UNIVERSIDADE FEDERAL DO CEARÁ
Campus Russas

Deep Learning

Prof. Dsc. Nauber Gois

Redes Convolucionais

Apresentação

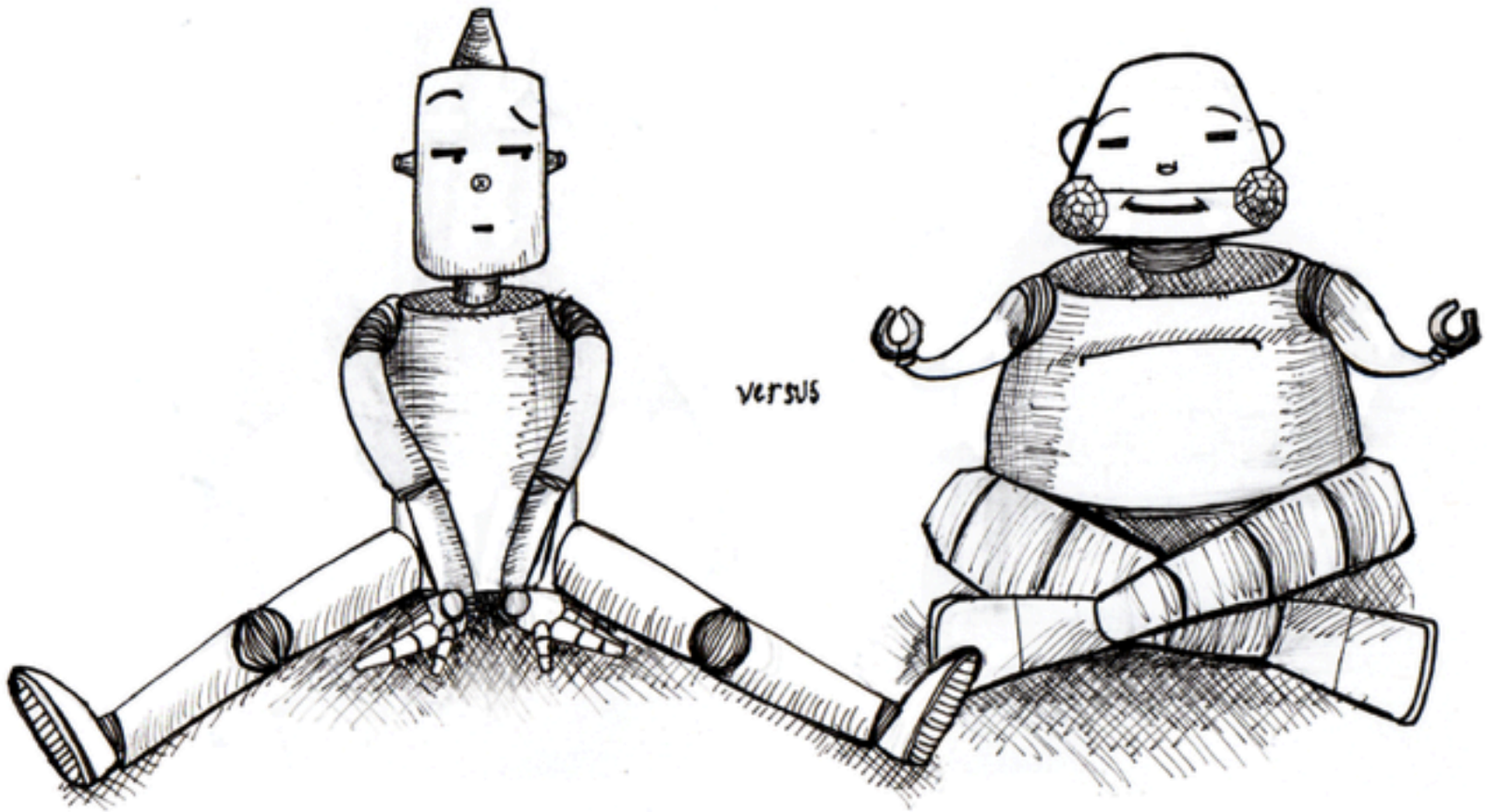
Francisco Nauber Bernardo Gois

Doutor em Informática Aplicada

Professor da Universidade Federal do Ceará

Dúvidas: naubergois@ufc.br

How can developments in deep learning make for a better approach to value investing?



MACHINE LEARNING

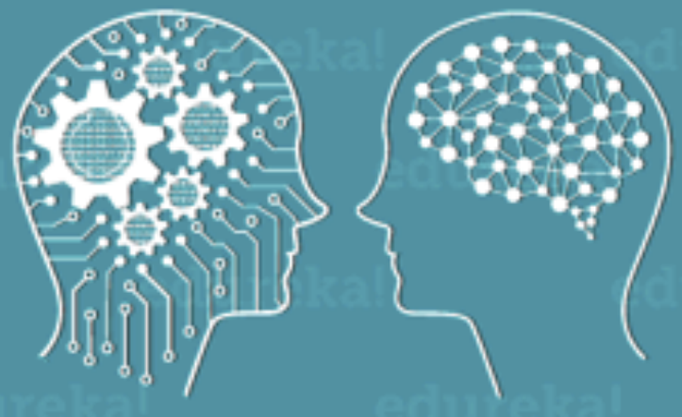
DEEP LEARNING

ARTIFICIAL INTELLIGENCE

Engineering of making Intelligent Machines and Programs

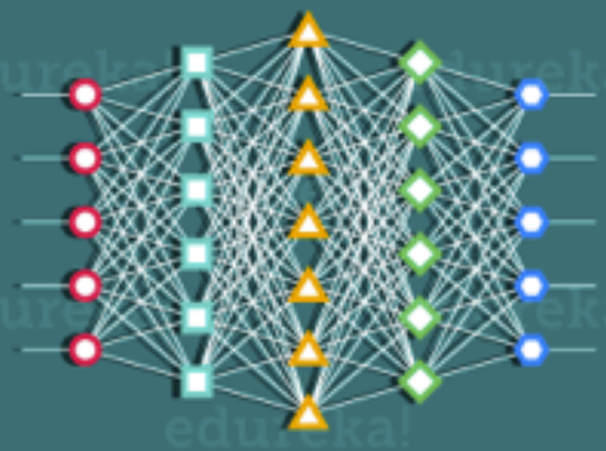
MACHINE LEARNING

Ability to learn without being explicitly programmed



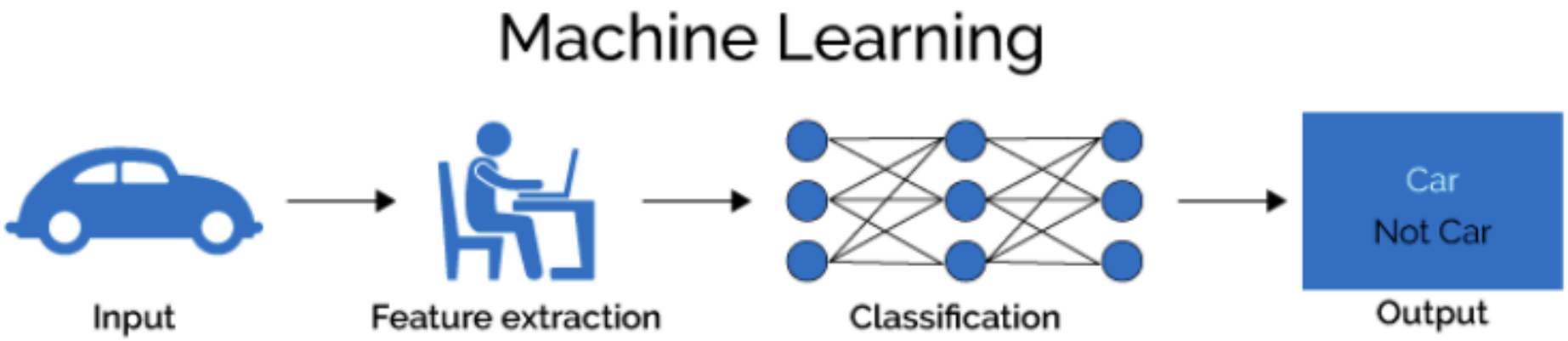
DEEP LEARNING

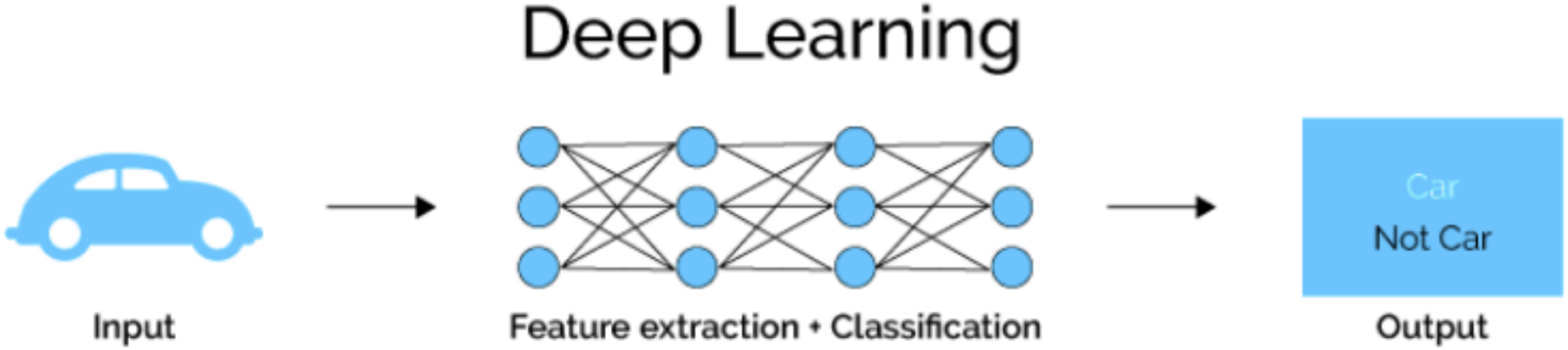
Learning based on Deep Neural Network



1950's 1960's 1970's 1980's 1990's 2000's 2006's 2010's 2012's 2017's

Deep Learning





Deep Learning

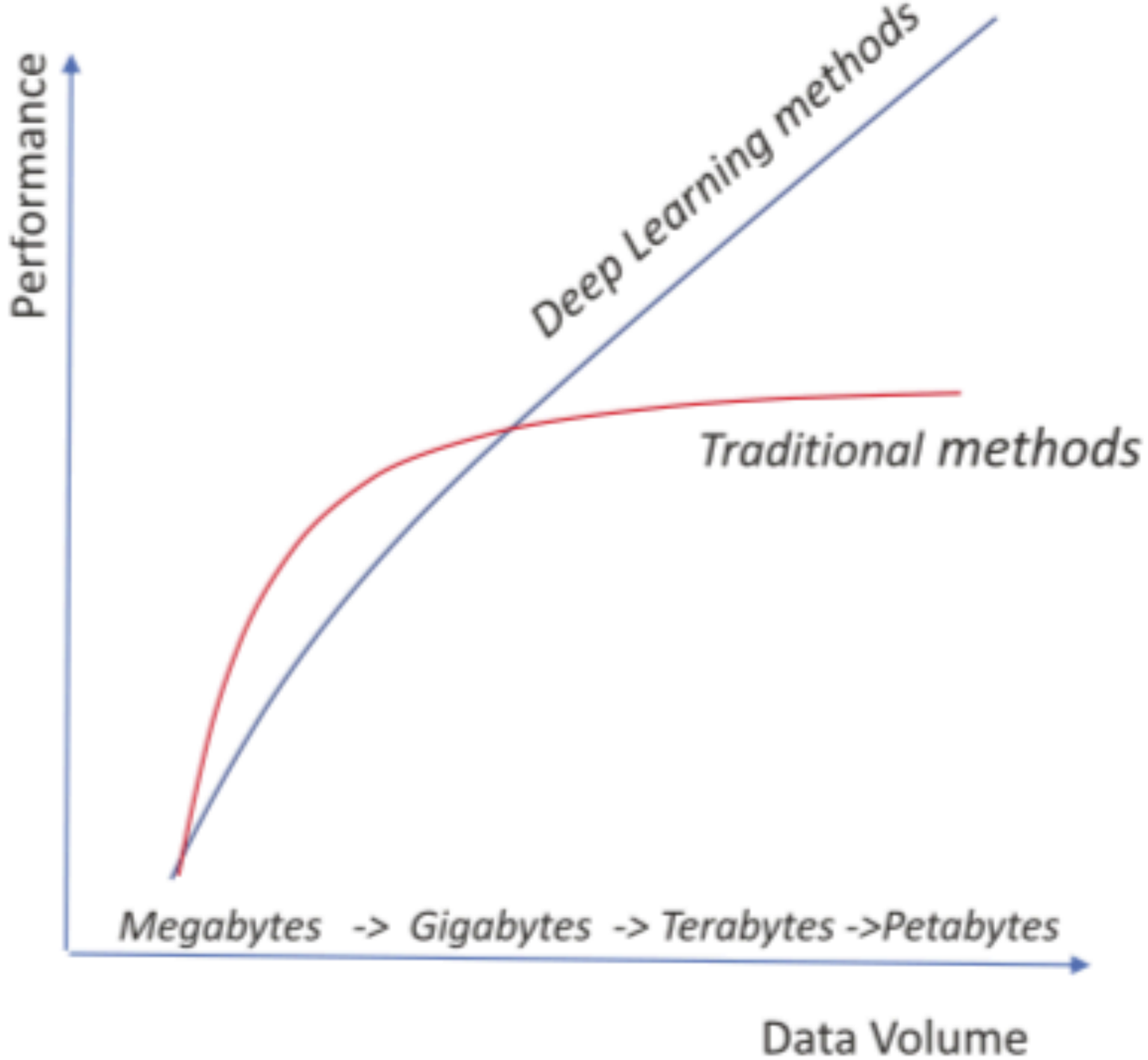


Figure 2-17. *Performance comparison of traditional methods versus deep-learning methods*

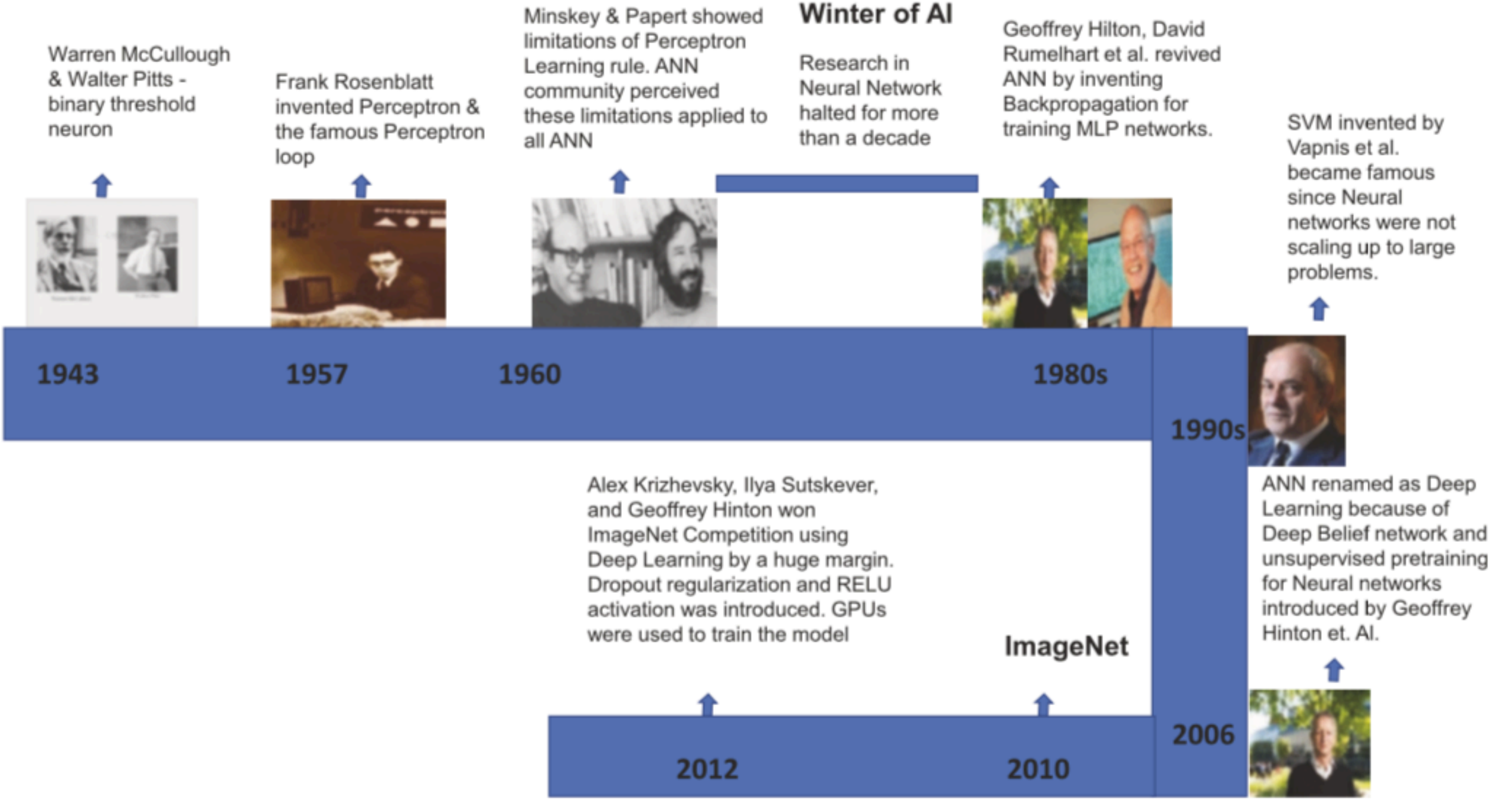
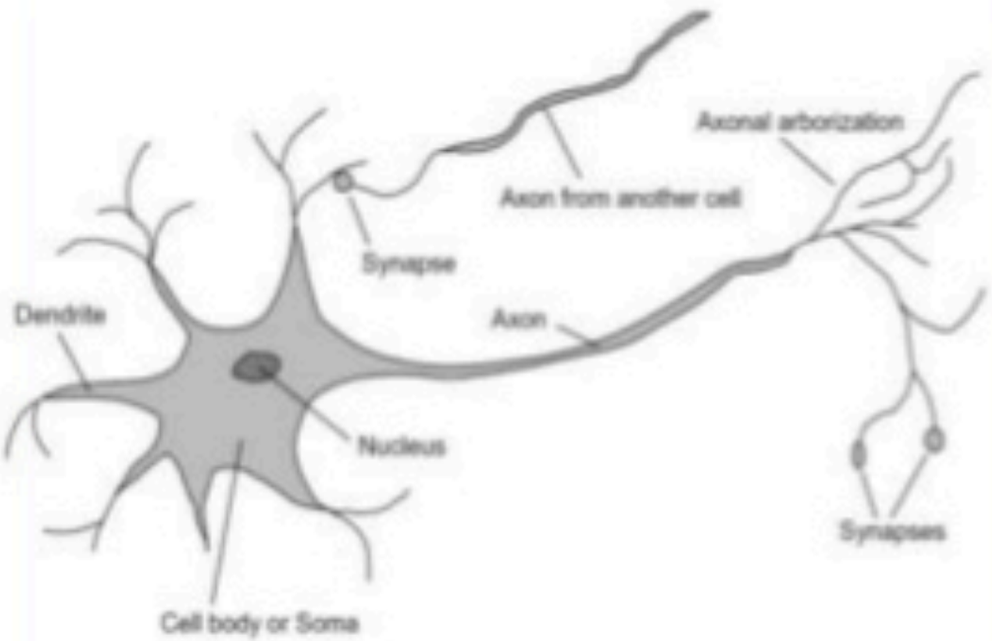
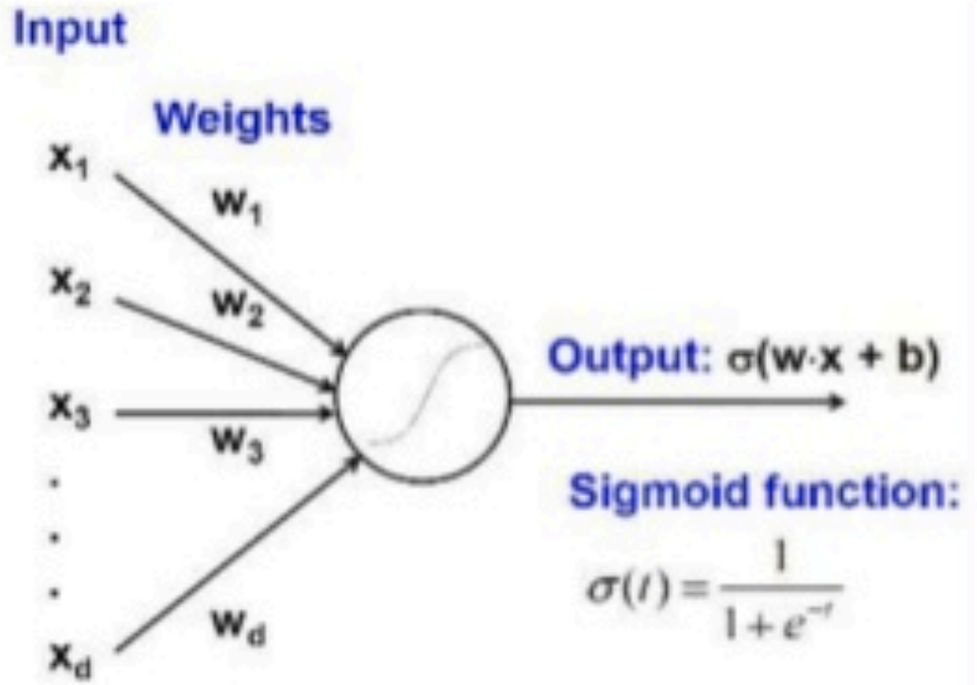


Figure 2-3. *Evolution of artificial neural networks*

Biological neuron and Perceptrons



A biological neuron



An artificial neuron (Perceptron)
- a linear classifier

Deep Learning

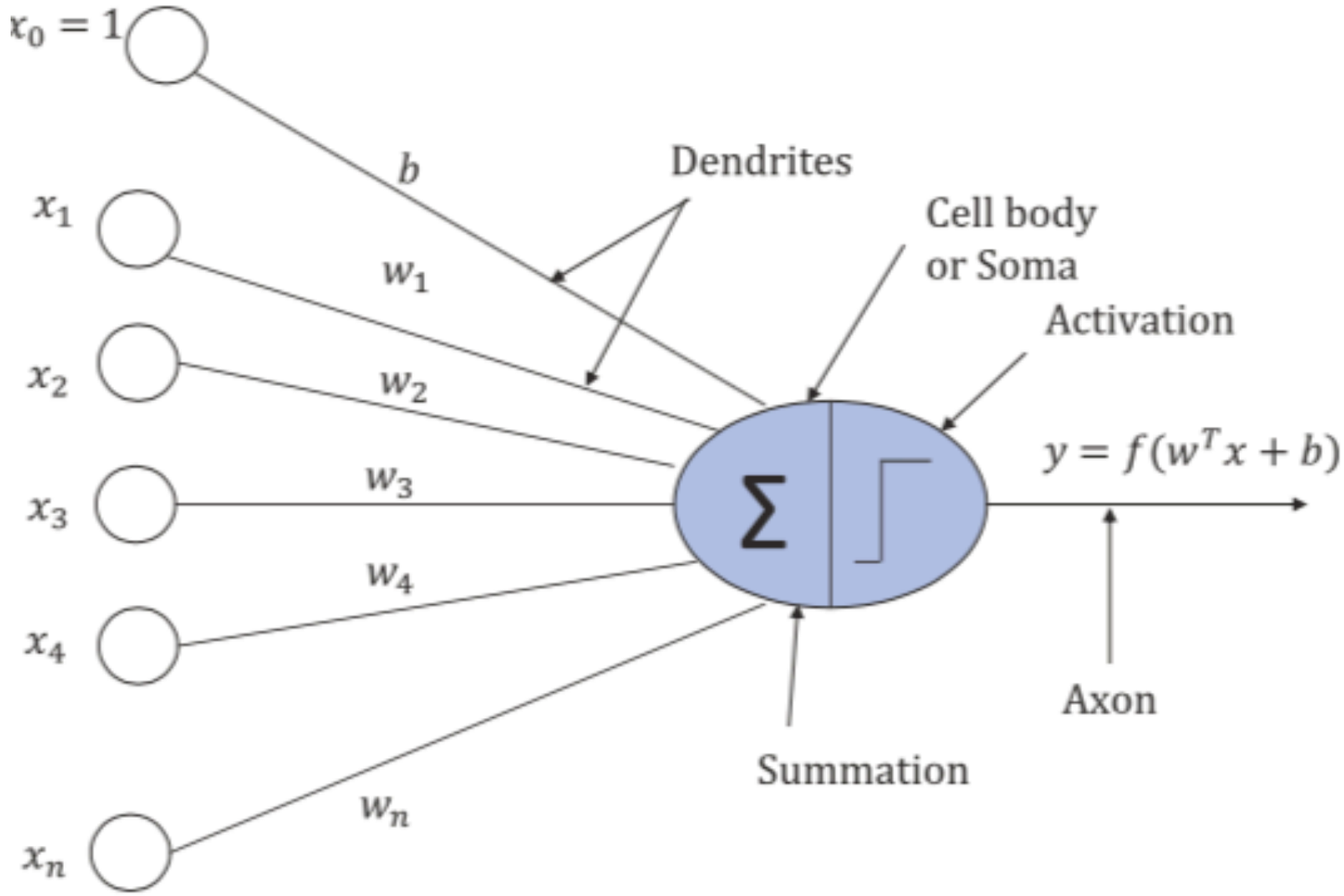
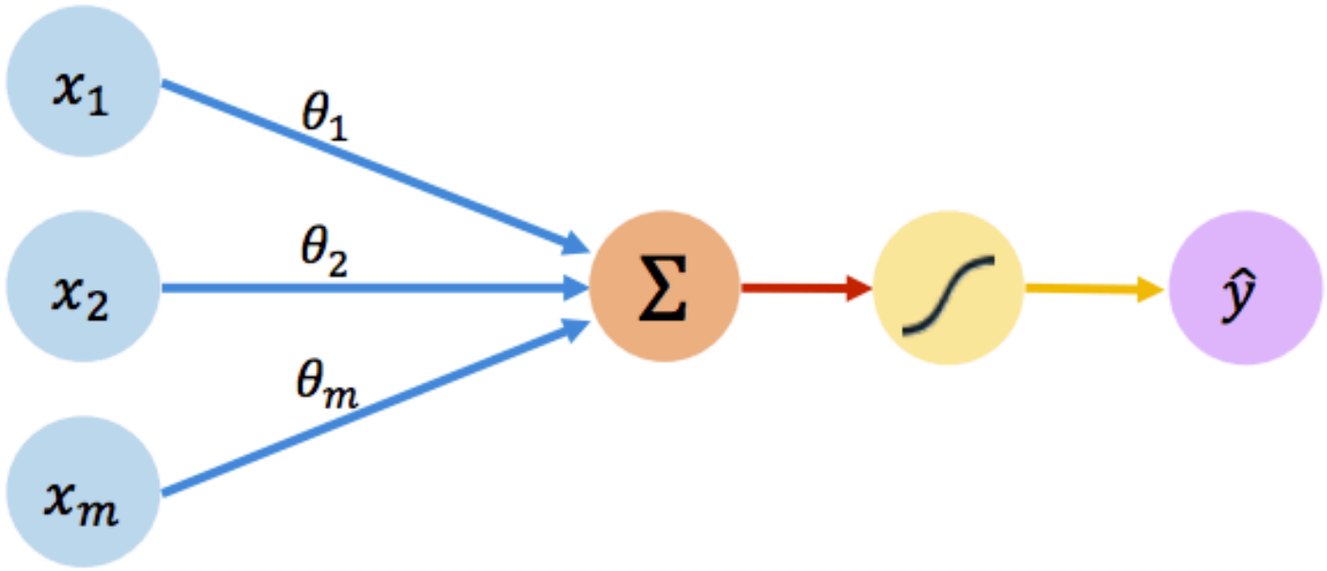


Figure 2-2. *Structure of an artificial neuron*

Dúvidas: naubergois@ufc.br

The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

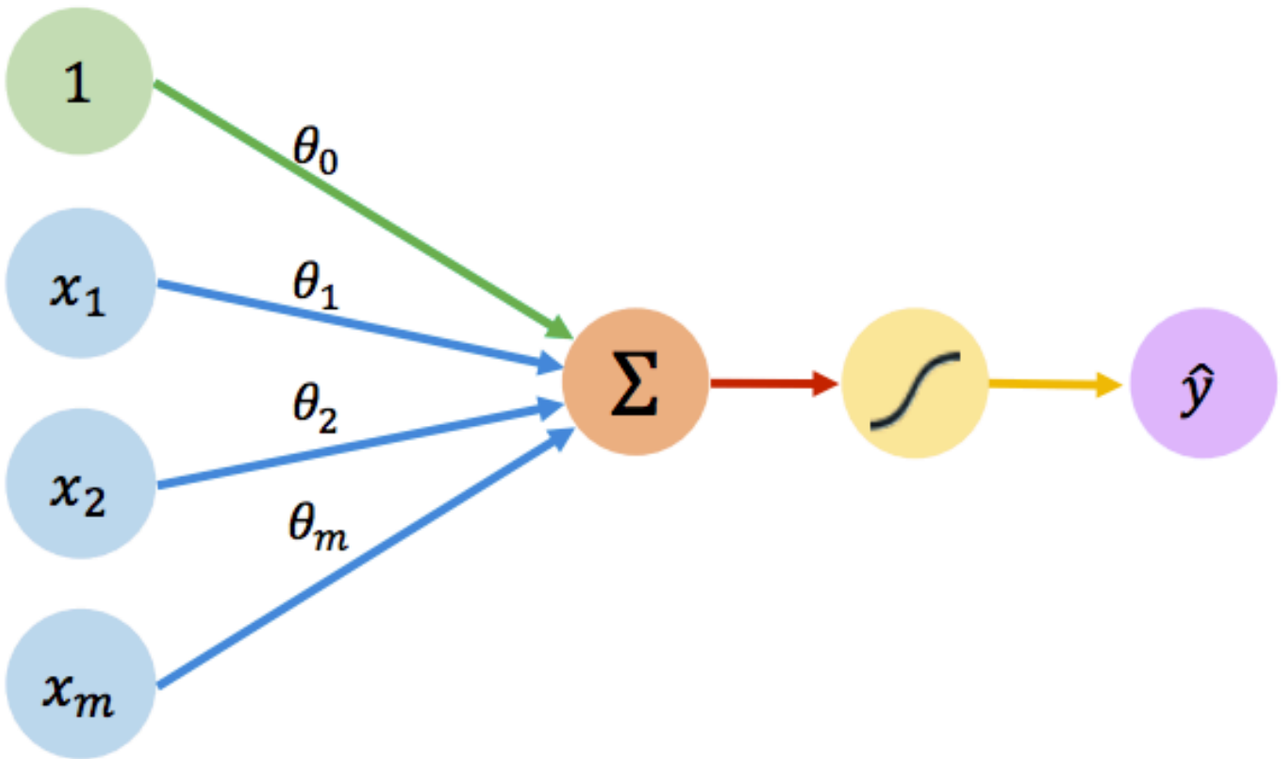
Output

Linear combination of inputs

$$\hat{y} = g\left(\sum_{i=1}^{m} x_i \, \theta_i\right)$$

Non-linear activation function

The Perceptron: Forward Propagation



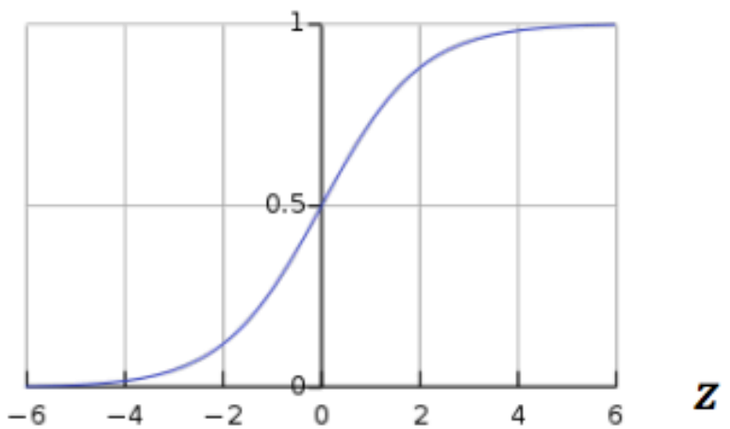
Inputs Weights Sum Non-Linearity Output

Activation Functions

$$\hat{y} = g(\theta_0 + \boldsymbol{X}^T \boldsymbol{\theta})$$

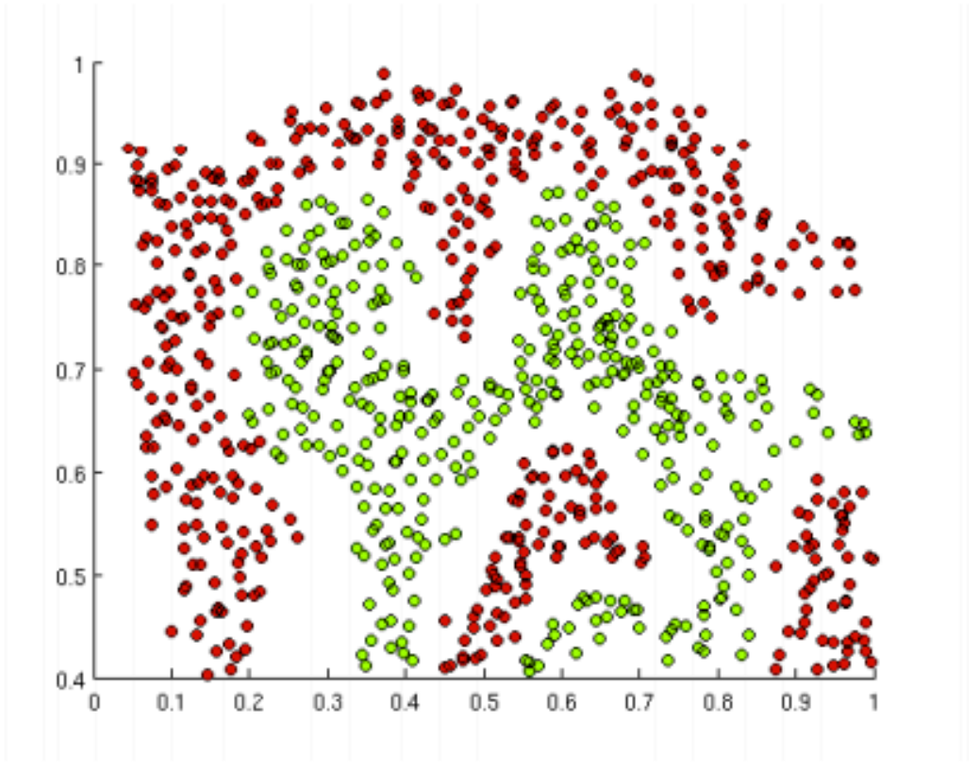
- Example: sigmoid function

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



Importance of Activation Functions

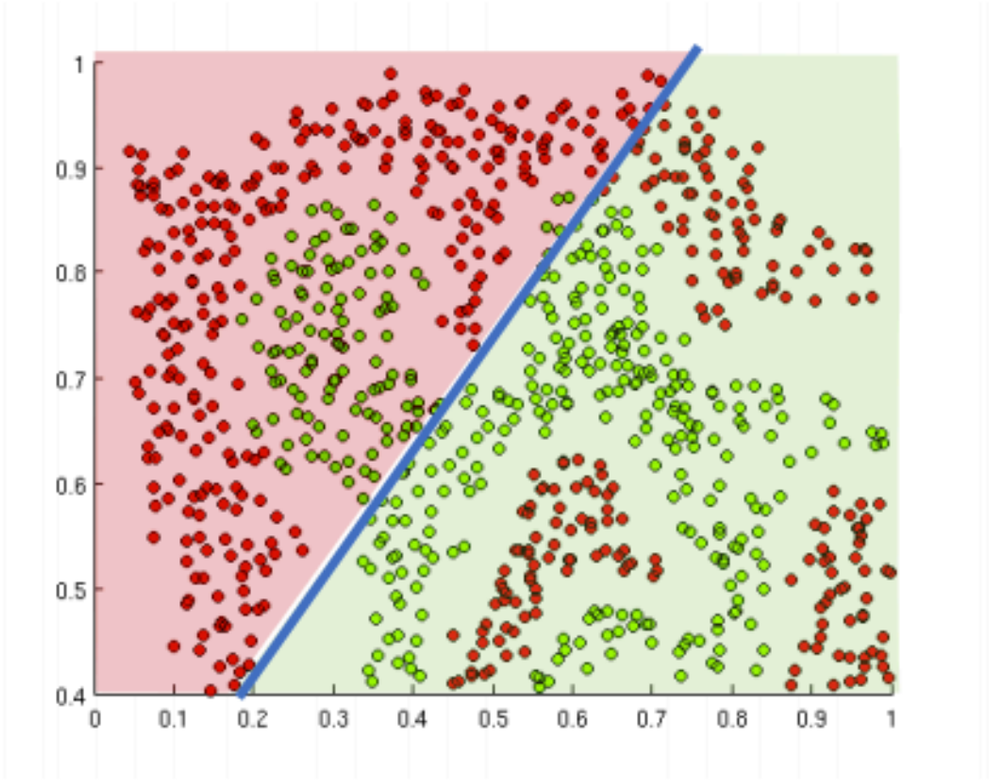
The purpose of activation functions is to ***introduce non-linearities*** *into the network*



What if we wanted to build a Neural Network to distinguish green vs red points?

Importance of Activation Functions

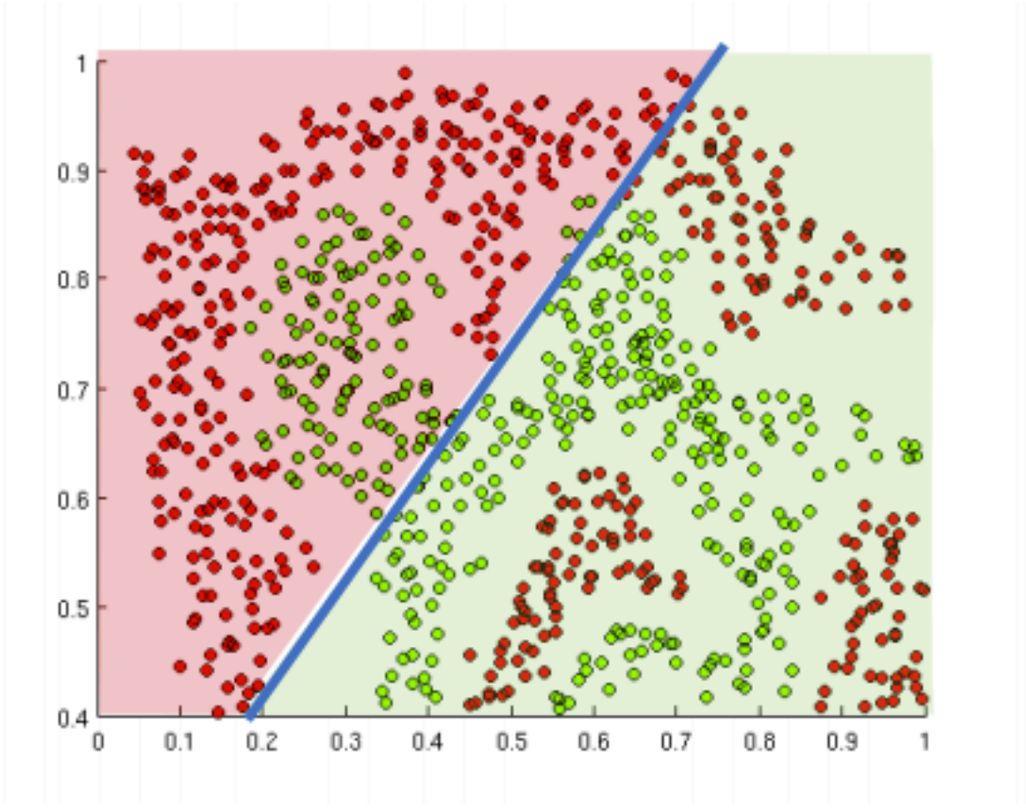
The purpose of activation functions is to ***introduce non-linearities*** *into the network*



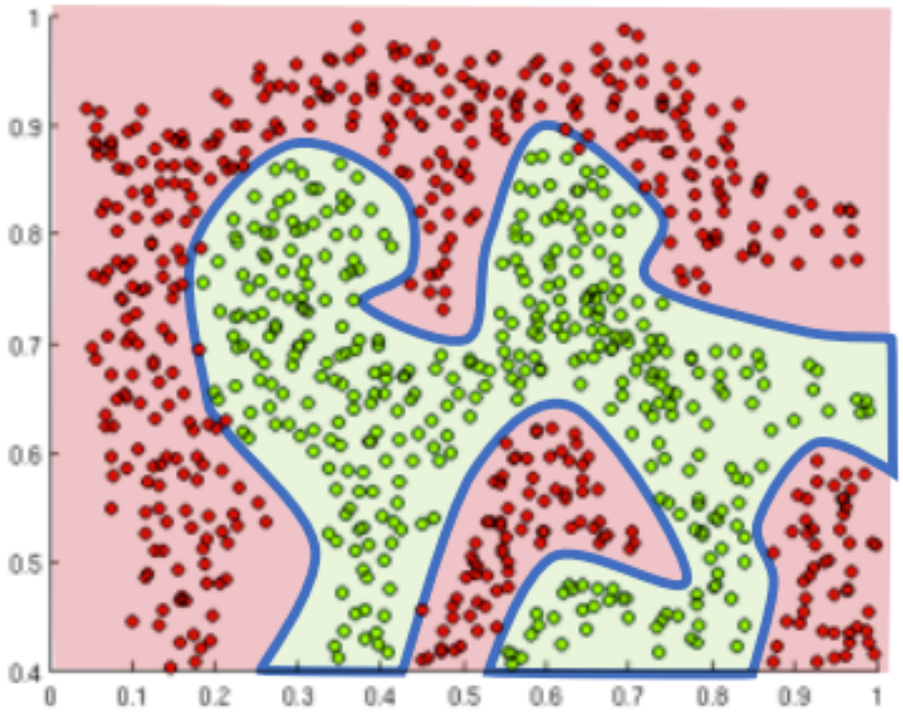
Linear Activation functions produce linear decisions no matter the network size

Importance of Activation Functions

The purpose of activation functions is to ***introduce non-linearities*** *into the network*

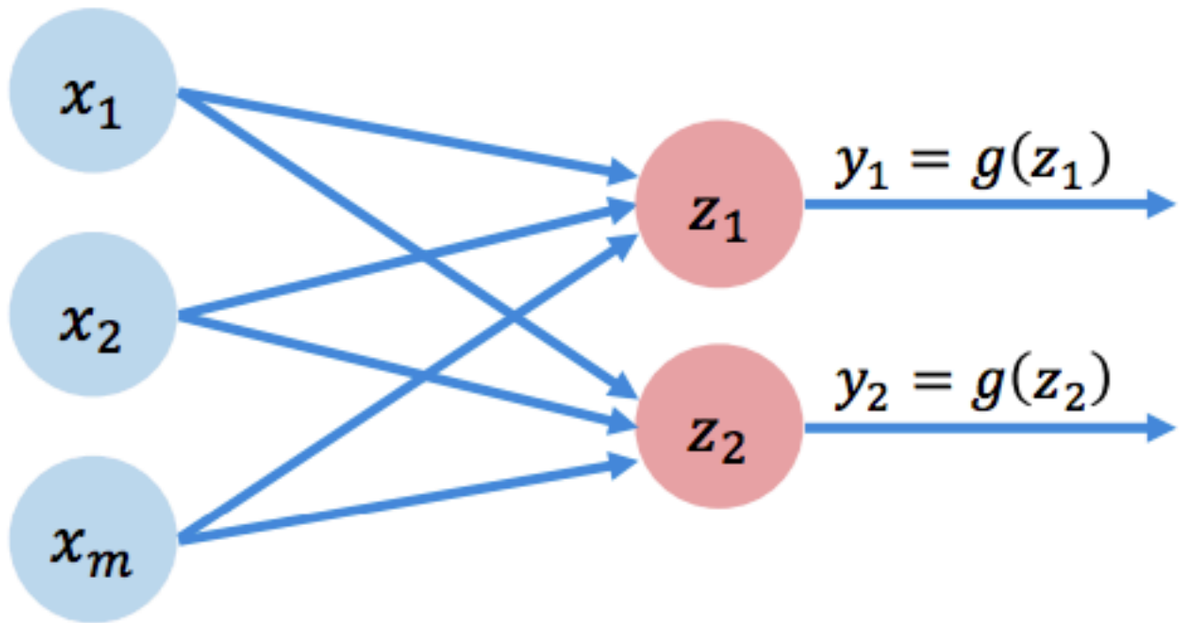


Linear Activation functions produce linear decisions no matter the network size



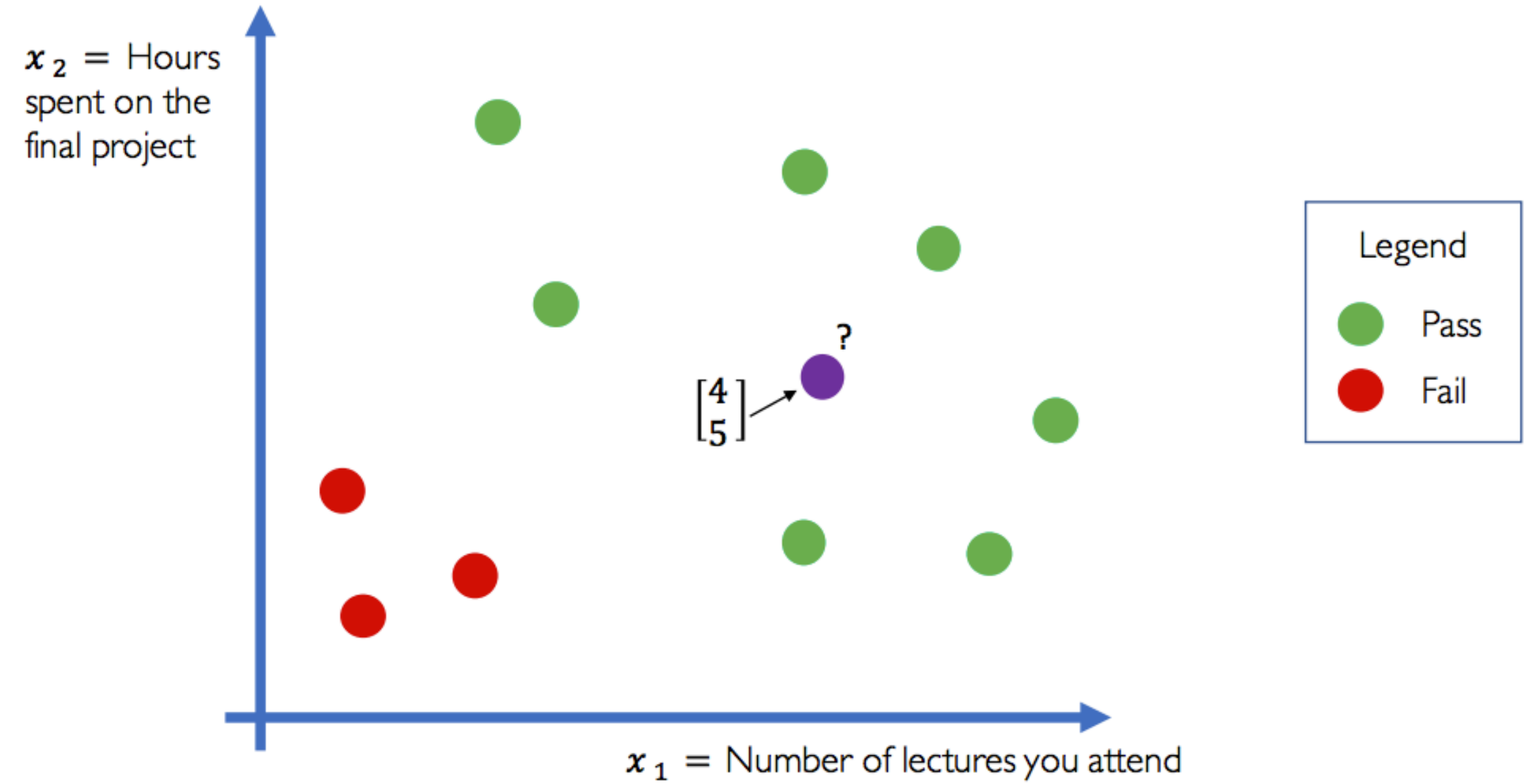
Non-linearities allow us to approximate arbitrarily complex functions

Multi Output Perceptron

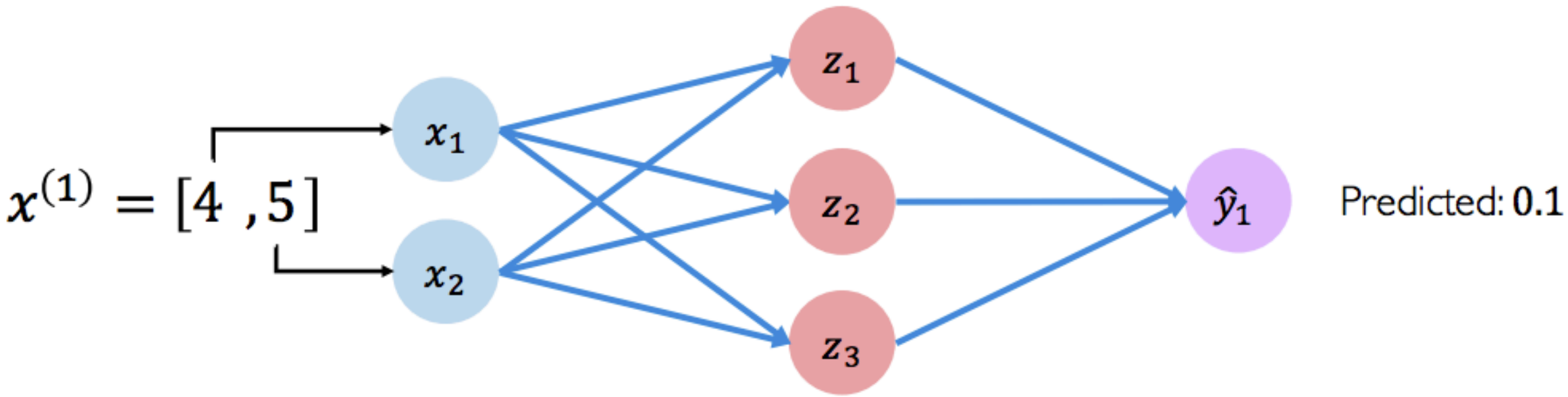


$$z_i = \theta_{0,i} + \sum_{j=1}^{m} x_j \, \theta_{j,i}$$

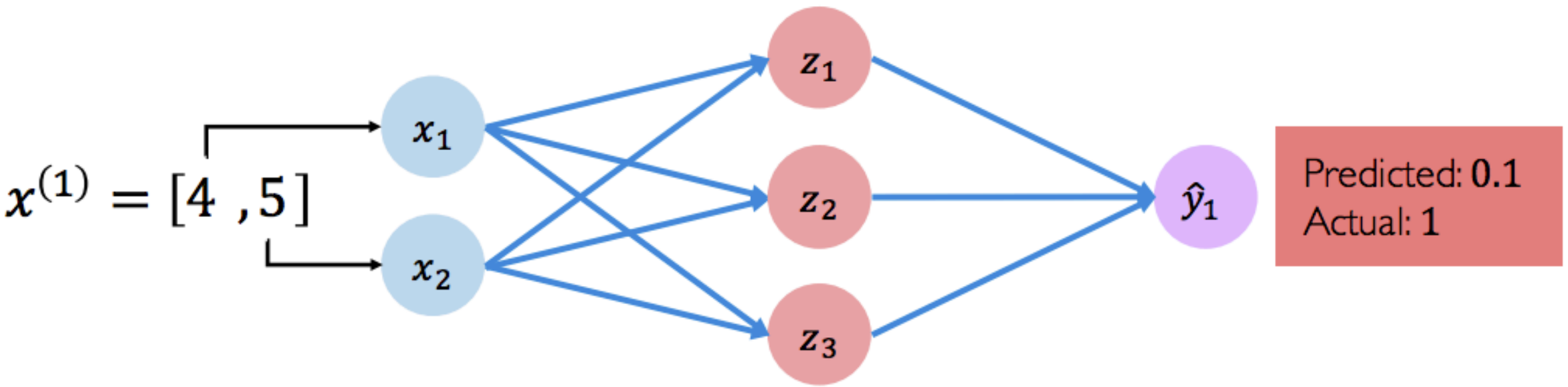
Example Problem: Will I pass this class?



Example Problem: Will I pass this class?

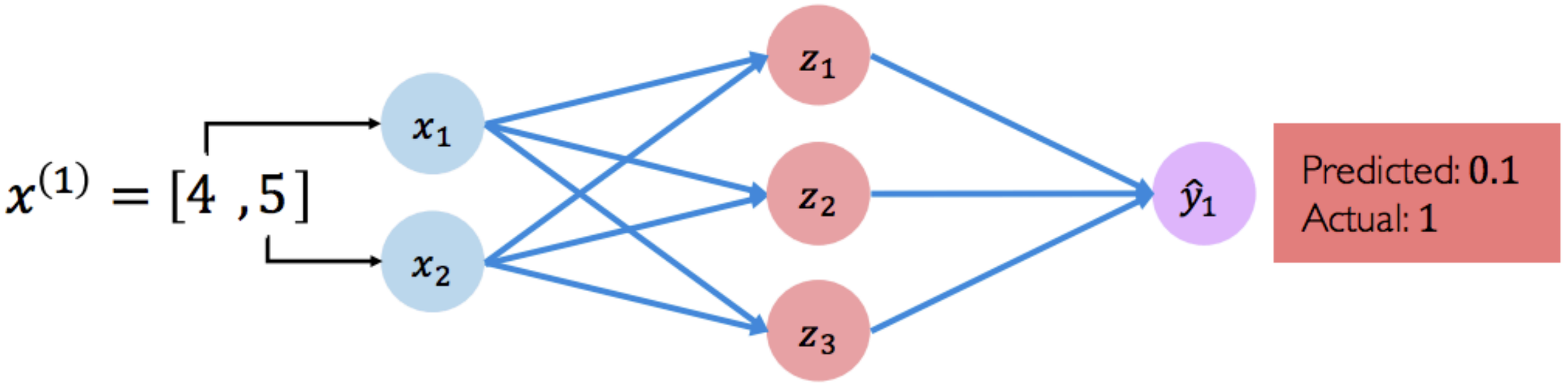


Example Problem: Will I pass this class?



Quantifying Loss

*The **loss** of our network measures the cost incurred from incorrect predictions*

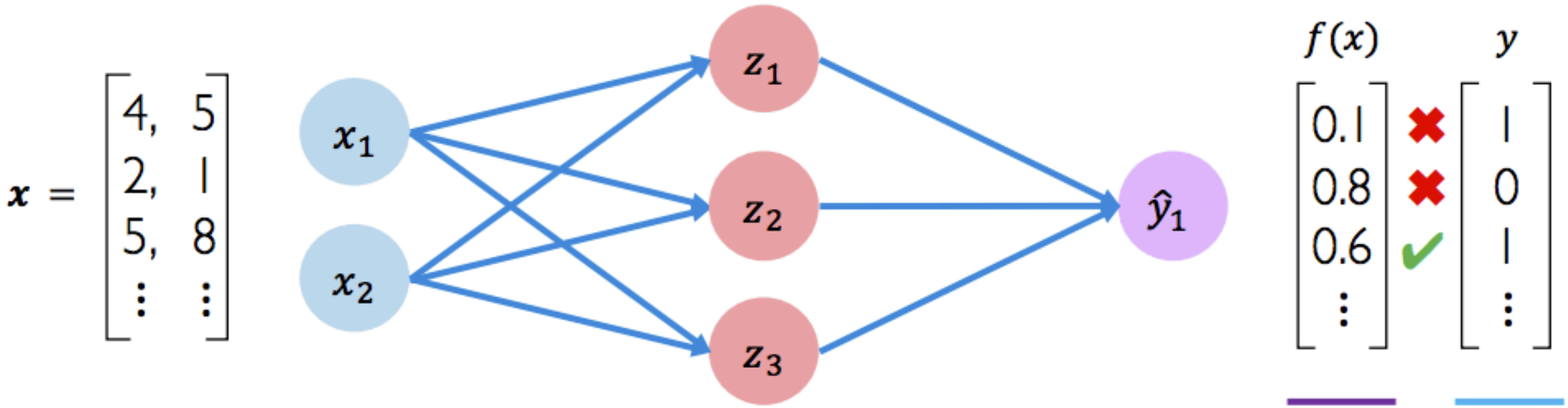


$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right)$$

Predicted Actual

Empirical Loss

*The **empirical loss** measures the total loss over our entire dataset*



Also known as:

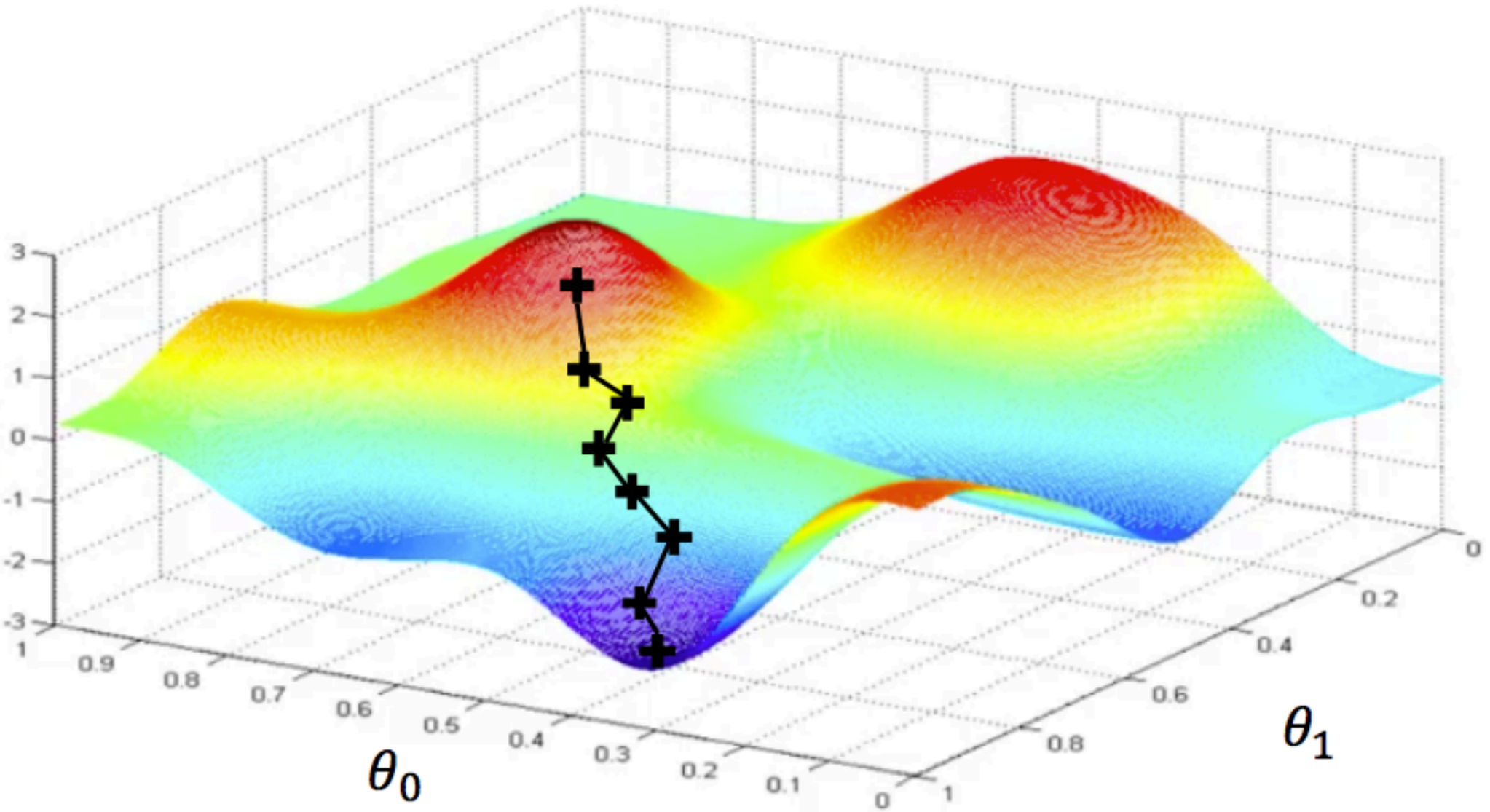
- Objective function
- Cost function
- Empirical Risk

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$$

Predicted Actual

Gradient Descent

Repeat until convergence

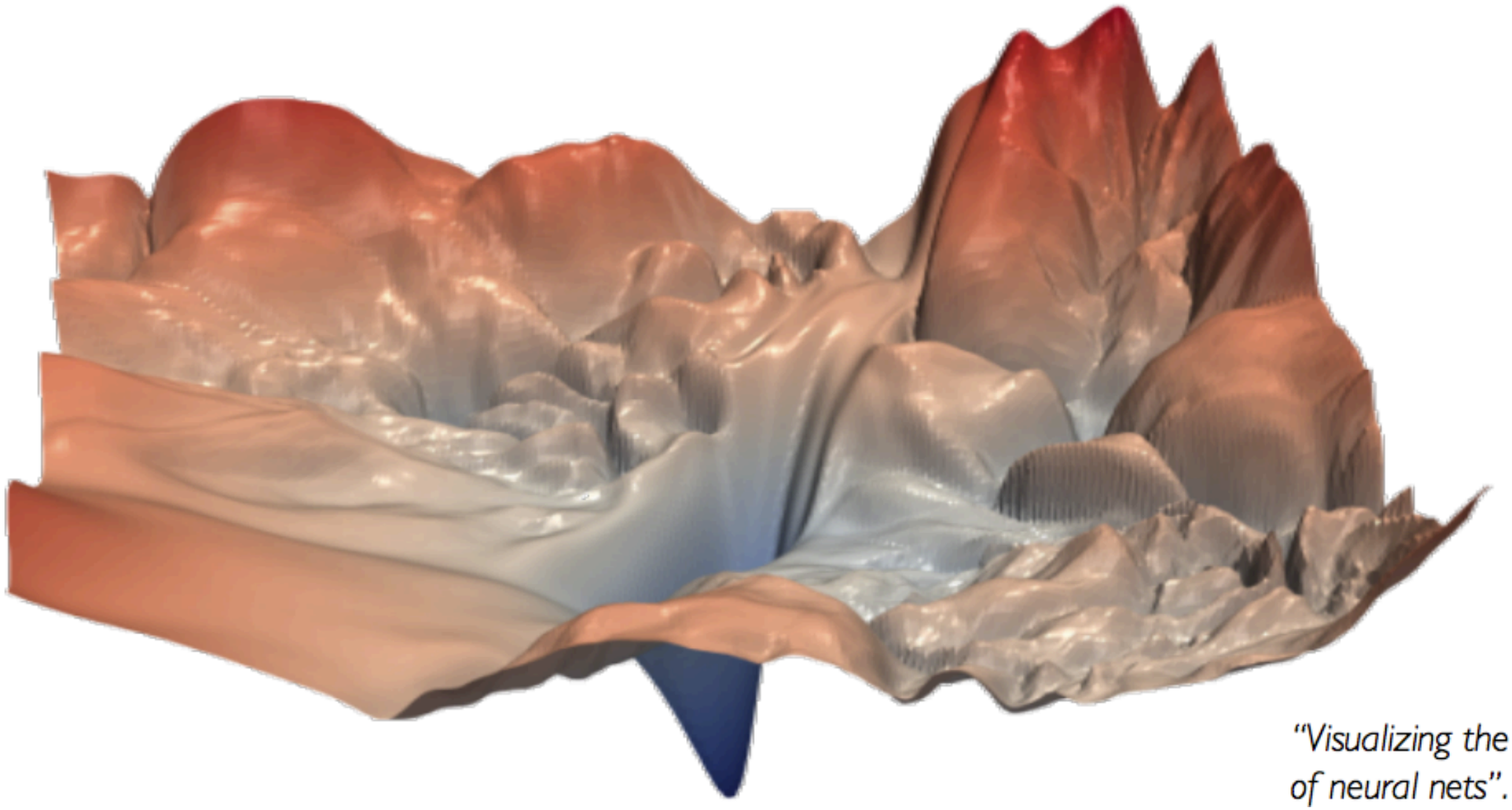


$J(\theta_0, \theta_1)$

θ_0

θ_1

Training Neural Networks is Difficult



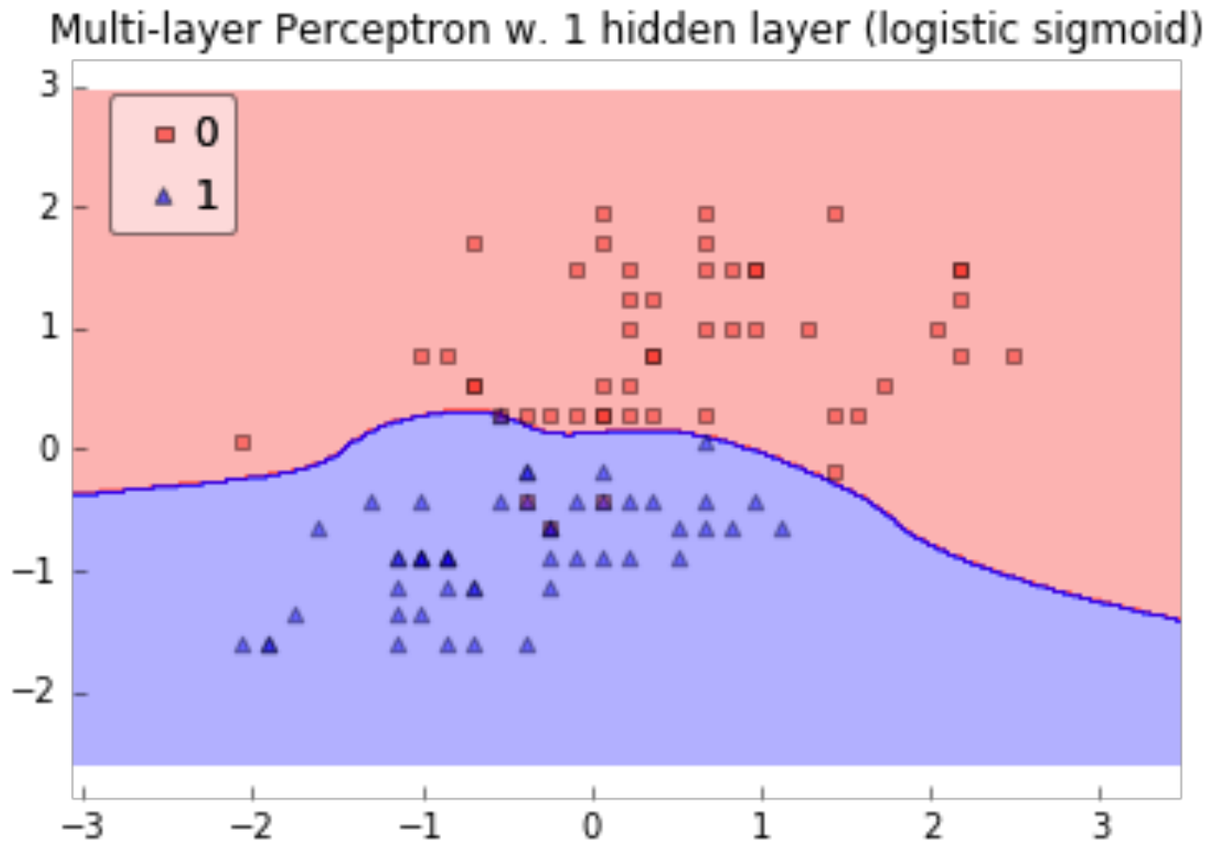
"Visualizing the loss landscape of neural nets". Dec 2017.

Perceptron

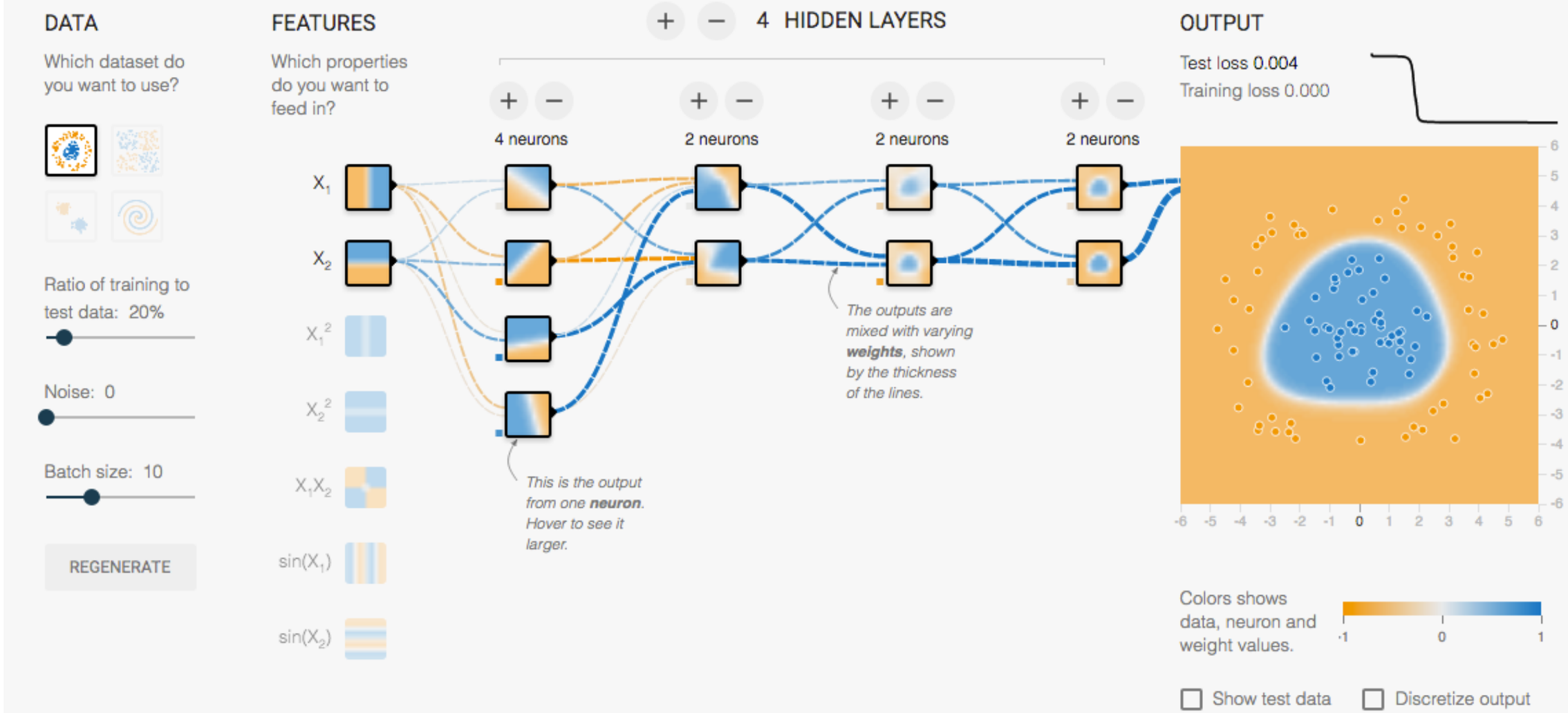
```
activation = sum(weight_i * x_i) + bias
```

```
prediction = 1.0 if activation >= 0.0 else 0.0
```

Função de Ativação



http://playground.tensorflow.org/



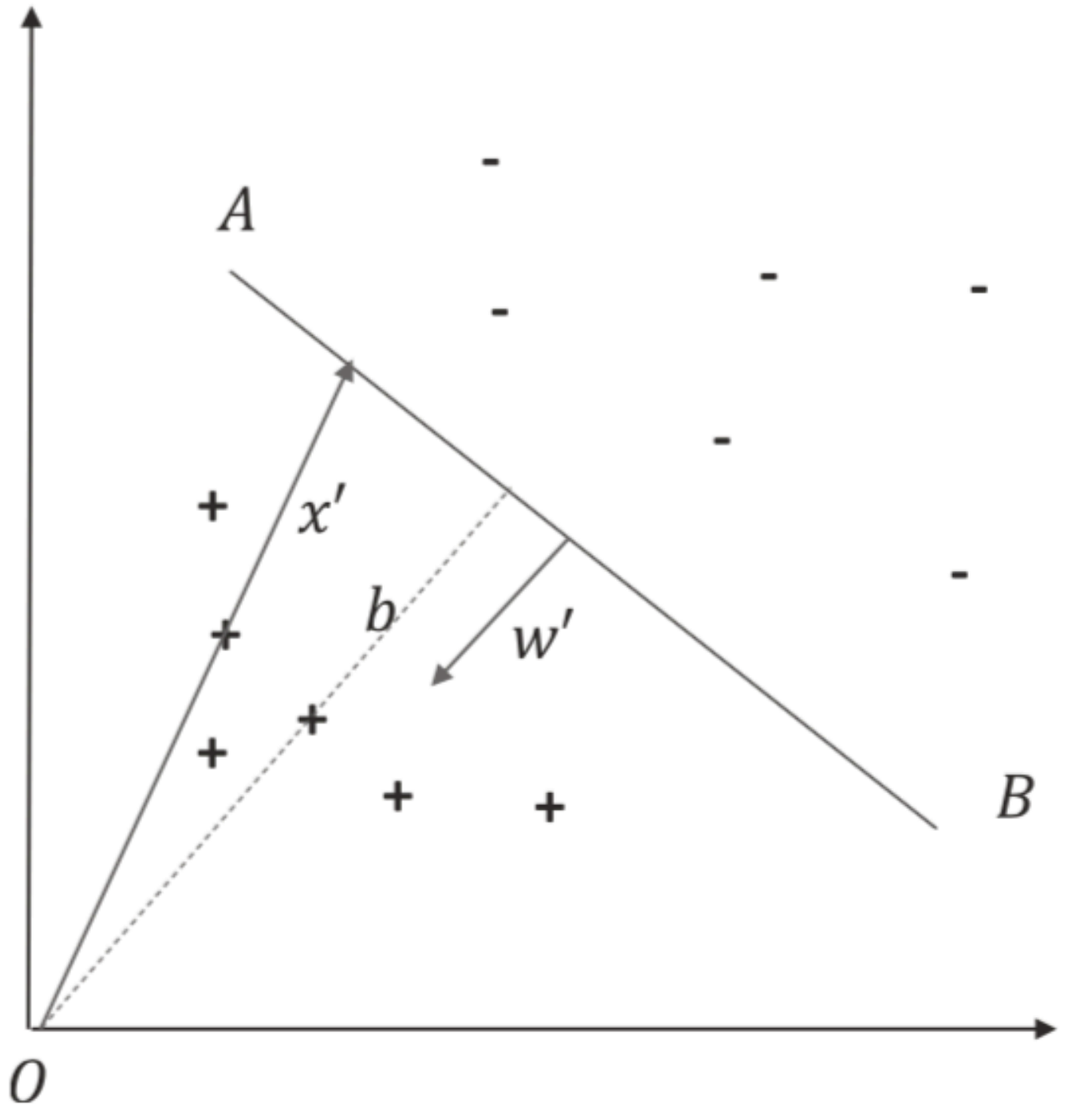
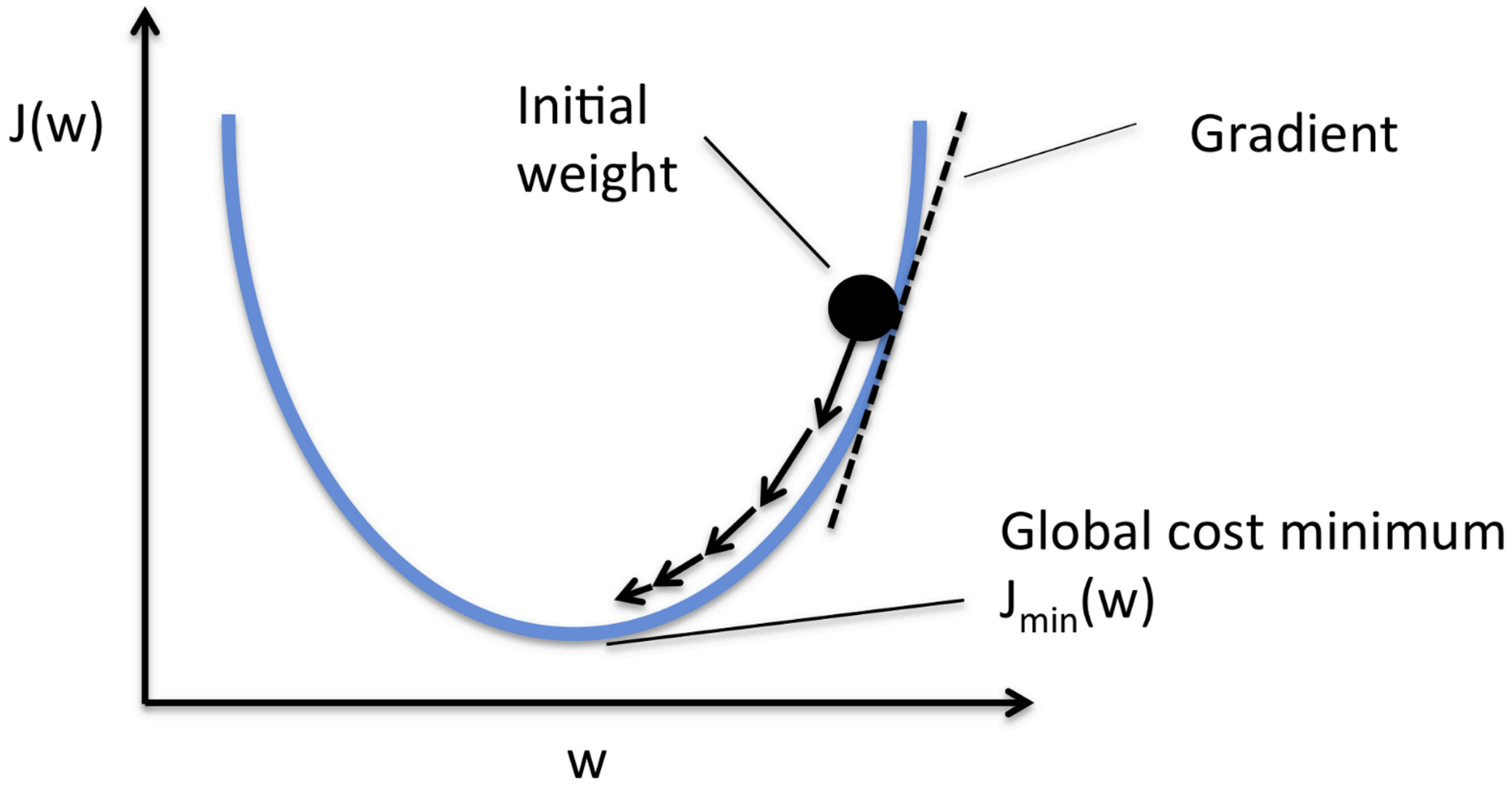


Figure 2-4. *Hyperplane separating two classes*

$$-w'^{T}x' = b \Rightarrow w'^{T}x' + b = 0$$

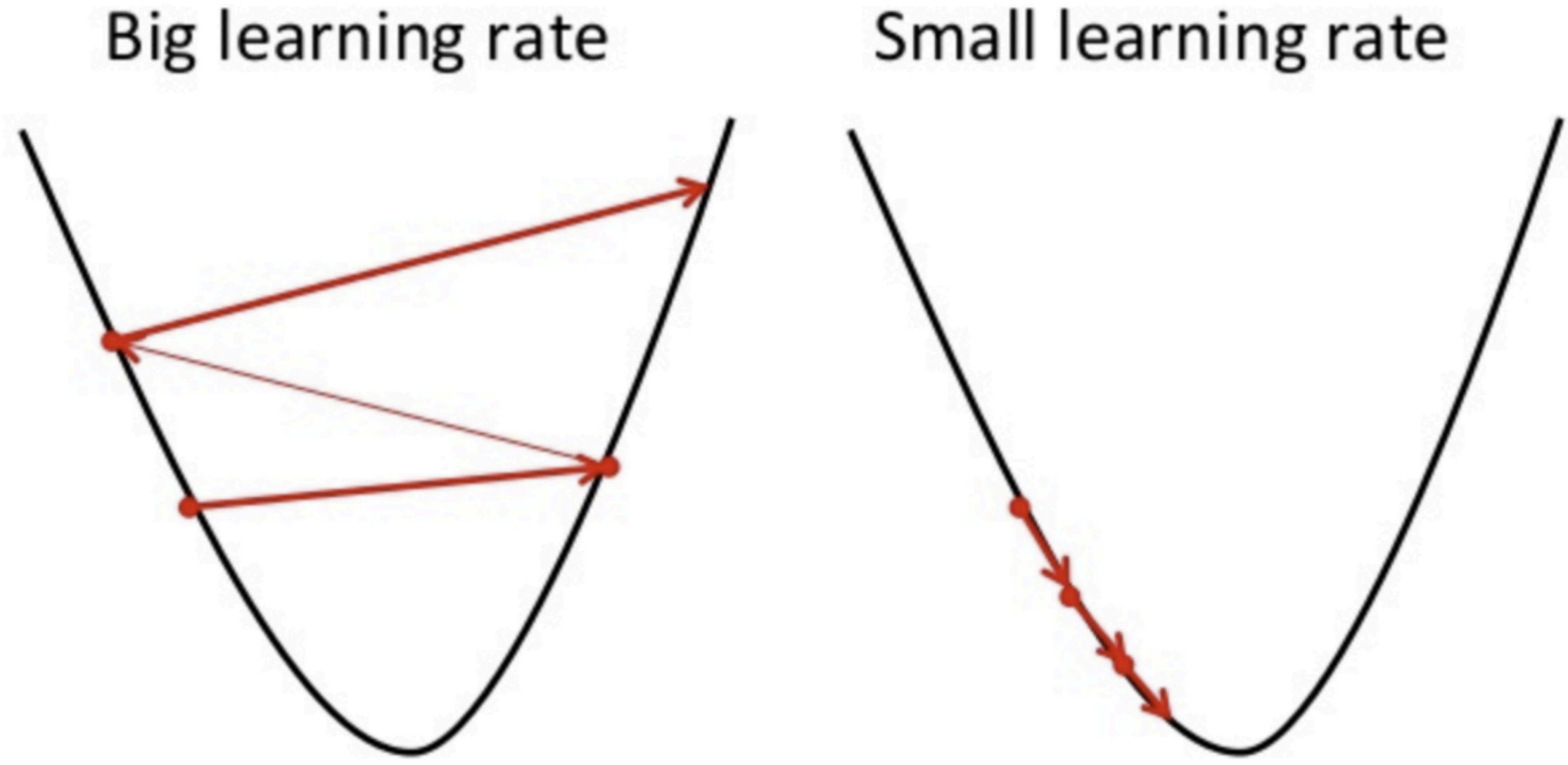

ALL MY FRIENDS ARE AT PARTY
BUT, I'M HERE LEARNING BACK PROPAGATION
memegenerator.net

Backpropagation



Backpropagation

```
w(t+1)= w(t) + learning_rate * (expected(t) - predicted(t)) * x(t)
```



SoftMax

$$C = \sum_{i=1}^{k} - y_i \log P(y_i = 1/x)$$

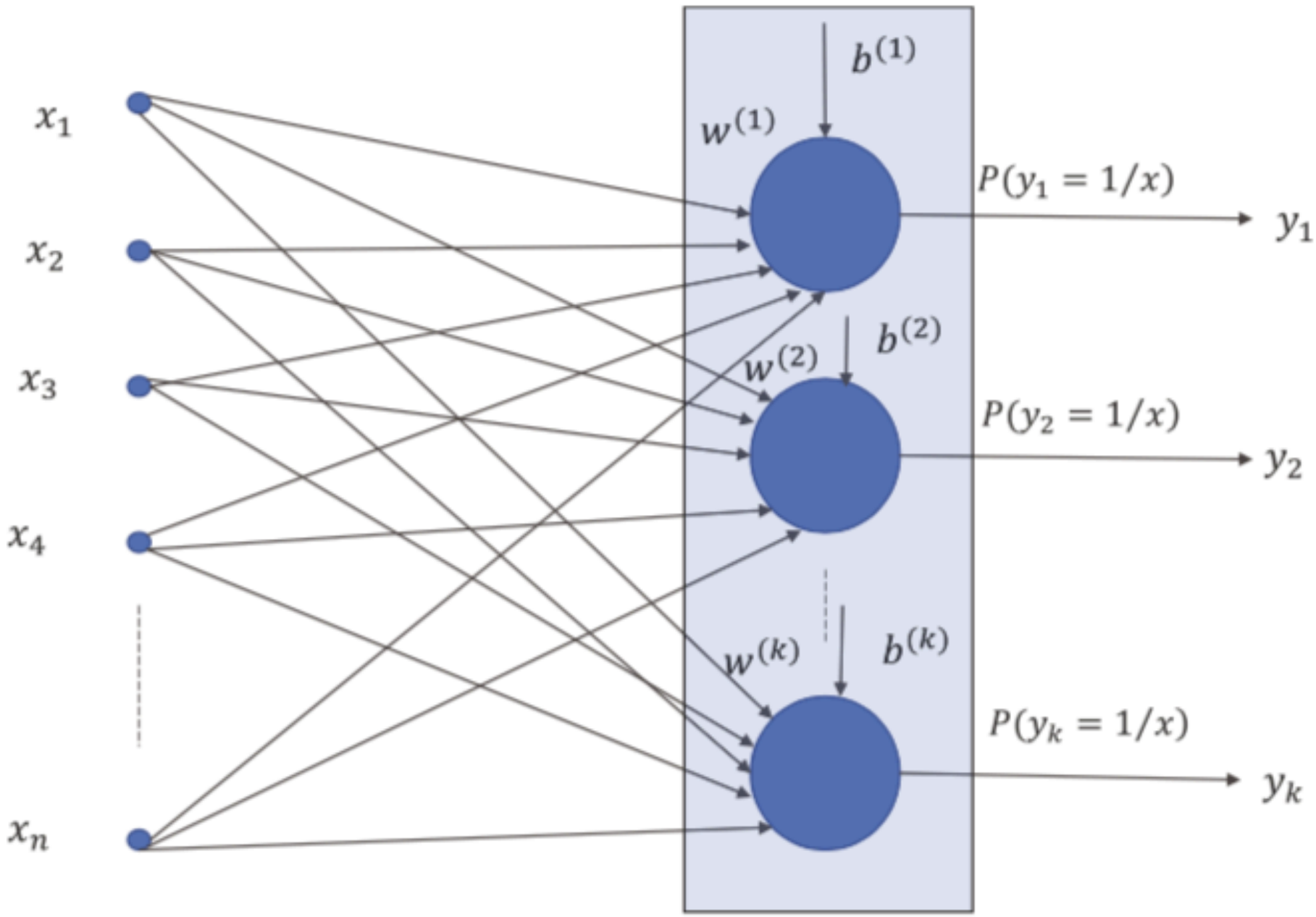


Figure 2-12. *SoftMax activation function*

Momentum

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}})$$

weight increment

learning rate

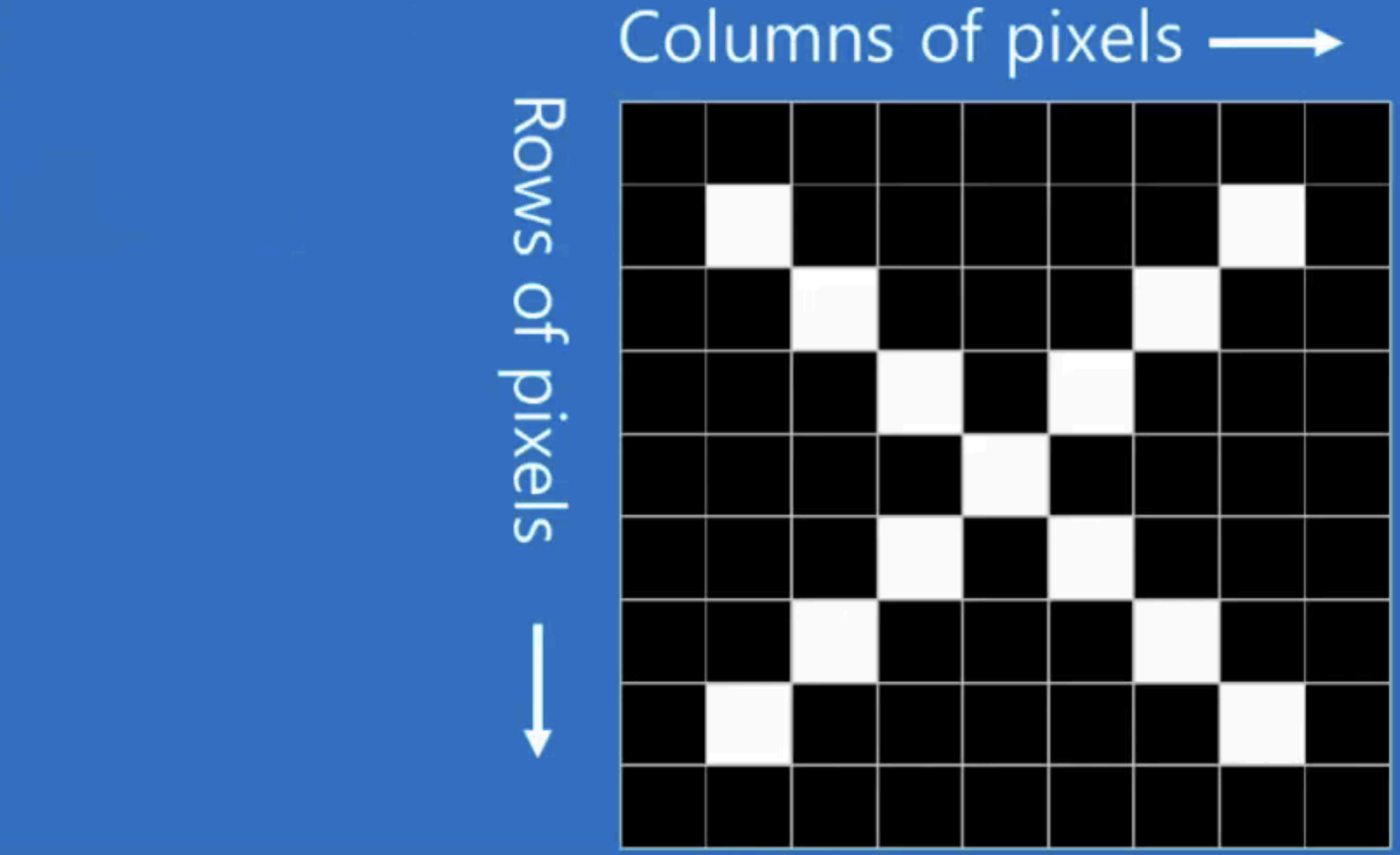
weight gradient

$$\Delta w_{ij} = (\eta * \frac{\partial E}{\partial w_{ij}}) + (\gamma * \Delta w_{ij}^{t-1})$$

momentum factor

weight increment, previous iteration

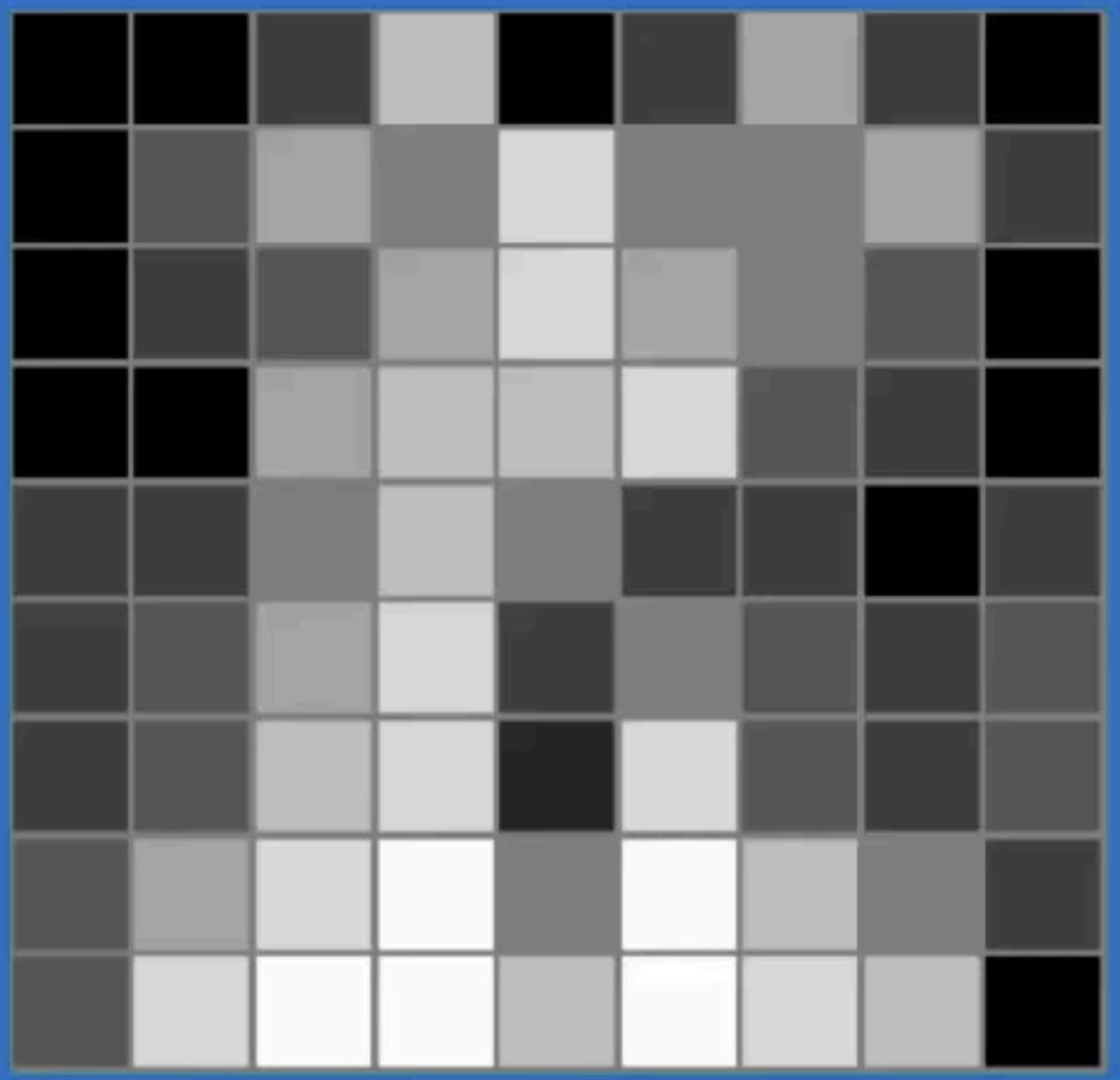
Images



Sound

Time steps →

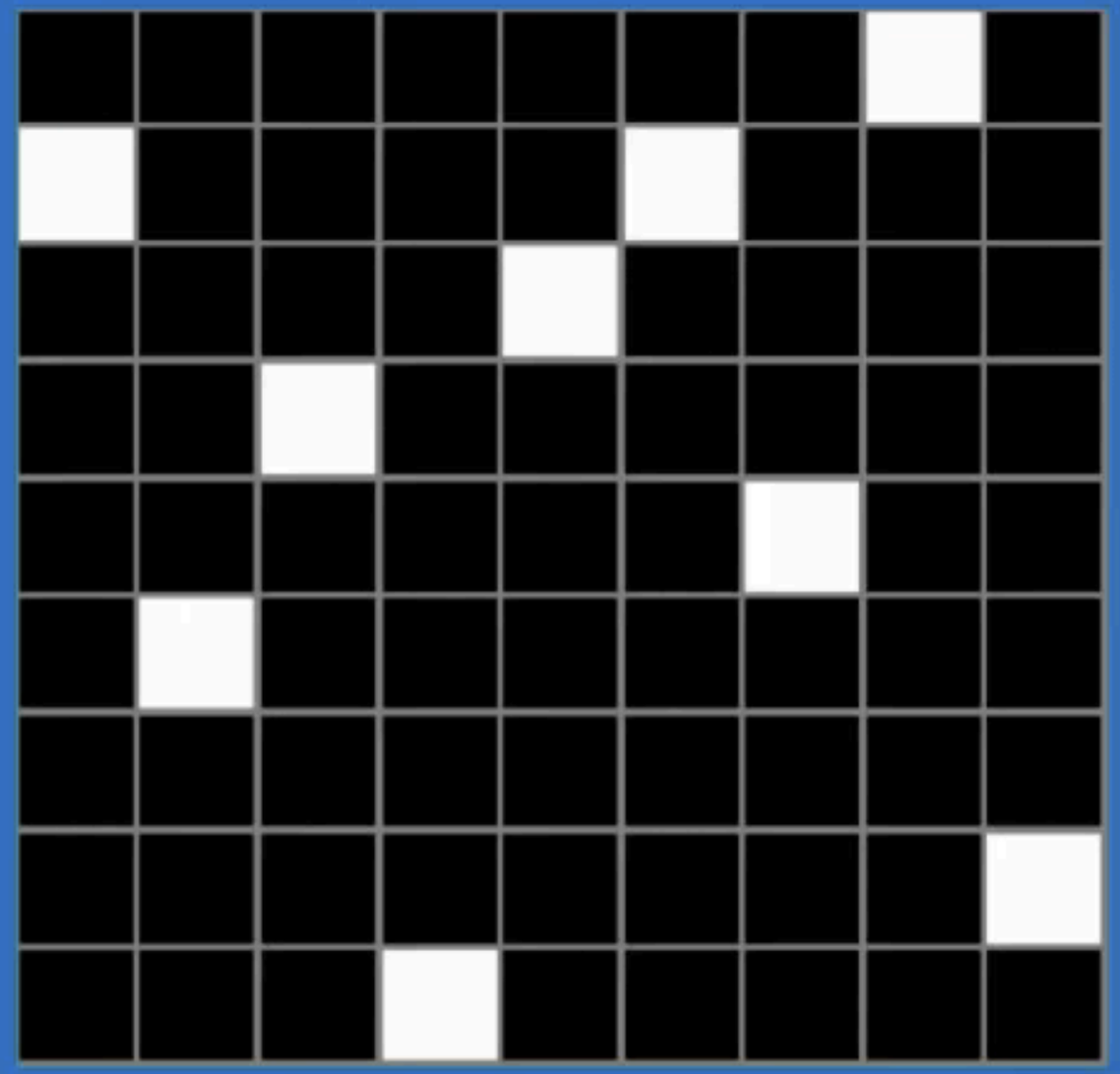
Intensity in each frequency band →



Text

Position in sentence →

Words in dictionary ↓



Customer data

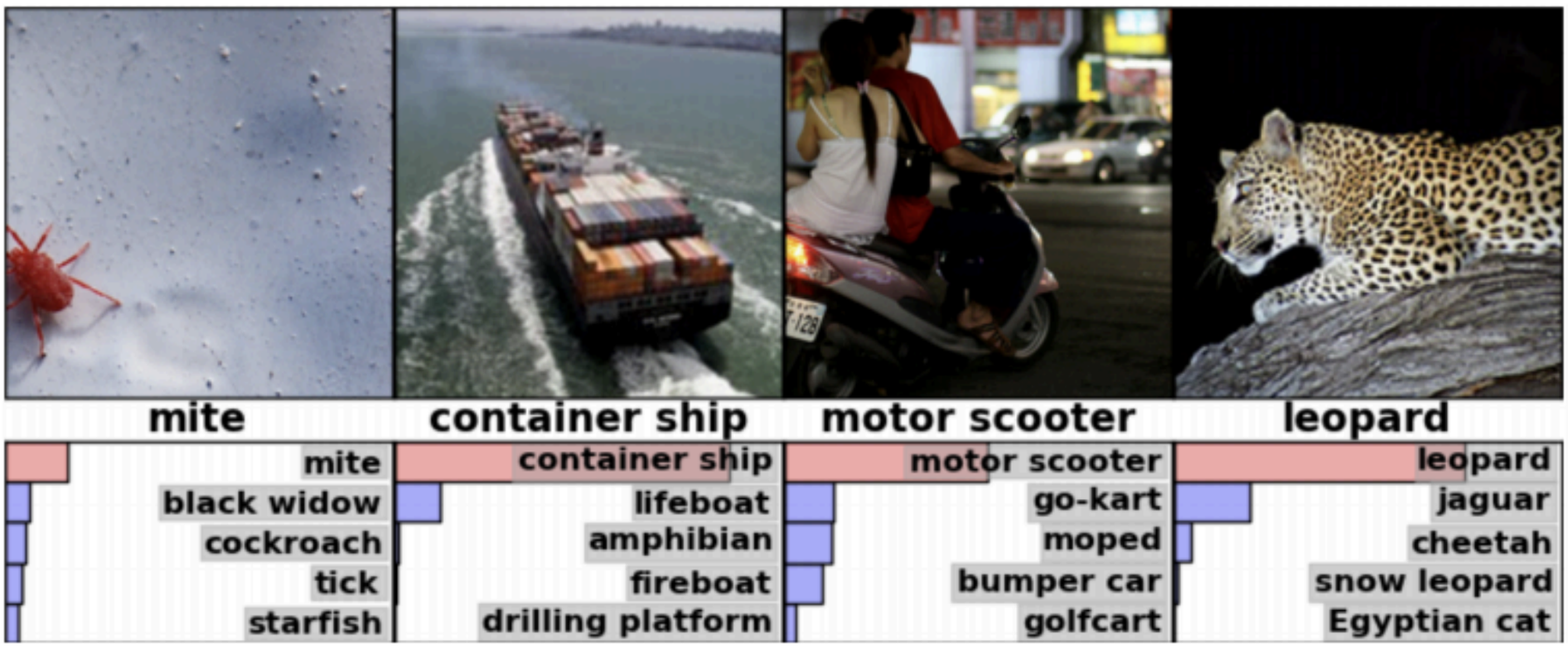
Name, age,
address, email,
purchases,
browsing activity,...

Customers

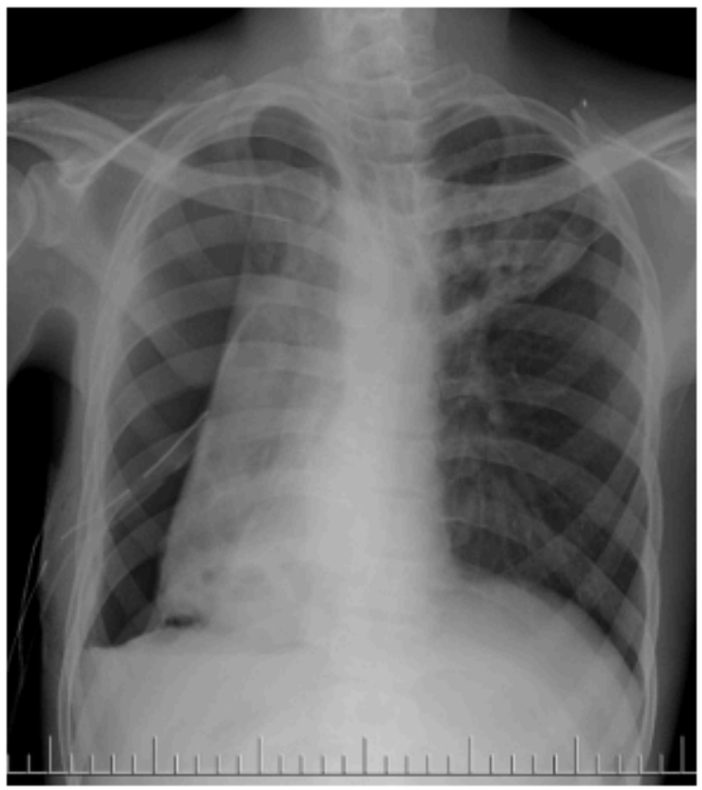
A	22	1A	a@a	1	aa	a1.a	123	aa1
B	33	2B	b@b	2	bb	b2.b	234	bb2
C	44	3C	c@c	3	cc	c3.c	345	cc3
D	55	4D	d@d	4	dd	d4.d	456	dd4
E	66	5E	e@e	5	ee	e5.e	567	ee5
F	77	6F	f@f	6	ff	f6.f	678	ff6
G	88	7G	g@g	7	gg	g7.g	789	gg7
H	99	8H	h@h	8	hh	h8.h	890	hh8
I	111	9I	i@i	9	ii	i9.i	901	ii9

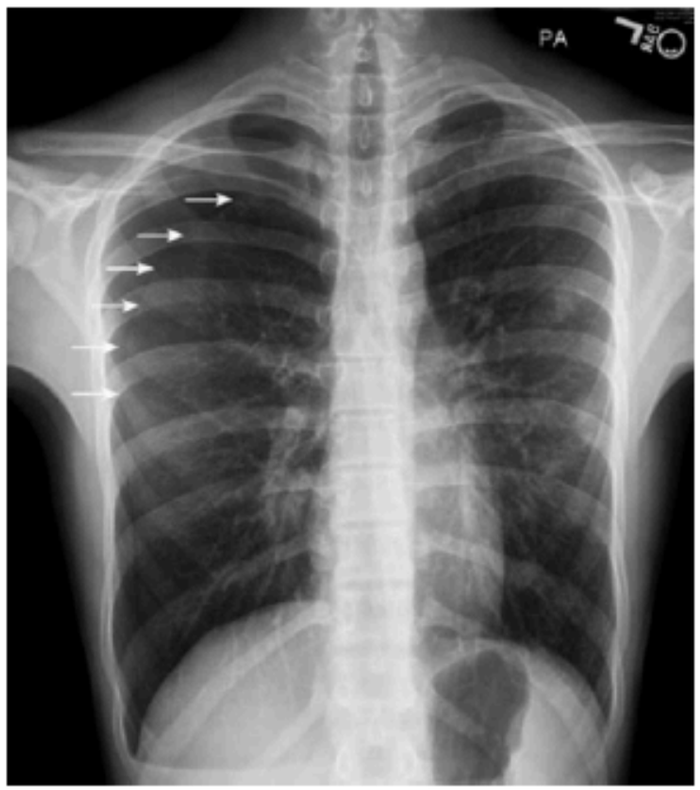
Deep Learning Success: Vision

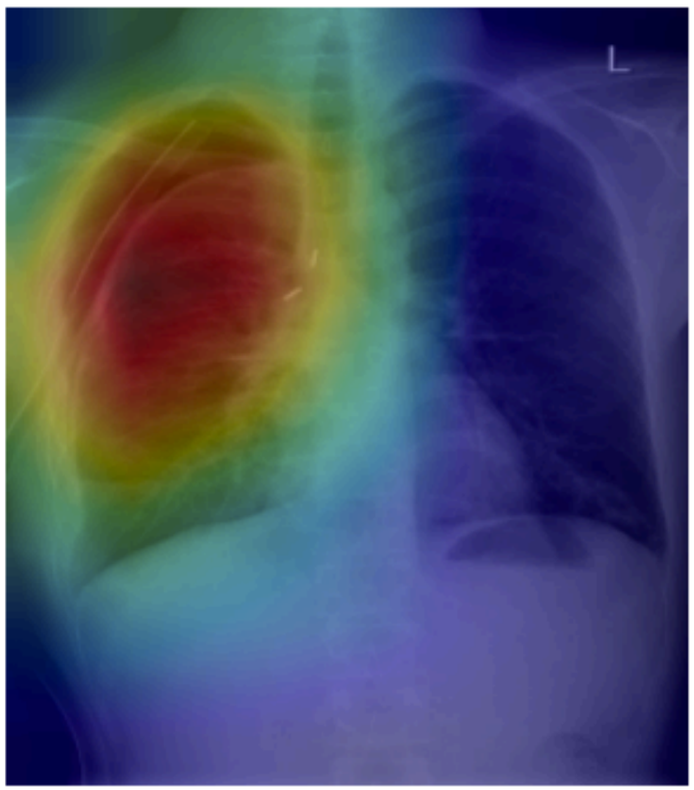
Image Recognition



Detect pneumothorax in real X-Ray scans





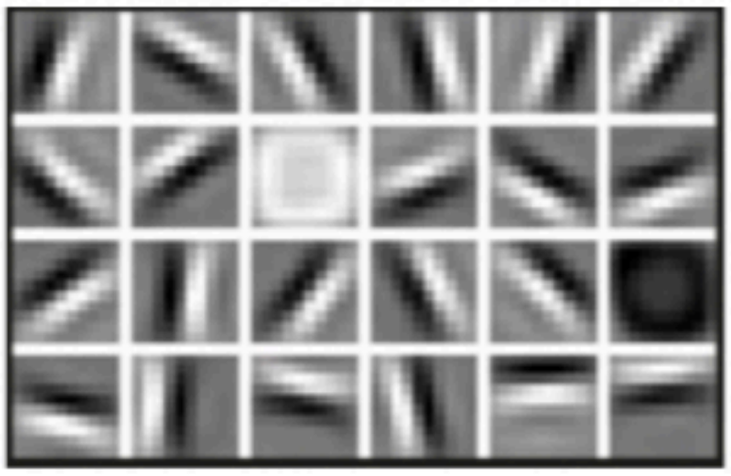


Why Deep Learning?

Hand engineered features are time consuming, brittle and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features



Lines & Edges

Mid Level Features

Eyes & Nose & Ears

High Level Features

Facial Structure

Image Categorization: Training phase

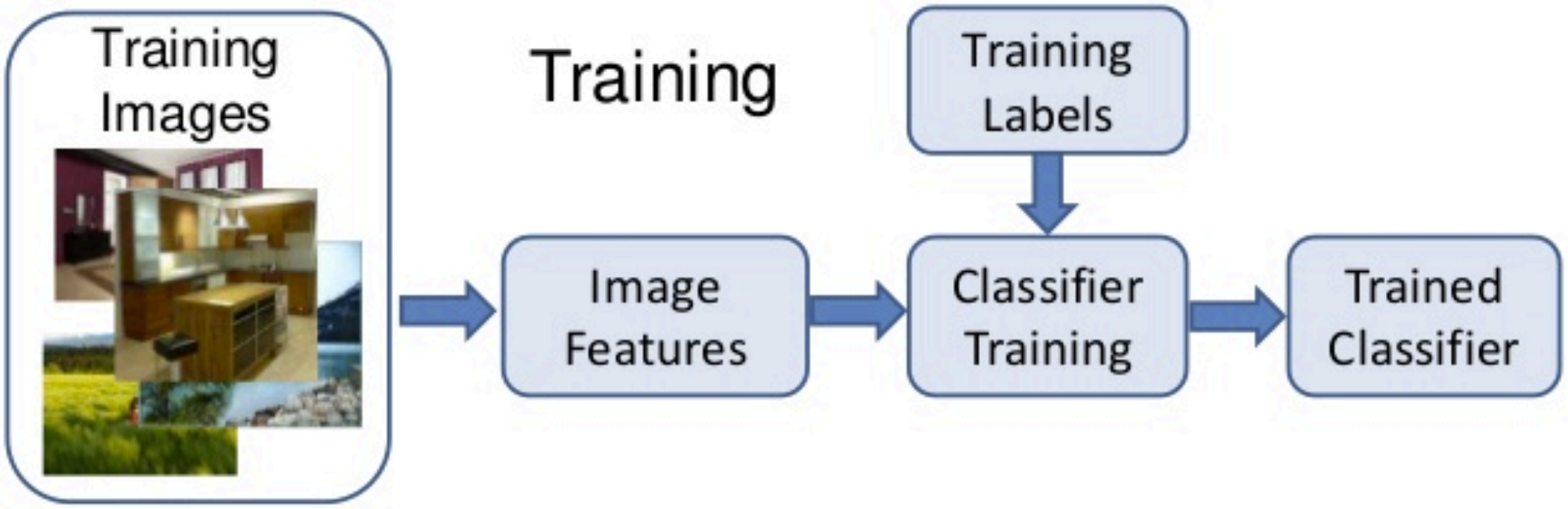
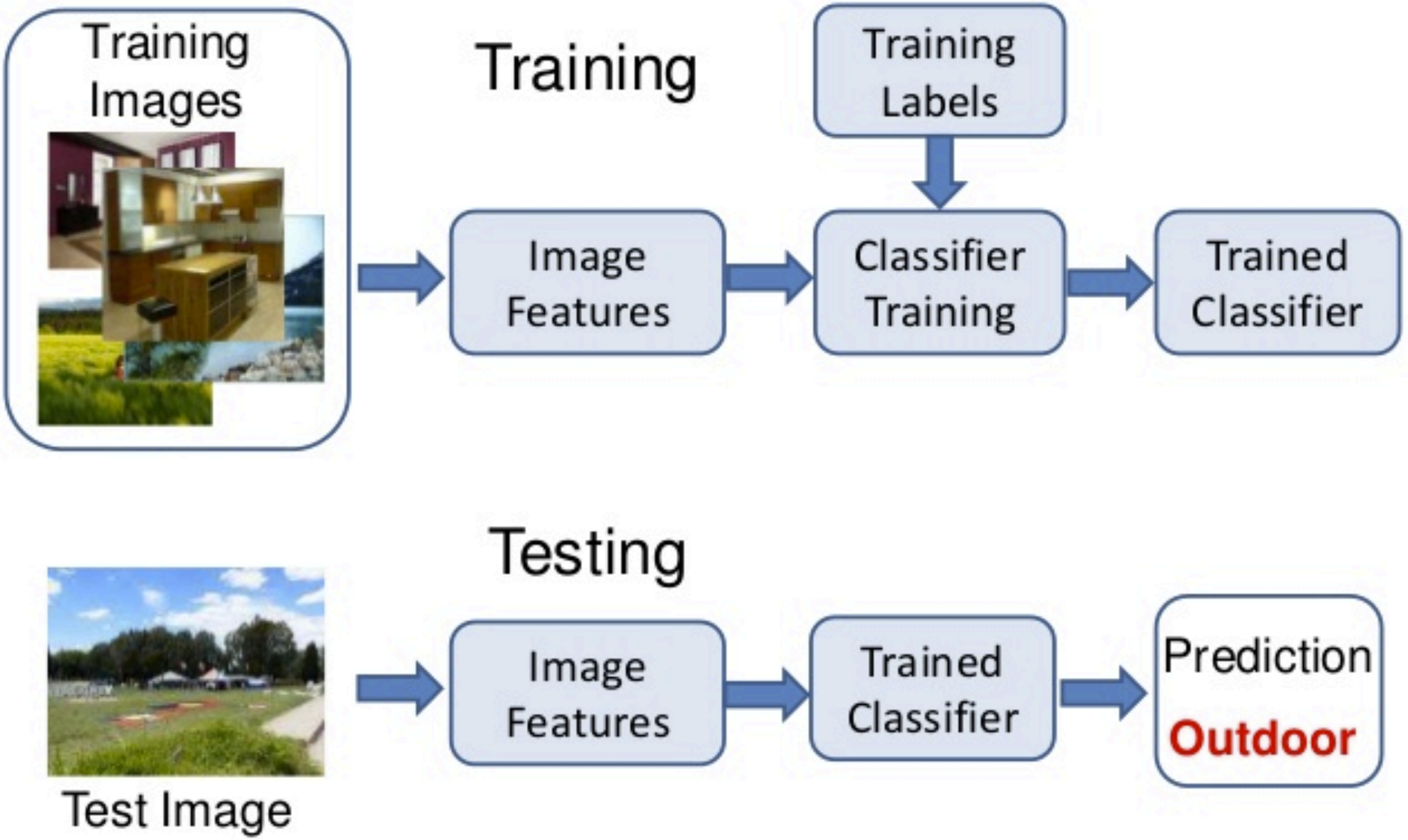
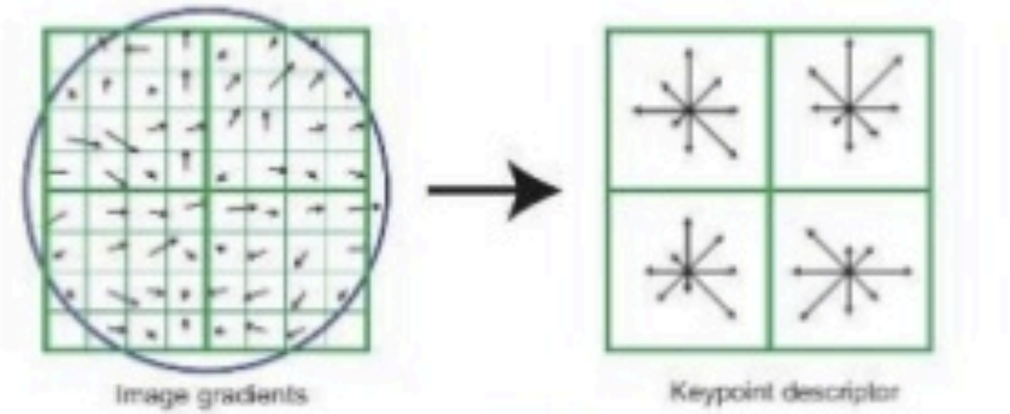


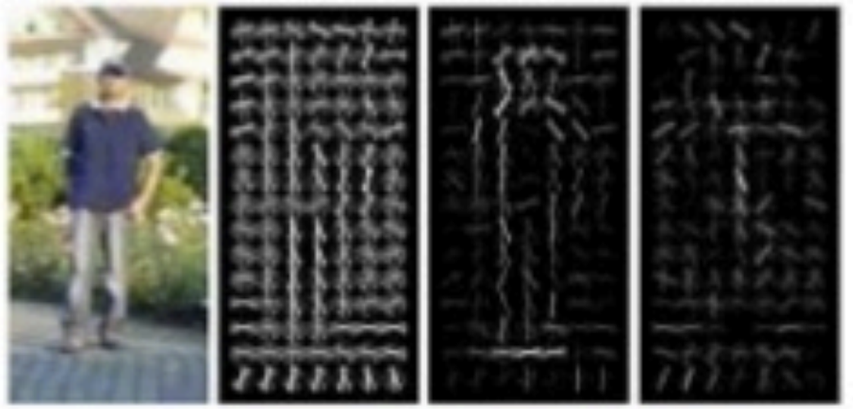
Image Categorization: Testing phase



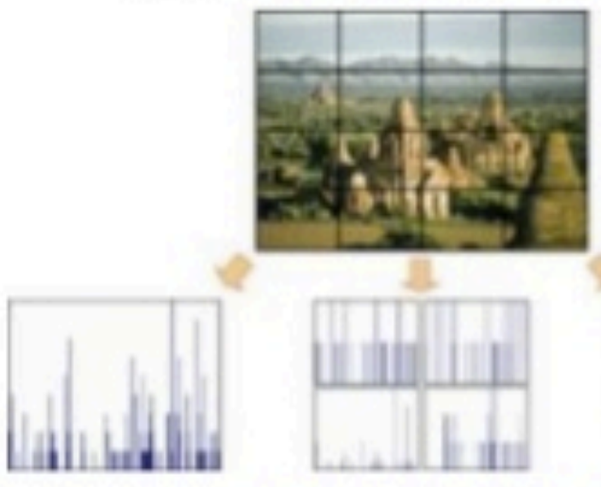
Features are the Keys



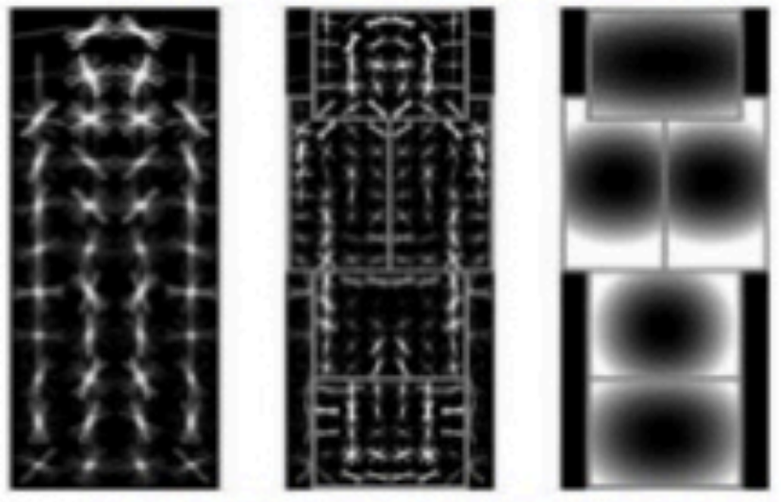
SIFT [Loewe IJCV 04]



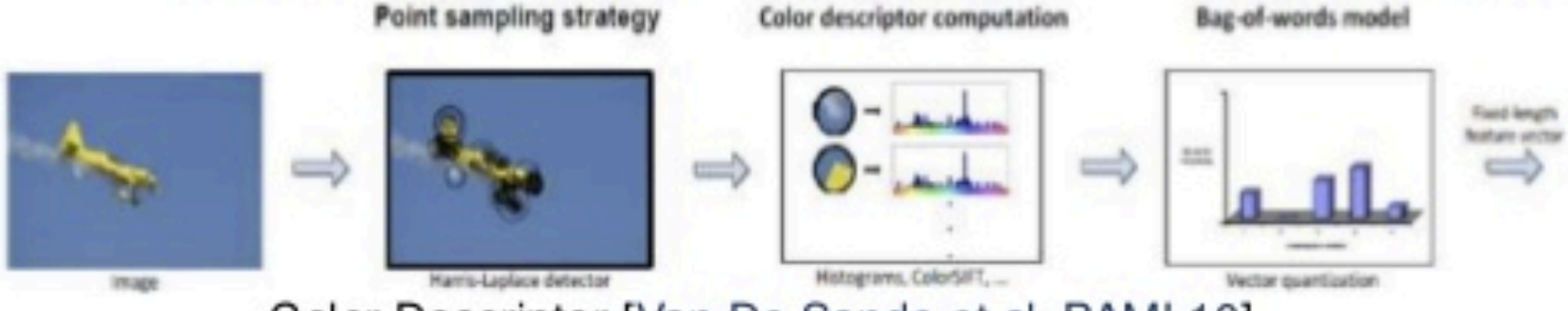
HOG [Dalal and Triggs CVPR 05]



SPM [Lazebnik et al. CVPR 06]



DPM [Felzenszwalb et al. PAMI 10]

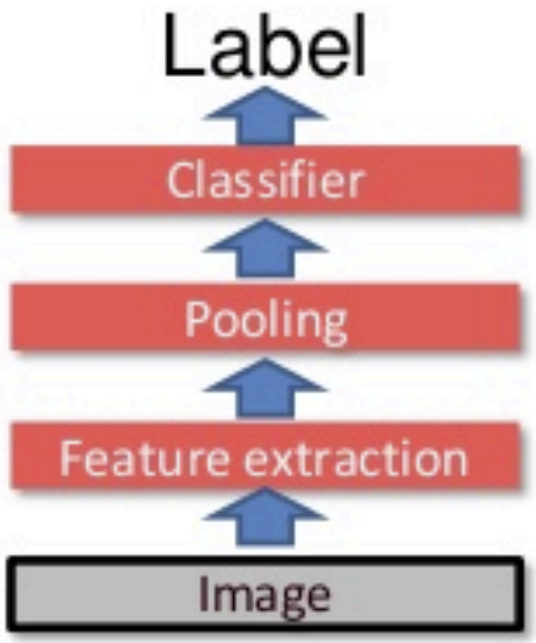


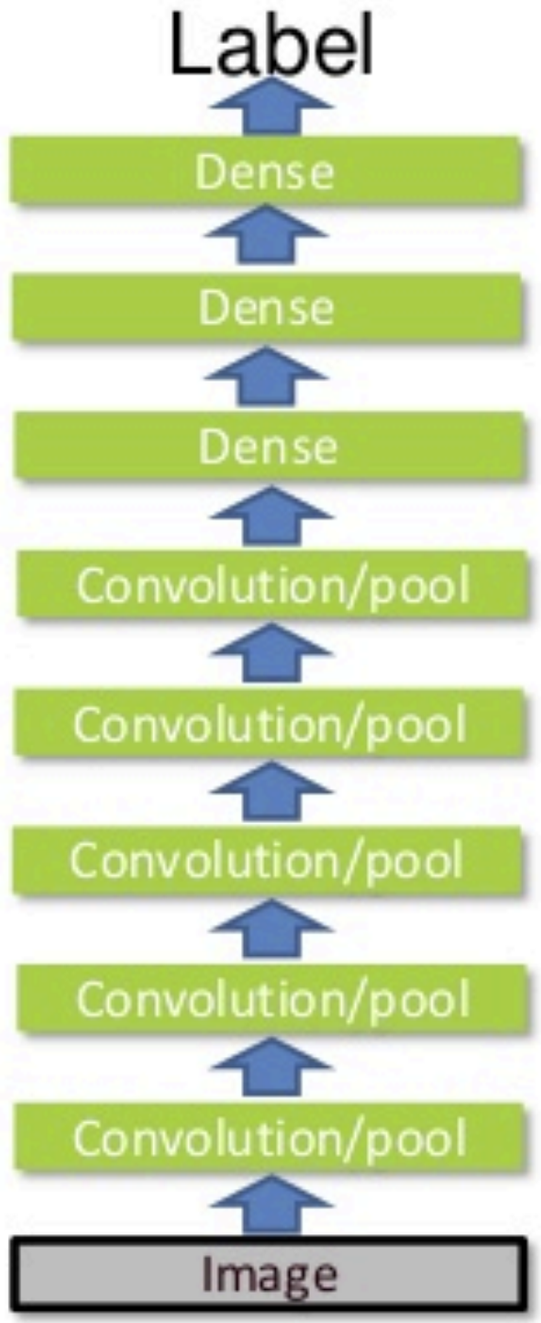
Color Descriptor [Van De Sande et al. PAMI 10]

Learning a Hierarchy of Feature Extractors

- Each layer of hierarchy extracts features from output of previous layer
- All the way from pixels → classifier
- Layers have the (nearly) same structure

Engineered vs. learned features

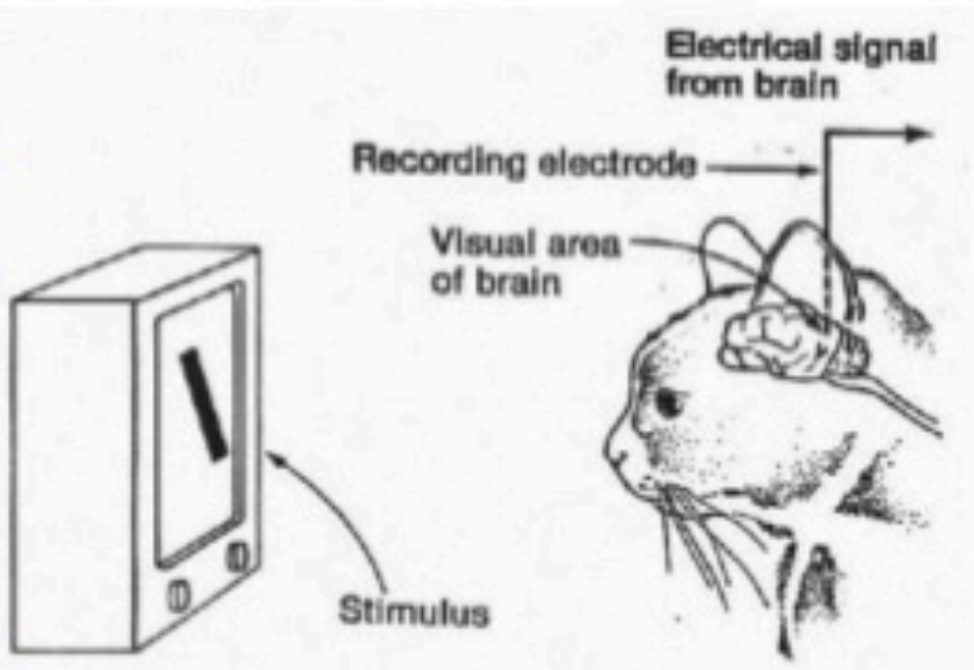




Simple, Complex and Hypercomplex cells

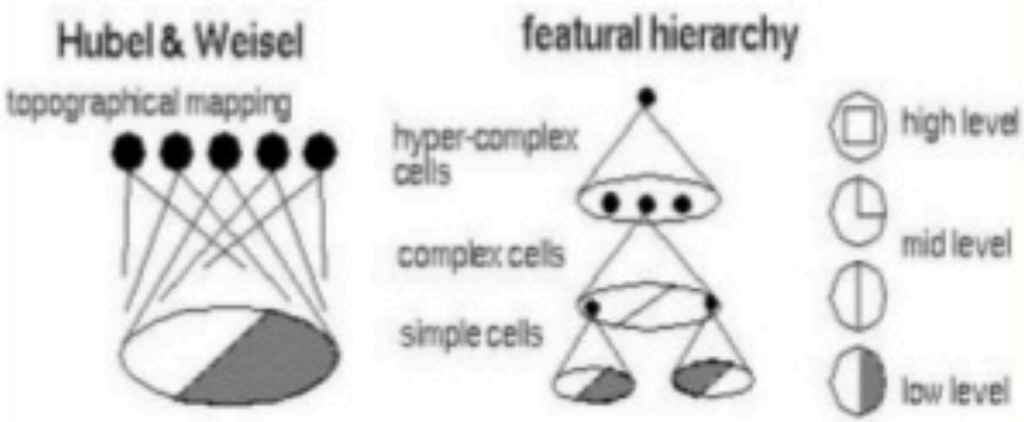
David H. Hubel and Torsten Wiesel

Suggested a **hierarchy** of **feature detectors** in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.

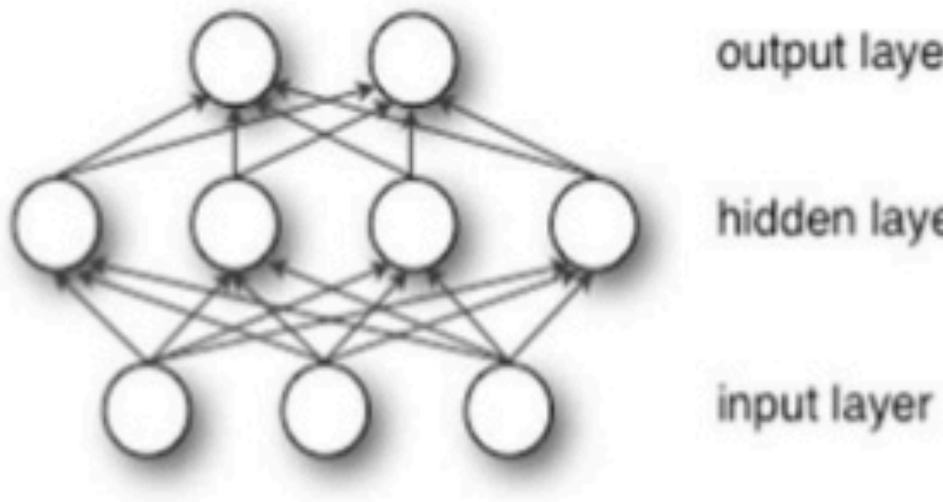


David Hubel's Eye, Brain, and Vision

Hubel/Wiesel Architecture and Multi-layer Neural Network



Hubel and Weisel's architecture



Multi-layer Neural Network
- A *non-linear* classifier

Convolutional Neural Networks
(CNN, ConvNet, DCN)

- CNN = a multi-layer neural network with
  - **Local** connectivity
  - **Share** weight parameters across spatial positions

- One activation map (a depth slice), computed with one set of weights

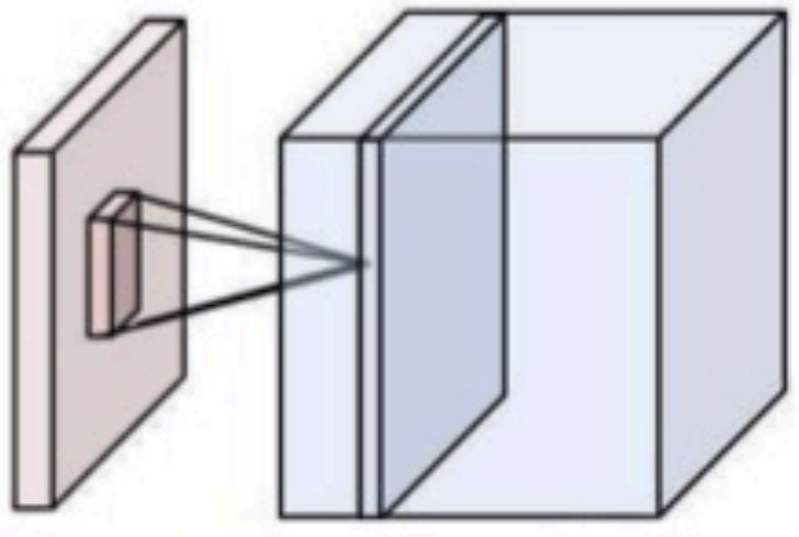


Image credit: A. Karpathy

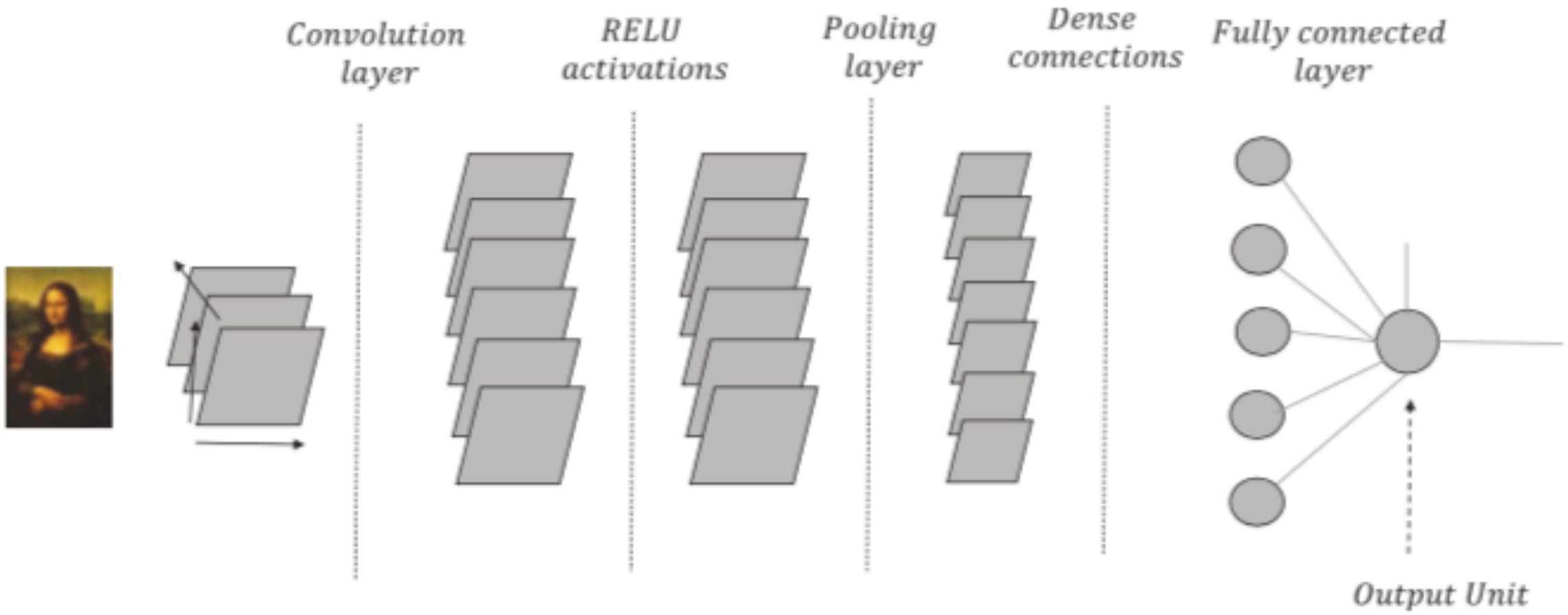


Figure 3-20. *Basic flow diagram of a convolutional neural network*

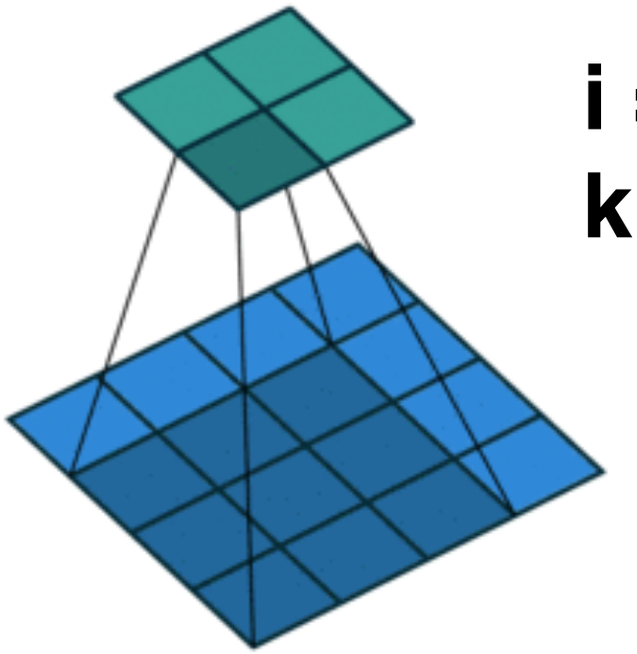
Convolução

$$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

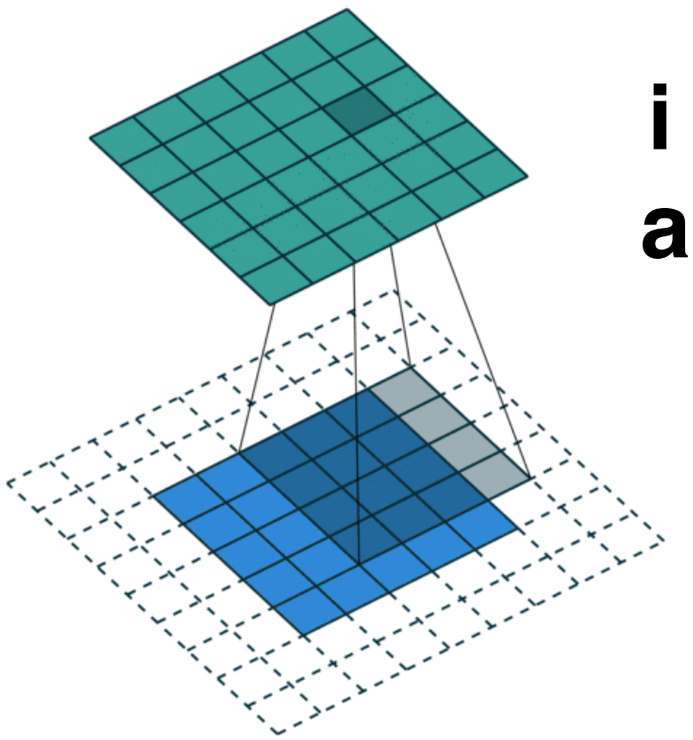
Kernel

Input Vector

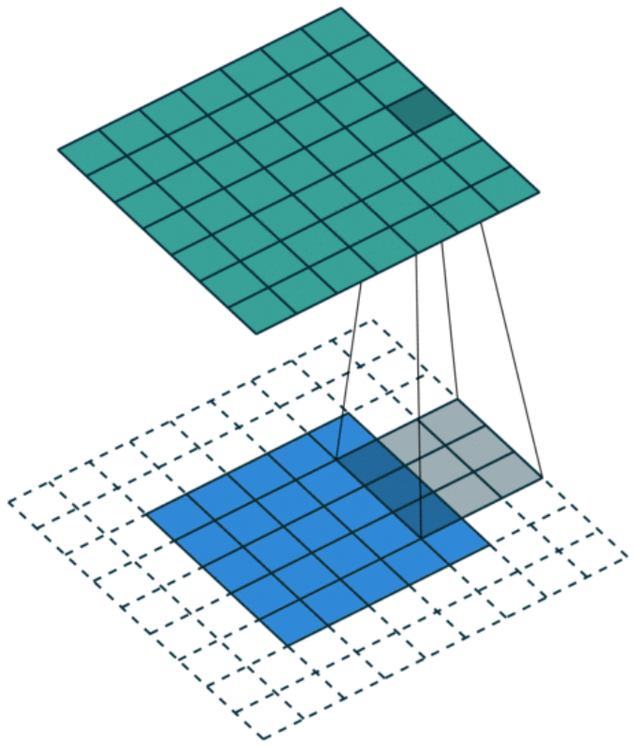
Padding



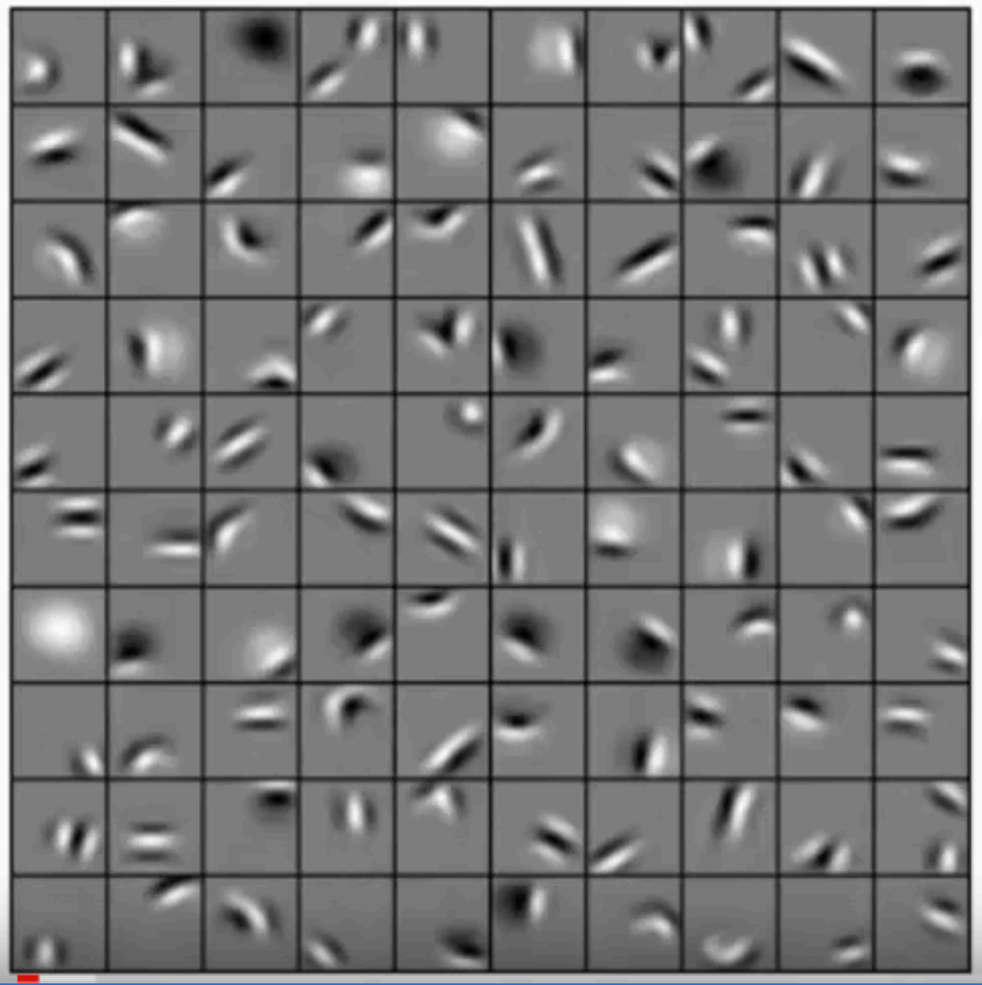
i = 4 and
k = 3



i = 5, k = 4
and p = 2:



i = 5, k = 3 and
p = 2:

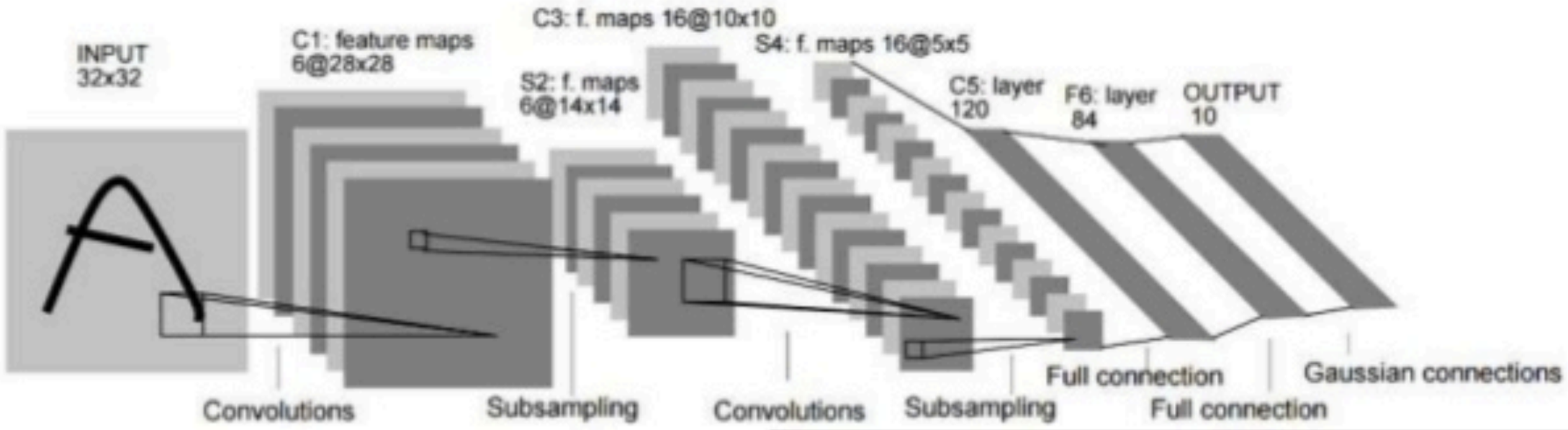


faces

cars

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations
Honglak Lee, Roger Grosse, Rajesh Ranganath, Andrew Y. Ng

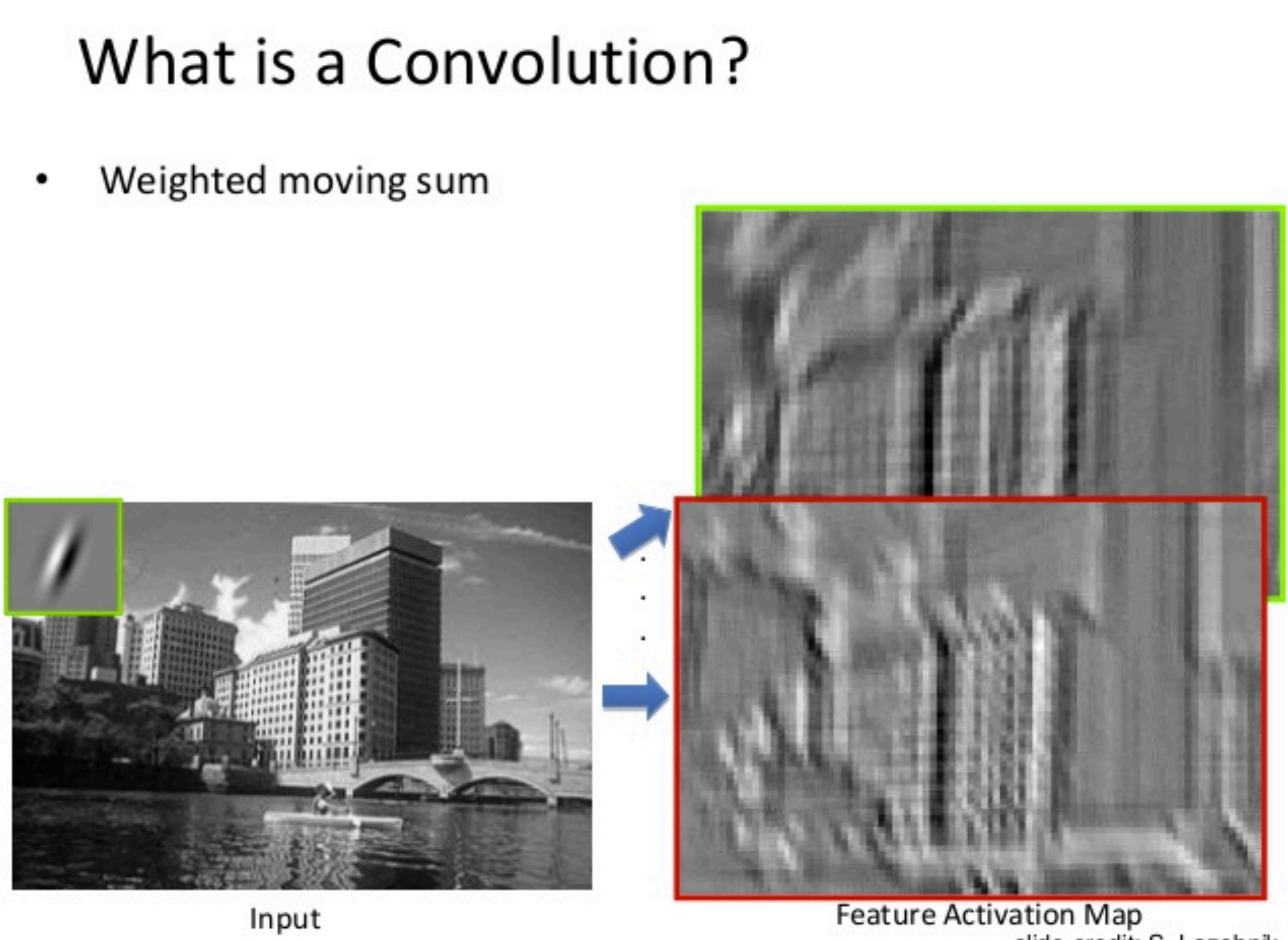
LeNet [LeCun et al. 1998]



Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner 1998]

What is a Convolution?

- Weighted moving sum

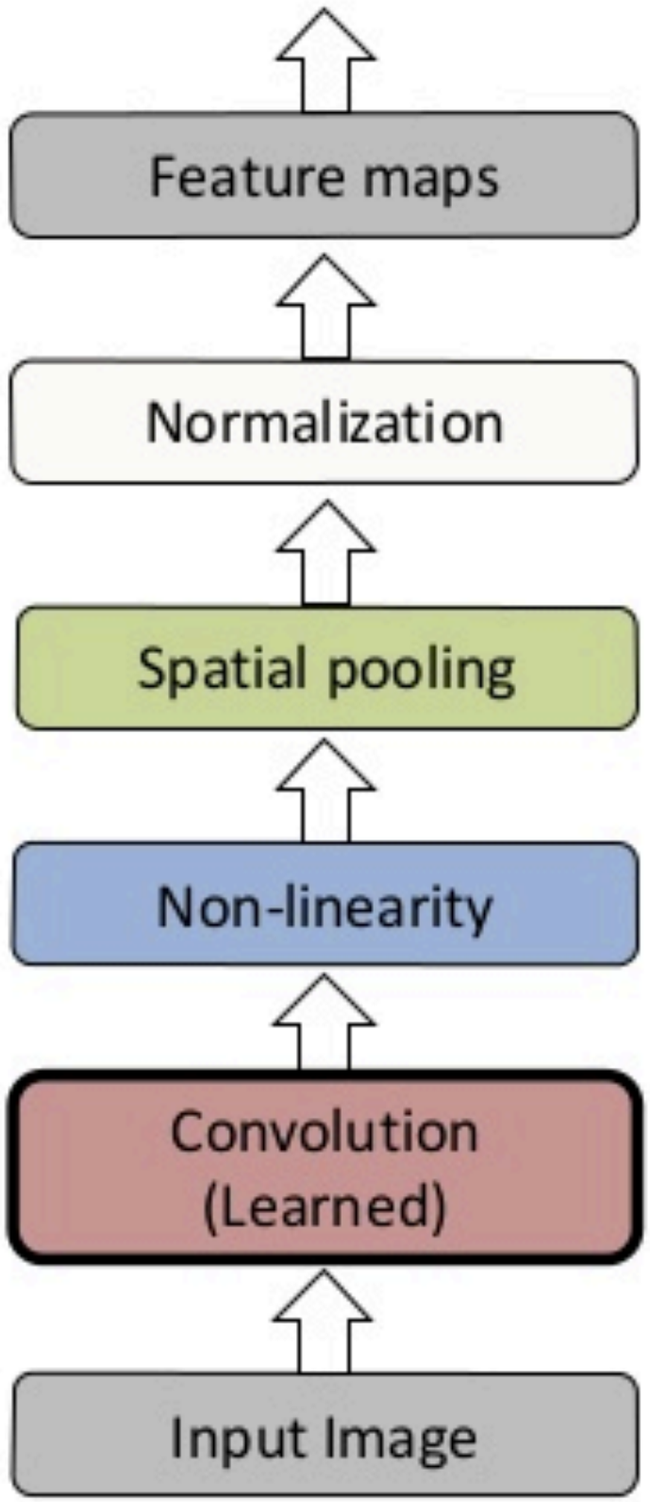


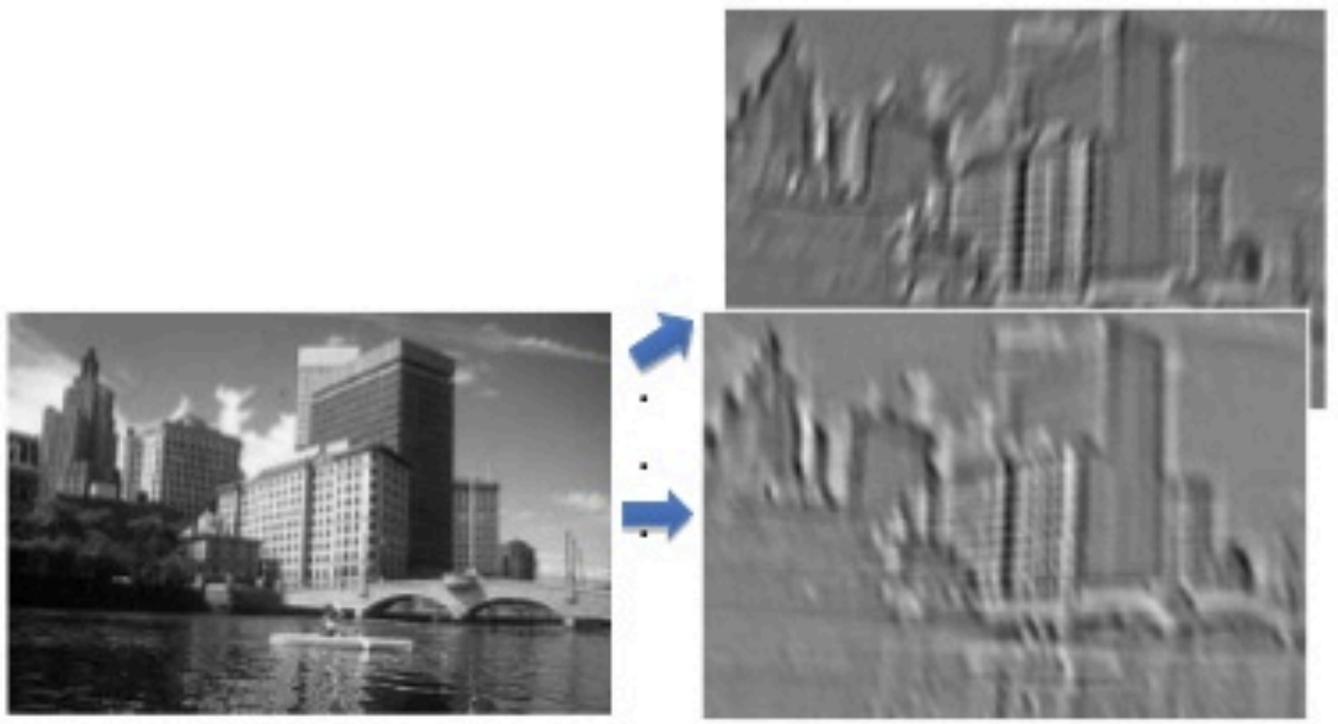
Input

Feature Activation Map

slide credit: S. Lazebnik

Convolutional Neural Networks

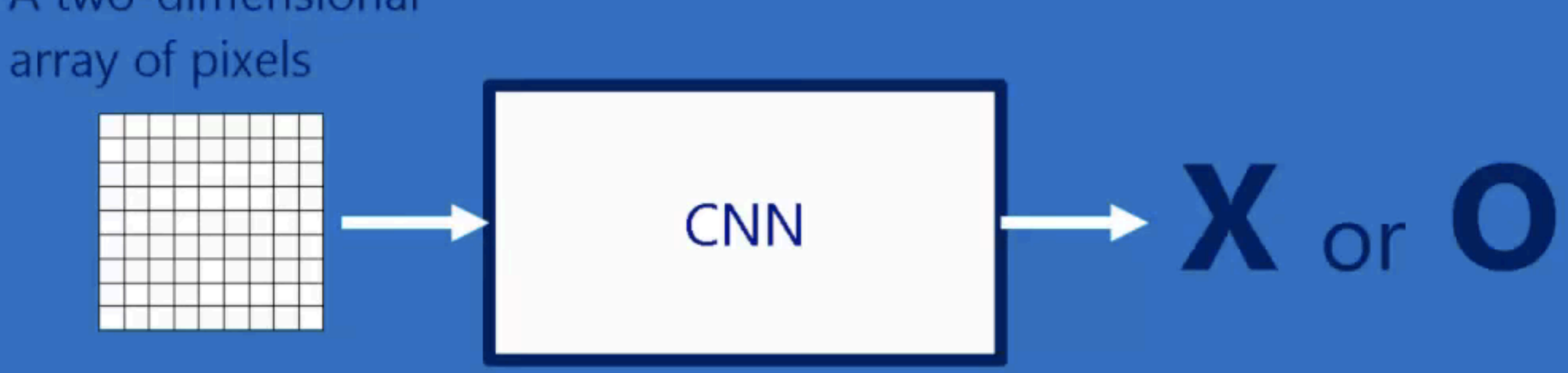




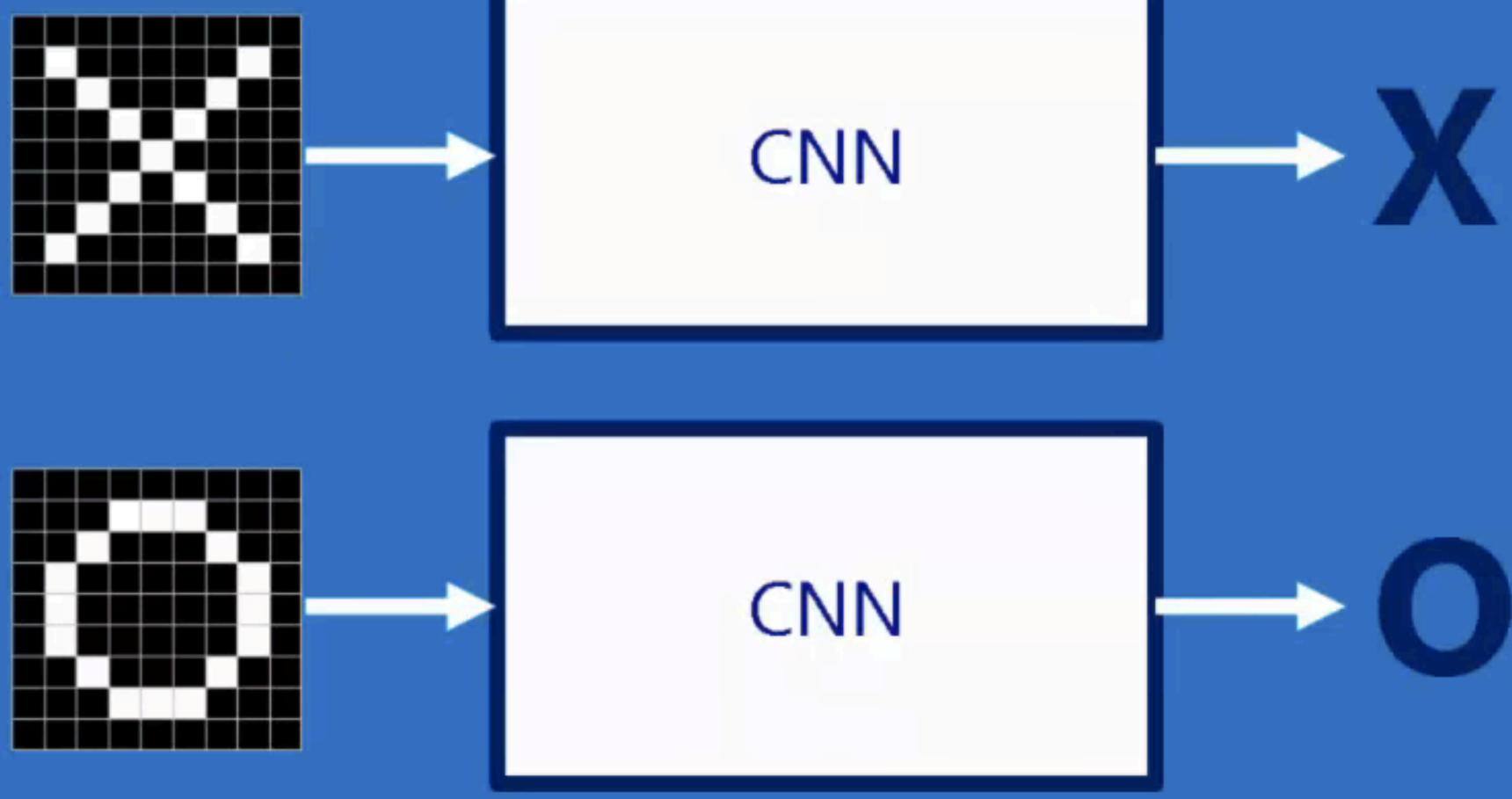
slide credit: S. Lazebnik

A toy ConvNet: X's and O's

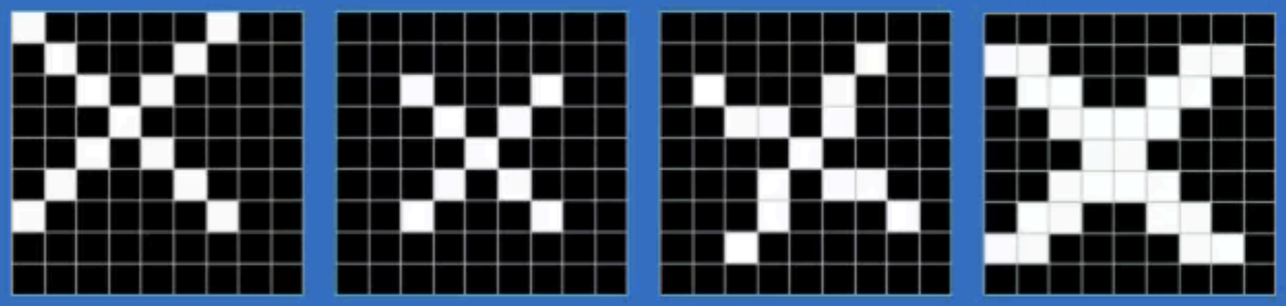
Says whether a picture is of an X or an O



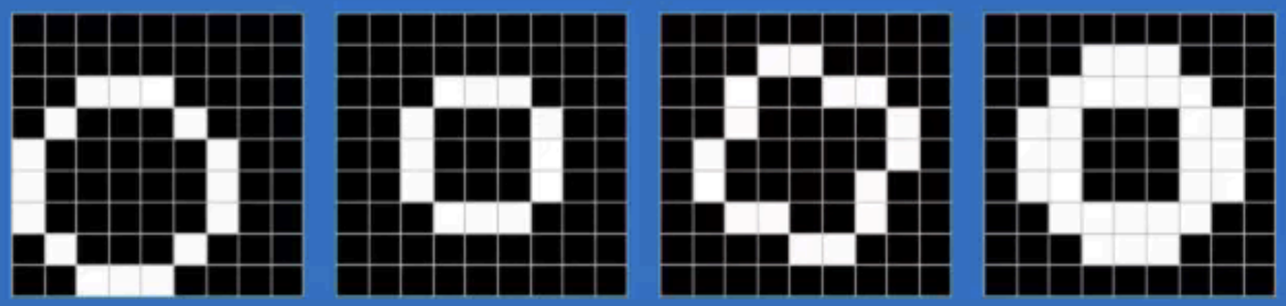
For example

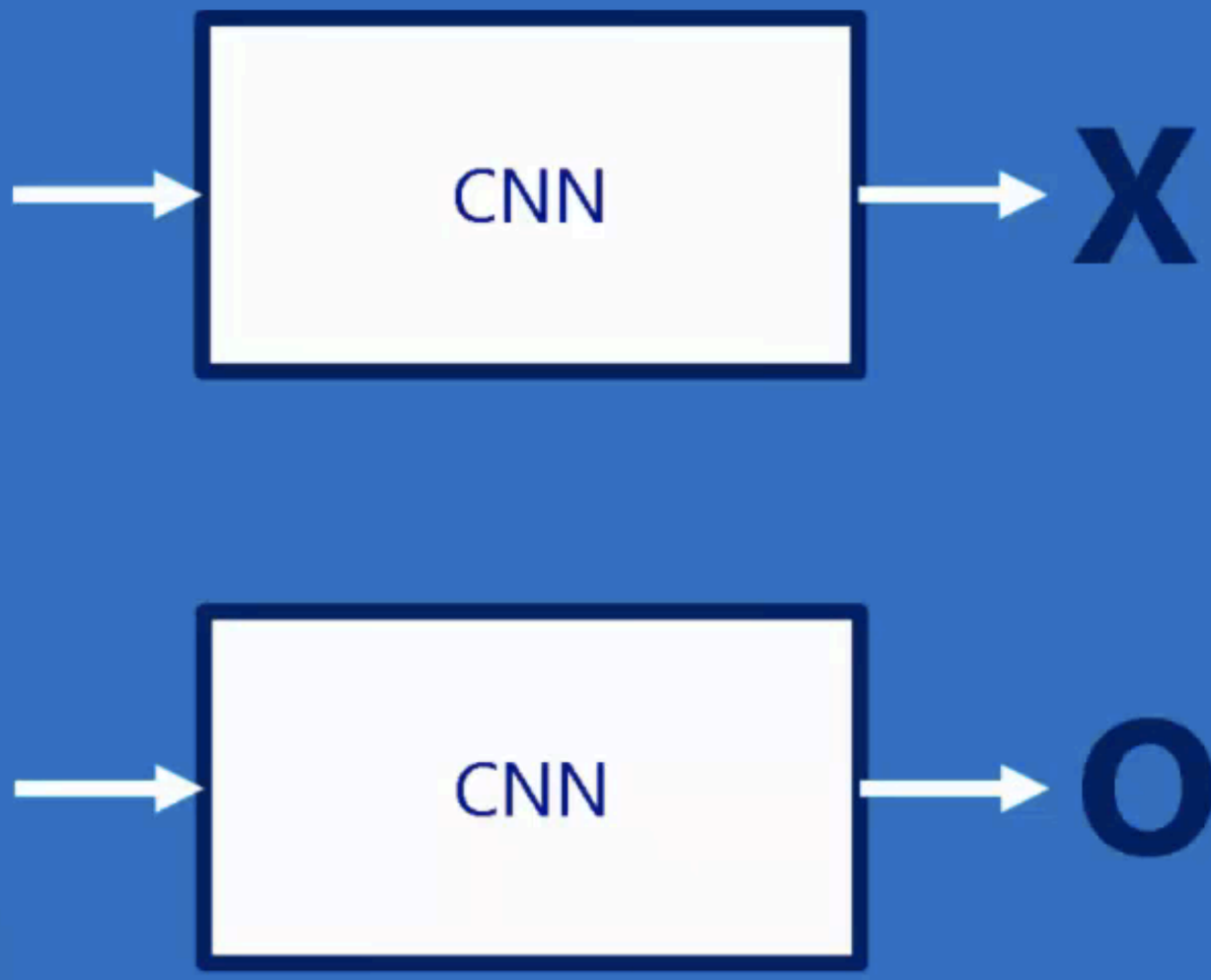


Trickier cases



translation scaling rotation weight





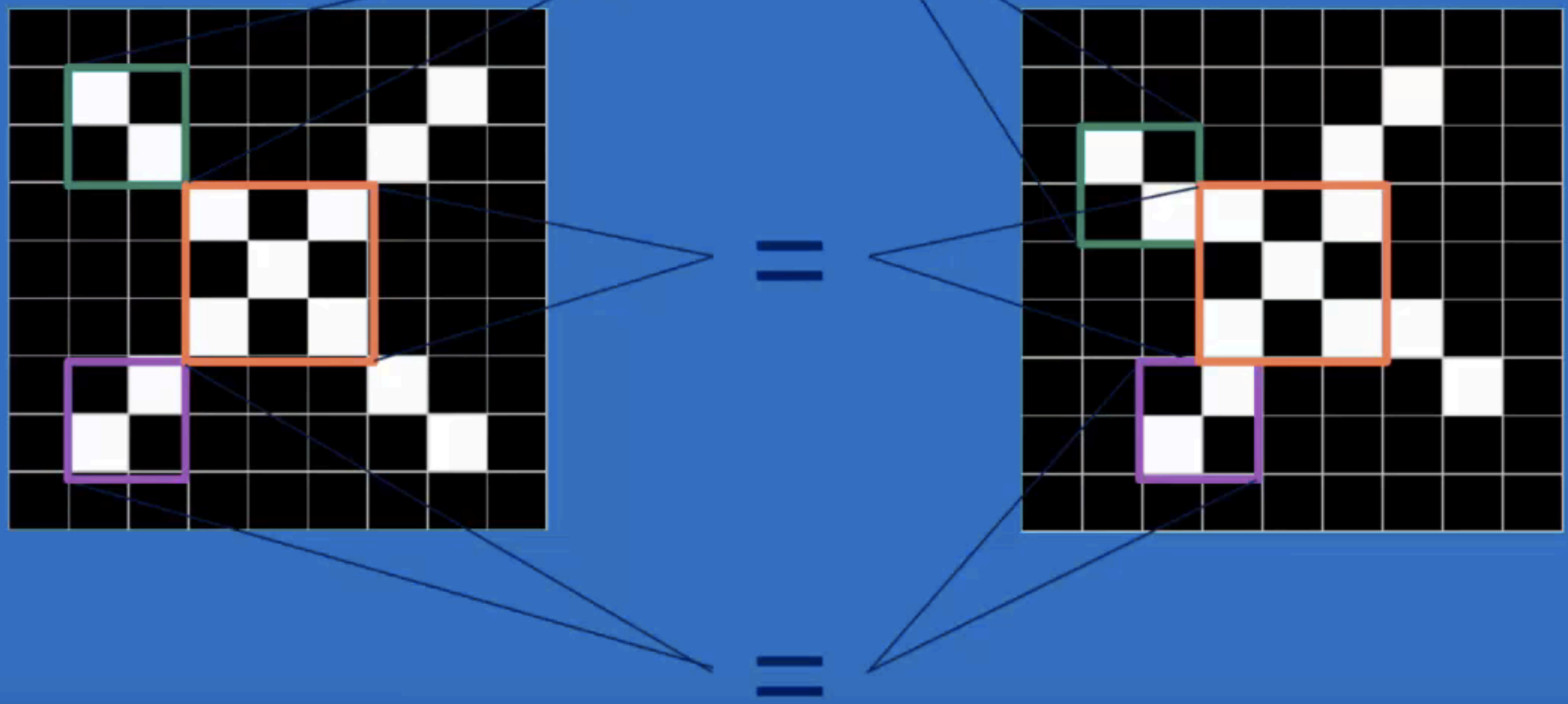
What computers see

-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
-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

ConvNets match pieces of the image



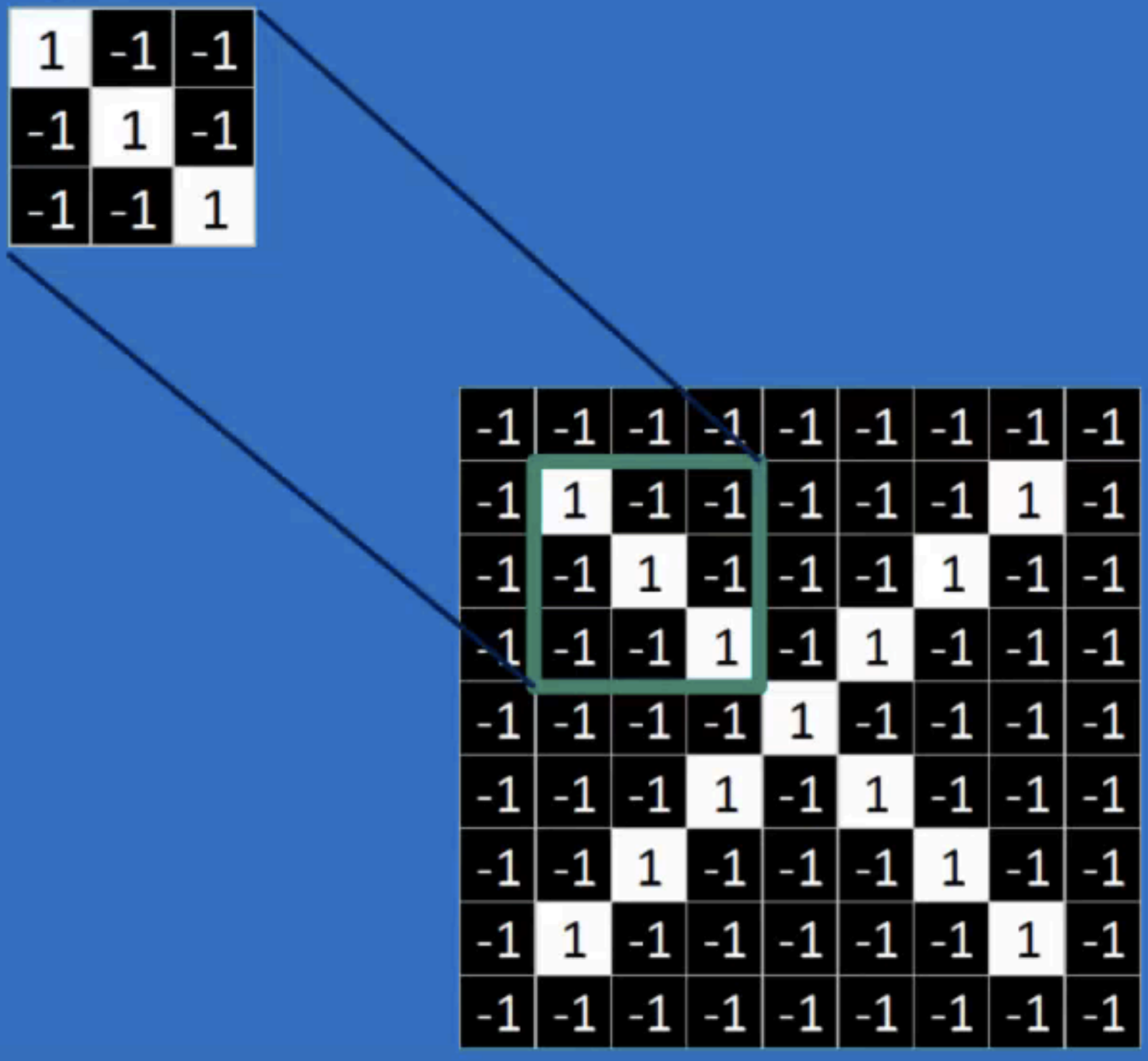
Features match pieces of the image

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

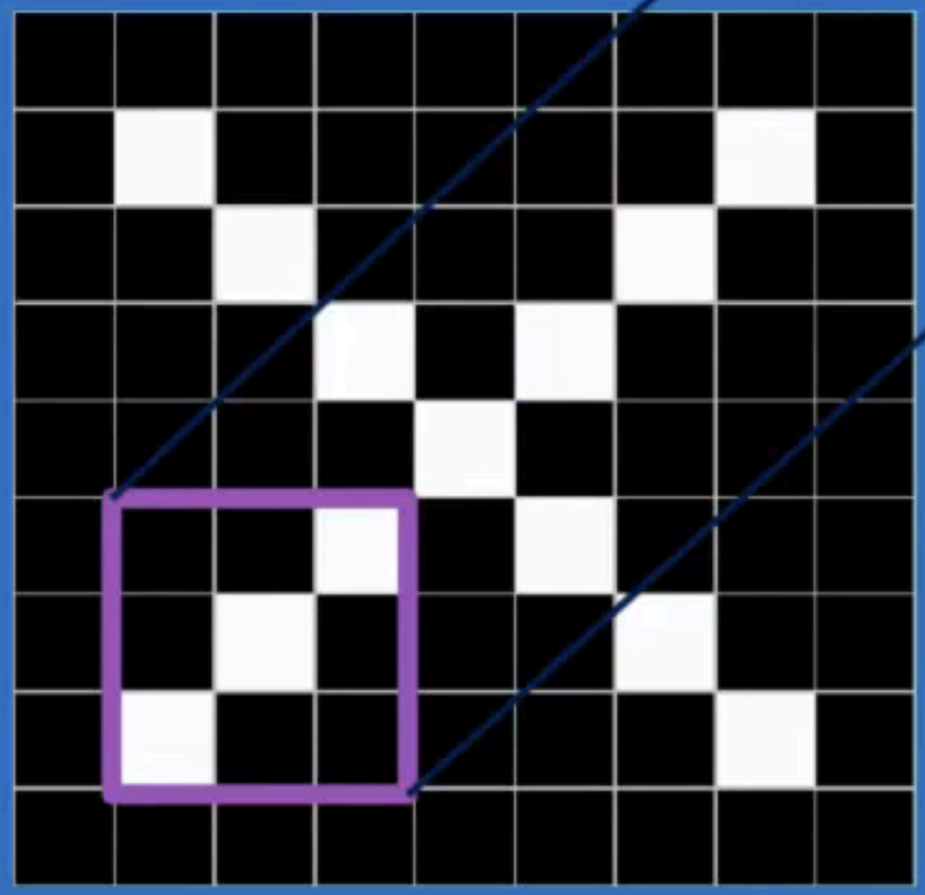
Filtering: The math behind the match



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

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

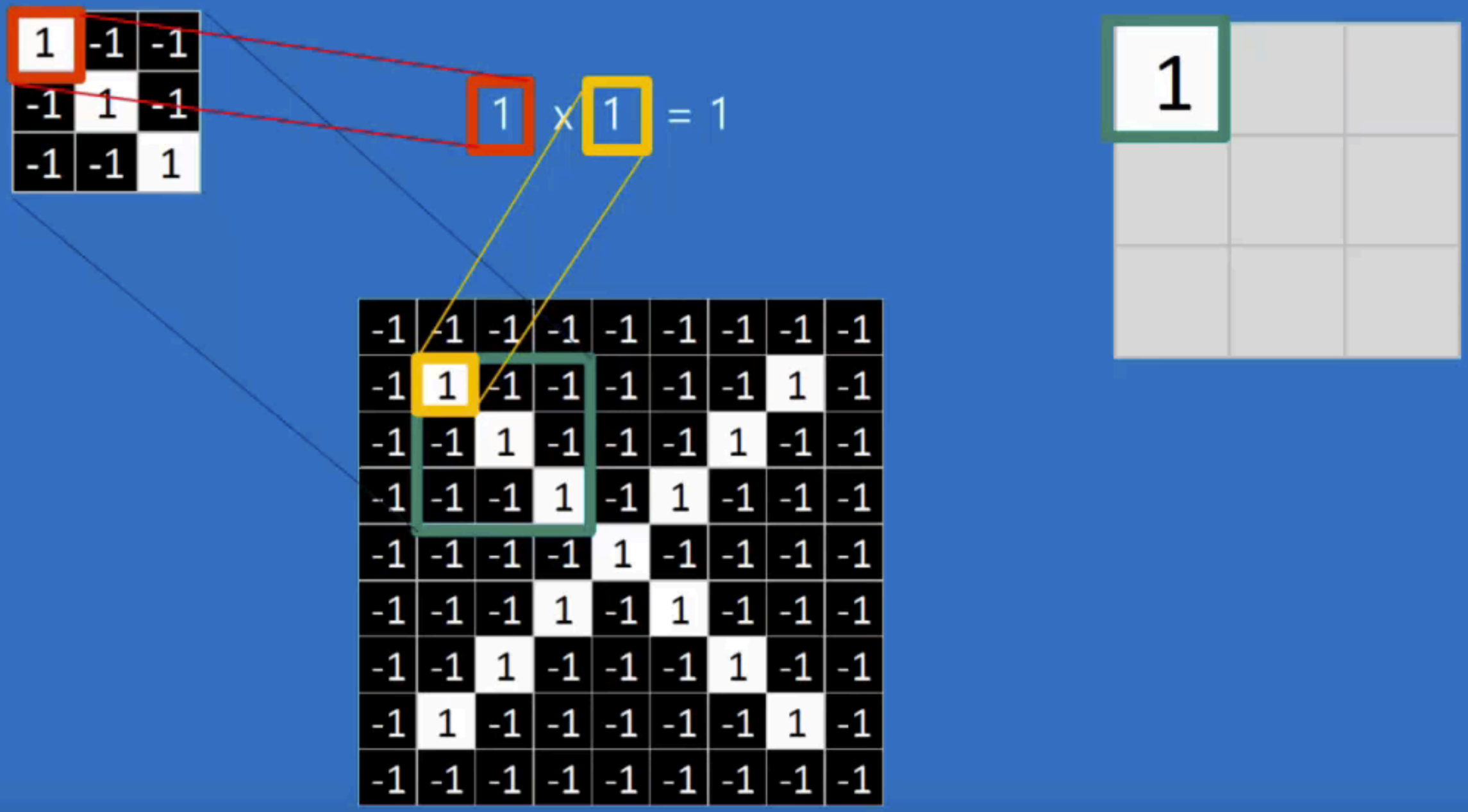
-1	-1	1
-1	1	-1
1	-1	-1



Filtering: The math behind the match

1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

Filtering: The math behind the match



Filtering: The math behind the match

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

1	1	1
1	1	1
1	1	1

$$\frac{1+1+1+1+1+1+1+1+1}{9} = 1$$

-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

Filtering: The math behind the match

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

1	1	-1
1	1	1
-1	1	1

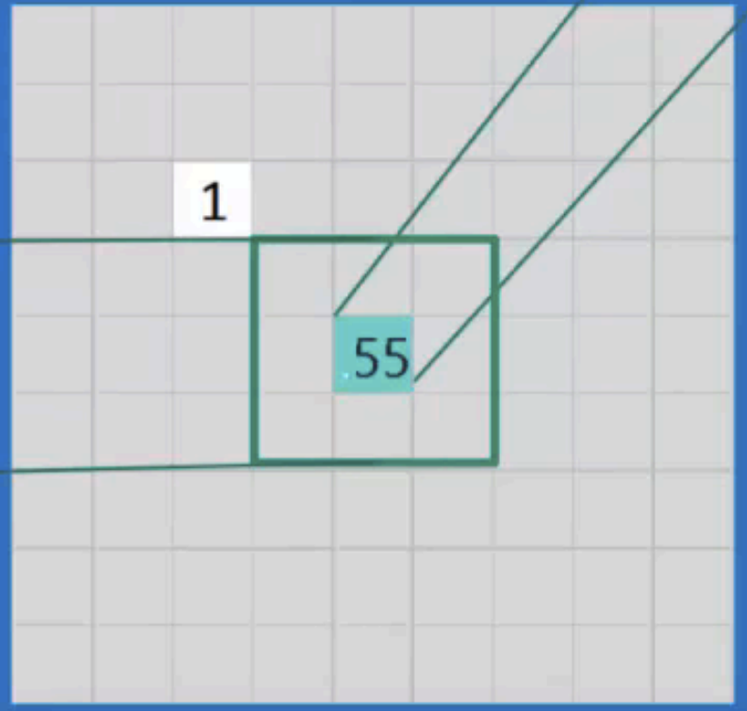
Filtering: The math behind the match

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

1	1	-1
1	1	1
-1	1	1

$$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$$

-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: Trying every possible match

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

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

-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

=

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

-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

=

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

-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

⊗

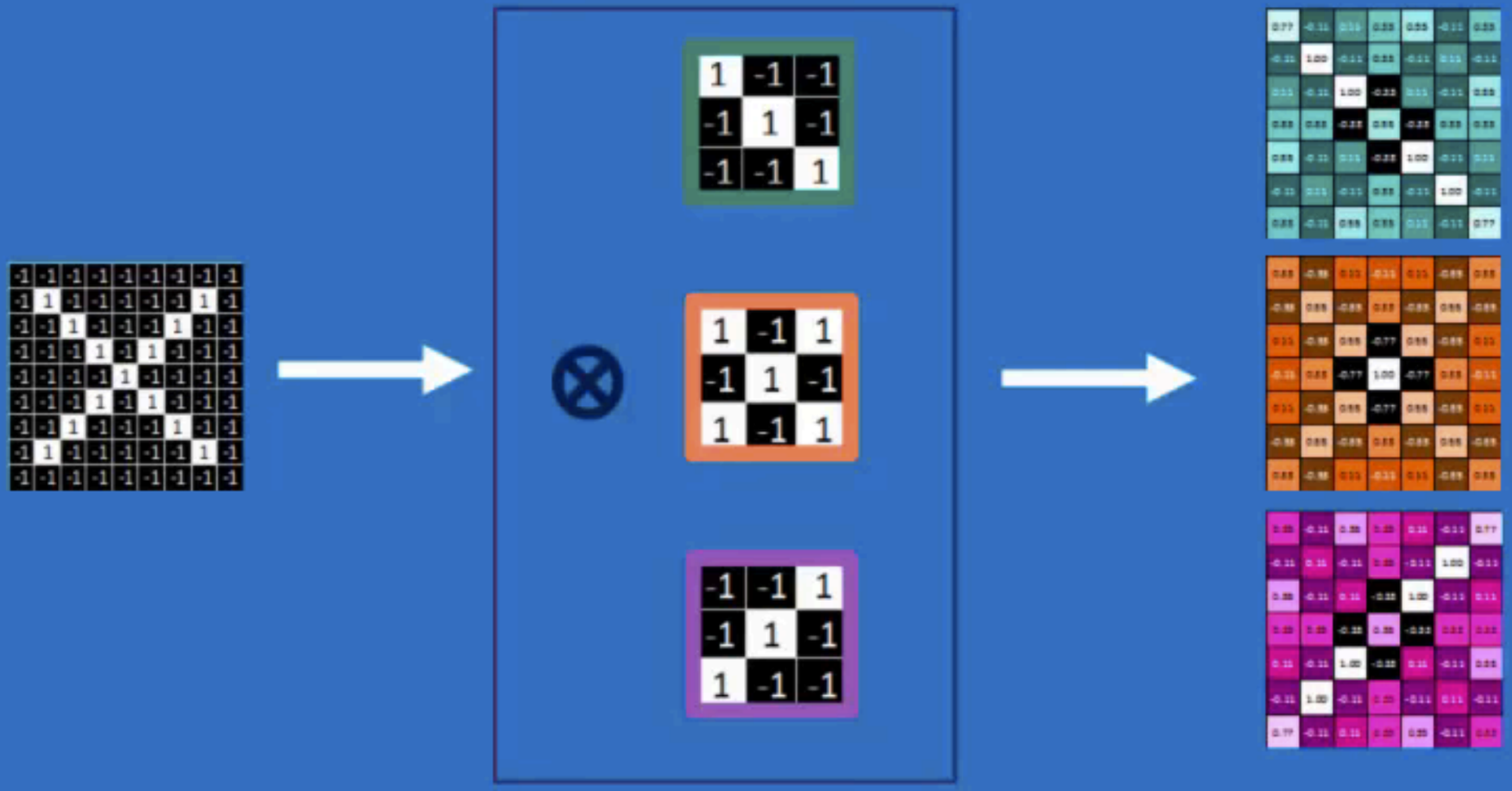
-1	-1	1
-1	1	-1
1	-1	-1

=

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

Convolution layer

One image becomes a stack of filtered images



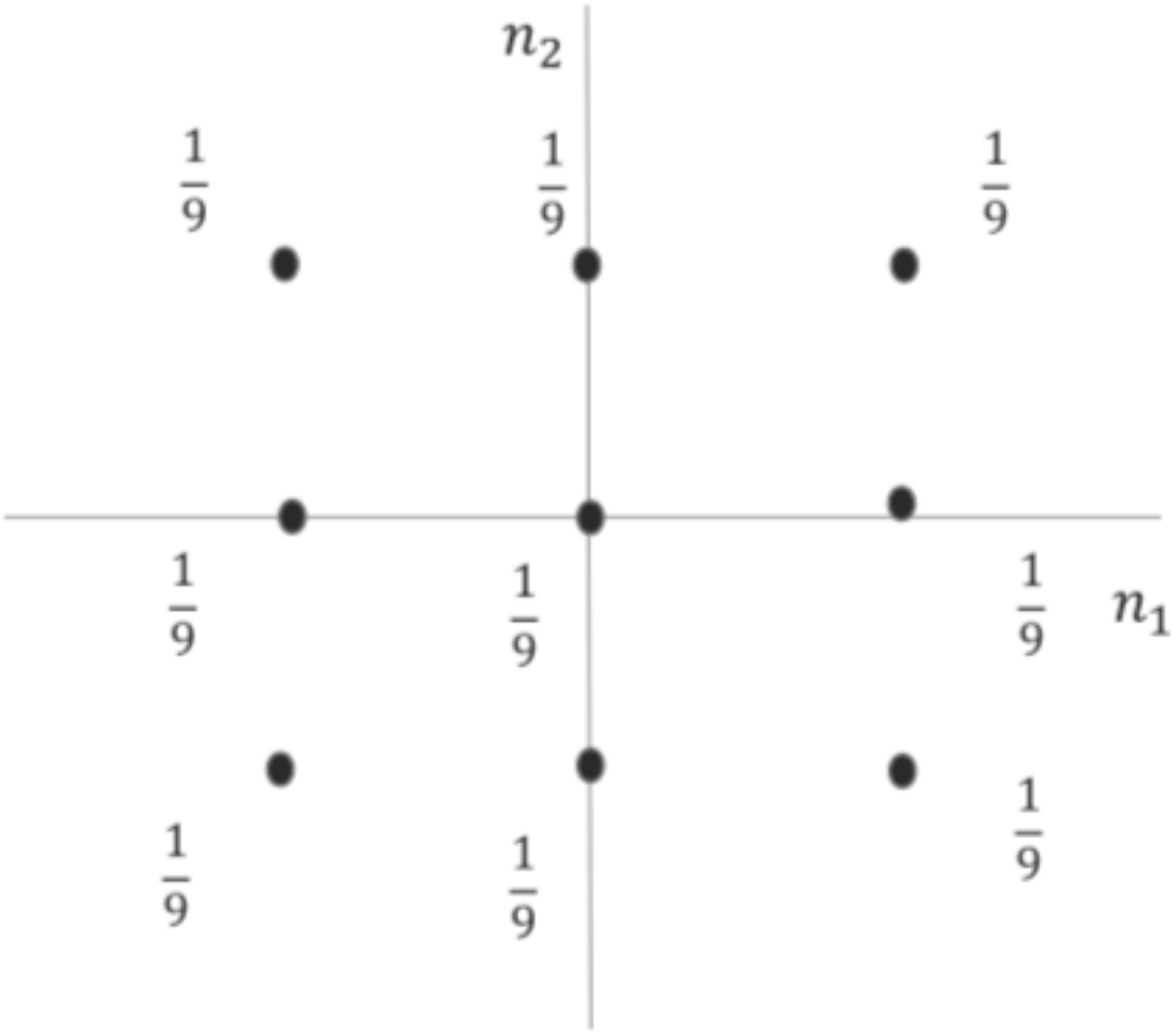
Common Image-Processing Filters

Let's discuss image-processing filters commonly used on 2D images. Make sure to be clear with notations since the natural way of indexing an image doesn't align well with how one would prefer to define the *x* and *y* axes. Whenever we represent an image-processing filter or an image in the coordinate space, n_1 and n_2 are the discrete coordinates for the *x* and *y* directions. The column index of the image in numpy matrix form coincides nicely with the *x* axis, whereas the row index moves in the opposite direction of the *y* axis. Also, it doesn't matter which pixel location one chooses as the origin for the image signal while doing convolution. Based on whether zero padding is used or not, one can handle the edges accordingly. Since the filter kernel is of a smaller size, we generally flip the filter kernel and then slide it over the image and not the other way around.

Mean Filter

The Mean filter or Average filter is a low-pass filter that computes the local average of the pixel intensities at any specific point. The impulse response of the Mean filter can be any of the form seen here (see Figure 3-12):

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$


n_2
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
$\frac{1}{9}$
n_1

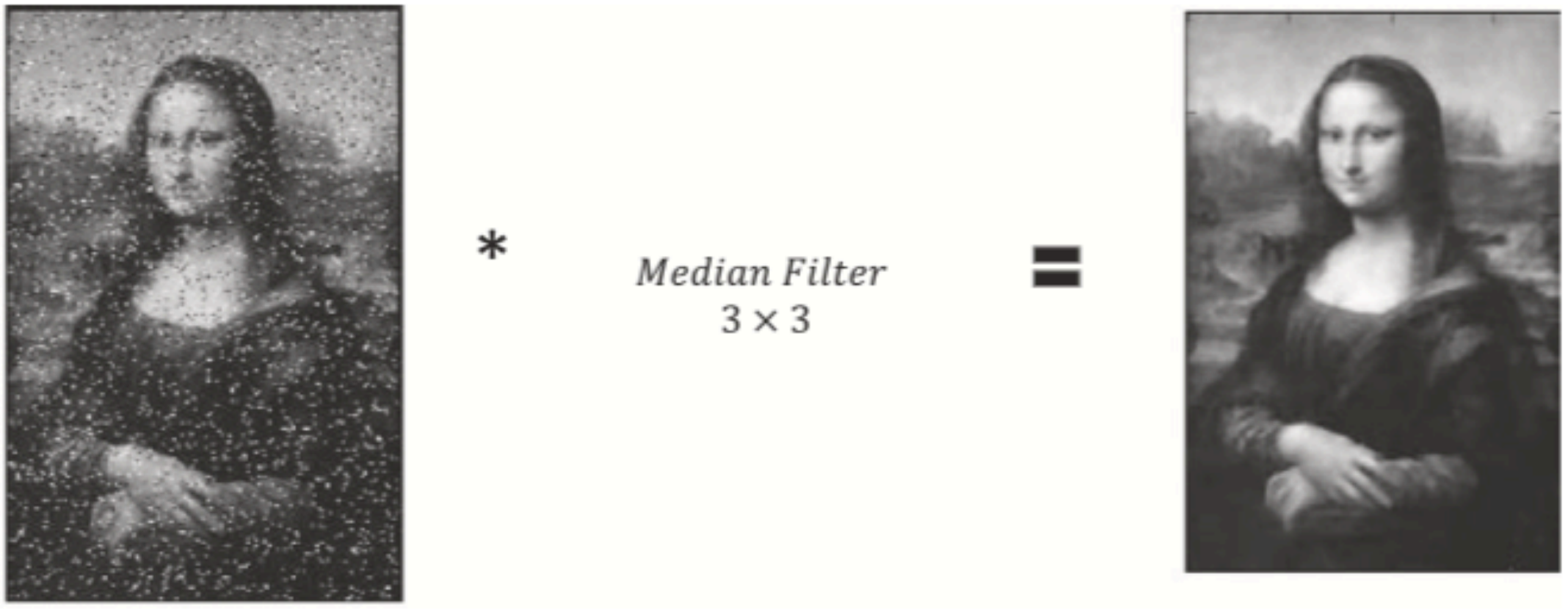


Figure 3-14. *Median filter processing*

As we can see, the salt and pepper noise has been removed by the Median filter.

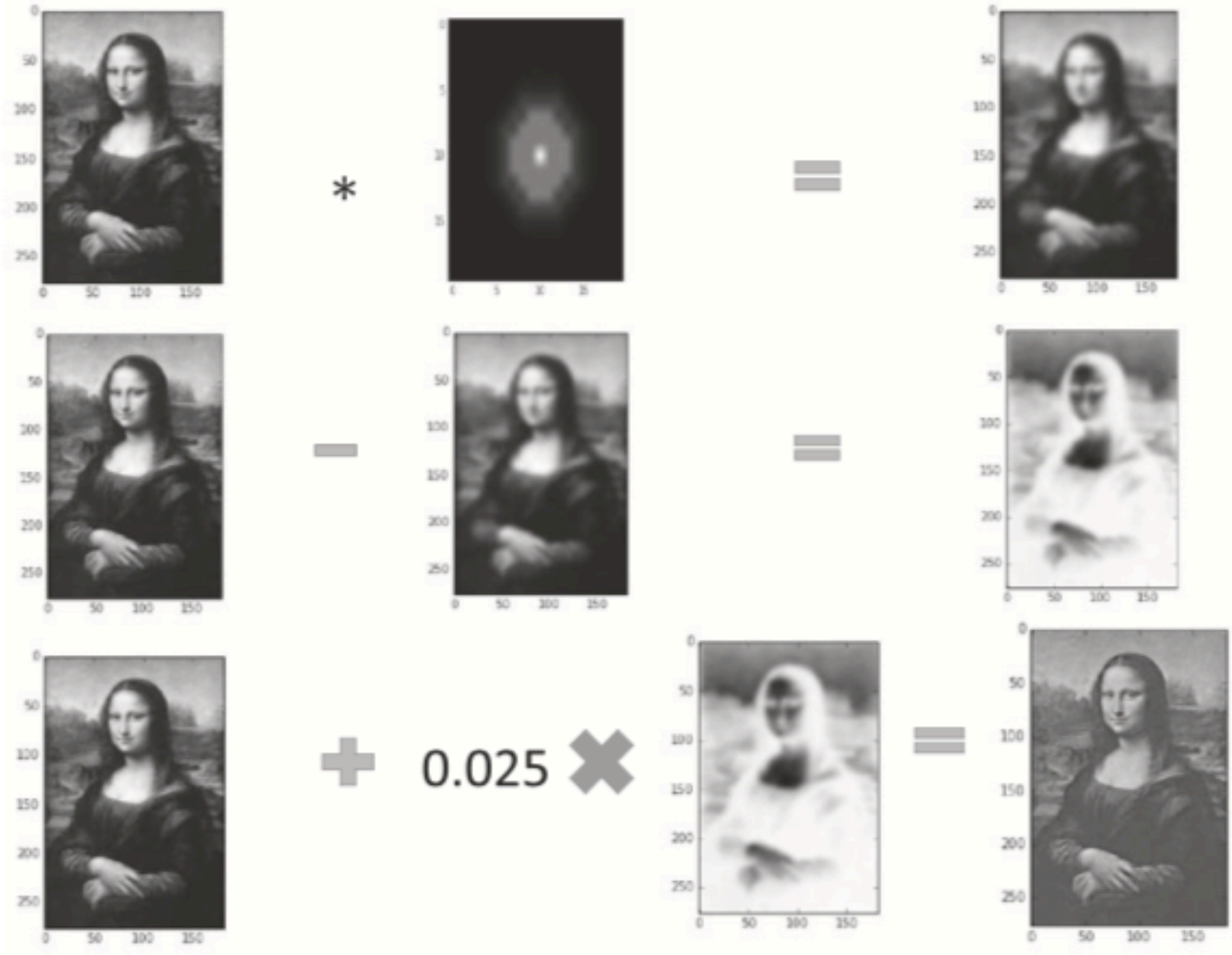


Figure 3-15. *Various activities with Gaussian filter kernel*

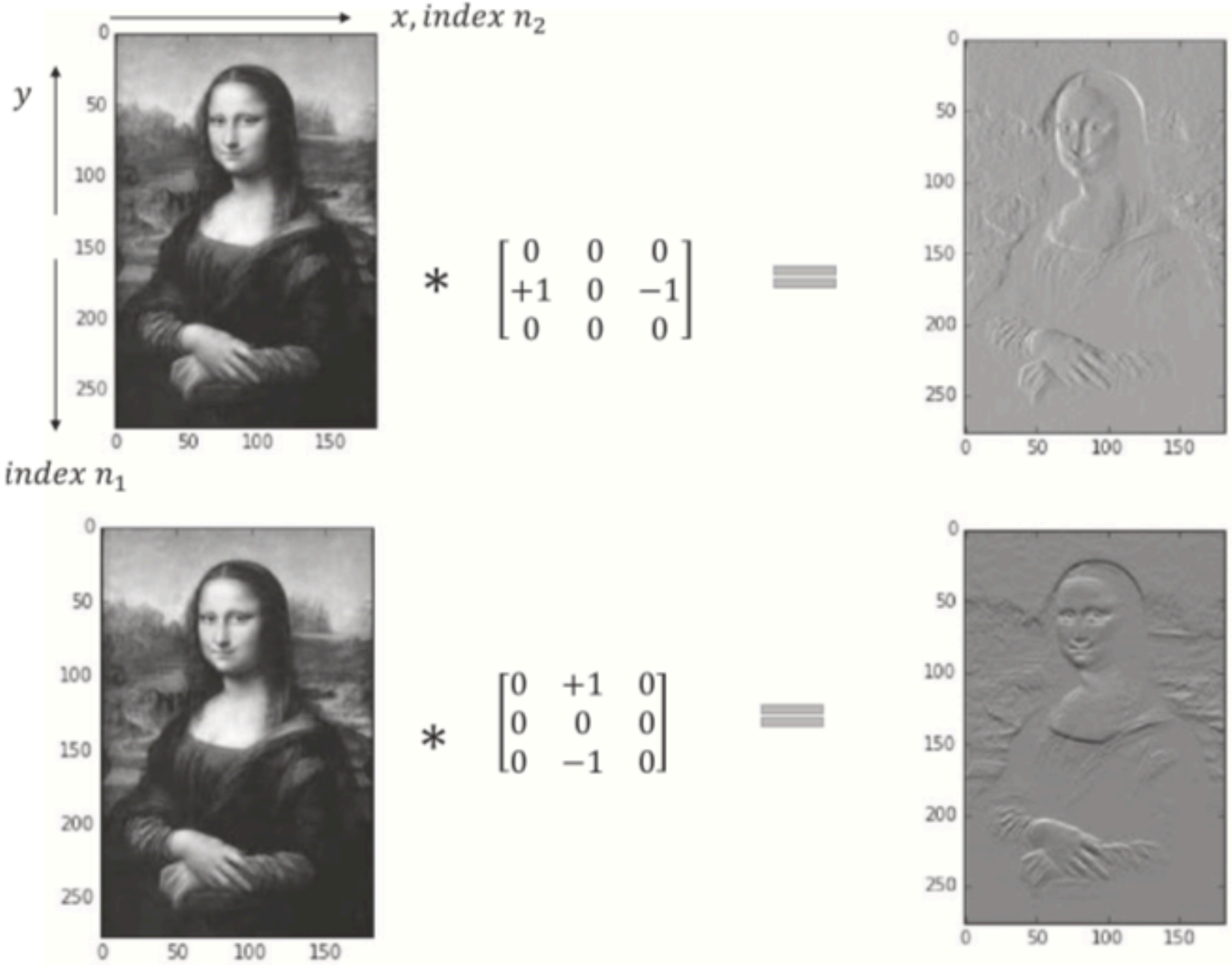
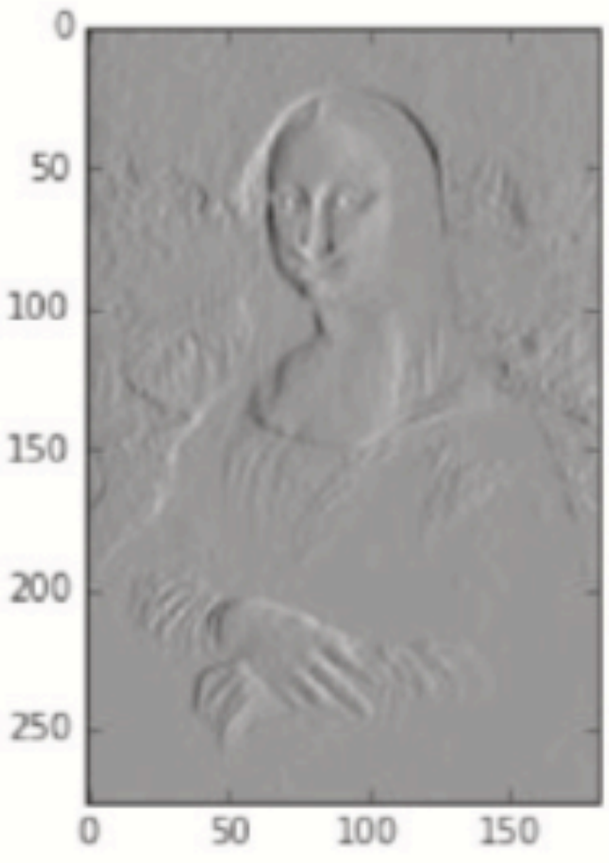
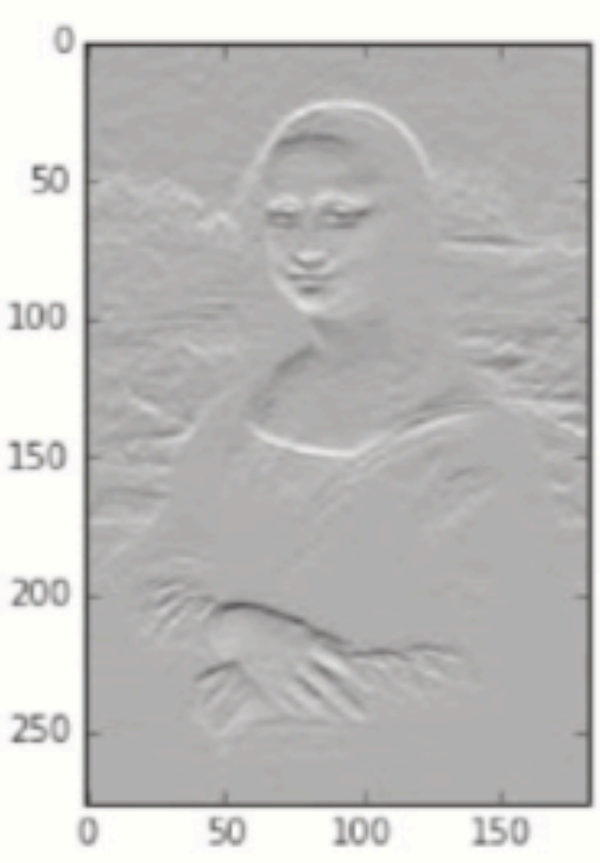


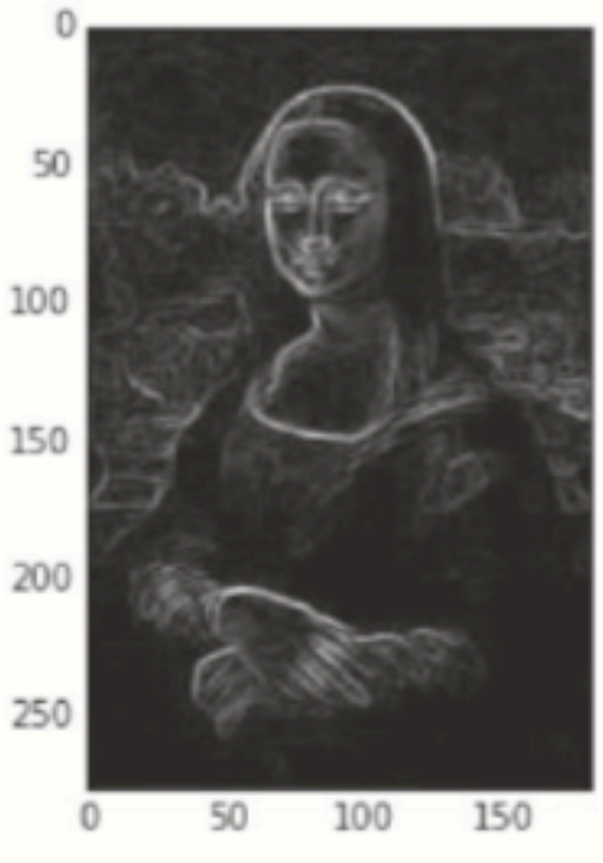
Figure 3-16. *Vertical and horizontal gradient filters*



Output of Convolution with Horizontal Sobel Filter



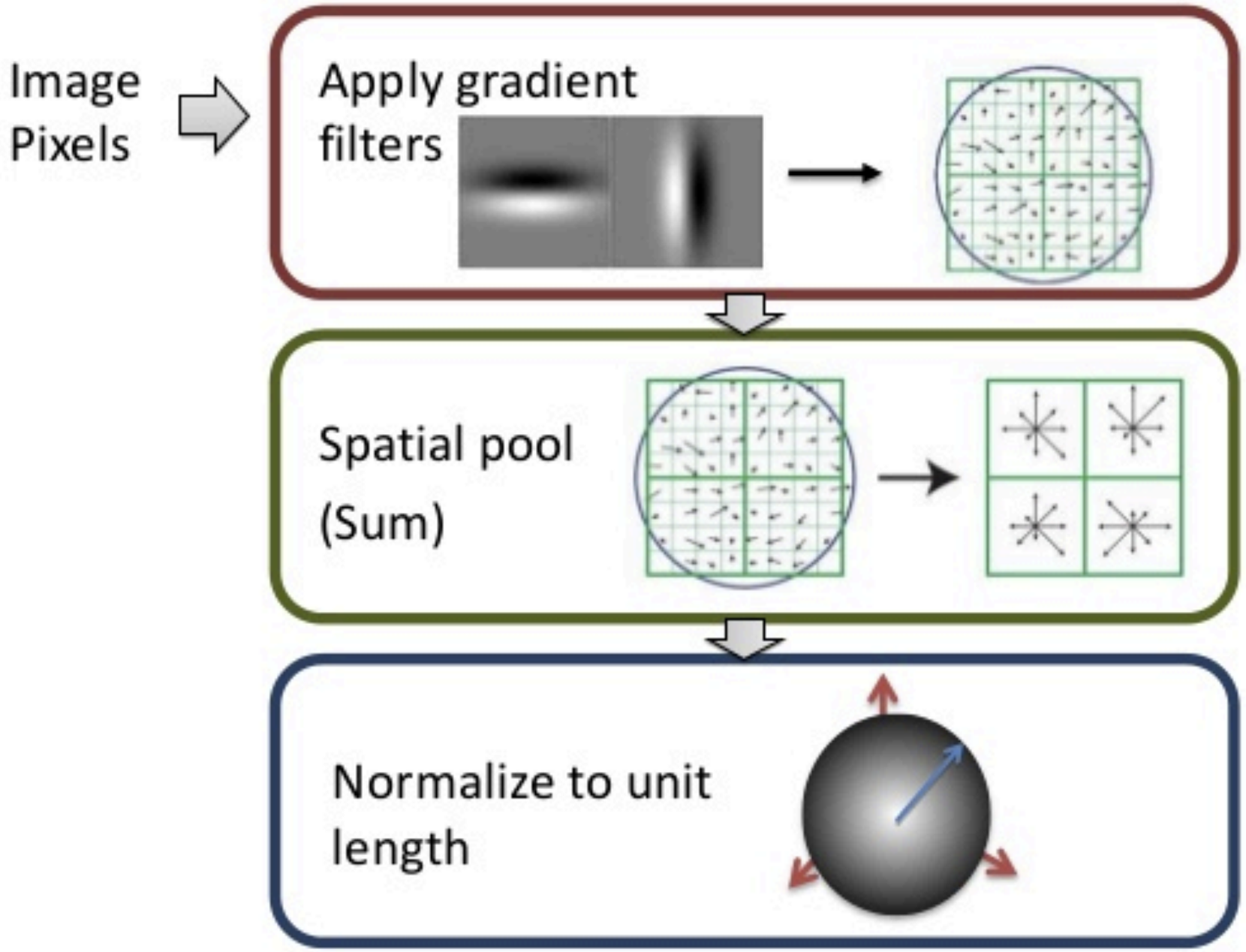
Output of Convolution with Vertical Sobel Filter



Output from Combined Sobel Filter

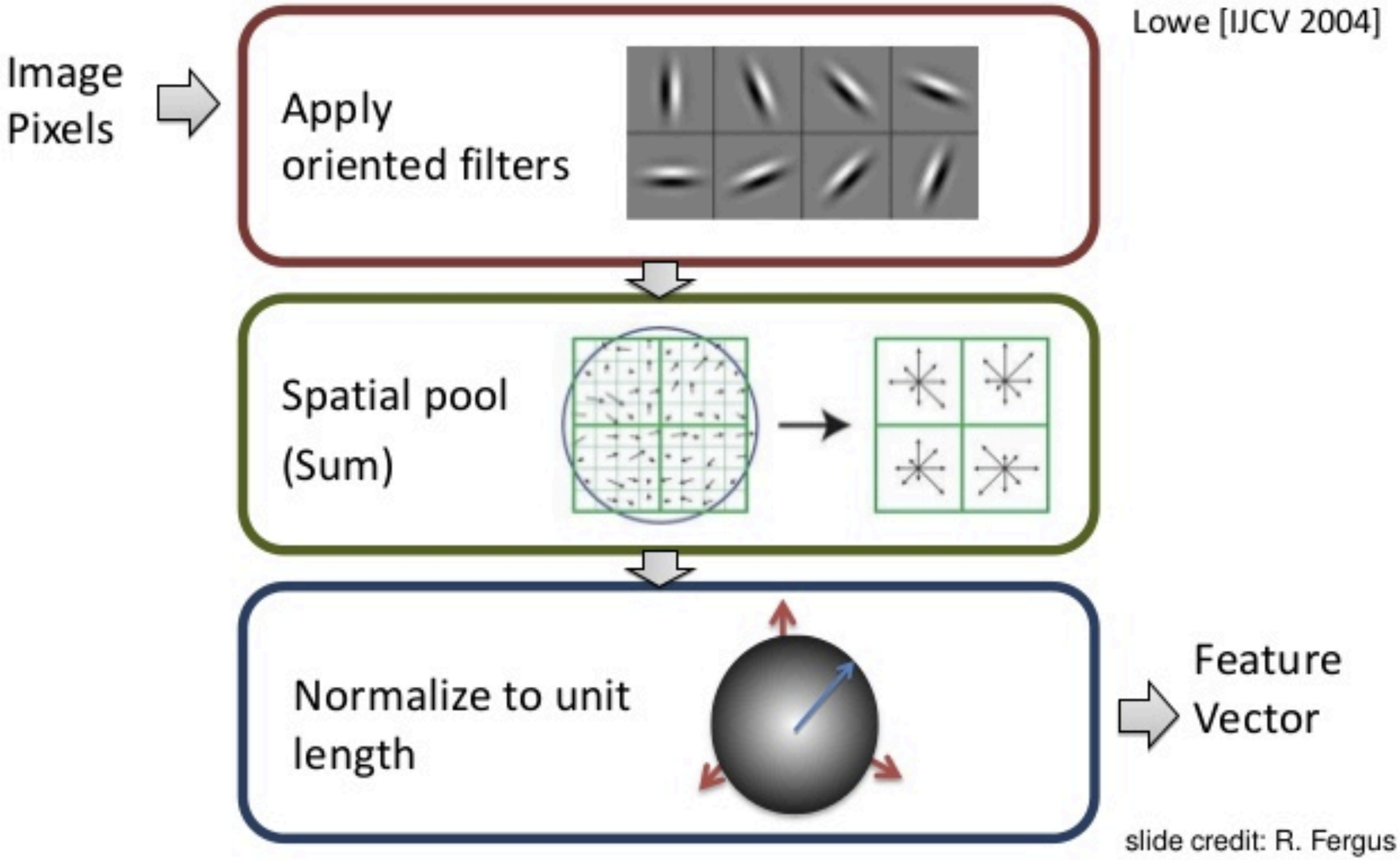
Figure 3-18. *Output of various Sobel filters*

SIFT Descriptor



Lowe [IJCV 2004]

SIFT Descriptor



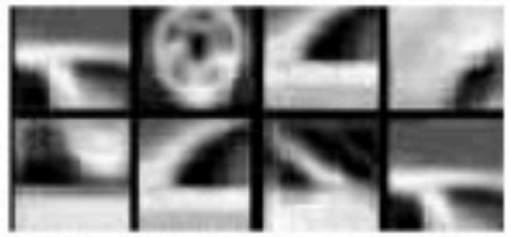
Lowe [IJCV 2004]

slide credit: R. Fergus

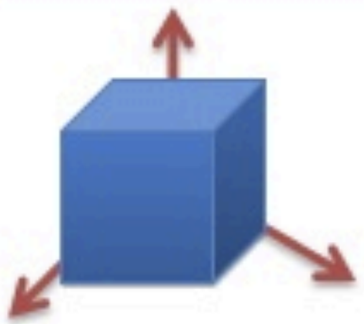
Spatial Pyramid Matching

SIFT Features

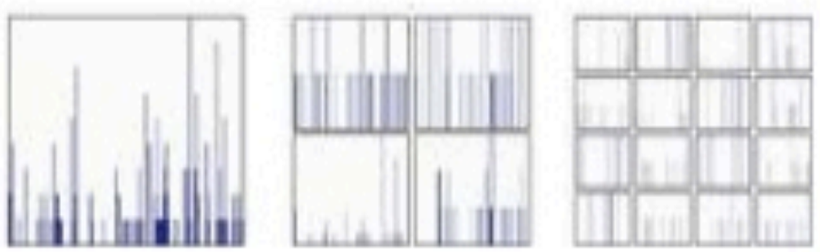
Filter with Visual Words



Max



Multi-scale spatial pool (Sum)



Classifier

Lazebnik, Schmid, Ponce [CVPR 2006]

slide credit: R. Fergus

Pooling: Shrinking the image stack

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling

maximum

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

1.00			

Pooling

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

max pooling

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

Pooling

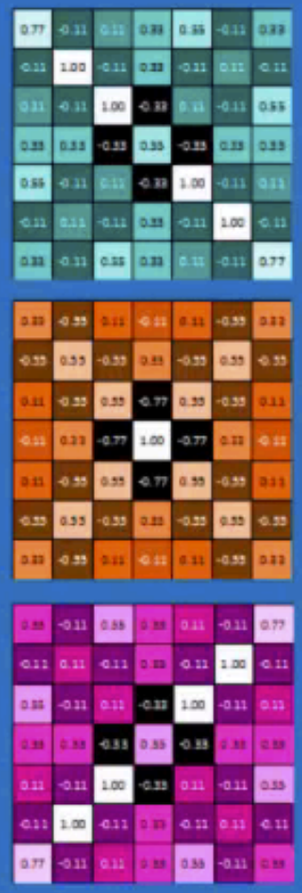
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

max pooling

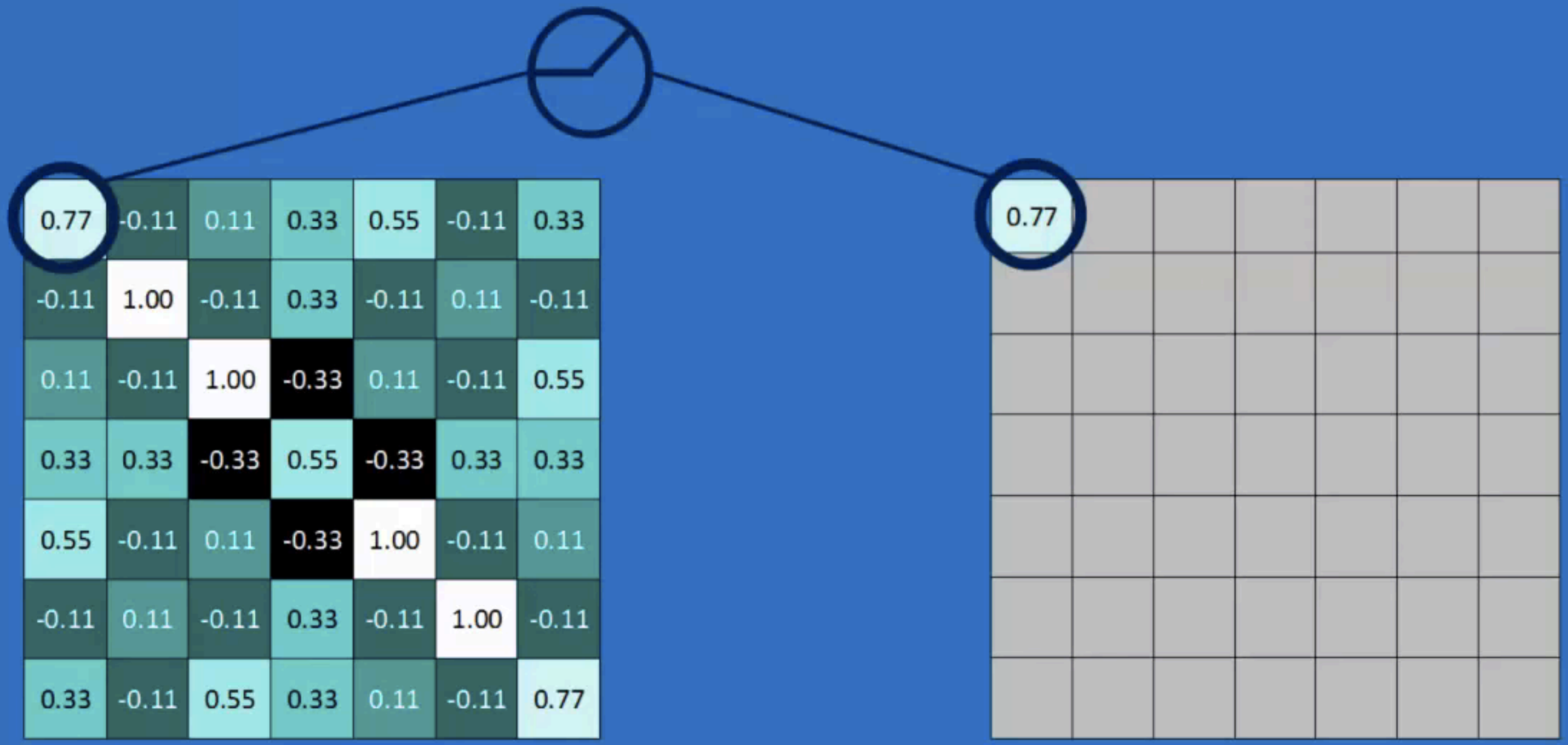
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Pooling layer

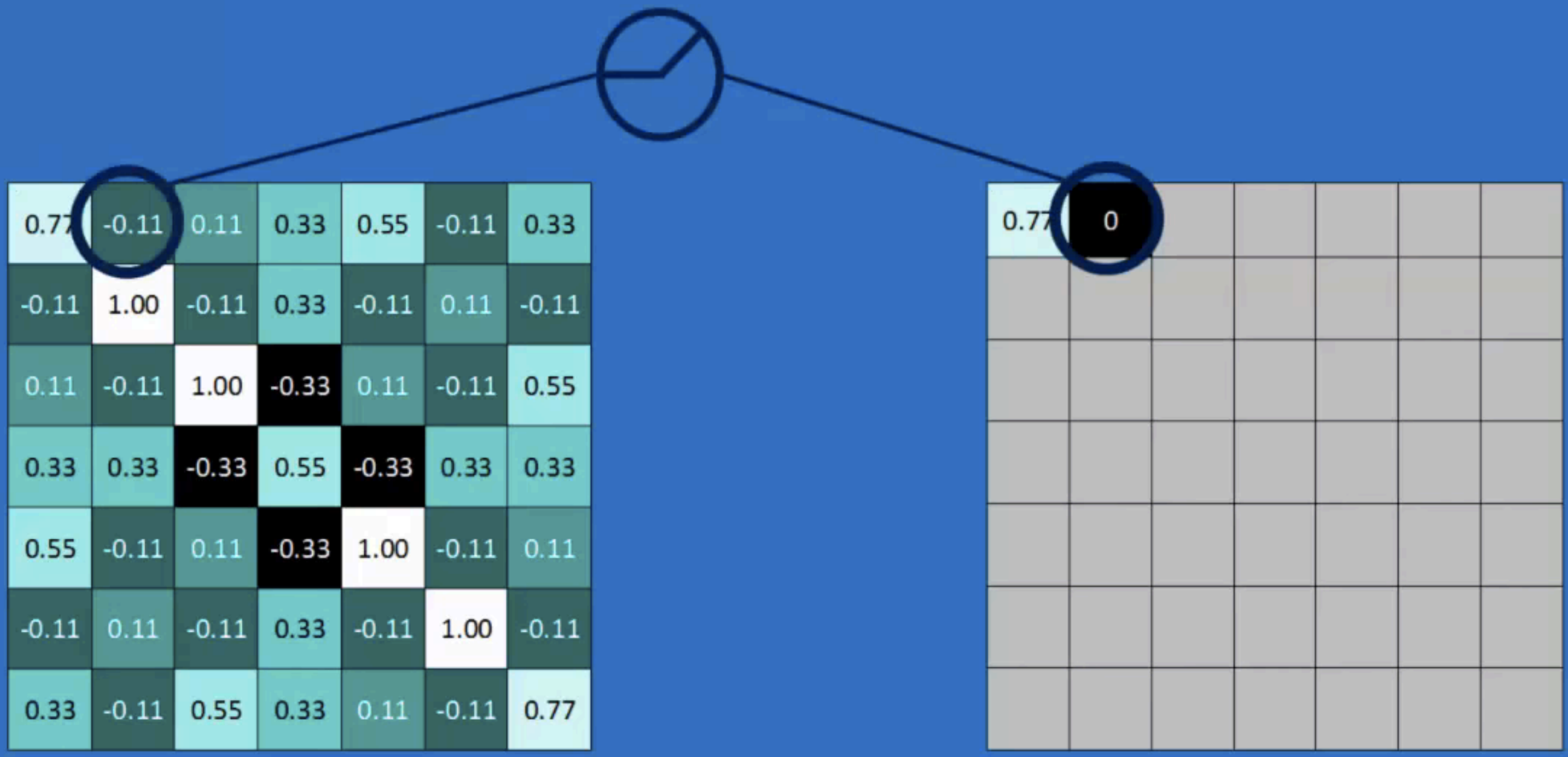
A stack of images becomes a stack of smaller images.



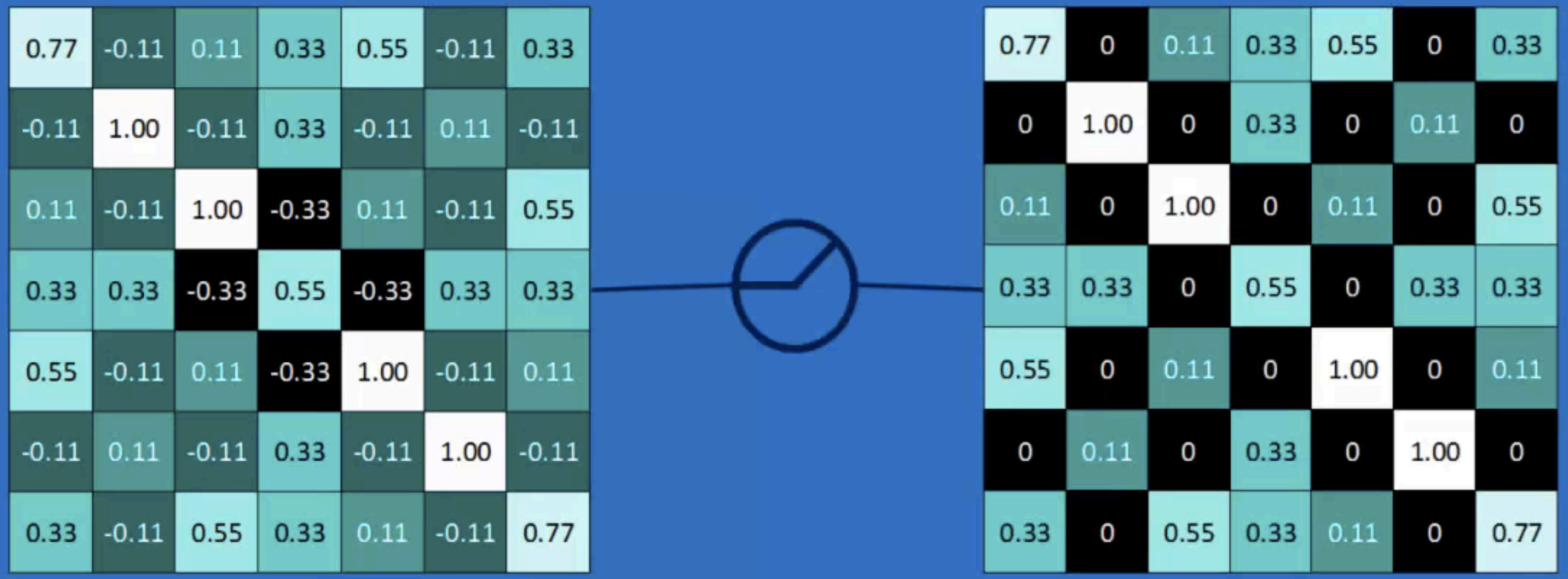
Rectified Linear Units (ReLUs)



Rectified Linear Units (ReLUs)

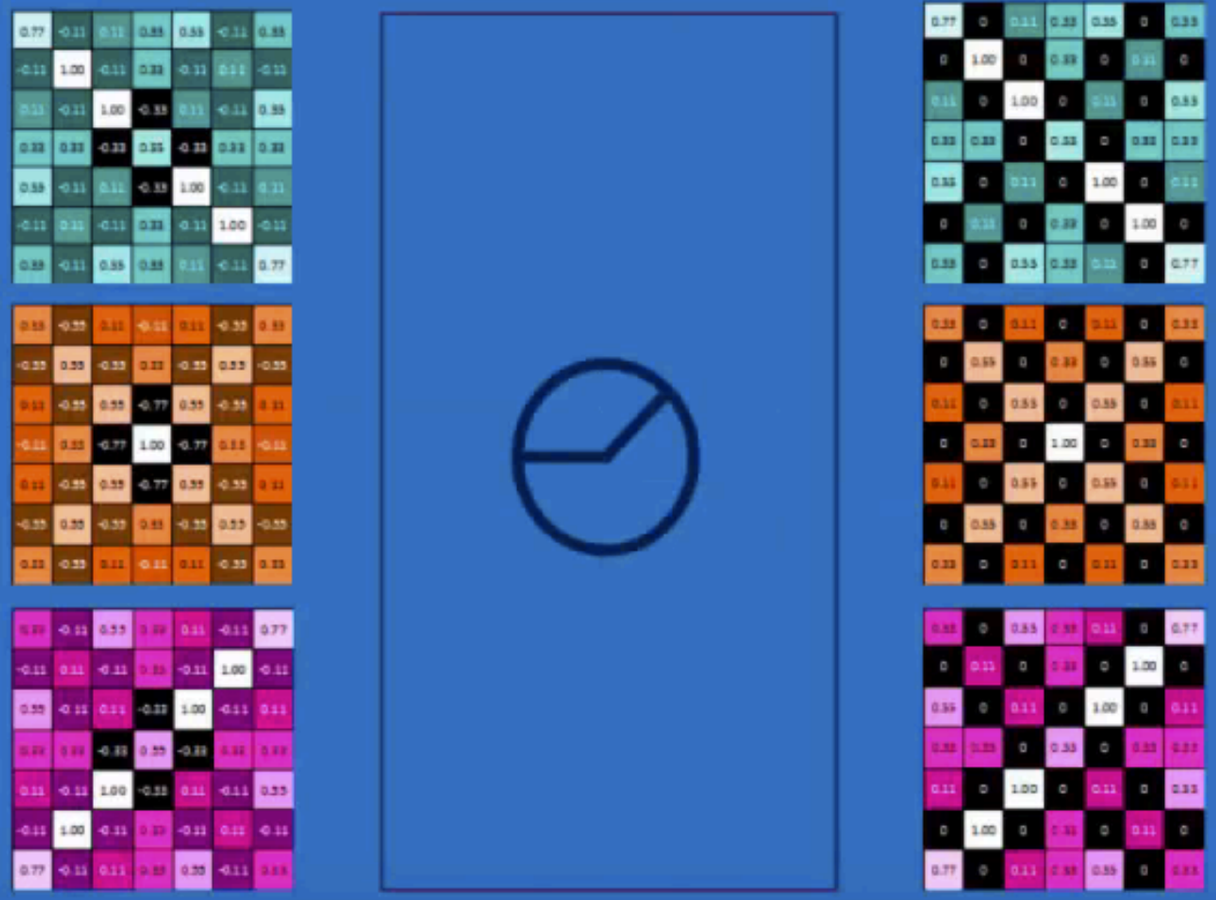


Rectified Linear Units (ReLUs)



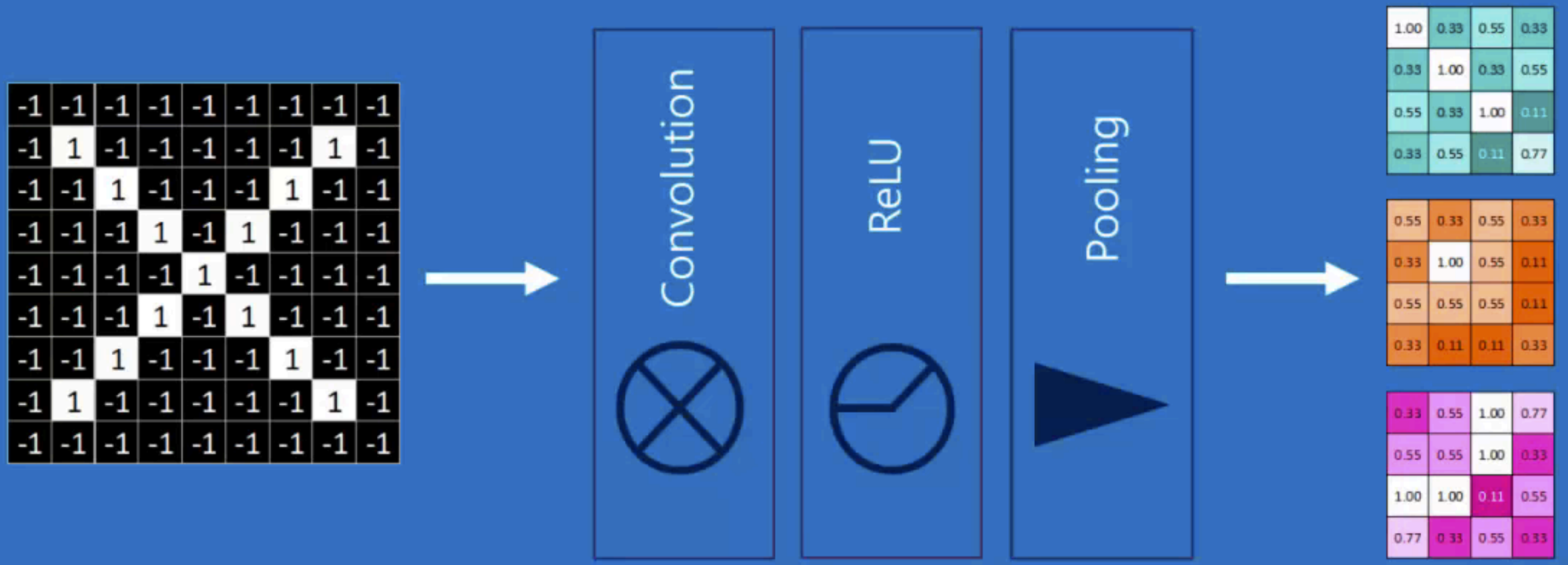
ReLU layer

A stack of images becomes a stack of images with no negative values.



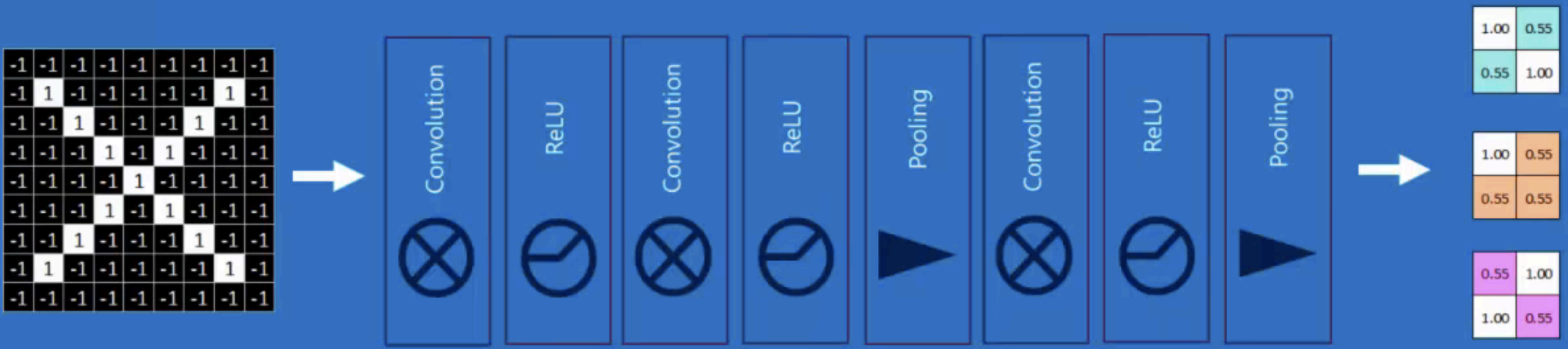
Layers get stacked

The output of one becomes the input of the next.



Deep stacking

Layers can be repeated several (or many) times.

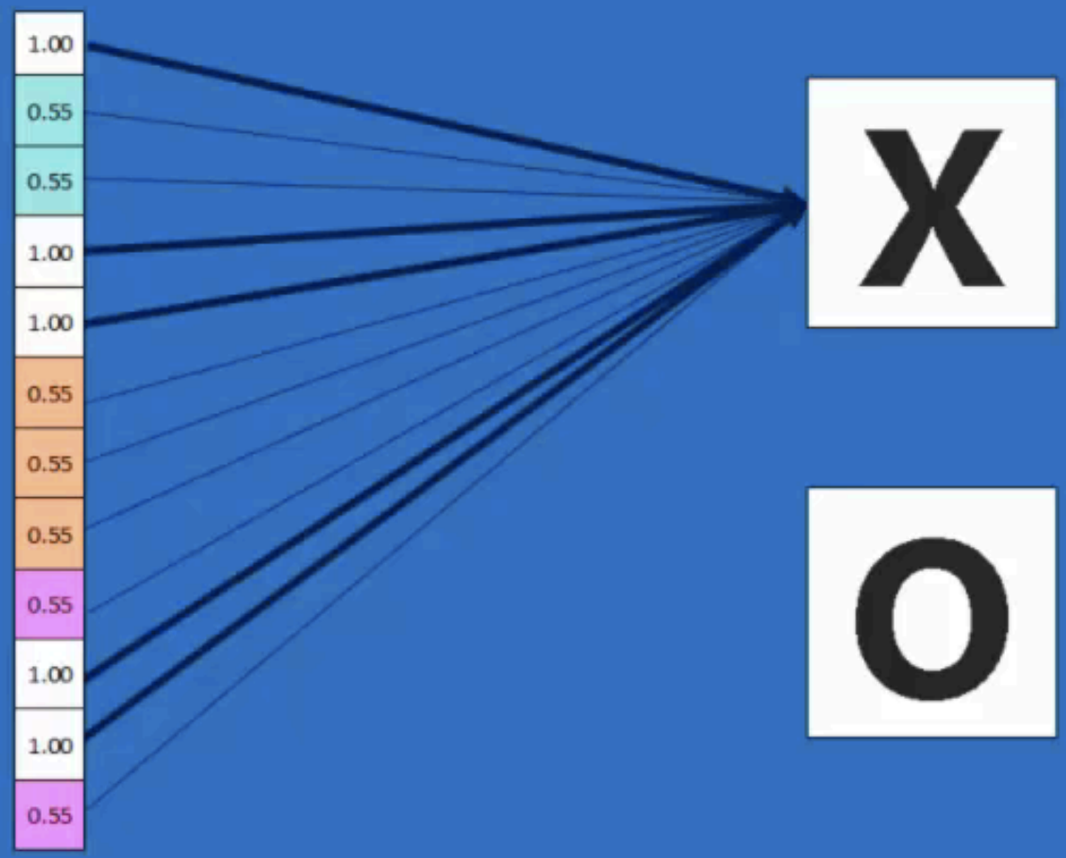


Fully connected layer

Every value gets a vote

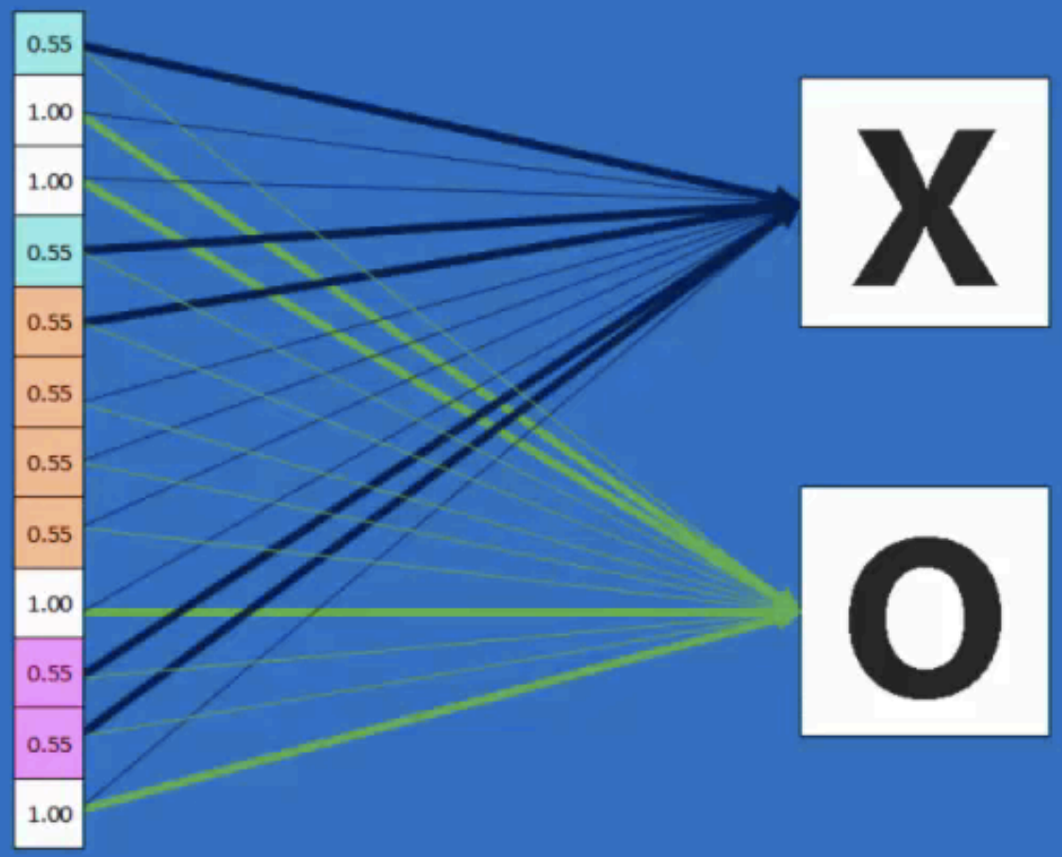
Fully connected layer

Vote depends on how strongly a value predicts X or O



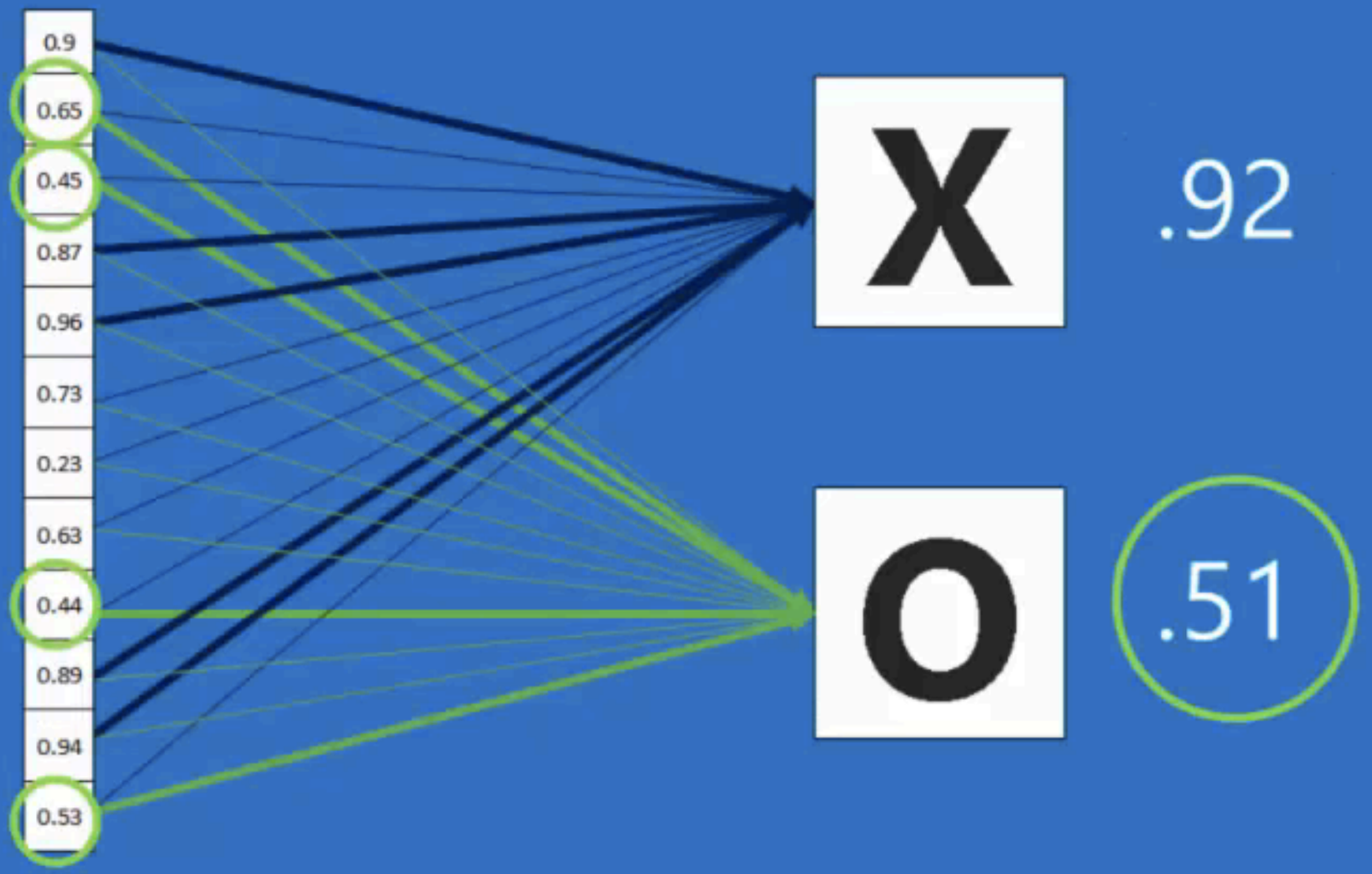
Fully connected layer

Vote depends on how strongly a value predicts X or O



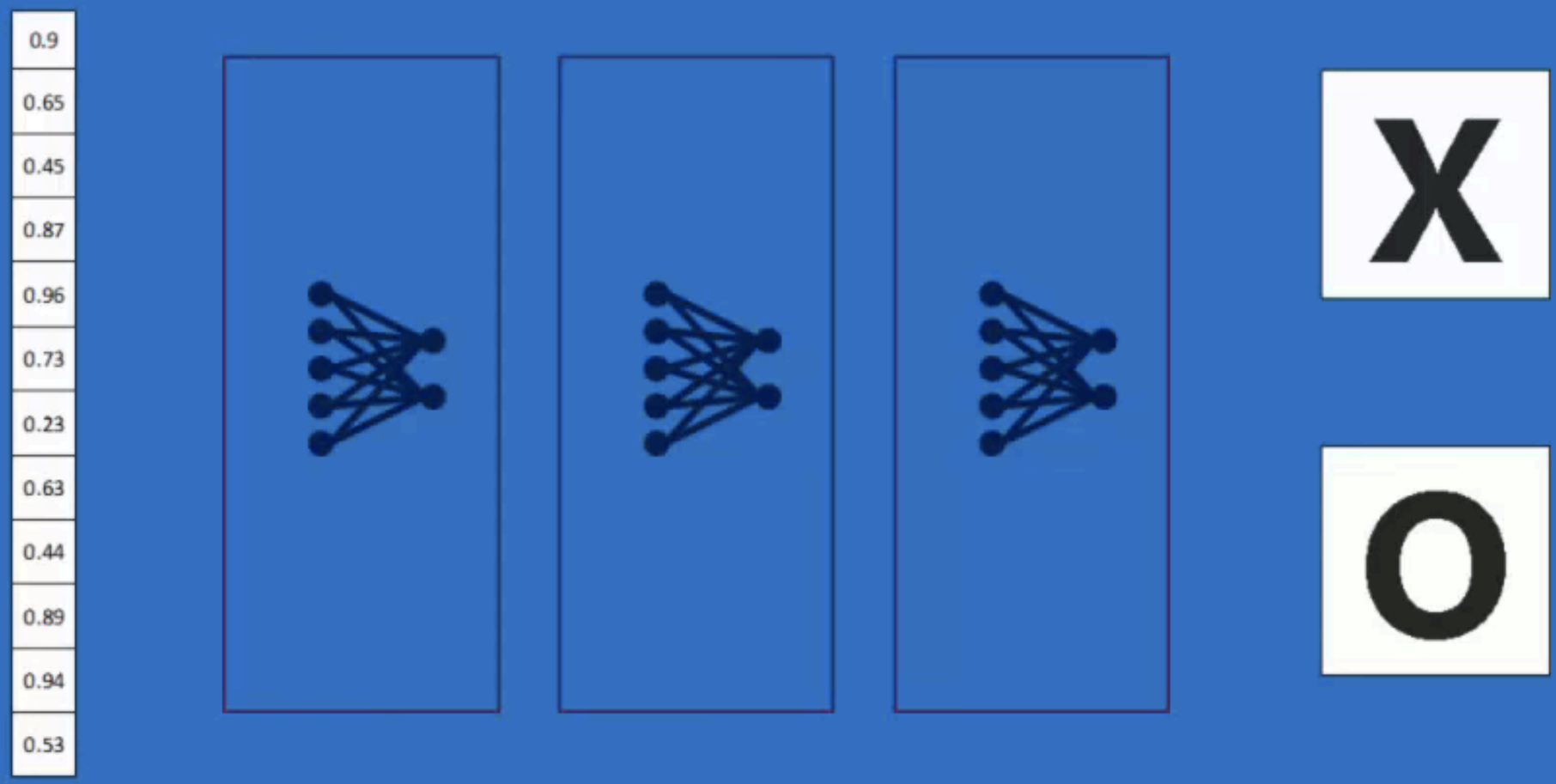
Fully connected layer

Future values vote on X or O



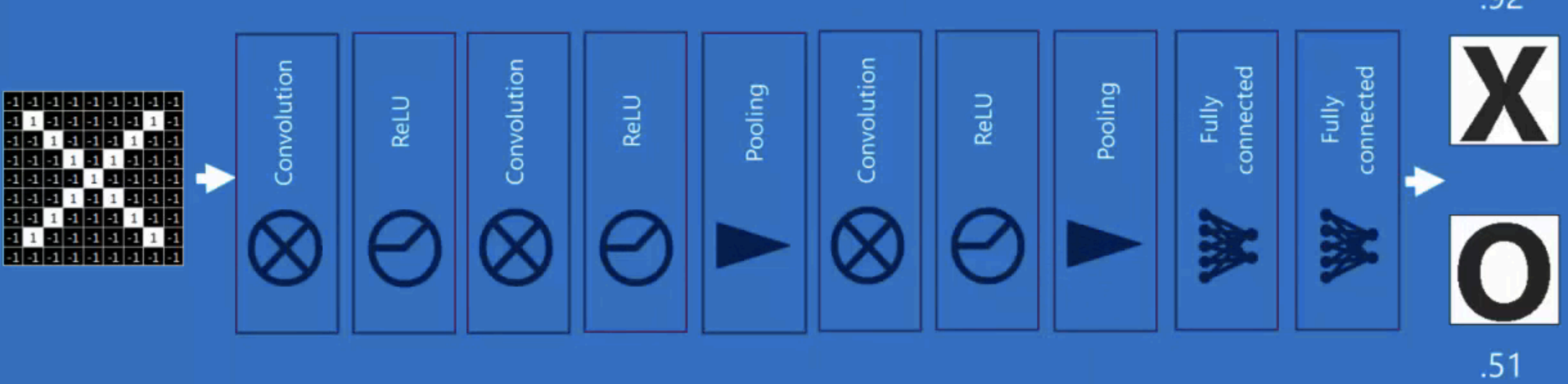
Fully connected layer

These can also be stacked.



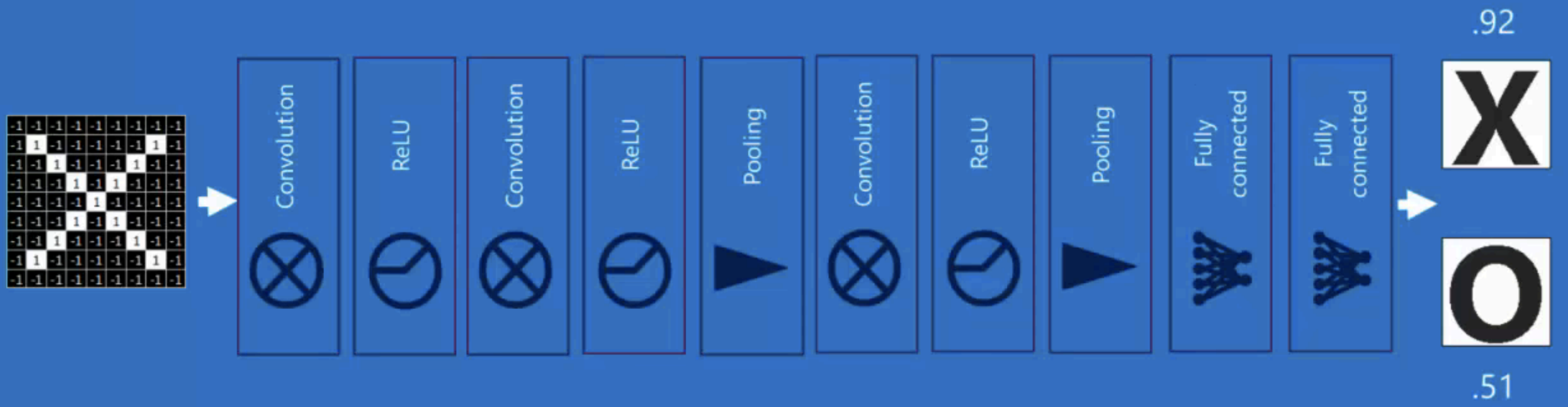
Putting it all together

A set of pixels becomes a set of votes.



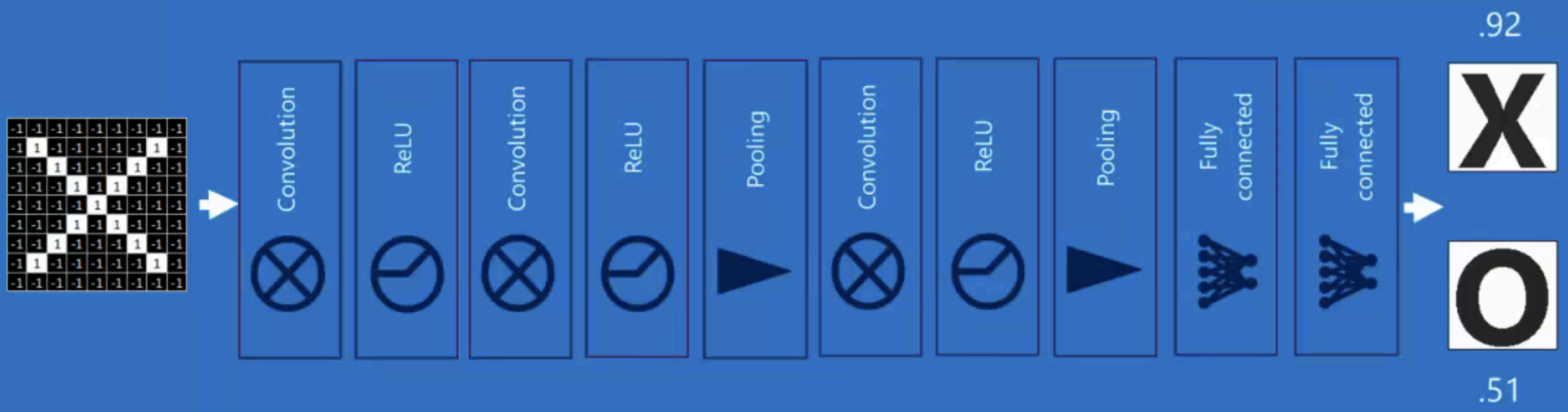
Backprop

Error = right answer – actual answer



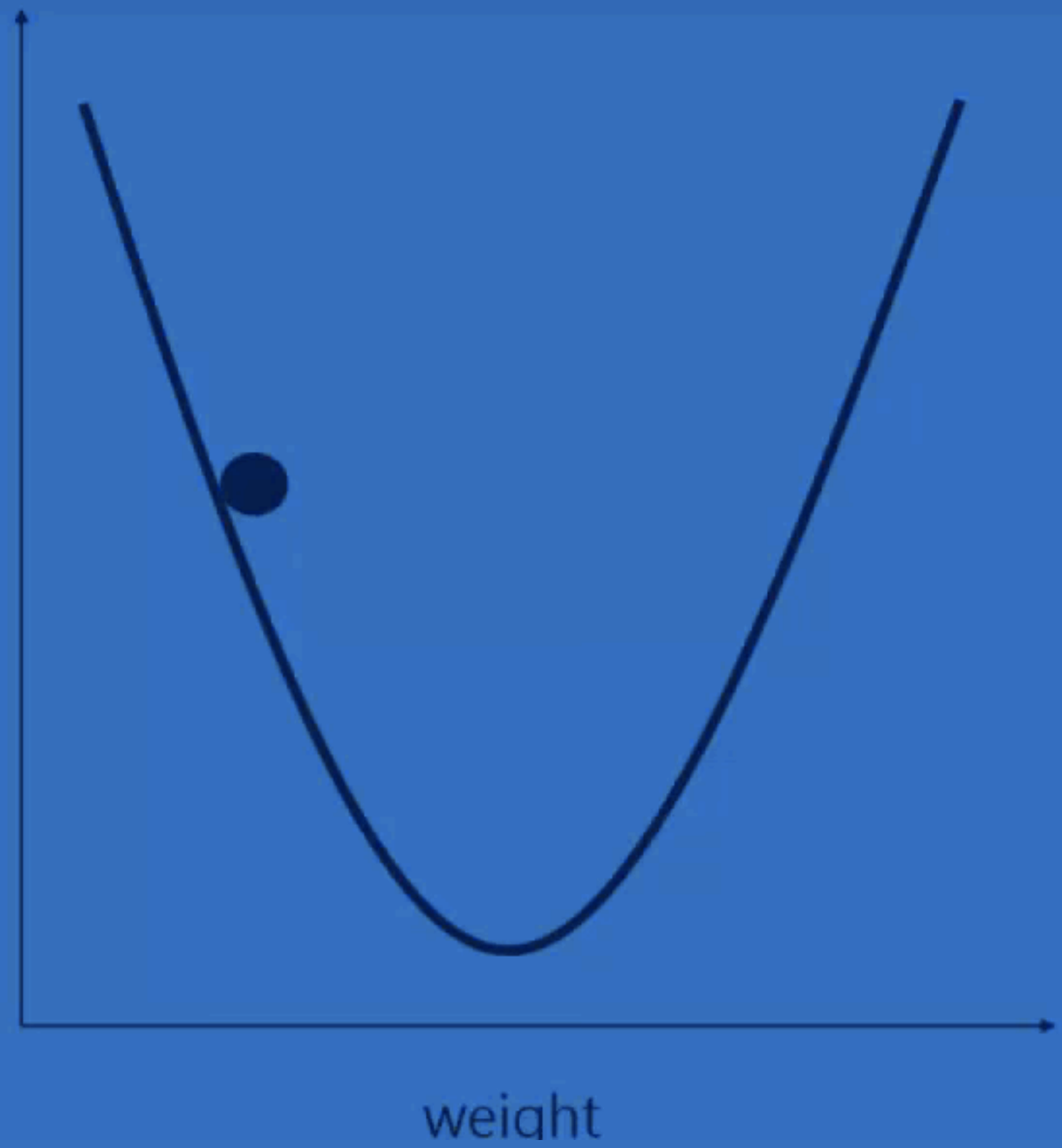
Backprop

	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49



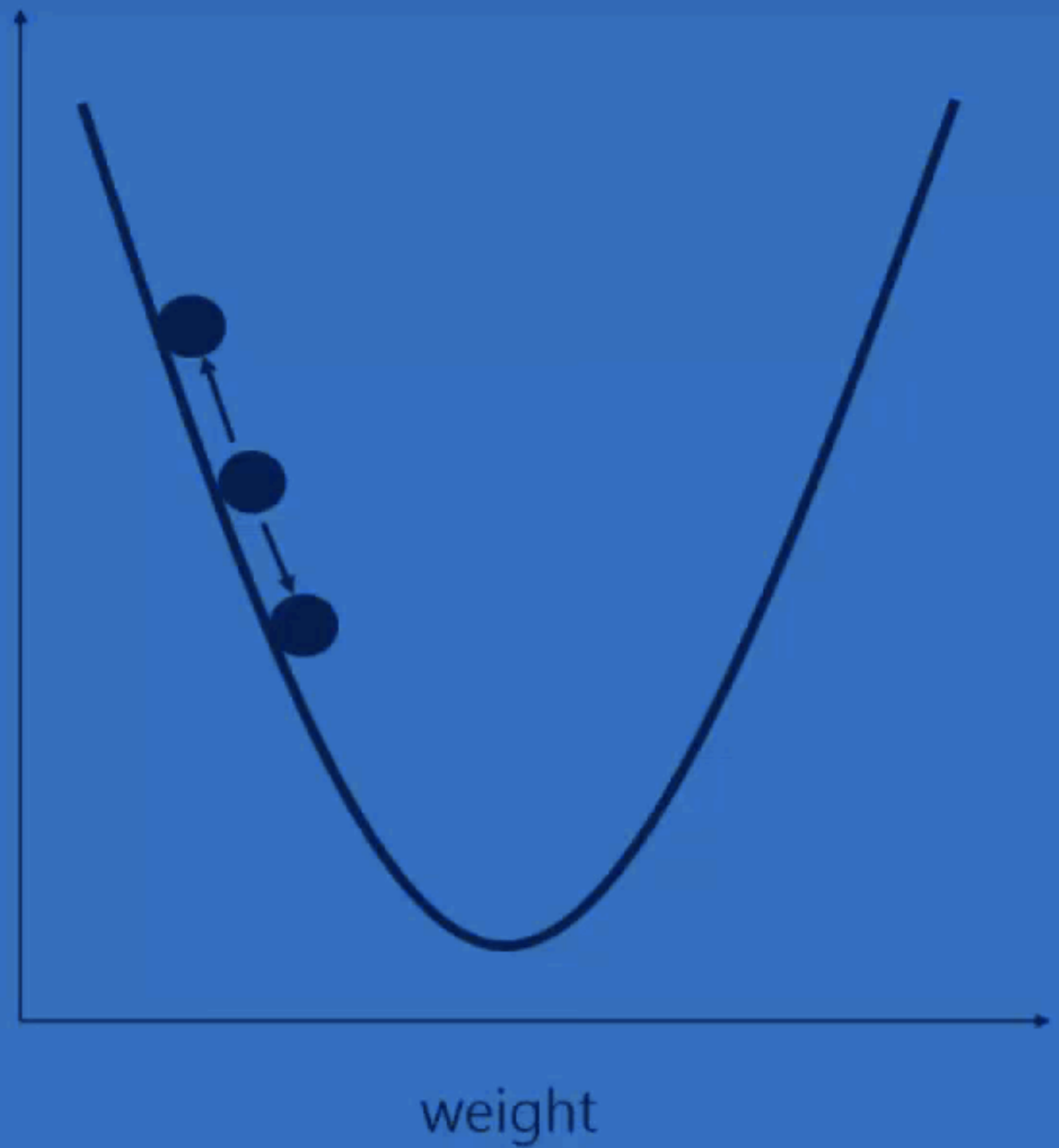
Gradient descent

For each feature pixel and voting weight, adjust it up and down a bit and see how the error changes.



Gradient descent

For each feature pixel and voting weight, adjust it up and down a bit and see how the error changes.



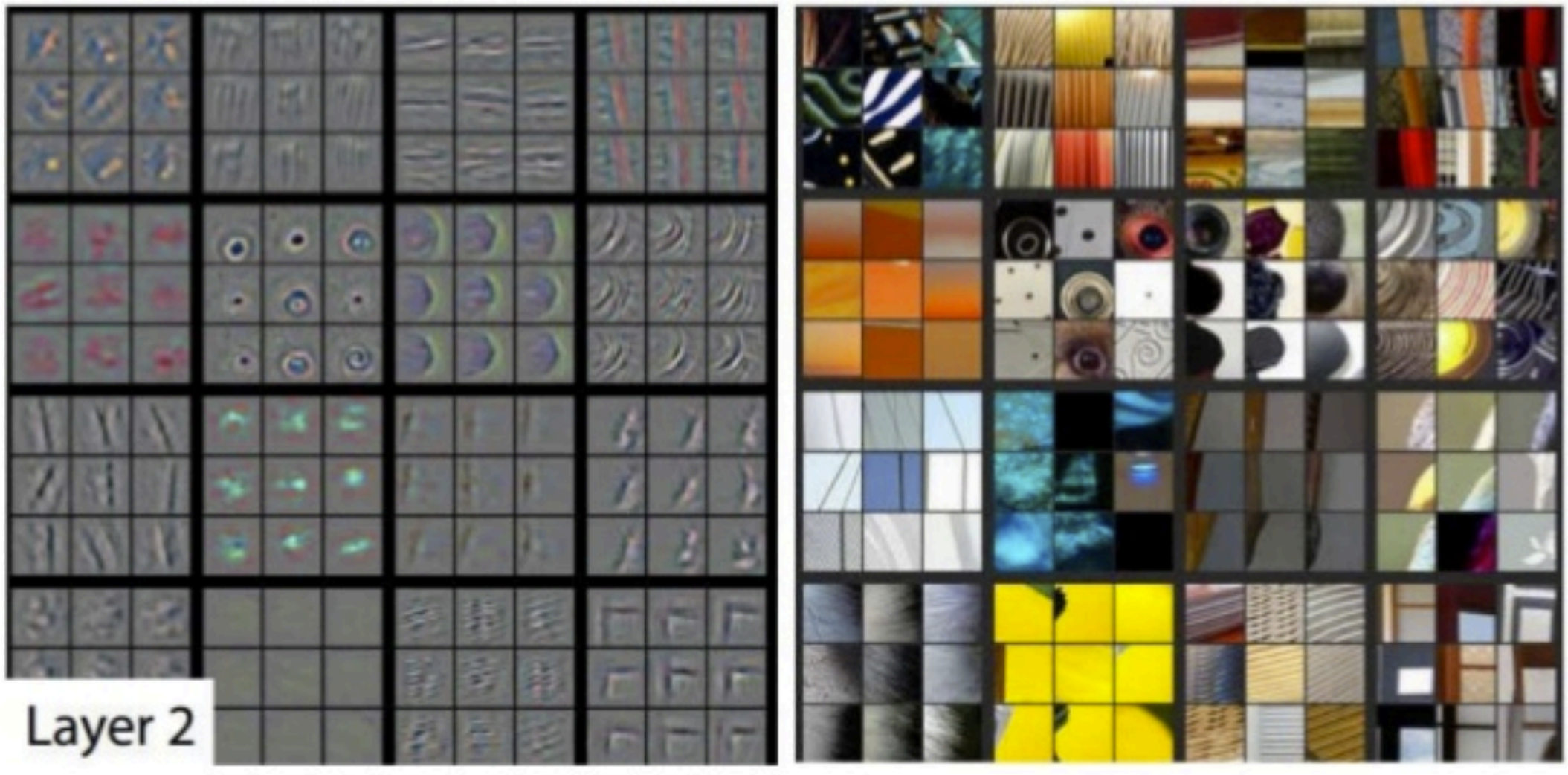
Hyperparameters (knobs)

- Convolution
  - Number of features
  - Size of features
- Pooling
  - Window size
  - Window stride
- Fully Connected
  - Number of neurons

Layer 1

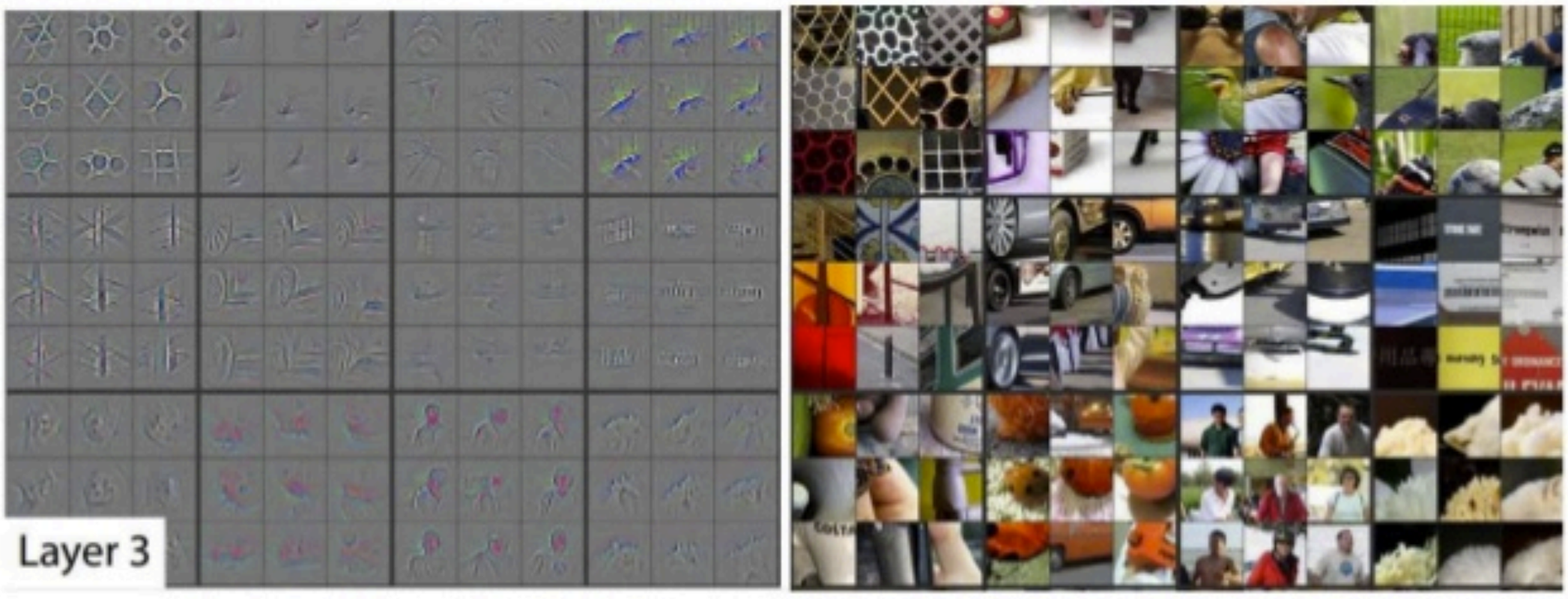
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 2



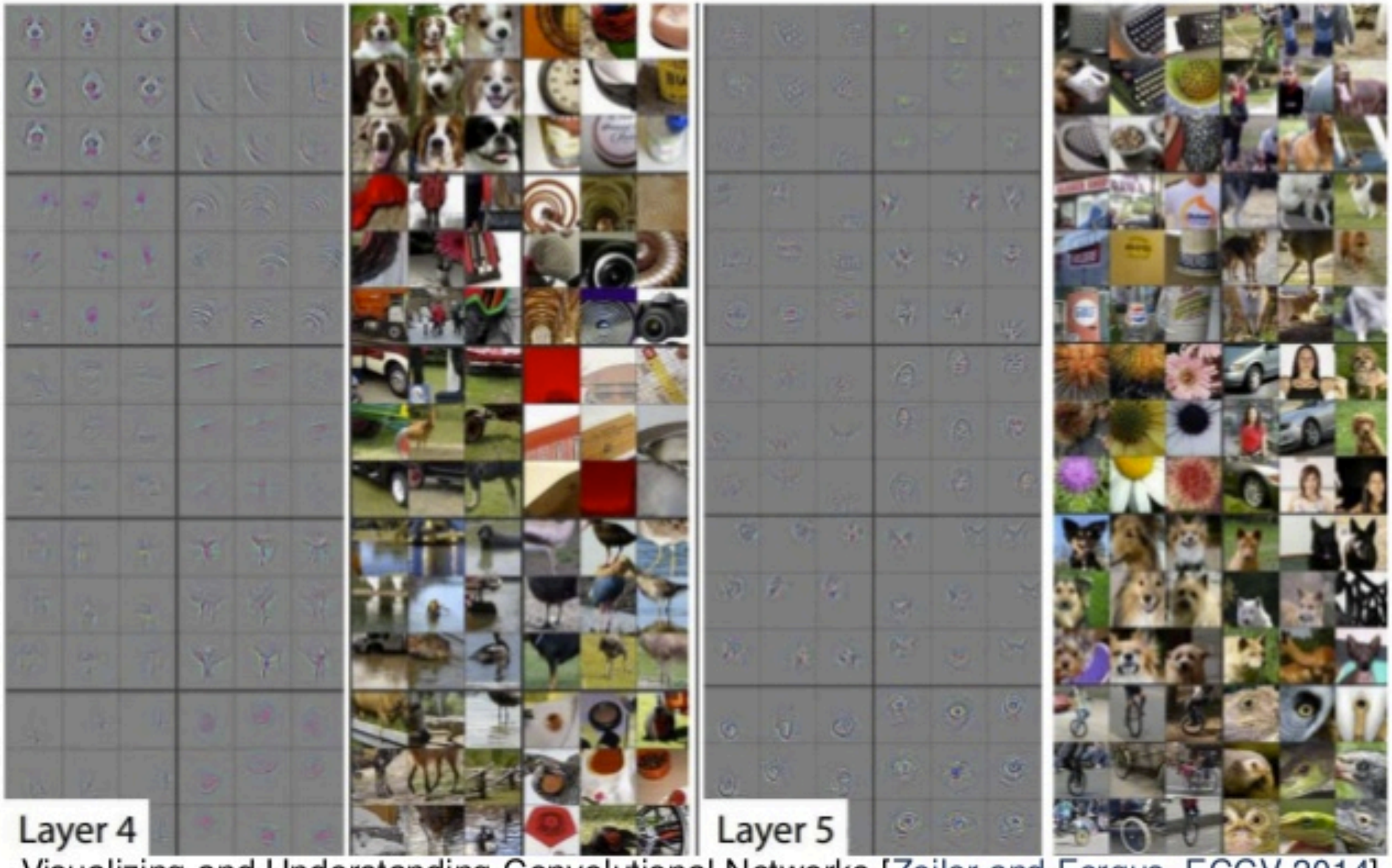
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 3



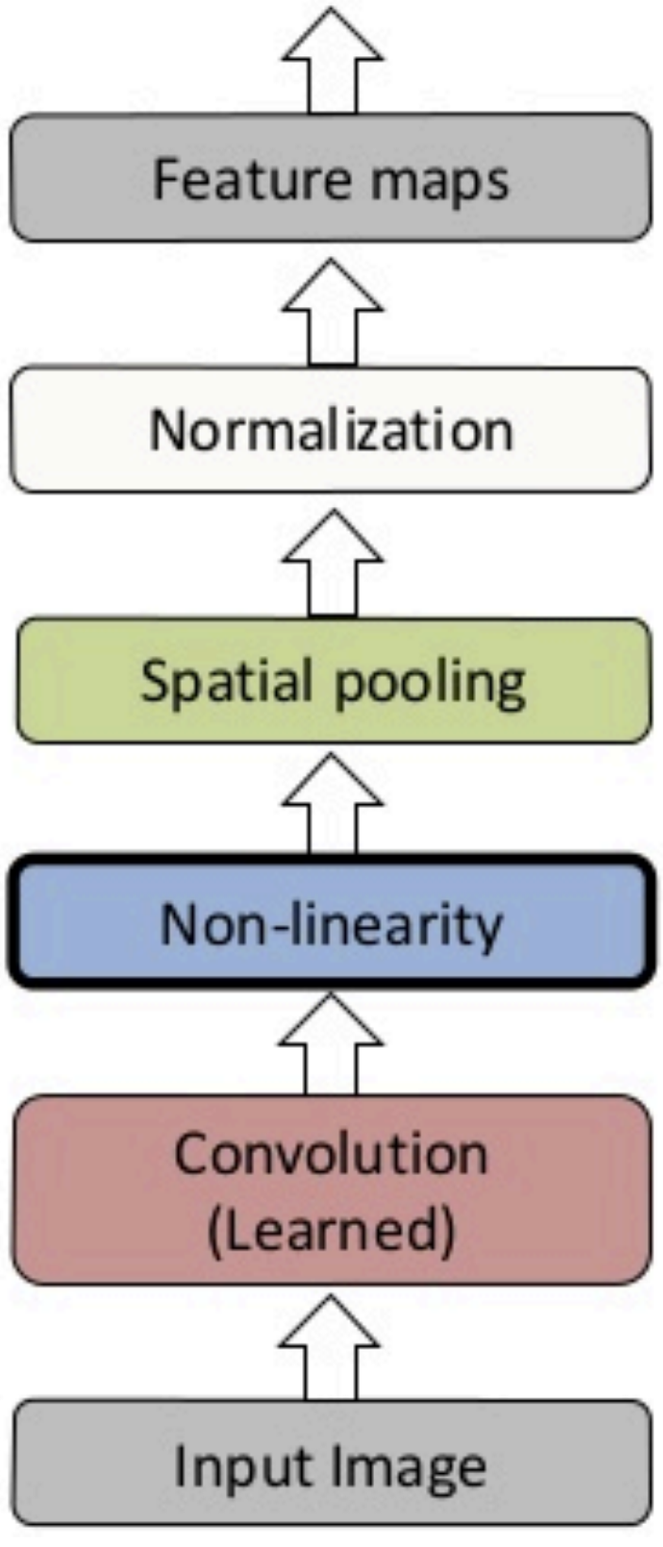
Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Layer 4 and 5

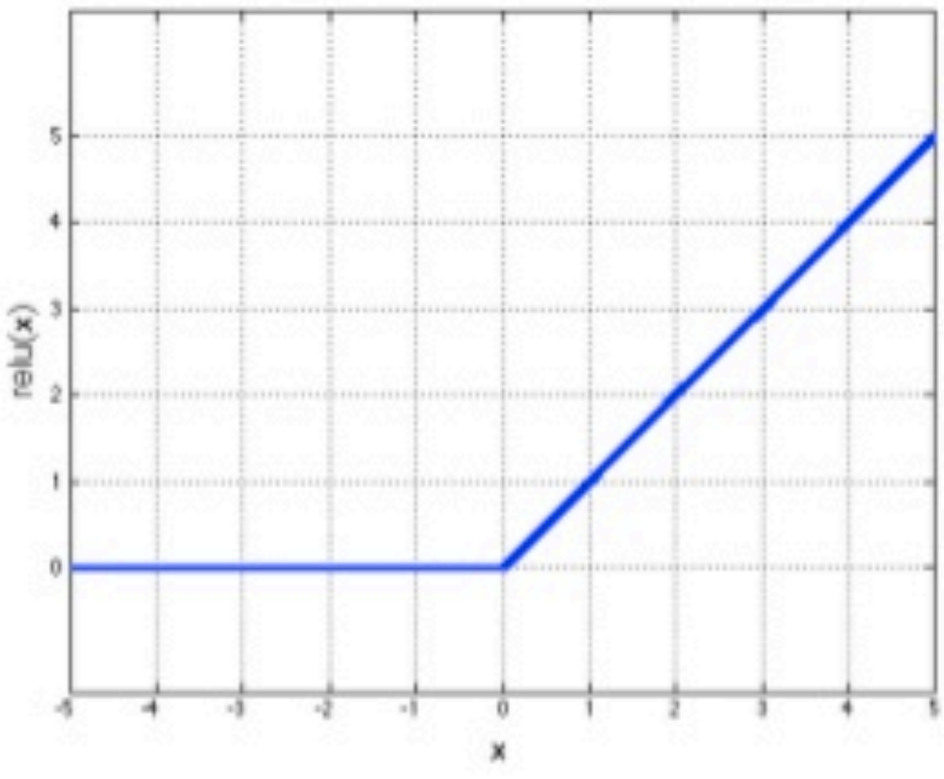


Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

Convolutional Neural Networks

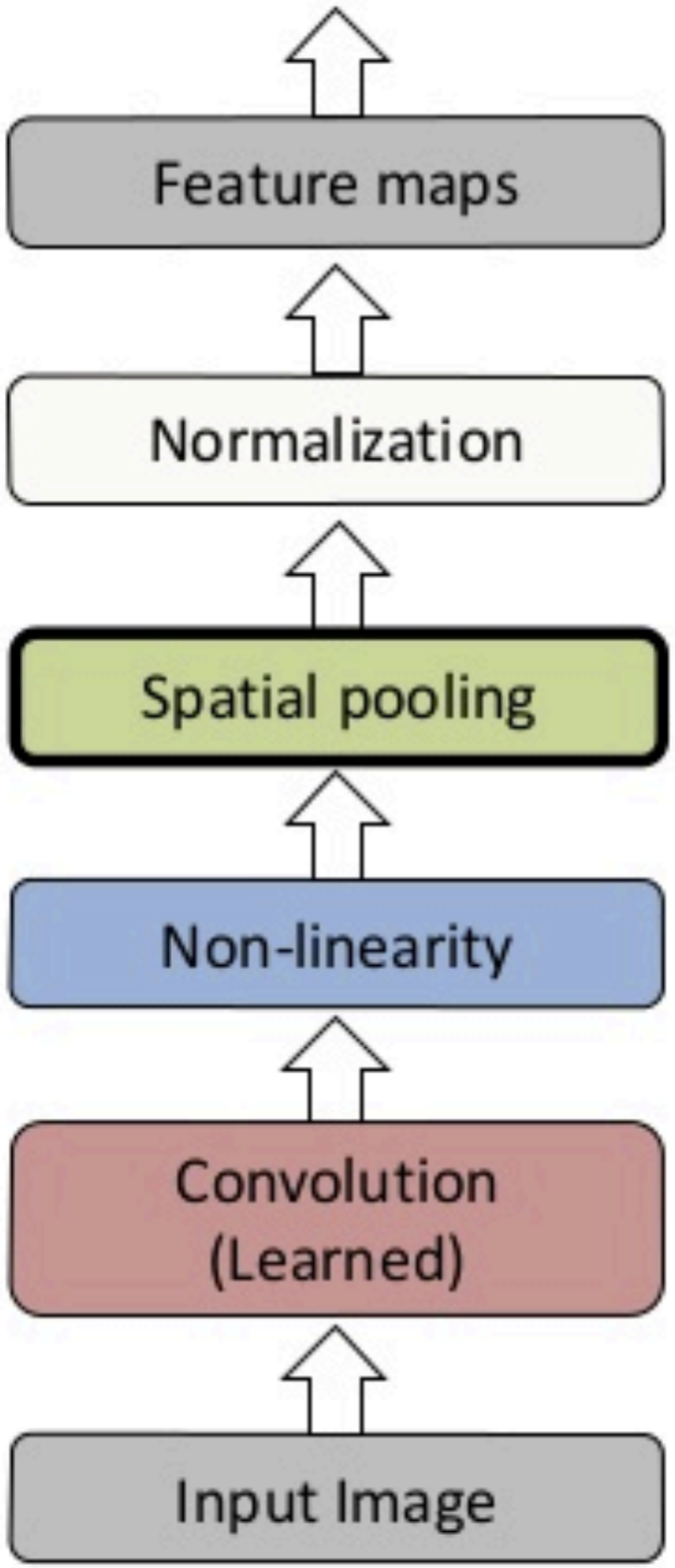


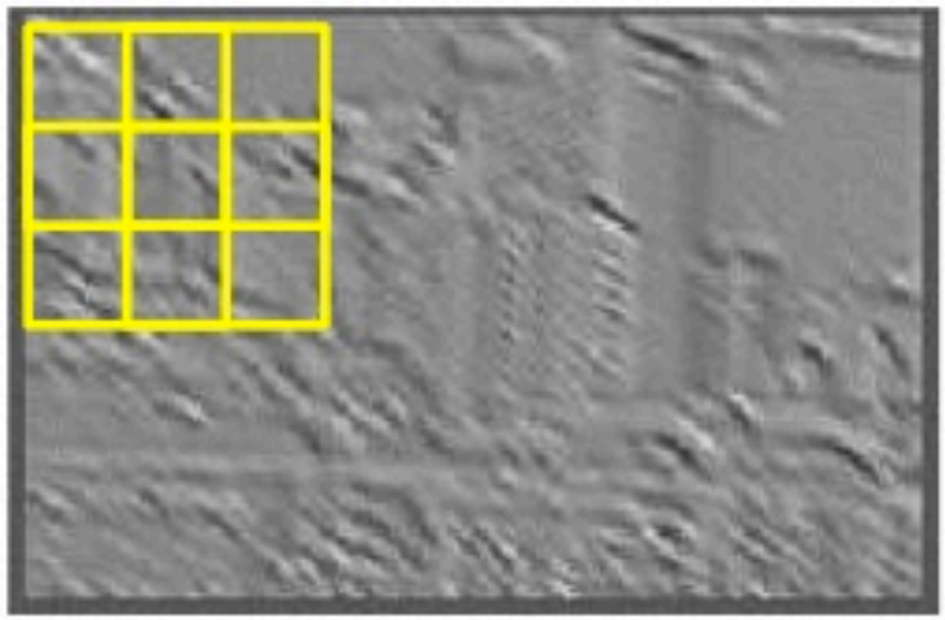
Rectified Linear Unit (ReLU)



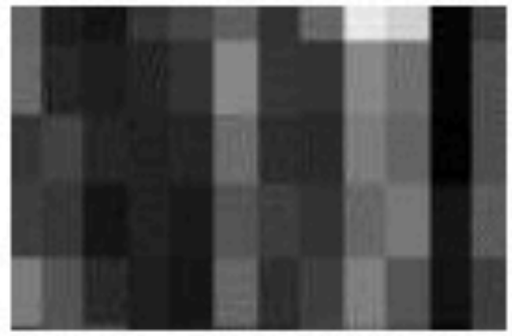
slide credit: S. Lazebnik

Convolutional Neural Networks



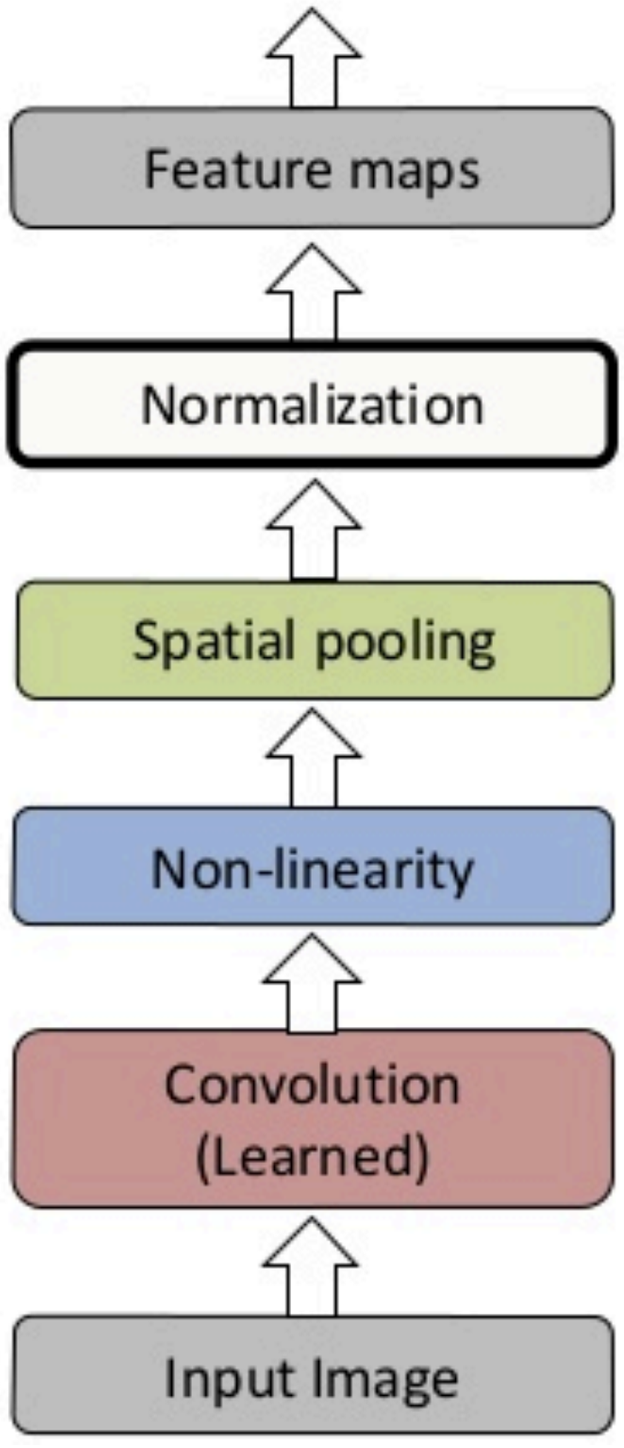


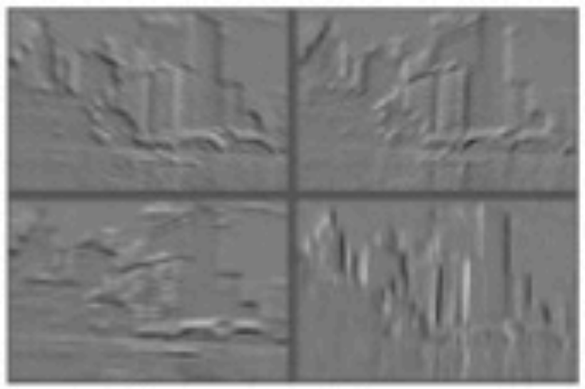
Max pooling



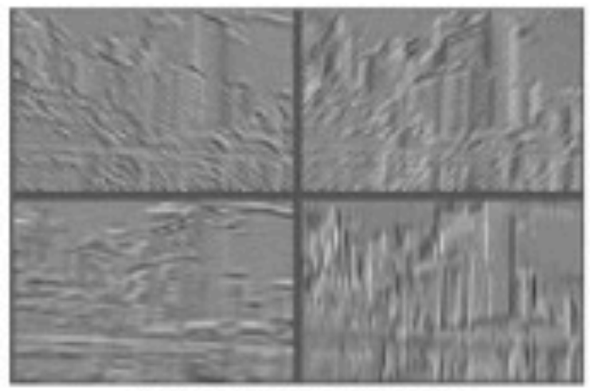
slide credit: S. Lazebnik

Convolutional Neural Networks





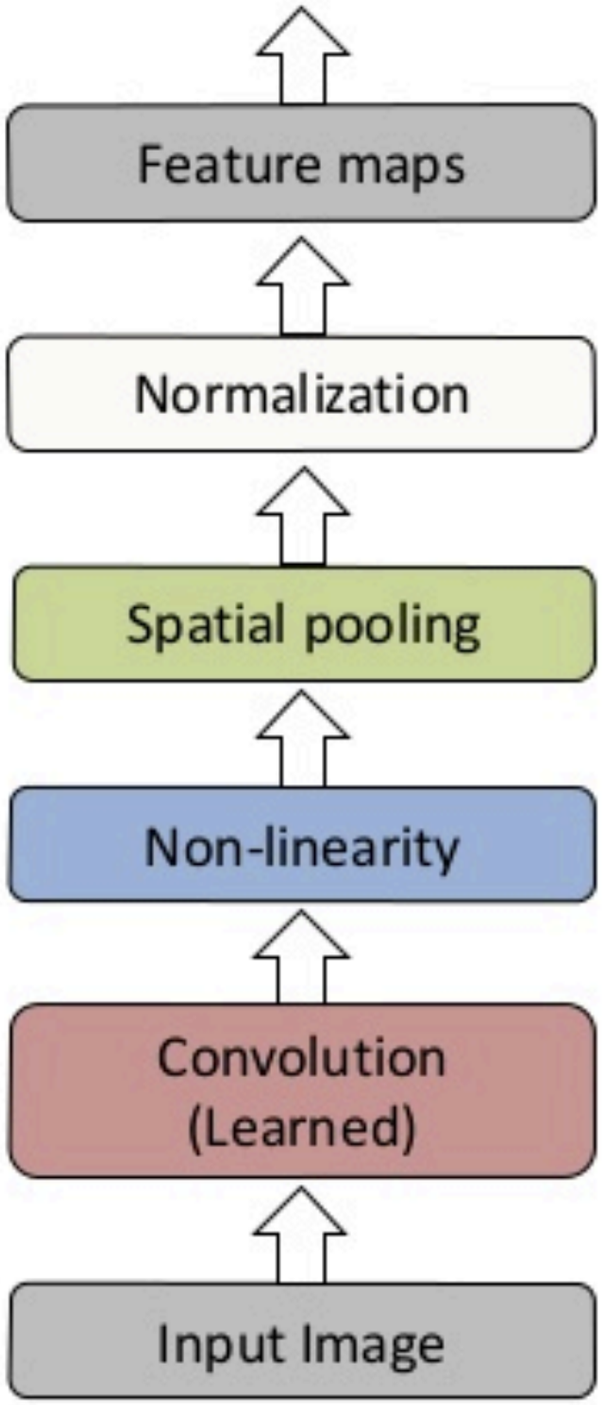
Feature Maps



Feature Maps
After Contrast
Normalization

slide credit: S. Lazebnik

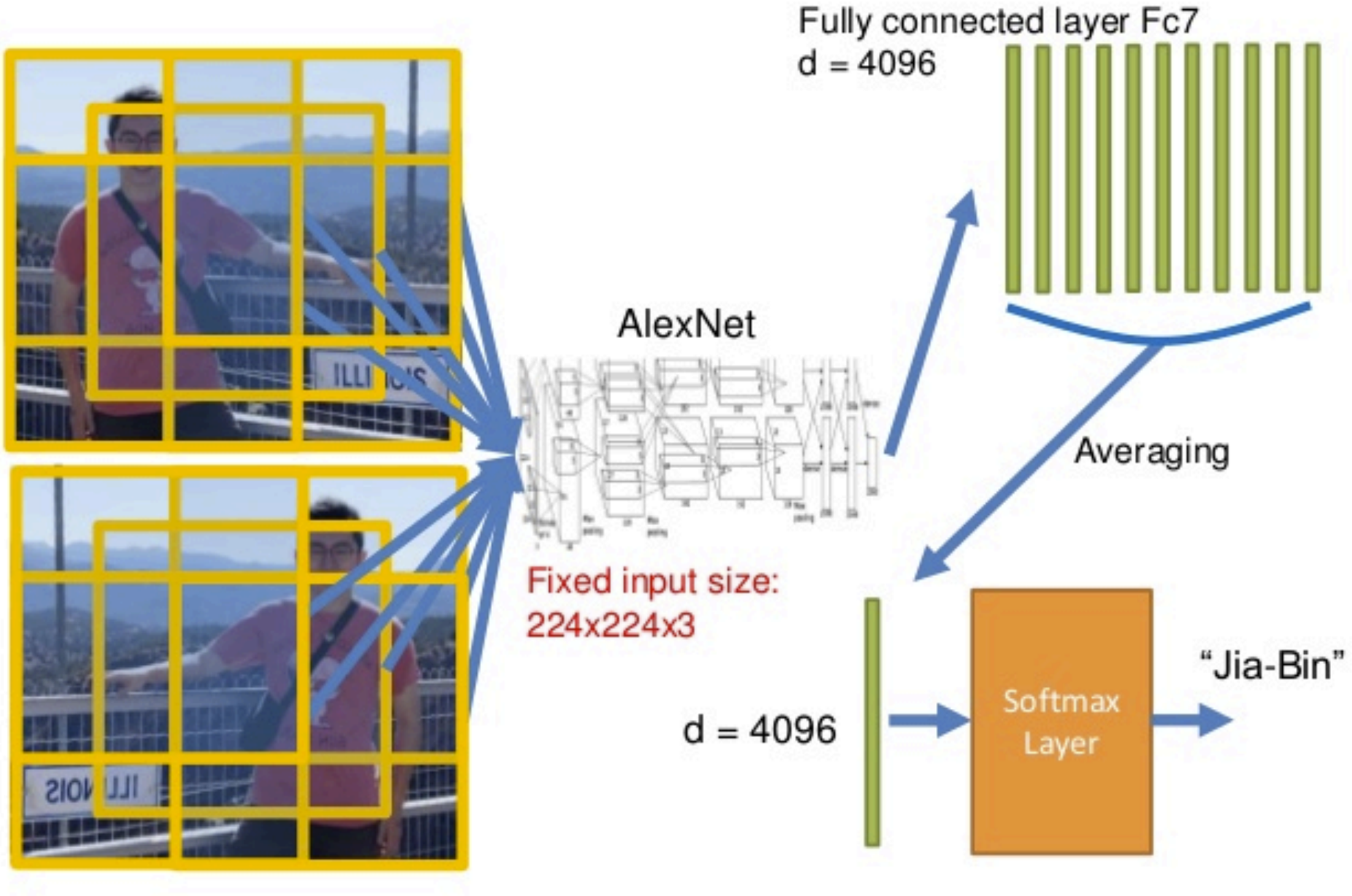
Convolutional Neural Networks



Convolutional filters are trained in a supervised manner by back-propagating classification error

slide credit: S. Lazebnik

Using CNN for Image Classification



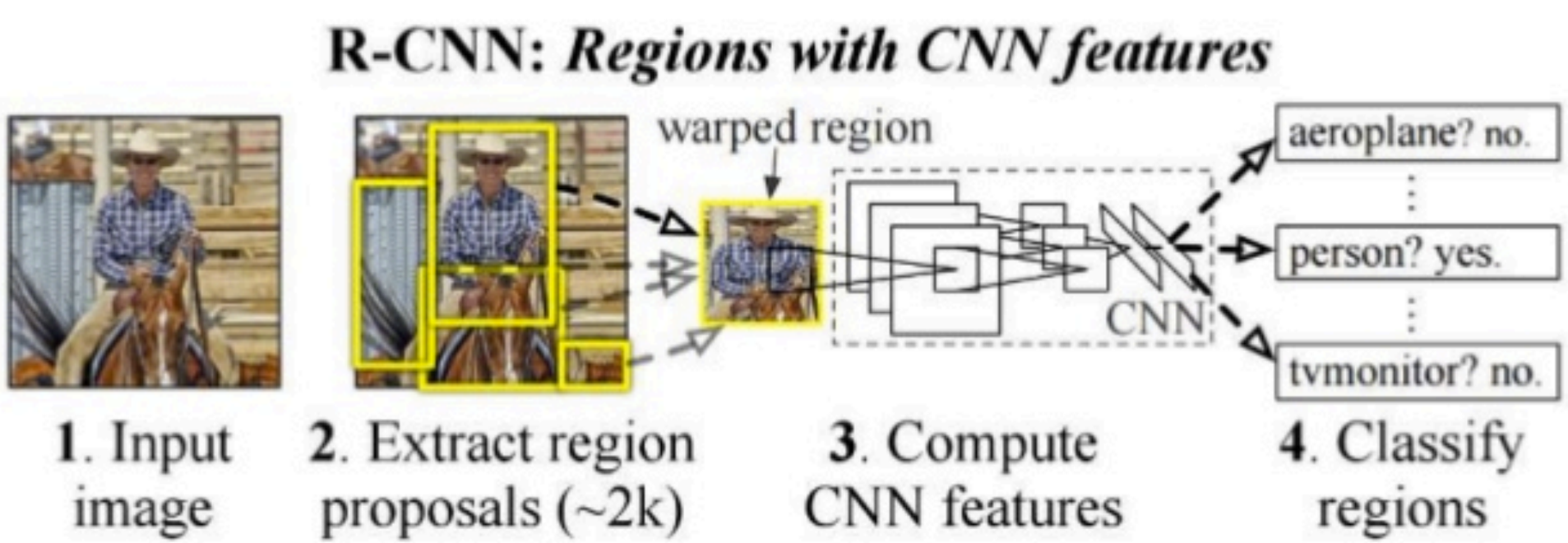
Limitations

ConvNets only capture local "spatial" patterns in data.

If the data can't be made to look like an image, ConvNets are less useful.

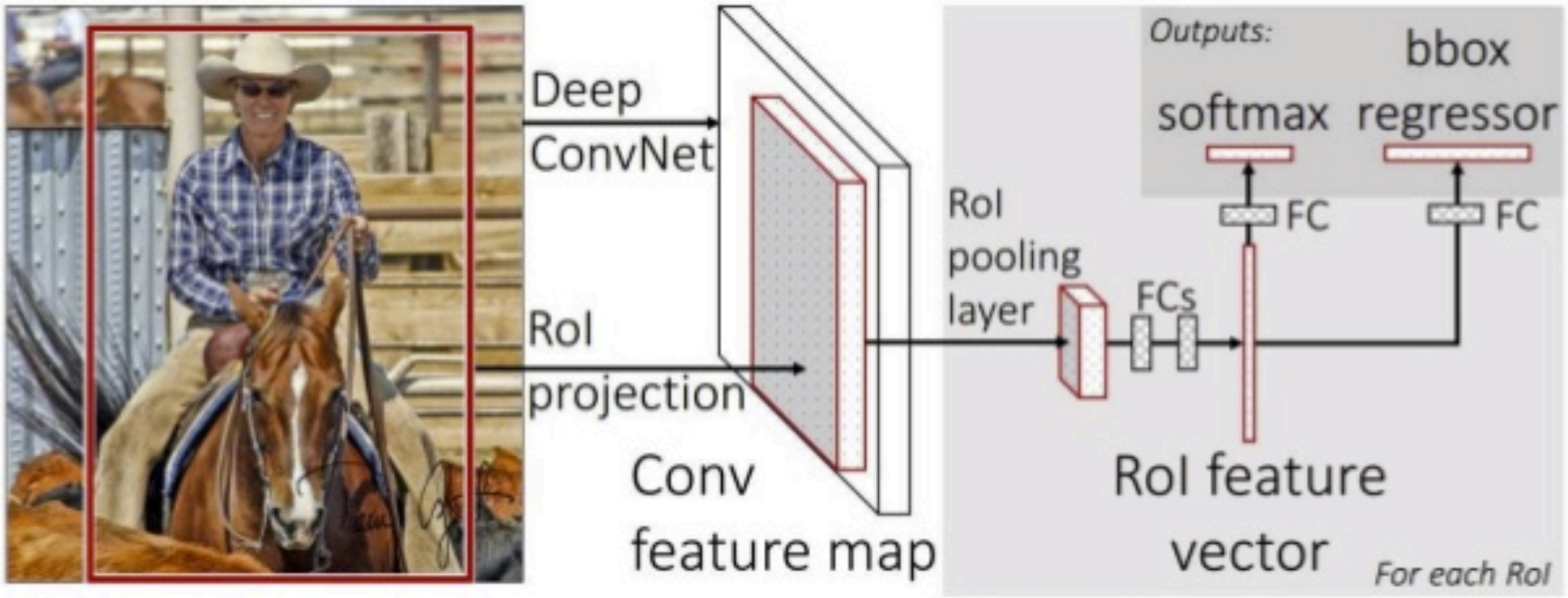
R-CNN: Regions with CNN features

- Trained on ImageNet classification
- Finetune CNN on PASCAL



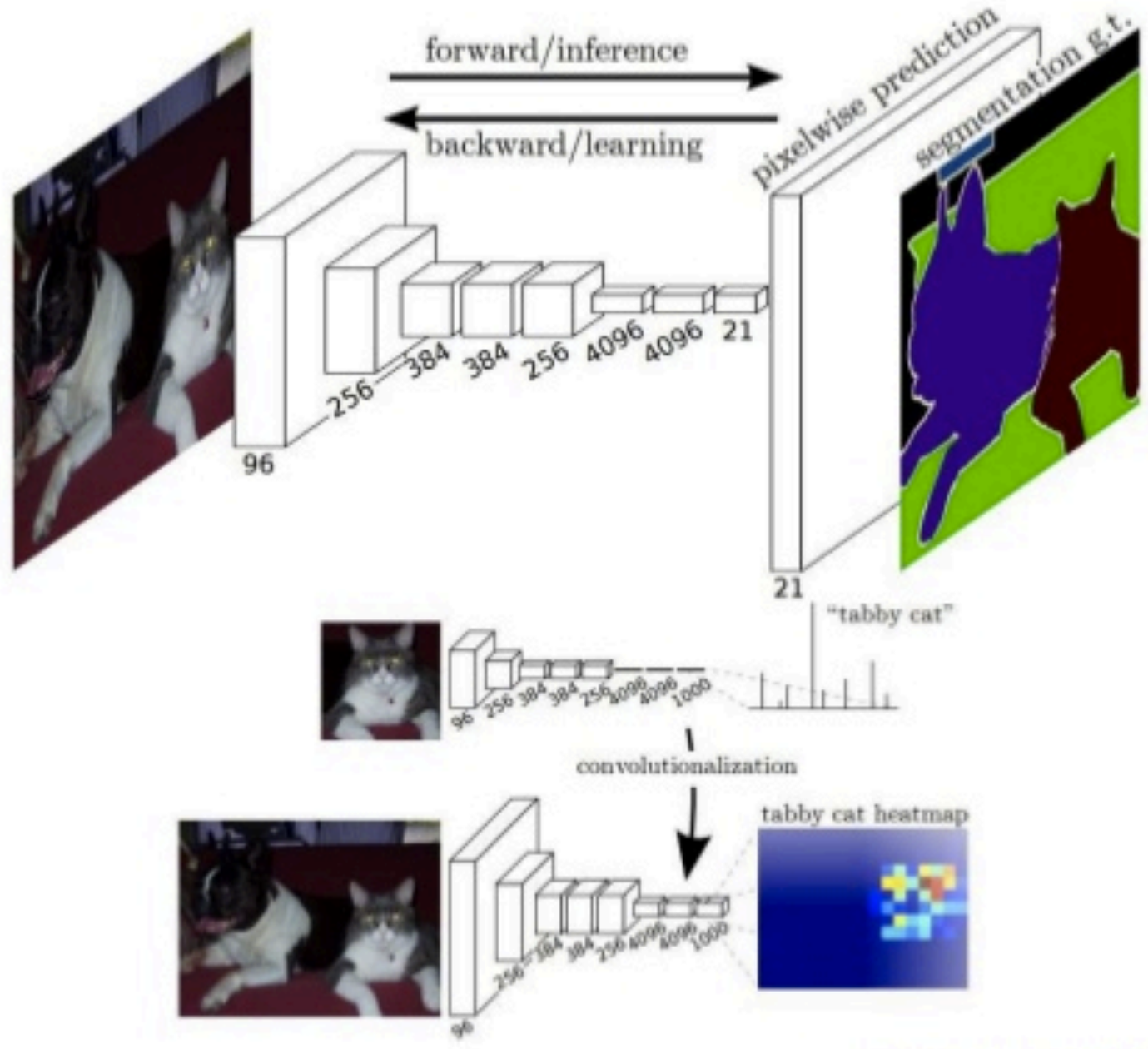
RCNN [Girshick et al. CVPR 2014]

Fast R-CNN



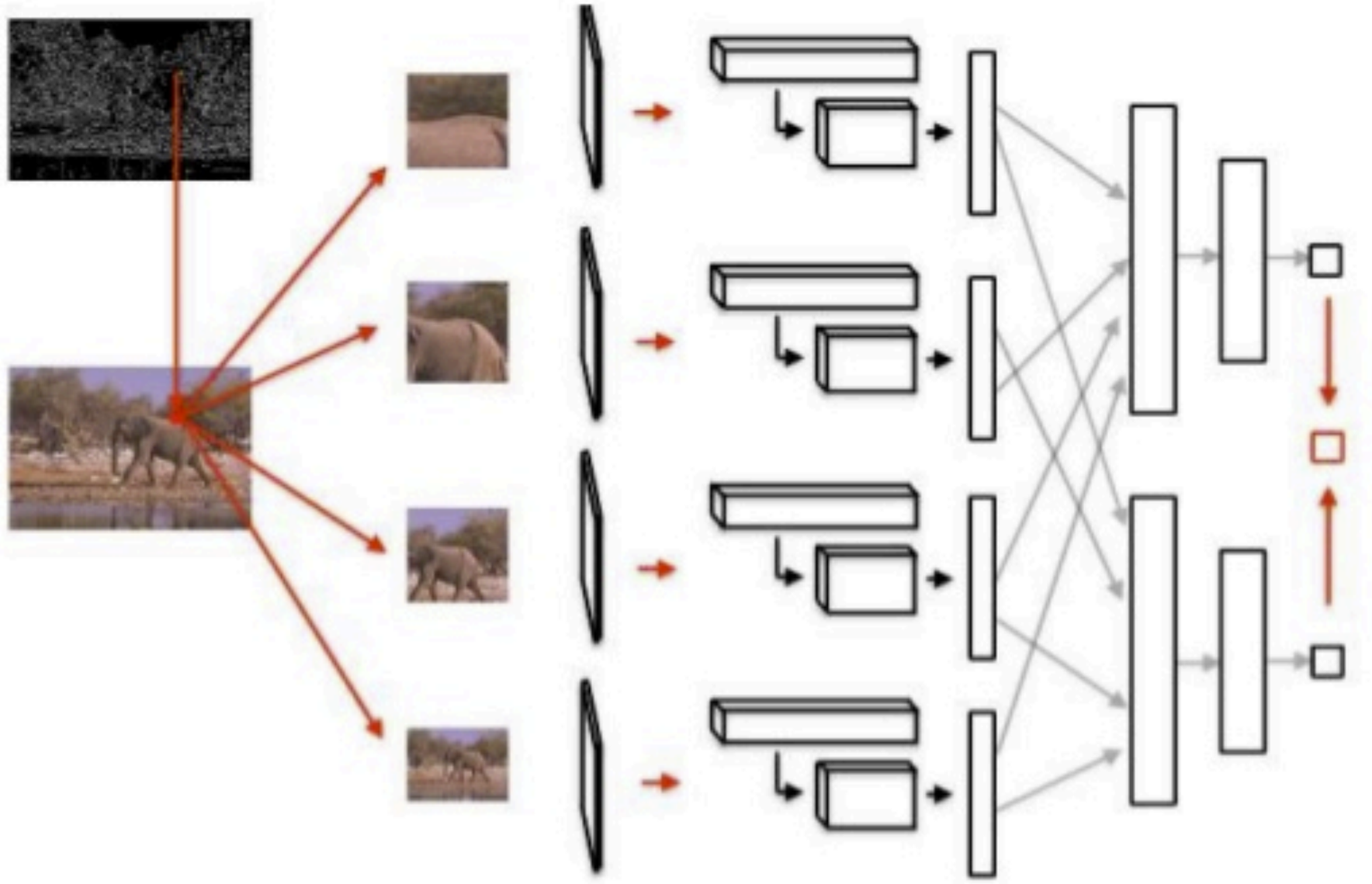
Fast RCNN [Girshick, R 2015]
https://github.com/rbgirshick/fast-rcnn

Labeling Pixels: Semantic Labels



Fully Convolutional Networks for Semantic Segmentation [Long et al. CVPR 2015]

Labeling Pixels: Edge Detection

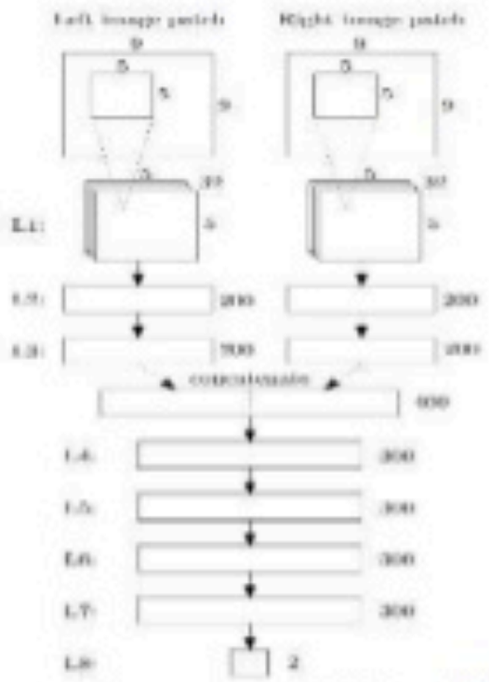


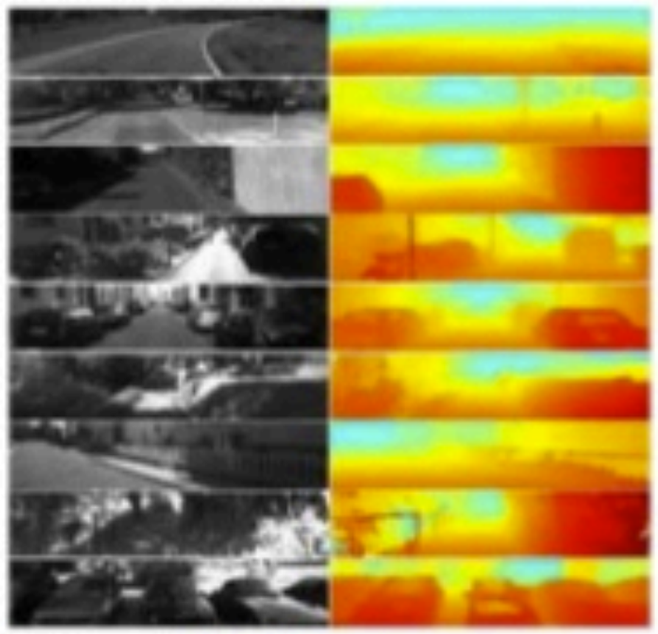
DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection
[Bertasius et al. CVPR 2015]

CNN for Regression

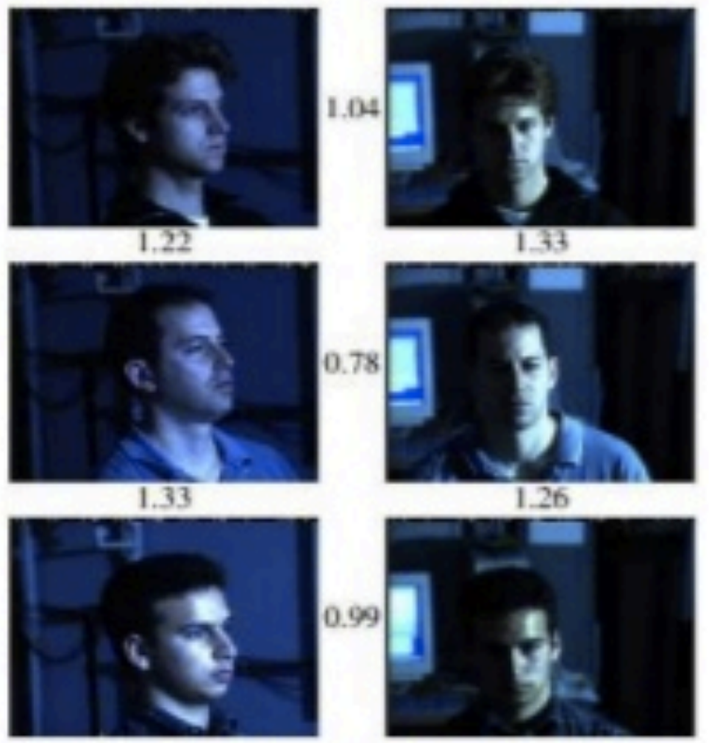
DeepPose [Toshev and Szegedy CVPR 2014]

CNN as a Similarity Measure for Matching

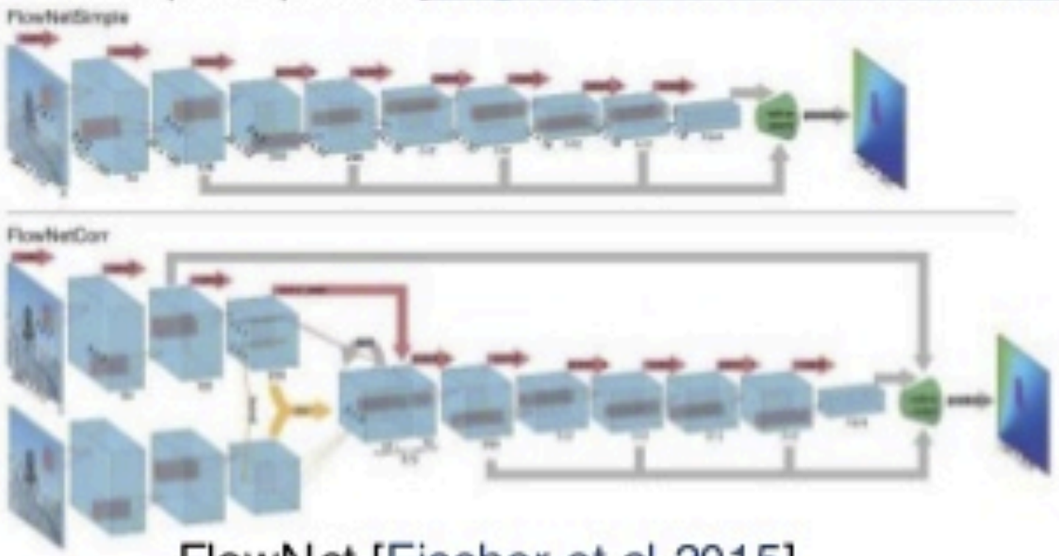




Stereo matching [Zbontar and LeCun CVPR 2015]
Compare patch [Zagoruyko and Komodakis 2015]



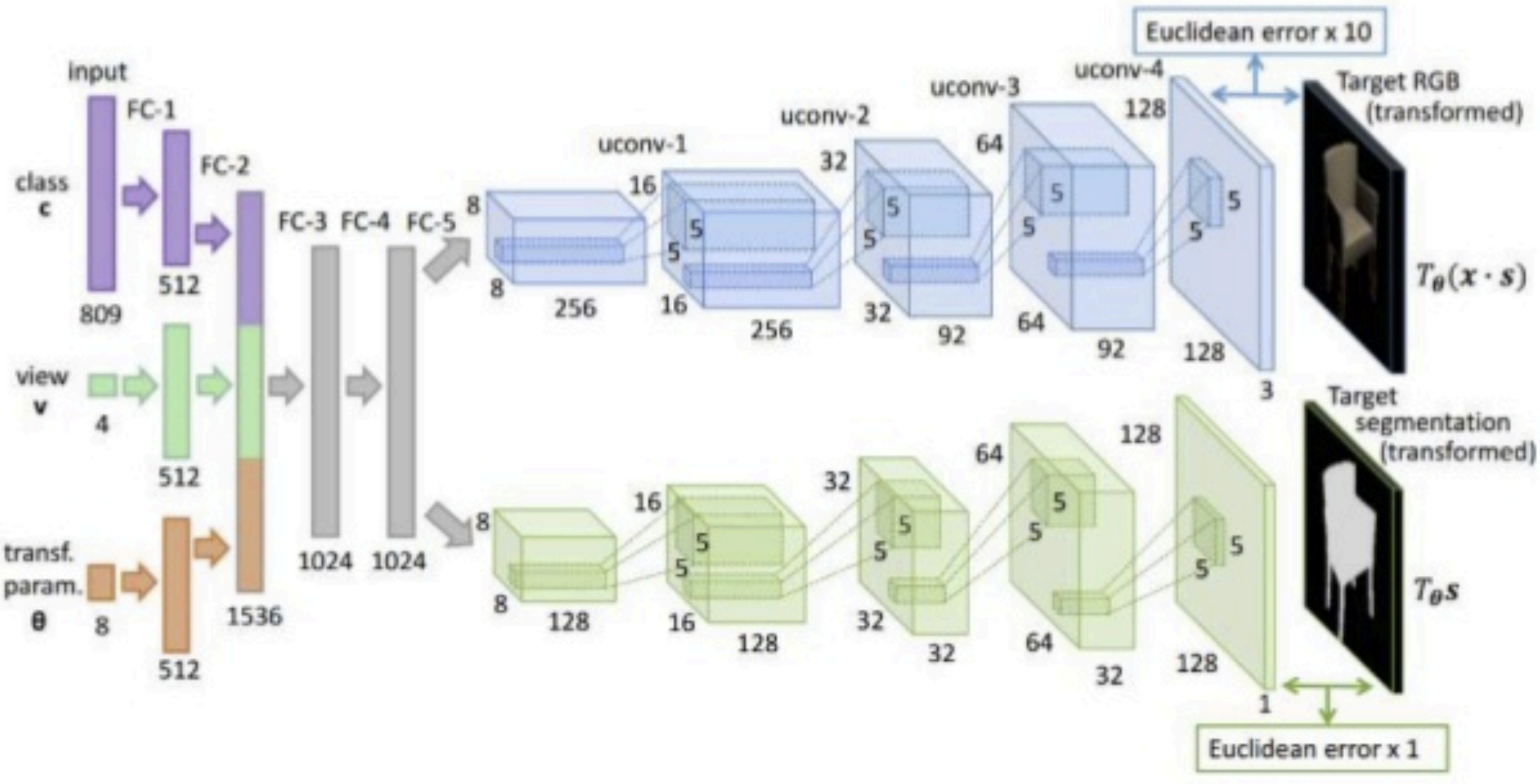
FaceNet [Schroff et al. 2015]



FlowNet [Fischer et al 2015]

Match ground and aerial images
[Lin et al. CVPR 2015]

CNN for Image Generation



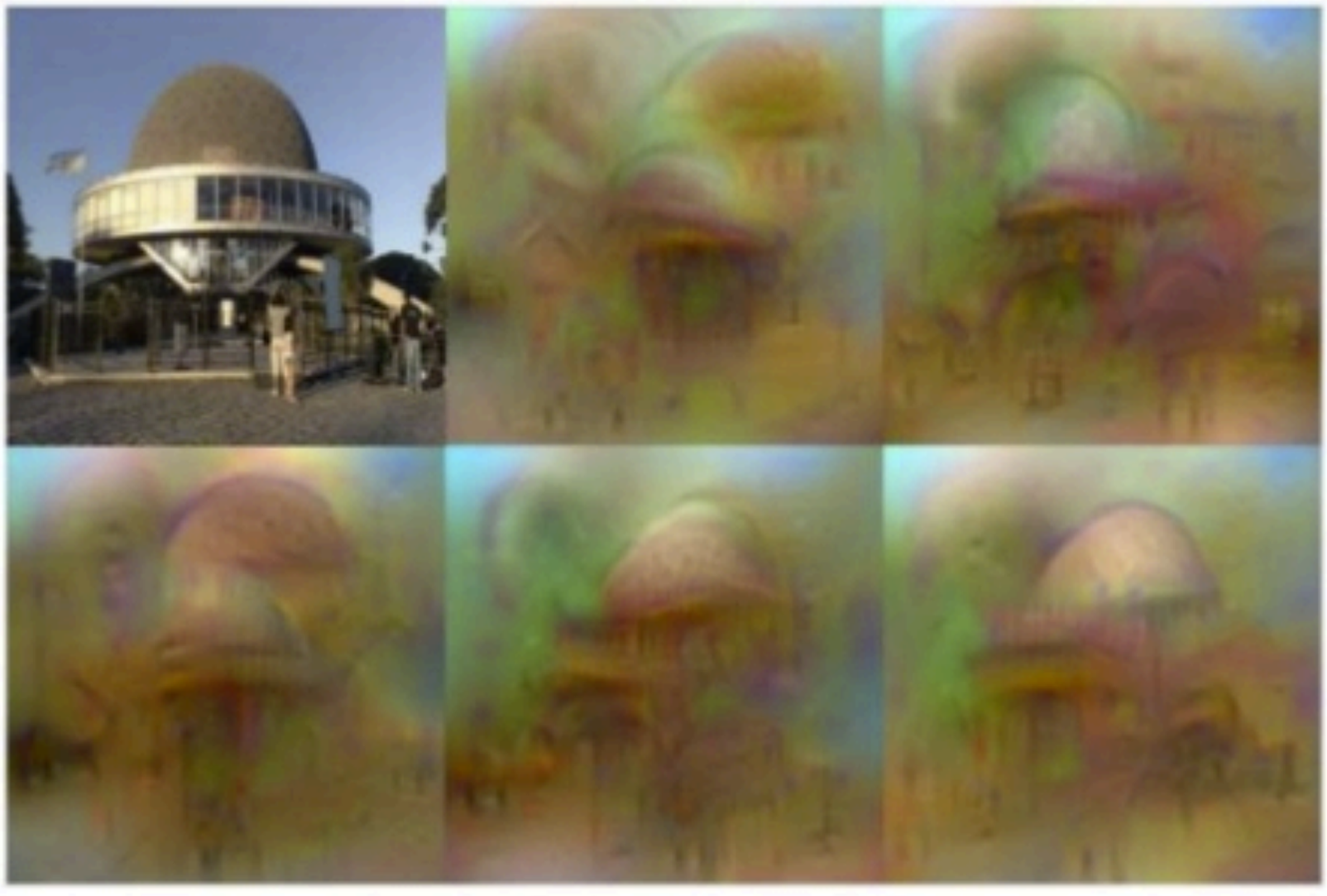
Learning to Generate Chairs with Convolutional Neural Networks [Dosovitskiy et al. CVPR 2015]

Individual Neuron Activation

RCNN [Girshick et al. CVPR 2014]

Invert CNN features

- Reconstruct an image from CNN features

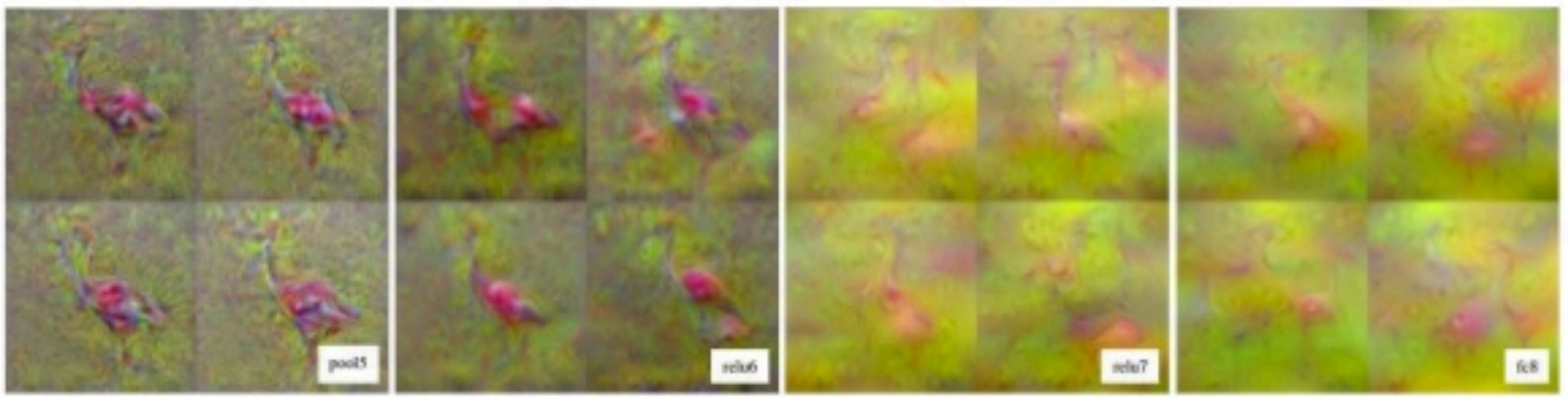


Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

CNN Reconstruction

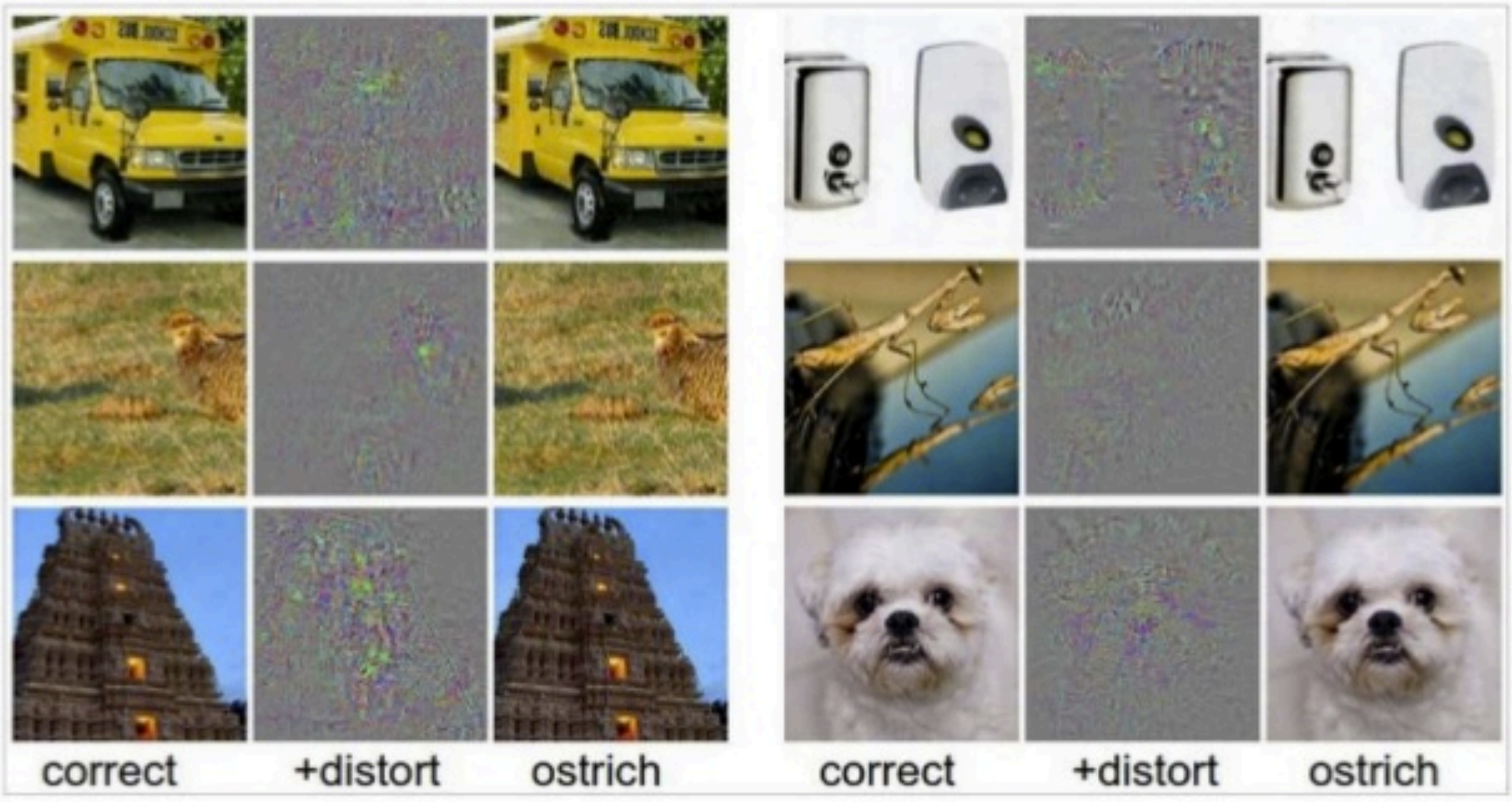
Reconstruction from different layers

Multiple reconstructions



Understanding deep image representations by inverting them
[Mahendran and Vedaldi CVPR 2015]

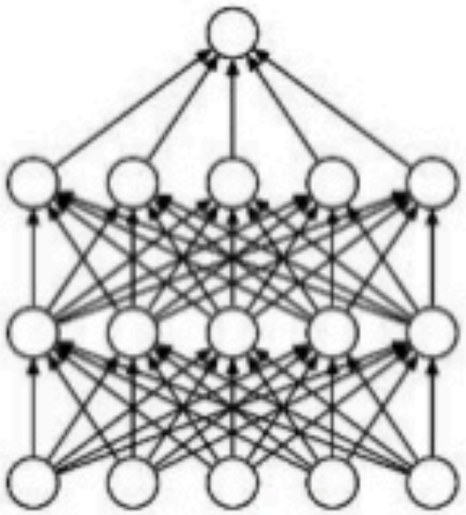
Breaking CNNs



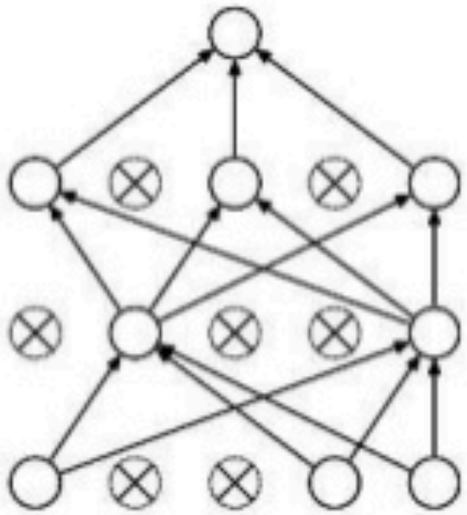
Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

Intriguing properties of neural networks [Szegedy ICLR 2014]

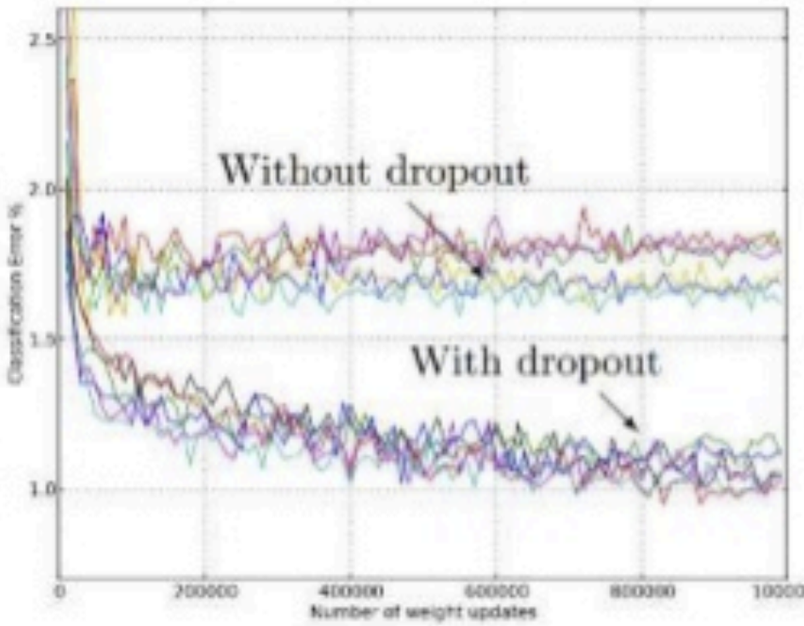
Dropout



(a) Standard Neural Net



(b) After applying dropout.



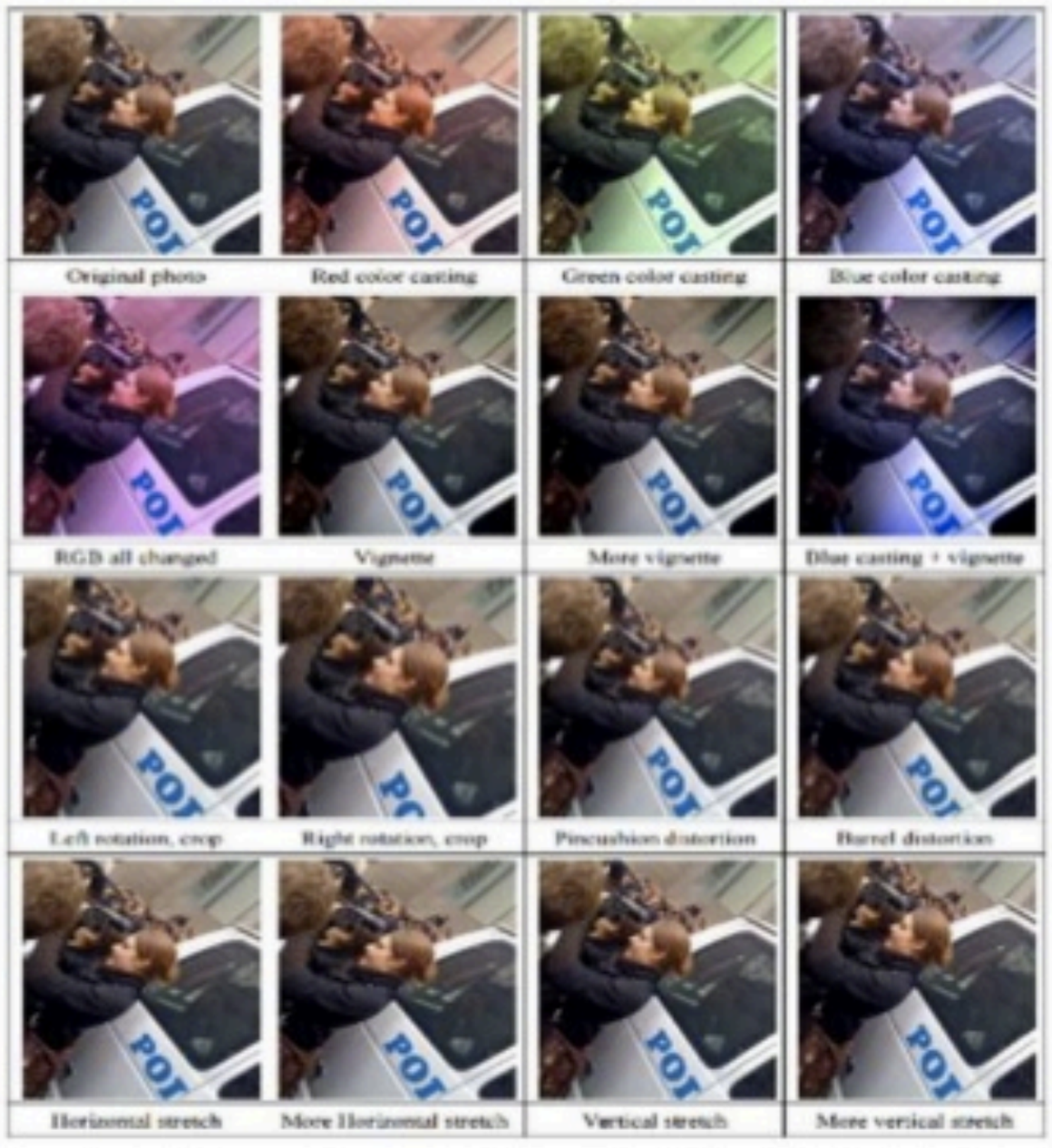
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Data Augmentation (Jittering)

- Create virtual training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion



Deep Image [Wu et al. 2015]