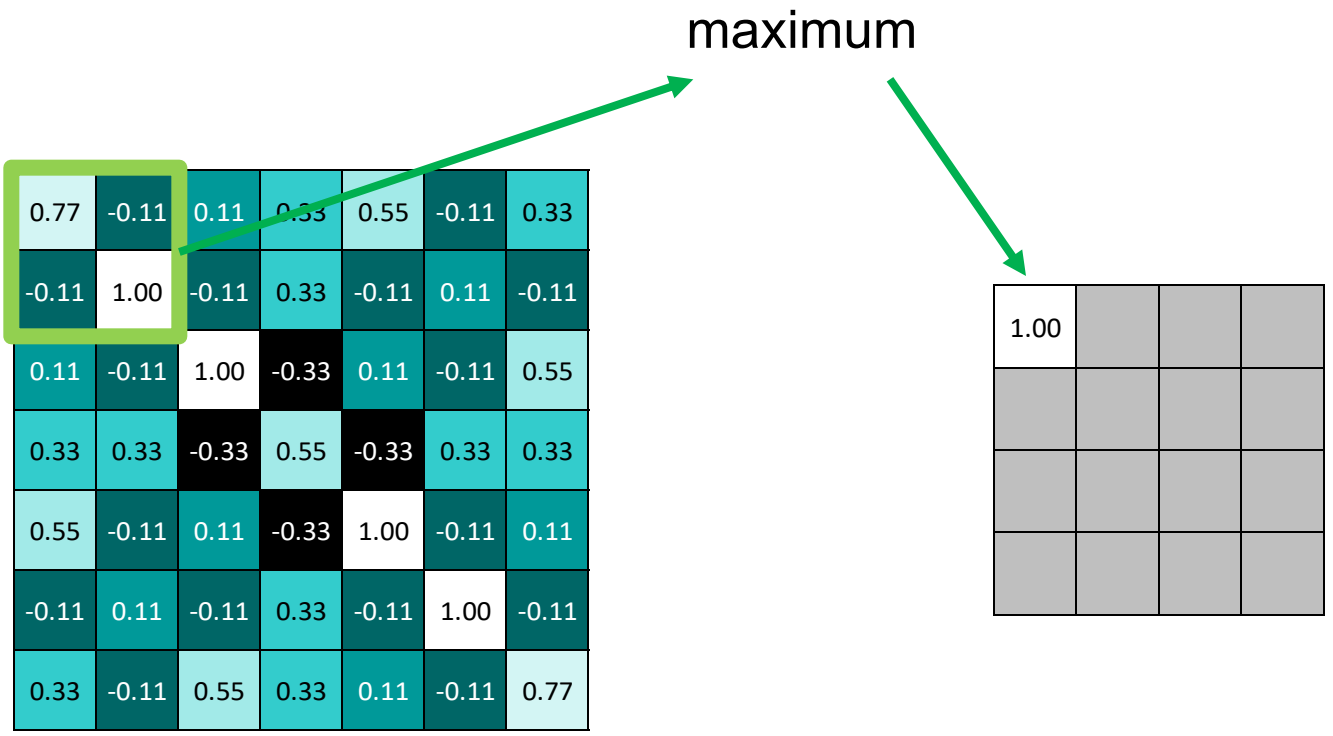
CNN Example

- Convolution: Filtering computation



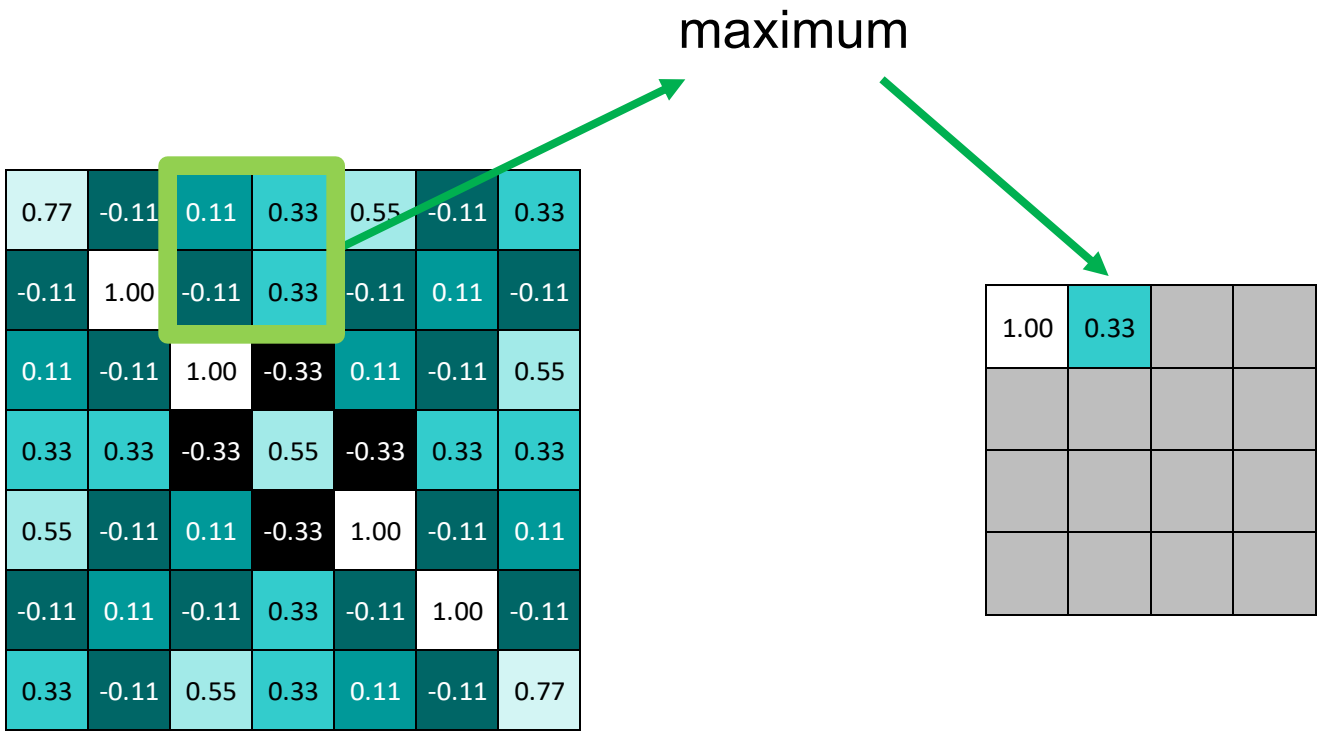
CNN Example

- Pooling: Shrinking image



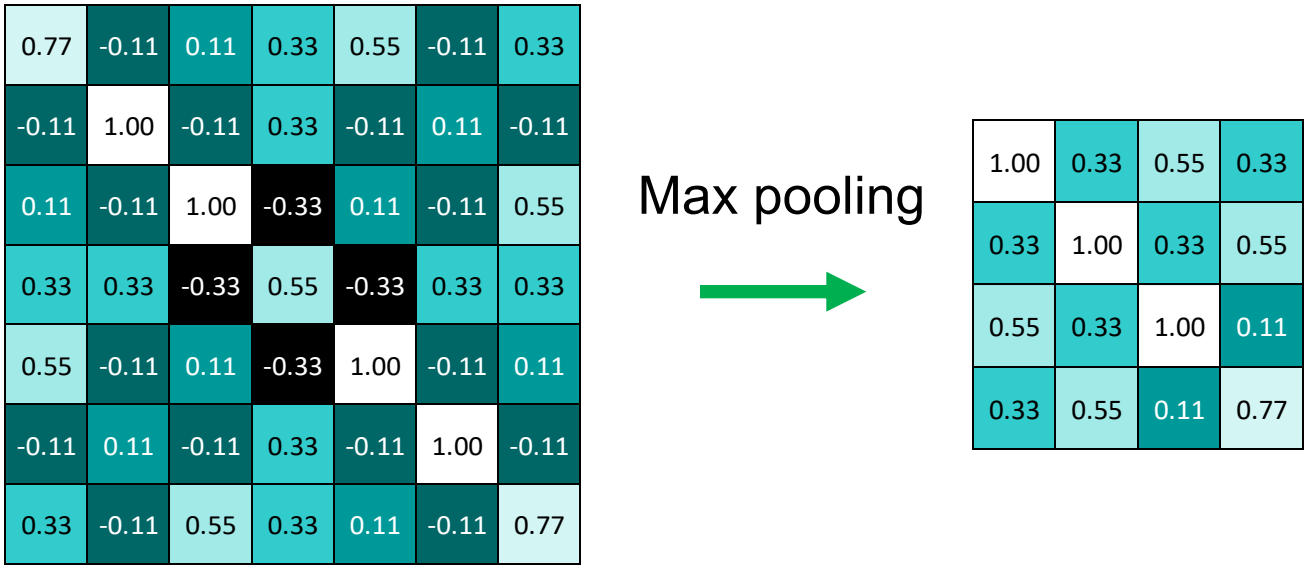
CNN Example

- Pooling: Shrinking image



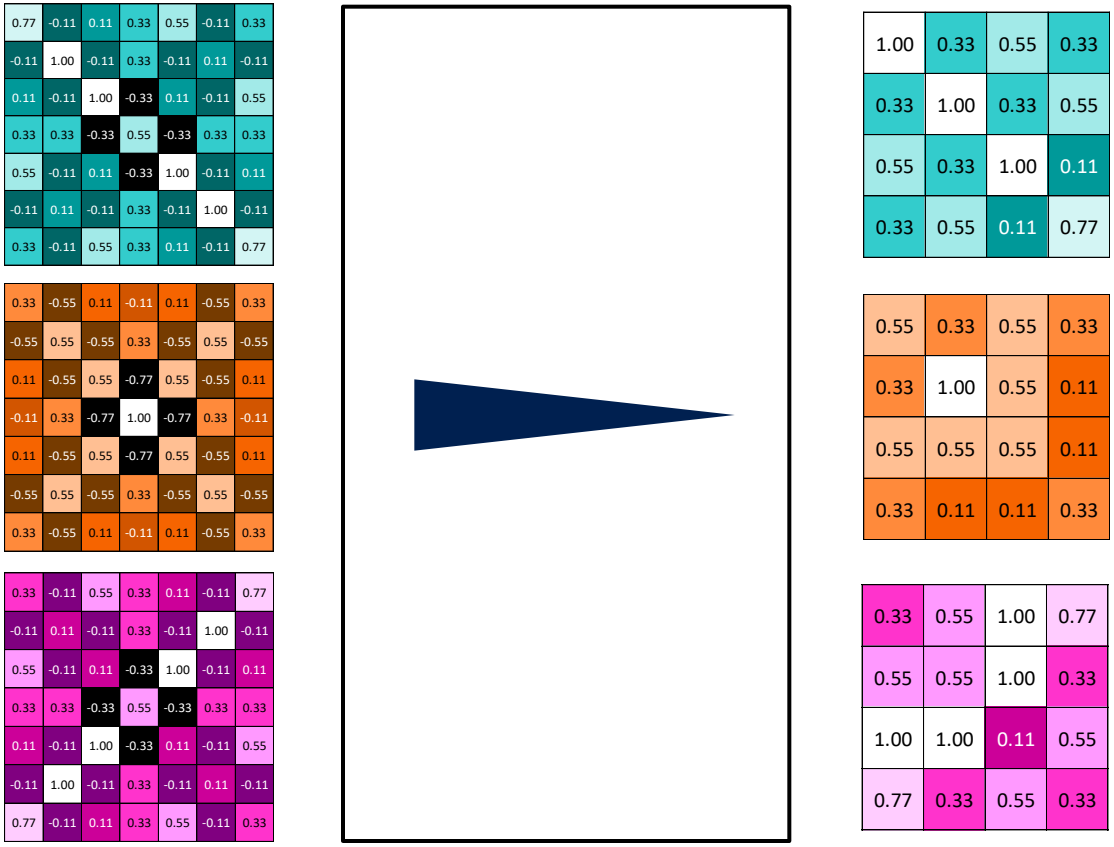
CNN Example

- Pooling: Shrinking image



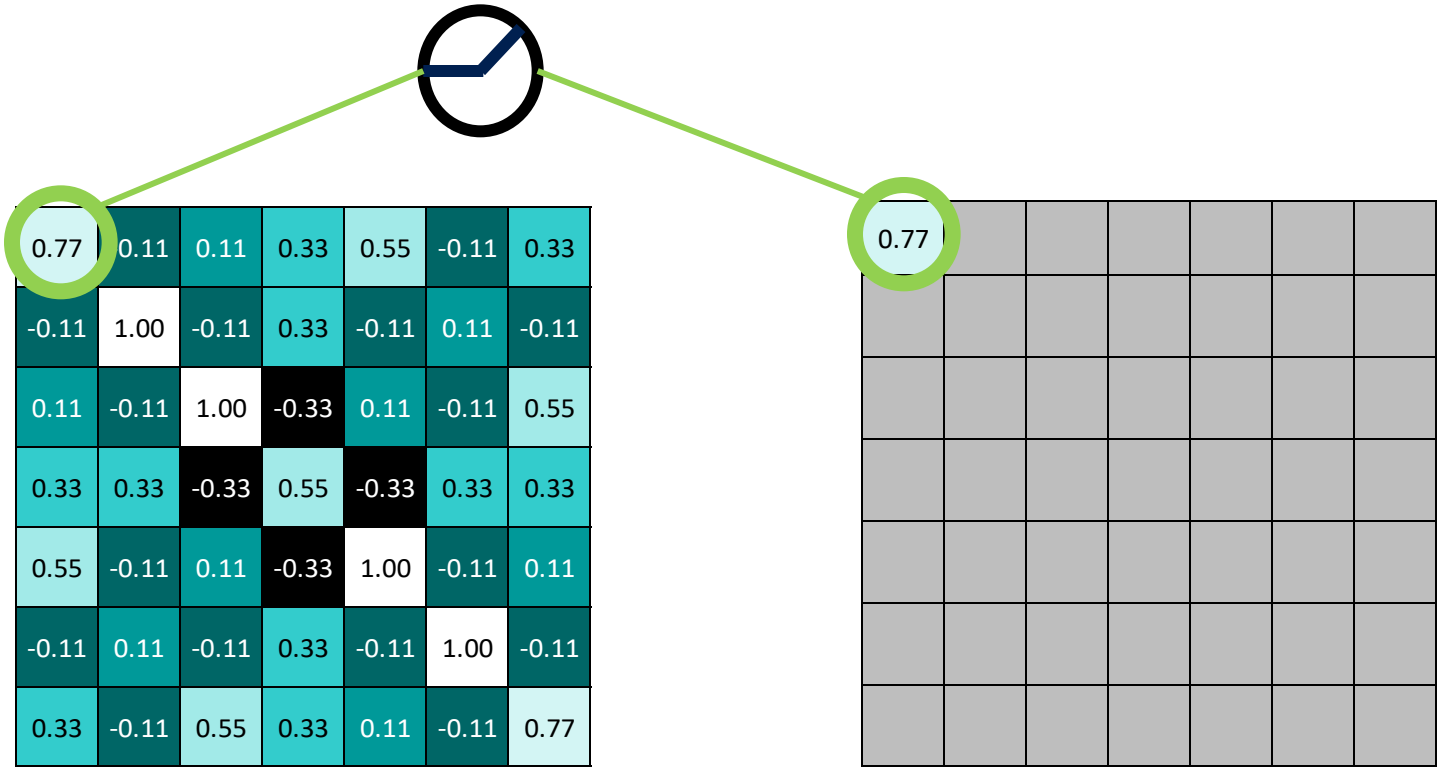
CNN Example

- Pooling: Shrinking image



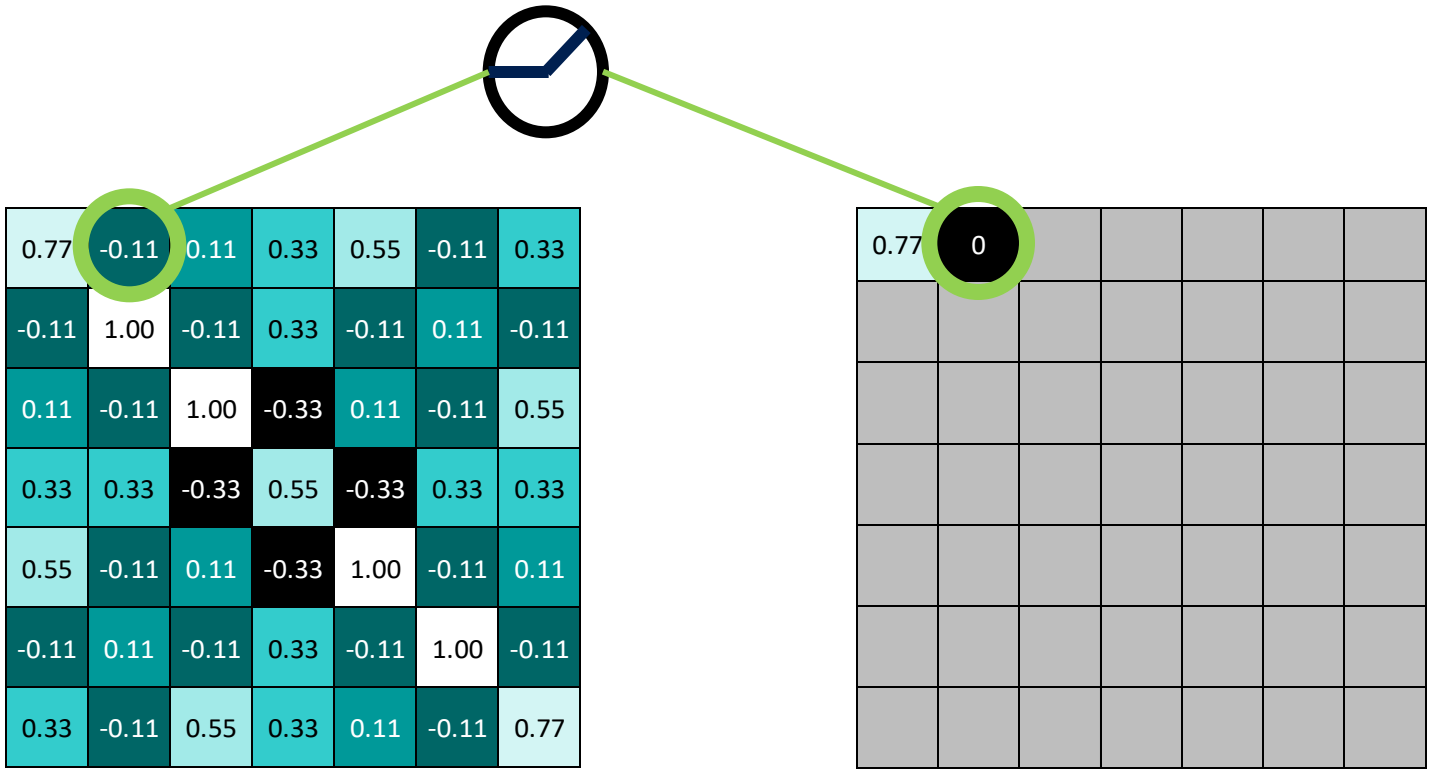
CNN Example

- Applying ReLU



CNN Example

- Applying ReLU



CNN Example

- Applying ReLU

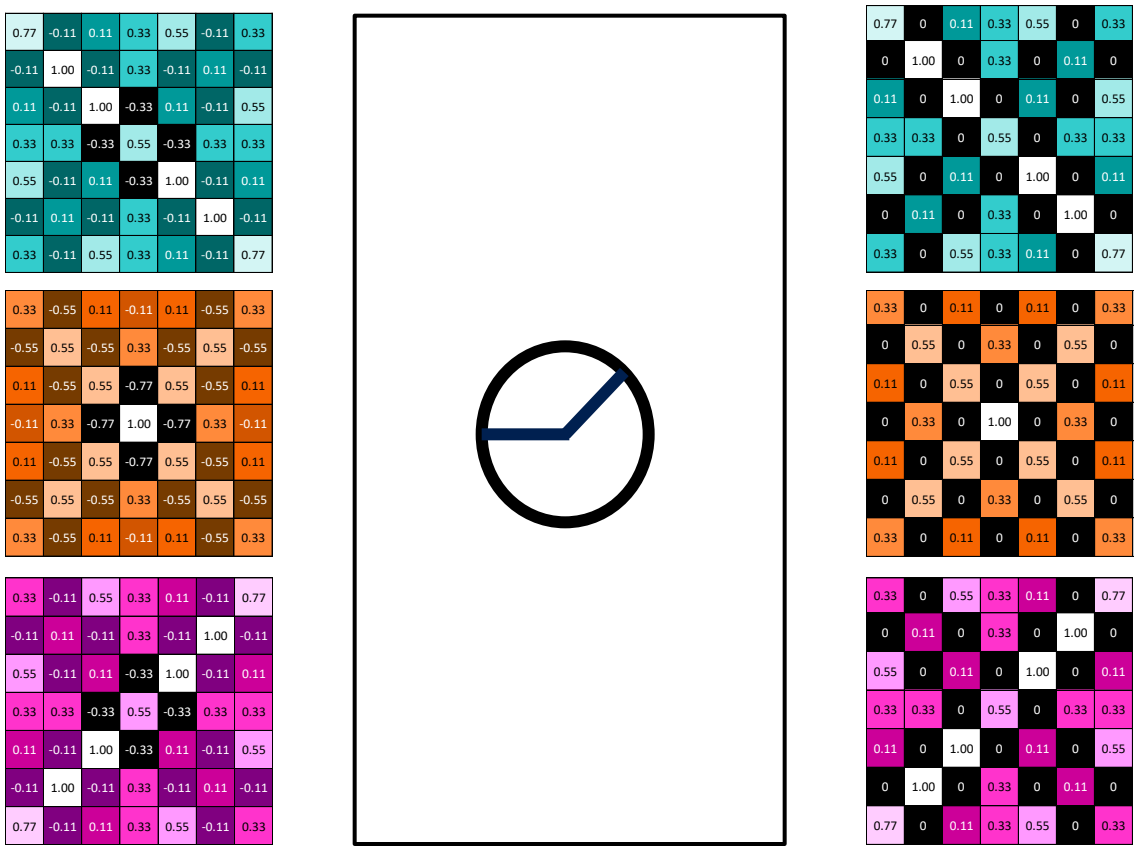
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Normalization

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

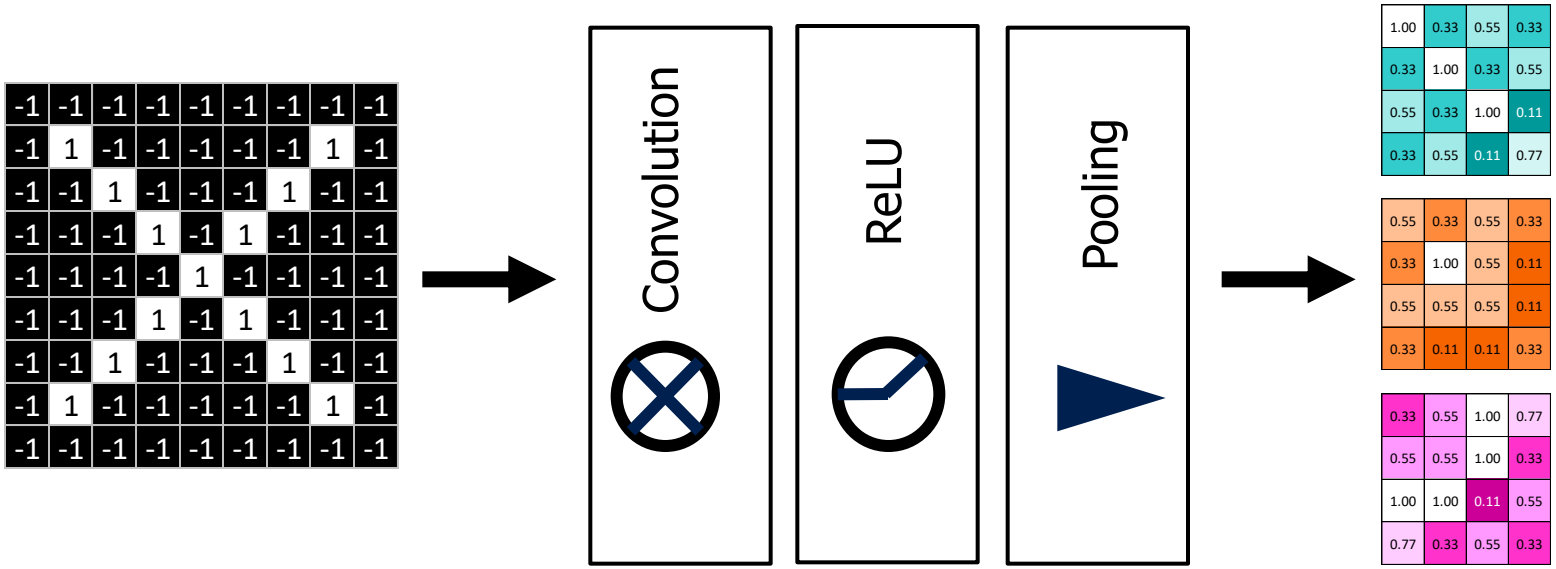
CNN Example

- Applying ReLU



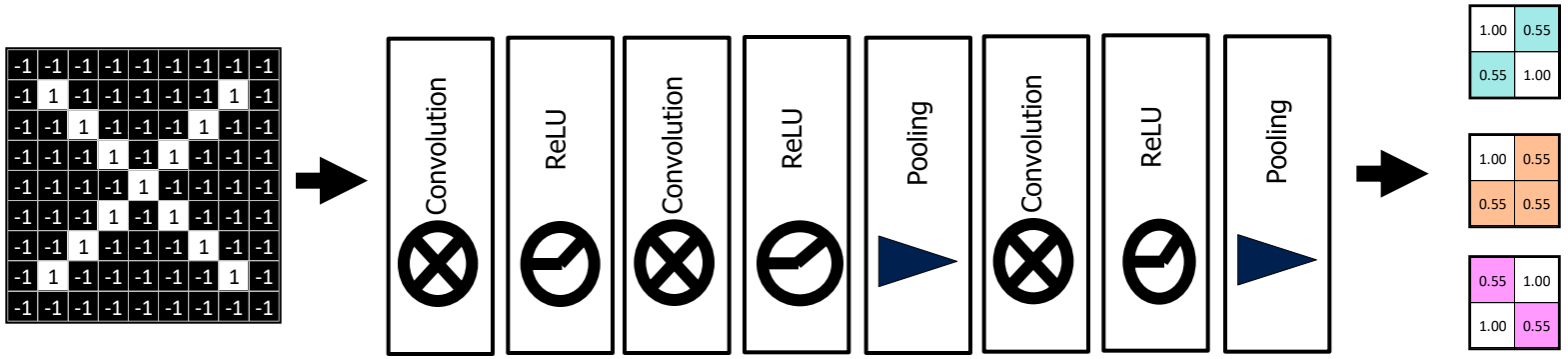
CNN Example

- Stacking layers



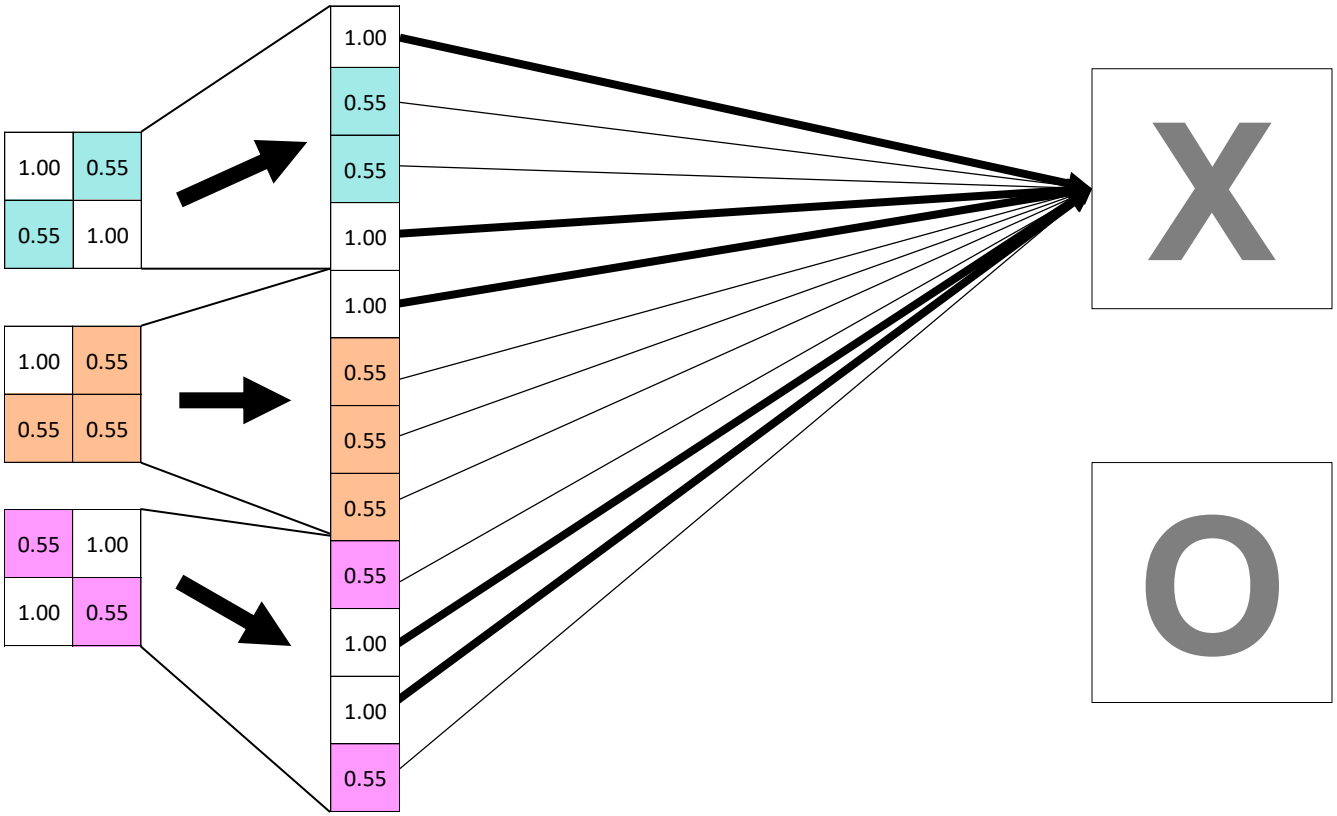
CNN Example

- Stacking layers



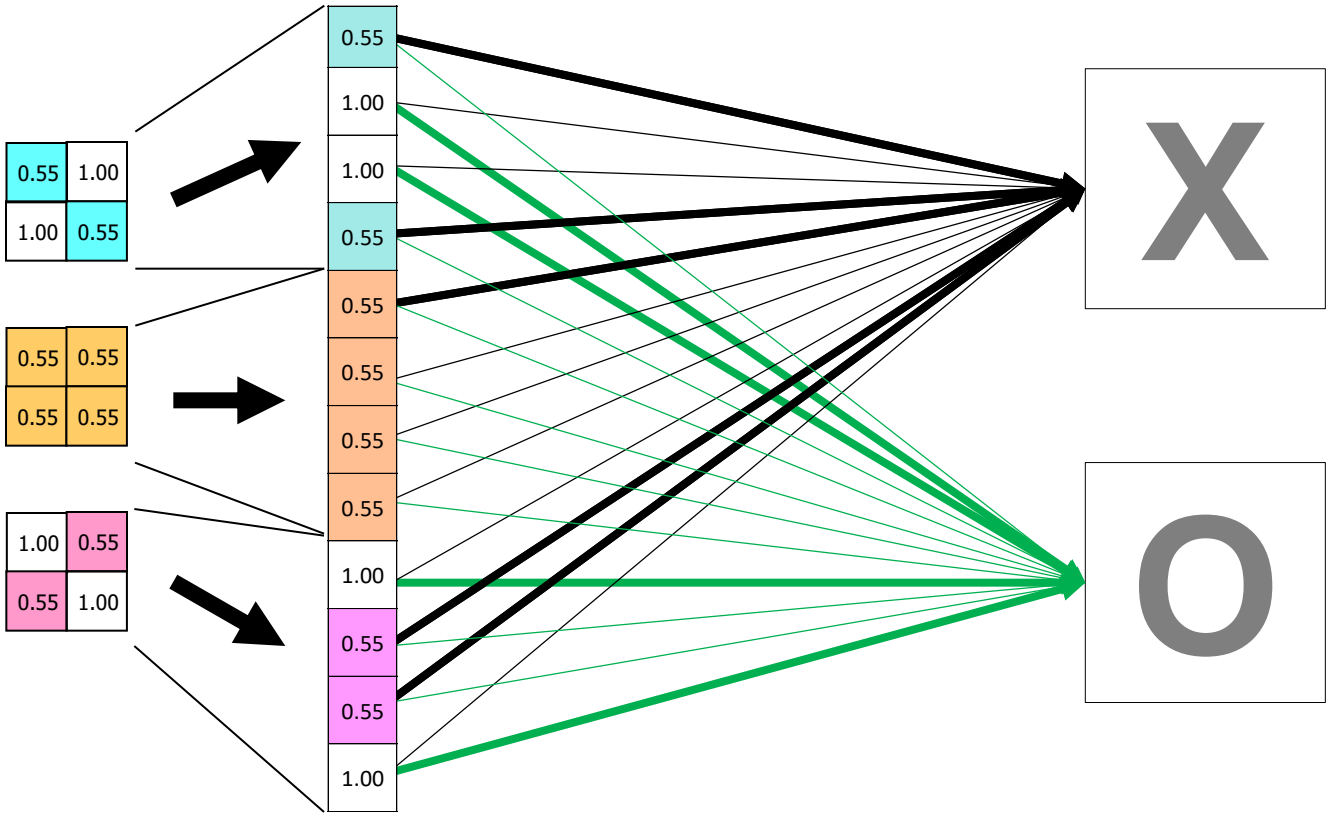
CNN Example

- Fully connected layer
  - Vote depends on how strongly a value predicts X or O



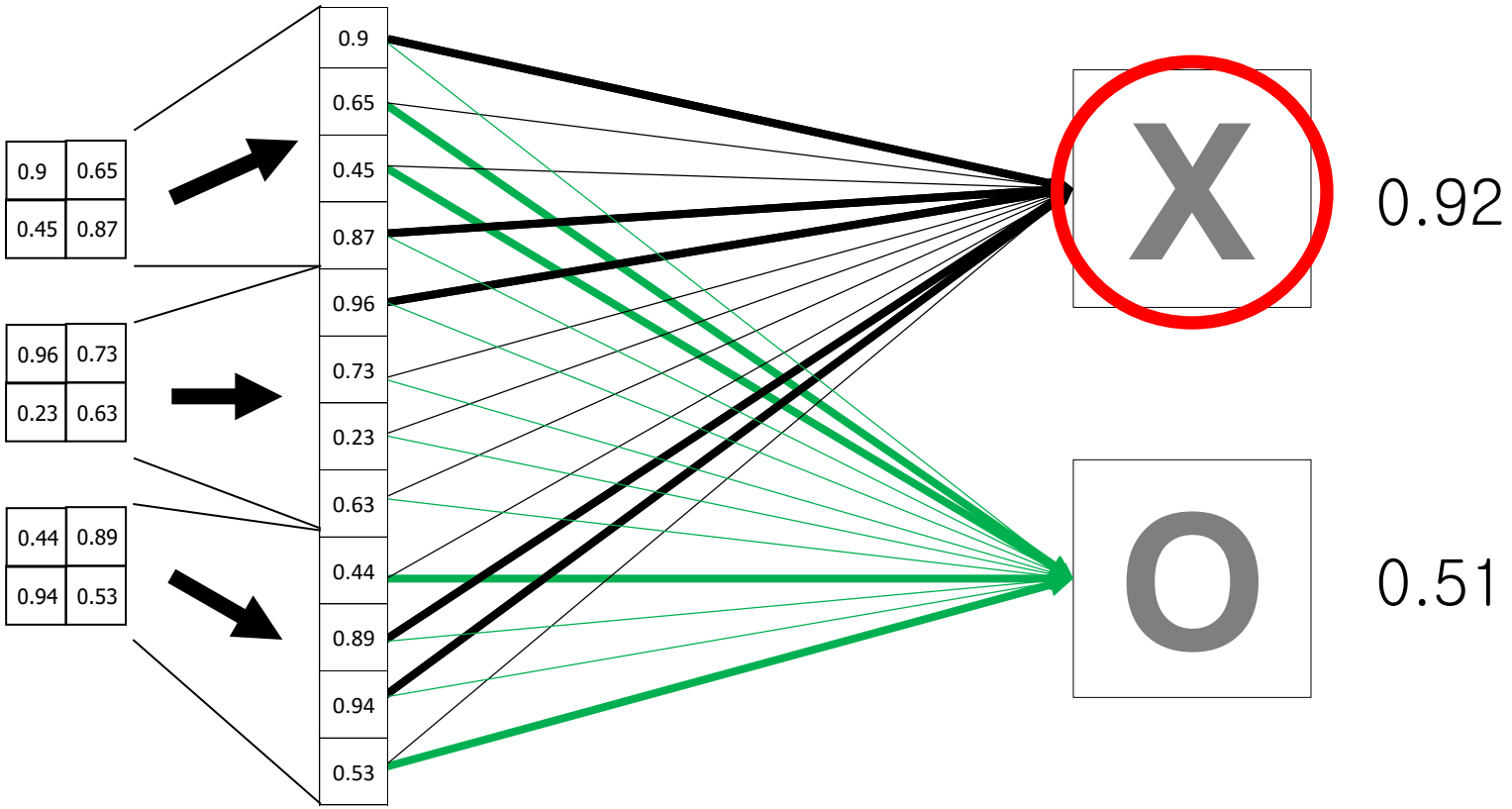
CNN Example

- Fully connected layer
  - Vote depends on how strongly a value predicts X or O



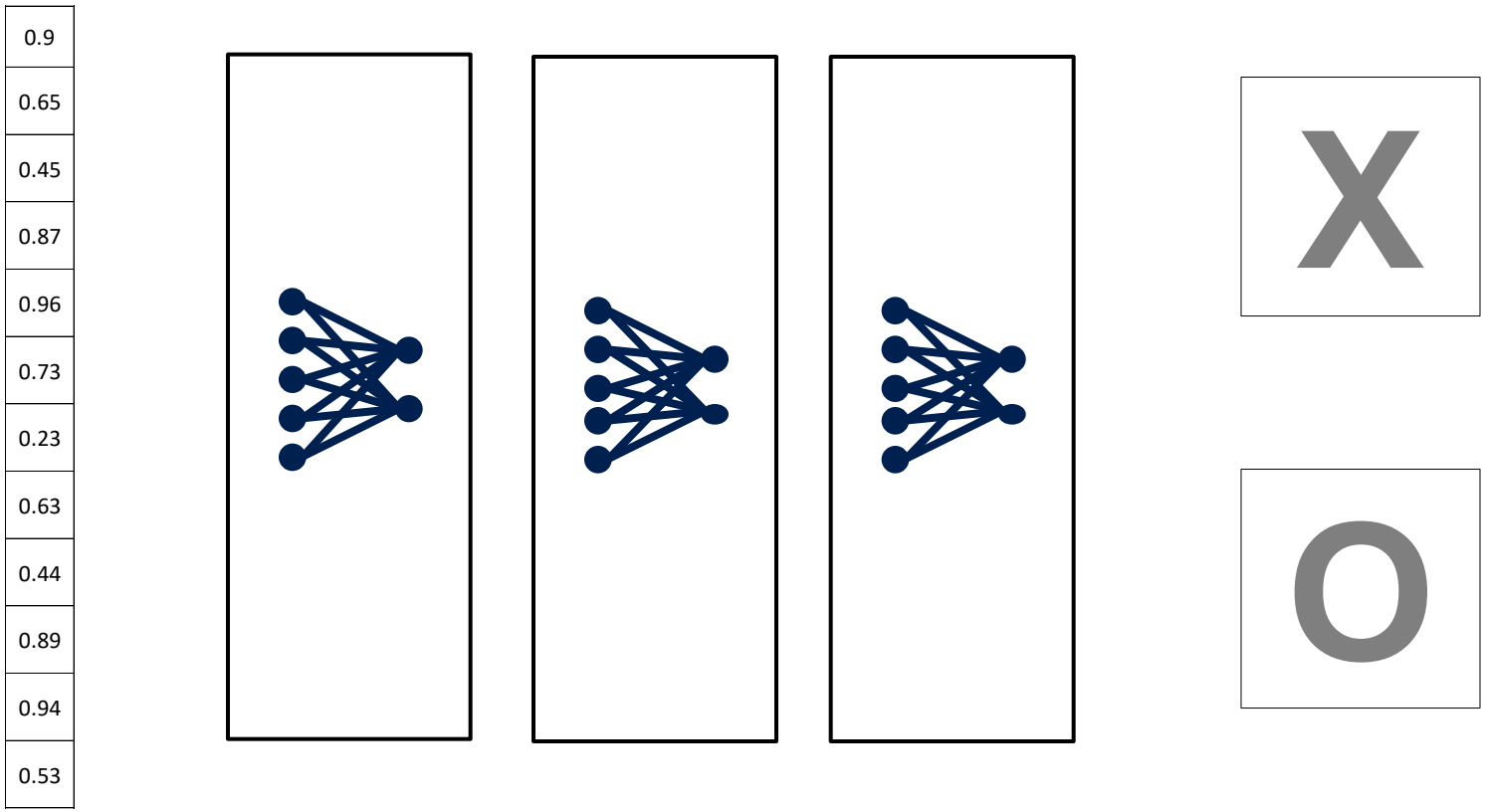
CNN Example

- Fully connected layer
  - For new data,



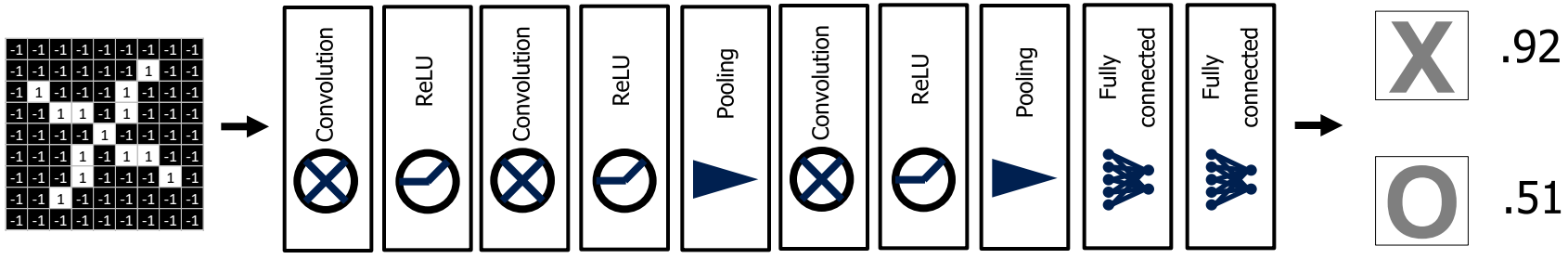
CNN Example

- Fully connected layer can also be multilayered



CNN Example

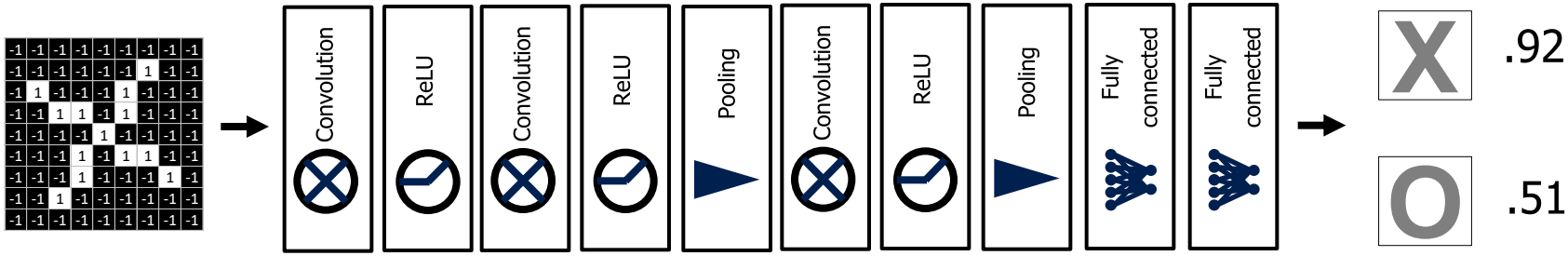
- Putting it altogether



CNN

CNN Example

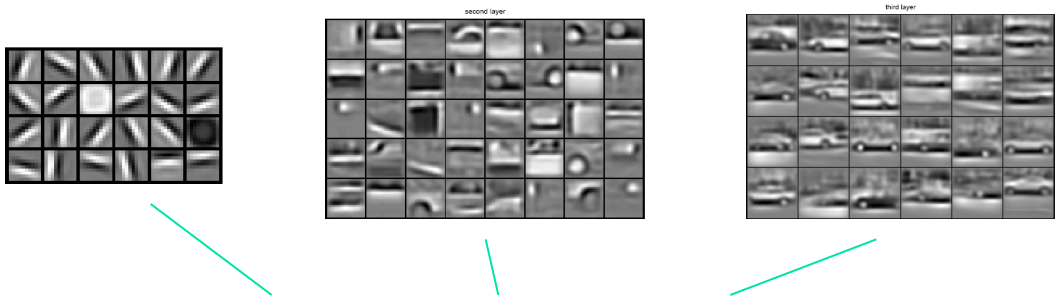
- Learning



Output unit	d (right answer)	O (Actual answer)	Error
X	1	0.92	0.08
O	0	0.51	0.49
Total			0.57

$$\Delta w = -c \cdot \frac{\partial E}{\partial w}$$

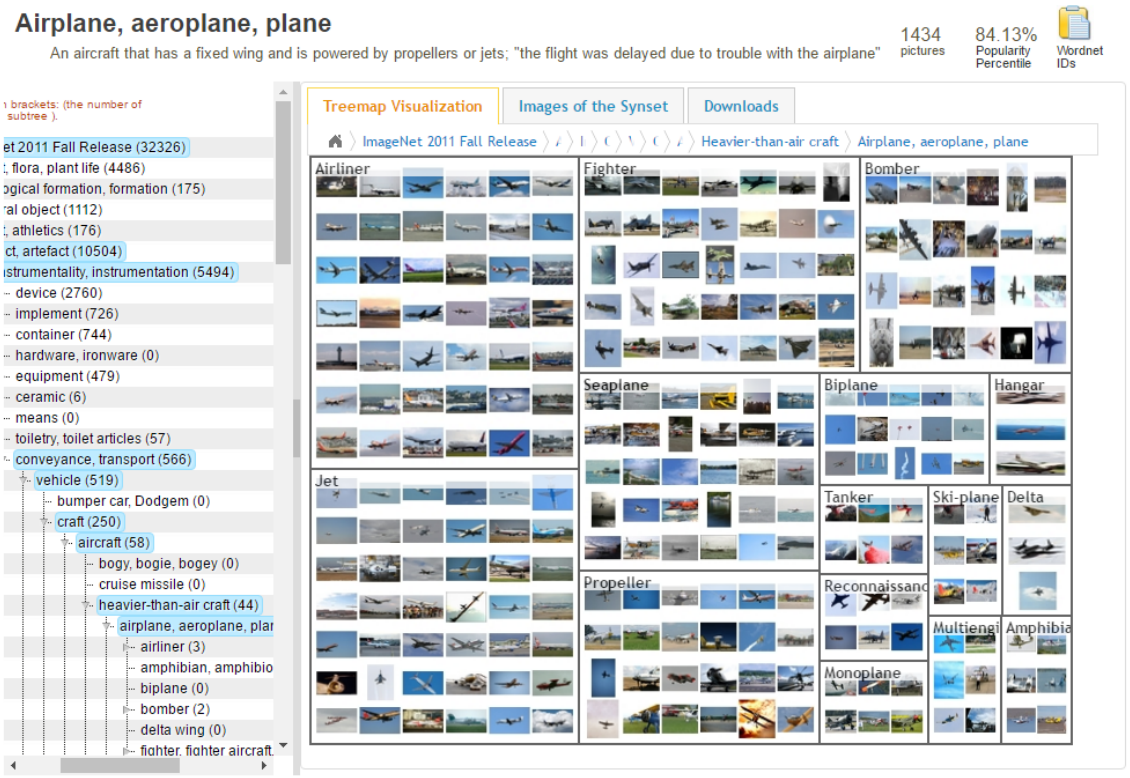
CNN Example

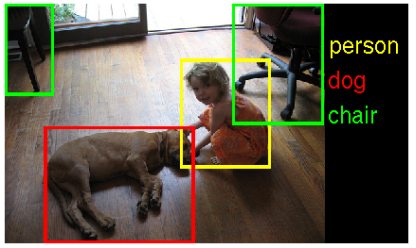


Car

Image Recognition

- ILSVRC (ImageNet Large Scale Visual Recognition Challenge) 2012
  - Image classification competition

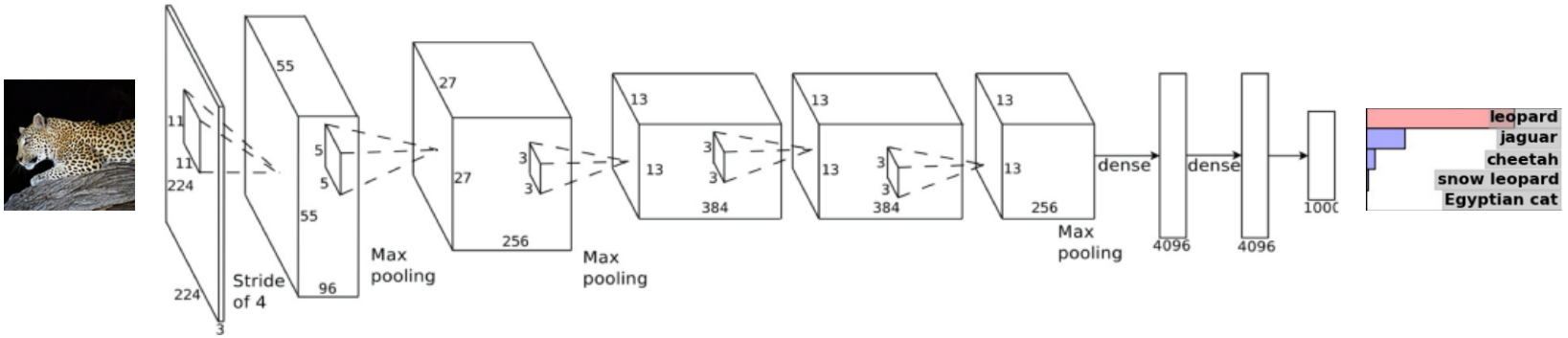




http://image-net.org/explore

AlexNet

- AlexNet (Krizhevsky, Sutskever, Hinton, 2012)
  - 5 convolutional layers + 2 fully connected layers
  - 650,000 neurons
  - 630,000,000 connections
  - 1.2 million labeled images for training
  - Trained with stochastic gradient descent on 2 NVIDIA GPUs for 1 week



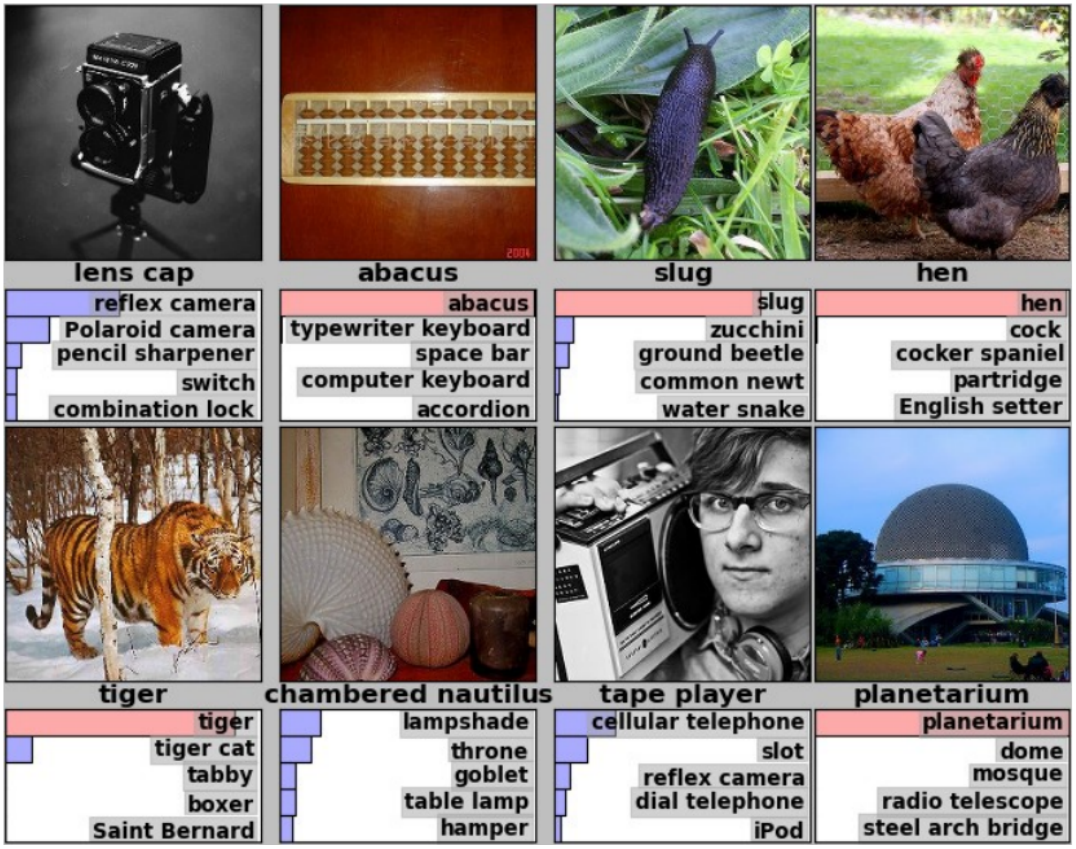
http://image-net.org/challenges/LSVRC/2012/supervision.pdf

AlexNet

- AlexNet structure

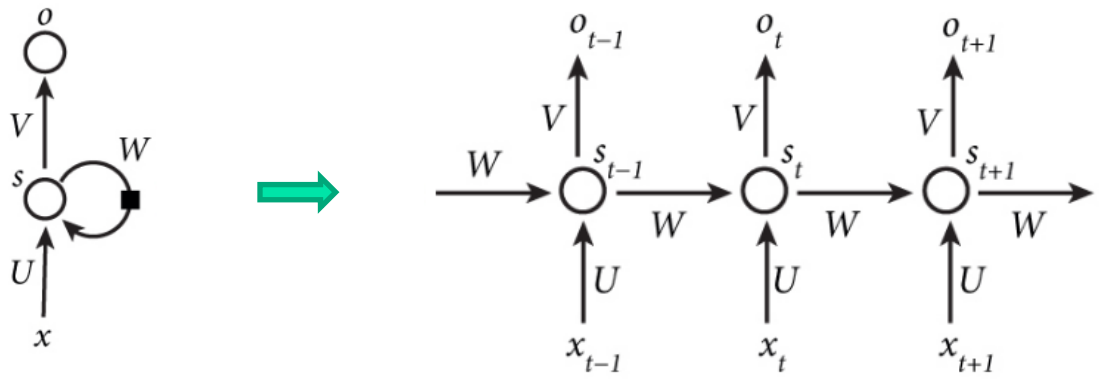
Layer	1	2	3	4	5	6	7	8
Type	c+m+n	c+m+n	c	c	c+m	full	full	full
Channels	96	256	384	384	256	4096	4096	1000
Filter size	11*11	5*5	3*3	3*3	3*3	-	-	-
Conv stride	4	1	1	1	1	-	-	-
Pooling size	3*3	3*3	-	-	3*3	-	-	-
Pooling stride	2	2	-	-	2	-	-	-
Padding size	2	1	1	1	1	-	-	-

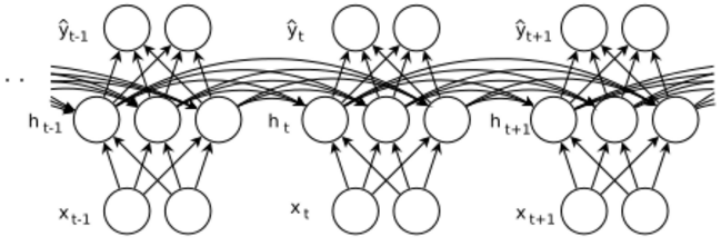
AlexNet


lens cap
reflex camera
Polaroid camera
pencil sharpener
switch
combination lock
abacus
abacus
typewriter keyboard
space bar
computer keyboard
accordion
slug
slug
zucchini
ground beetle
common newt
water snake
hen
hen
cock
cocker spaniel
partridge
English setter
tiger
tiger
tiger cat
tabby
boxer
Saint Bernard
chambered nautilus
lampshade
throne
goblet
table lamp
hamper
tape player
cellular telephone
slot
reflex camera
dial telephone
iPod
planetarium
planetarium
dome
mosque
radio telescope
steel arch bridge

RNN (Recurrent Neural Net)

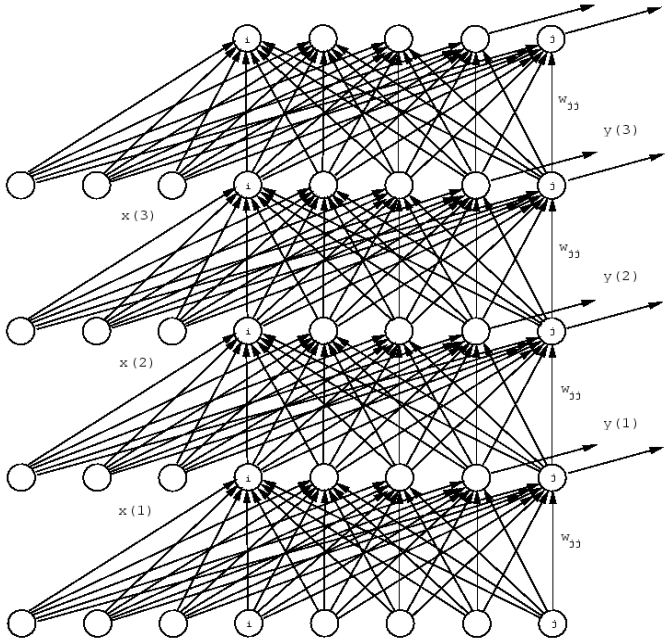
- Feedback loops exist to process sequential data
  - Ex> Sentence → sequence of words
  - Unfolding the network → multilayer neural networks





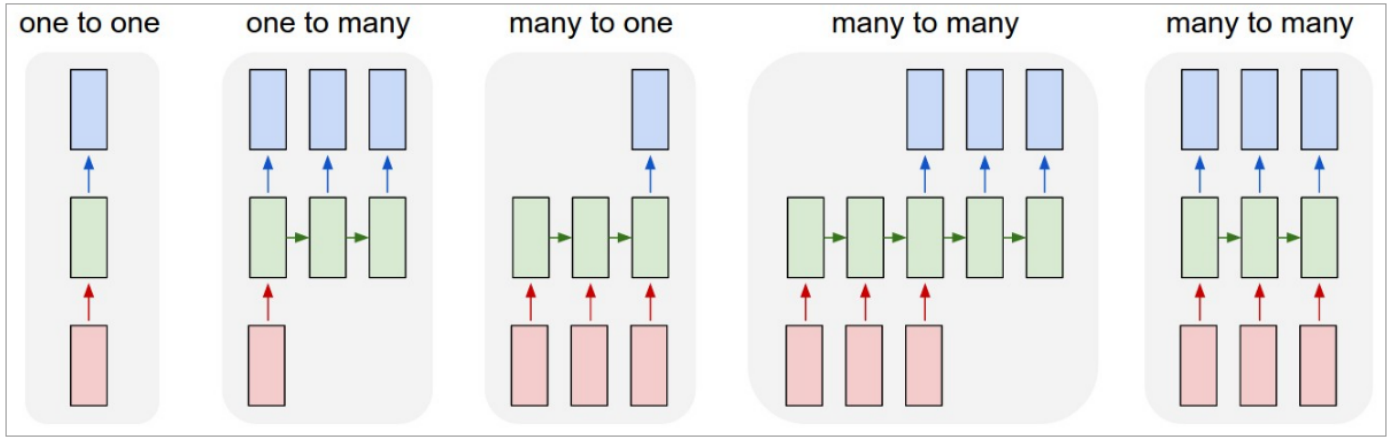
RNN (Recurrent Neural Net)

- Learning by backpropagation through time
  - The set of weights for each copy remains the same
  - The weight changes calculated for each network copy are summed



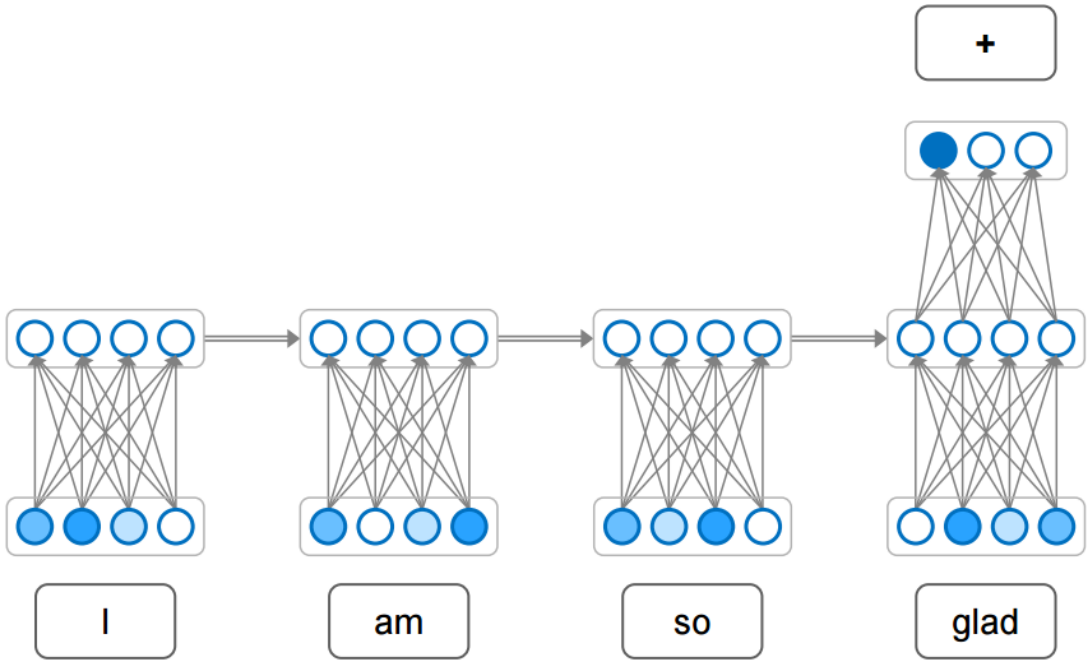
RNN (Recurrent Neural Net)

- Various types of input-out relations
  - The Fixed size (ex> image classification)
  - Sequence output (ex> image captioning)
  - Sequence input (ex> sentiment analysis)
  - Sequence input, sequence output (ex> machine translation)



RNN (Recurrent Neural Net)

- Example – sentiment analysis
  - Classify text according to sentiments (ex> positive or negative)

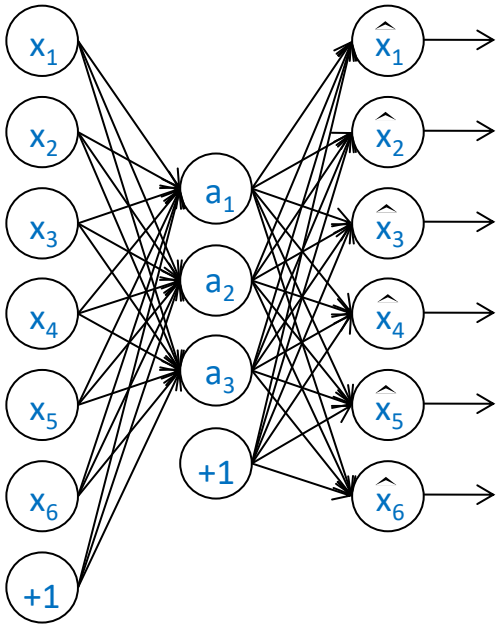


Autoencoder

- Unsupervised learning
  - Each layer is trained to learn good features automatically
  - Network is trained to

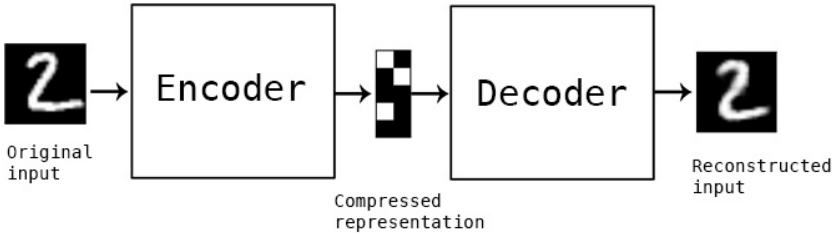
    output = input
  - Then the hidden layer represent

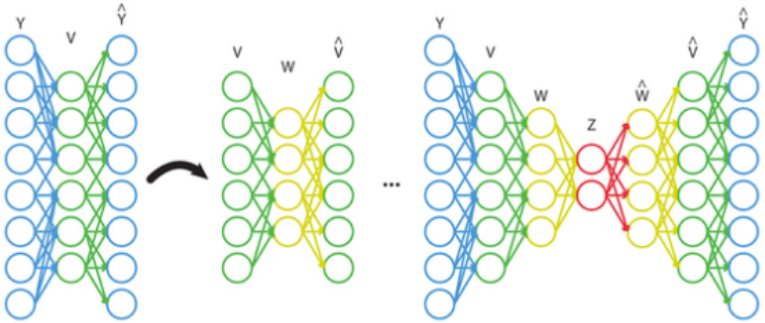
    features for the input



Autoencoder

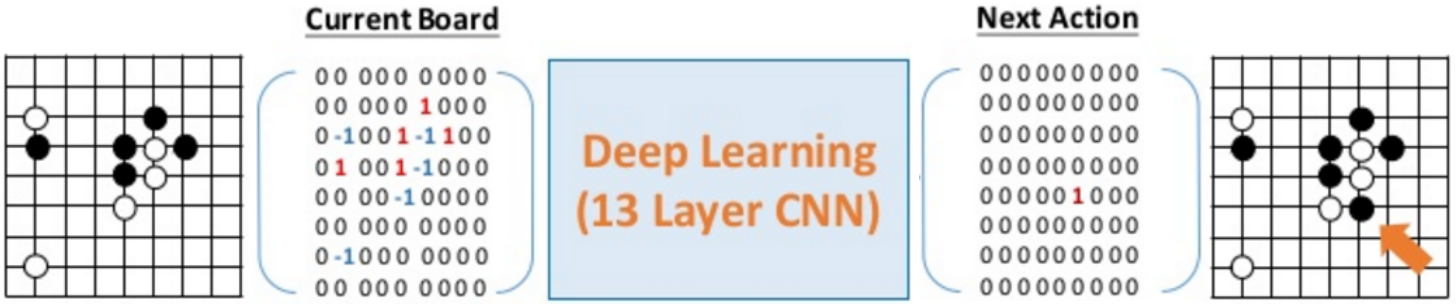
- It can be used for pre-training
  - Instead of random initialization of weights
  - Use stacked autoencoder to get initial weights





AlphaGo

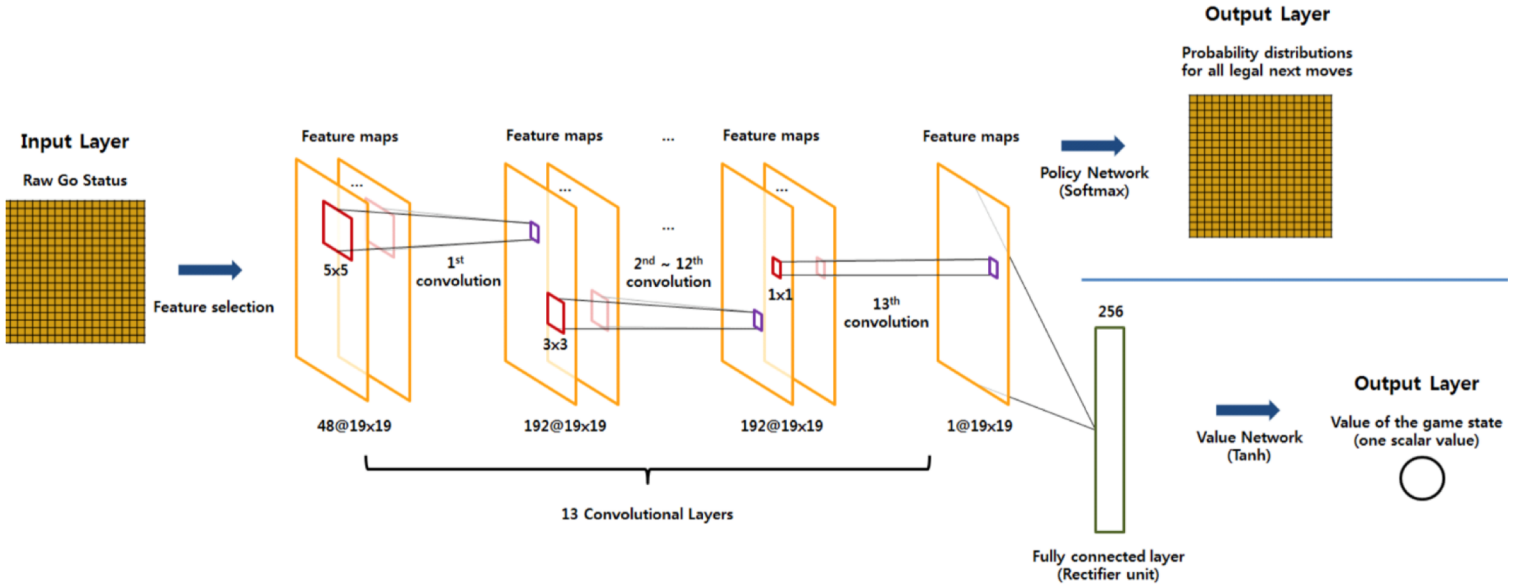
- AlphaGo
  - Google DeepMind 팀이 개발한 바둑 프로그램
  - 기존의 탐색 알고리즘 + 기계학습 사용
    - Monte Calro Tree Search, Supervised Learning, Reinforcement Learning
    - 3000만개 이상의 수를 이용하여 학습
  - GPU와 분산 처리 사용
    - 1202 CPU + 176 GPU



http://www.slideshare.net/ShaneSeungwhanMoon/how-alphago-works

AlphaGo

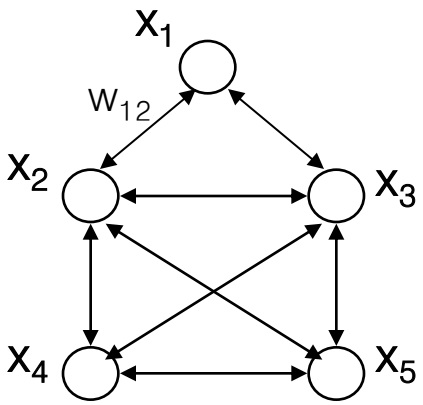
- CNN 구조
  - 입력층: 특정 대국에 대한 48가지 feature map
  - 은닉층: 13개의 컨볼루션 층
  - 결과층: Policy network – 착수 가능한 다음 수의 확률 분포(19 x 19)
    Value network – 현재 대국에서의 승산(한 개의 값)



http://sanghyukchun.github.io/97/

Feedback Networks and Memory

- Feedback network
  - Feedback connections exist → repeated computation
  - Output is the network states upon convergence



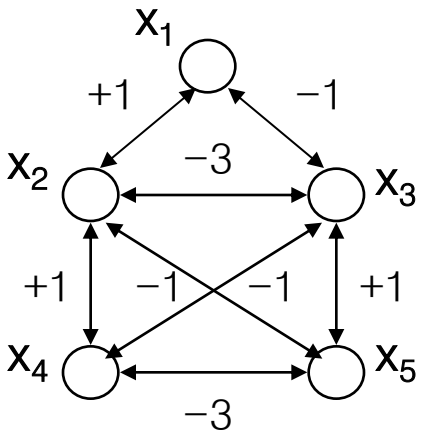
- $x_i(t+1) = 1$ if $\Sigma\, w_{ij}\, x_j > 0$ (x_j are the cells connected to x_i)
  -1 if $\Sigma\, w_{ij}\, x_j < 0$
  $x_i(t)$ if $\Sigma\, w_{ij}\, x_j = 0$

Feedback Networks and Memory

- Learning(memorizing)
  - $\Delta w_{ij} = x_i x_j$
  - $w_{ij} = \Sigma x_i x_j$ for all patterns(examples)
- Example
  - P1 = (1, 1, -1, 1, -1) (x_1=1, x_2=1, x_3=-1, …)
    P2 = (-1, -1, 1, -1, 1)
    P3 = (1, -1, 1, 1, -1)

$w_{12} = w_{21} = 1*1 + (-1)*(-1) + 1*(-1) = 1$
$w_{13} = w_{31} = 1*(-1) + (-1)*1 + 1*1 = -1$
$w_{23} = w_{32} = 1*(-1) + (-1)*1 + (-1)*1 = -3$
…

Feedback Networks and Memory

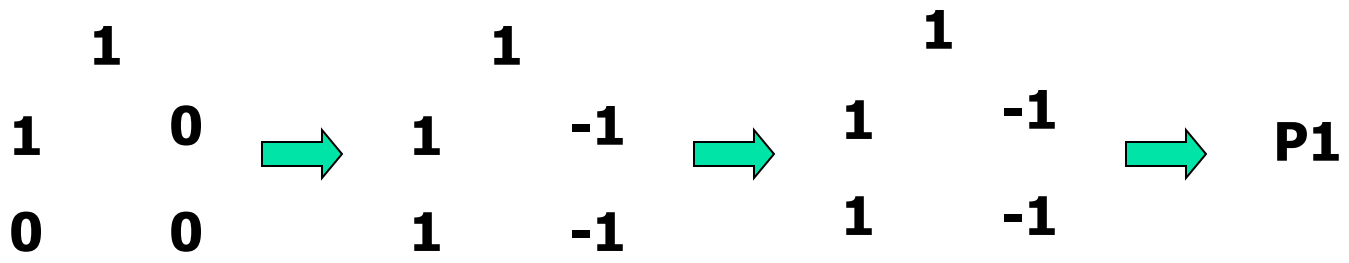


- Memorized patterns:

+		-		+	
+	-	-	+	-	+
+	-	-	+	+	-

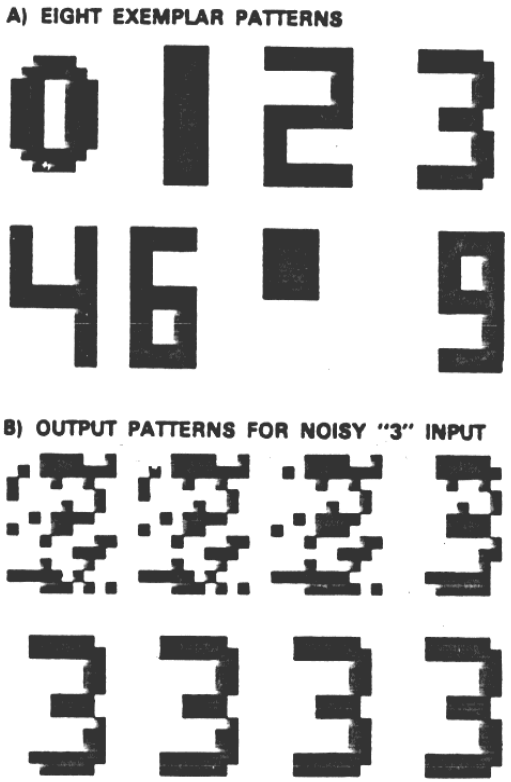
Feedback Networks and Memory

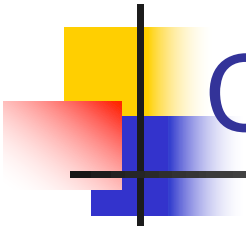
- Give P = (1, 1, 0, 0, 0)



- Characteristics
  - Data is represented as an activation pattern
    → **Distributed representation**
  - Even if one cell or connection fails, it can recall the pattern
    → **Robust**

Feedback Networks and Memory





Connectionist AI vs. Symbolic AI

- Symbolic AI
  - Knowledge: Set of symbols (predicate logic, etc.)
  - Search for solution: Logical inference
  - Learning: Generalization of descriptions
- Connectionist AI
  - Knowledge: Connection weights
  - Search for solution: Parallel computation
  - Learning: Adjusting connection weights