Computer Vision Fundamentals

Murtaza Taj/Usman Nazir

murtaza.taj@lums.edu.pk

Convolutional Neural Network

Vertical Edge Detection

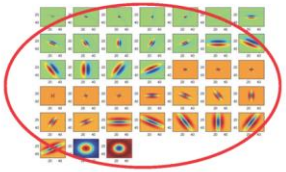
$$h_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

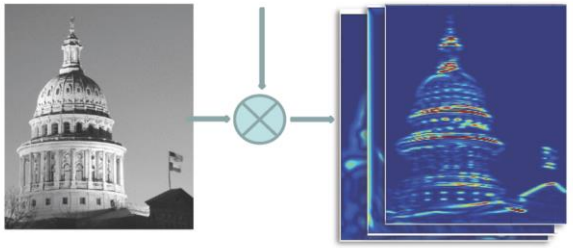
Original Image: Opera

Inverted Image for Visualisation

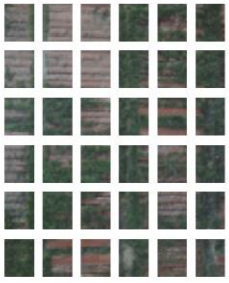
CNN Key Idea 1: Filter are learned

Learn from Data what these filters should be

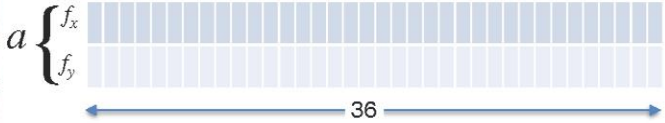




Texture representation: example



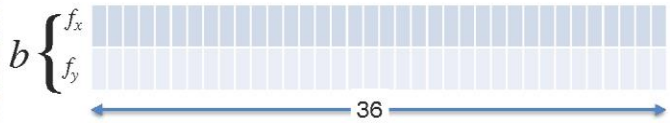
6x6



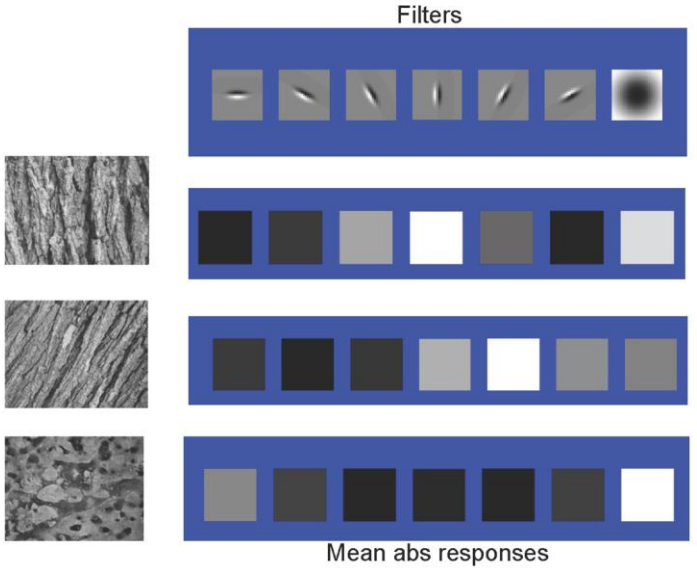
$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

$$= \sqrt{\sum_{i=1}^{2} (a_i - b_i)^2}$$

6x6



Representing texture by mean abs response



Derek Hoiem

Neuron

$$\mathbf{w} = \{w_0, w_1, w_2, \cdots\}^T$$
$$\mathbf{x} = \{1, x_1, x_2, \cdots\}^T$$
$$y = \mathbf{w}^T\mathbf{x} = \sum w_j x_j$$

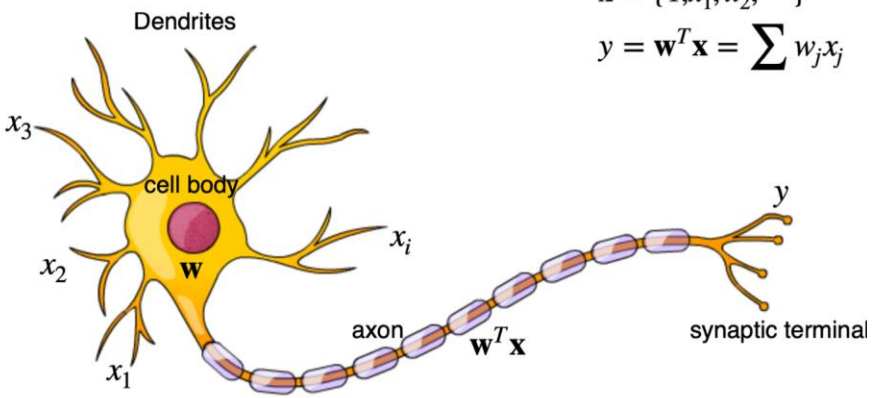
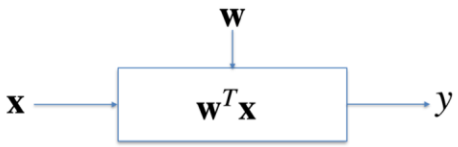
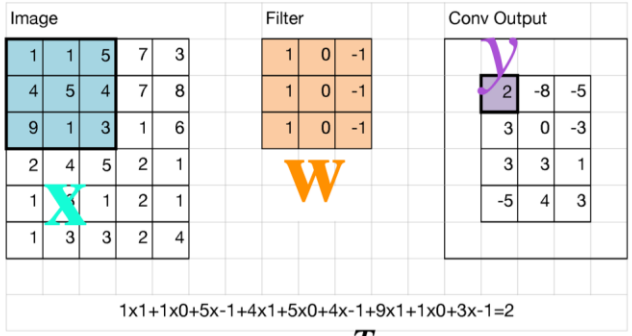


Figure 1-6. A functional description of a biological neuron's structure

Recap: Single Step of Convolution

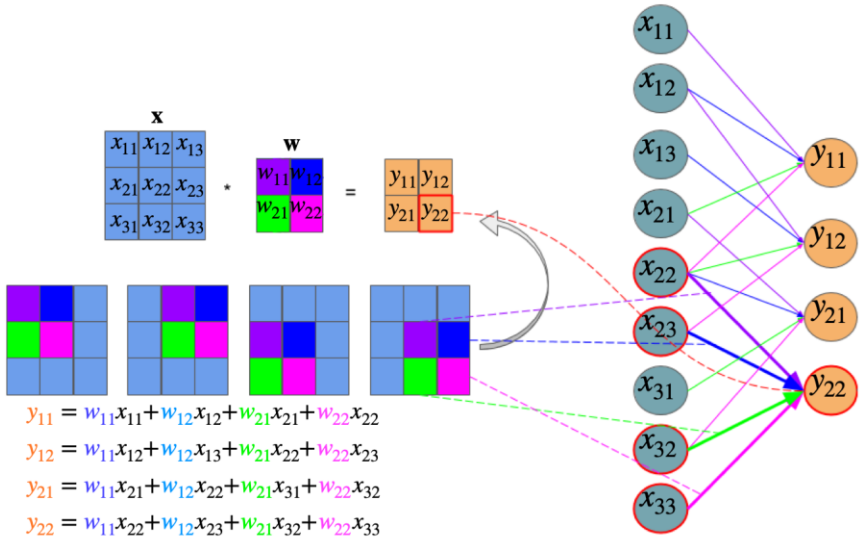
$$\mathbf{w} = \{w_0, w_1, w_2, \cdots\}^T$$
$$\mathbf{x} = \{1, x_1, x_2, \cdots\}^T$$
$$y = \mathbf{w}^T\mathbf{x} = \sum w_j x_j$$





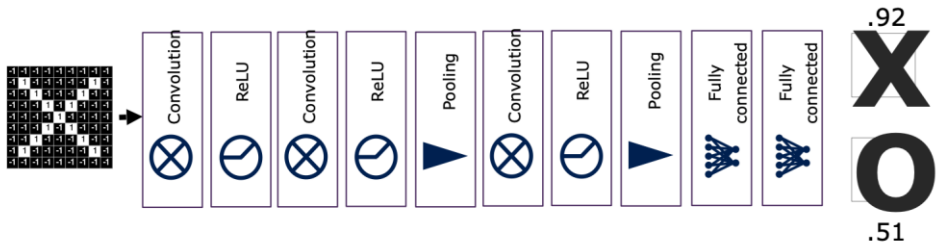
$$y = \mathbf{w}^T\mathbf{x}$$

Feedforward in CNN is identical with convolution operation



Putting it all together

A set of pixels becomes a set of votes.



https://grzegorzgwardys.wordpress.com/2016/04/22/

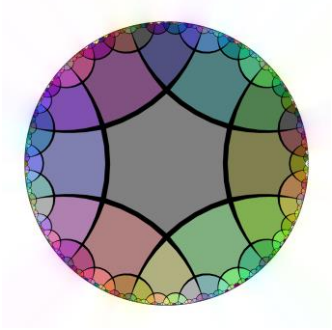
Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

"It is only slightly overstating the case to say that physics is the study of symmetry"

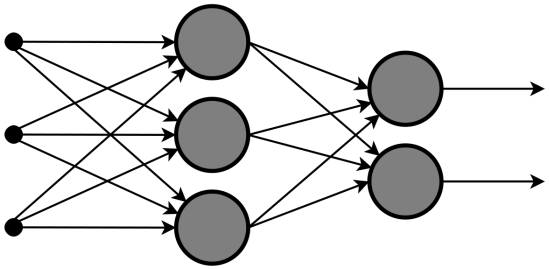
Philip Anderson

Unification of Computer Vision and Pattern Recognition from Symmetry and Equivariance ?

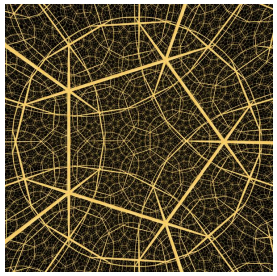
Overview



History of geometry

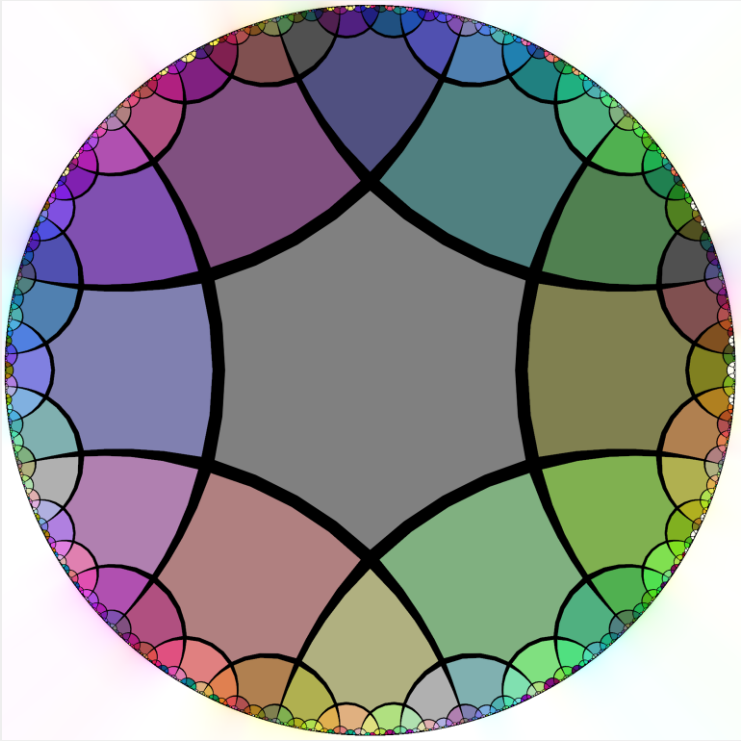


History of Neural Networks



Erlangen Programme of ML

History of geometry



Euclid

300BC

= geometry

N. Lobachevsky

1826

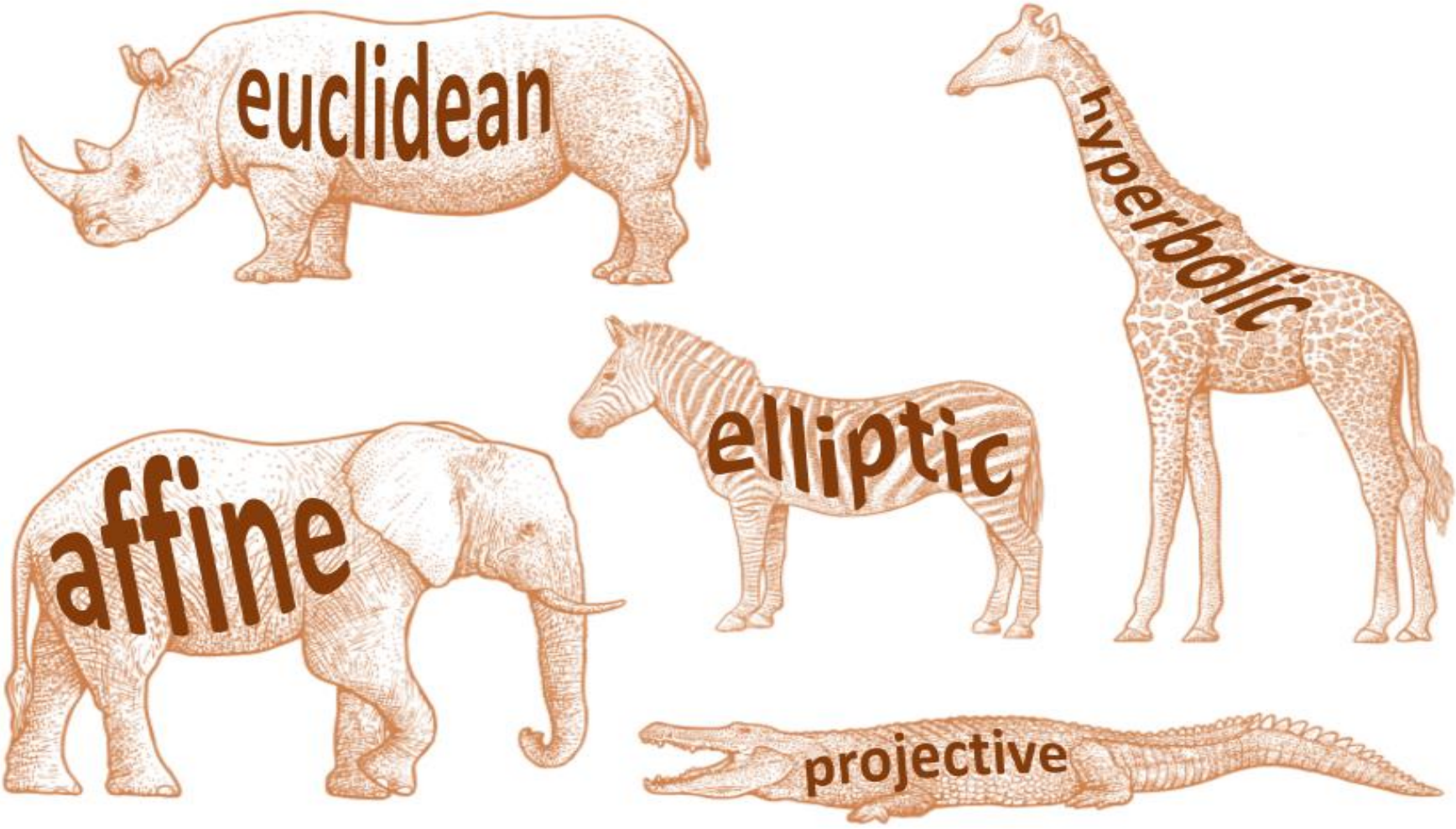
C. F. Gauss

J. Bolyai

1832

B. Riemann

1856


euclidean
hyperbolic
elliptic
affine
projective

F. Klein

1872

Vergleichende Betrachtungen

über

neuere geometrische Forschungen

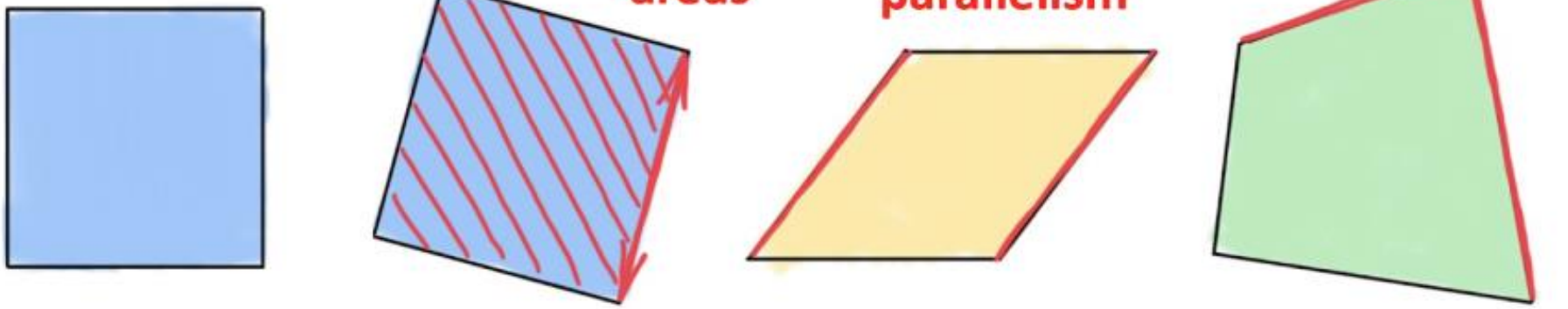
von

Dr. Felix Klein,
o. ö. Professor der Mathematik an der Universität Erlangen.

Programm

zum Eintritt in die philosophische Facultät und den Senat
der k. Friedrich-Alexanders-Universität
zu Erlangen.

Erlangen.
Verlag von Andreas Deichert.
1872.



Euclidean **Affine** **Projective**

$\mathrm{Iso} \leq \mathrm{Aff} \leq \mathrm{PGL}$

E. Noether

1918

Invariante Variationsprobleme.

(F. Klein zum fünfzigjährigen Doktorjubiläum.)

Von

Emmy Noether in Göttingen.

Vorgelegt von F. Klein in der Sitzung vom 26. Juli 1918[1]).

Es handelt sich um Variationsprobleme, die eine kontinuierliche Gruppe (im Lieschen Sinne) gestatten; die daraus sich ergebenden Folgerungen für die zugehörigen Differentialgleichungen finden ihren allgemeinsten Ausdruck in den in § 1 formulierten, in den folgenden Paragraphen bewiesenen Sätzen. Über diese aus Variationsproblemen entspringenden Differentialgleichungen lassen sich viel präzisere Aussagen machen als über beliebige, eine Gruppe gestattende Differentialgleichungen, die den Gegenstand der Lieschen Untersuchungen bilden. Das folgende beruht also auf einer Verbindung der Methoden der formalen Variationsrechnung mit denen der Lieschen Gruppentheorie. Für spezielle Gruppen und Variationsprobleme ist diese Verbindung der Methoden nicht neu; ich erwähne Hamel und Herglotz für spezielle endliche, Lorentz und seine Schüler (z. B. Fokker), Weyl und Klein für spezielle unendliche Gruppen[2]). Insbesondere sind die zweite Kleinsche Note und die vorliegenden Ausführungen gegenseitig durch einander beein-

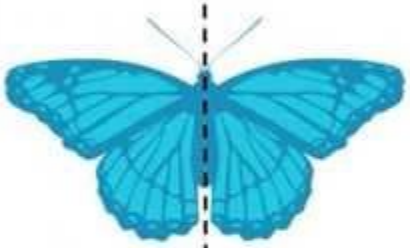
1) Die endgiltige Fassung des Manuskriptes wurde erst Ende September eingereicht.

2) Hamel: Math. Ann. Bd. 59 und Zeitschrift f. Math. u. Phys. Bd. 50. Herglotz: Ann. d. Phys. (4) Bd. 36, bes. § 9, S. 511. Fokker, Verslag d. Amsterdamer Akad., 27./1. 1917. Für die weitere Litteratur vergl. die zweite Note von Klein: Göttinger Nachrichten 19. Juli 1918.

In einer eben erschienenen Arbeit von Kneser (Math. Zeitschrift Bd. 2) handelt es sich um Aufstellung von Invarianten nach ähnlicher Methode.

Noether's Theorem

Line Symmetry



Noether's Theorem says if a system has a continuous symmetry, then there must be corresponding quantities whose values are conserved.

H. Weyl

1929

Gauge invariance

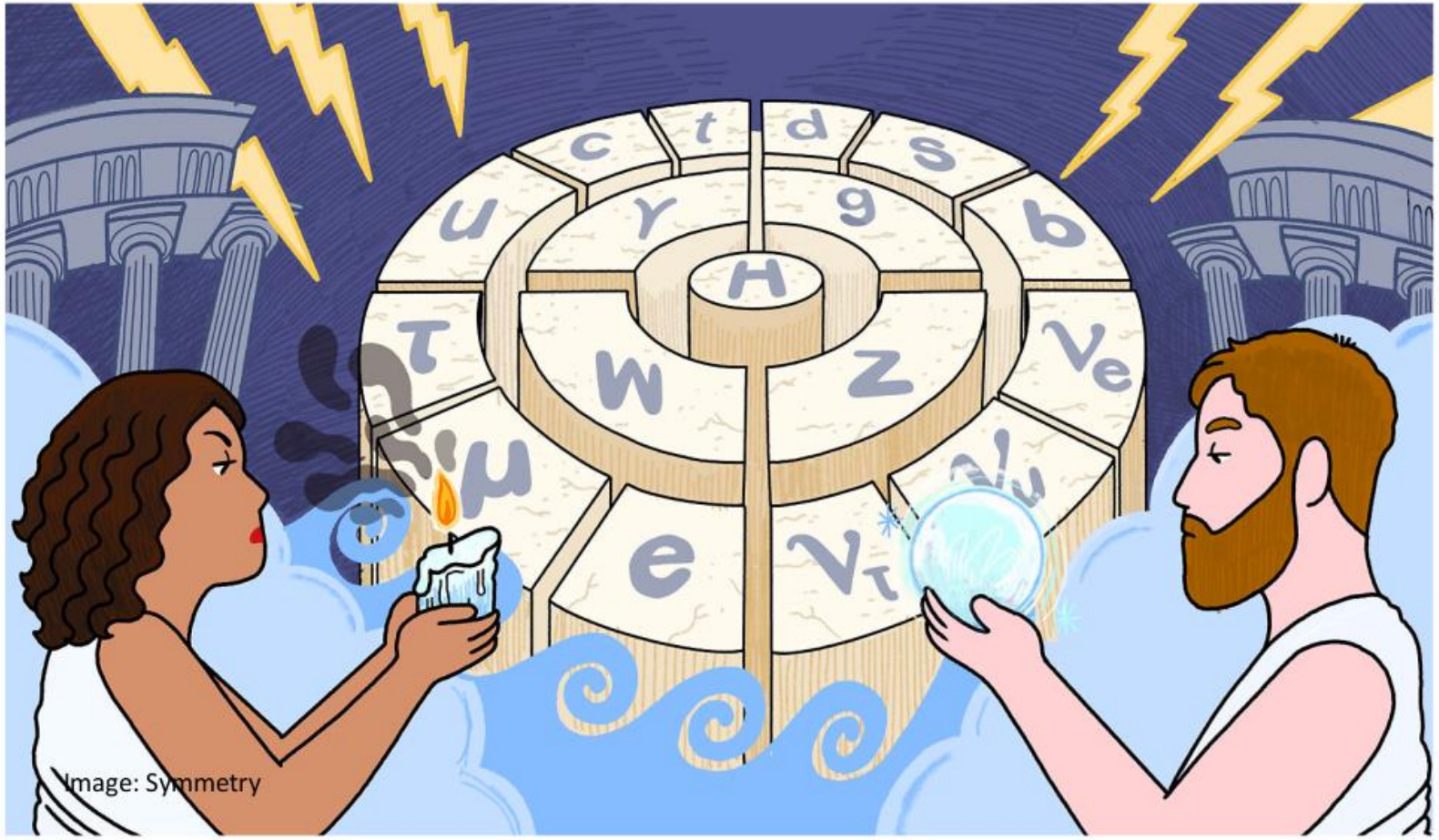
Yang

Mills

1954

Unification of forces

Standard Model

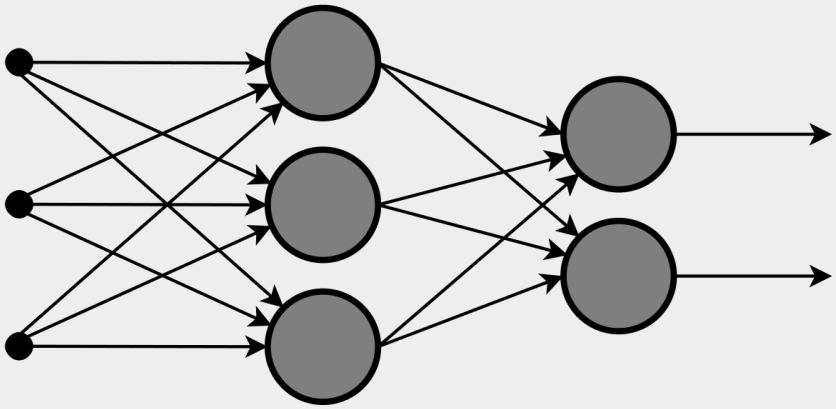


"It is only slightly overstating the case to say that physics is the study of symmetry"

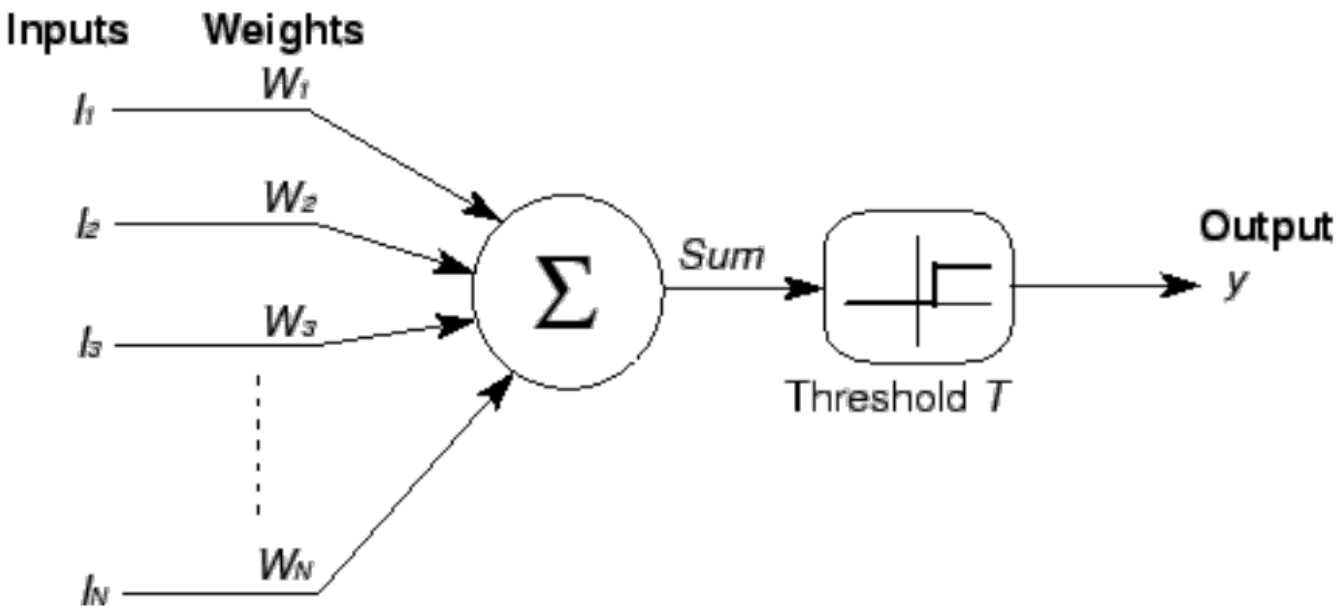
Philip Anderson

Unification of Deep Learning from Symmetry and Equivariance

History of Neural Networks

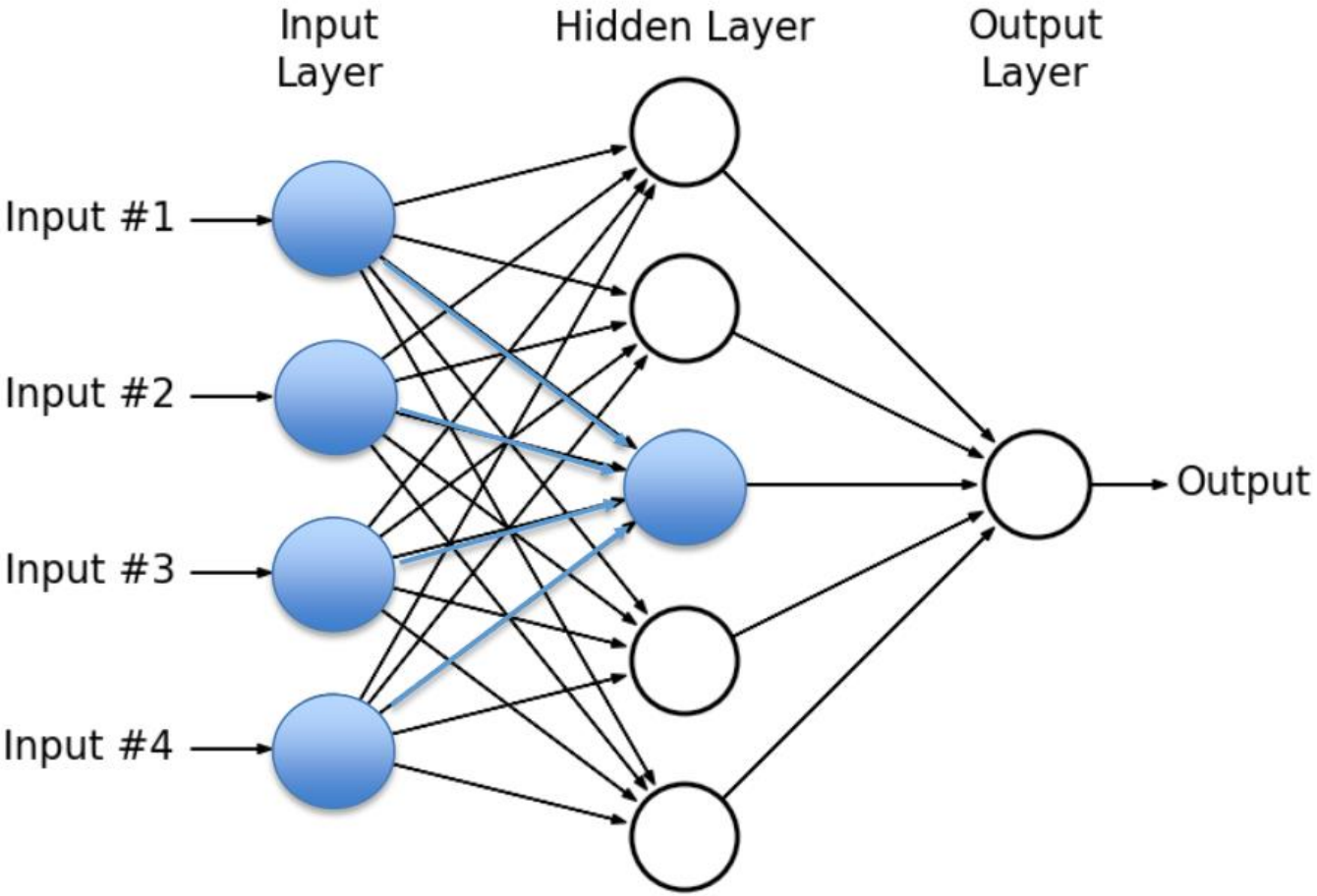


70 Years Ago



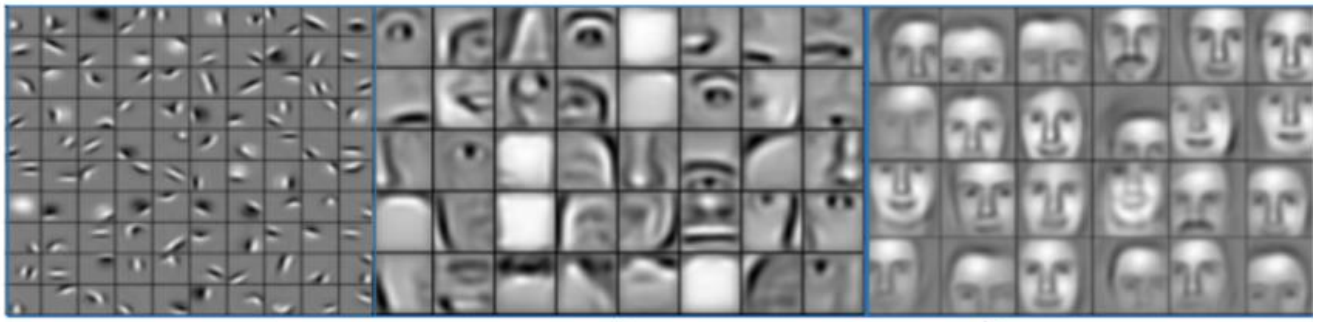
McCullogh & Pitts 1943
Rosenblatt 1957

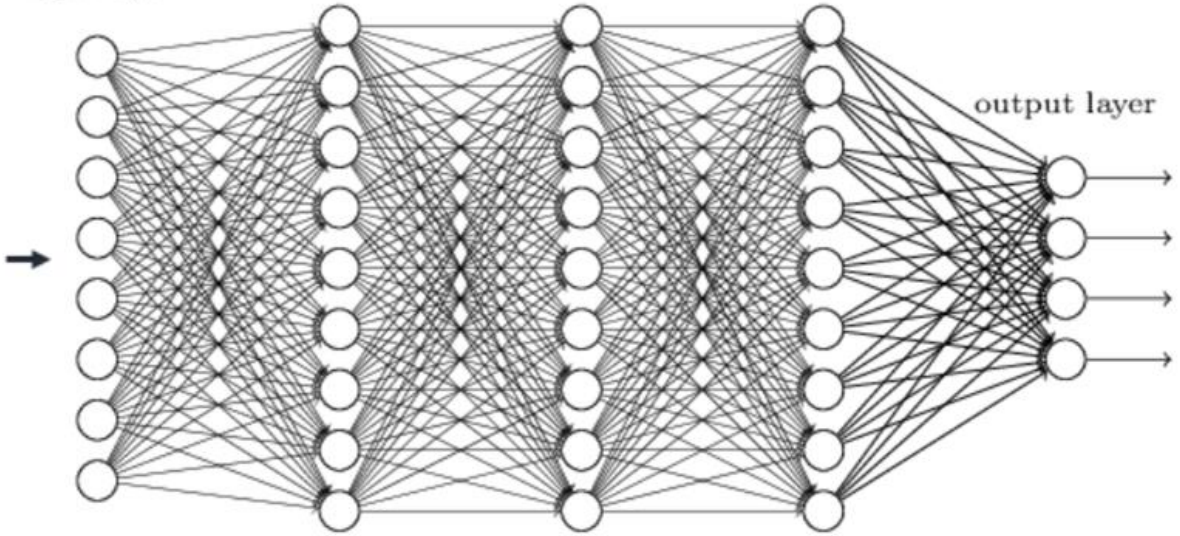
50 Years Ago


Input Layer
Hidden Layer
Output Layer
Input #1
Input #2
Input #3
Input #4
Output

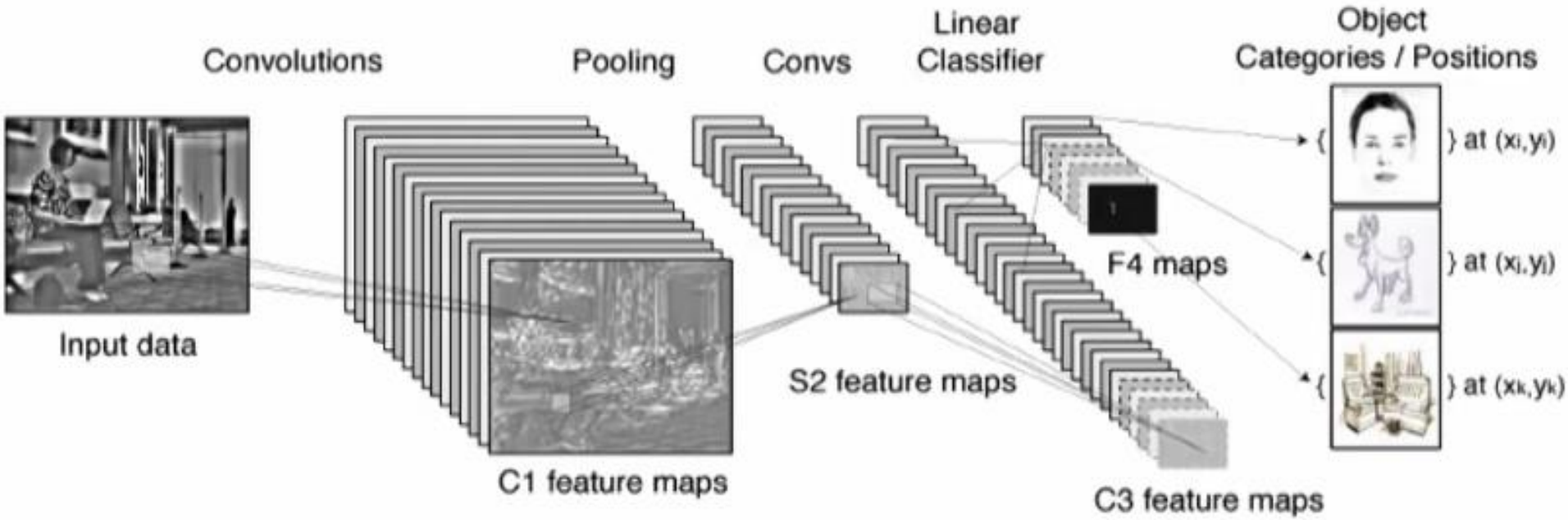
8 Years Ago

Deep neural networks learn hierarchical feature representations

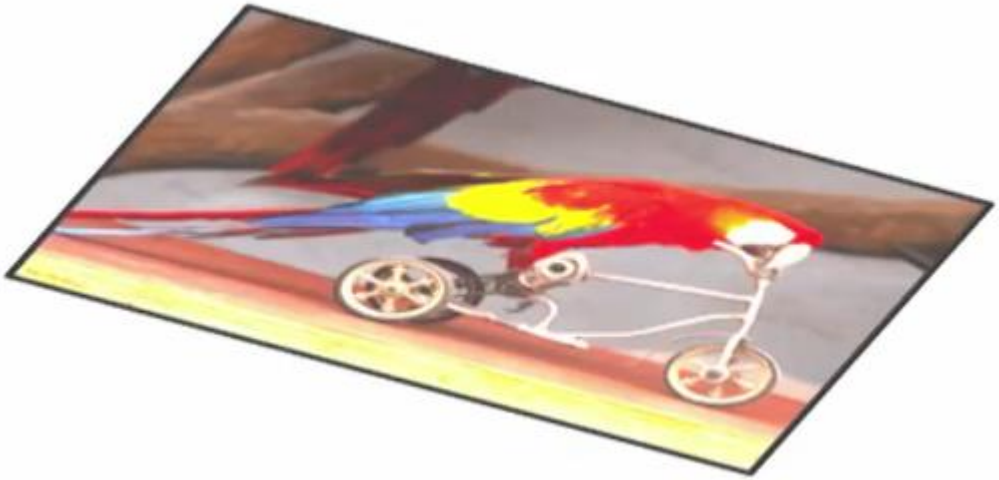




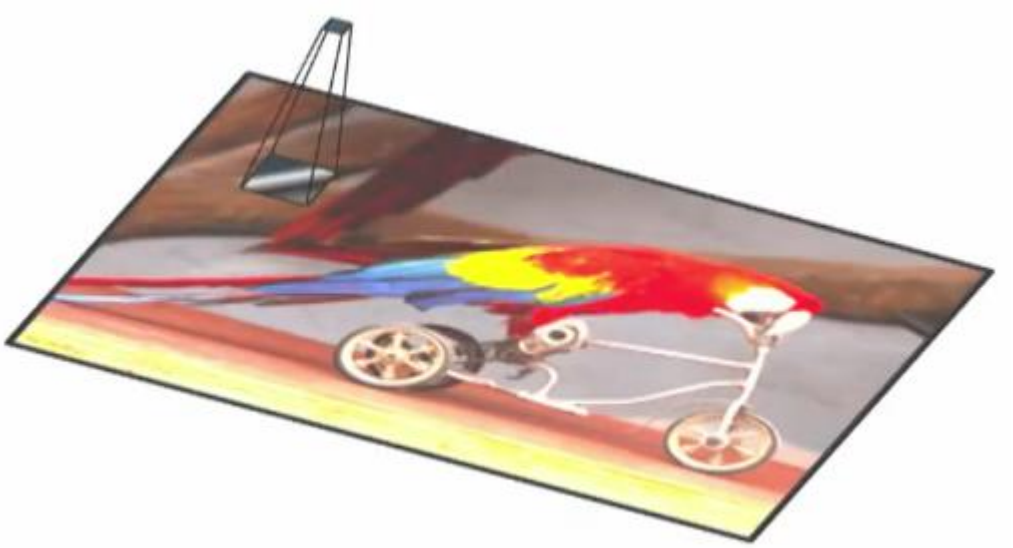
Forward: Filter, subsample, filter, nonlinearity, subsample, …., classify



Backward: backpropagation (propagate error signal backward)

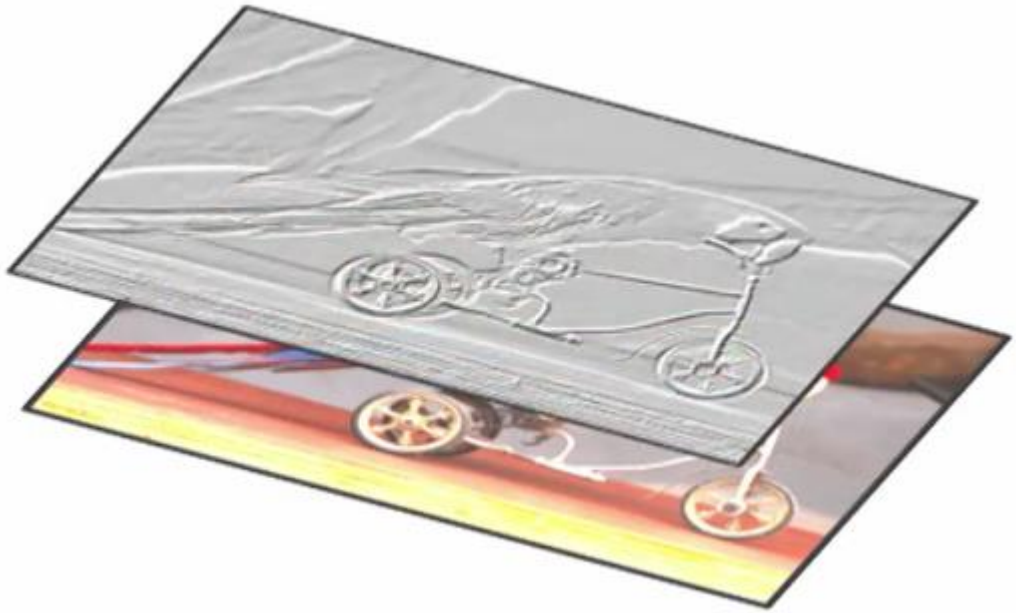


Image



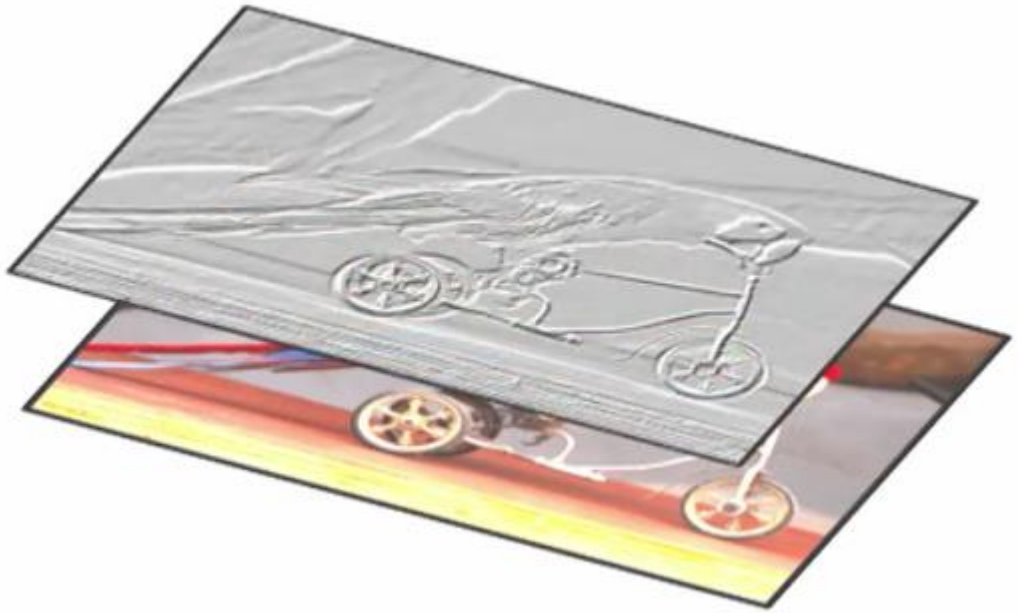
Convolution

Image



Convolution

Image



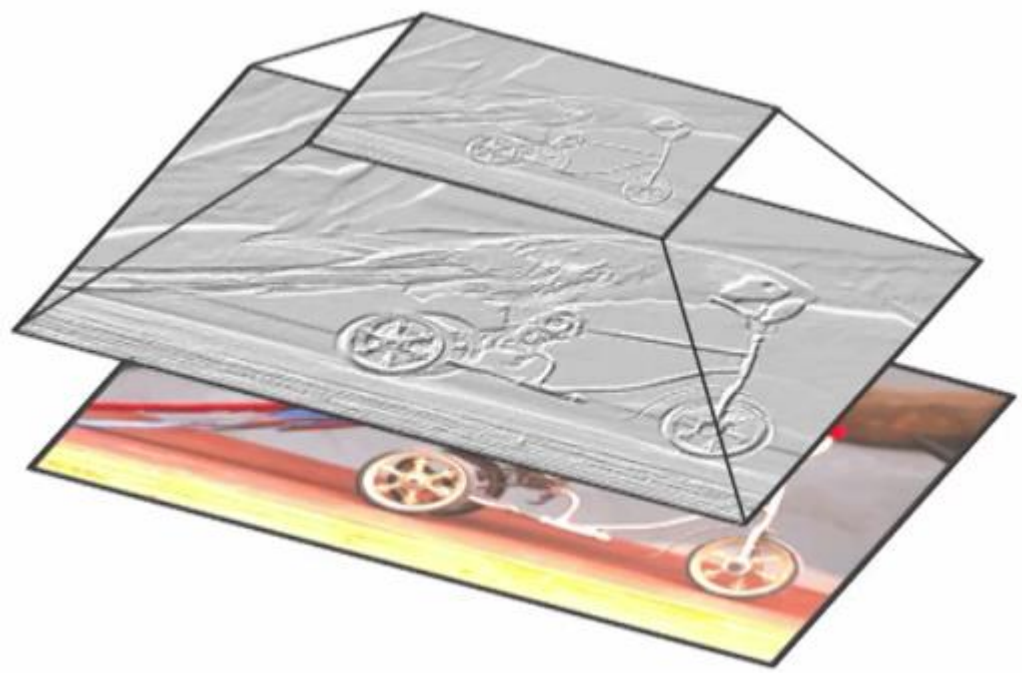
Pooling

↑

Convolution

↑

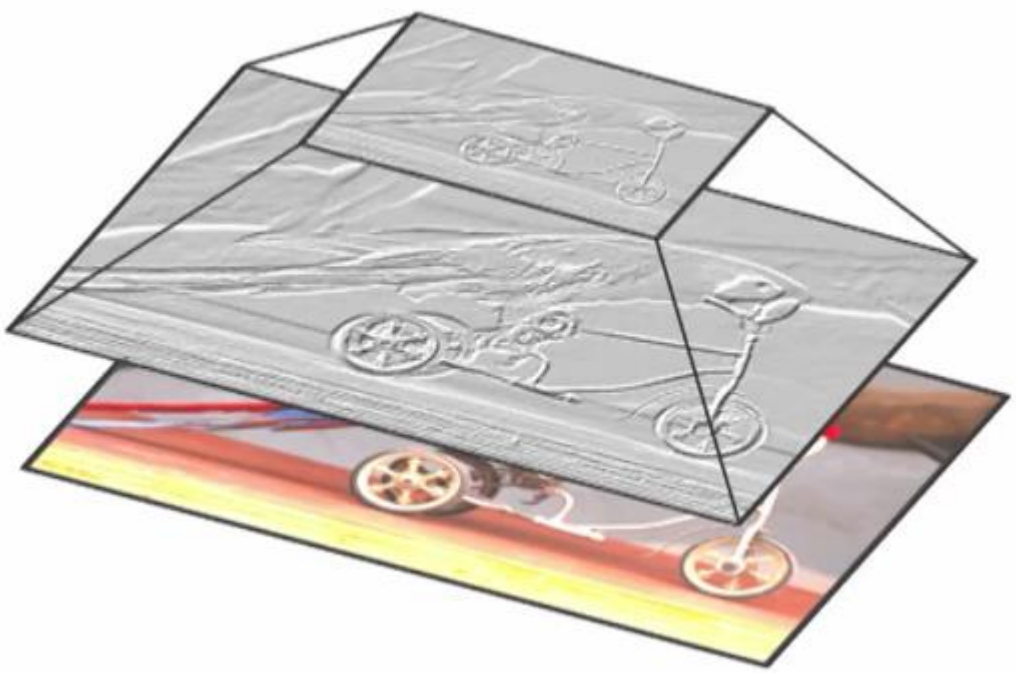
Image



Pooling

Convolution

Image



Nonlinearity

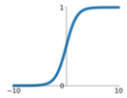
Pooling

Convolution

Image

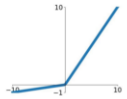
Sigmoid

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



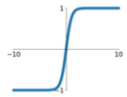
Leaky ReLU

$\max(0.1x, x)$



tanh

$\tanh(x)$

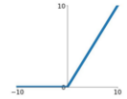


Maxout

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

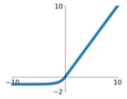
ReLU

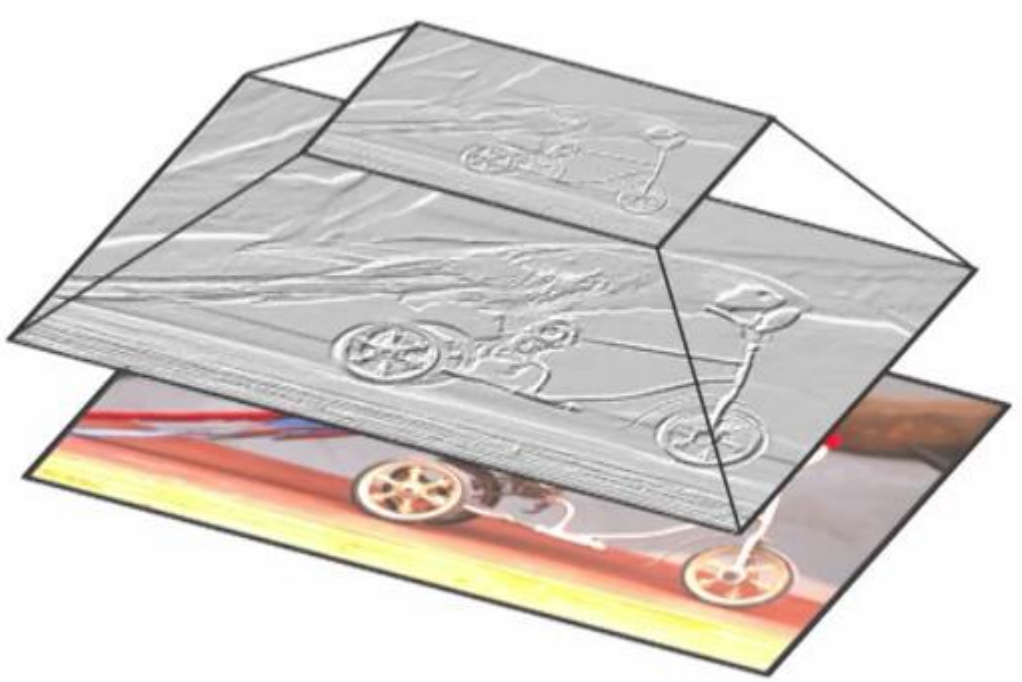
$\max(0, x)$



ELU

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





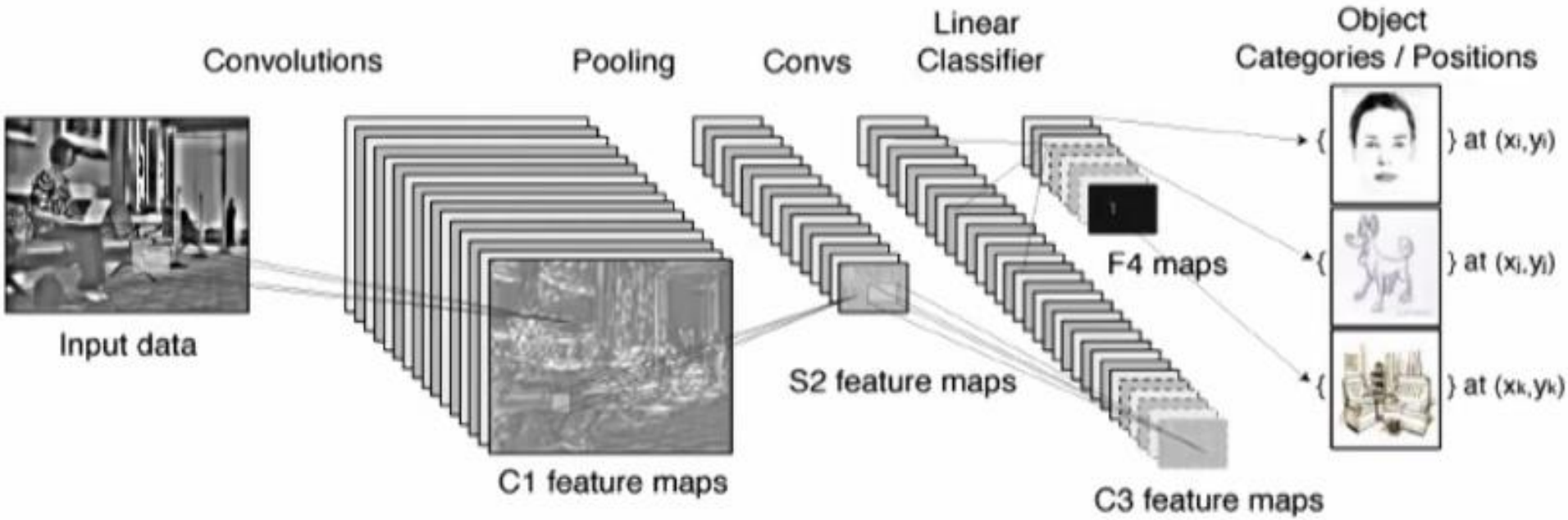
Nonlinearity

Pooling

Convolution

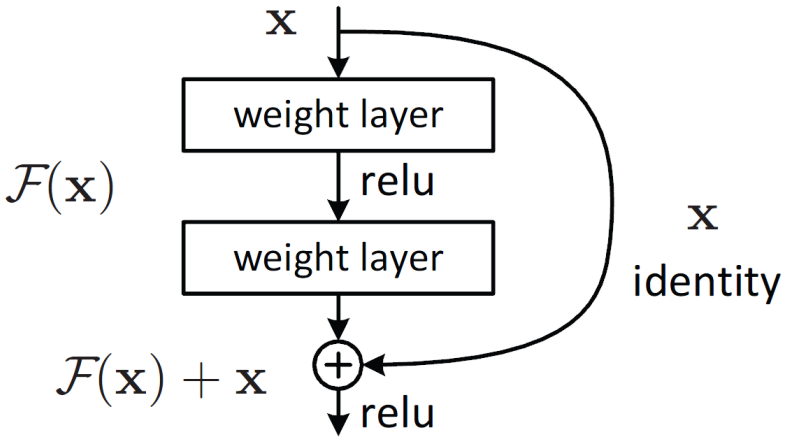
Image

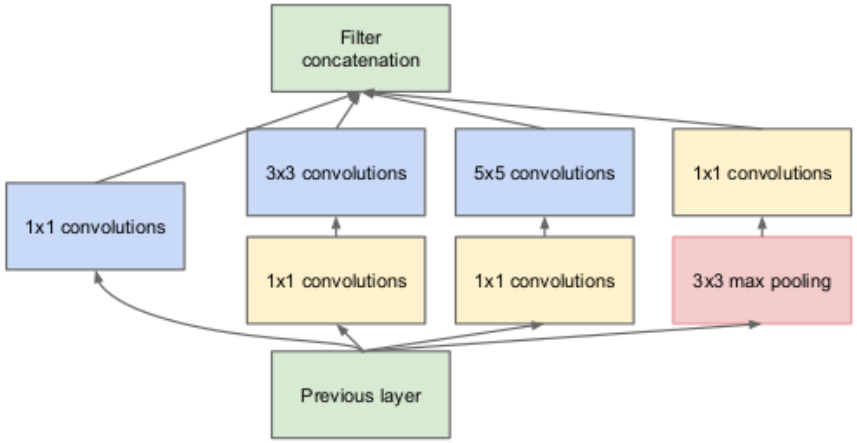
Forward: Filter, subsample, filter, nonlinearity, subsample, …., classify

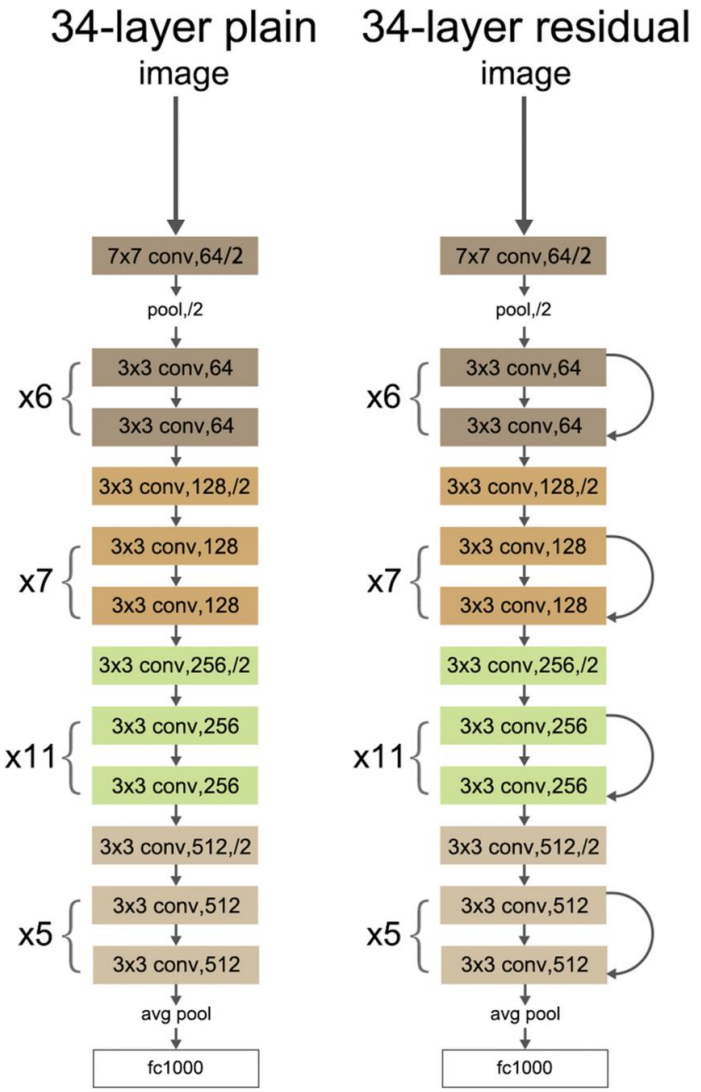


Backward: backpropagation (propagate error signal backward)

2 Year Ago

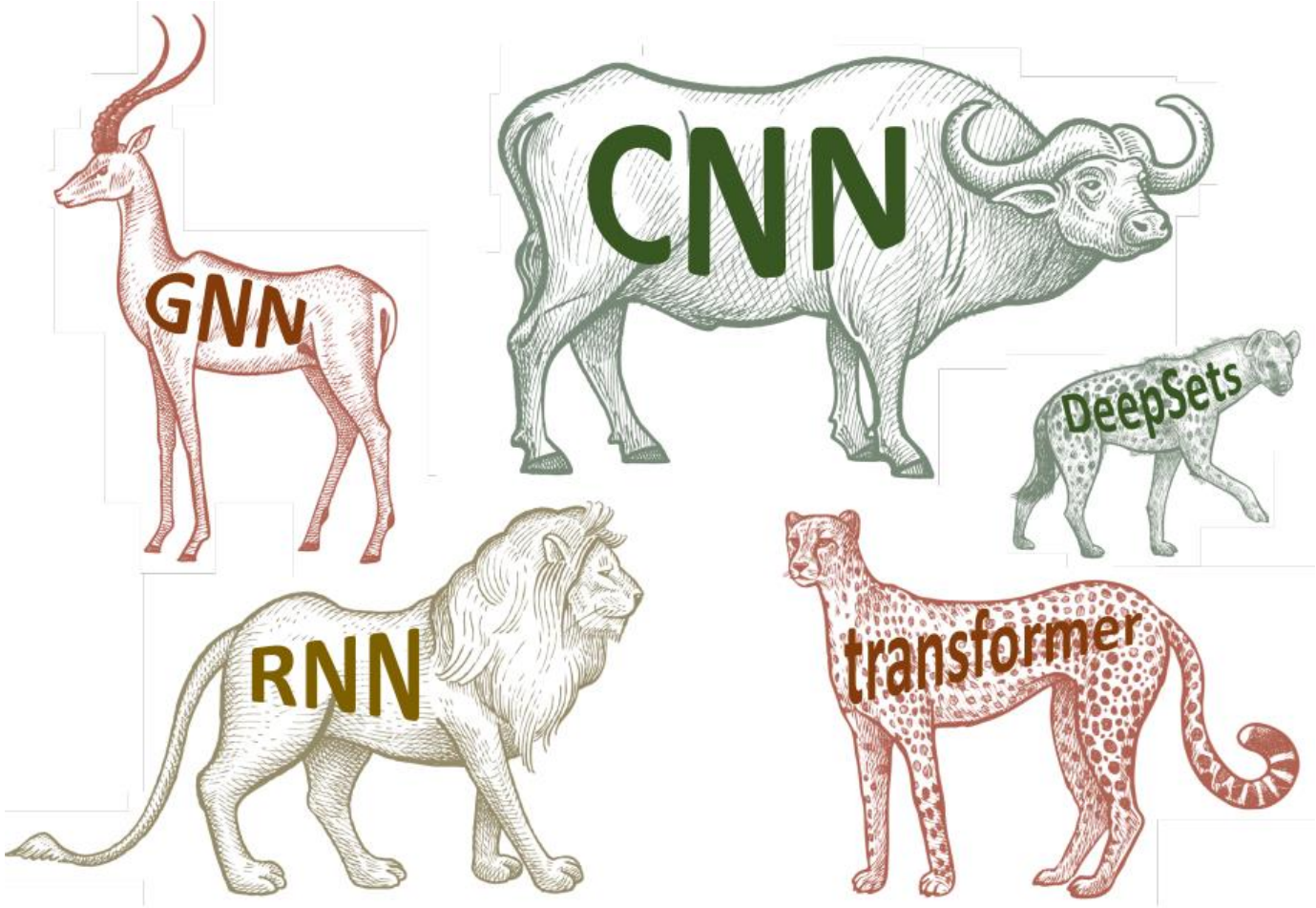

x
weight layer
relu
weight layer
F(x)
x
identity
F(x) + x
relu


Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

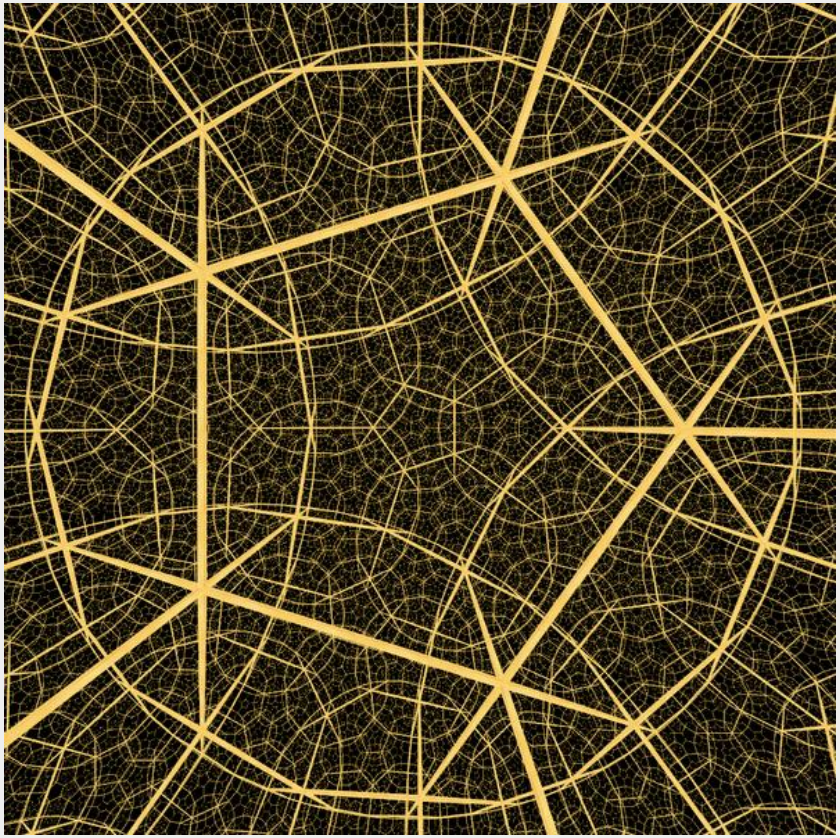

34-layer plain
image
7x7 conv,64/2
pool,/2
x6
3x3 conv,64
3x3 conv,64
3x3 conv,128,/2
x7
3x3 conv,128
3x3 conv,128
3x3 conv,256,/2
x11
3x3 conv,256
3x3 conv,256
3x3 conv,512,/2
x5
3x3 conv,512
3x3 conv,512
avg pool
fc1000
34-layer residual
image
7x7 conv,64/2
pool,/2
x6
3x3 conv,64
3x3 conv,64
3x3 conv,128,/2
x7
3x3 conv,128
3x3 conv,128
3x3 conv,256,/2
x11
3x3 conv,256
3x3 conv,256
3x3 conv,512,/2
x5
3x3 conv,512
3x3 conv,512
avg pool
fc1000

Deep Learning

Many Architectures, Few Principles

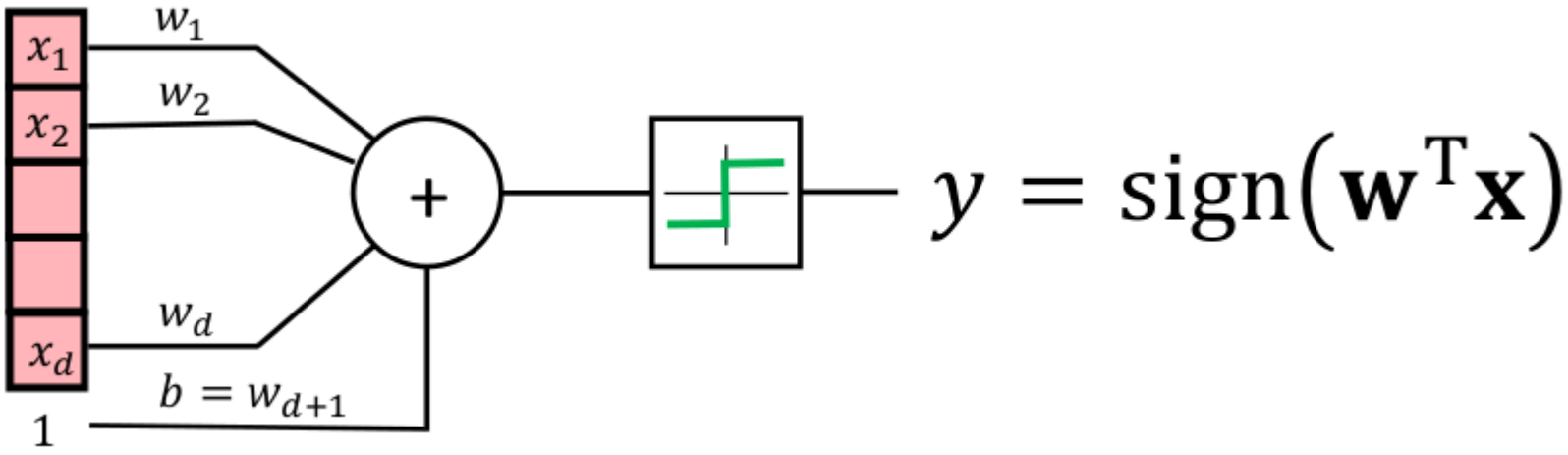


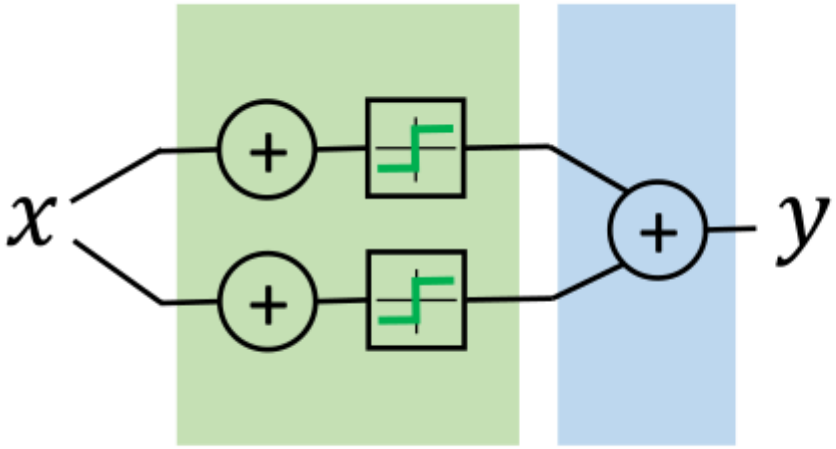
The Erlangen Programme of ML

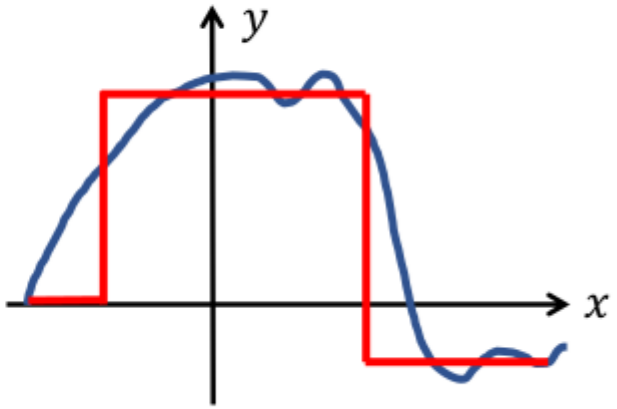


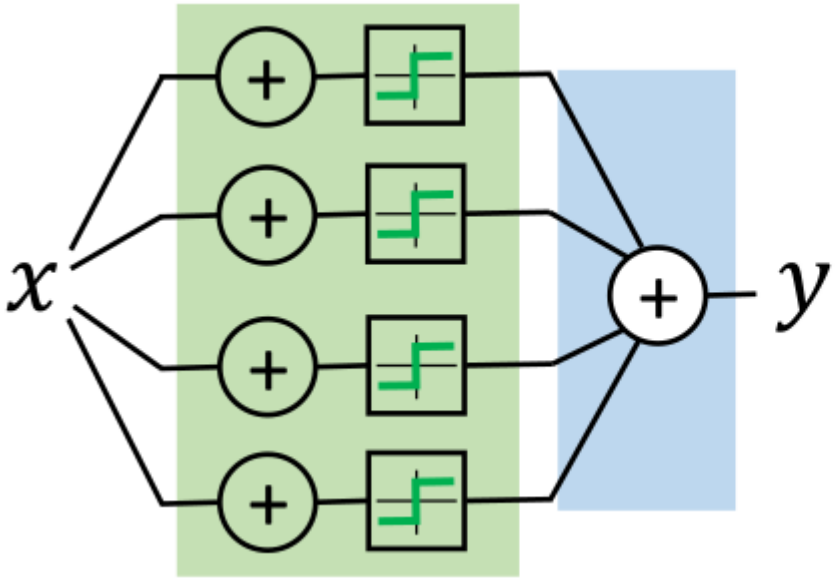

ML
black
box
{cat,dog}

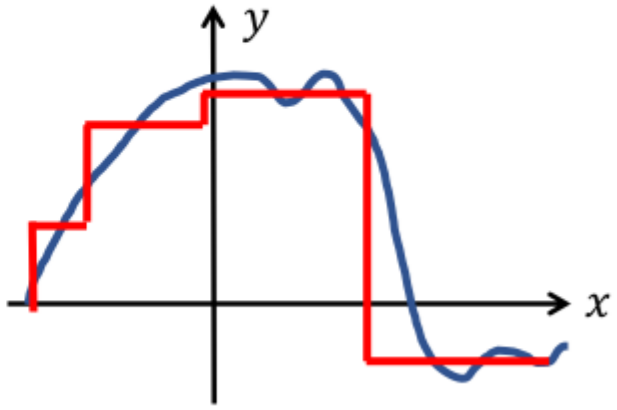
Simplest Neural Network



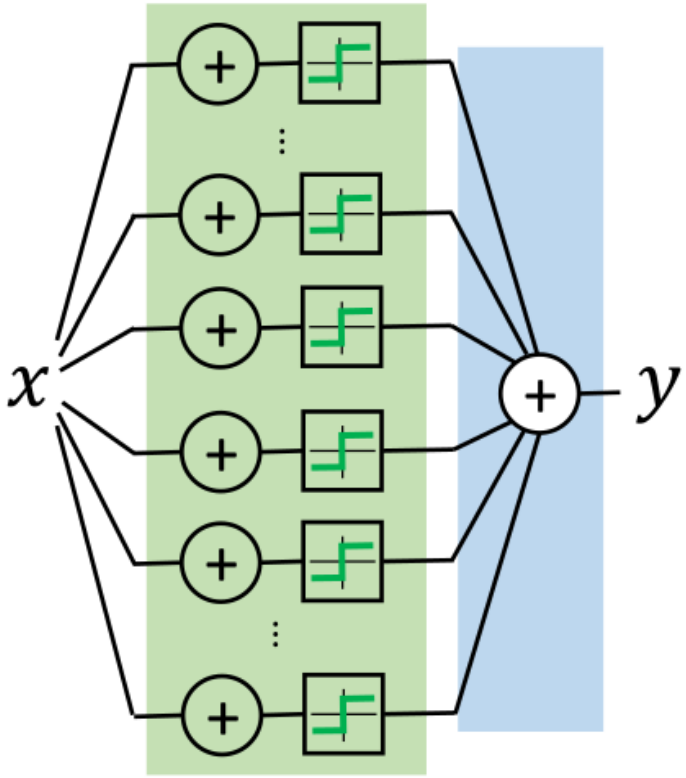

x
y

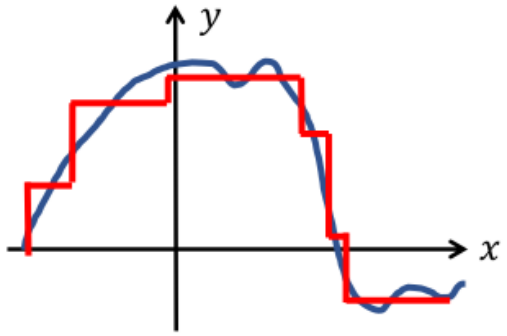

y
x


x
y


y
x

Universal Approximation



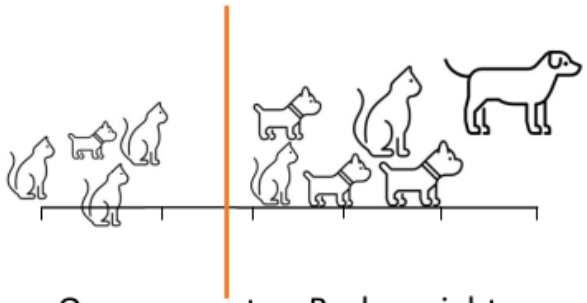


can approximate a continuous function to any desired accuracy

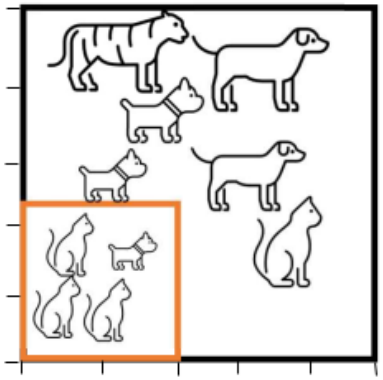
Cybenko 1989; Hornik 1991

Curse of Dimensionality

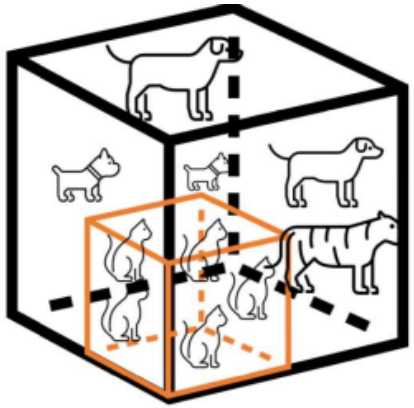
As the number of features or dimension grows, the amount of data we need to generalize accurately grows exponentially!



One parameter: Body weight or Body size, …



Two parameters: Body weight and Body size, …



Three parameters: Body weight and Body size and has a leech …

10 samples:

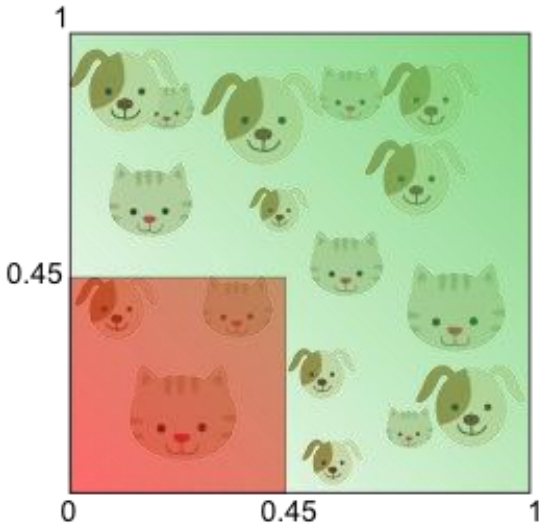
20% samples = 0.2
5 unit intervals
10/5=2 samples/interval

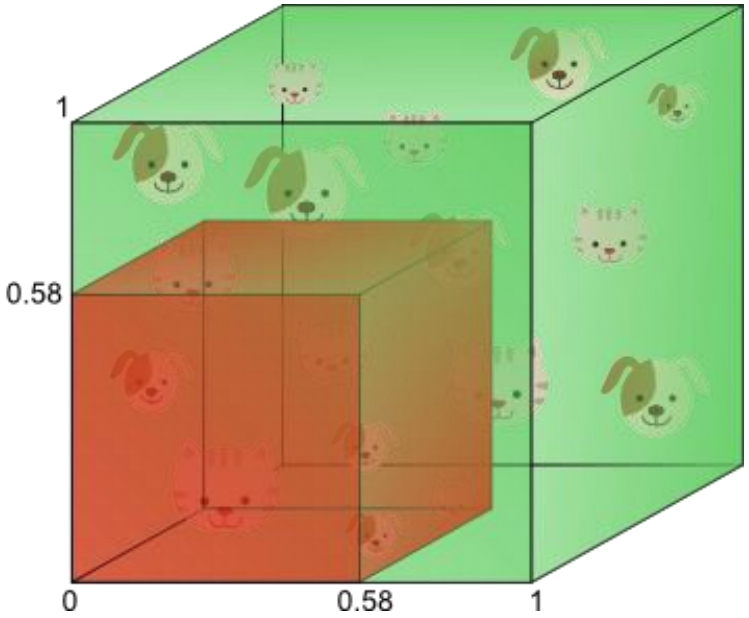
20% samples = 0.45^2
5×5=25 unit squares
10/25 = 0.4 samples/interval

20% samples = 0.58^3
5x5x5 unit intervals
10/125 = 0.08 samples/interval

https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

Curse of Dimensionality

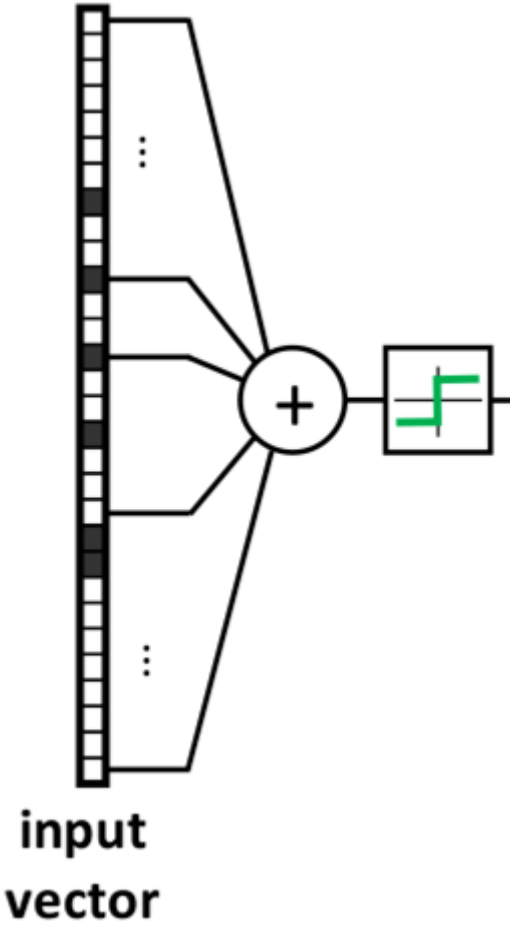




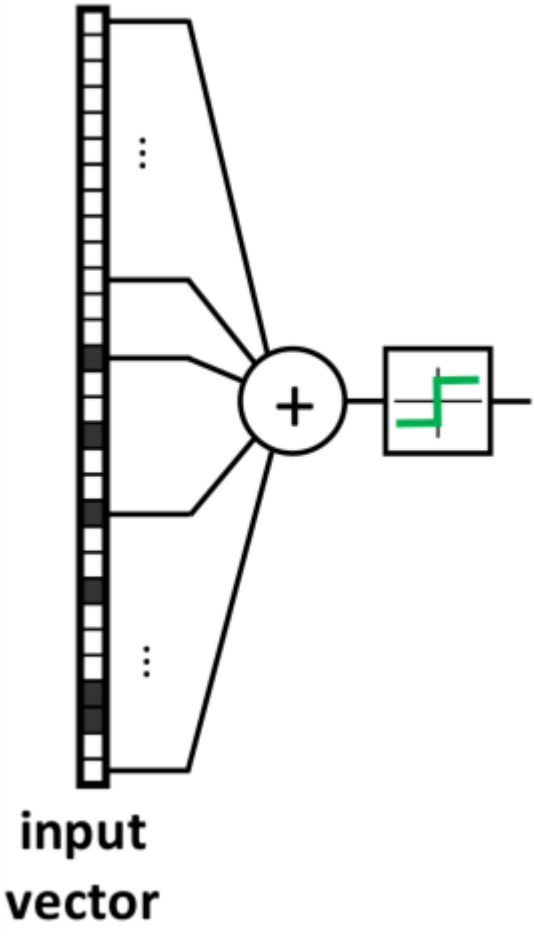
$O(\varepsilon^{-d})$ samples

https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

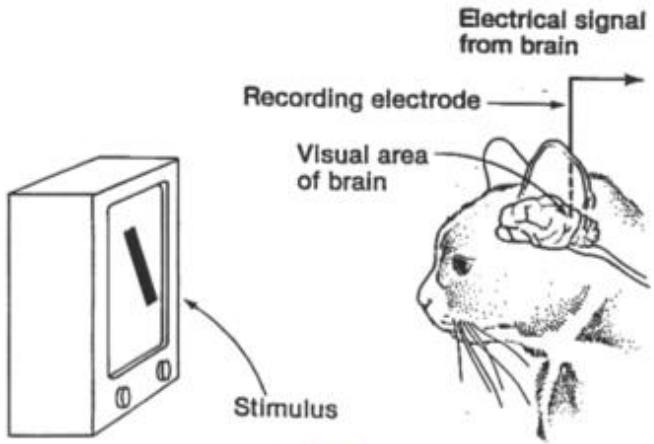
Curse of Dimensionality



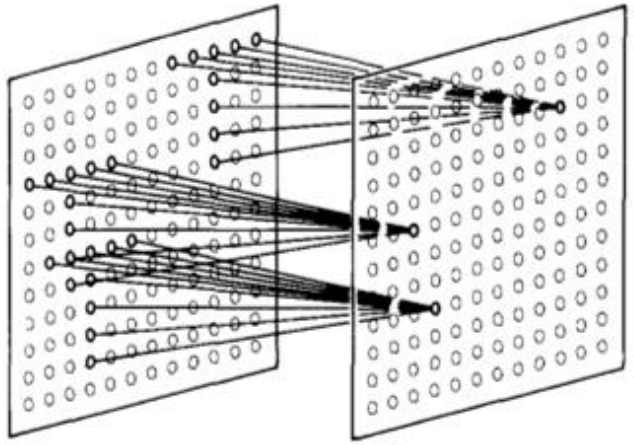
Curse of Dimensionality



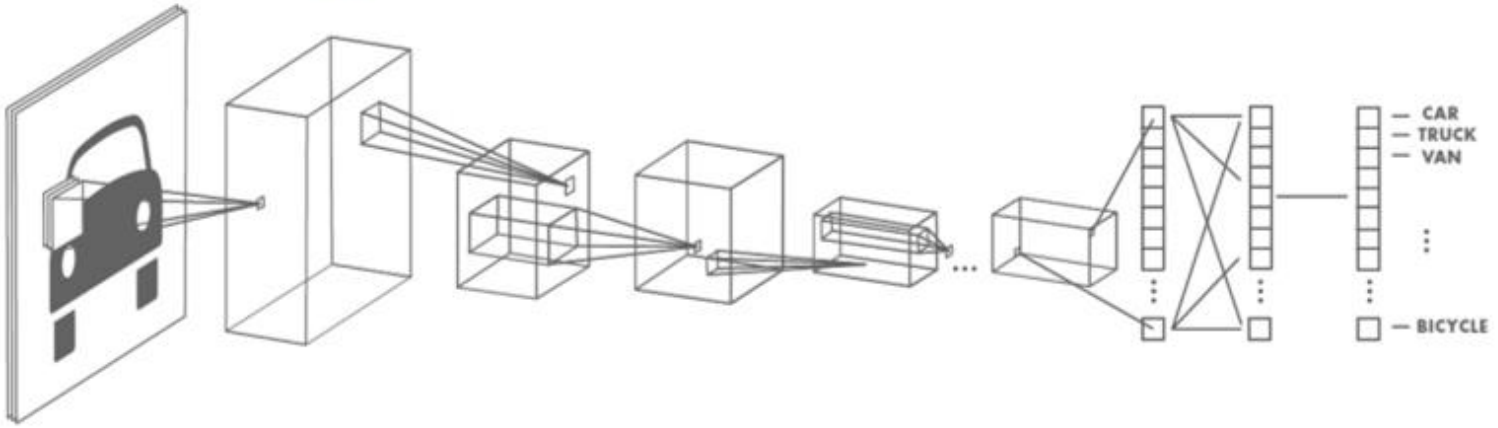
must learn shift invariance from data!



Hubel, Wiesel 1962;  1981



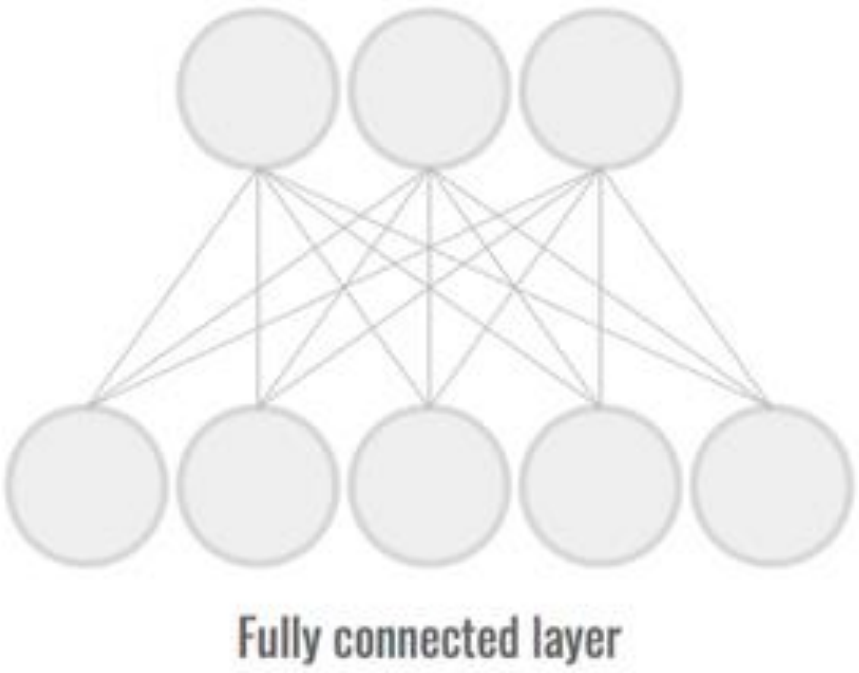
Fukushima 1980



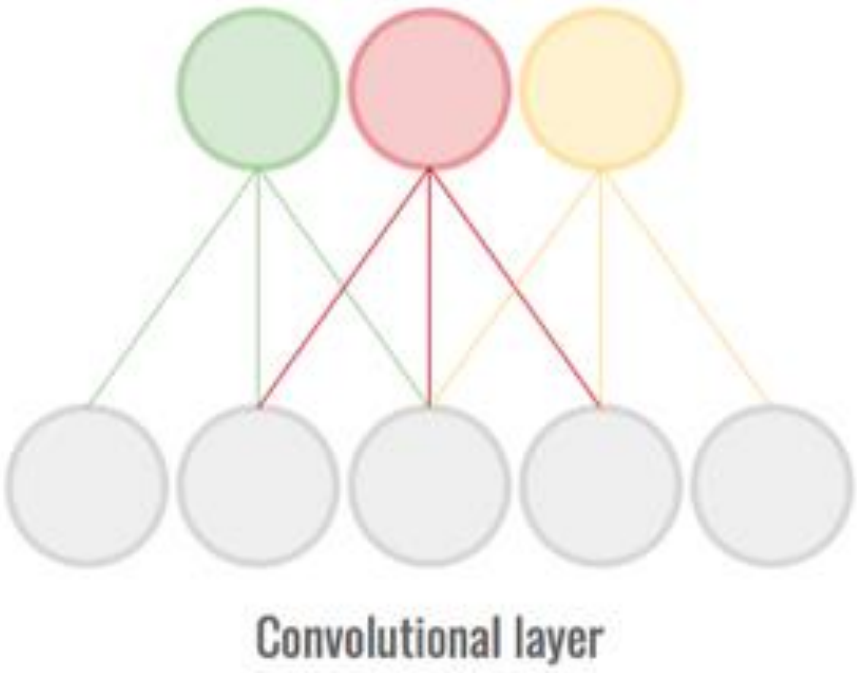
LeCun et al. 1989

Convolution

1. Local filters (local receptive field connectivity)



Fully connected layer

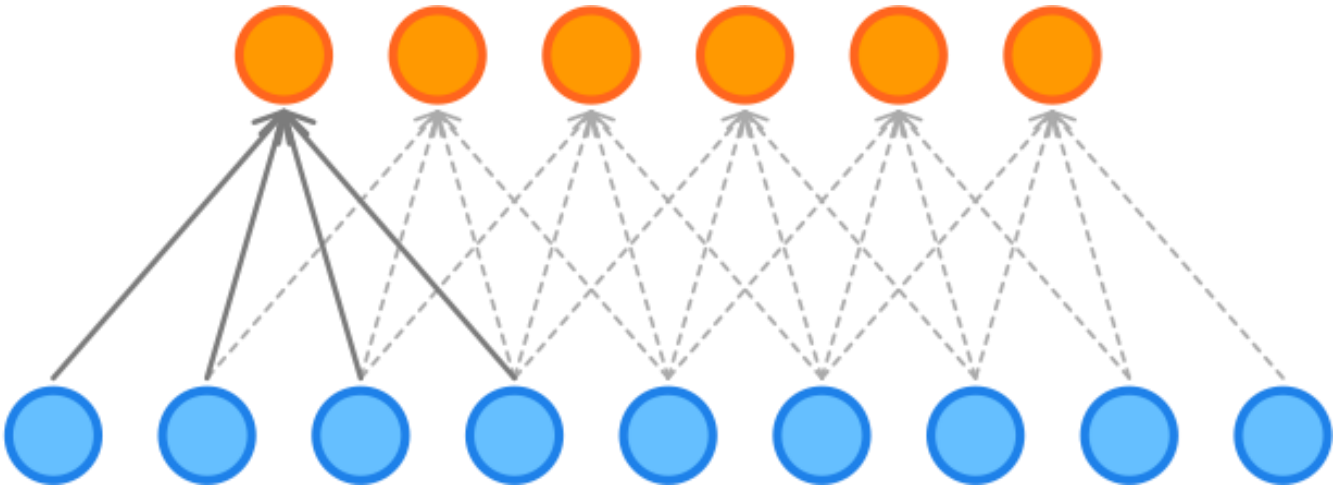


Convolutional layer

Convolution

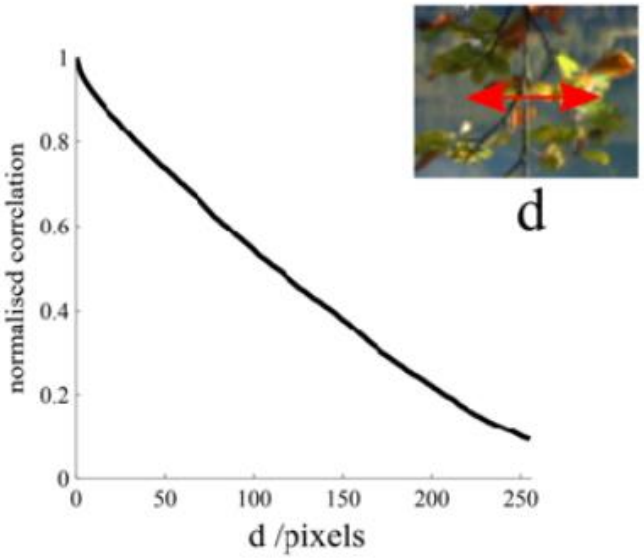
1. Local filters (local receptive field connectivity)

```
kernel_size = 4
strides = 1
padding = 'valid'
```



Convolution

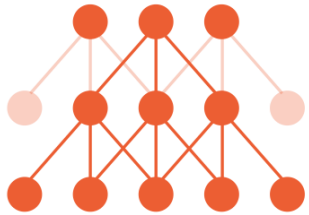
1. Local filters (local receptive field connectivity)



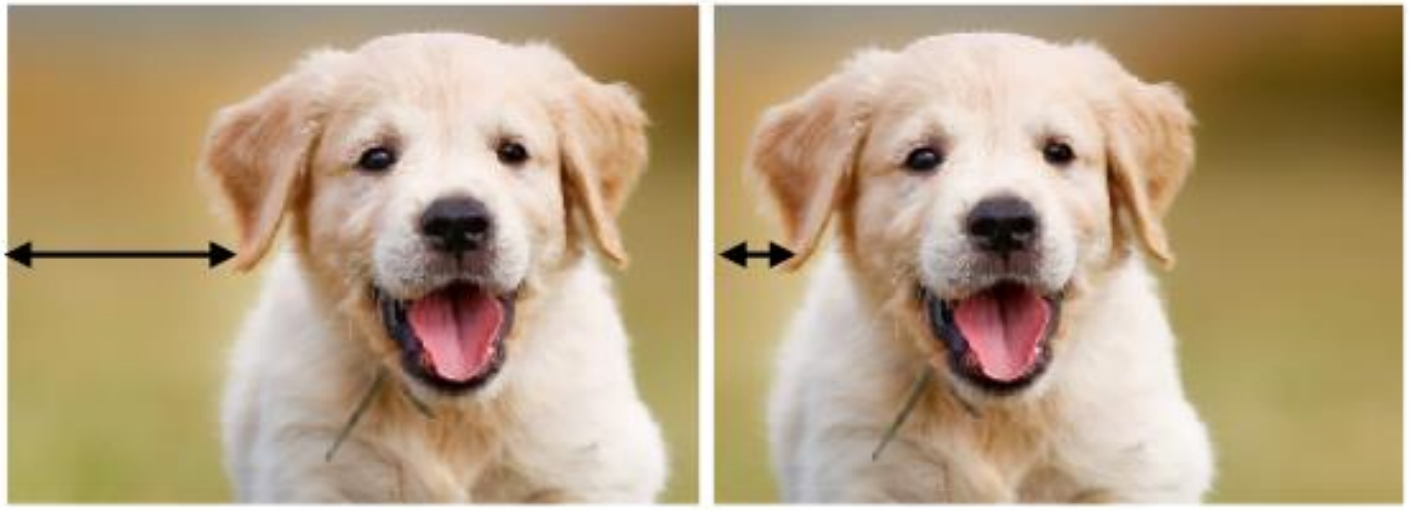
Locality of pixel statistics (Property of data)

http://www.gatsby.ucl.ac.uk/~turner/teaching/4g3/2013/stats-recep-field.pdf

Convolution

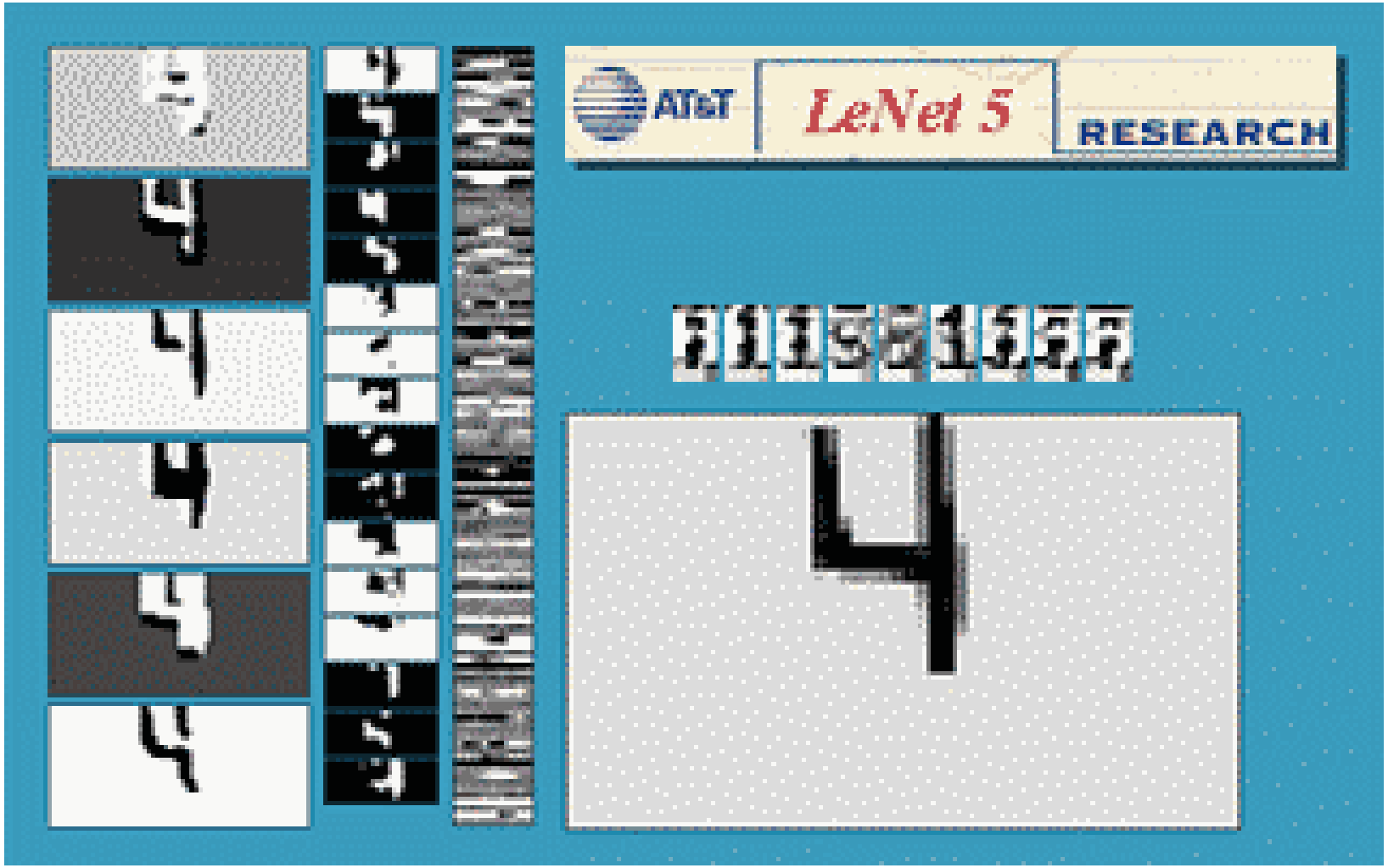


1. Local filters (local receptive field connectivity)
2. Translation weight tying (Weight sharing exploits translation symmetry)

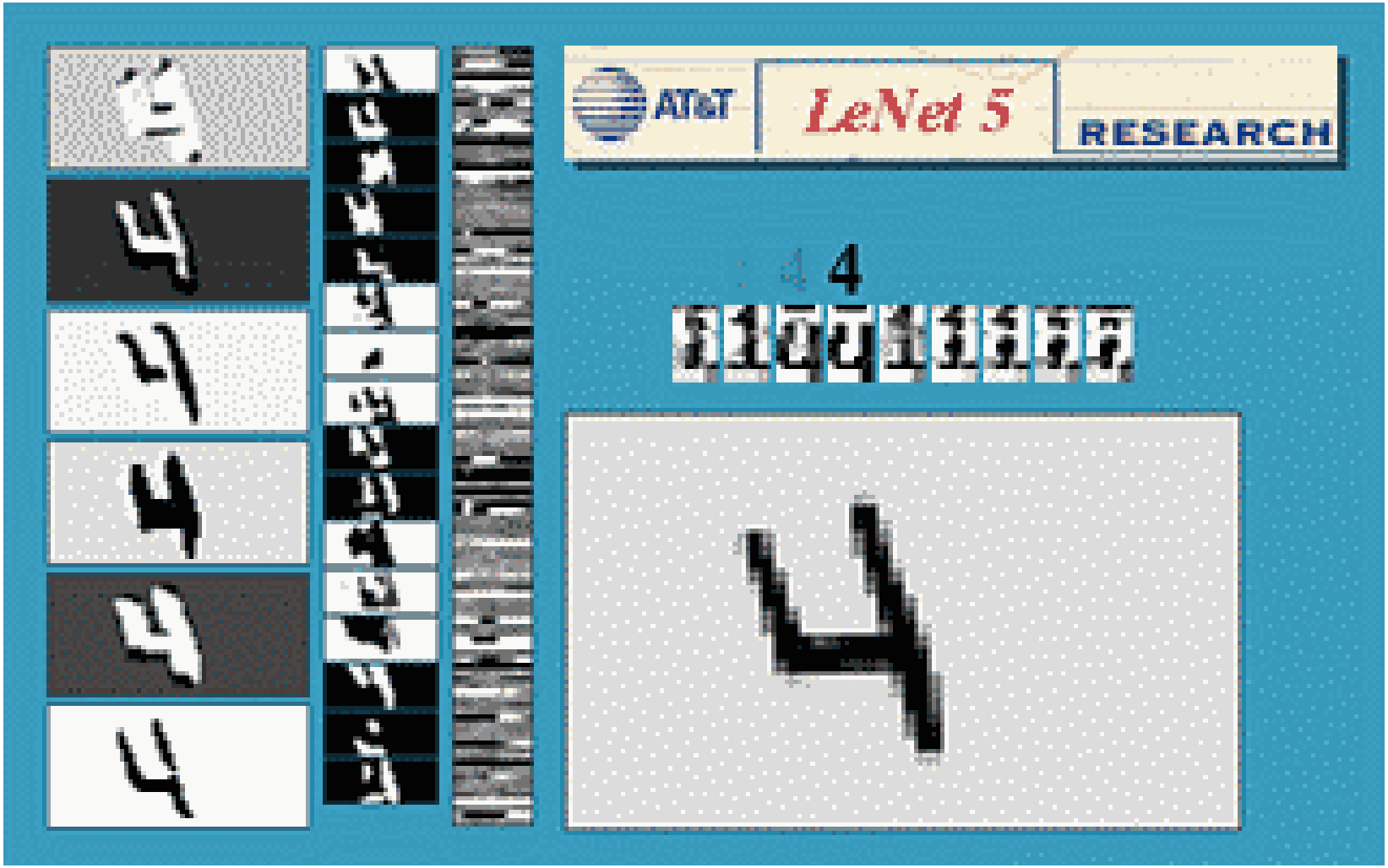


Translational invariance of classification (Property of task)

https://www.healthypawspetinsurance.com/Images/



http://yann.lecun.com/exdb/lenet/translation.html



http://yann.lecun.com/exdb/lenet/translation.html

Symmetry: Invariance

Set of input transformations leaving f invariant

$$f(\mathbf{I}) = f(\mathcal{T}_\theta[\mathbf{I}])$$

image

function/ feature mapping

transformation

Notational aside:

e.g. Geometric translation

$$\mathcal{T}_\theta[\mathbf{I}](\mathbf{x}) = \mathbf{I}(\mathbf{x} - \boldsymbol{\theta})$$

e.g. Geometric rotation

$$\mathcal{T}_\theta[\mathbf{I}](\mathbf{x}) = \mathbf{I}(\mathbf{R}_{\boldsymbol{\theta}}^{-1}\mathbf{x})$$

e.g. Pixel normalisation

$$\mathcal{T}[\mathbf{I}] = (\mathbf{I} - \boldsymbol{\mu})/.\boldsymbol{\sigma}^{-1}$$

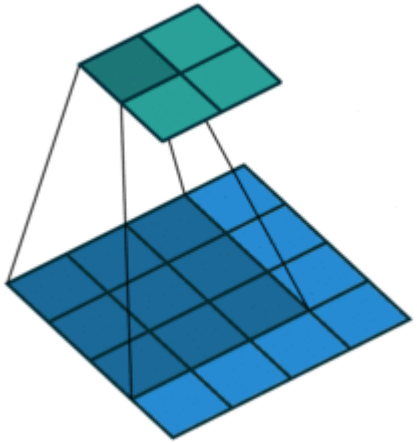
Symmetry: Equivariance

Different representations of same transformation

$$\mathcal{S}_\theta[f](\mathbf{I}) = f(\mathcal{T}_\theta[\mathbf{I}])$$

transformation in feature space

Mapping (equivariance transformations) preserves algebraic structure of transformation



Invariance

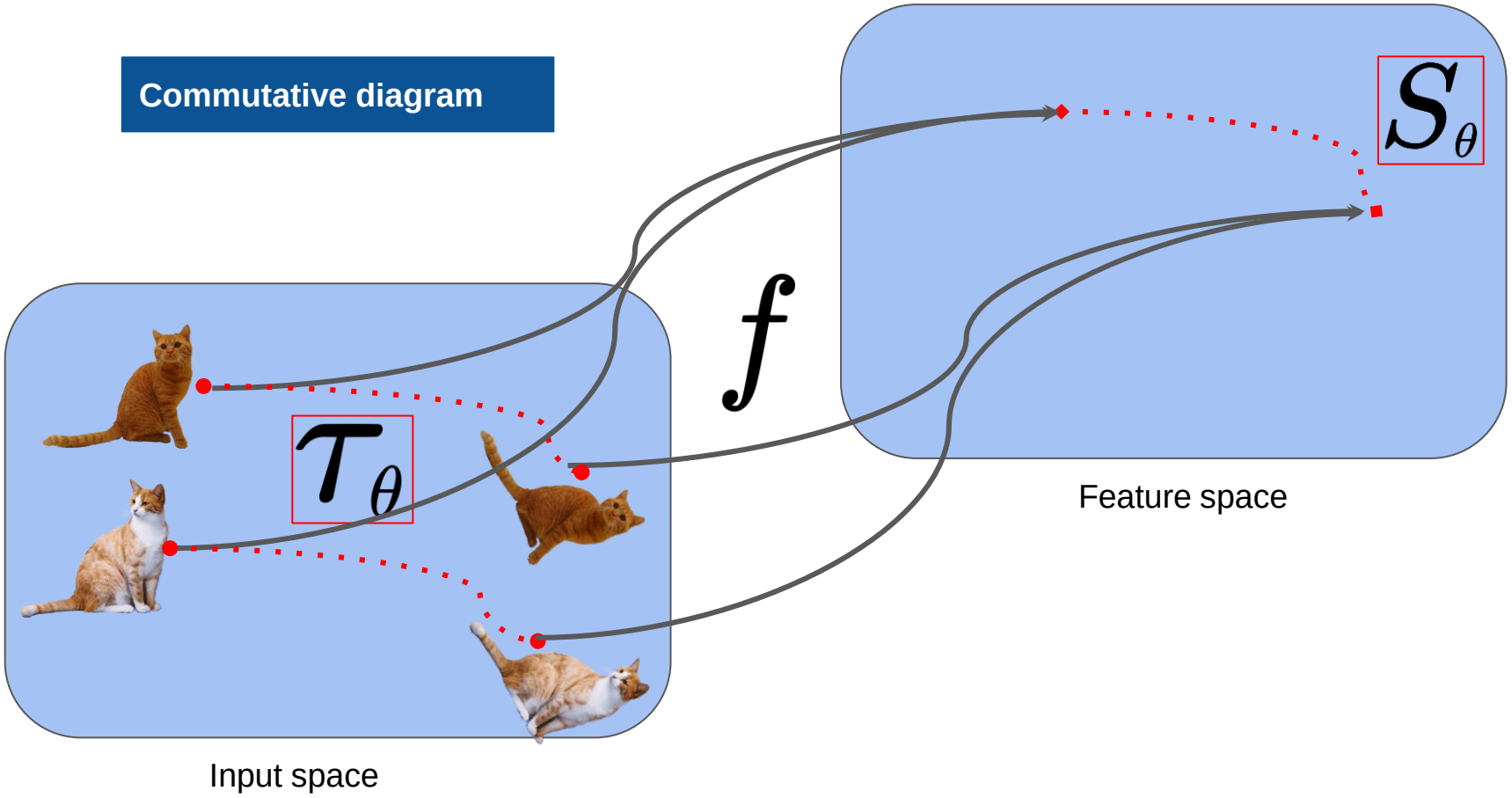
$$\mathcal{S}_\theta = \mathrm{Id}$$

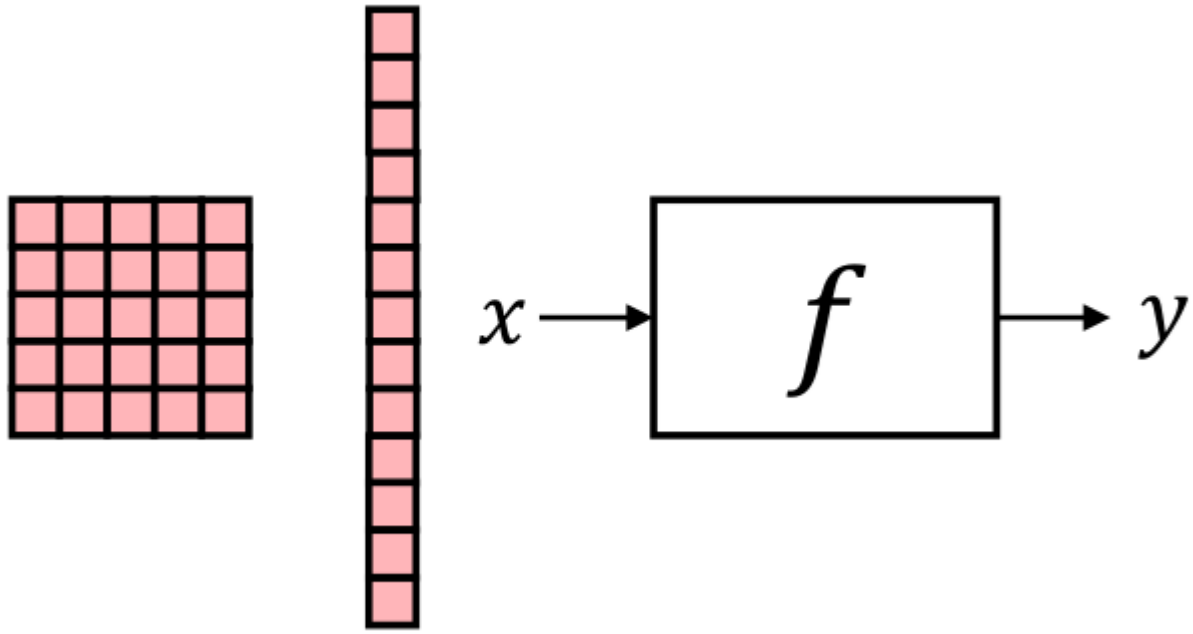
Convolution (and correlation)

$$[\mathbf{I} * \mathbf{W}](\mathbf{x} - \theta) = \mathcal{T}_\theta[\mathbf{I}] * \mathbf{W}(\mathbf{x})$$

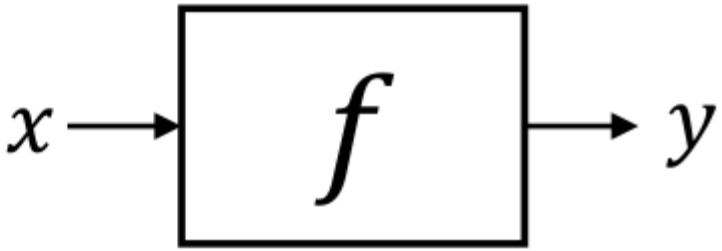
Equivariance

Commutative diagram

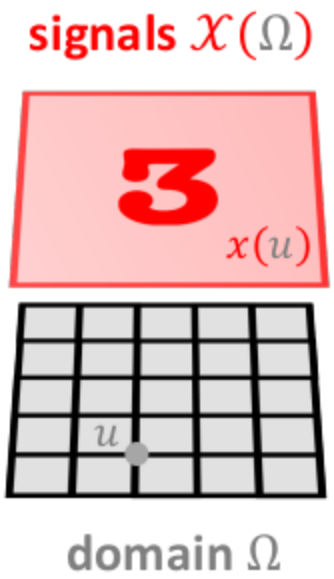


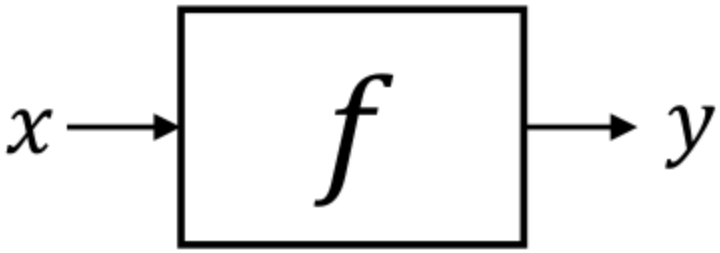

x
f
y

“geometric prior”

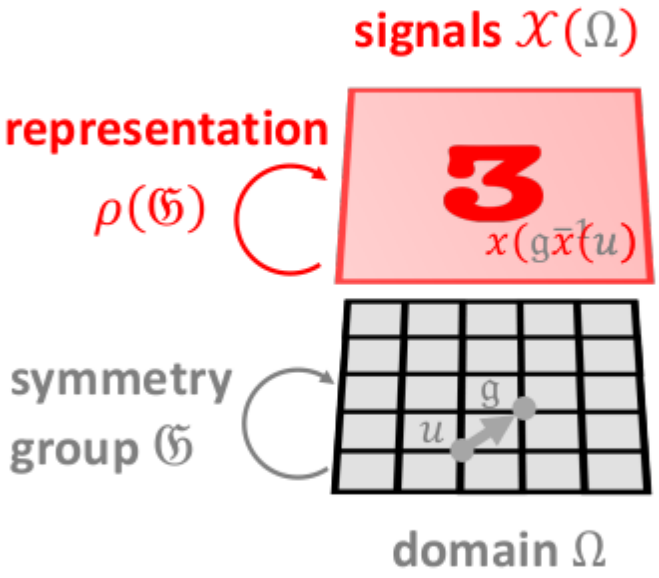


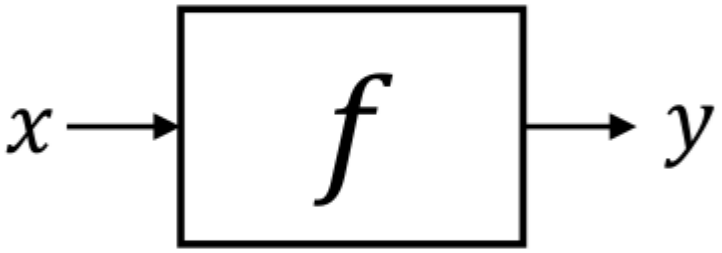
“geometric prior”





“geometric prior”





$$\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$$

$\mathfrak{G}$-invariance $f(\rho(\mathrm{g})x) = f(x)$

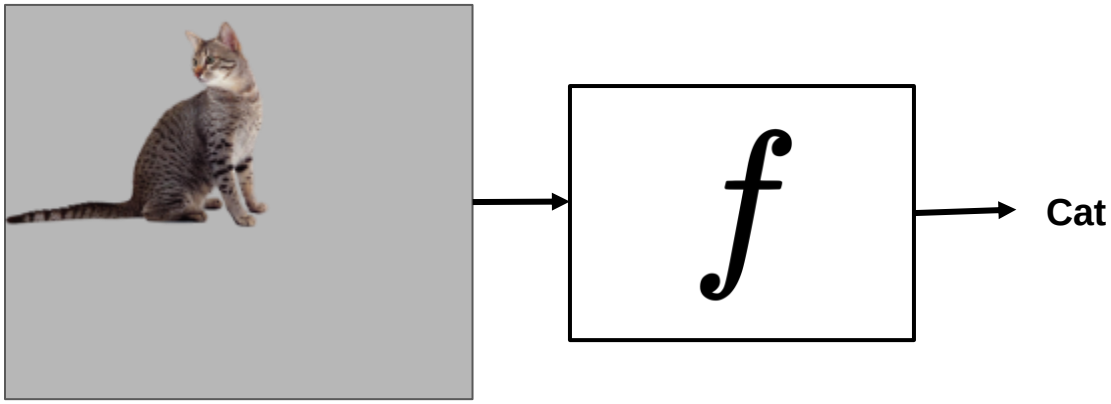


Image Classification

$\mathfrak{G}$-invariance $f(\rho(\mathrm{g})x) = f(x)$

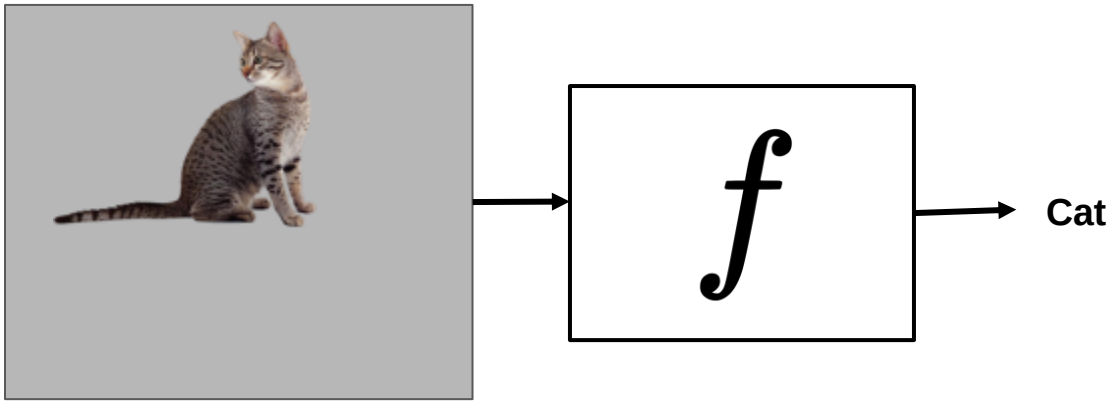


Image Classification

$\mathfrak{G}$-invariance $f(\rho(\mathrm{g})x) = f(x)$

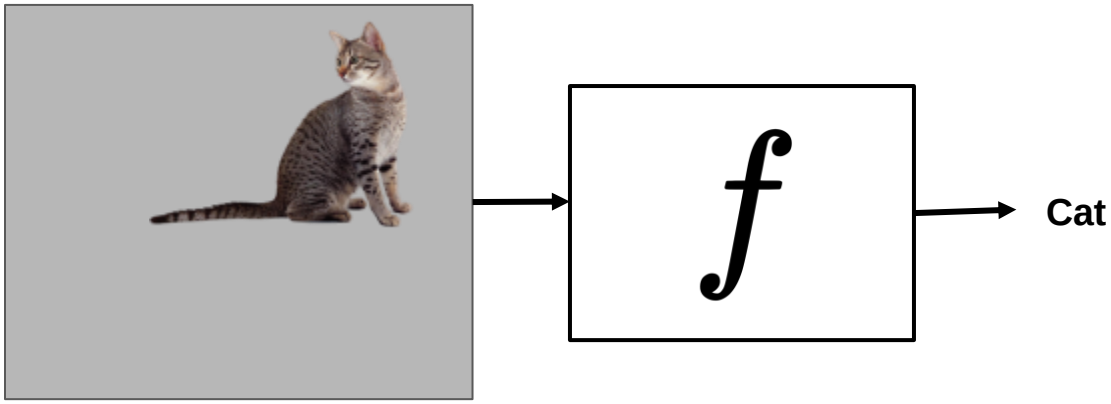


Image Classification

$\mathfrak{G}$-invariance $f(\rho(\mathrm{g})x) = f(x)$

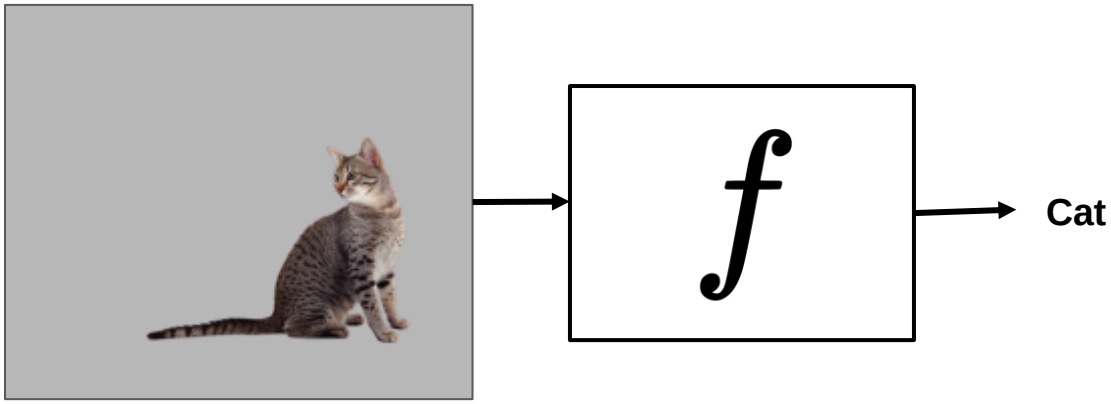


Image Classification

$\mathfrak{G}$-invariance $f(\rho(\mathrm{g})x) = f(x)$

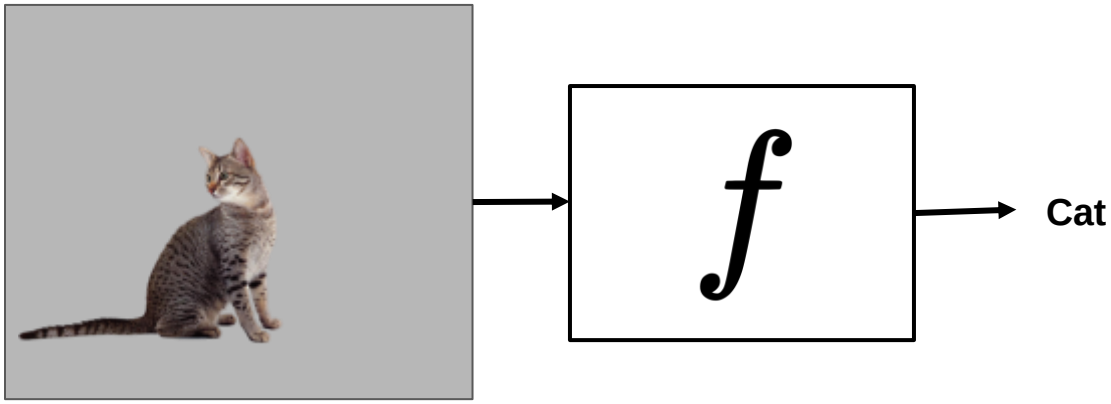


Image Classification

$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$

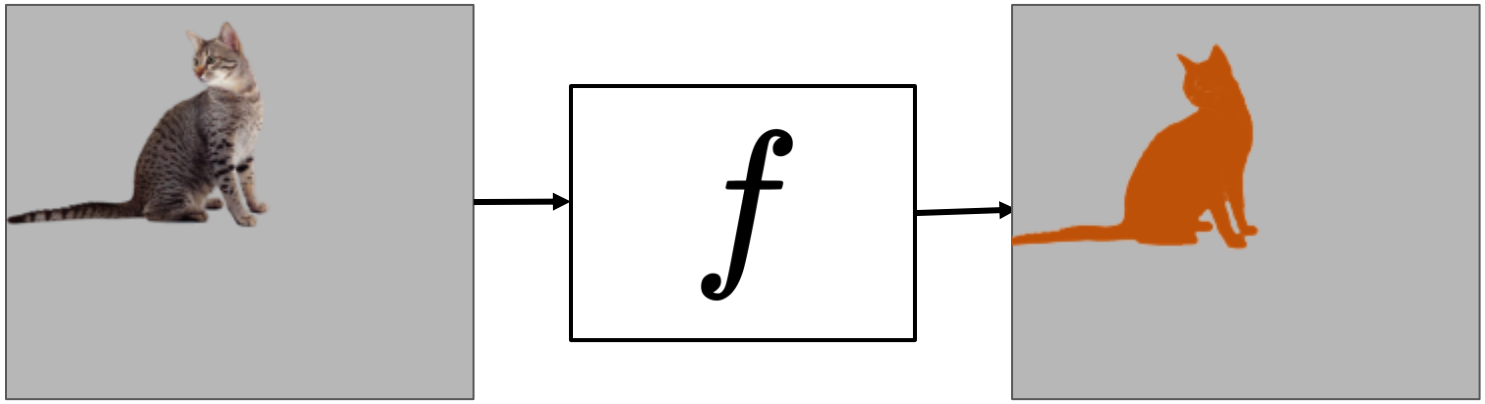


Image Segmentation

$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$

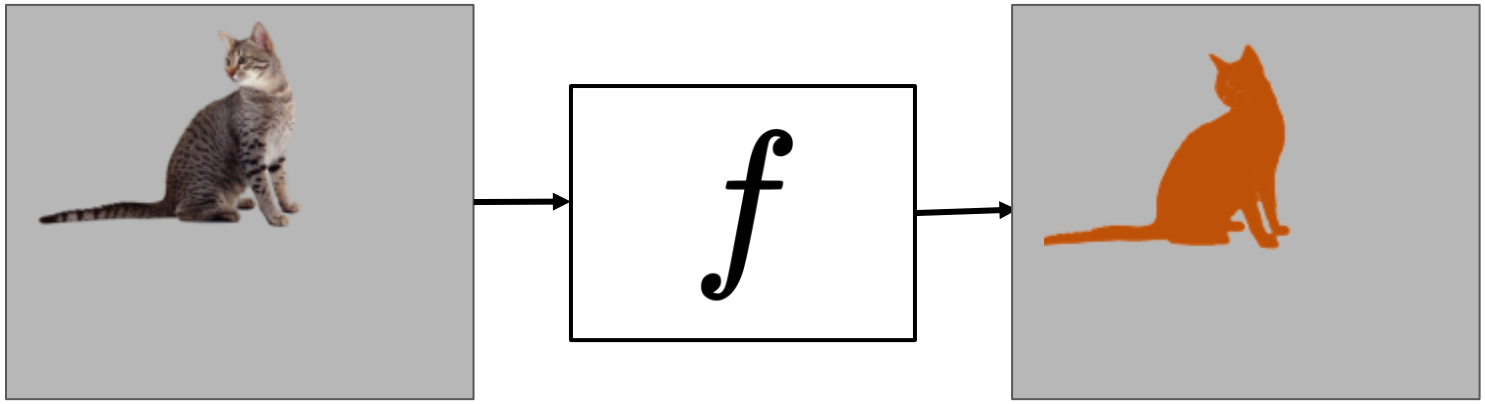


Image Segmentation

$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$

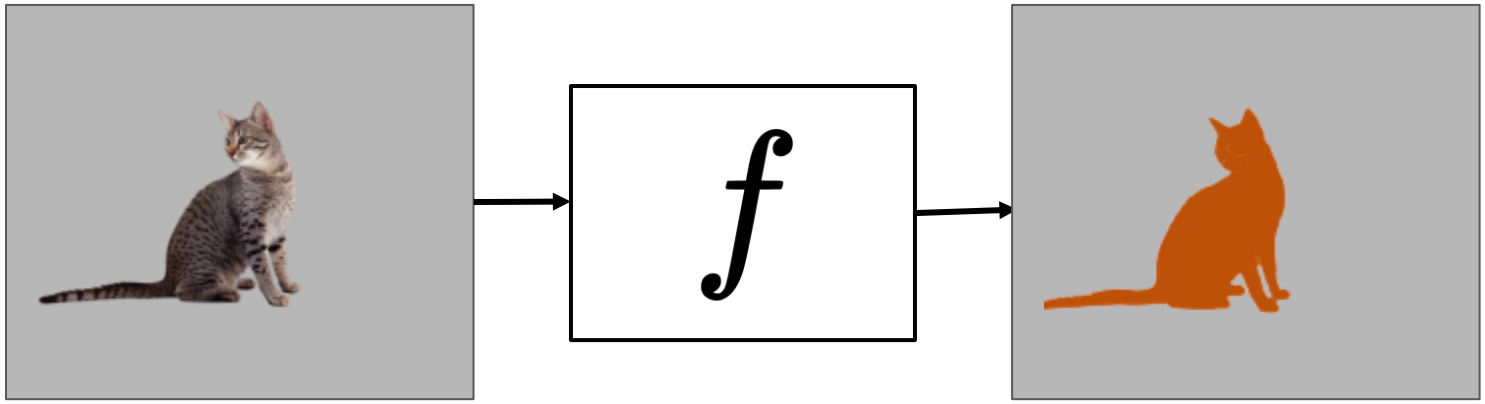


Image Segmentation

$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$

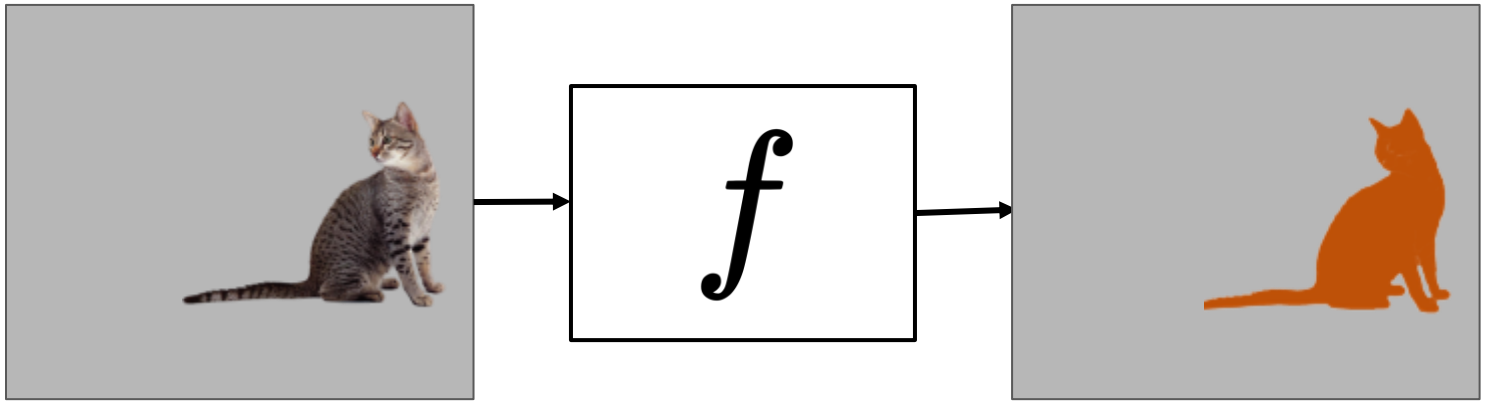
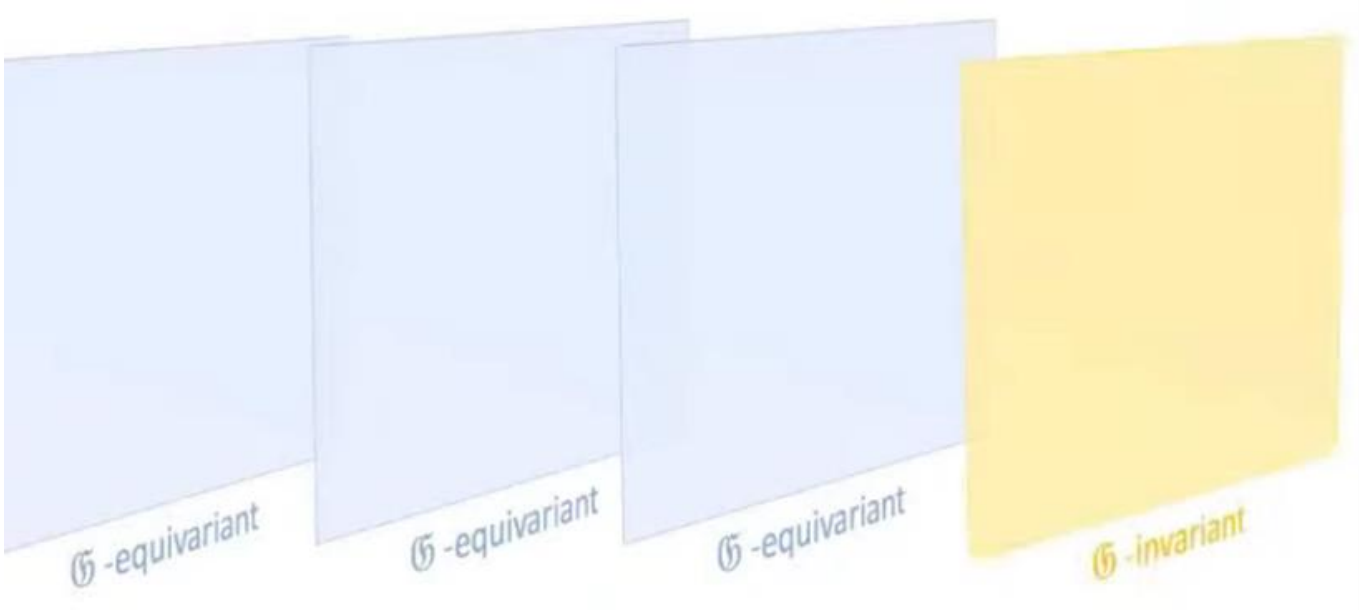


Image Segmentation

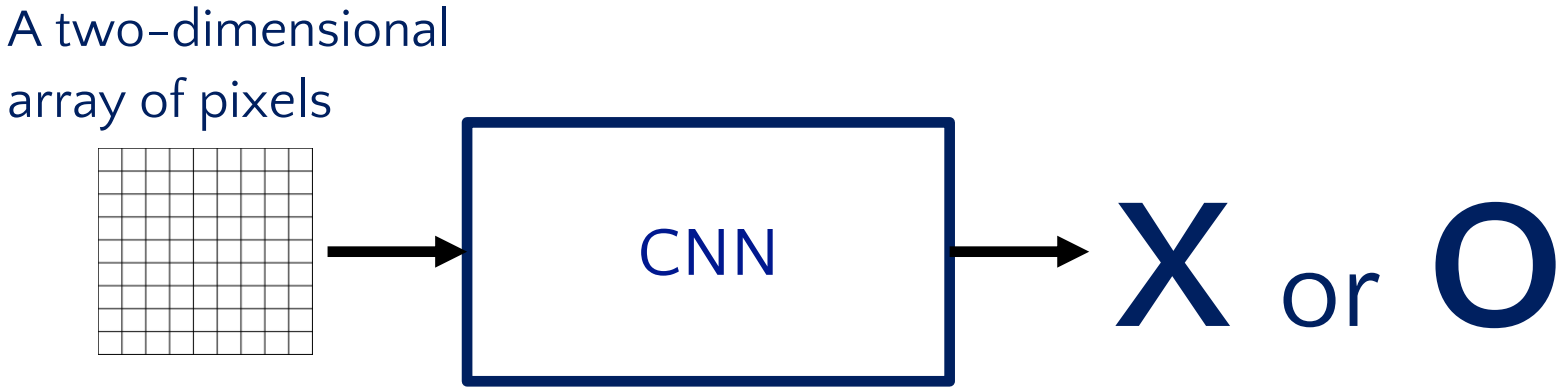


These two principles (Invariance and Equivariance) give us a very general blueprint of geometric deep learning.

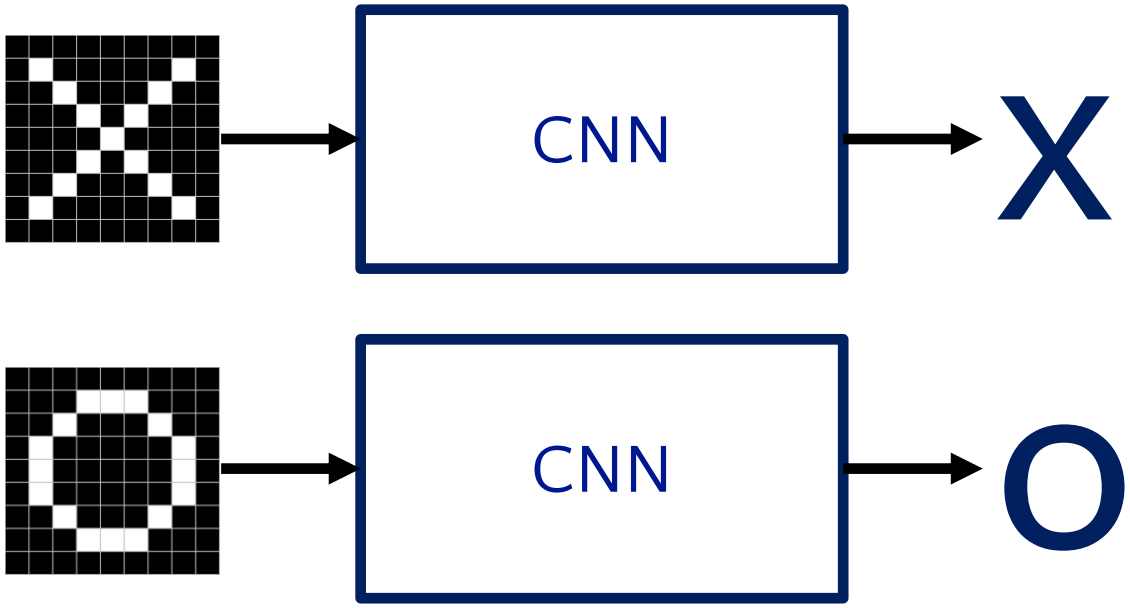
A toy ConvNet: X’s and O’s

A toy ConvNet: X's and O's

Says whether a picture is of an X or an O

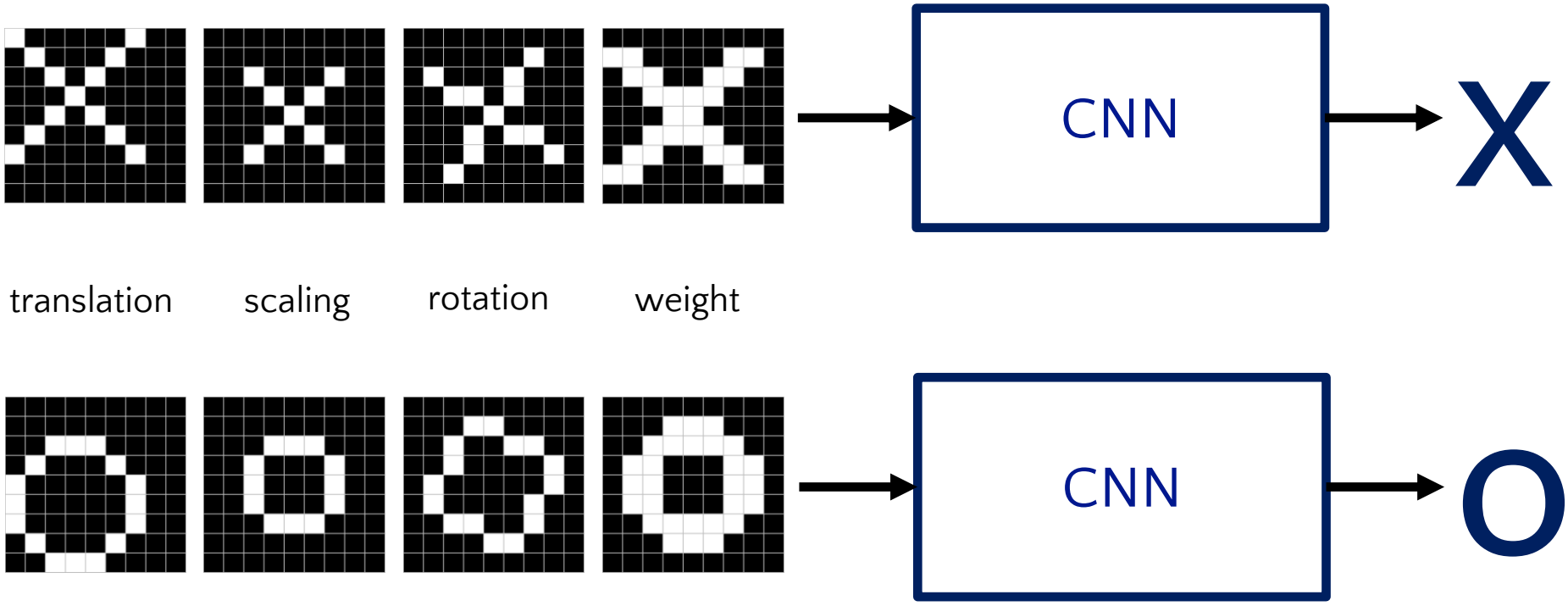


For example

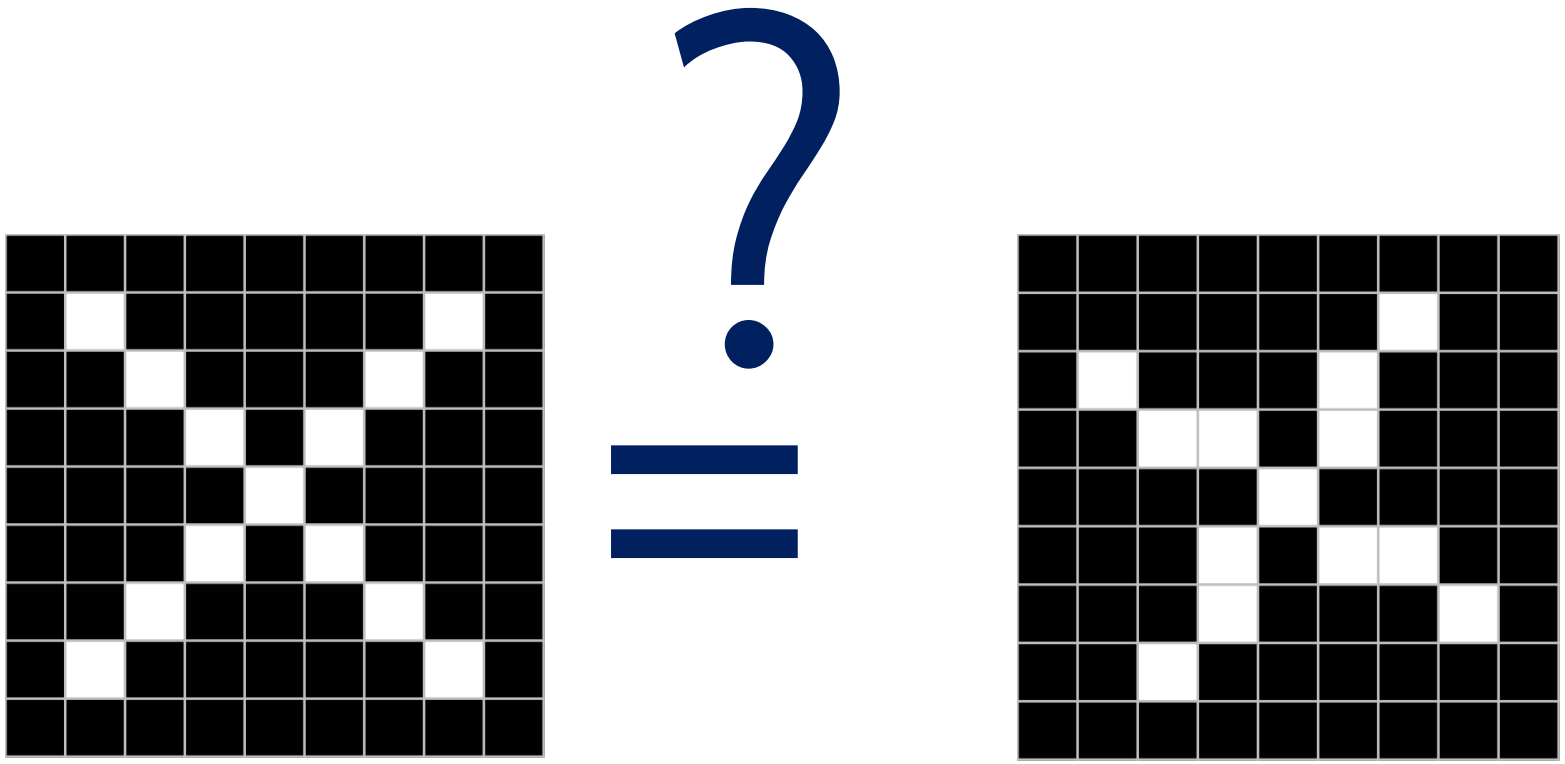


Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

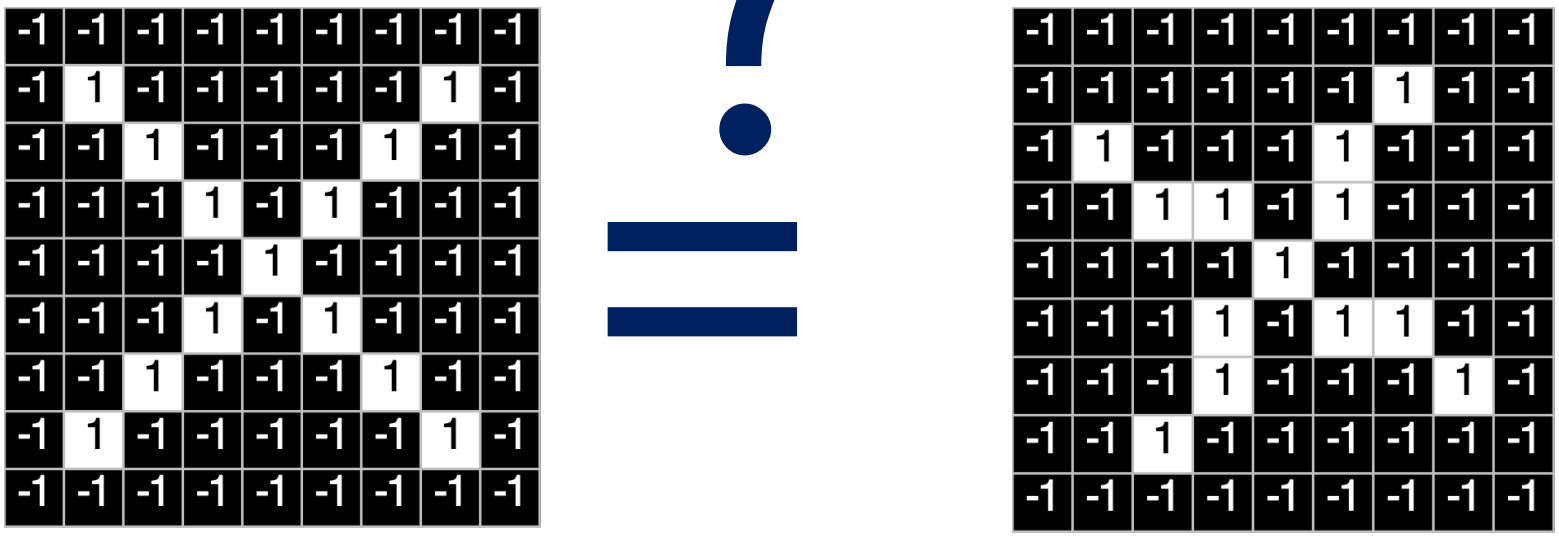
Trickier cases



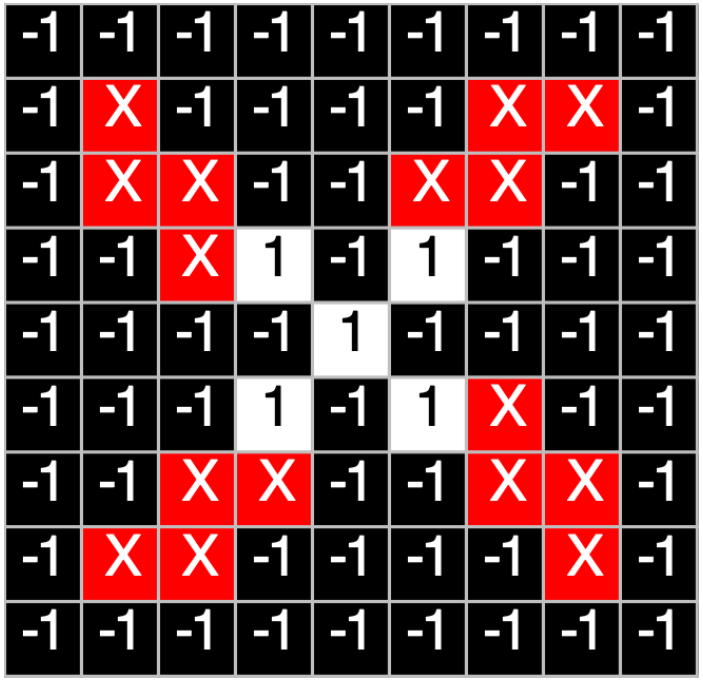
Deciding is hard



What computers see

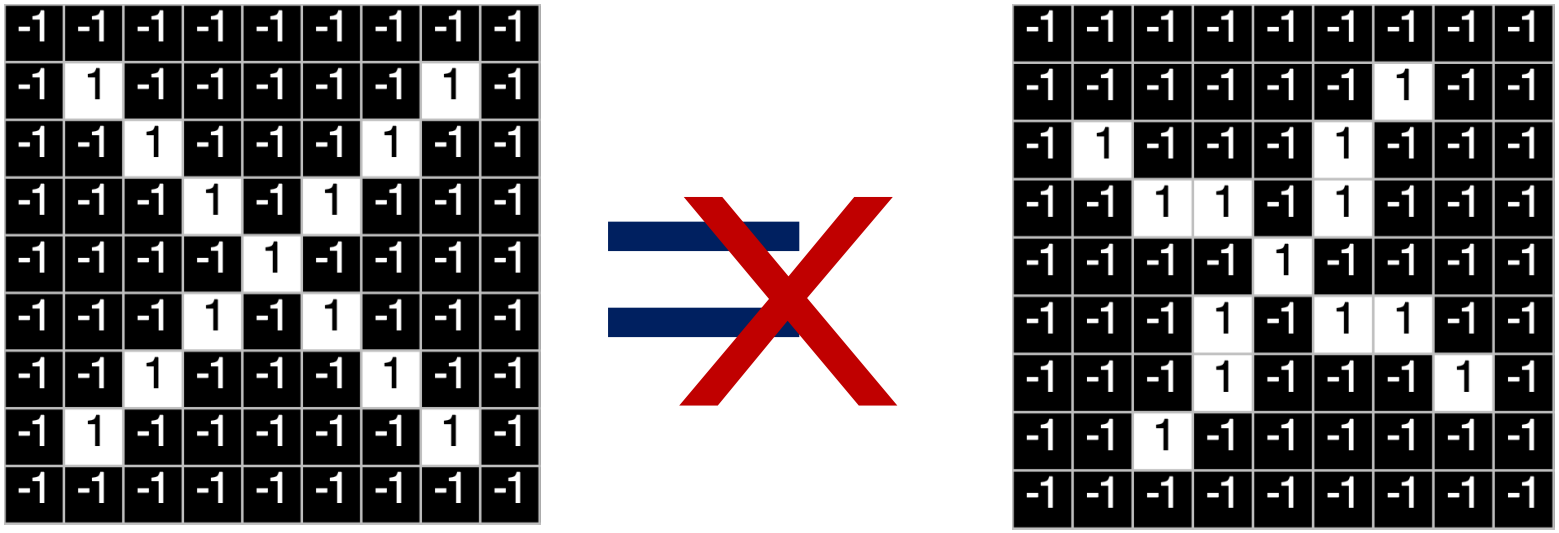


What computers see

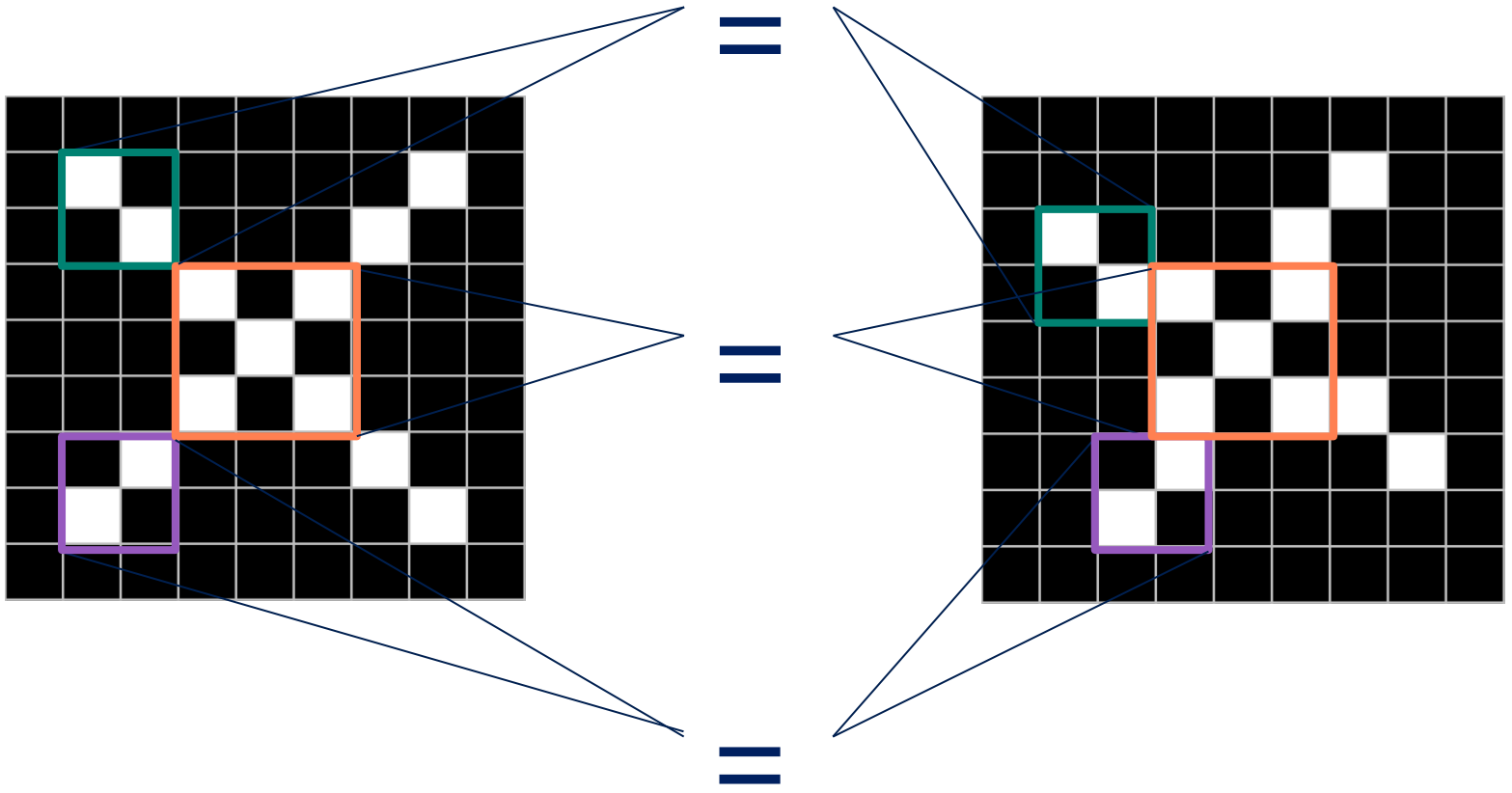


Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

Computers are literal

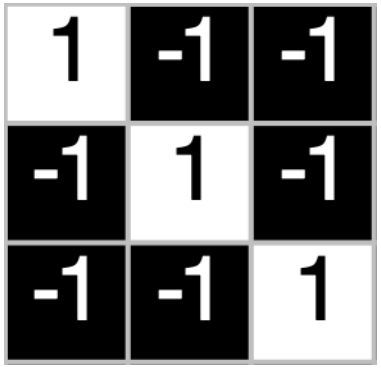


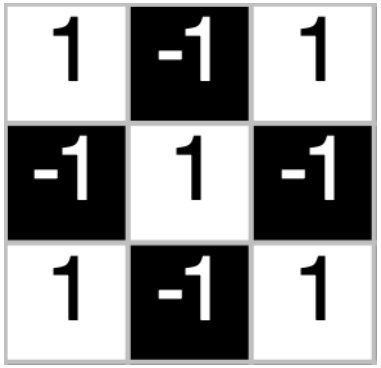
ConvNets match pieces of the image

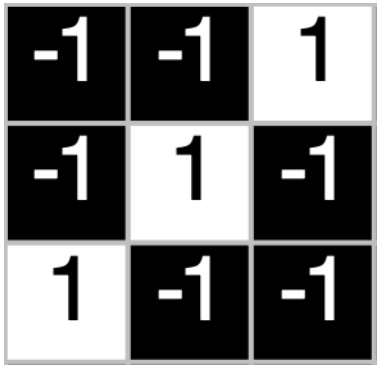


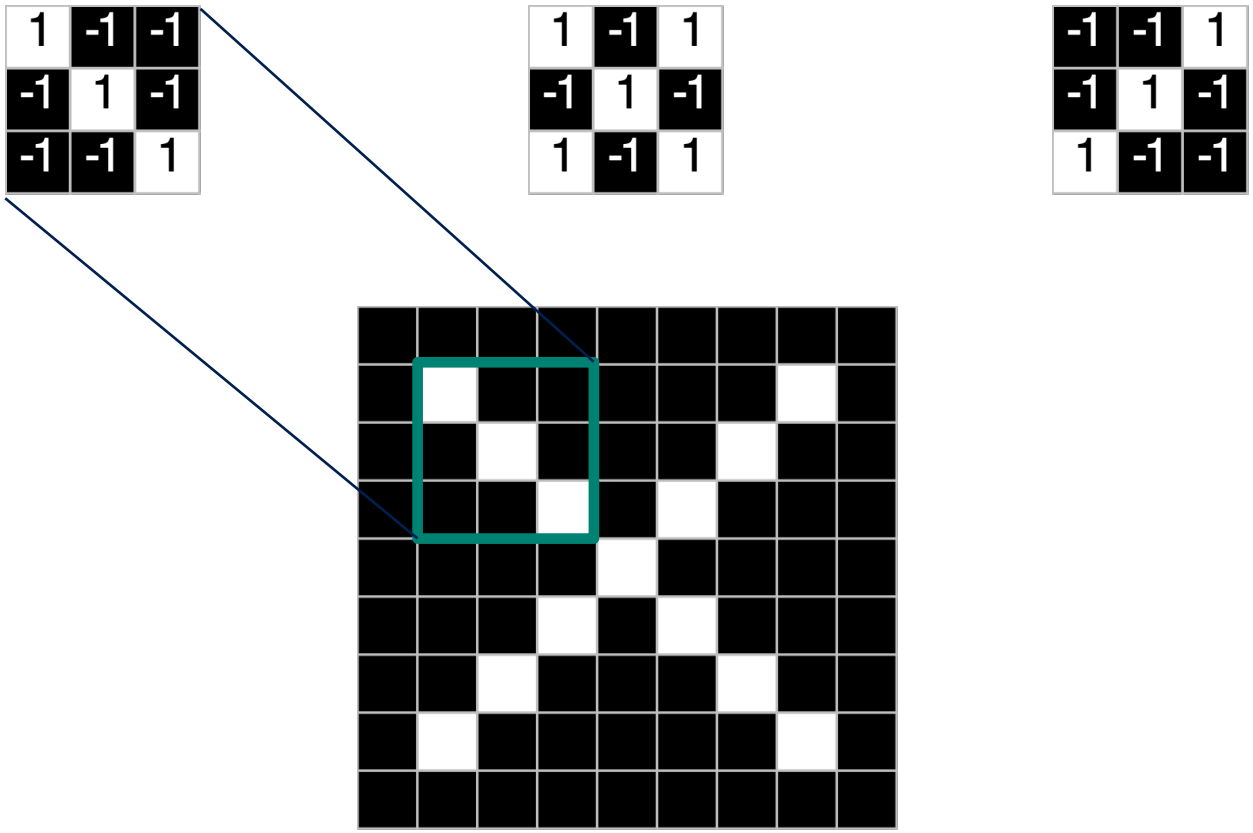
Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

Features match pieces of the image

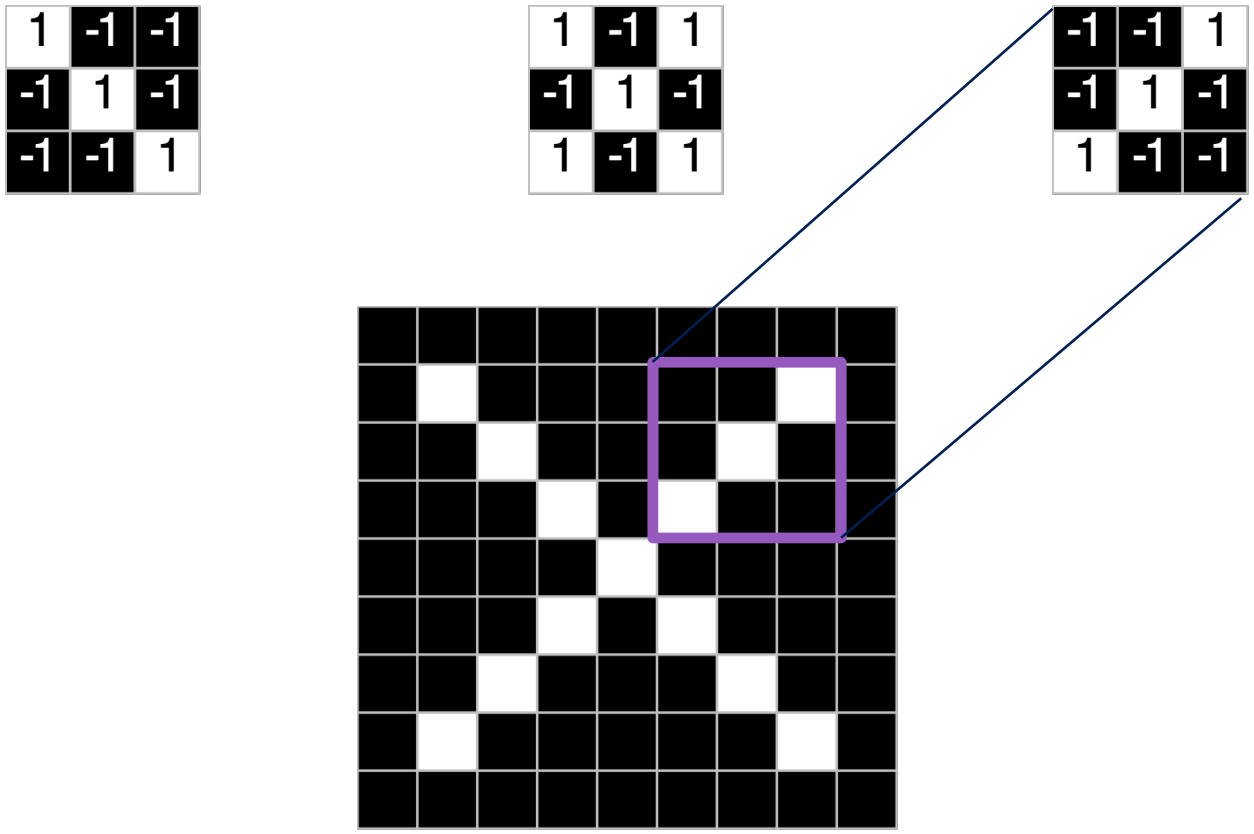




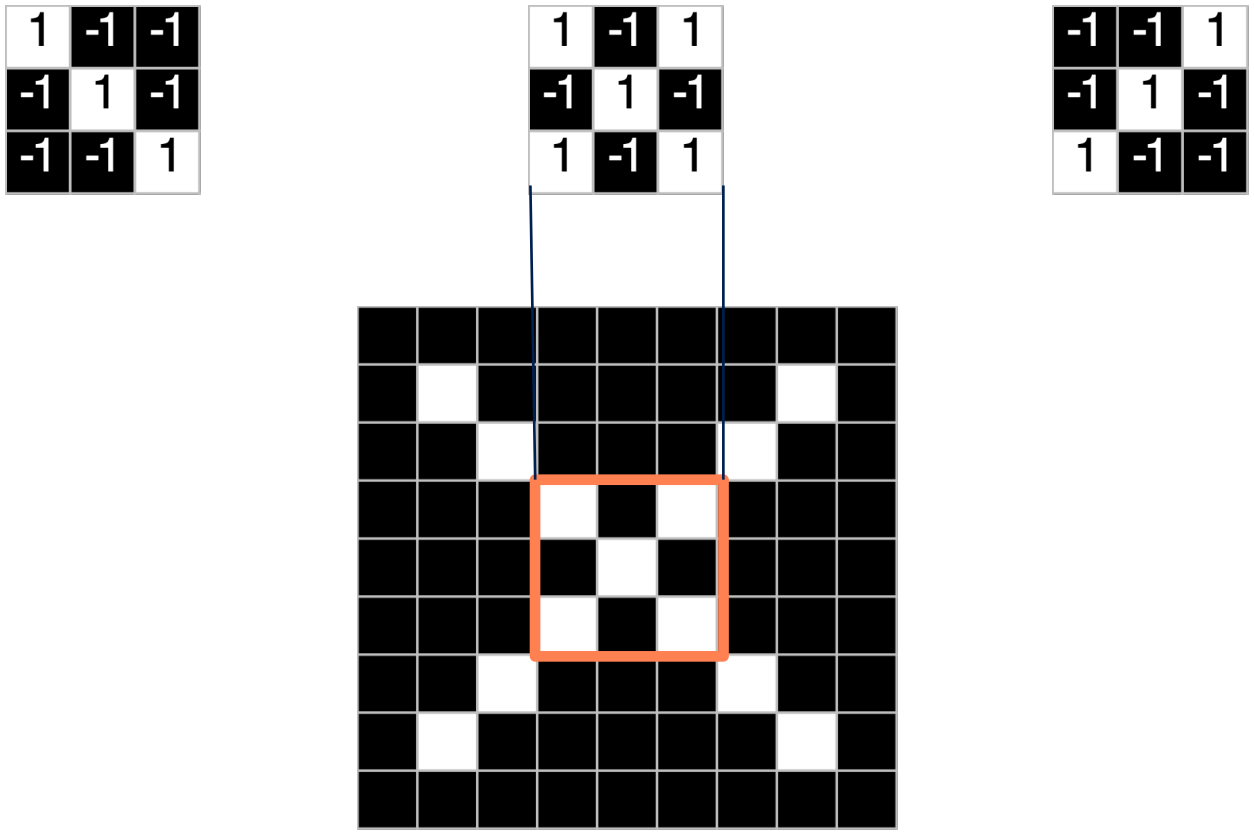




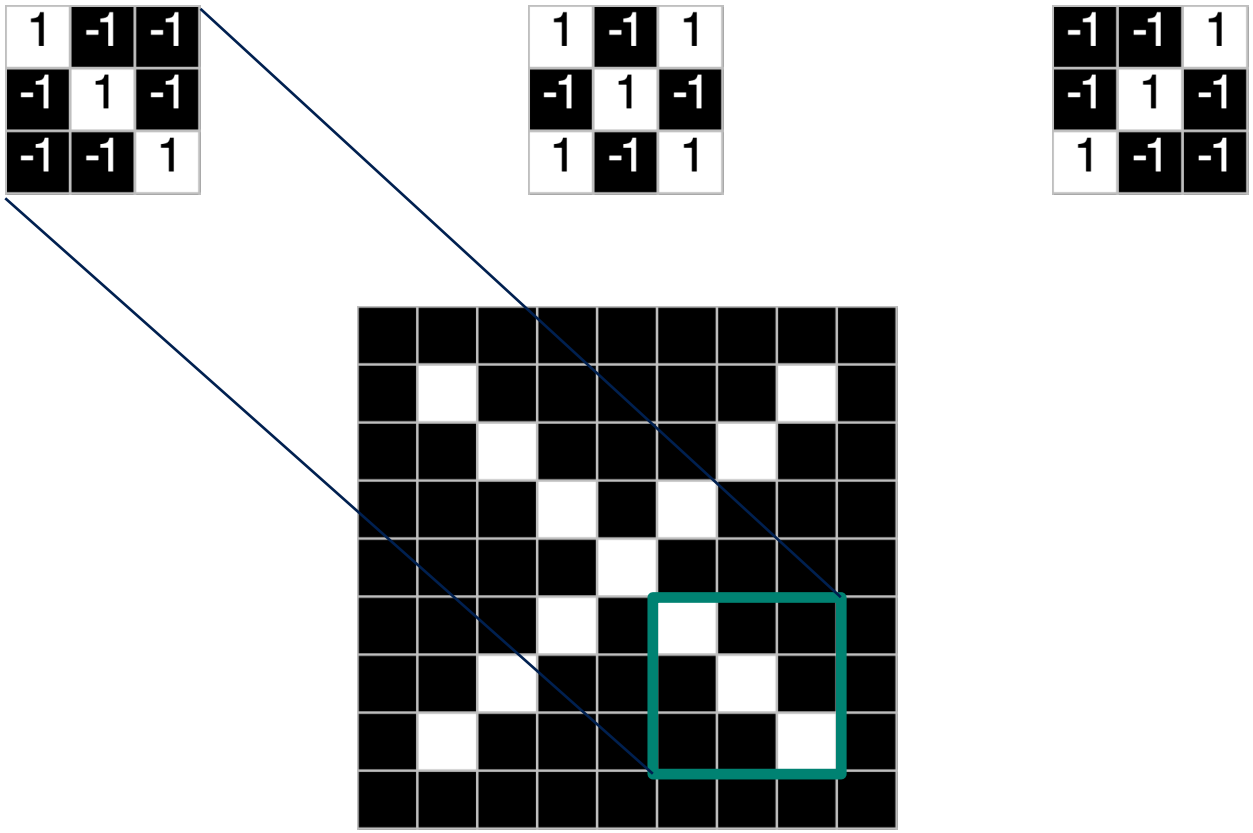
Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html



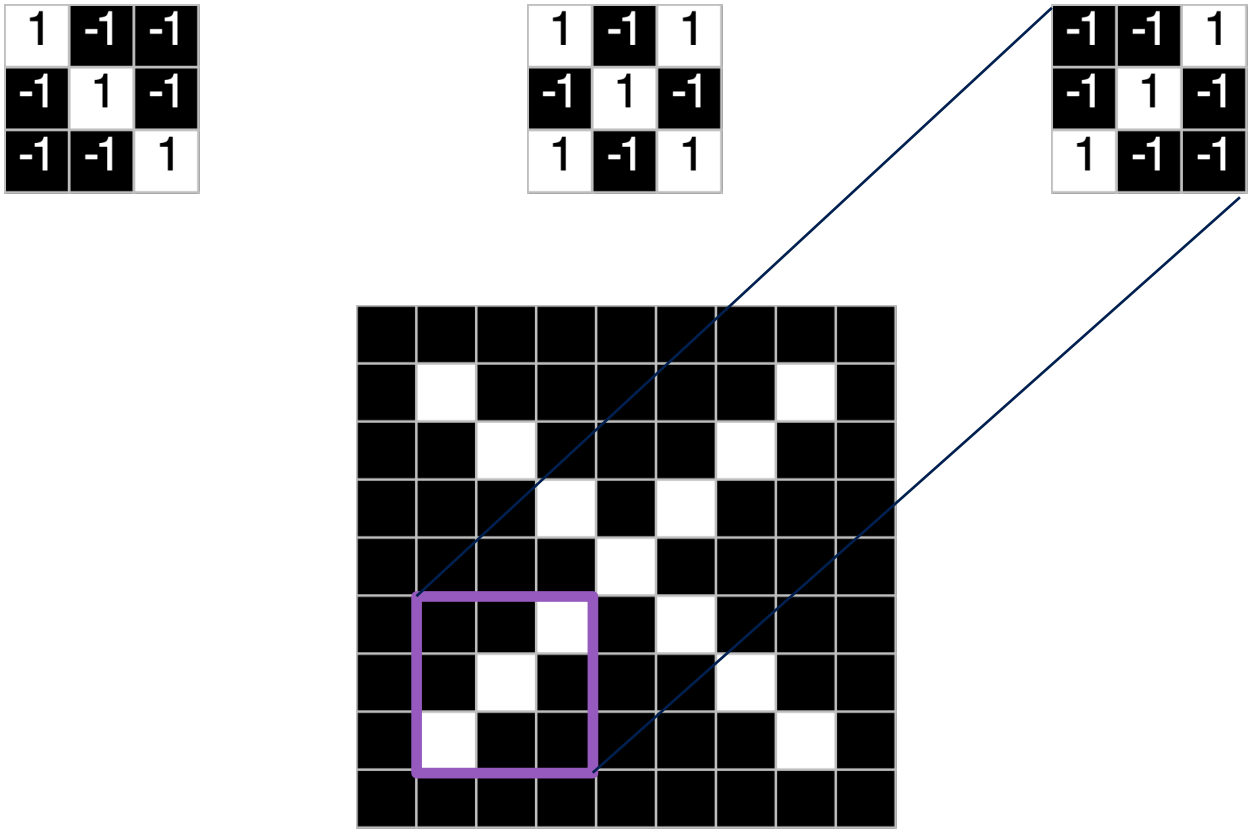
Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html



Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

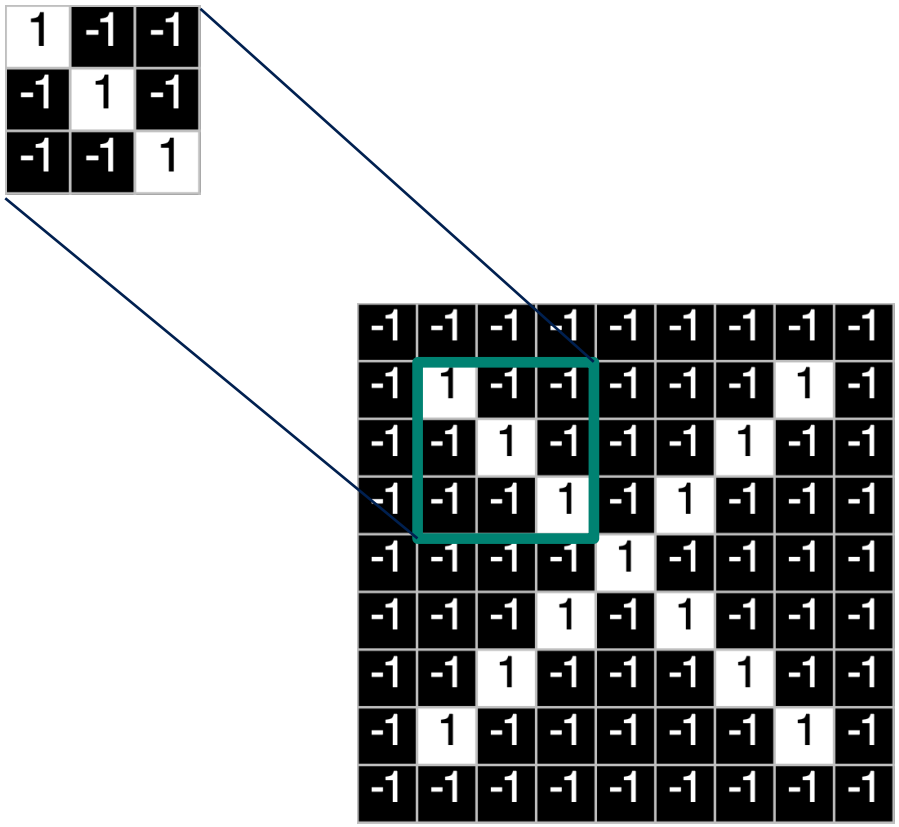


Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html



Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

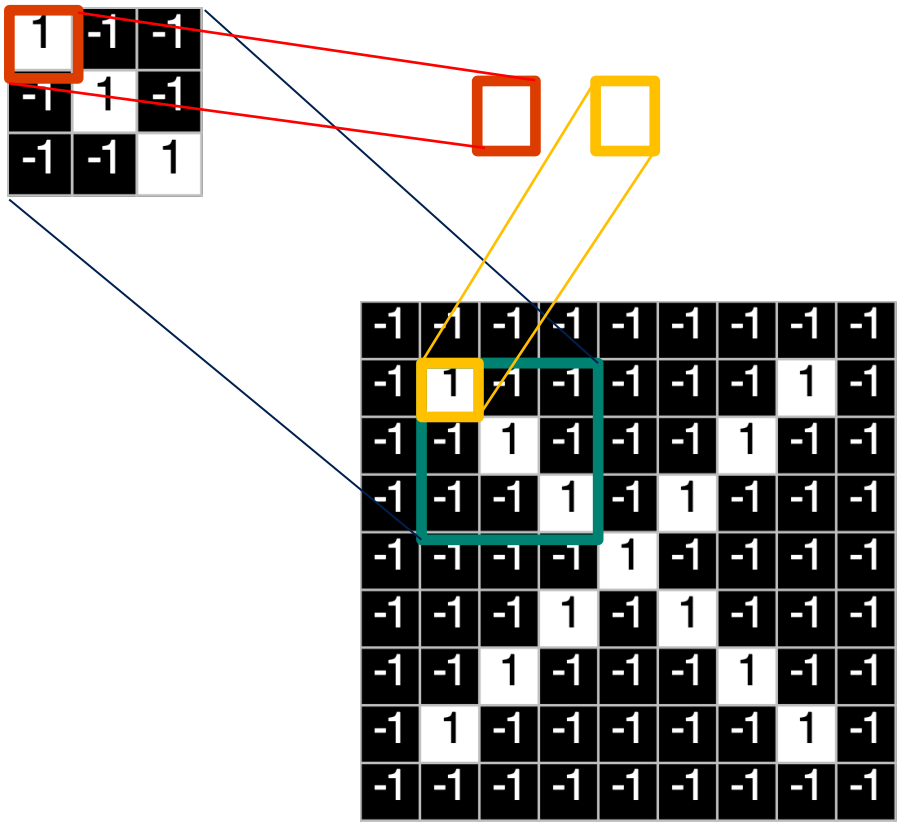
Filtering: The math behind the match



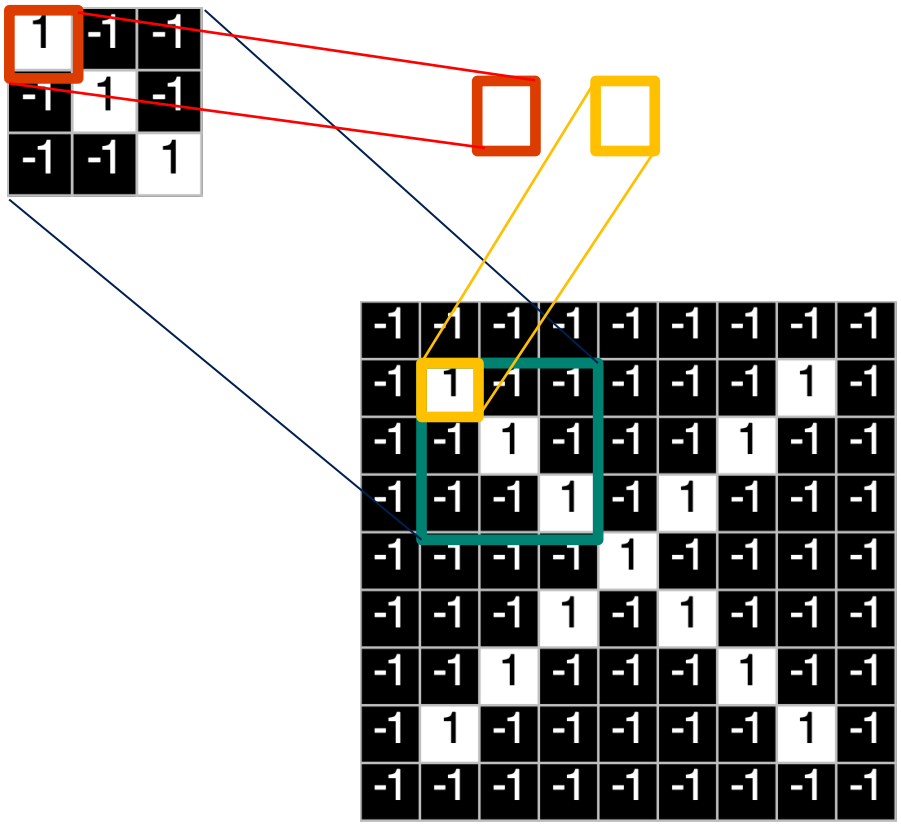
Filtering: The math behind the match

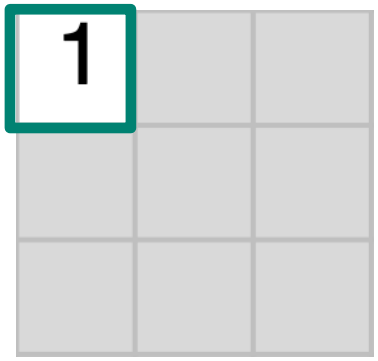
1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

Filtering: The math behind the match

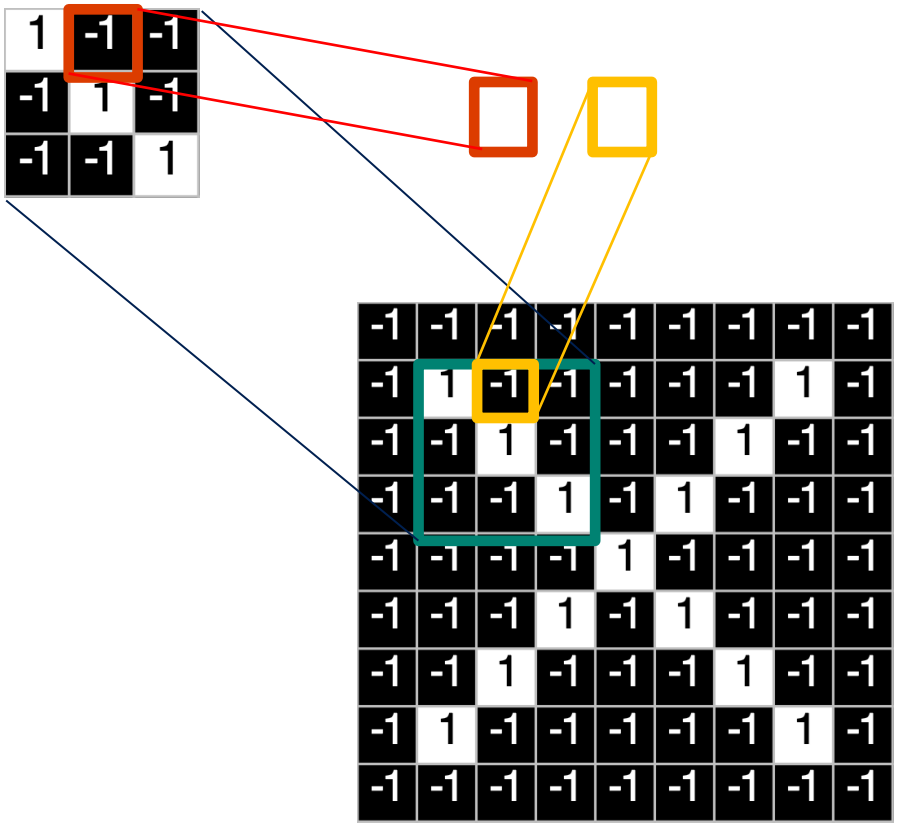


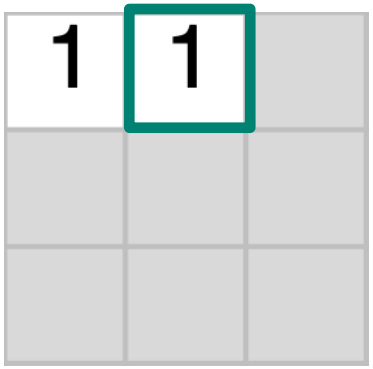
Filtering: The math behind the match



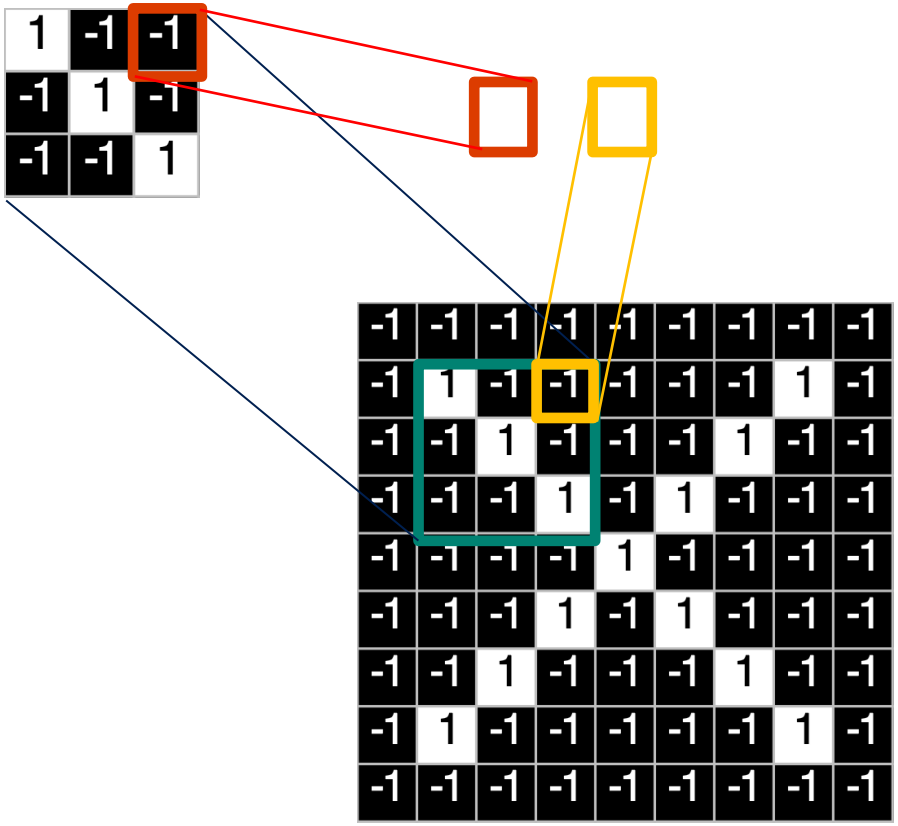


Filtering: The math behind the match

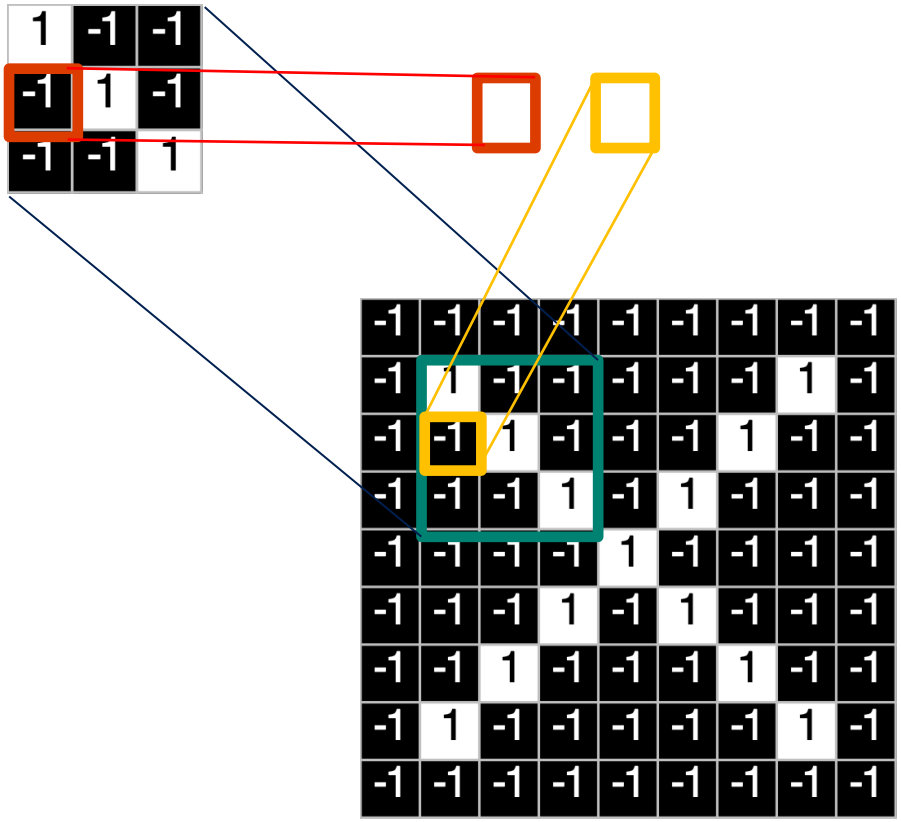


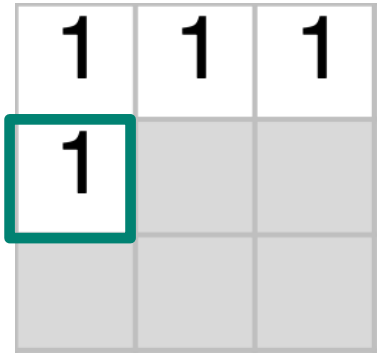


Filtering: The math behind the match

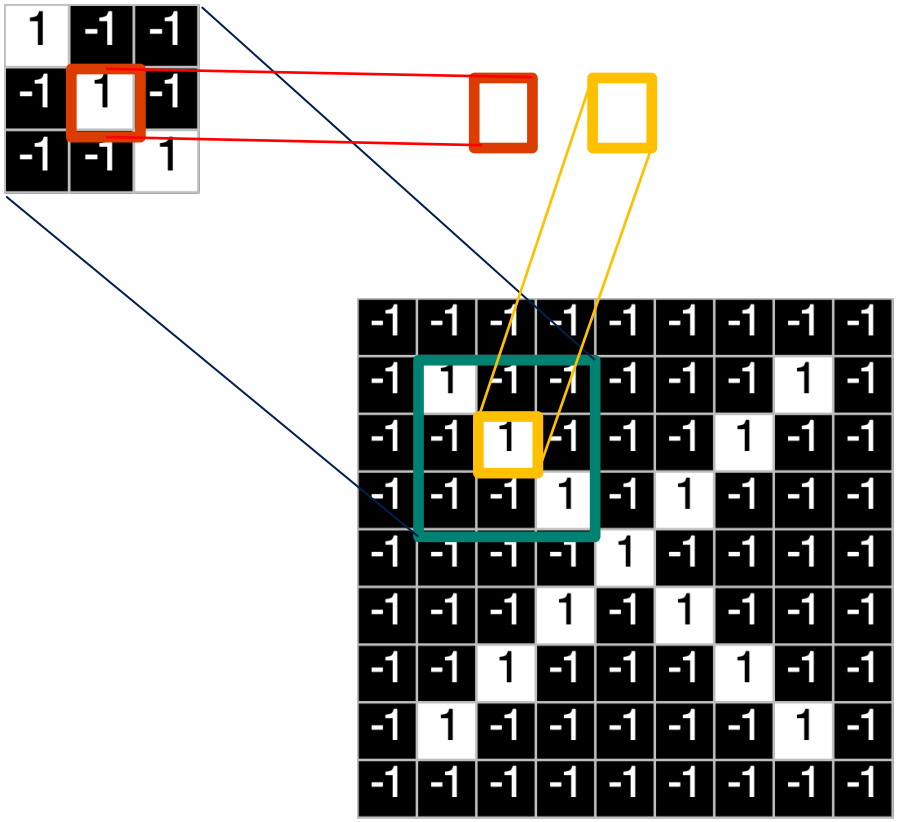


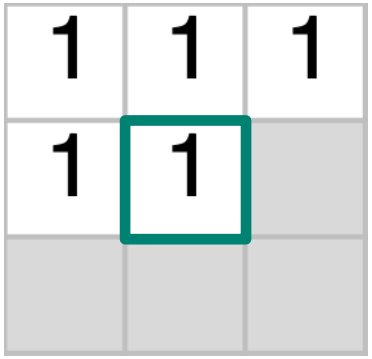
Filtering: The math behind the match



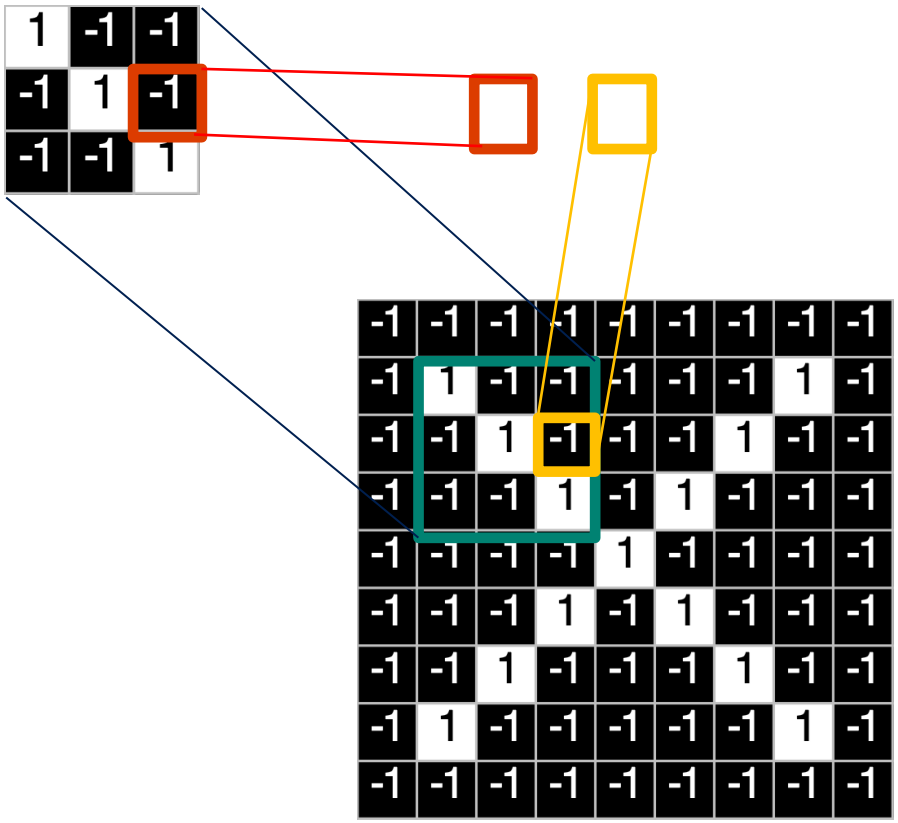


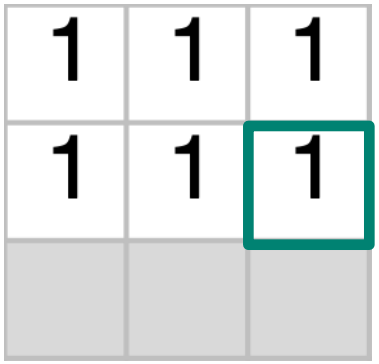
Filtering: The math behind the match



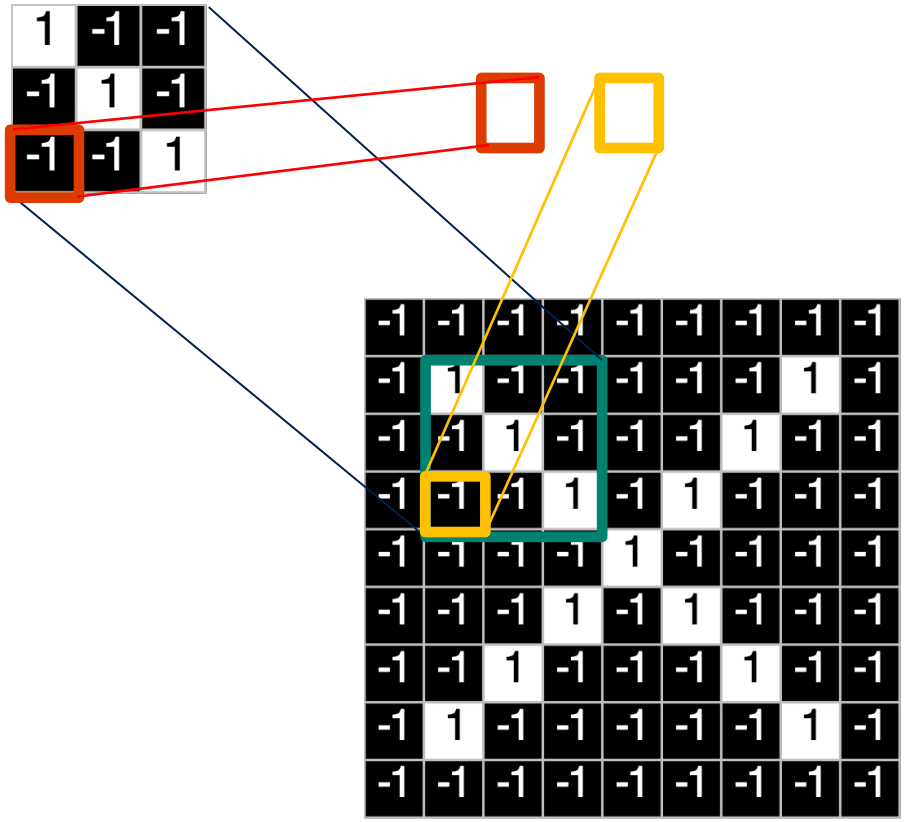


Filtering: The math behind the match

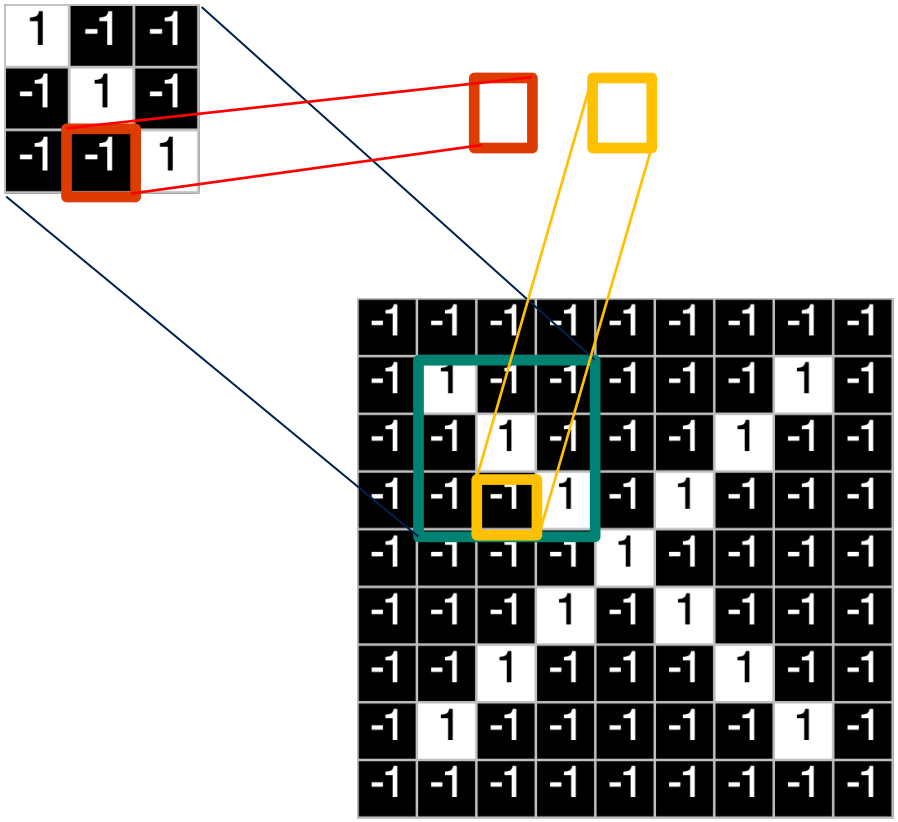


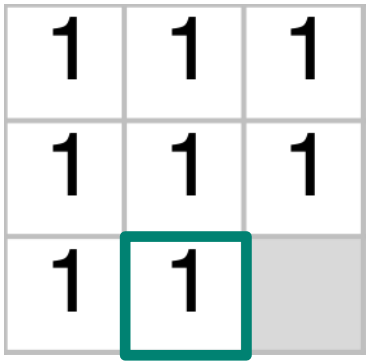


Filtering: The math behind the match

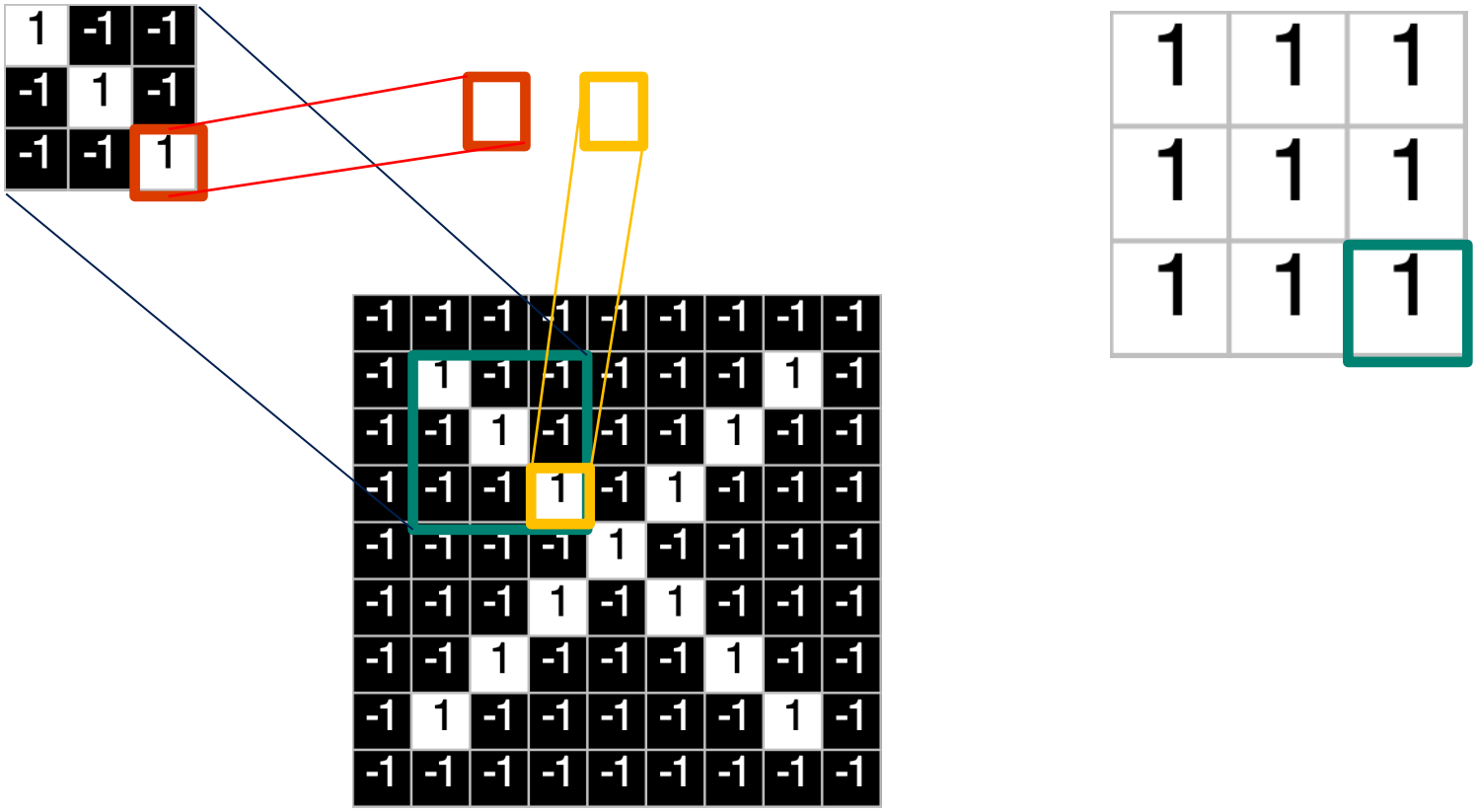


Filtering: The math behind the match

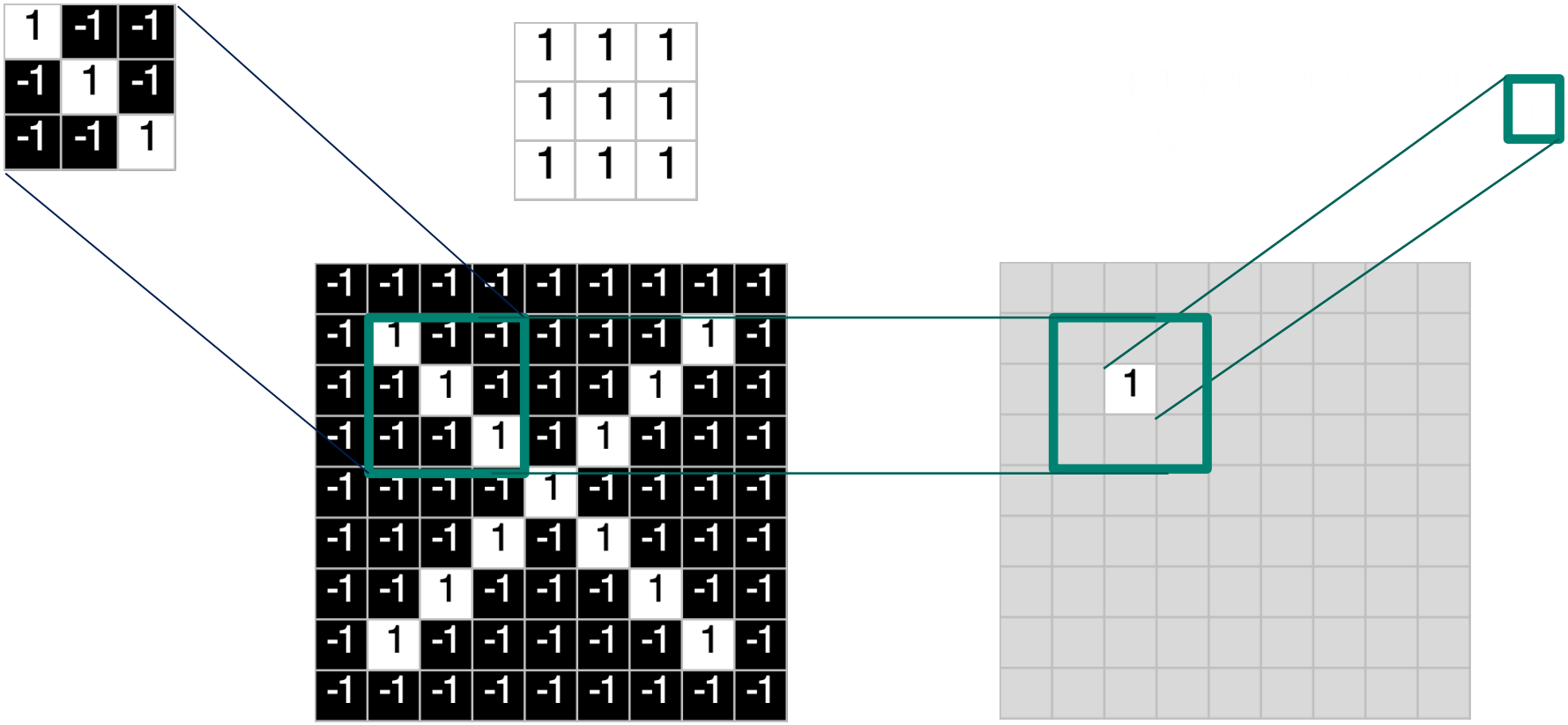




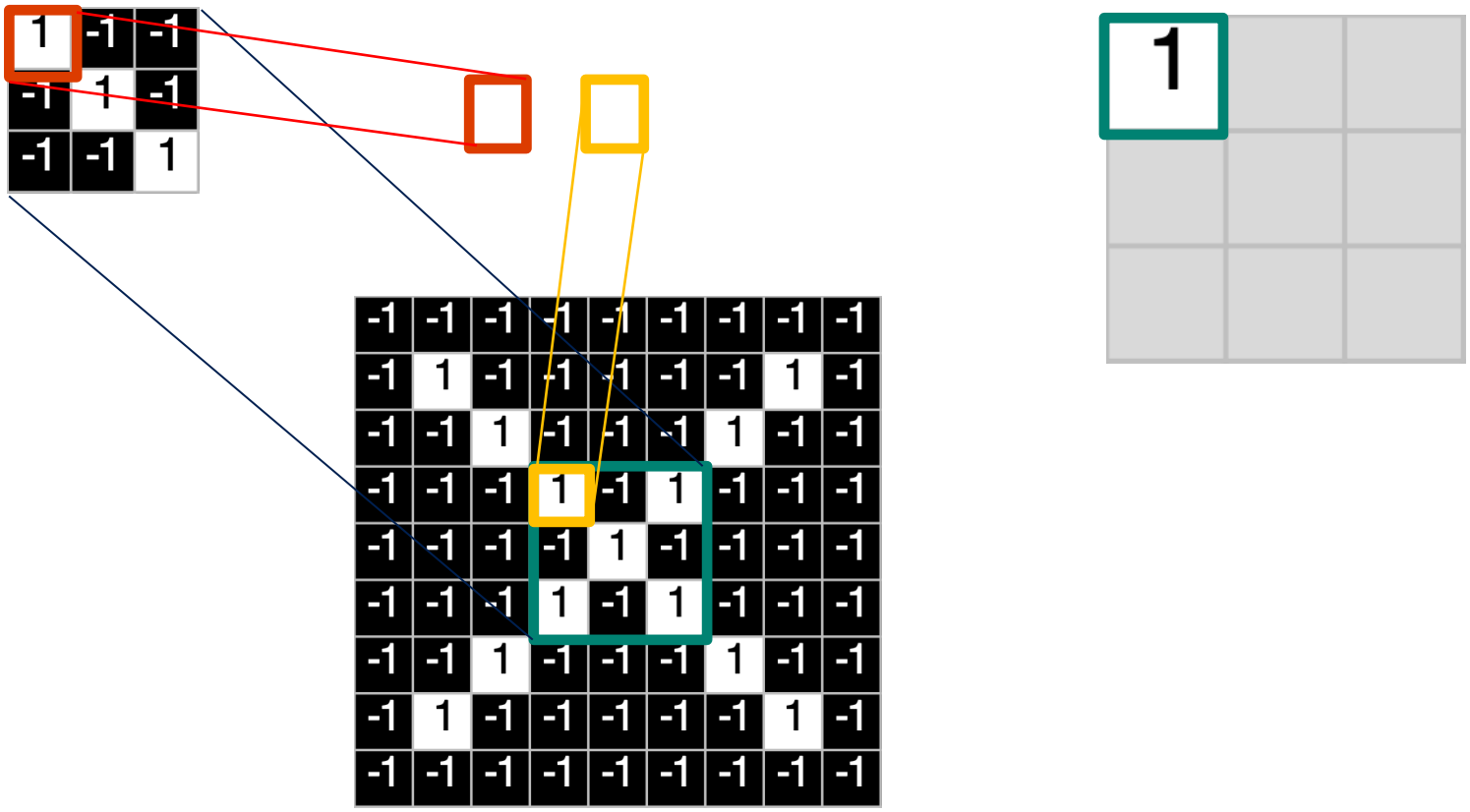
Filtering: The math behind the match



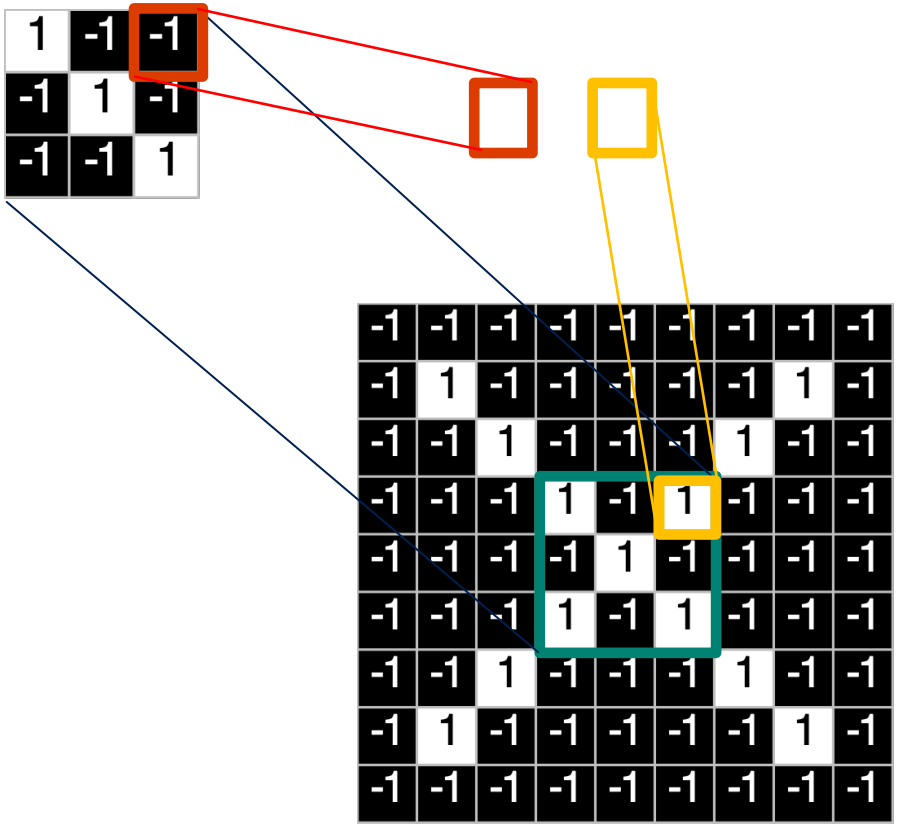
Filtering: The math behind the match



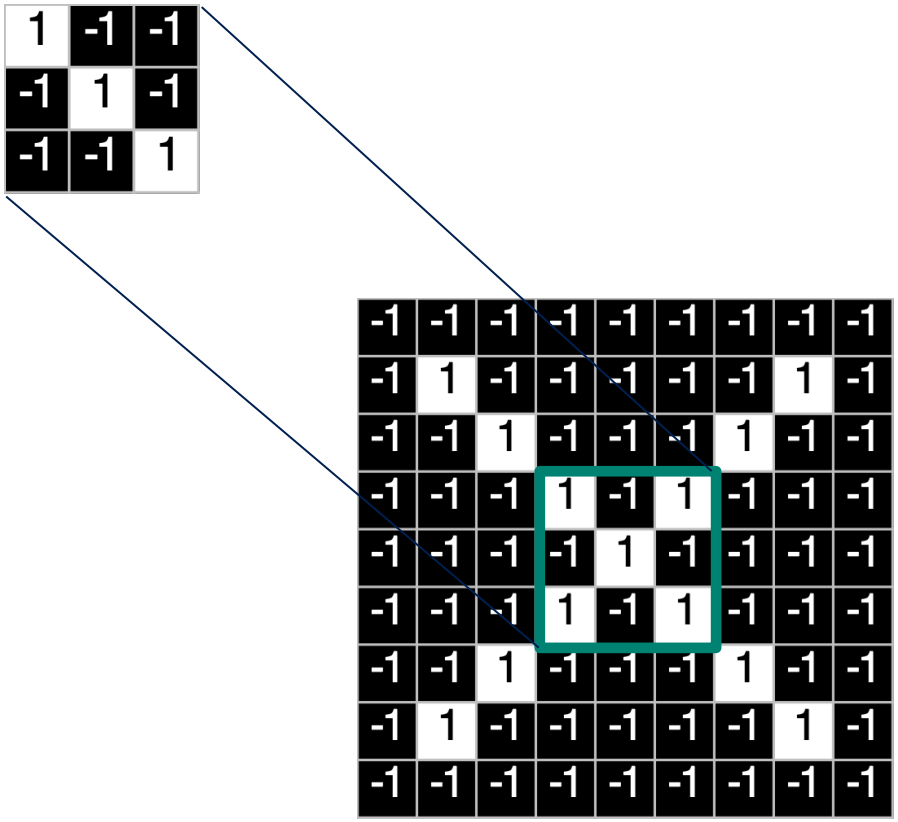
Filtering: The math behind the match

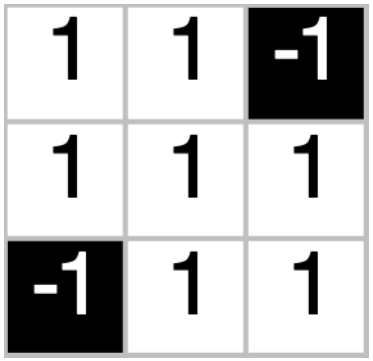


Filtering: The math behind the match

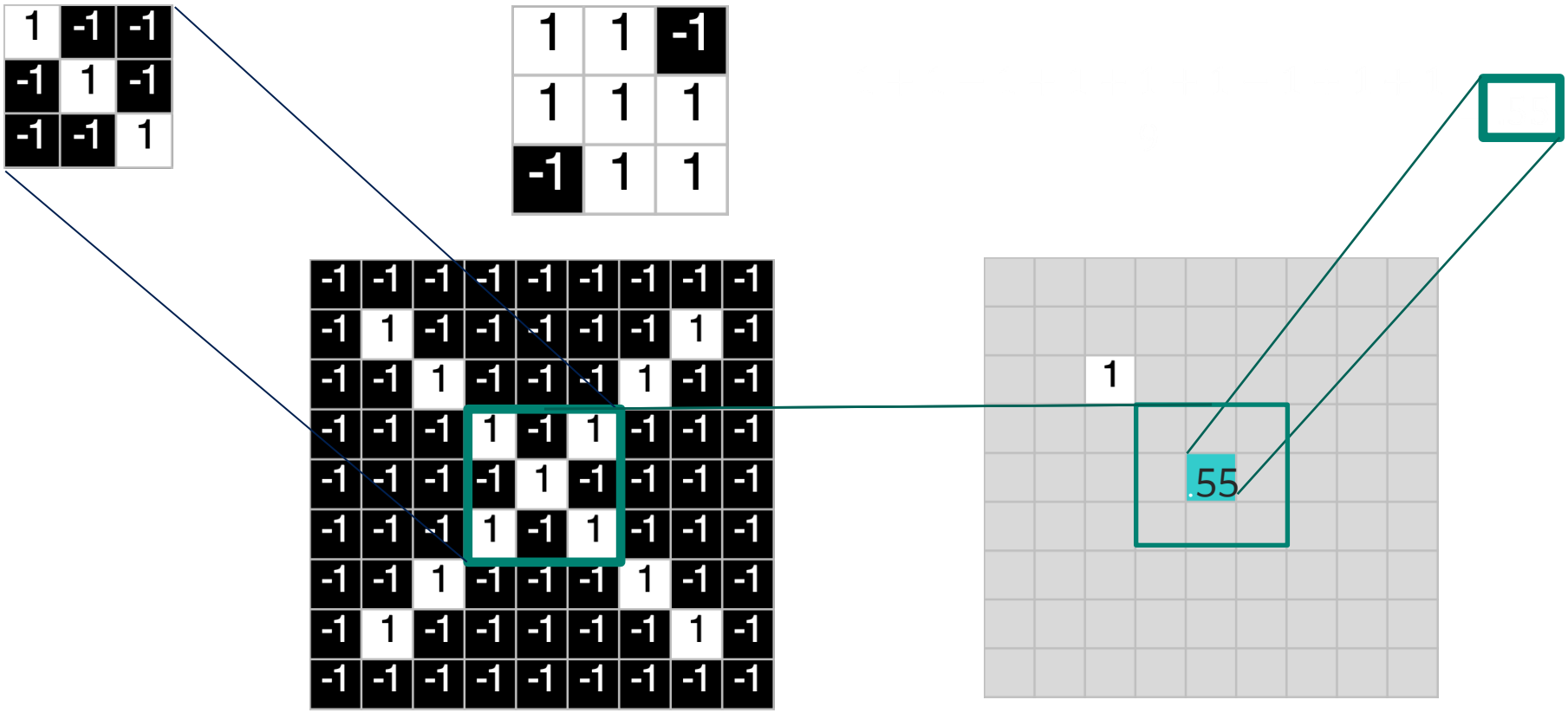


Filtering: The math behind the match



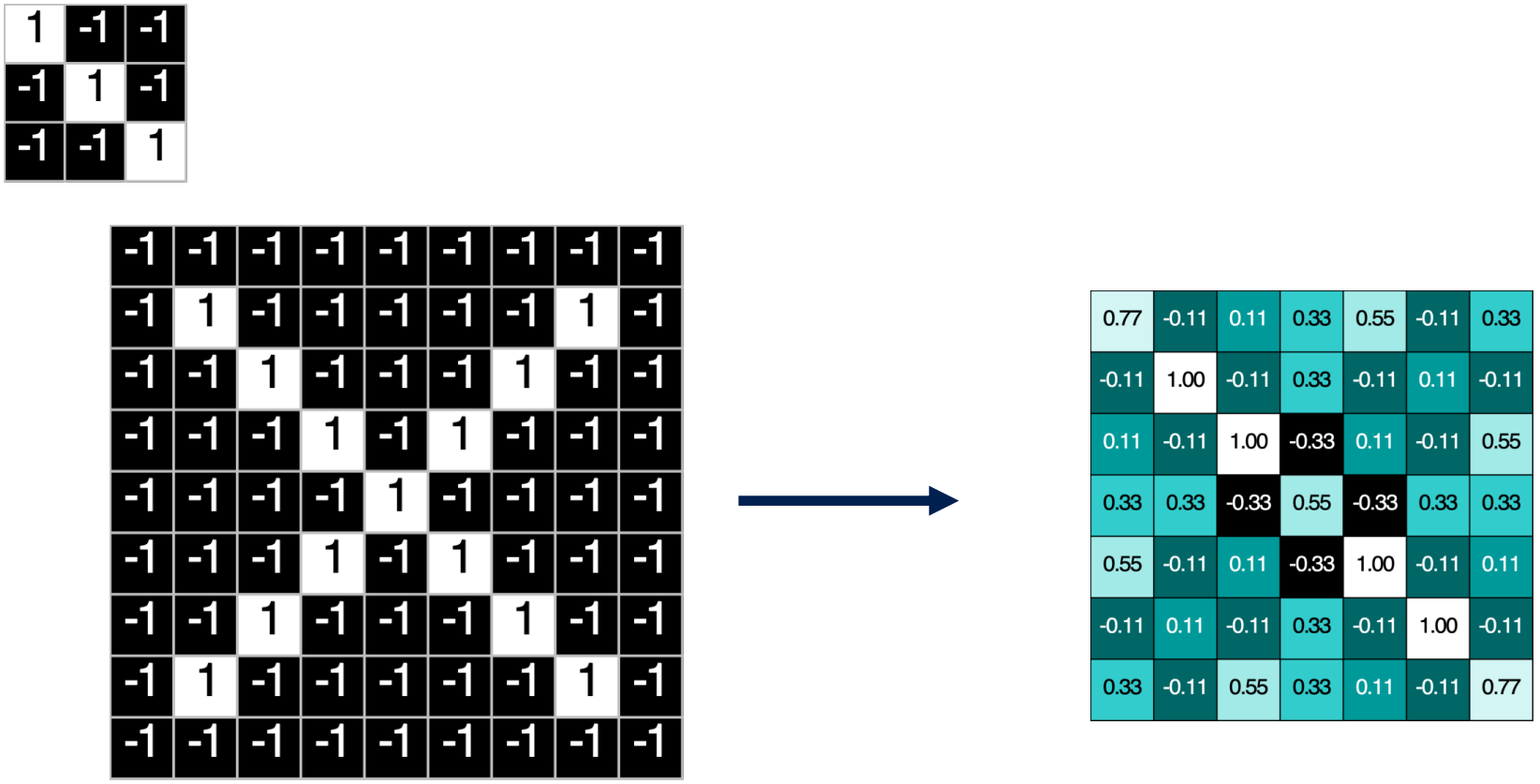


Filtering: The math behind the match

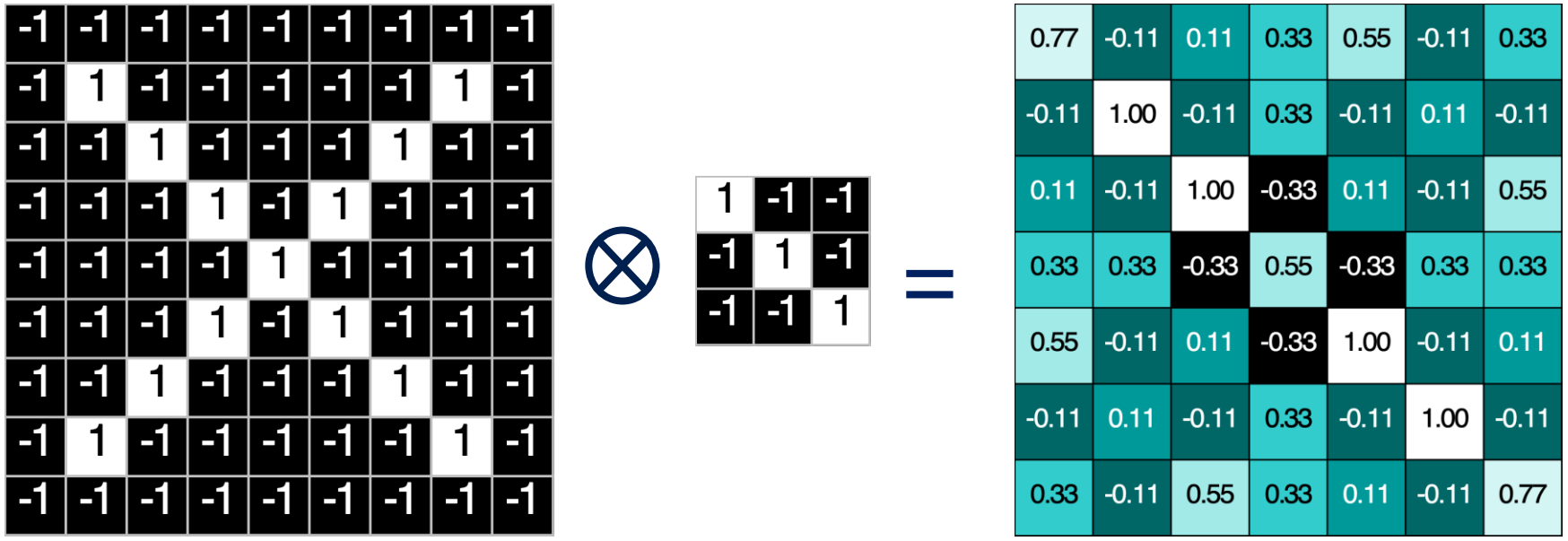


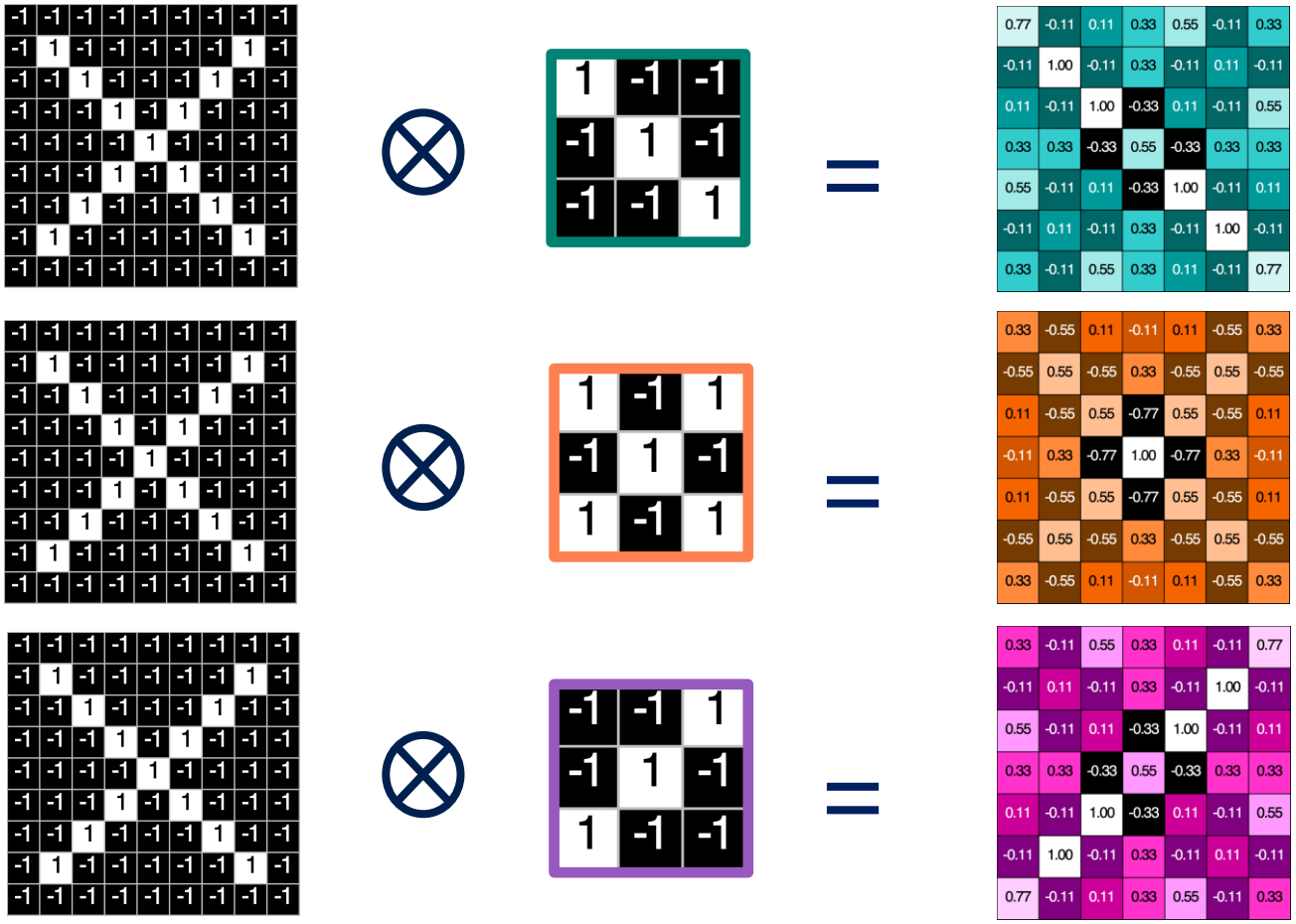
Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

Convolution: Trying every possible match



Convolution: Trying every possible match

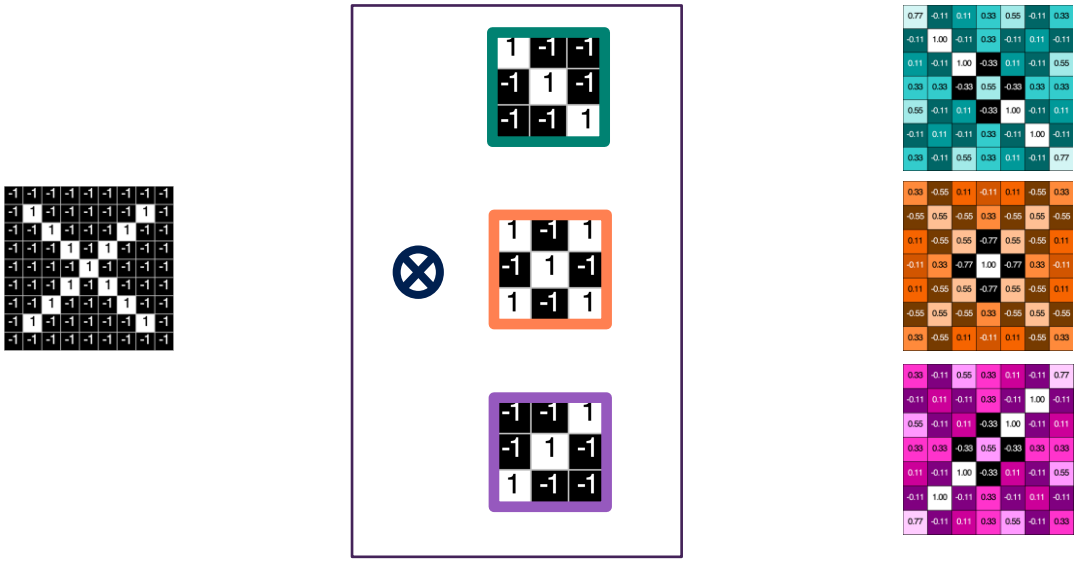




Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

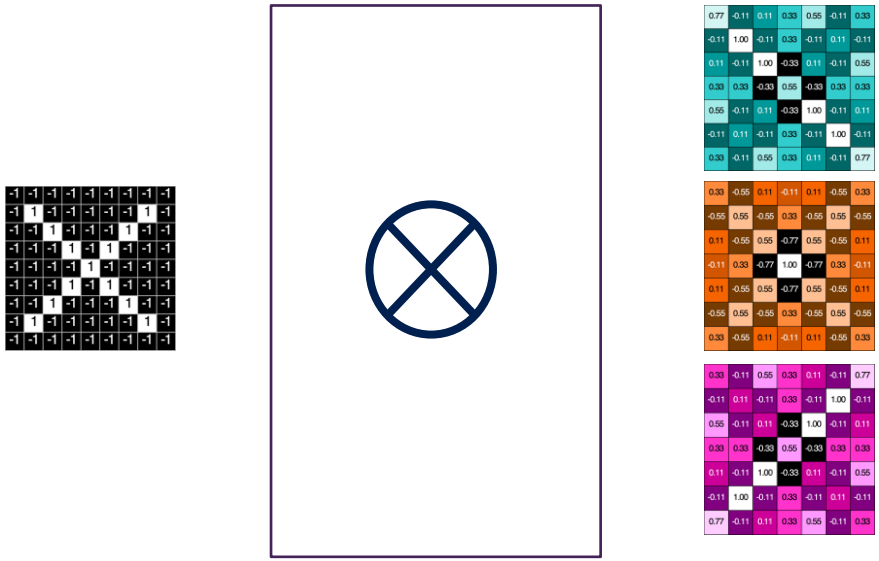
Convolution layer

One image becomes a stack of filtered images



Convolution layer

One image becomes a stack of filtered images

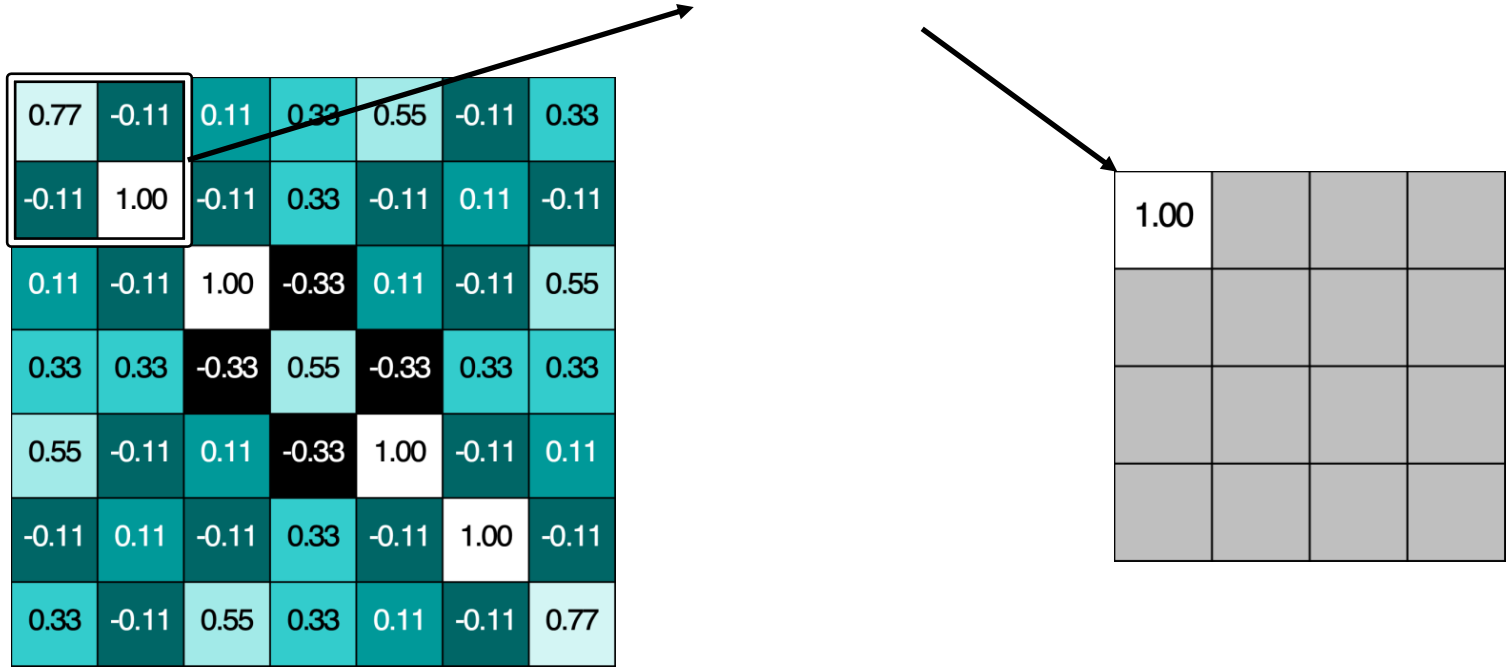


Pooling: Shrinking the image stack

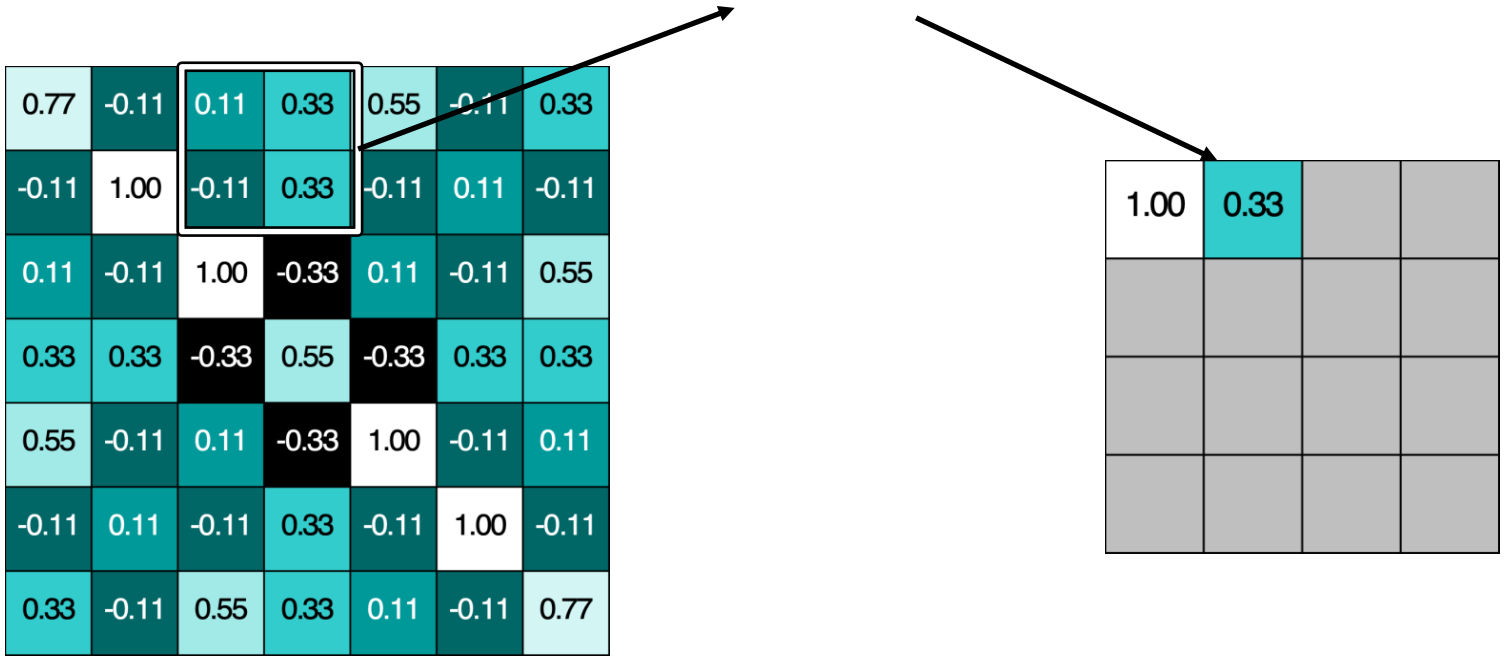
1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

Pooling

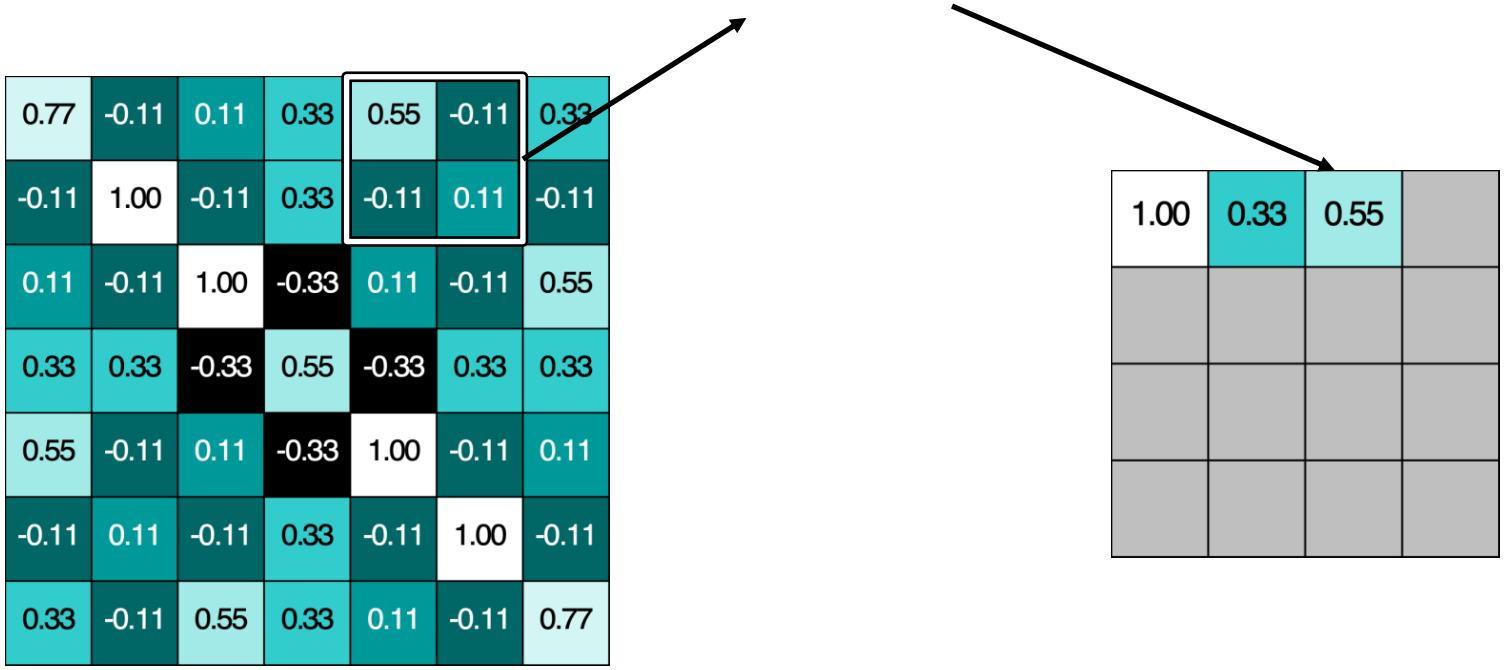
- Max Pooling



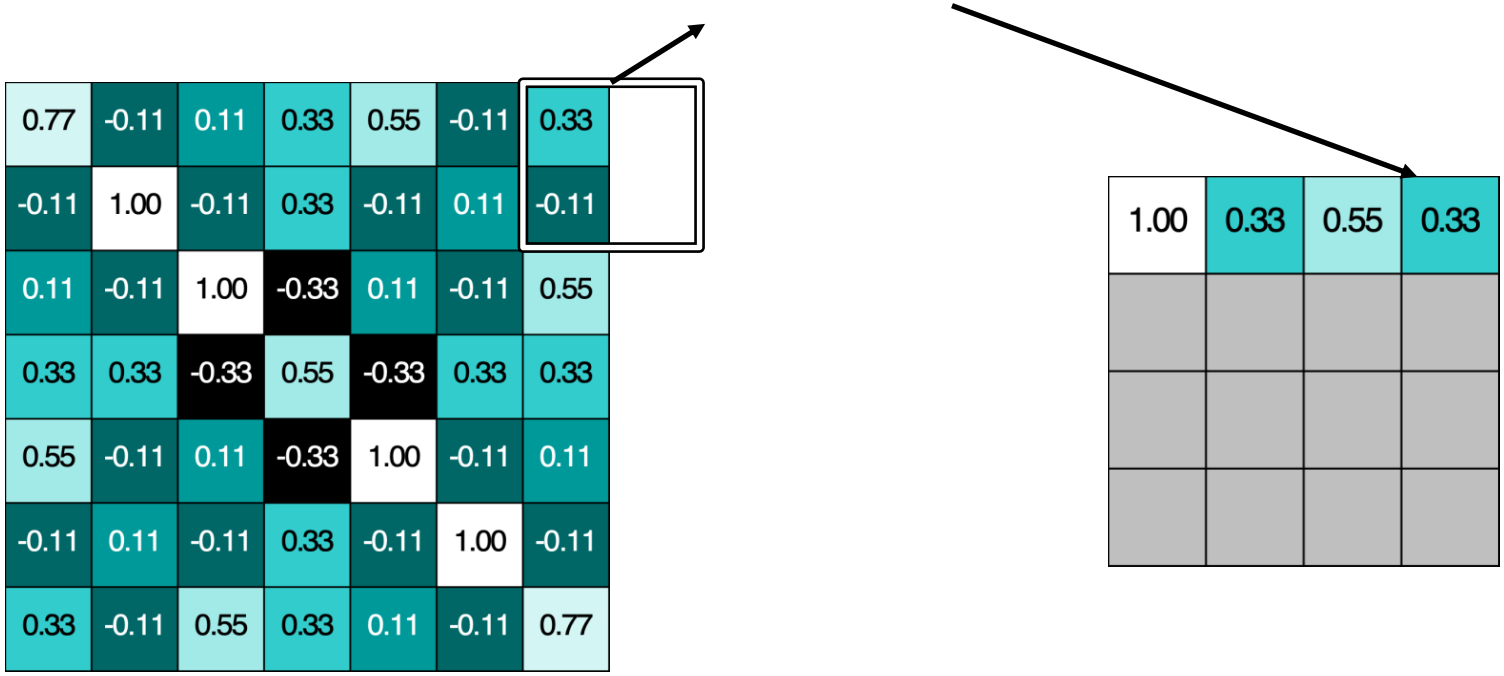
Pooling


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
1.00 0.33

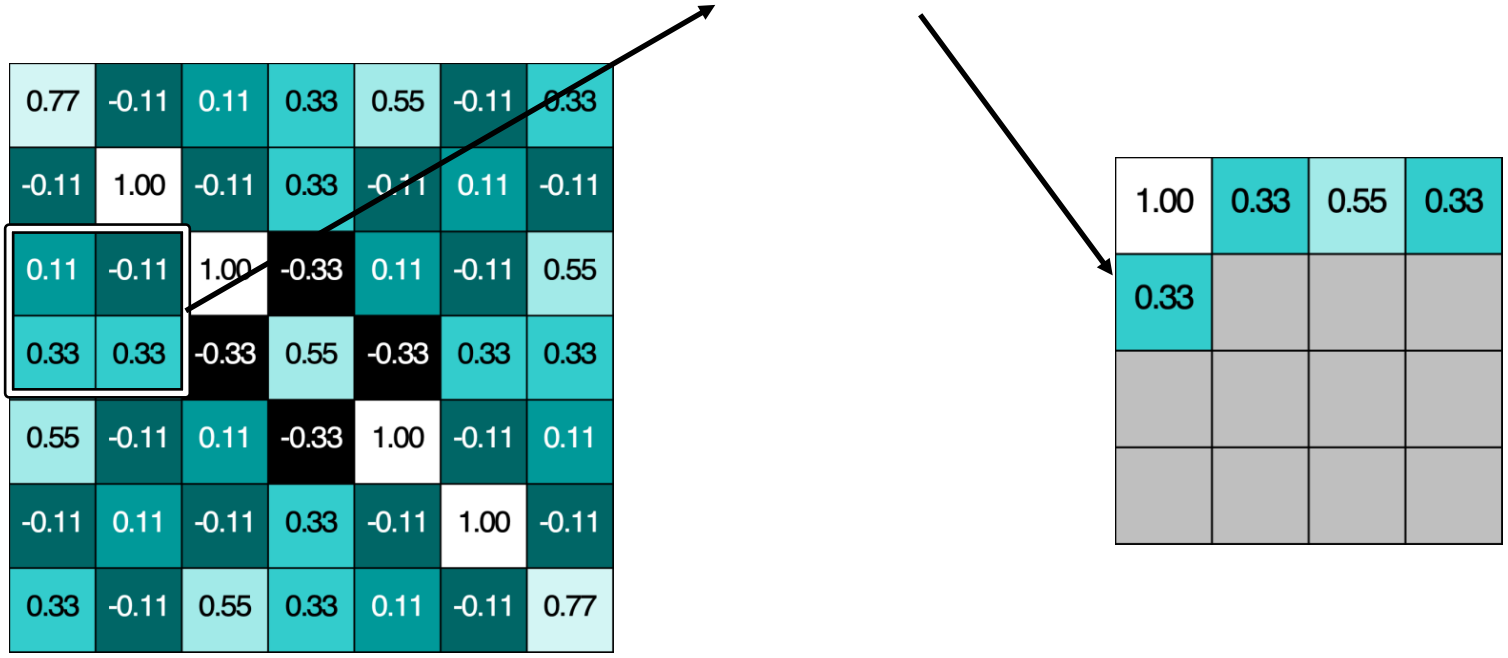
Pooling


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
1.00 0.33 0.55

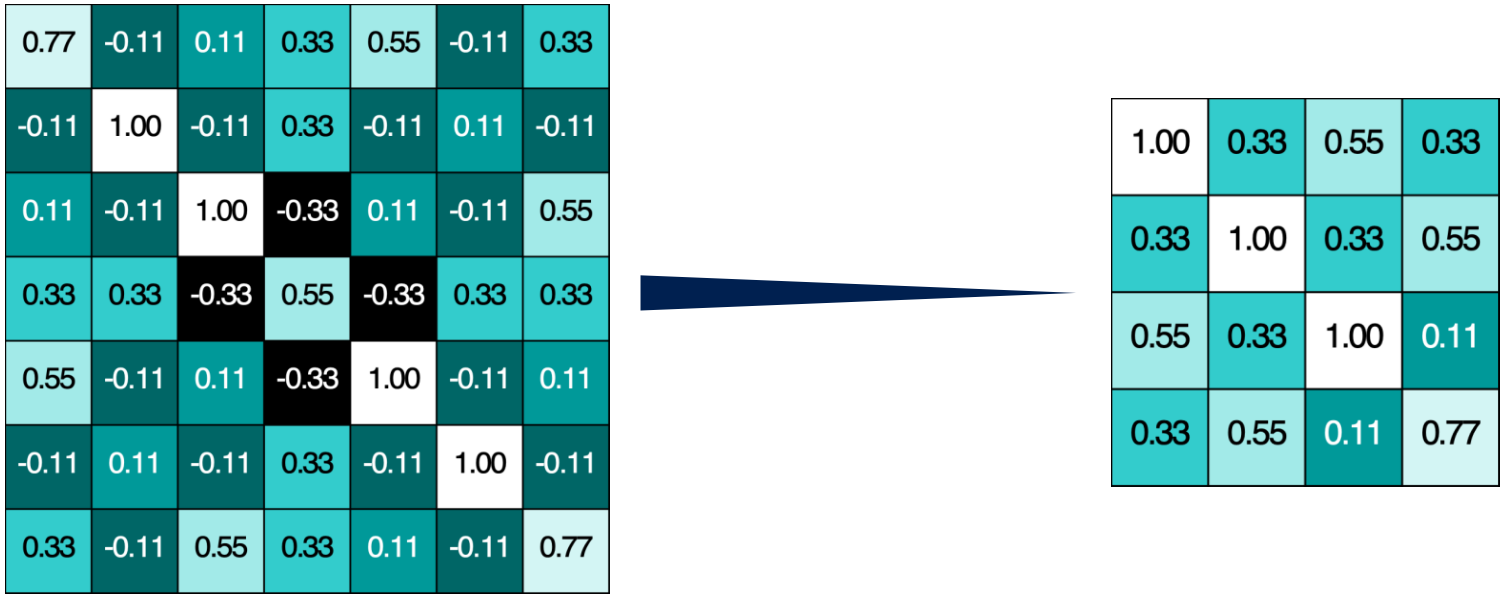
Pooling


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
1.00 0.33 0.55 0.33

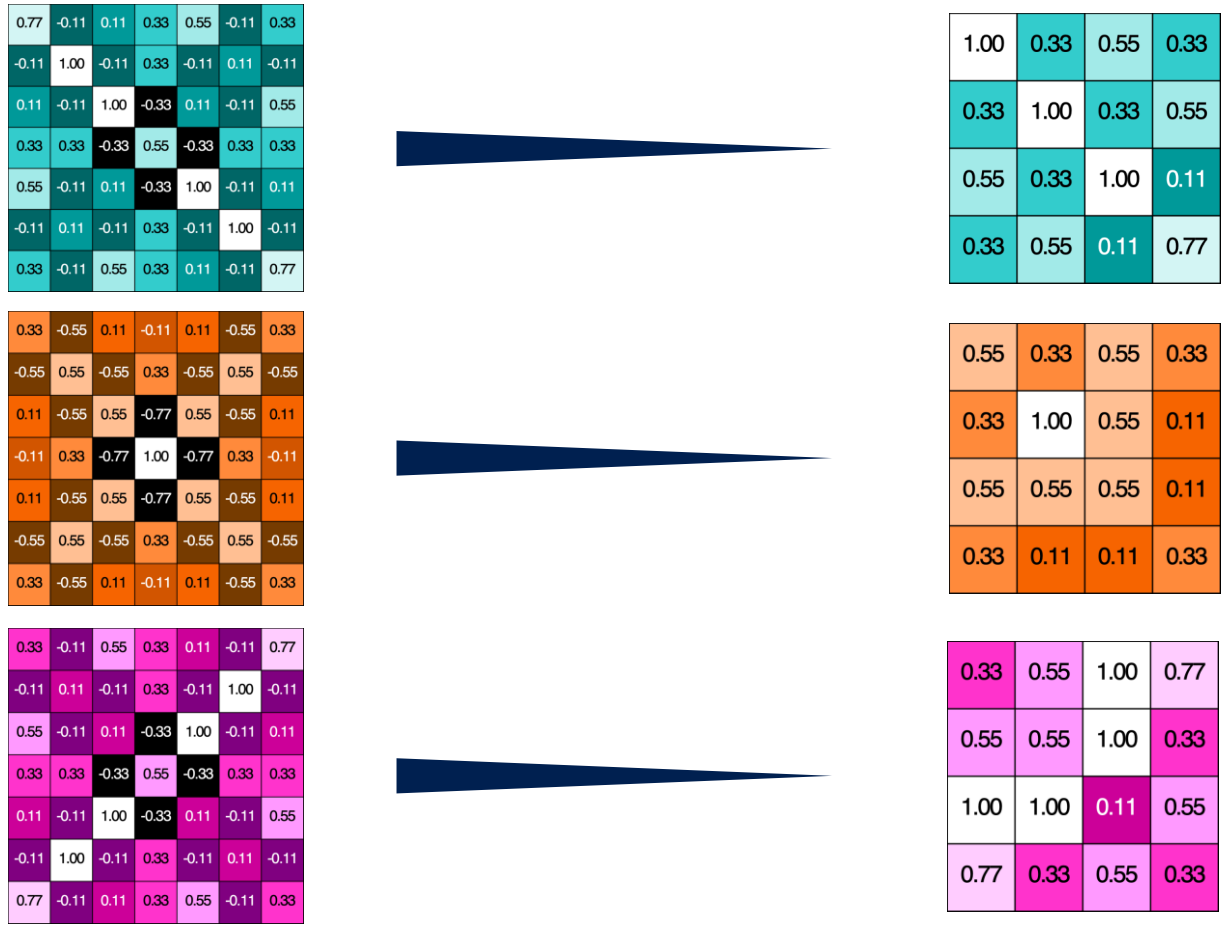
Pooling


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
1.00 0.33 0.55 0.33
0.33

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
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

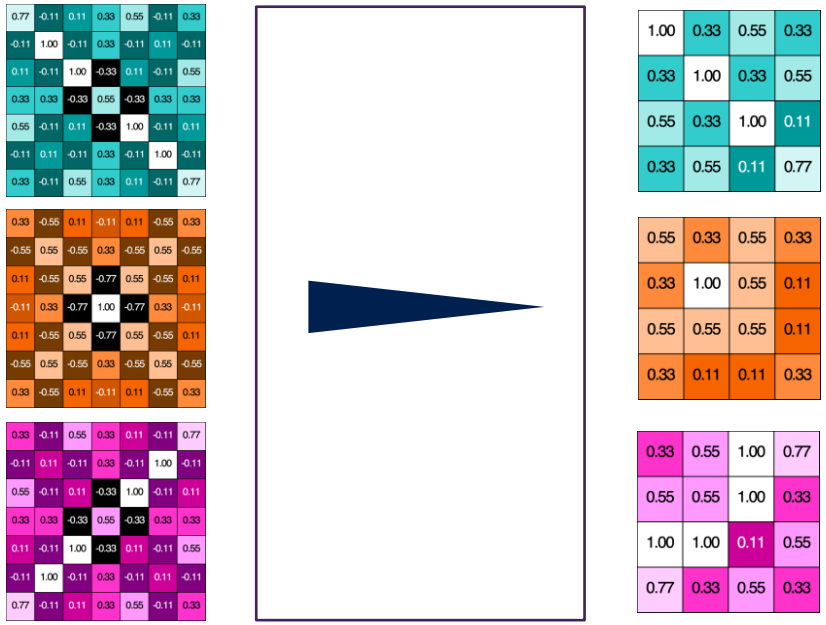
Pooling



Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

Pooling layer

A stack of images becomes a stack of smaller images.

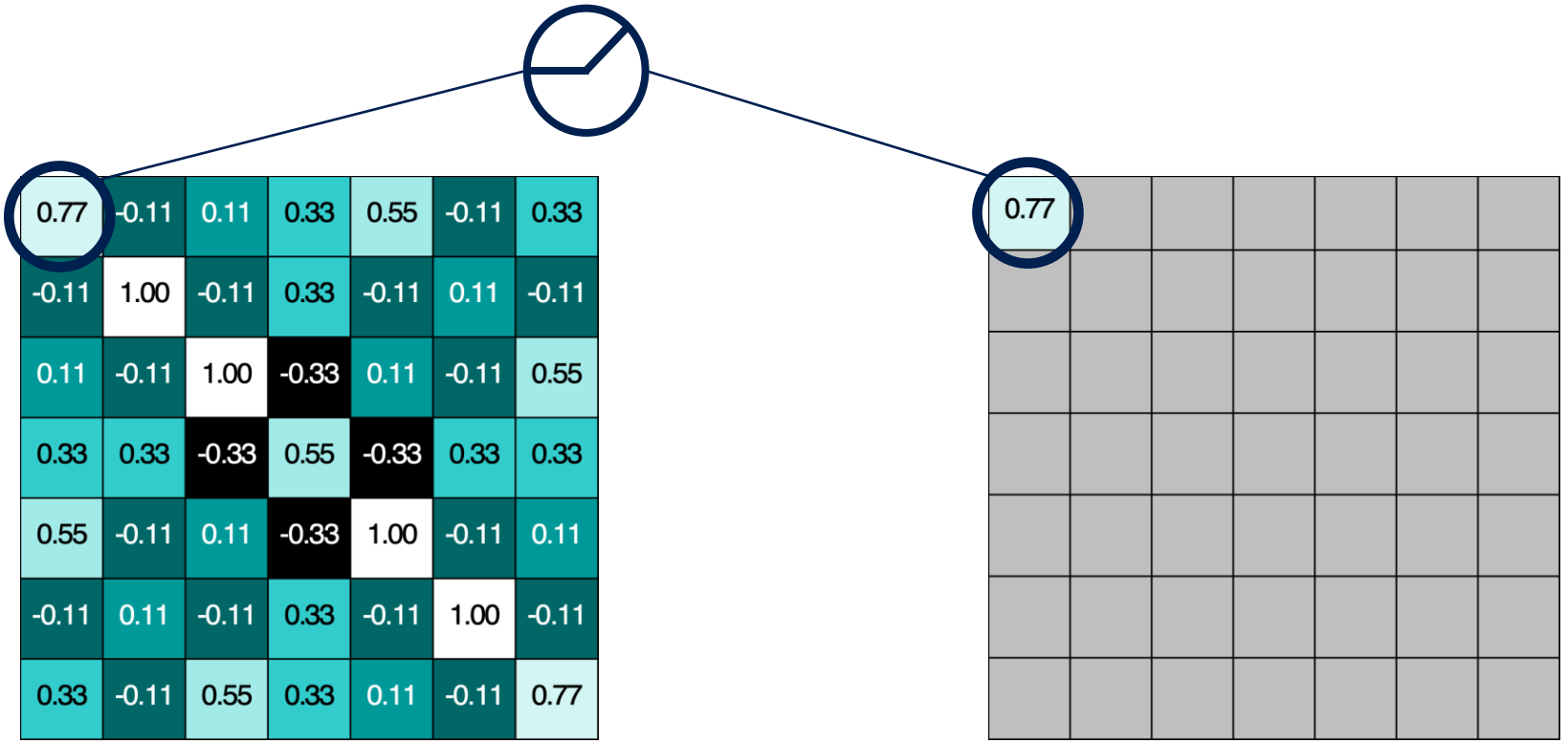


Normalization

Keep the math from breaking by tweaking each of the values just a bit.

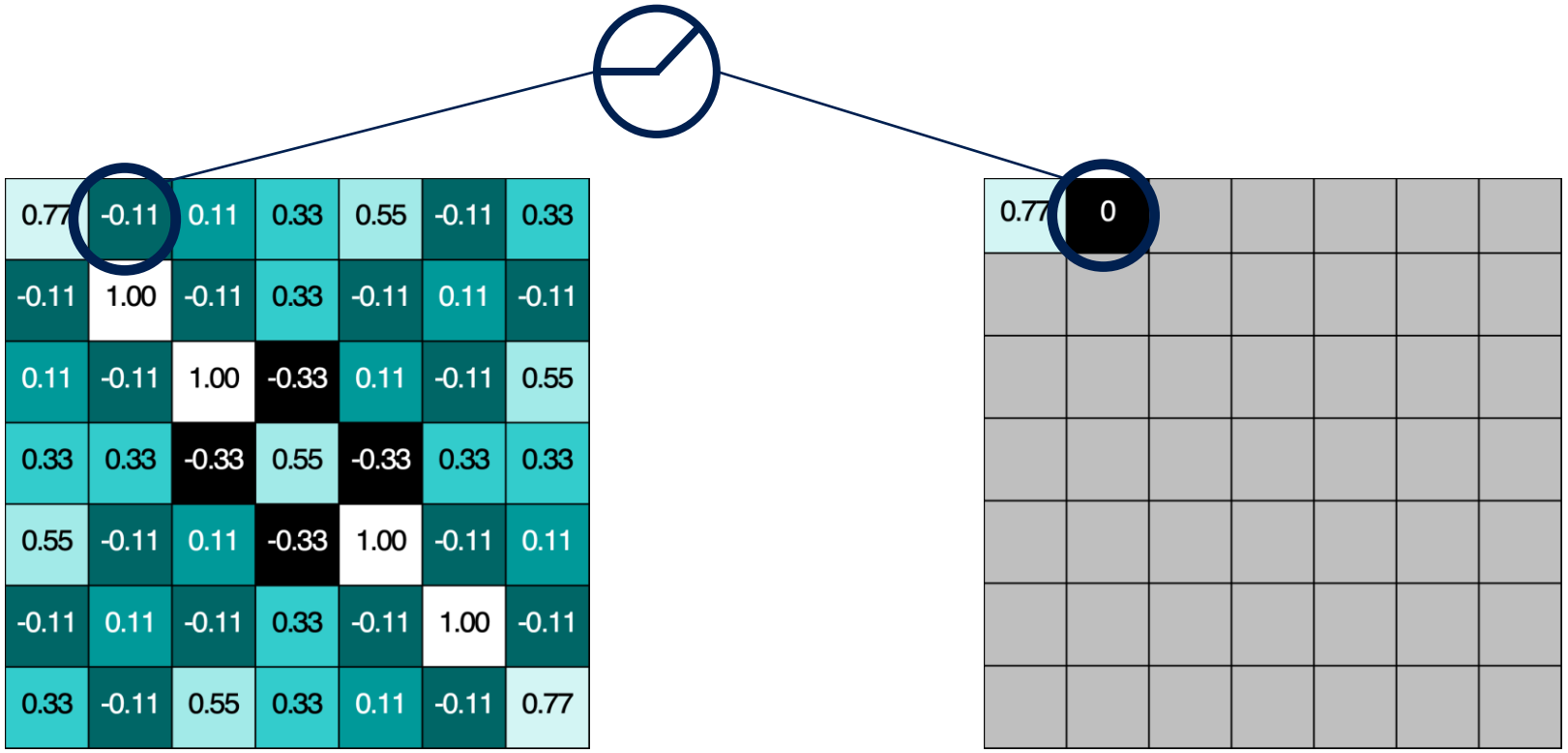
Change everything negative to zero.

Rectified Linear Units (ReLUs)

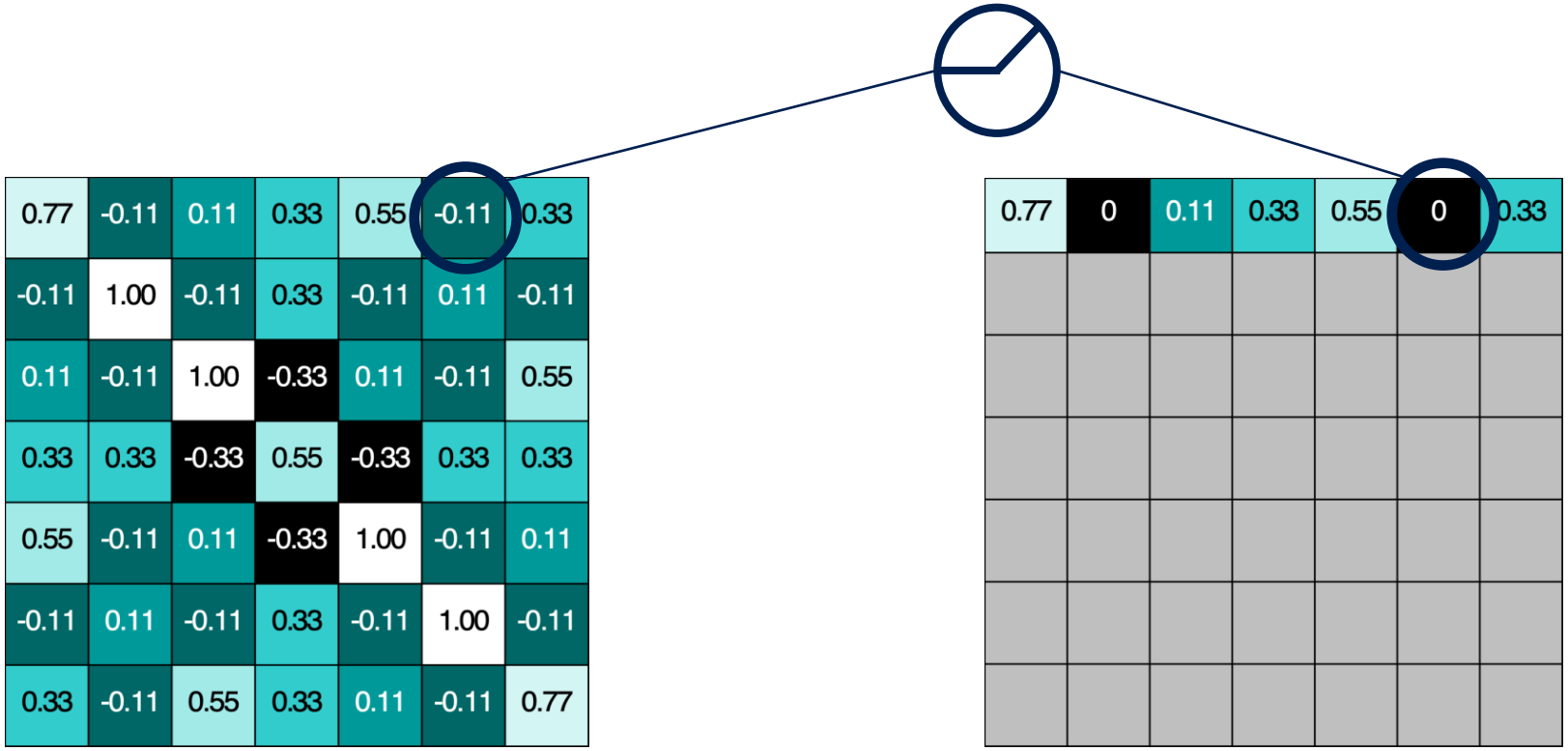


Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

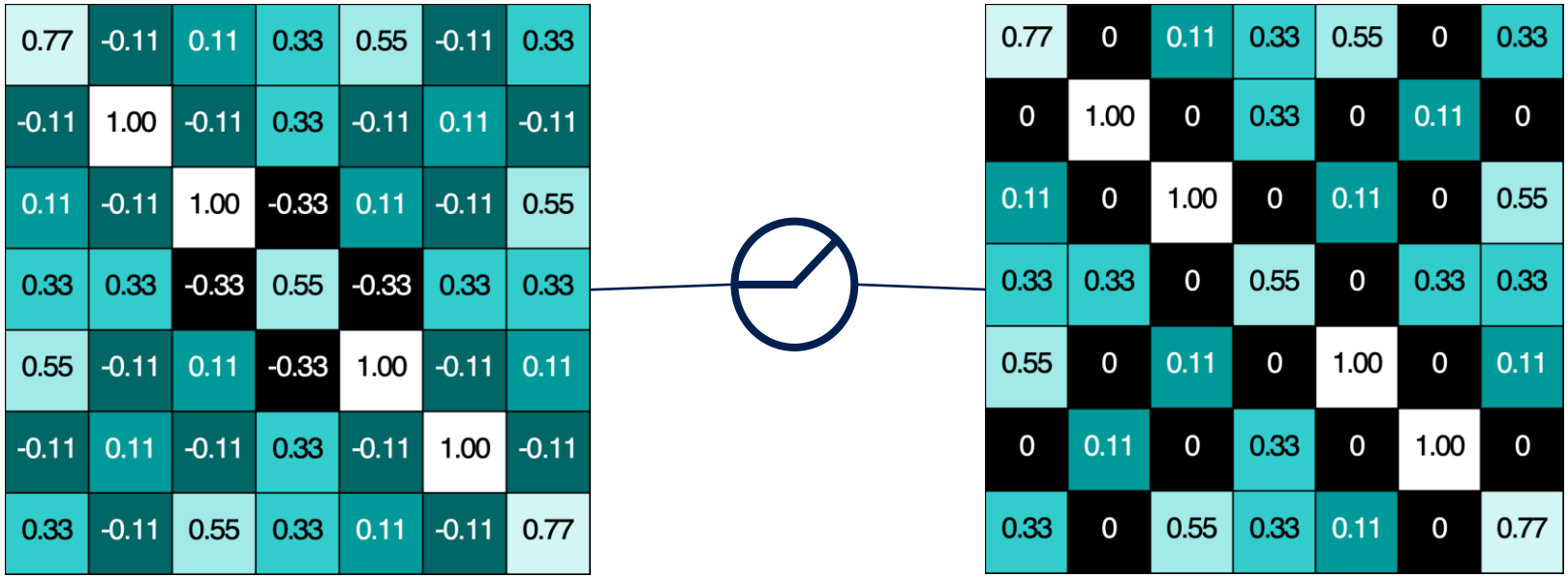
Rectified Linear Units (ReLUs)



Rectified Linear Units (ReLUs)



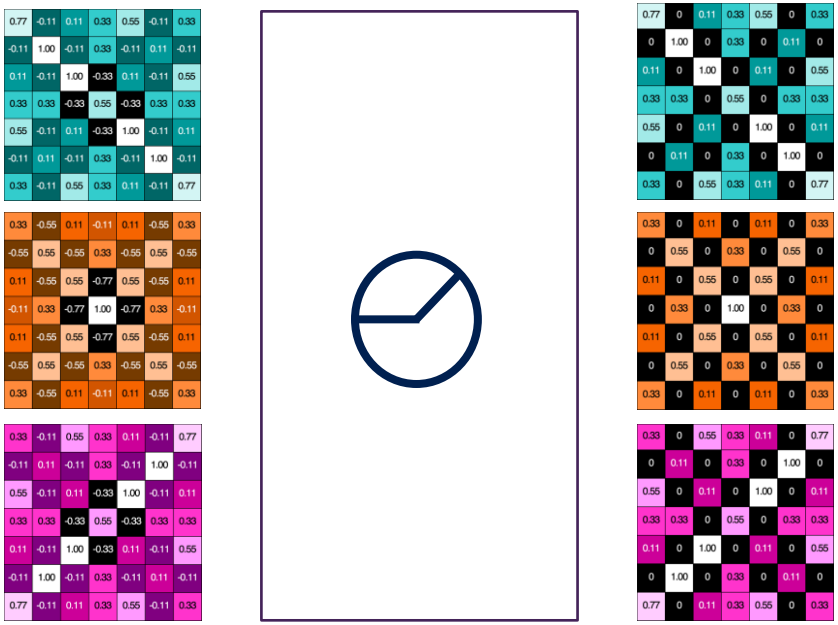
Rectified Linear Units (ReLUs)



Brandon Rohrer https://brohrer.github.io/how_convolutional_neural_networks_work.html

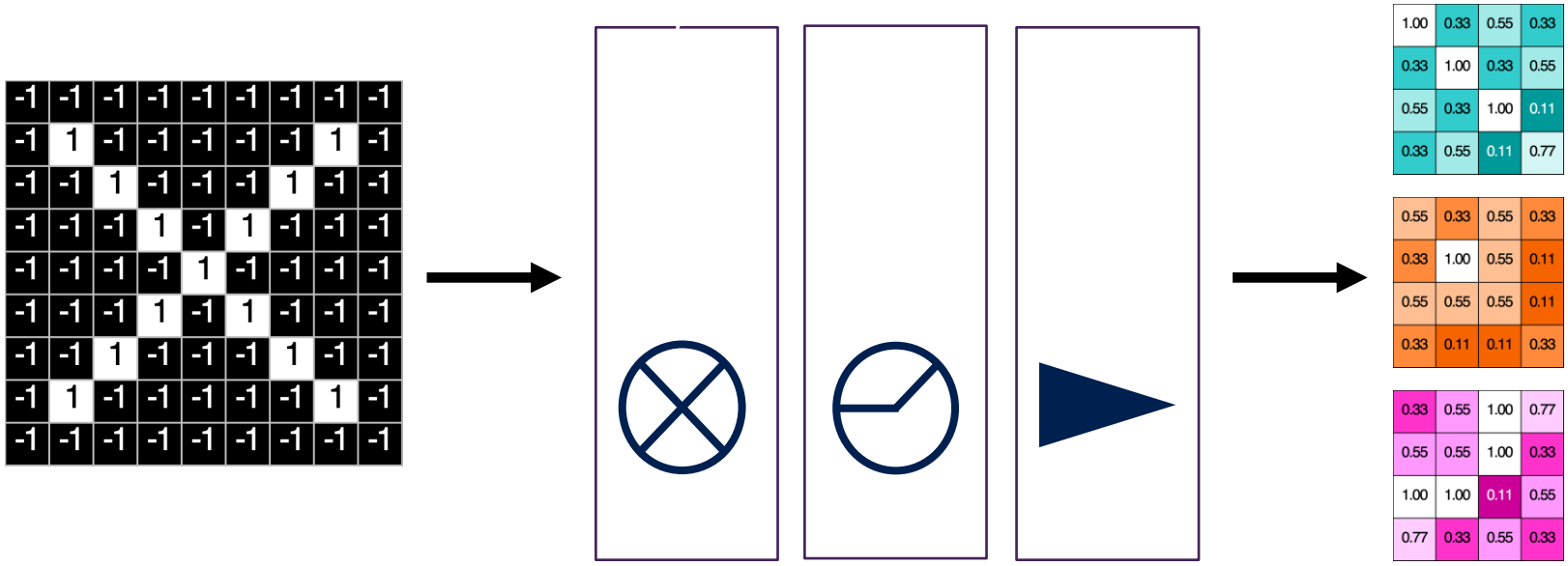
ReLU layer

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



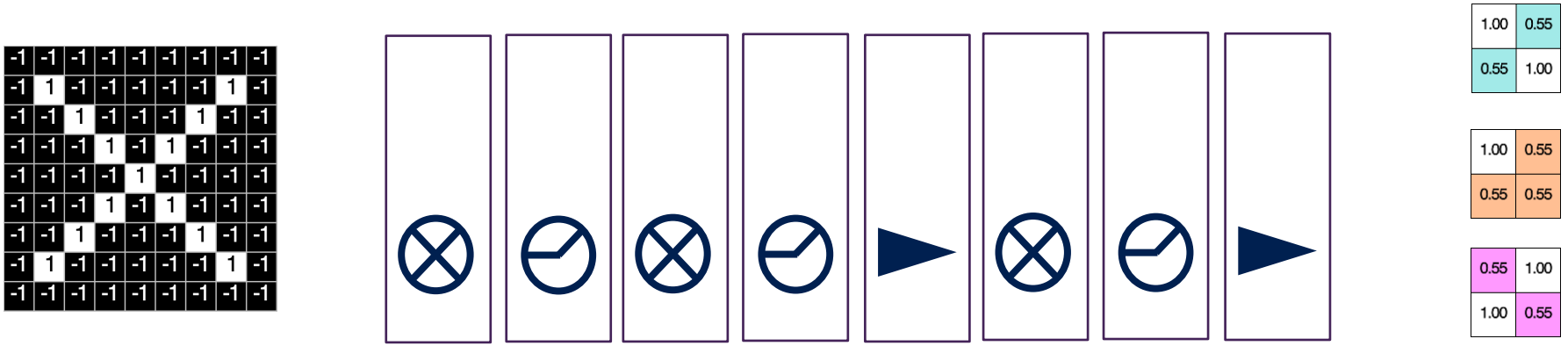
Layers get stacked

The output of one becomes the input of the next.



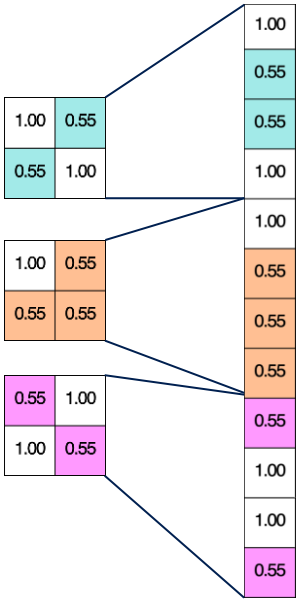
Deep stacking

Layers can be repeated several (or many) times.

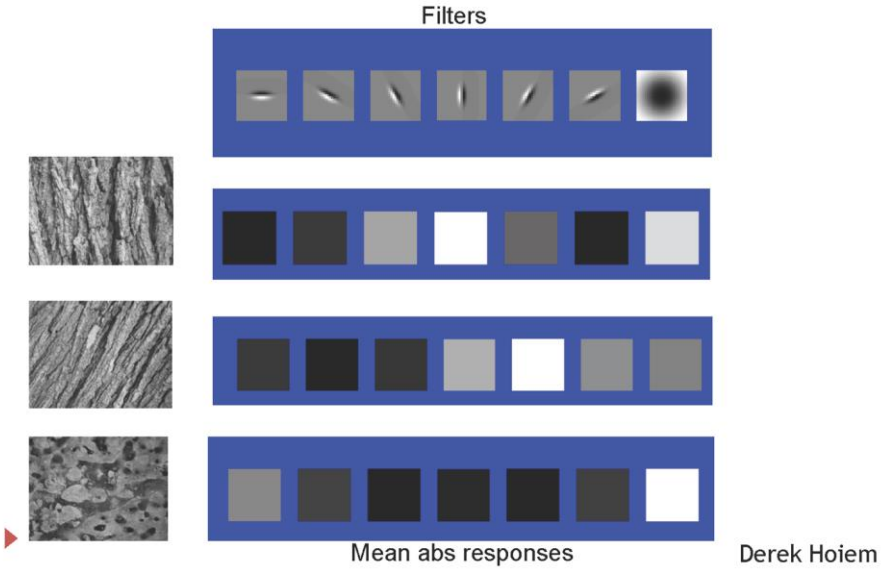


Fully connected layer

Every value gets a vote

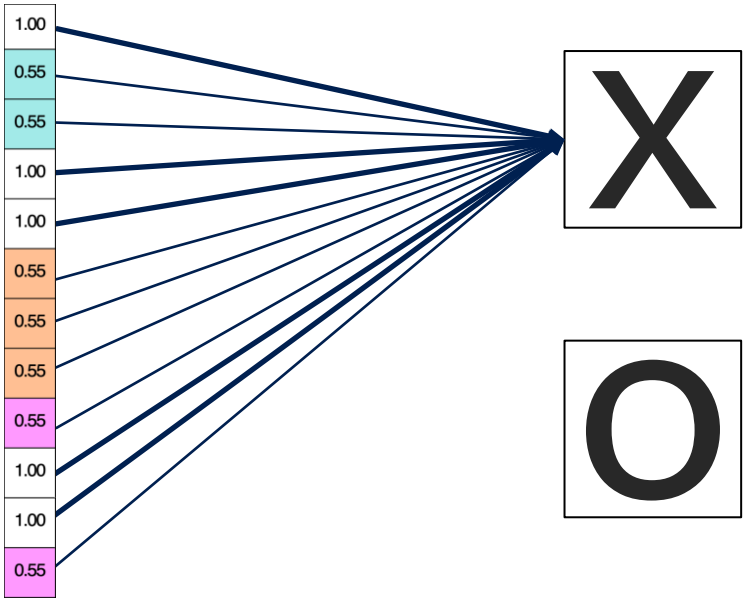


Representing texture by mean abs response



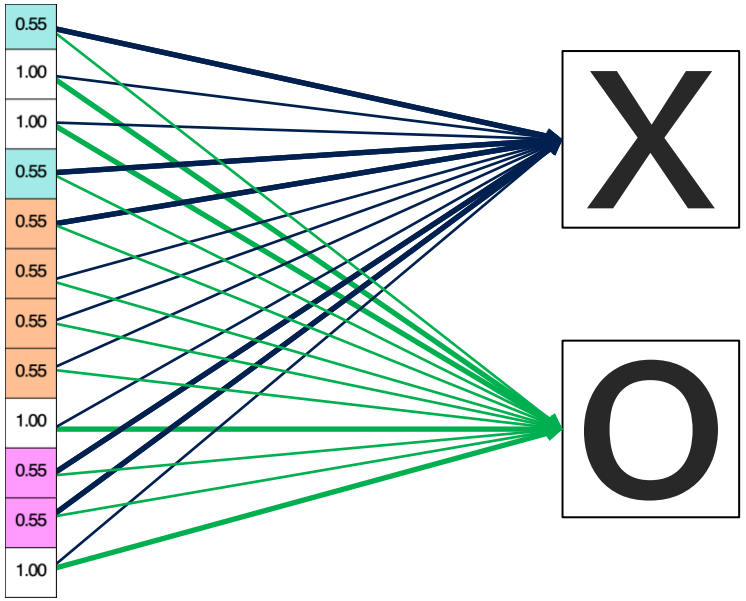
Fully connected layer

Vote depends on how strongly a value predicts X or O



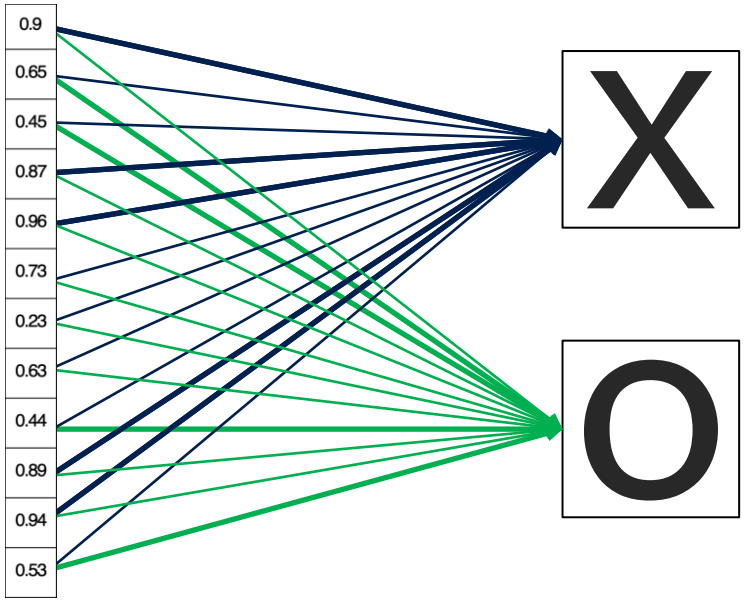
Fully connected layer

Vote depends on how strongly a value predicts X or O



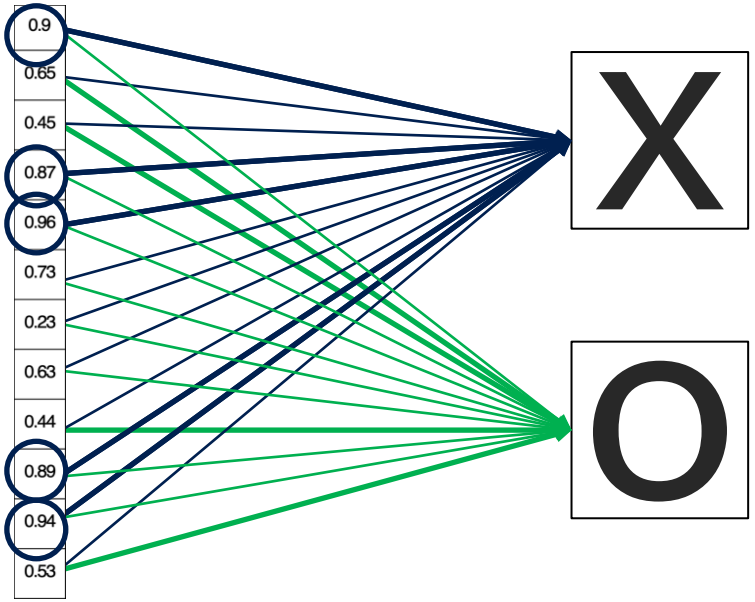
Fully connected layer

Future values vote on X or O



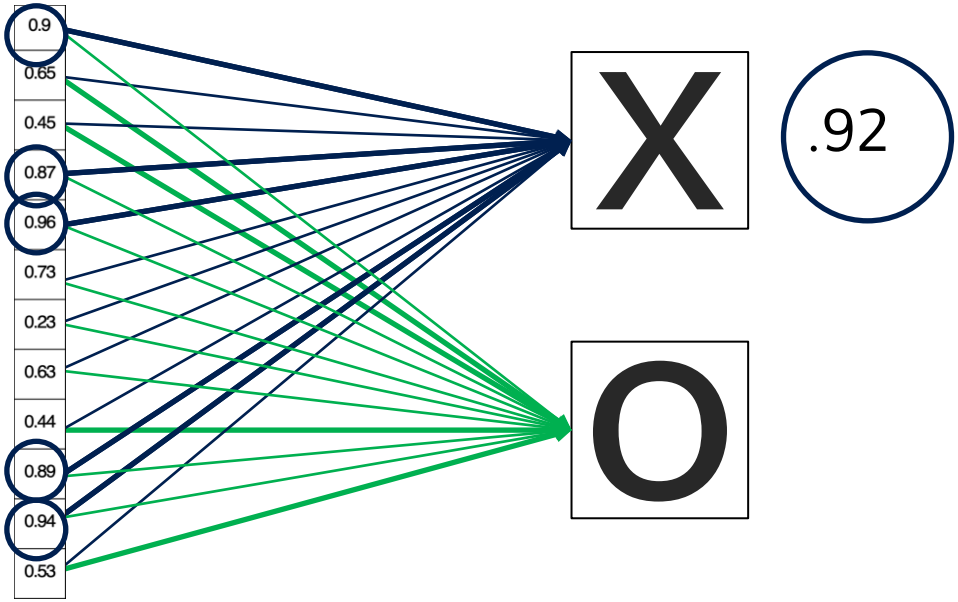
Fully connected layer

Future values vote on X or O



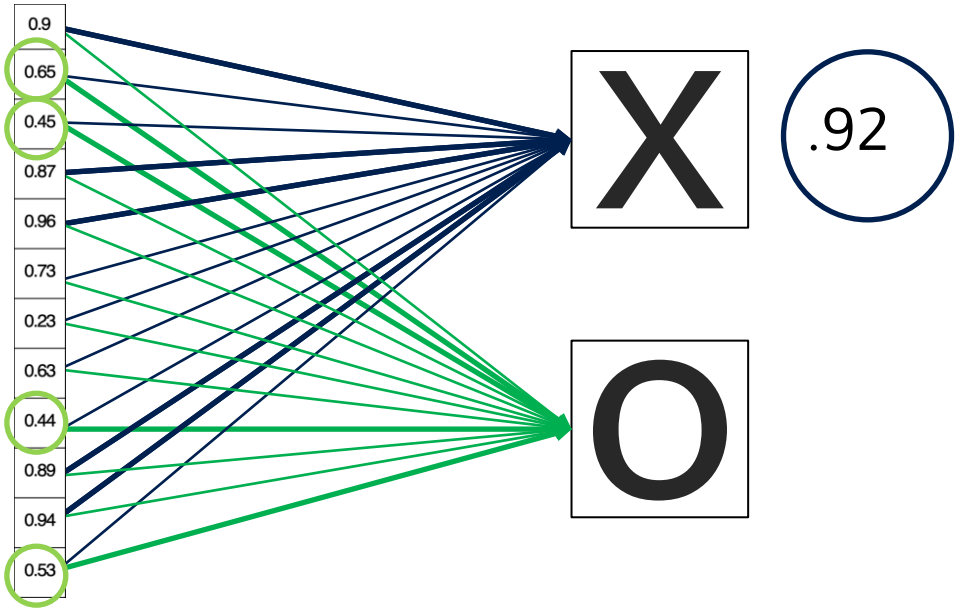
Fully connected layer

Future values vote on X or O



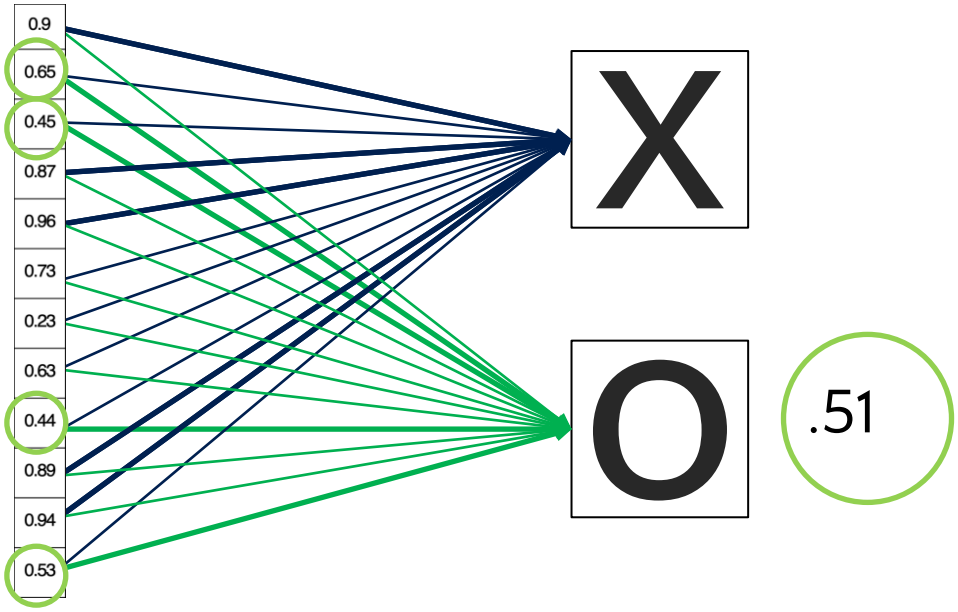
Fully connected layer

Future values vote on X or O



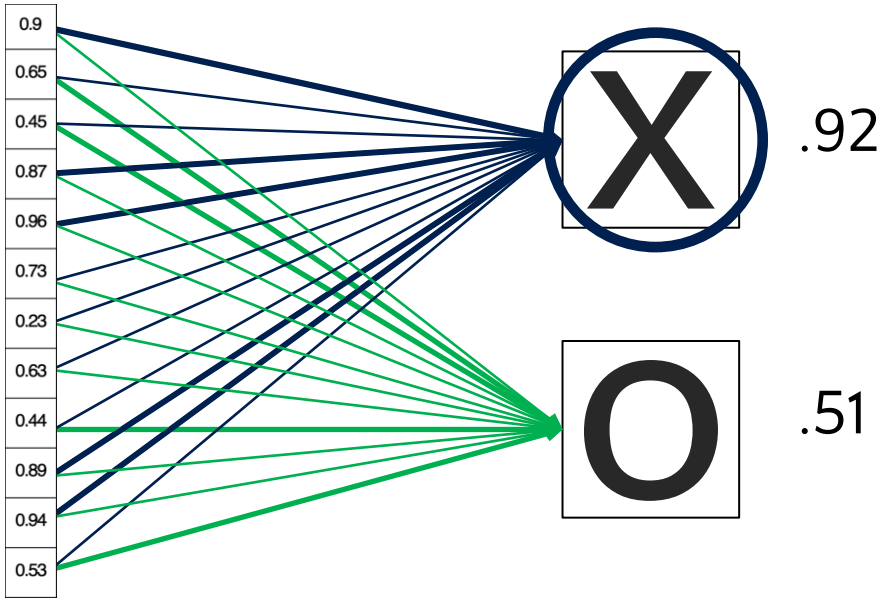
Fully connected layer

Future values vote on X or O



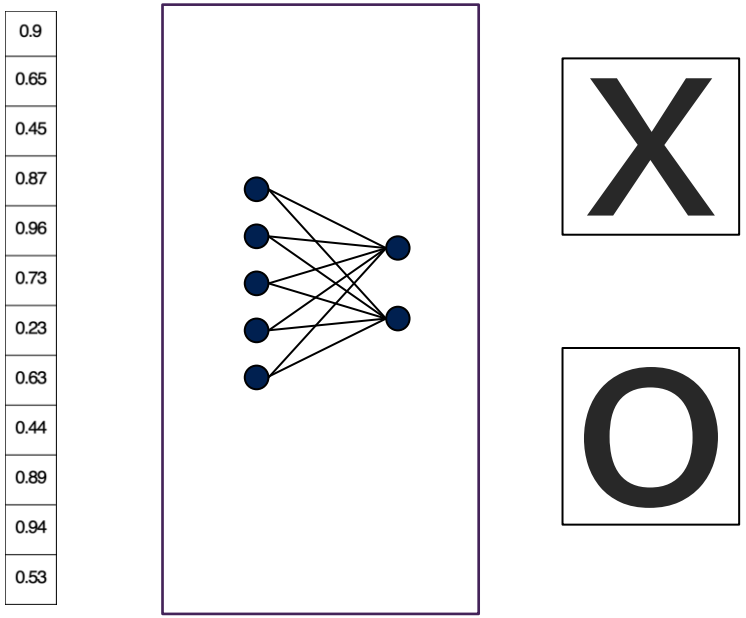
Fully connected layer

Future values vote on X or O



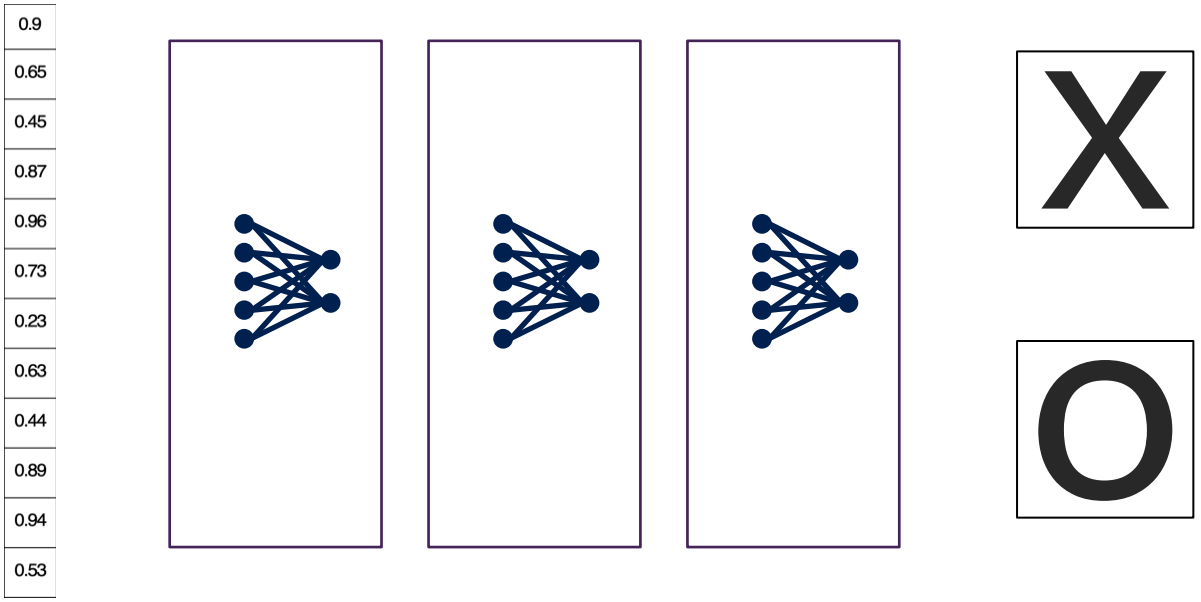
Fully connected layer

A list of feature values becomes a list of votes.



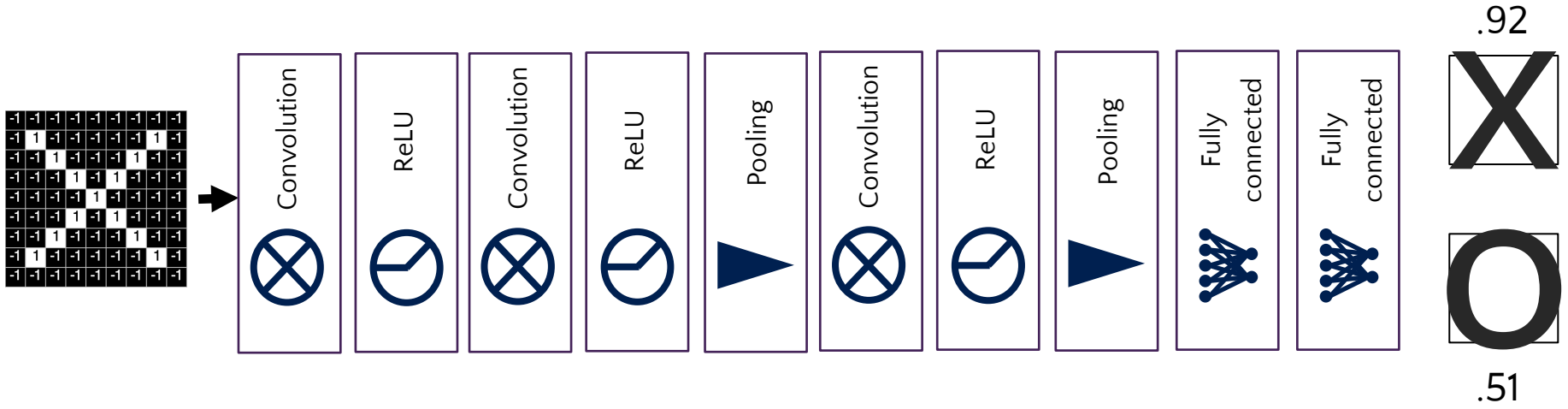
Fully connected layer

These can also be stacked.

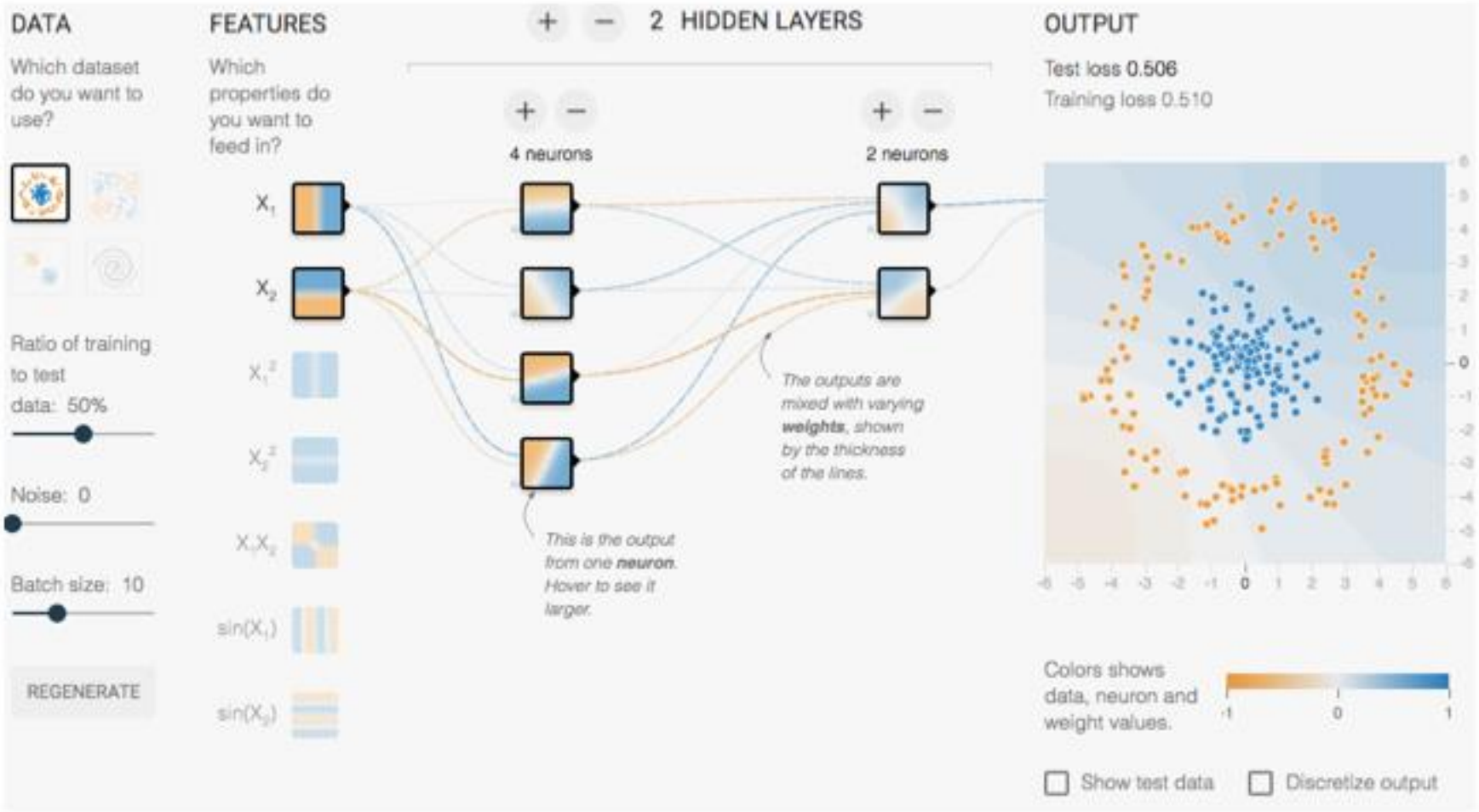


Putting it all together

A set of pixels becomes a set of votes.



Demo: http://playground.tensorflow.org/



Learning

Q: Where do all the magic numbers come from?

Features in convolutional layers

Voting weights in fully connected layers

A: Backpropagation

Hyperparameters (knobs)

Convolution

- Number of features
- Size of features

Pooling

- Window size
- Window stride

Fully Connected

- Number of neurons

Architecture

How many of each type of layer?

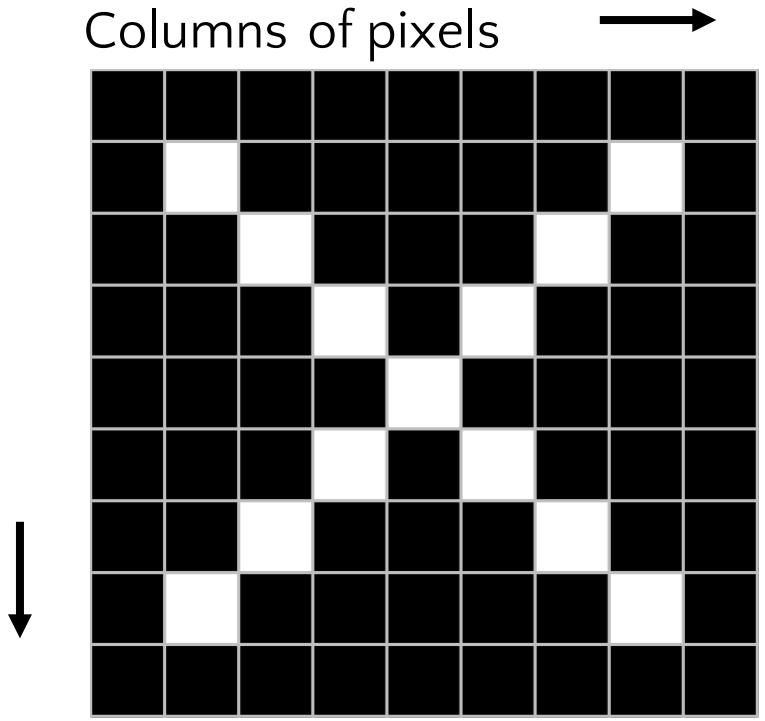
In what order?

Not just images

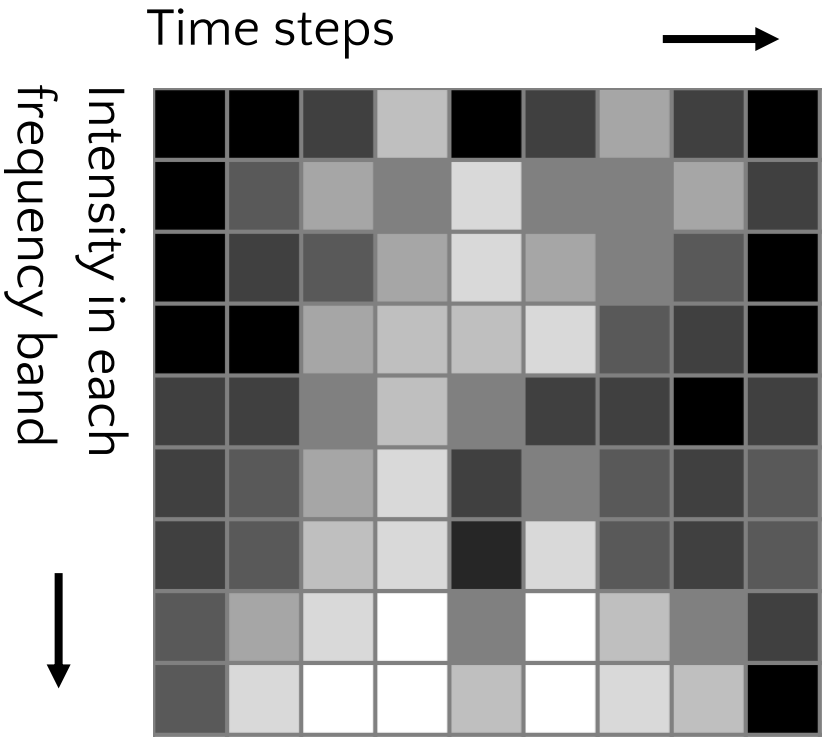
Any 2D (or 3D) data.

Things closer together are more closely related than things far away.

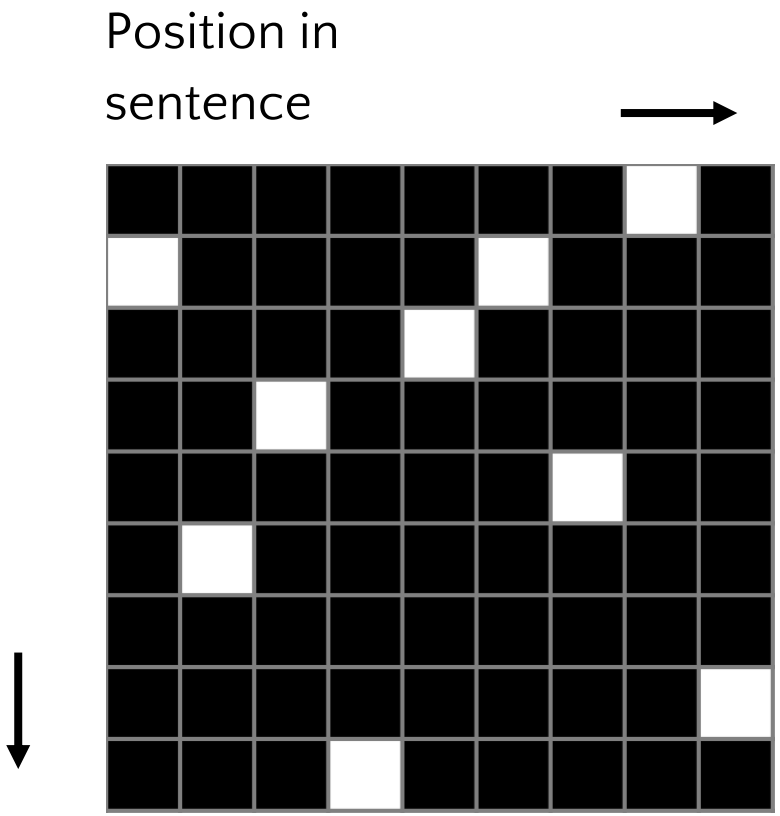
Images



Sound



Text



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.

Customer data

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

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

More Details

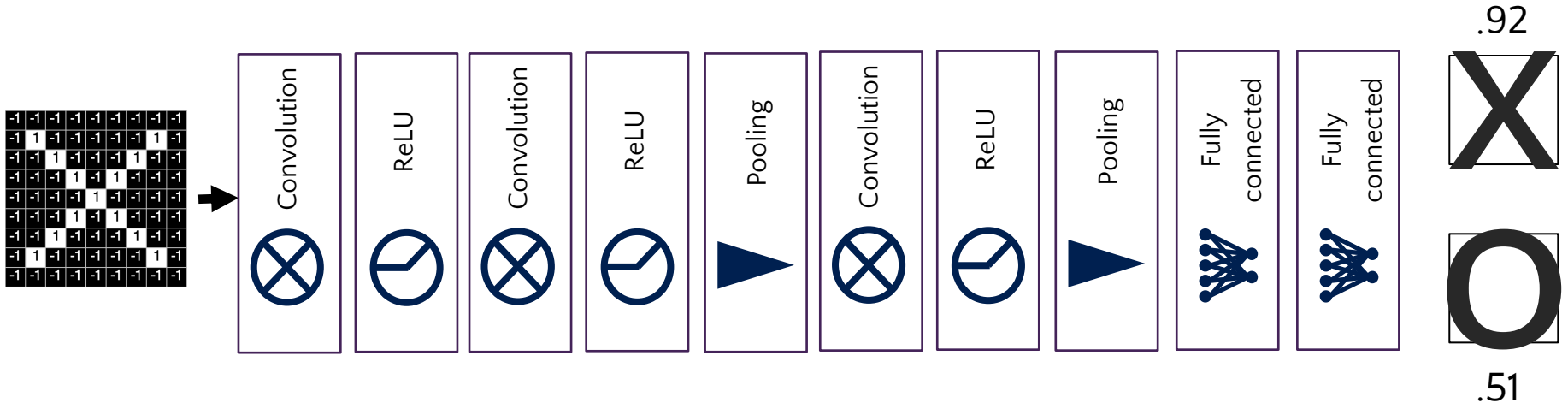
- Linear Regression
- https://youtu.be/EI6GES6q3wo

- Gradient Descent & Neural Network
- https://youtu.be/nE3FWsBU3Gs

- Multi-layer Neural Network Backpropagation
- https://youtu.be/827Pg9fn_1o

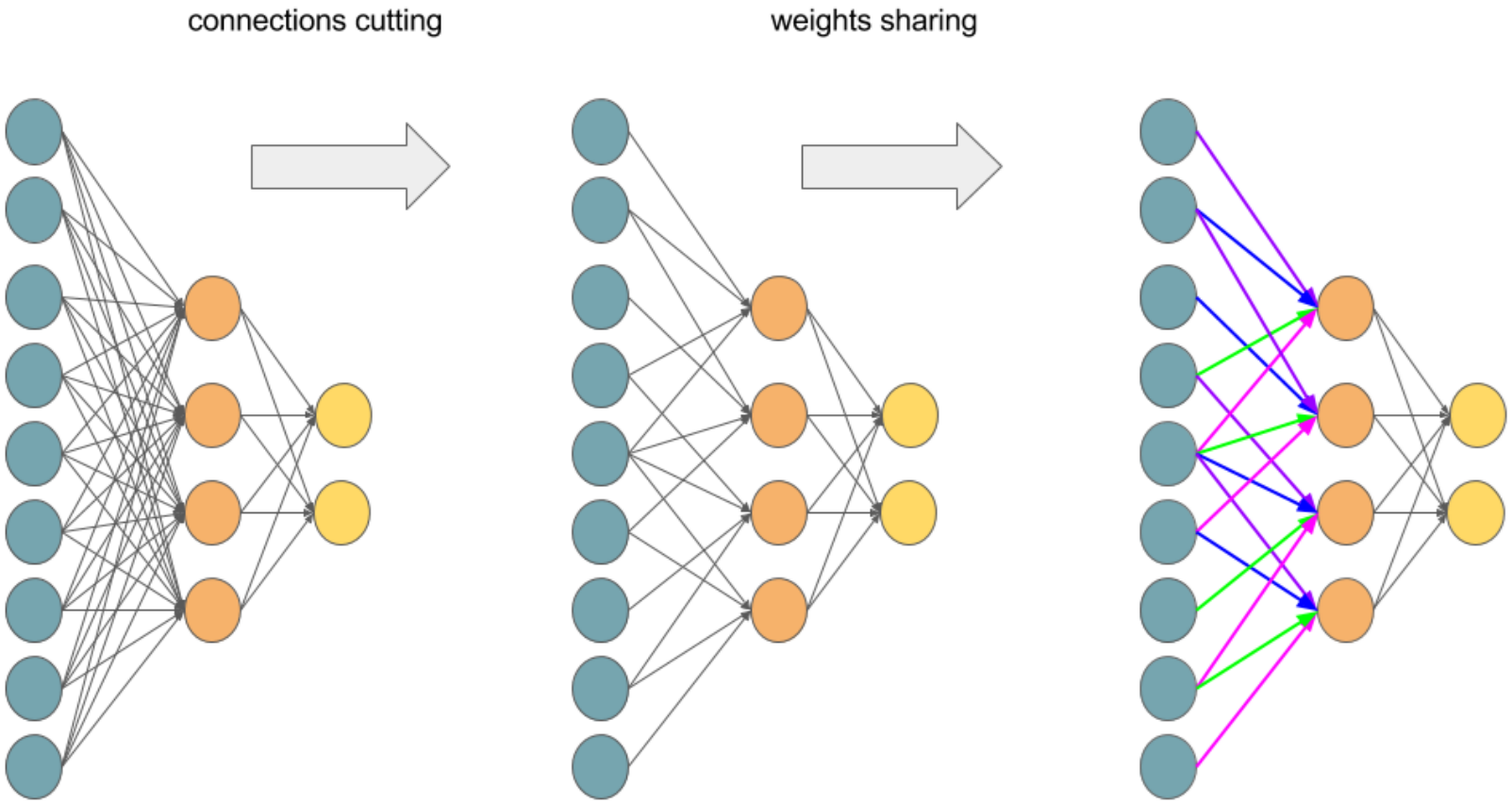
Recap: Putting it all together

A set of pixels becomes a set of votes.



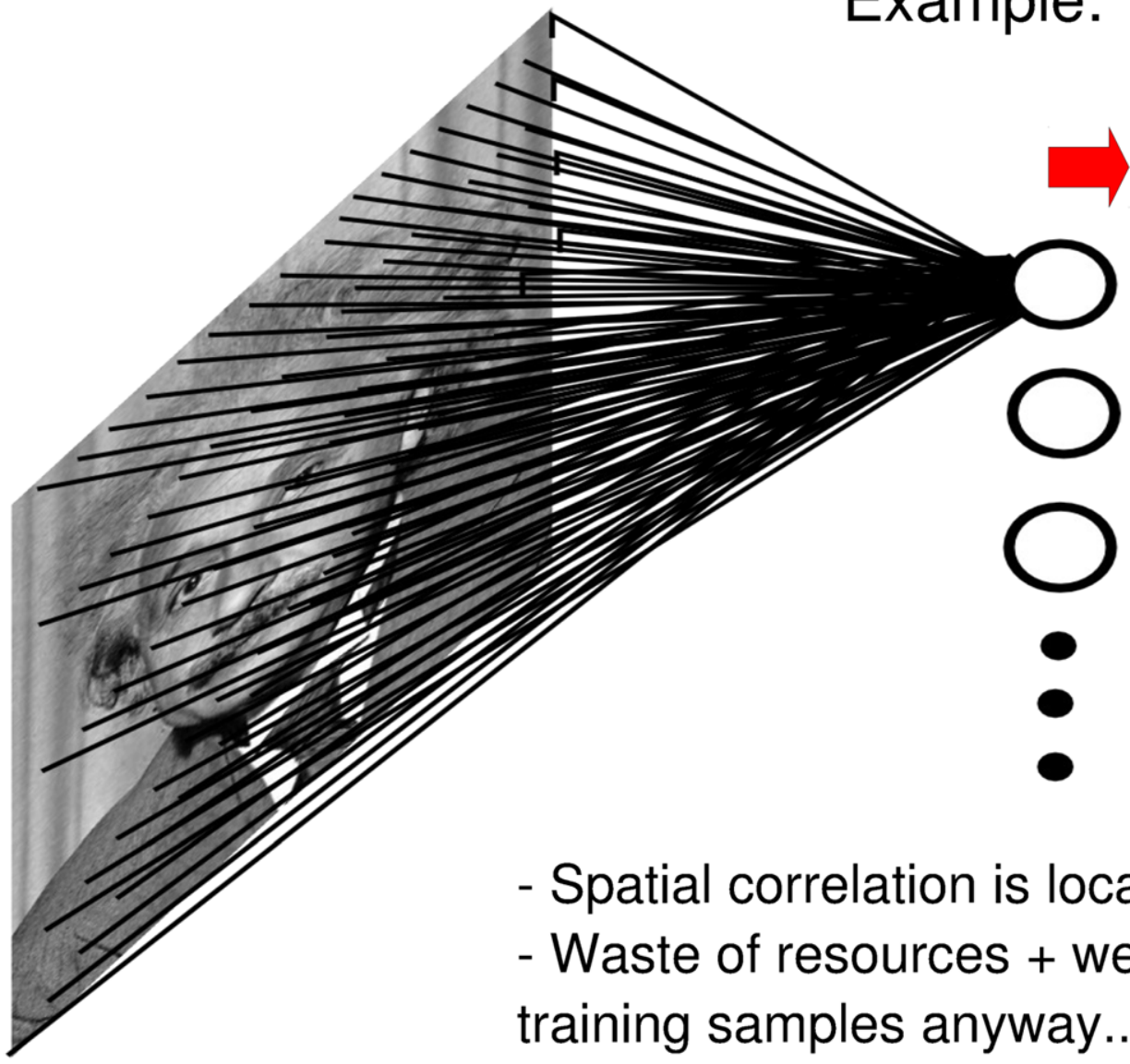
Any Questions?

Transforming Multilayer Perceptron to CNN



https://grzegorzgwardys.wordpress.com/2016/04/22/8/

Fully Connected Layer



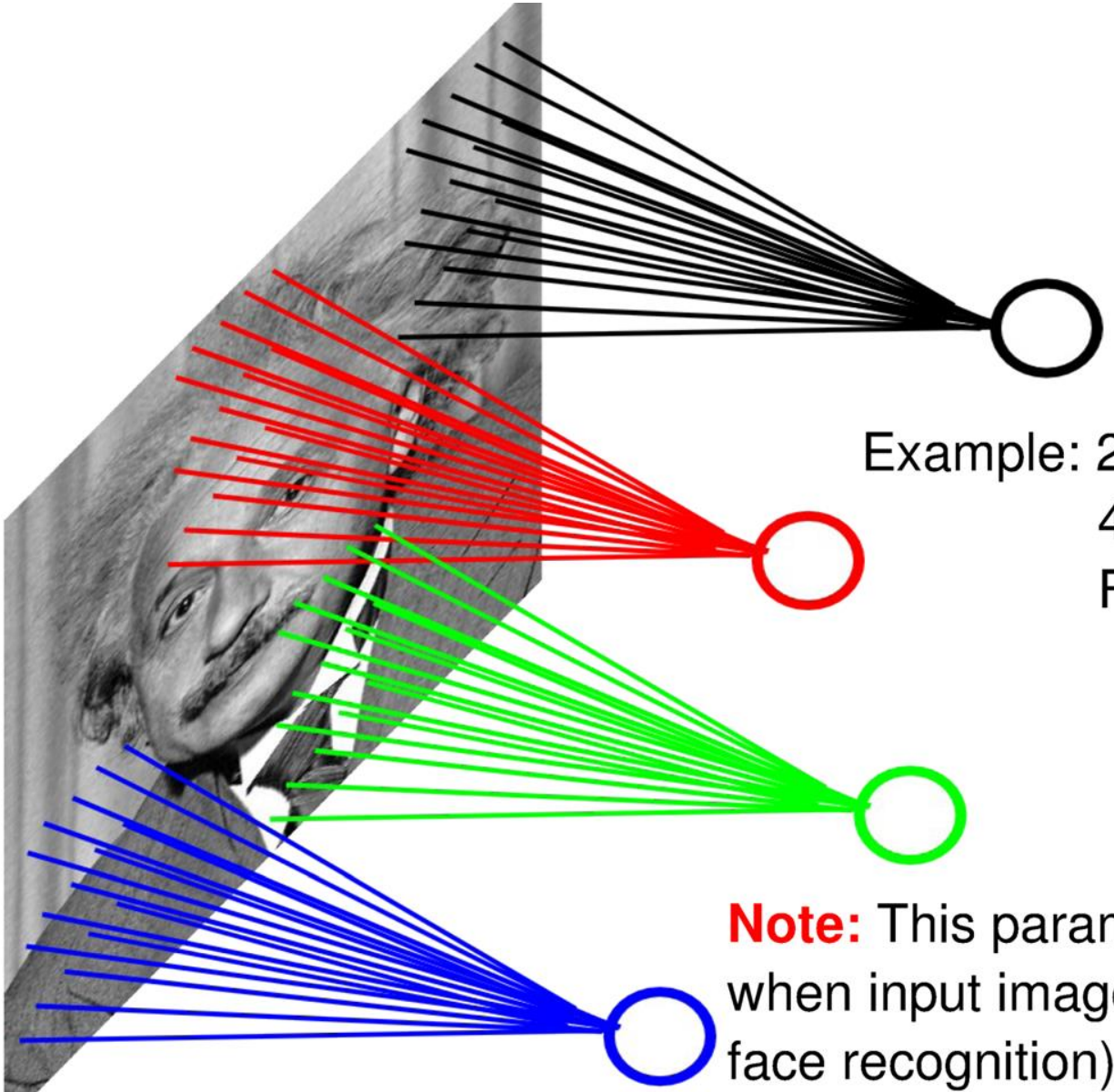
Example: 200x200 image

40K hidden units

~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

Locally Connected Layer



Example: 200x200 image
40K hidden units
Filter size: 10x10

Note: This parameterization is good when input image is registered (e.g., face recognition).

Next…More on CNNs

- Volumetric Convolution Filter

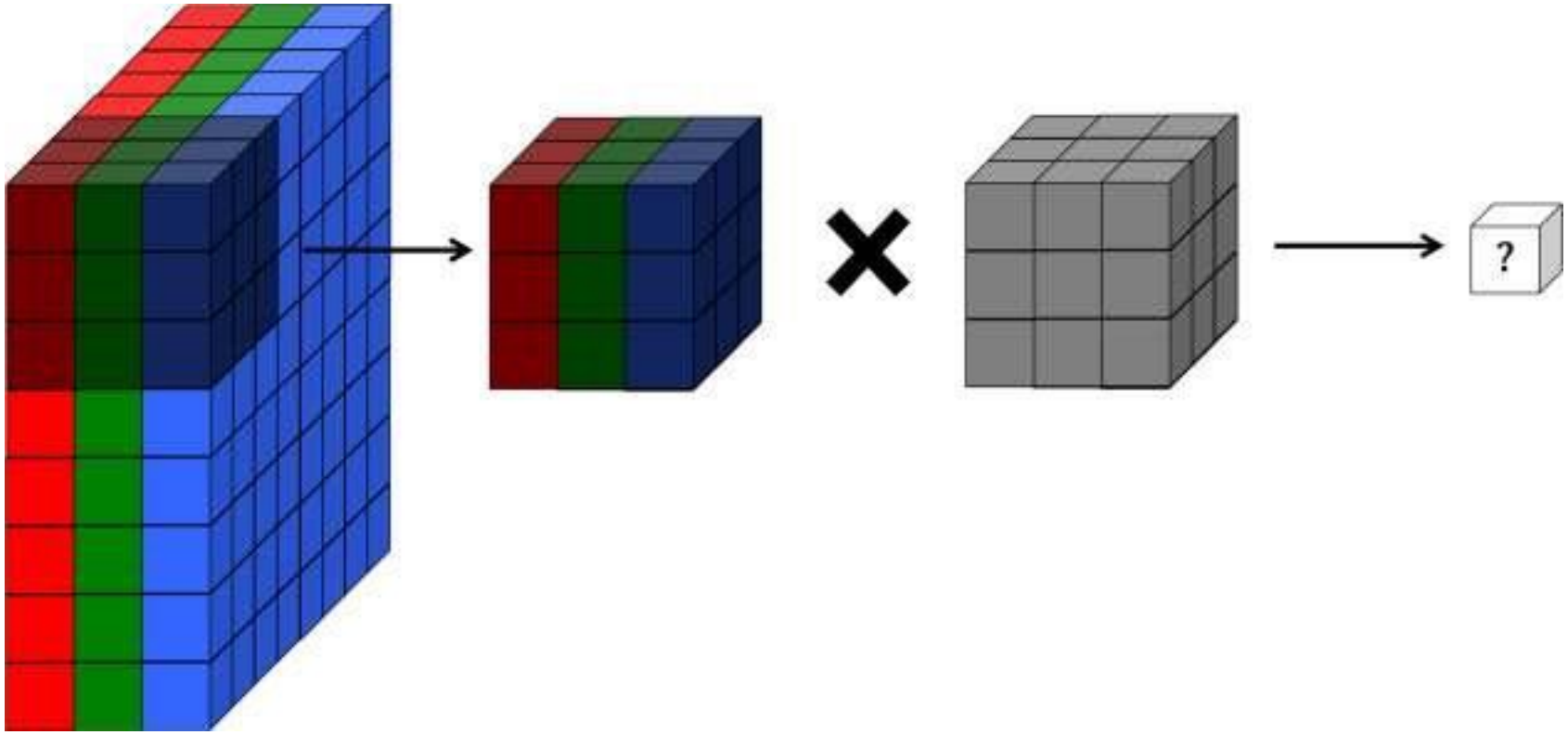


Figure 5-8. Representing a full-color RGB image as a volume and applying a volumetric convolutional filter

Volumetric Convolution Filter

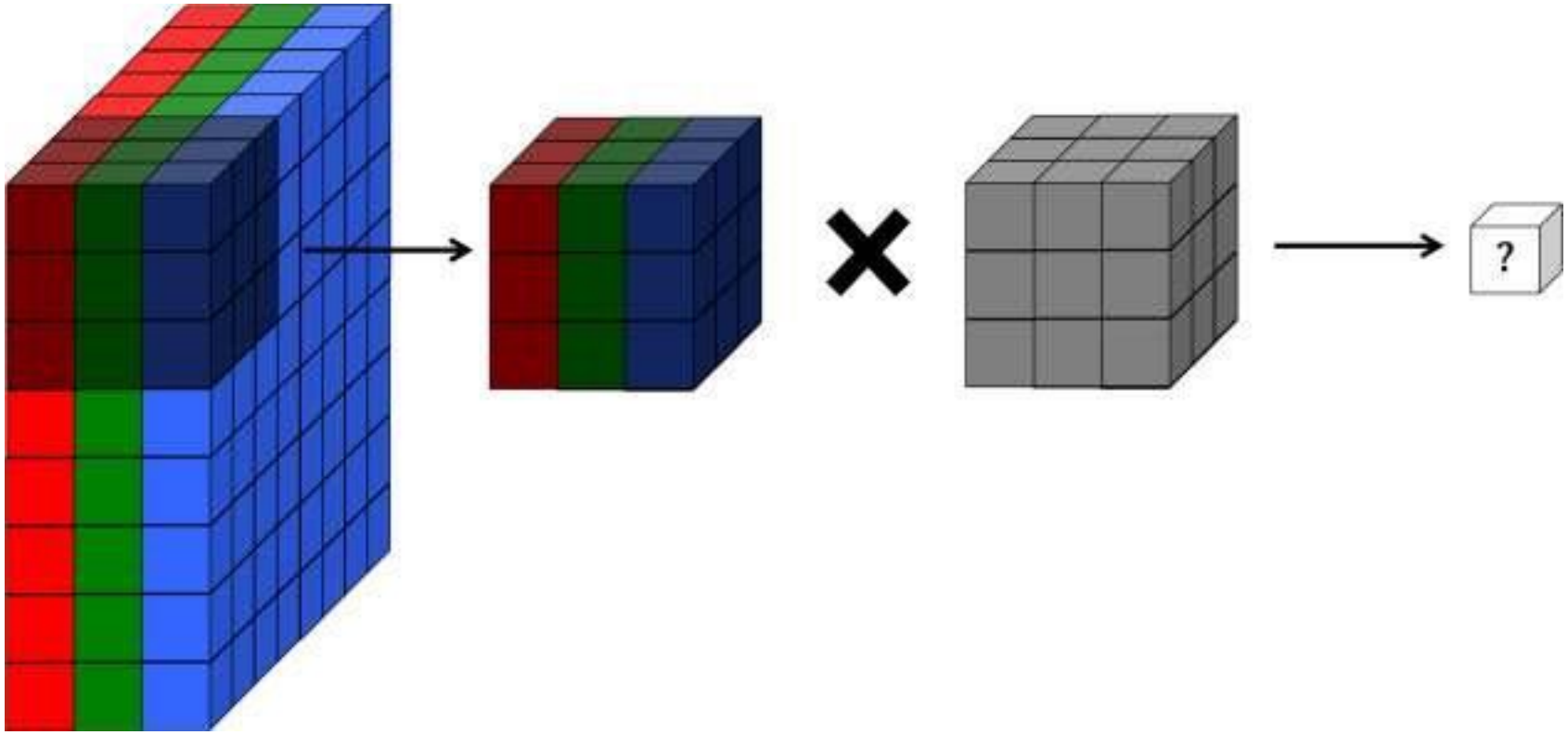


Figure 5-8. Representing a full-color RGB image as a volume and applying a volumetric convolutional filter

Volumetric Convolution Filter

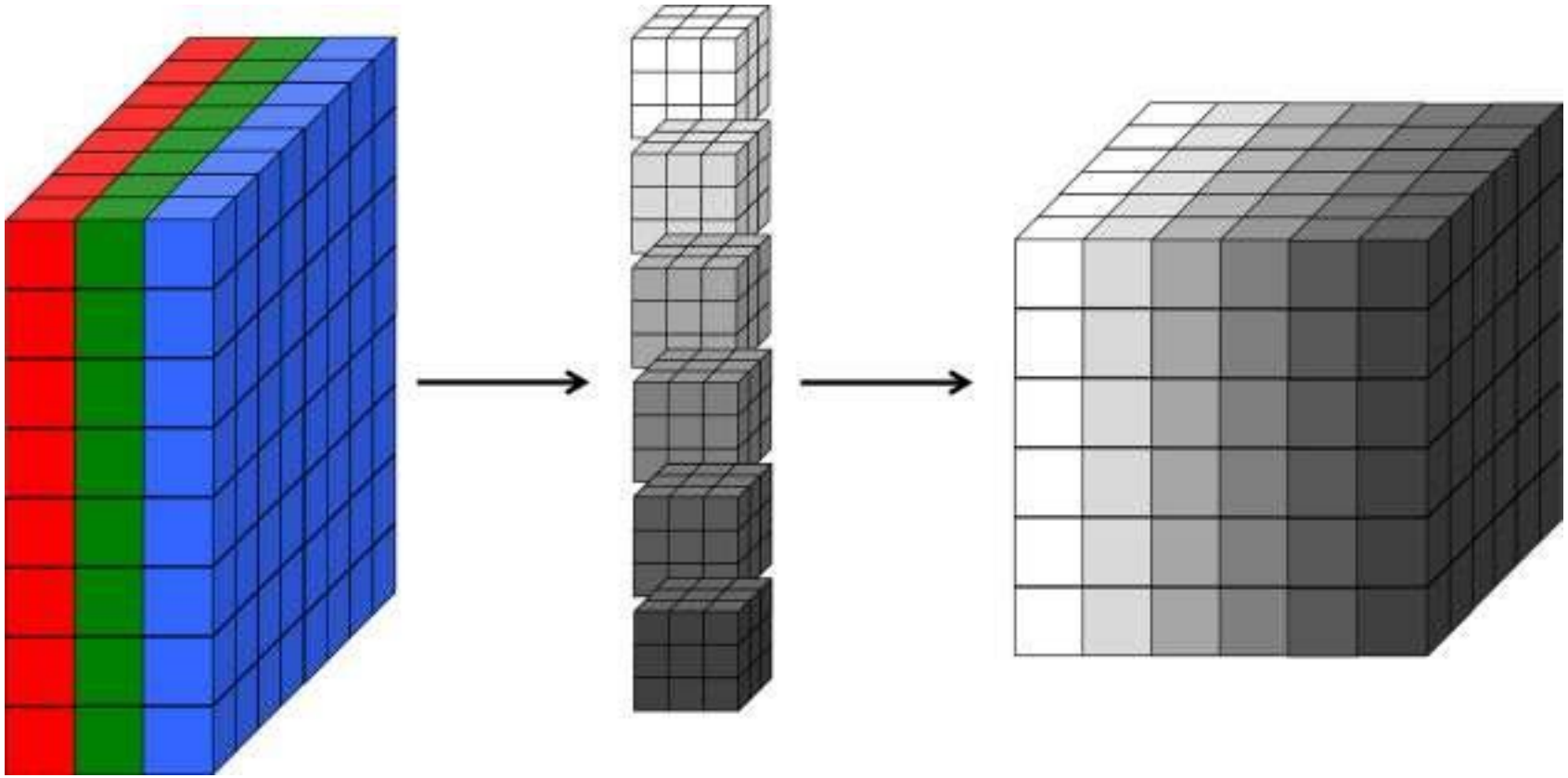
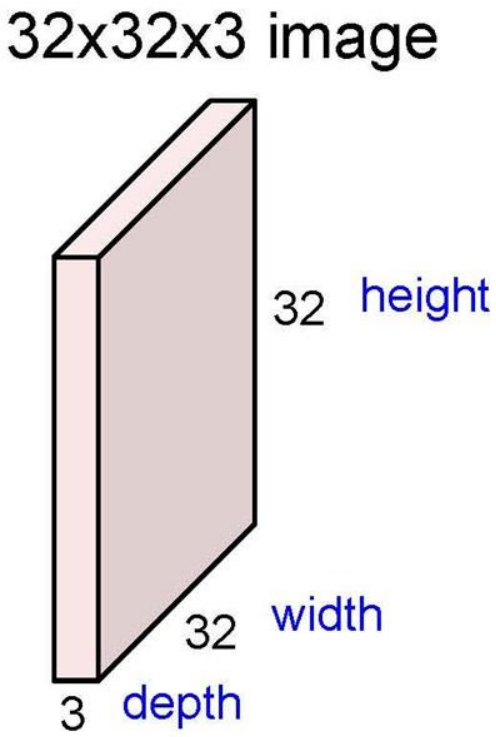
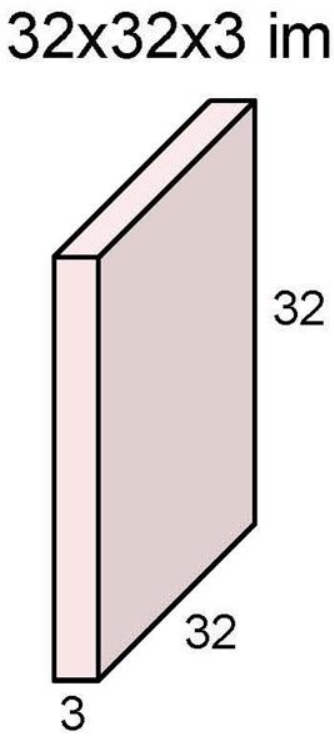


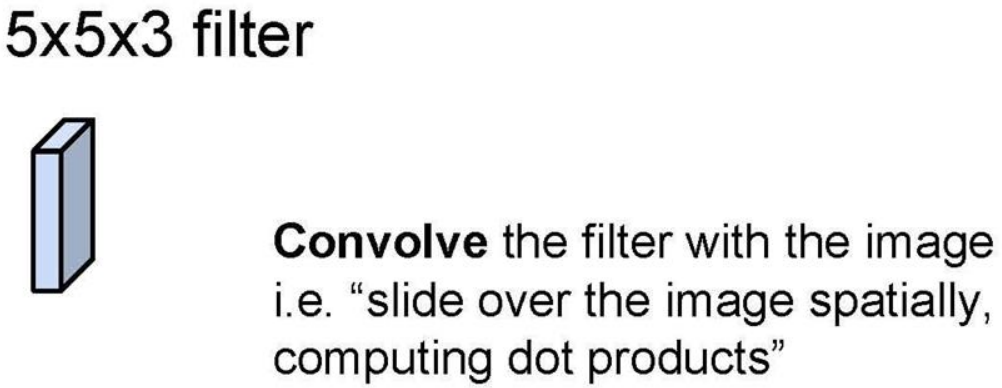
Figure 5-9. A three-dimensional visualization of a convolutional layer, where each filter corresponds to a slice in the resulting output volume

Convolution Layer

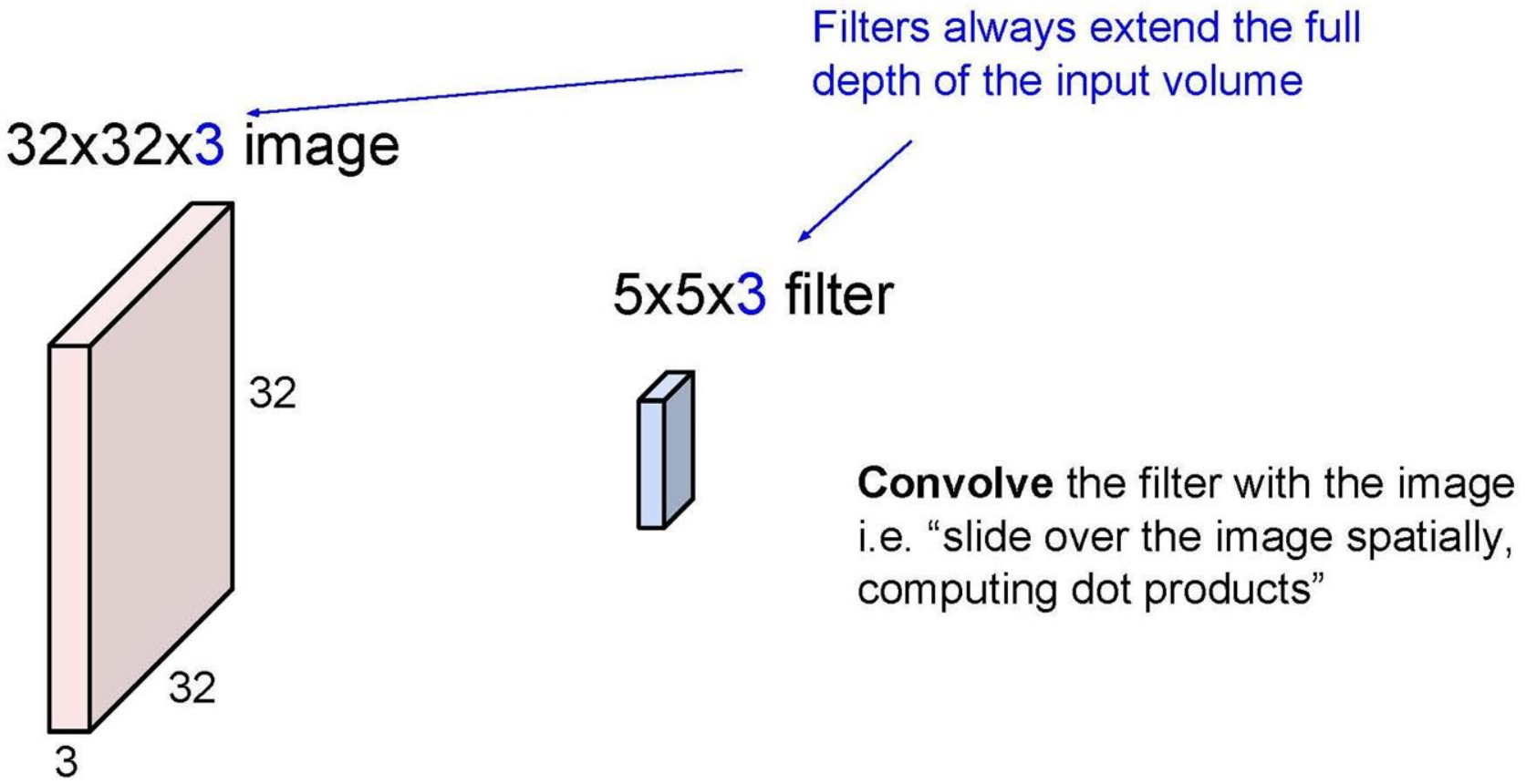


Convolution Layer

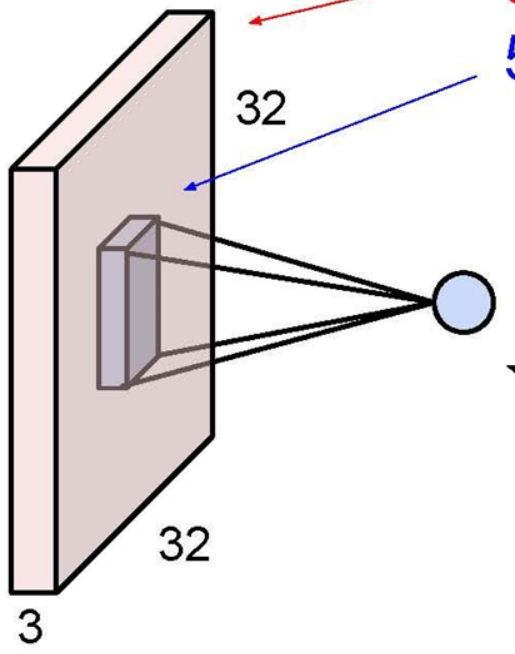




Convolution Layer



Convolution Layer



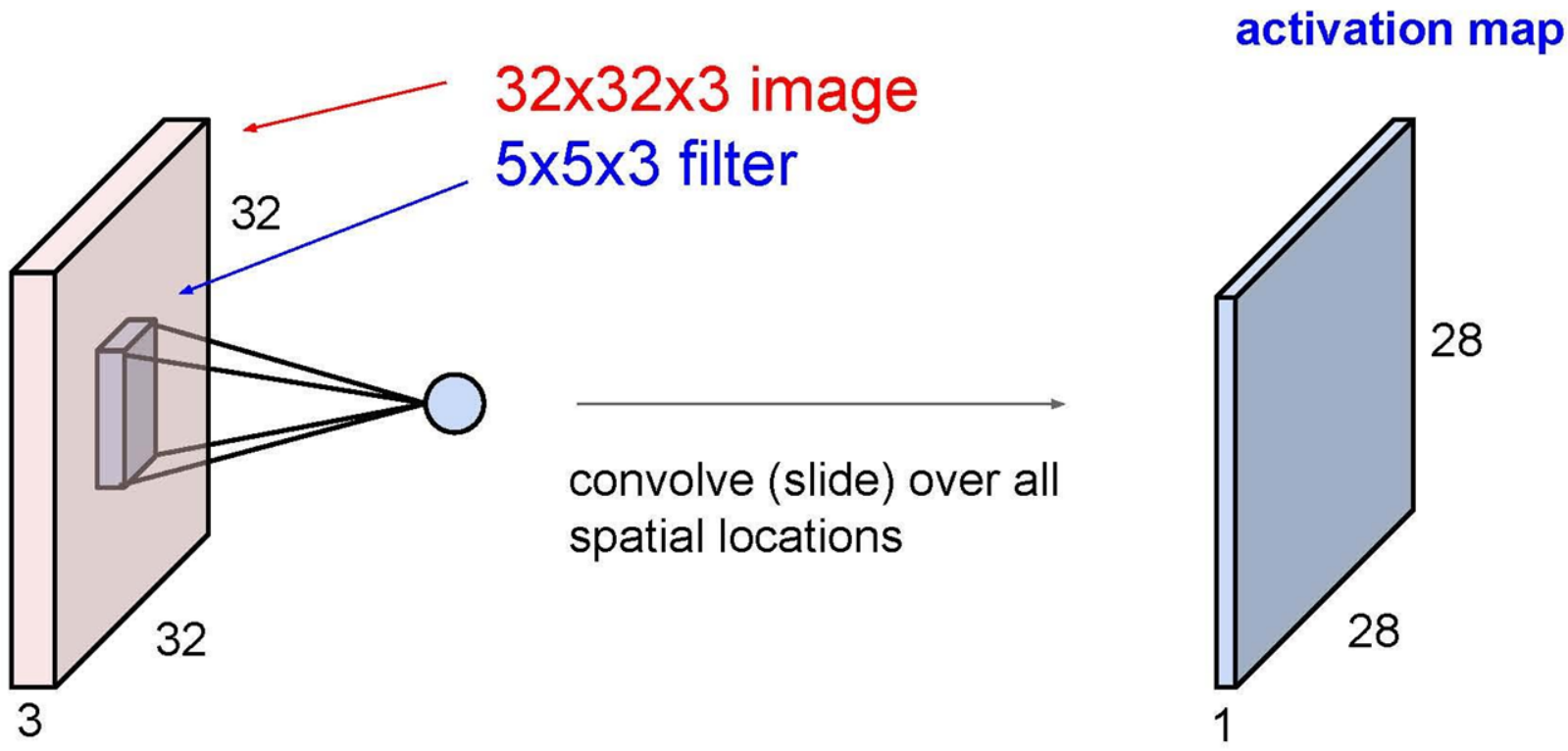
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

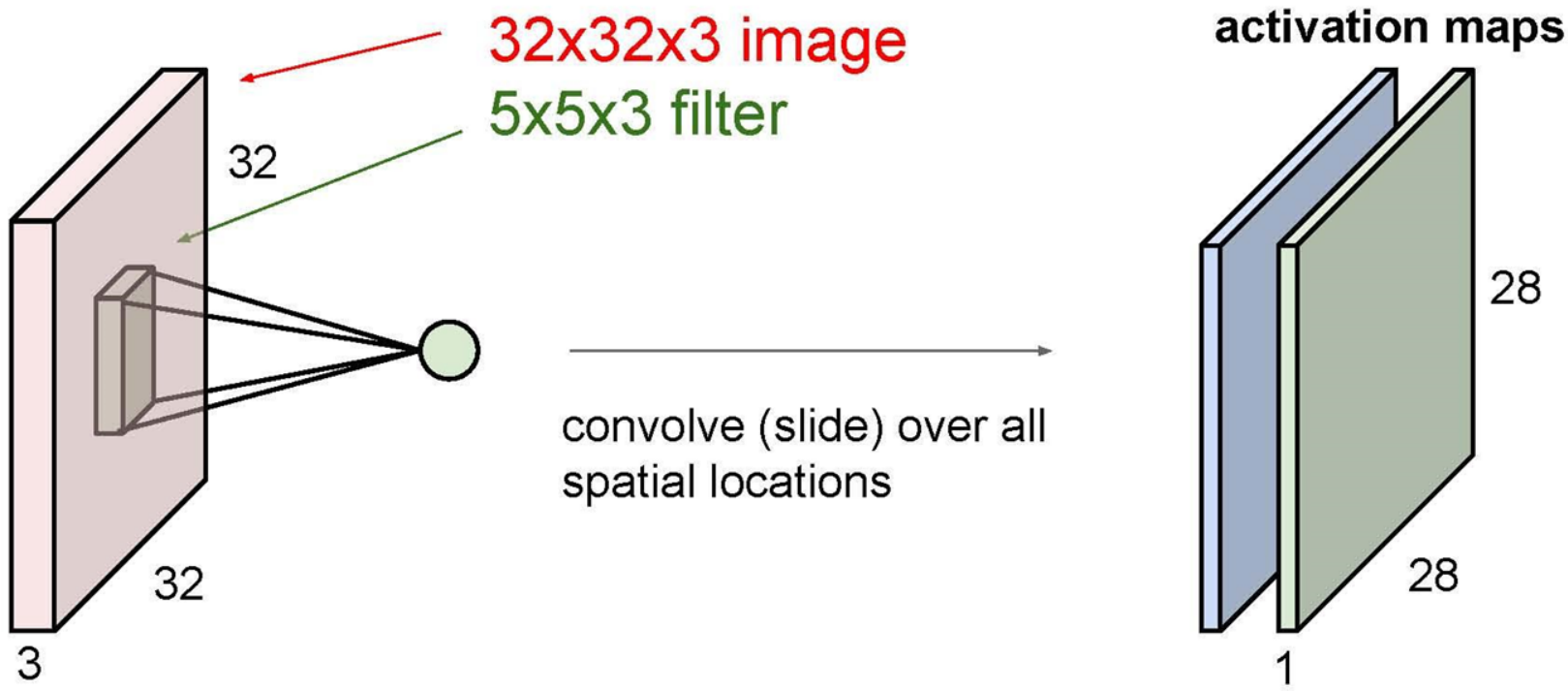
$$w^T x + b$$

Convolution Layer



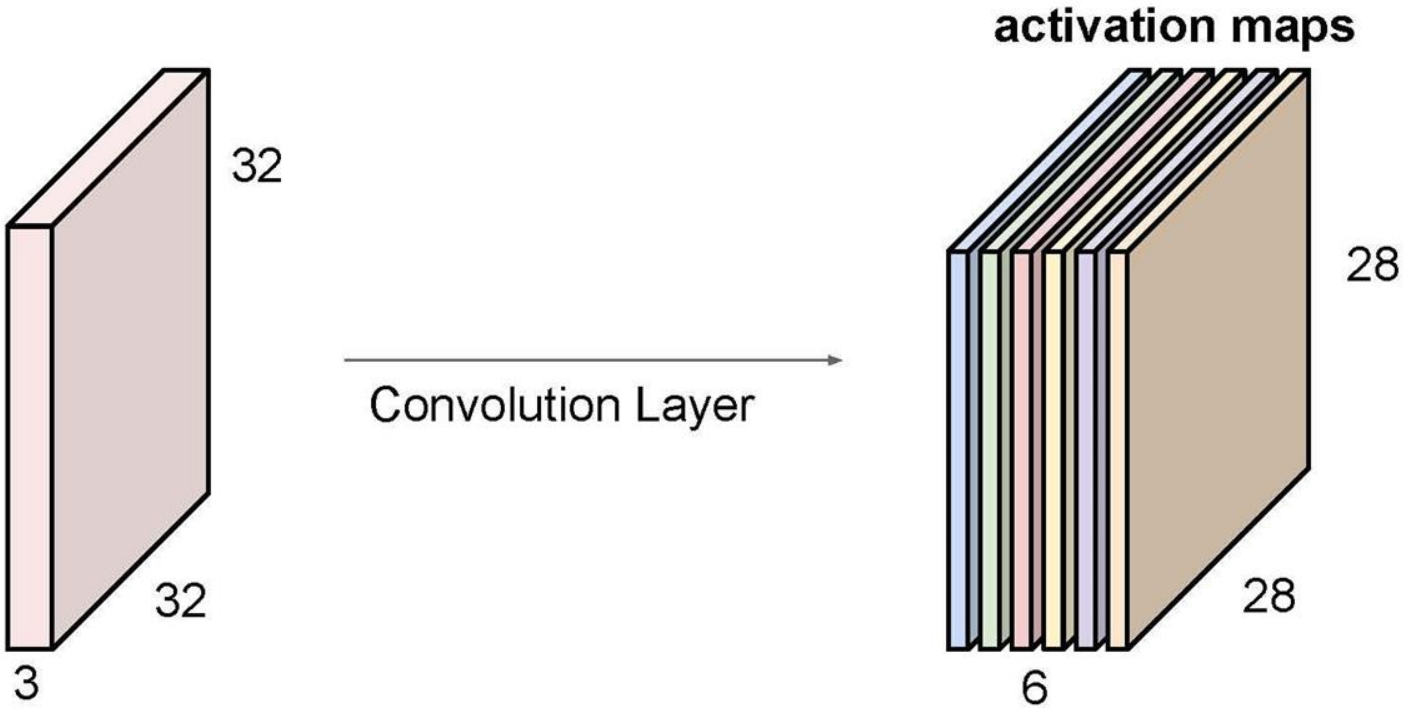
Convolution Layer

consider a second, green filter



Convolution Layer

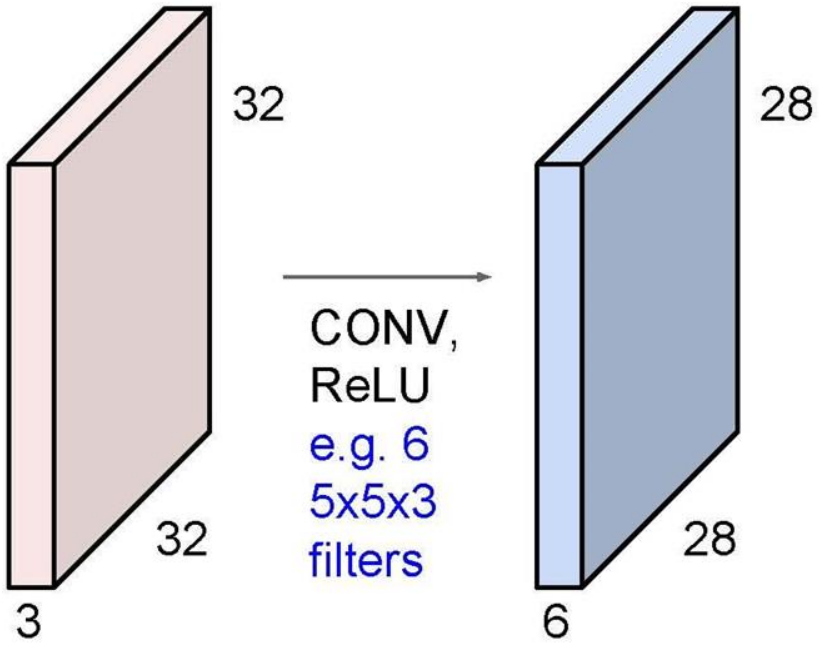
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

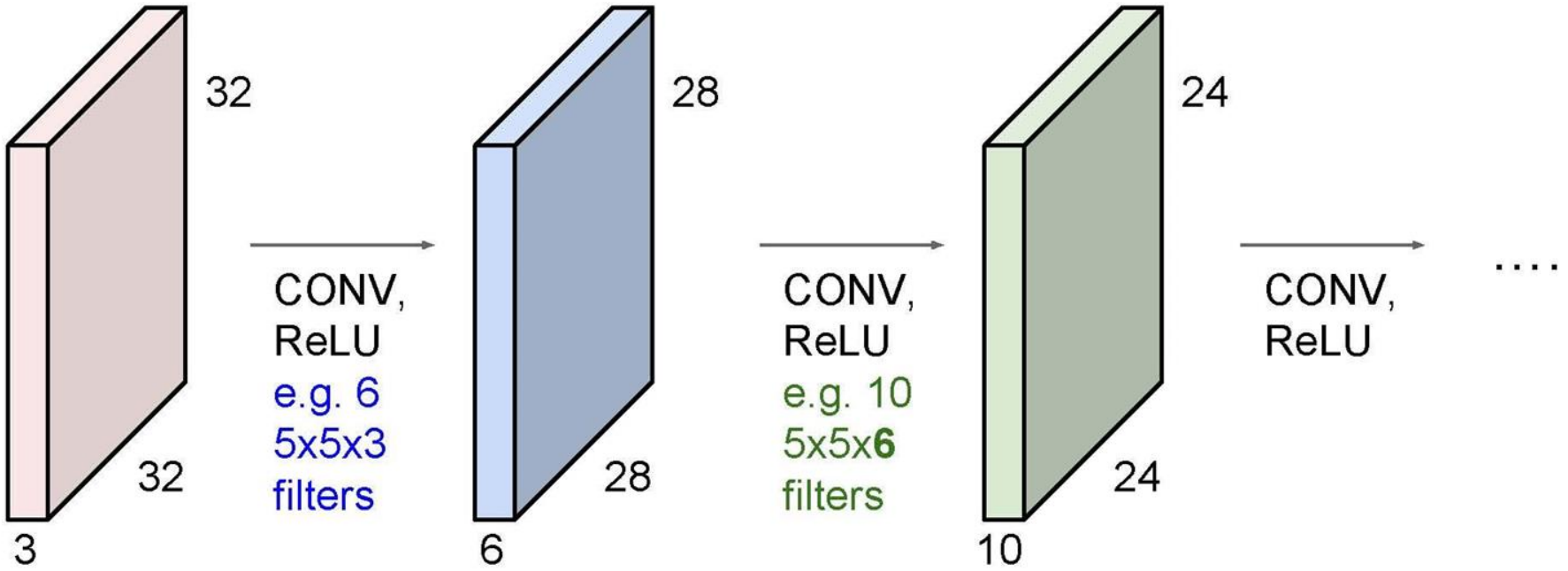
ConvNet

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



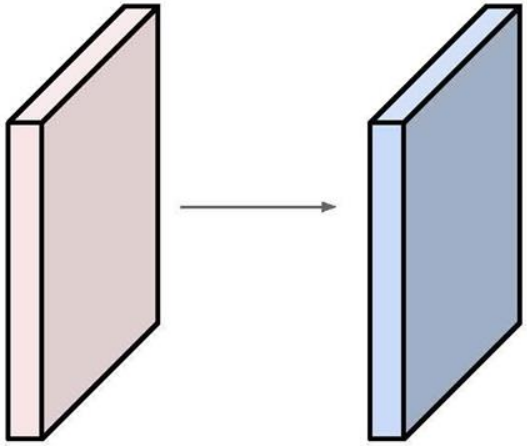
ConvNet

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



ConvNet

Examples time:



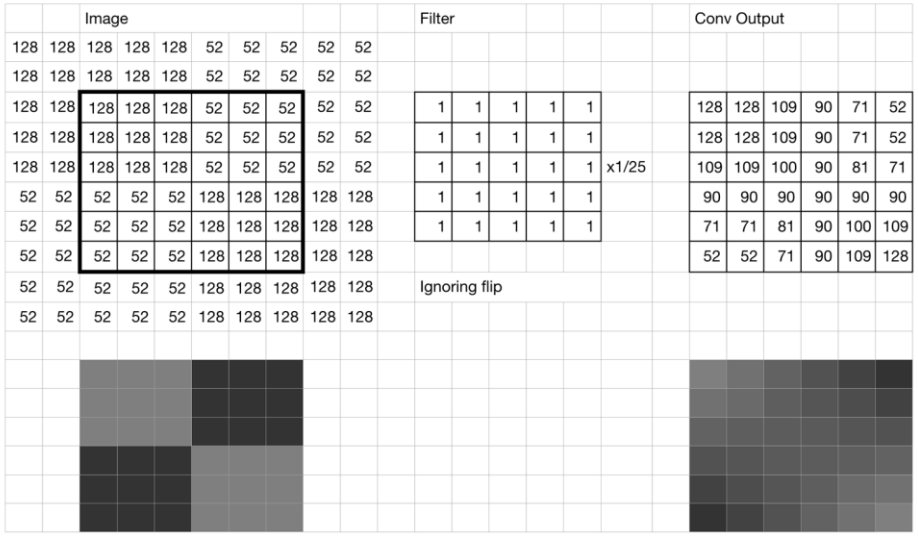
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

???

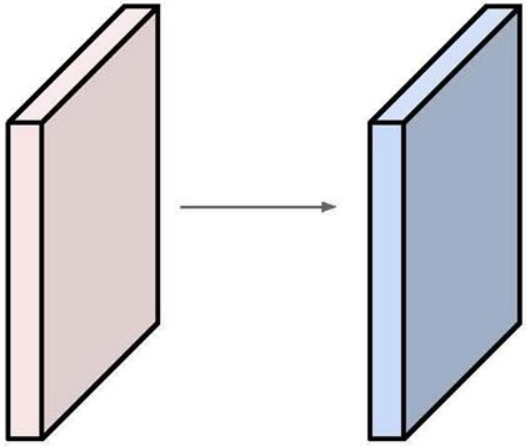
32x32x10

Mirror Padding 2 for Filter Size 5x5



ConvNet

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride **2**, pad 2

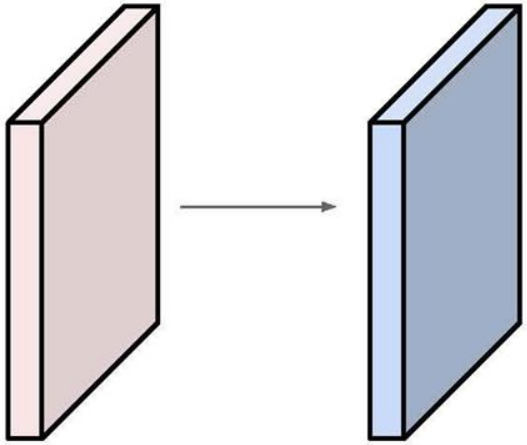
Output volume size: ?

???

16x16x10

ConvNet

Examples time:



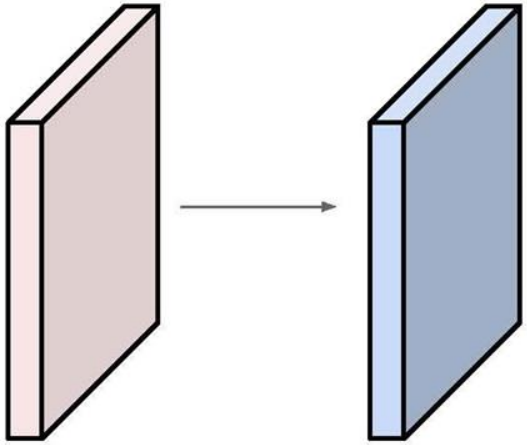
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

$$(W+2xP-F)/S + 1$$

ConvNet

Examples time:



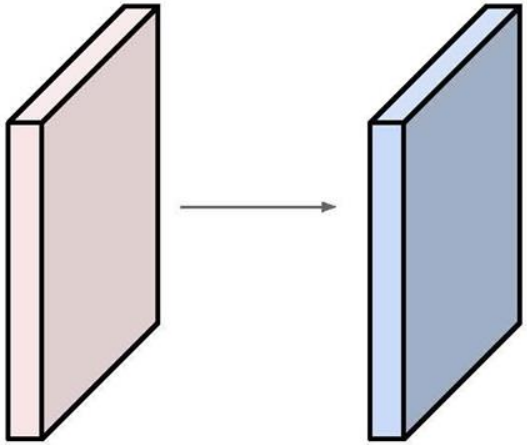
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

???

ConvNet

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

(FxFx3 +1)xK

Summary of Conv Layer

- Input: $W_1 \times H_1 \times D_1$
- Parameters
  - No. of Filters: K
  - Their spatial extent: F
  - Stride: S
  - Padding: P
- Output: $W_2 \times H_2 \times D_2$
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$
  - $D_2 = K$
- Parameters (with sharing)
  - Weights per filter $F \times F \times D_1$
  - For K filters and K biases $(F \times F \times D_1 + 1)K$

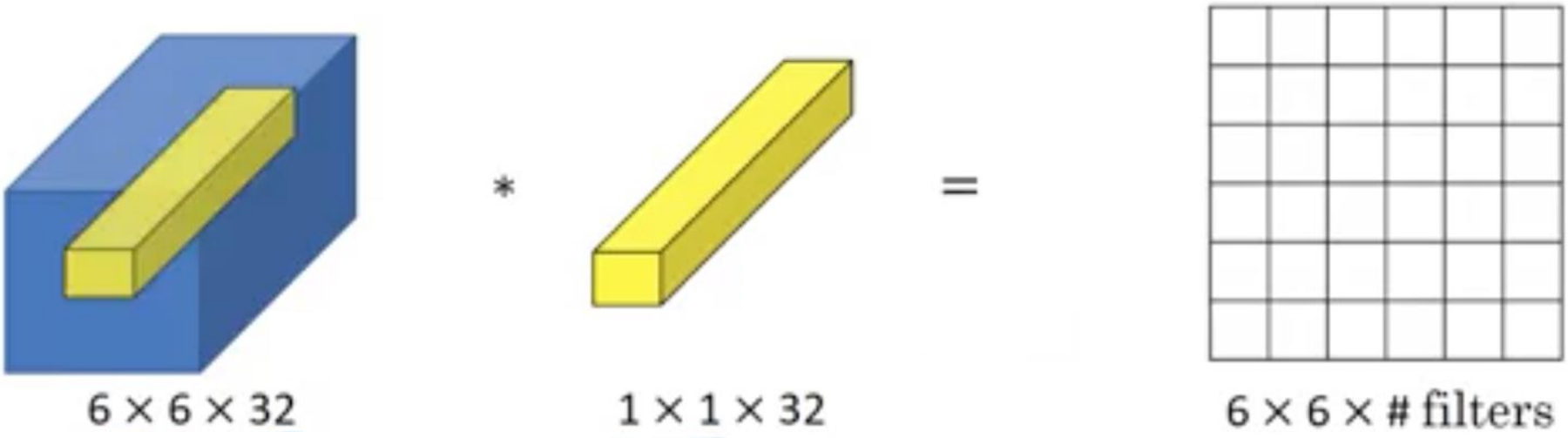
- Common Settings:
  - K =power of 2
  - e.g. 32, 64, 128, 512
  - $F = 3, S = 1, P = 1$
  - $F = 5, S = 1, P = 2$
  - $F = 5, S = 2, P =$? *(whatever fits)*
  - $F = 1, S = 1, P = 0$

1x1 Convolution

Table 1

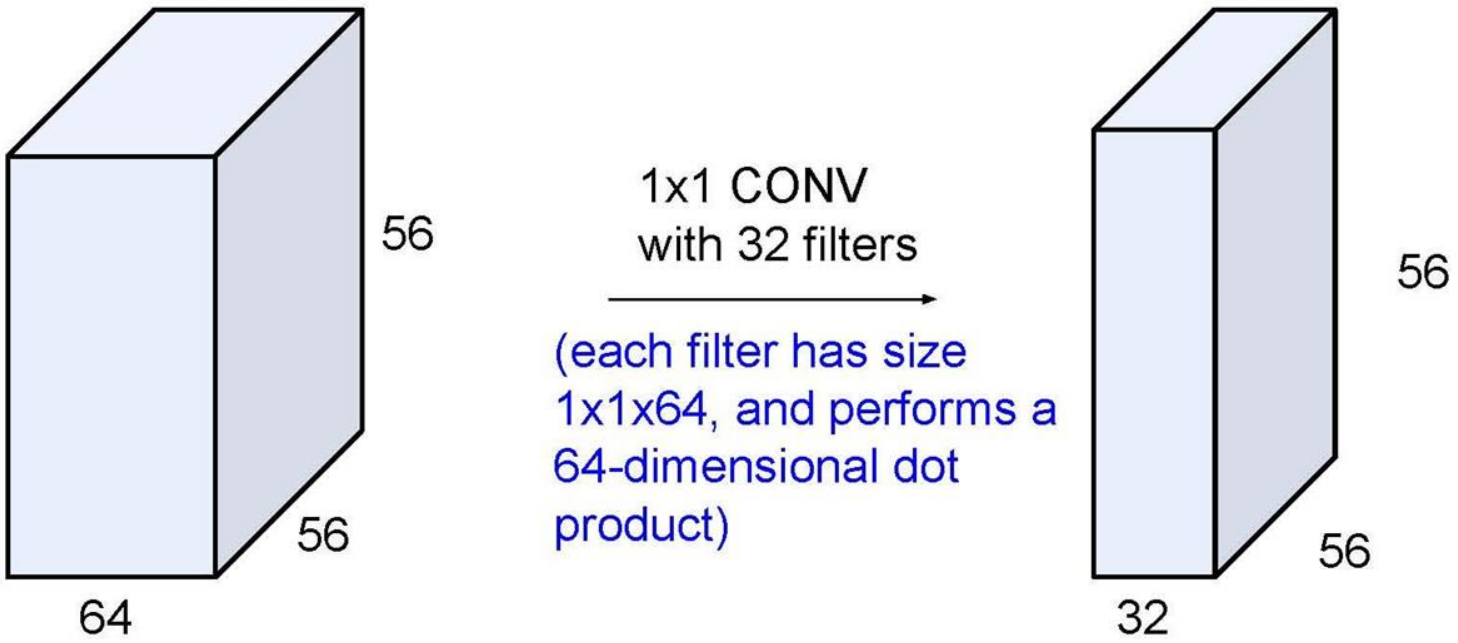
Image						Filter			Conv Output					
9	1	6	2	3	0		2		18	2	12	4	6	0
9	6	4	2	2	2				18	12	8	4	4	4
0	2	7	3	4	9				0	4	14	6	8	18
1	5	1	3	6	1				2	10	2	6	12	2
8	4	5	9	5	9				16	8	10	18	10	18
2	2	2	7	9	5				4	4	4	14	18	10

1x1 Convolution

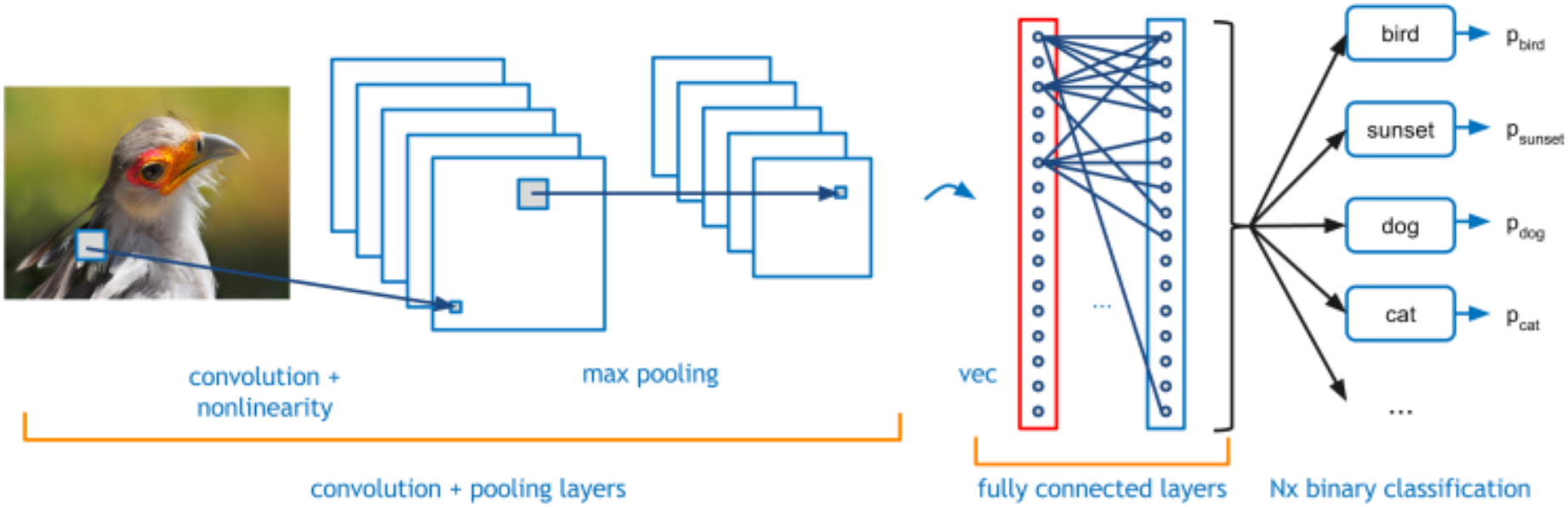


ConvNet

(btw, 1x1 convolution layers make perfect sense)

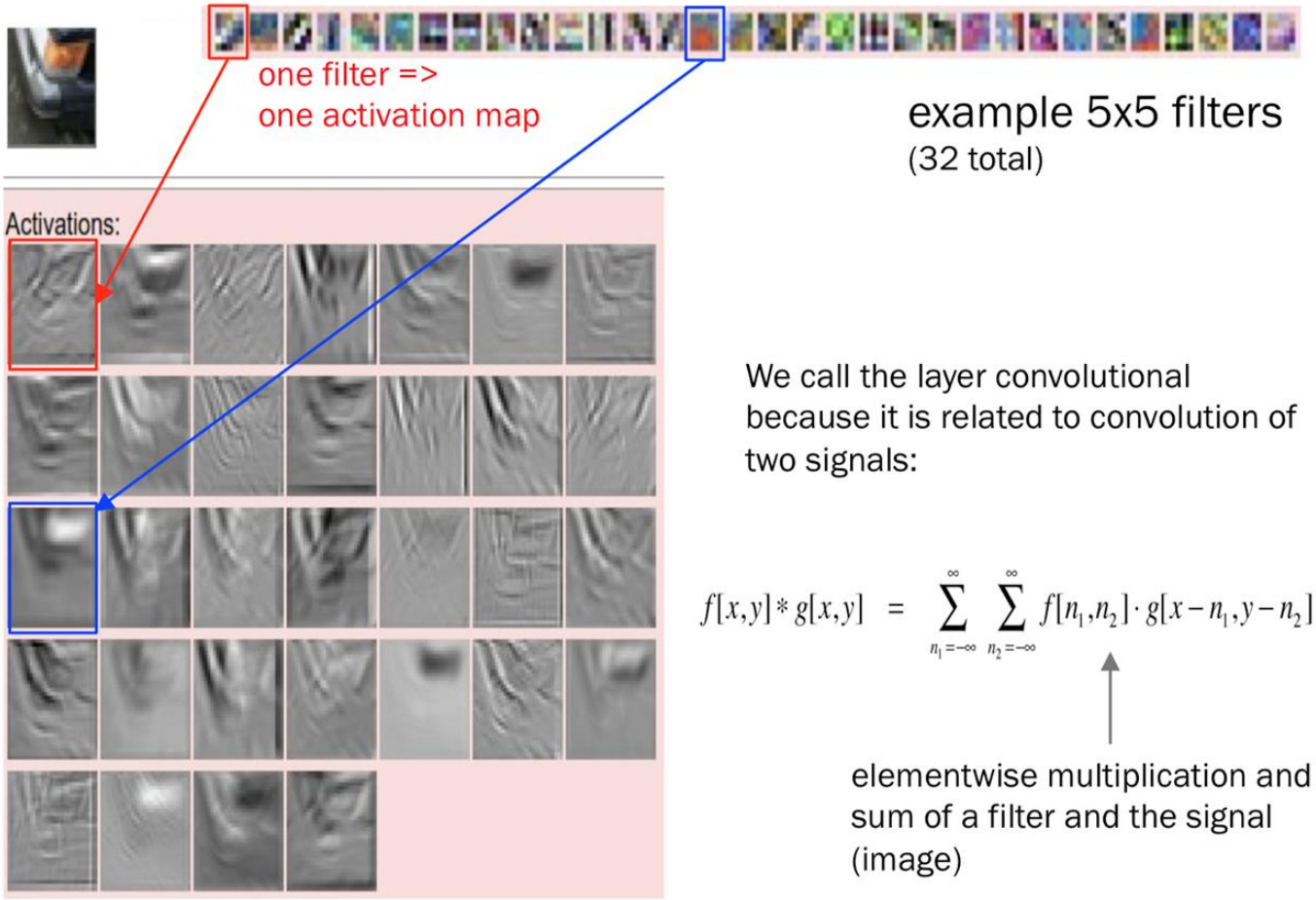


Method current used



http://code.flickr.net/2014/10/20/introducing-flickr-park-or-bird/

https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html



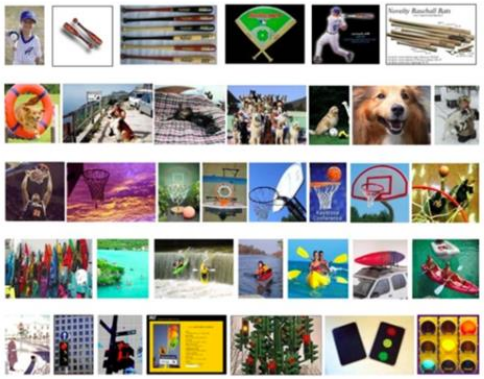
Any Questions?

Case Studies

- Classical Methods
  - LeNet-5
  - AlexNet
  - VGG-16
- ResNet-152
- Inception / GoogLeNet

Relentless research on visual recognition

Caltech 101



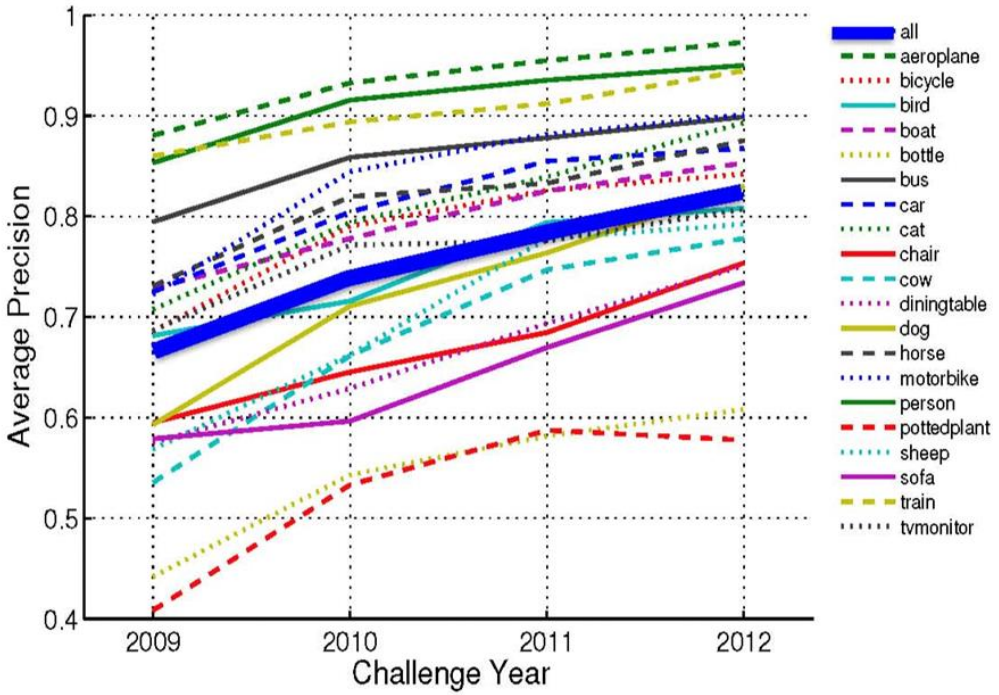
80 Million Tiny Images

PASCAL VOC

ImageNet

Pascal Visual Object Challenge

- 20 Object Categories



IMAGENET

www.image-net.org

22K categories and **14M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

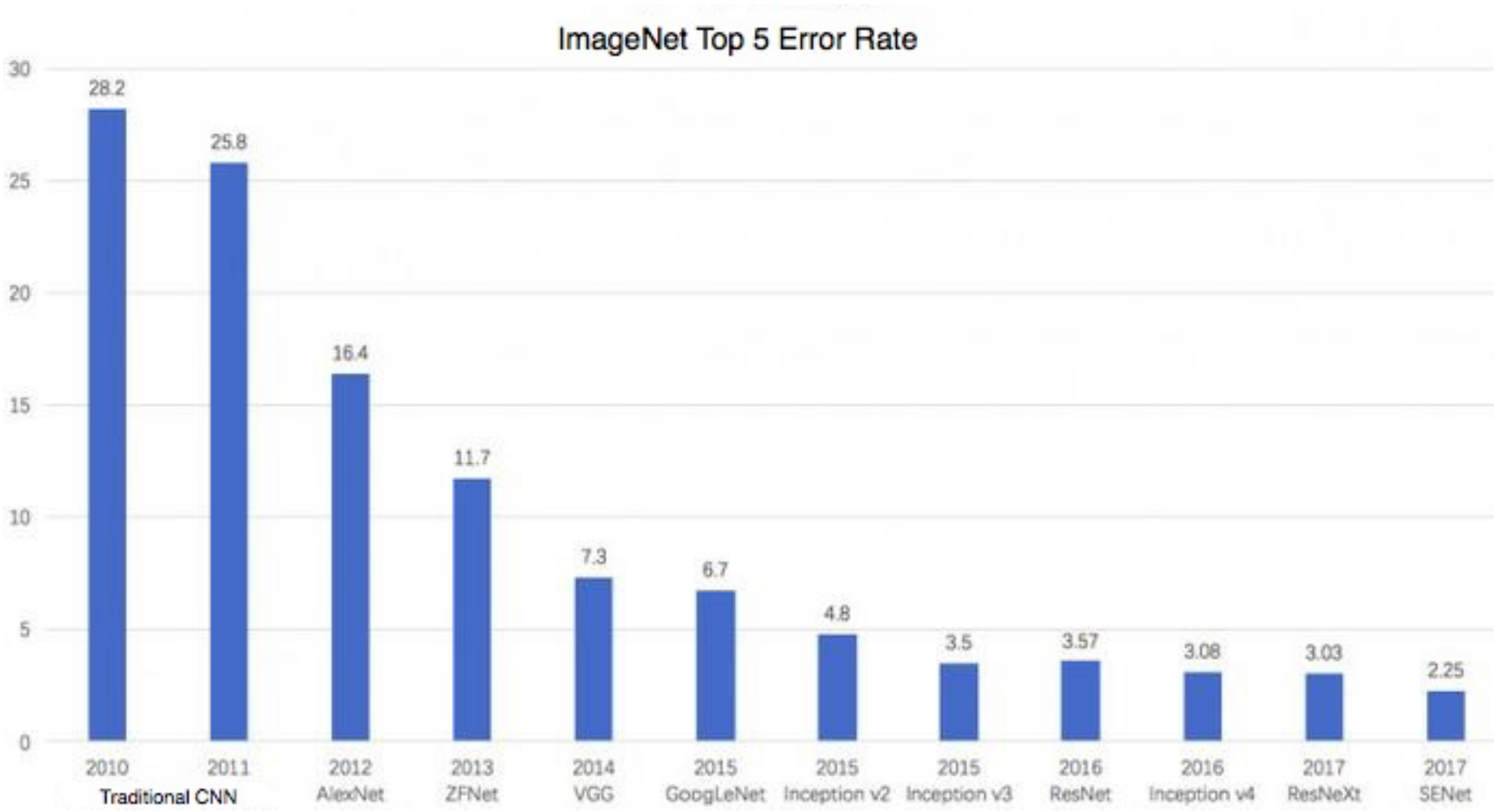
✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

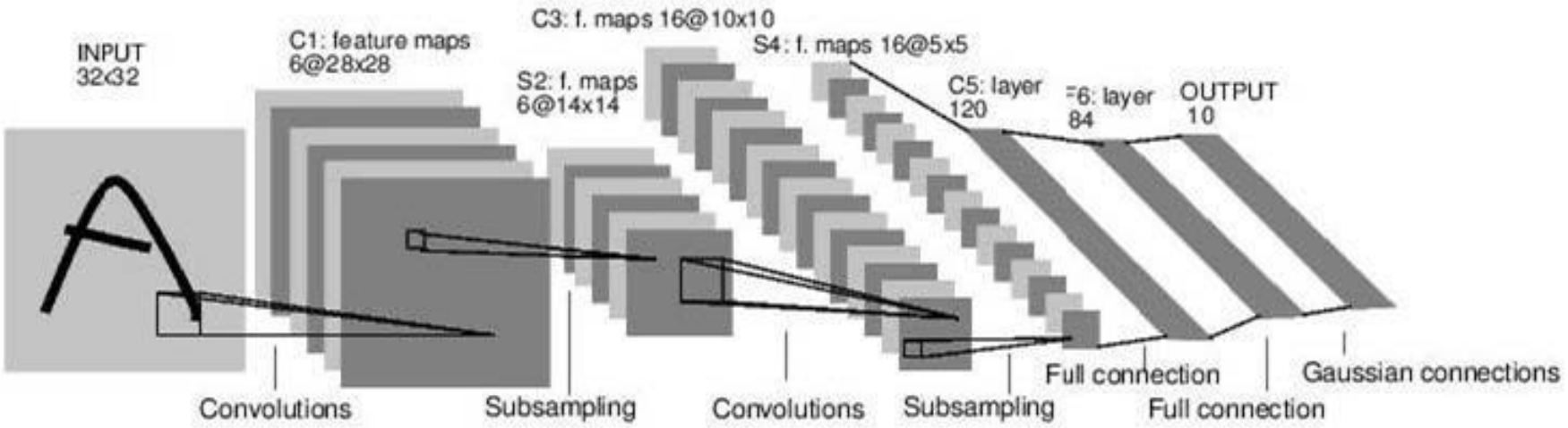
Russakovsky et al. arXiv, 2014

Case Studies



Case Study: LeNet-5

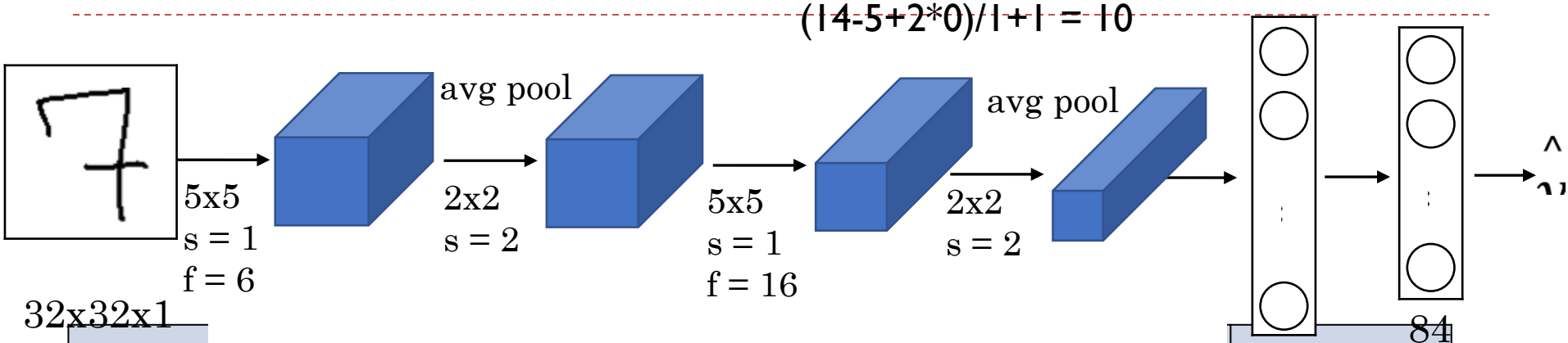
[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2

Case Study: LeNet-5

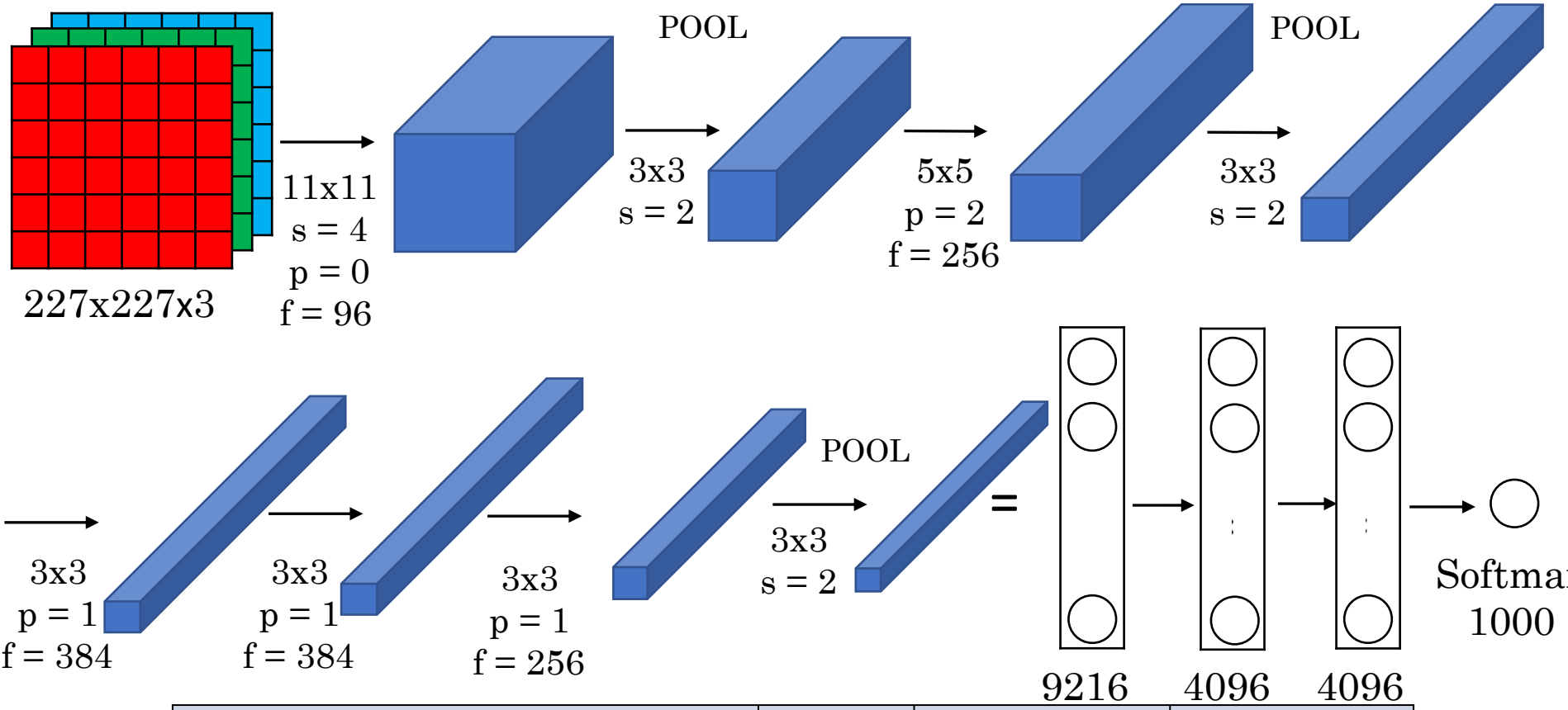
(W - F +2*P)/S + 1
(32 - 5 +2*0)/1 + 1 = 28
(14-5+2*0)/1+1 = 10



					Volume	(with bais)	Total Params (without bais)
Name	No. of filters	Filter size	Stride	Pad			
Input					32x32x1		
Conv1	6	5x5	1	0	28x28x6	(5x5x1+1)x6	150
Pool	N/A	2x2	2	0	14x14x6	0	0
Conv	16	5x5	1	0	10x10x16	(5x5x6+1)x16	2416
Pool	N/A	2x2	2	0	5x5x16	0	0
FC	N/A	N/A	N/A	N/A	120	5x5x16x120	48000
FC	N/A	N/A	N/A	N/A	84	120x84	10080

AlexNet

(W - F +2*P)/S + 1
(227 - 11 +2*0)/4 + 1 =



Layer					Output Volume	Total Params (with bais)	Total Params (without bais)
Name	No. of filters	Filter size	Stride	Pad			
Input							
Conv1	96	11x11	4	0	55x55x96	(11x11x3+1)*96	

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Case Study: VGGNet *[Simonyan & Zisserman, 2014]*

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Next: More on Case Studies

- ResNet-152
- Inception / GoogLeNet
- Yolo

CS 436 / CS 5310 / EE513
Computer Vision Fundamentals

Murtaza Taj

murtaza.taj@lums.edu.pk

Lecture 6: CNN Case Studies

Thurs 30th Sep 2021

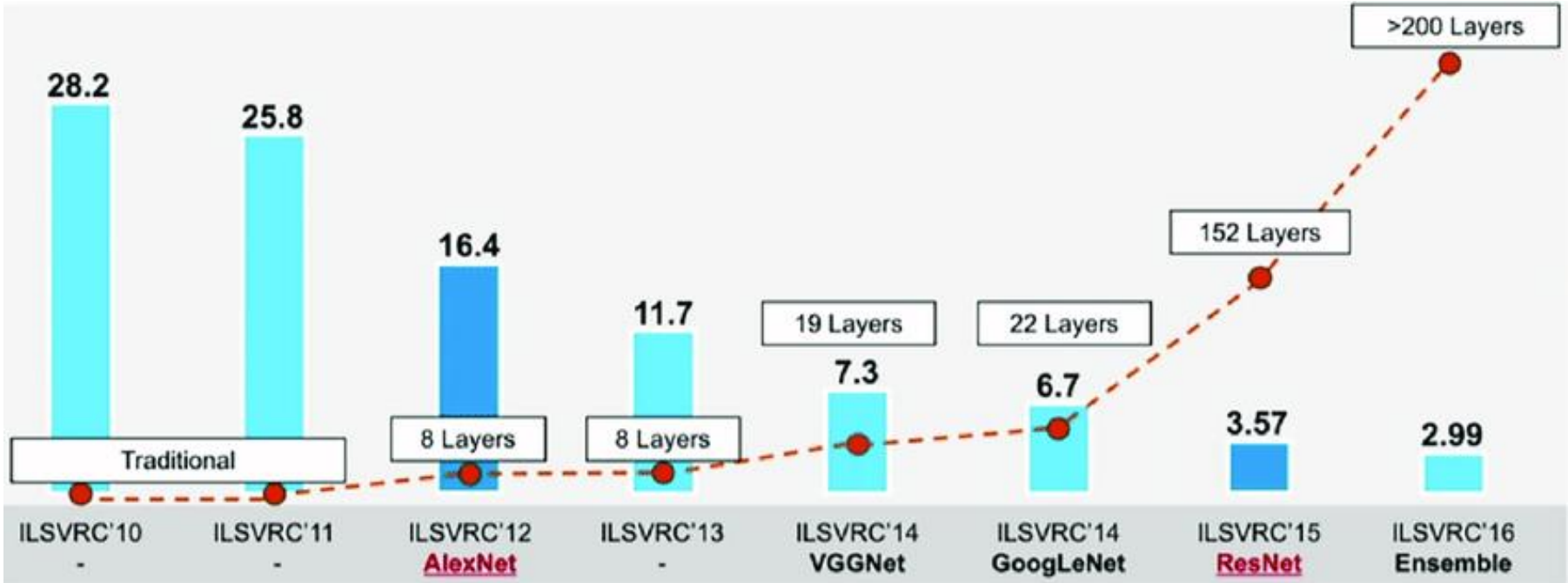
Questions about CNN

- How many layers?
- How many filters for each layer?
- What should be the size of filters
- Shall we use stride or pooling?
- Do we need fully connected layers or not?
- ….

- Answer: Case studies

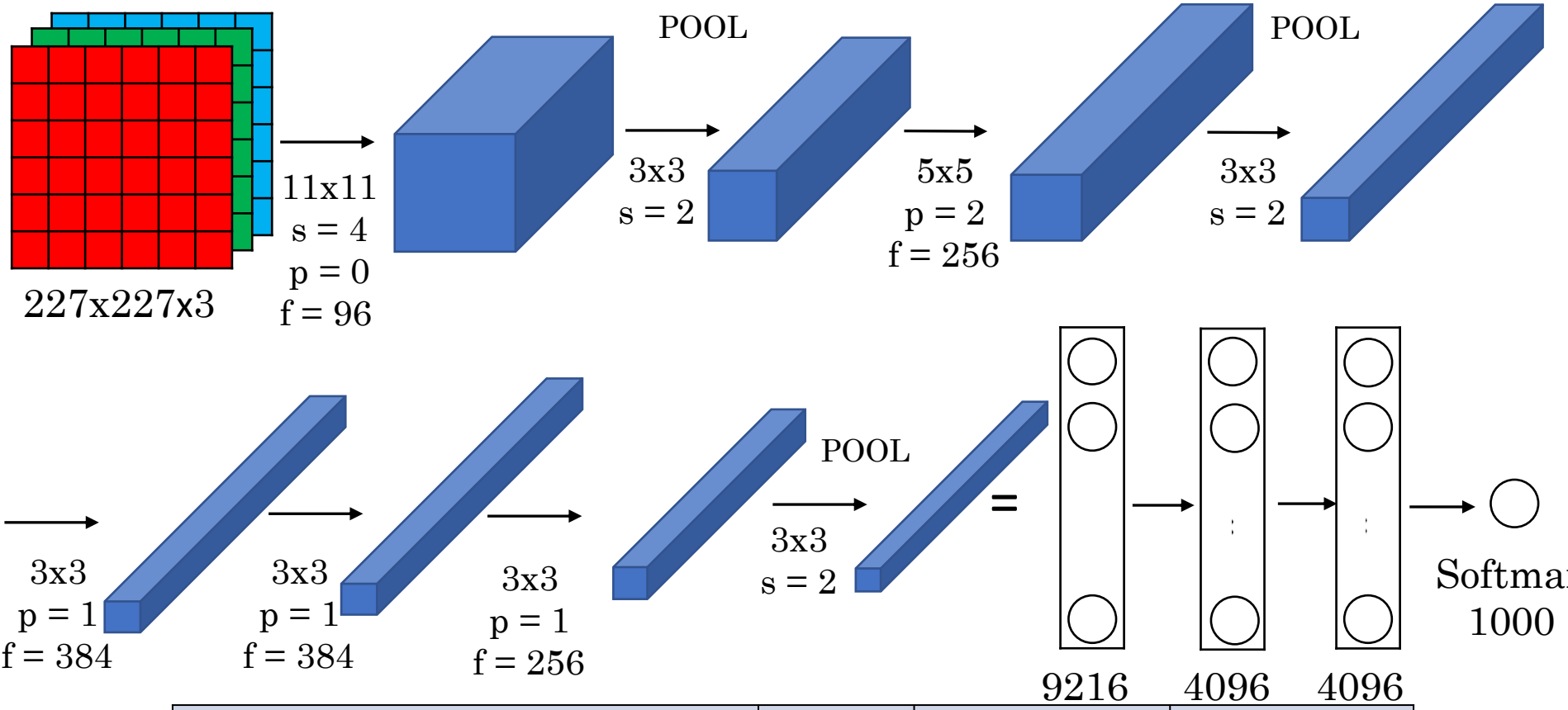
Case Studies

- ImageNet Challenge - 1000 class classification
- Classical Methods
  - LeNet-5 - 1998 (MNIST Digit classification)
  - AlexNet - 2012
  - VGG-16 - 2014
  - ResNet-152



AlexNet

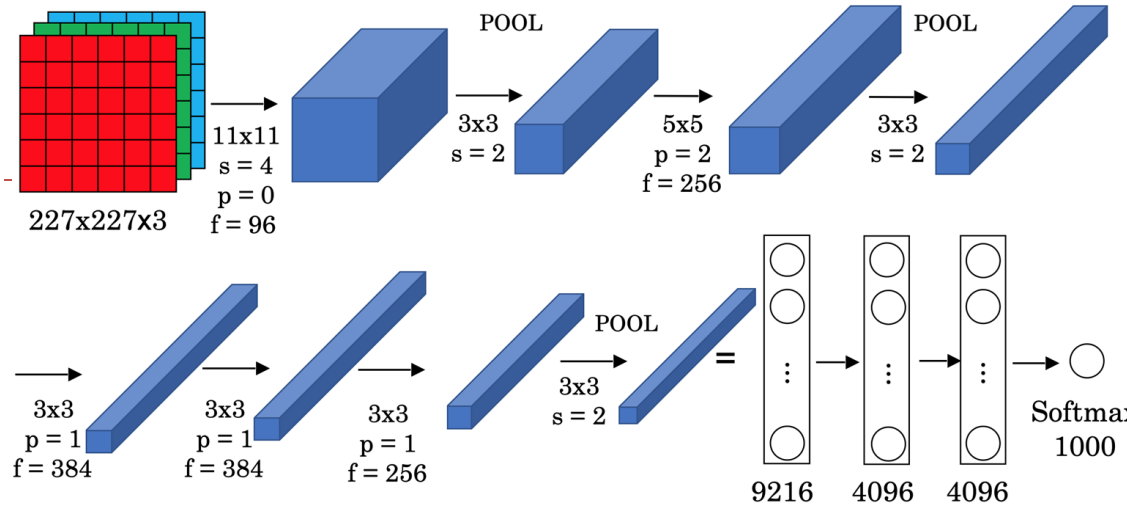
(W - F +2*P)/S + 1
(227 - 11 +2*0)/4 + 1 =



Layer					Output Volume	Total Params (with bais)	Total Params (without bais)
Name	No. of filters	Filter size	Stride	Pad			
Input							
Conv1	96	11x11	4	0	55x55x96	(11x11x3+1)*96	

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

AlexNet



AlexNet Network - Structural Details													
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
						Total							**62,378,344**

58.62mn parameters in FC as compared to total 62.38mn total parameters

Case Study: VGGNet *[Simonyan & Zisserman, 2014]*

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

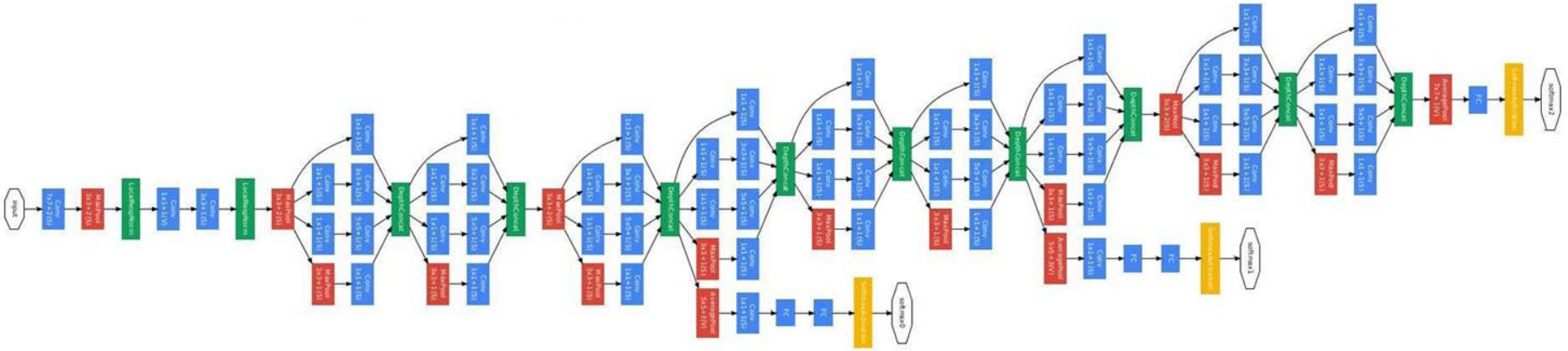
11.2% top 5 error in ILSVRC 2013
->
7.3% top 5 error

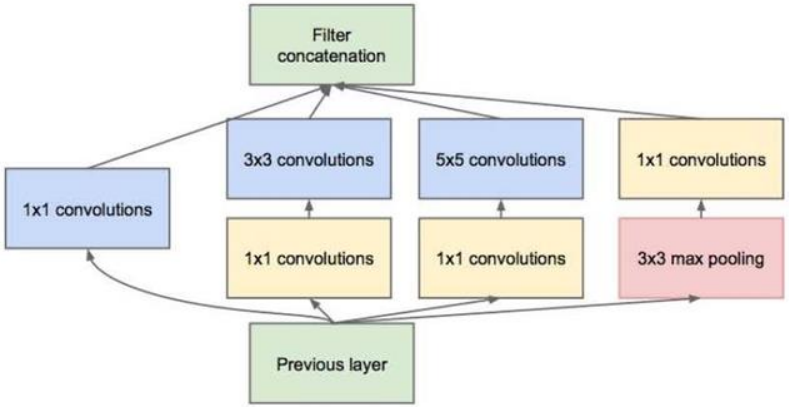
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Case Study: GoogLeNet *[Szegedy et al. 2014]*

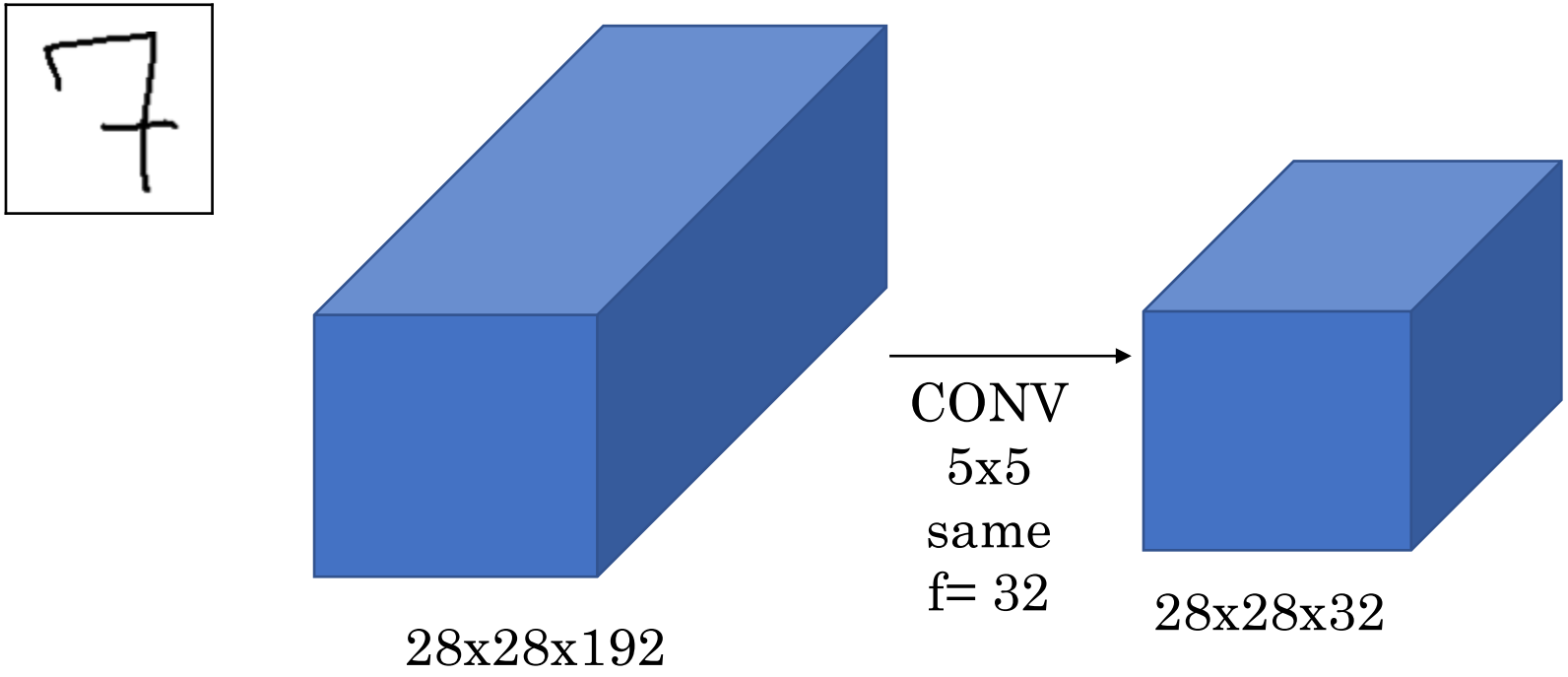




Inception module

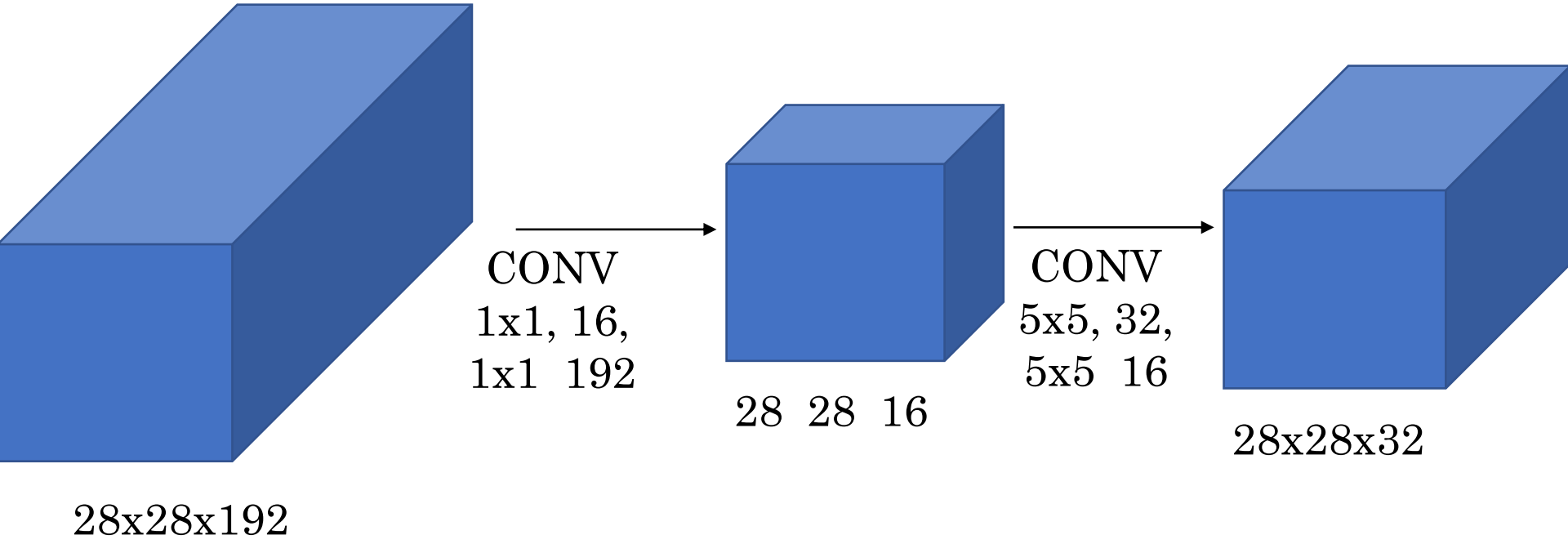
ILSVRC 2014 winner (6.7% top 5 error)

Case Study: GoogLeNet *[Szegedy et al. 2014]*



Multiplications: 5 x 5 x192 x 28 x 28 x 32 = 120m

Case Study: GoogLeNet *[Szegedy et al. 2014]*

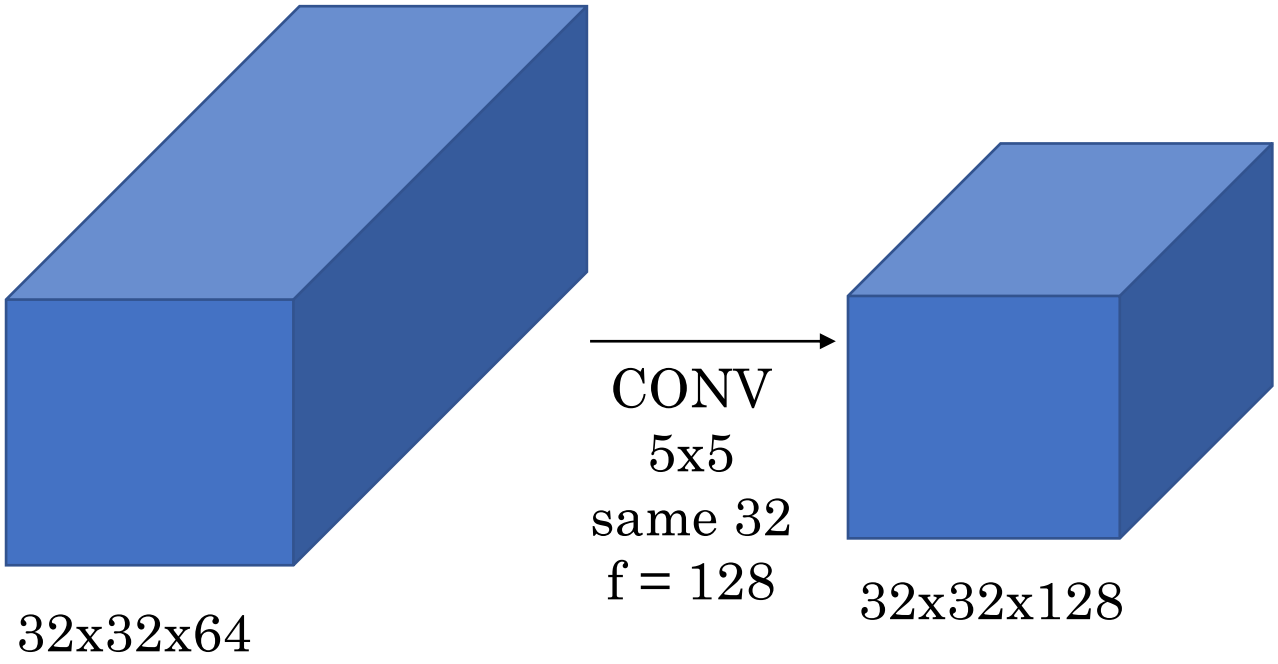


Multiplications: 1 x 1 x192 x 28 x 28 x 16= 2.4m

Multiplications: 5 x 5 x 16 x 28 x 28 x 32= 10.0m

12.4m vs. 120m

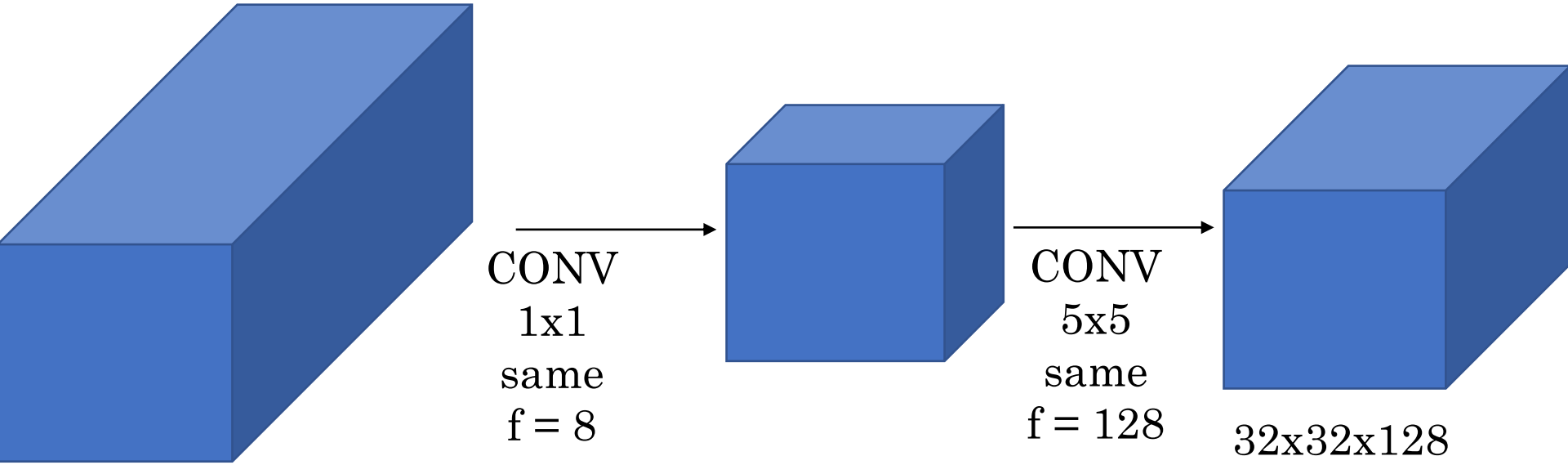
Case Study: GoogLeNet *[Szegedy et al. 2014]*



Multiplications: ???

Multiplications: 5 x 5 x64 x 32 x 32 x 128 = 209,715,200

Case Study: GoogLeNet *[Szegedy et al. 2014]*



Multiplications: ???

Multiplications: 1 x 1 x64 x 32 x 32 x 8= 5,24,288

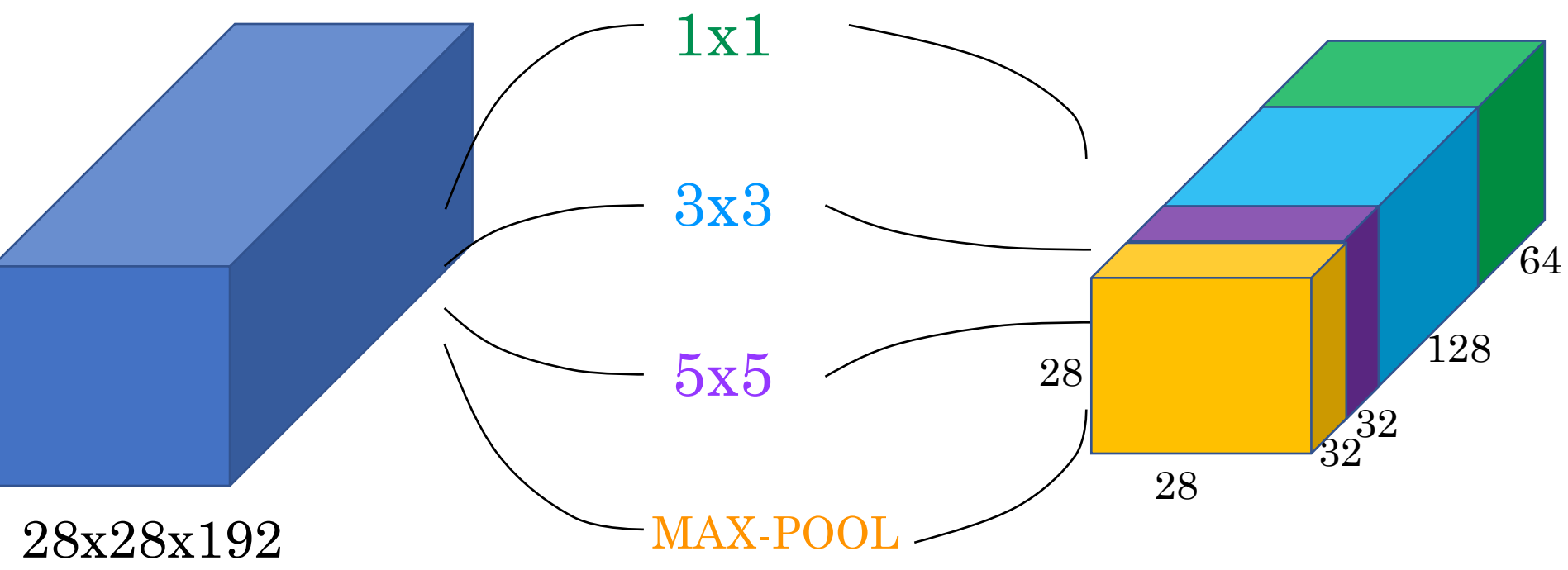
Multiplications: Poll

Multiplications: 5 x 5 x 8 x 32 x 32 x 128= 26,214,400

26.7m vs. 209.7m

Case Study: GoogLeNet *[Szegedy et al. 2014]*

- Inception Motivation



Multiplications: 1 x 1 x 192 x 28 x 28 x 64 = 9,633,792

Multiplications: 3 x 3 x 192 x 28 x 28 x 128 = 173,408,256

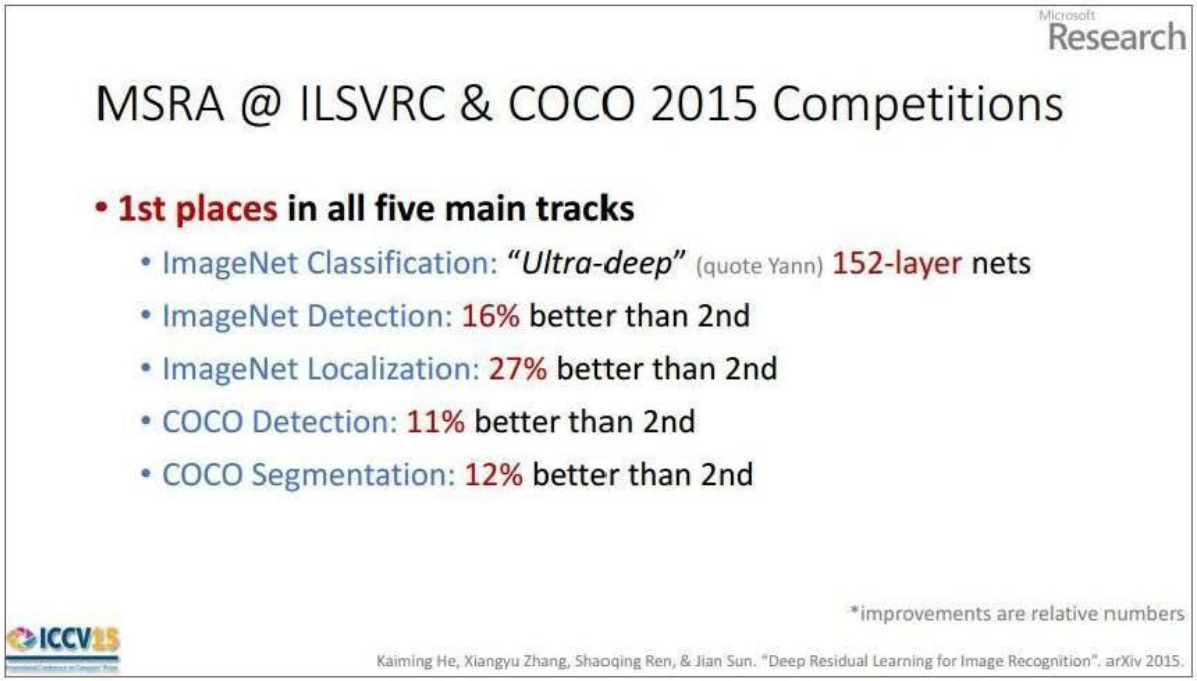
Multiplications: 5 x 5 x 192 x 28 x 28 x 32 = 120,422,400

303.5m vs. 963.4m

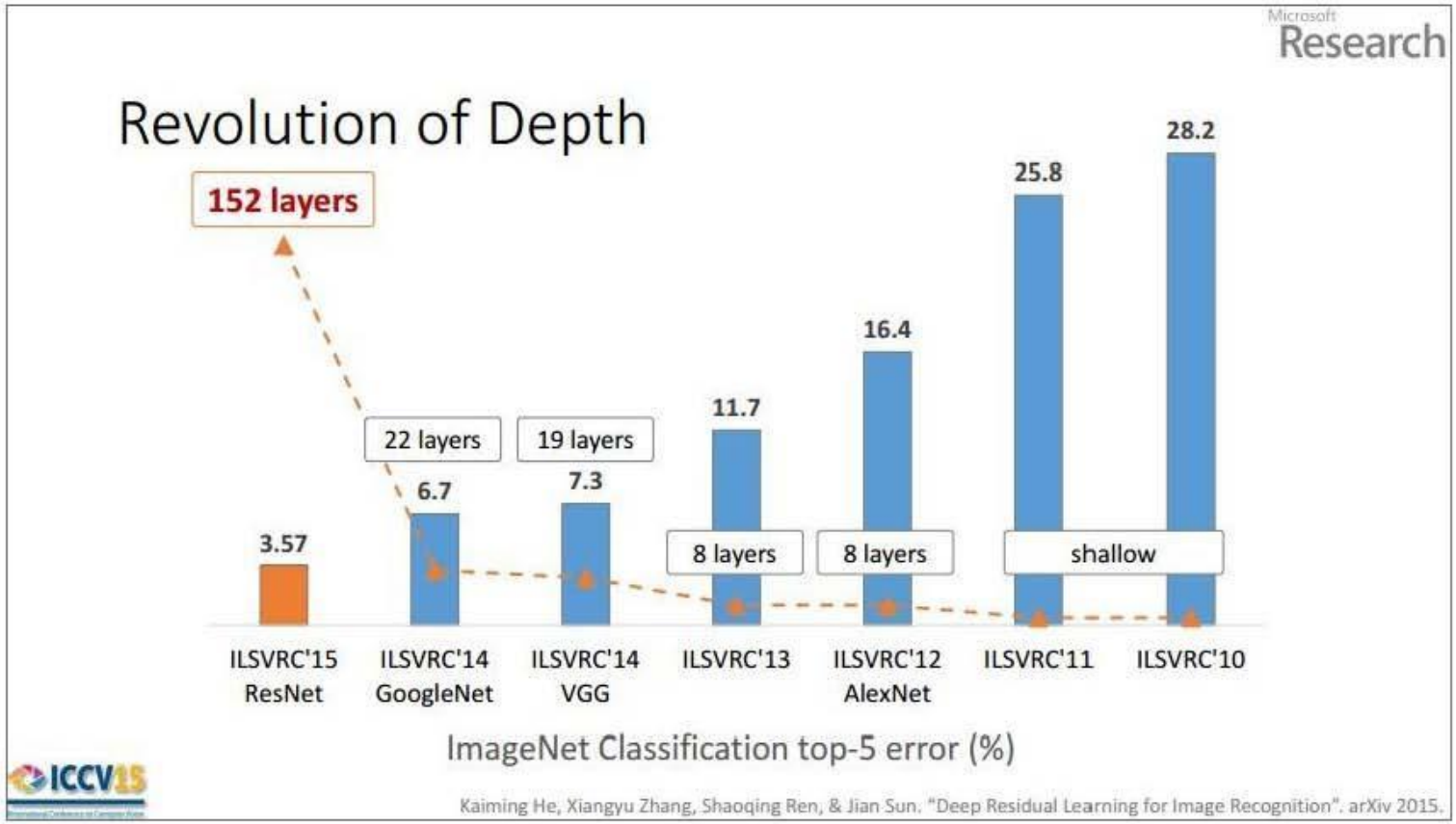
[Szegedy et al. 2014. Going deeper with convolutions]

Case Study: ResNet *[He et al. 2015]*

ILSVRC 2015 winner (3.6% top 5 error)

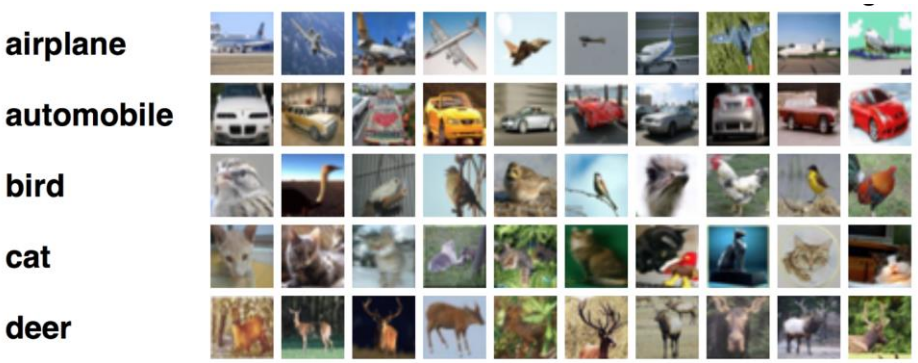


Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

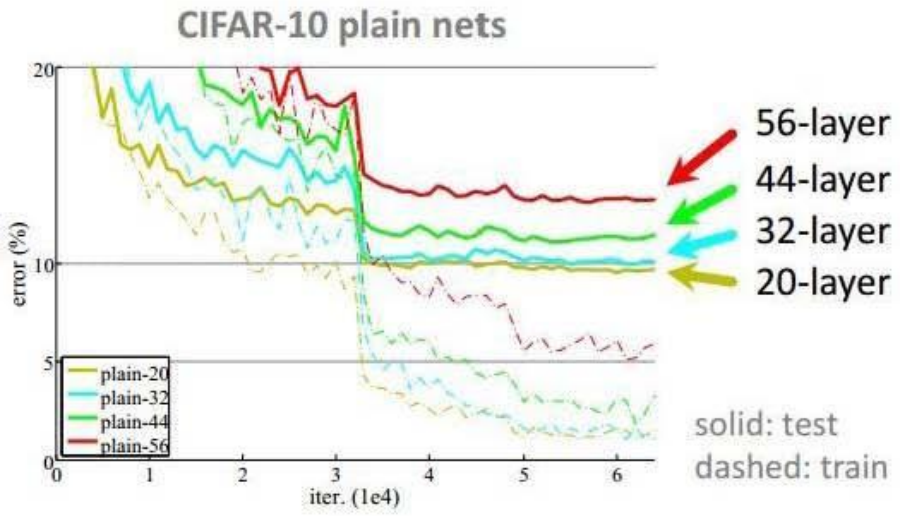


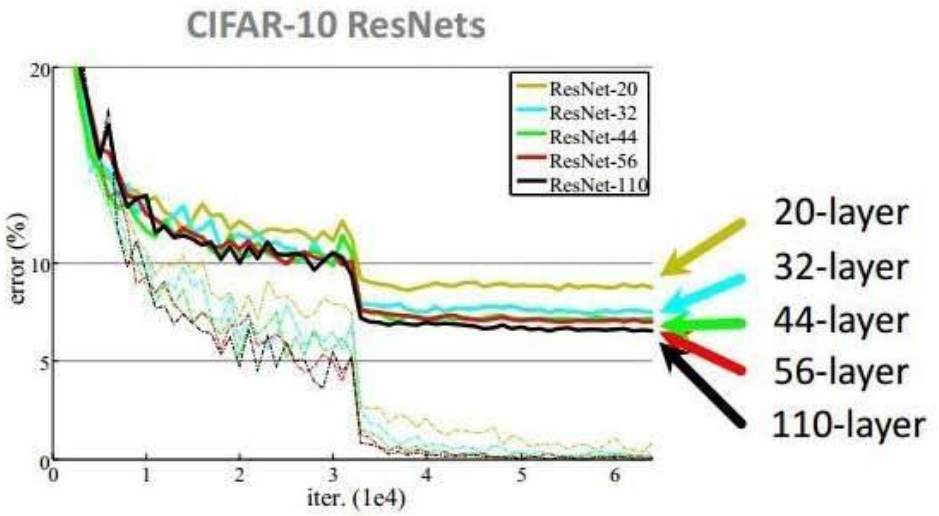
(slide from Kaiming He's recent presentation)

Case Study: ResNet *[He et al. 2015]*



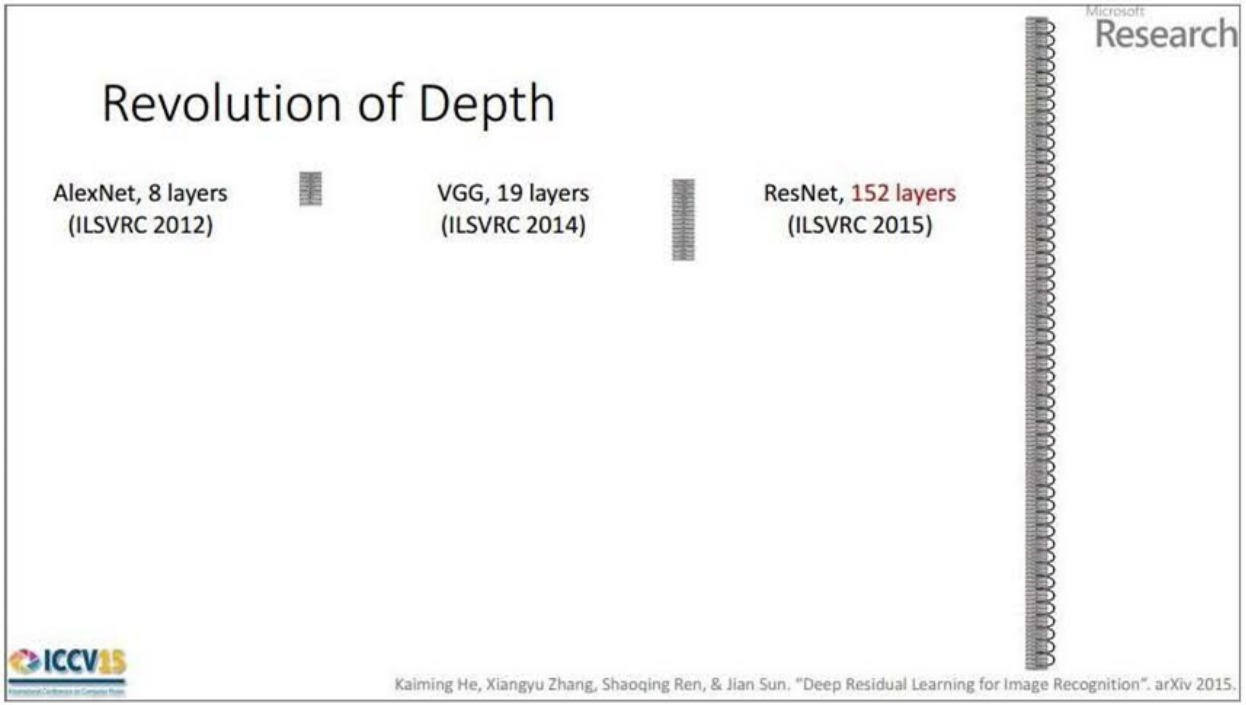
CIFAR-10 experiments





Case Study: ResNet *[He et al. 2015]*

ILSVRC 2015 winner (3.6% top 5 error)

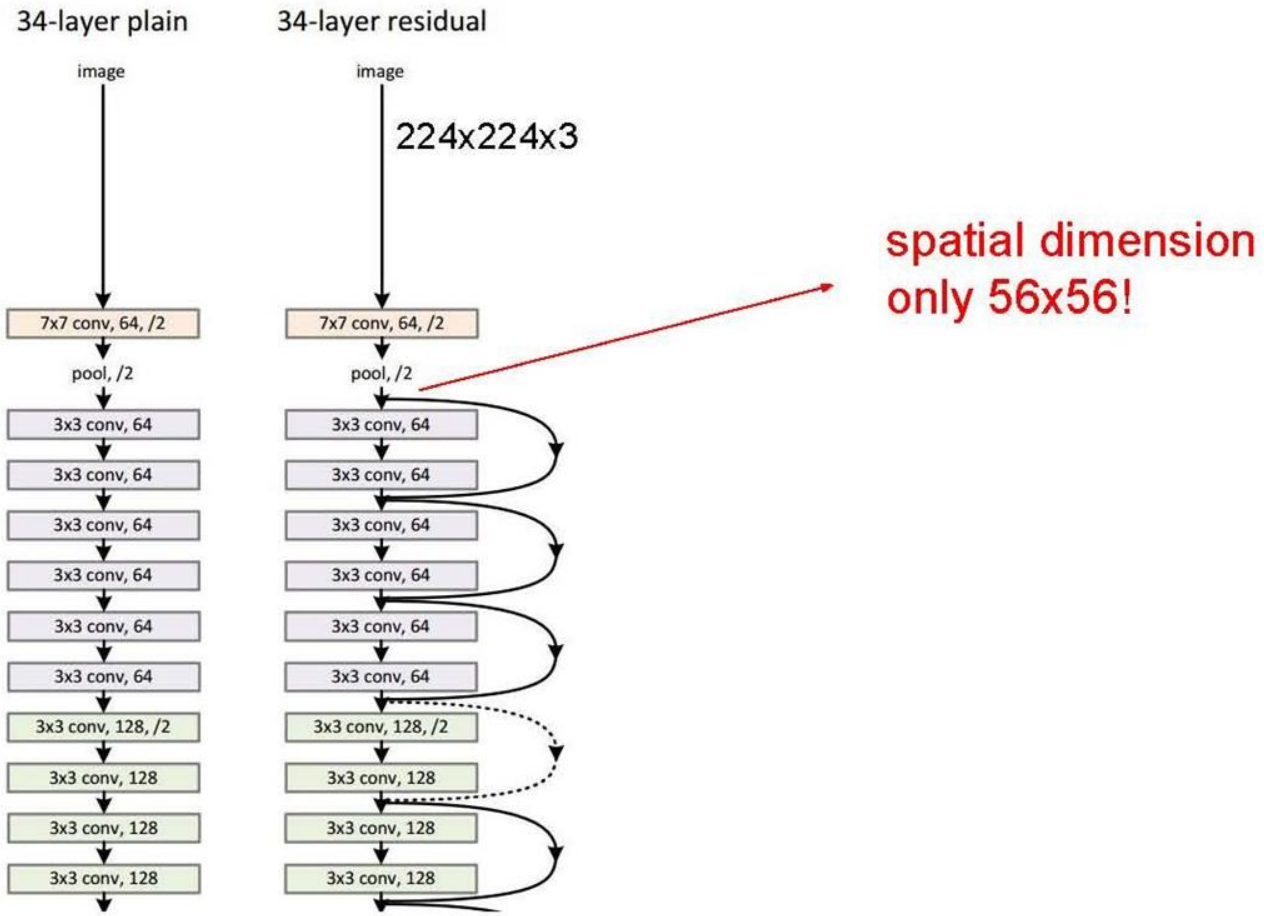


2-3 weeks of training on 8 GPU machine

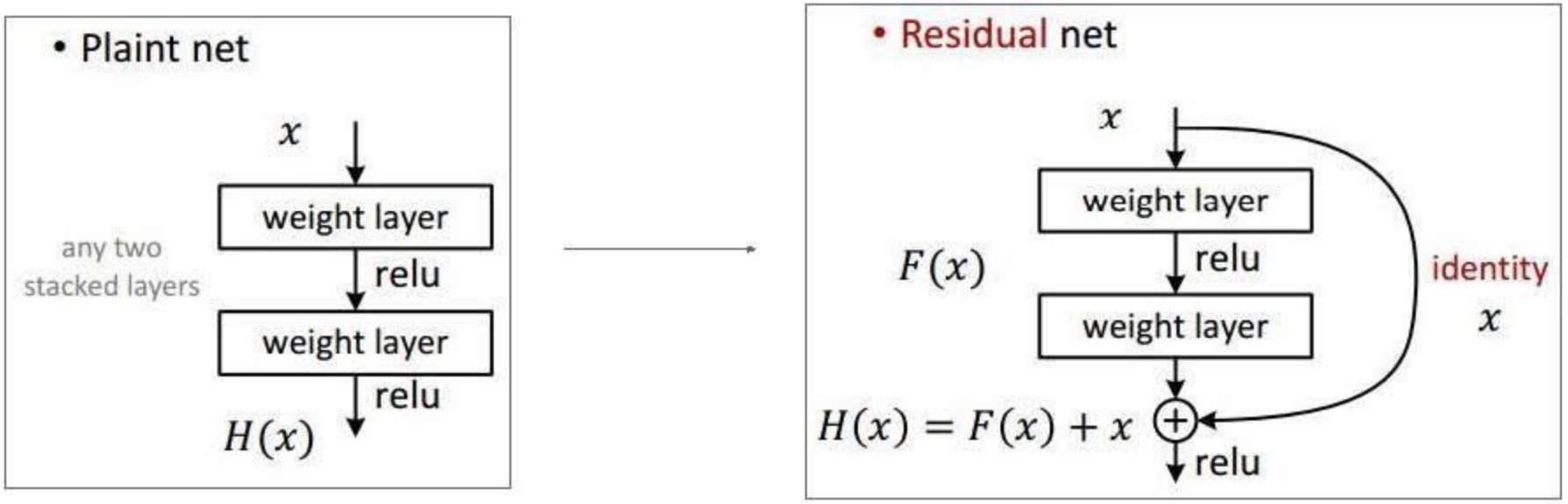
at runtime: faster than a VGGNet! (even though it has 8x more layers)

(slide from Kaiming He's recent presentation)

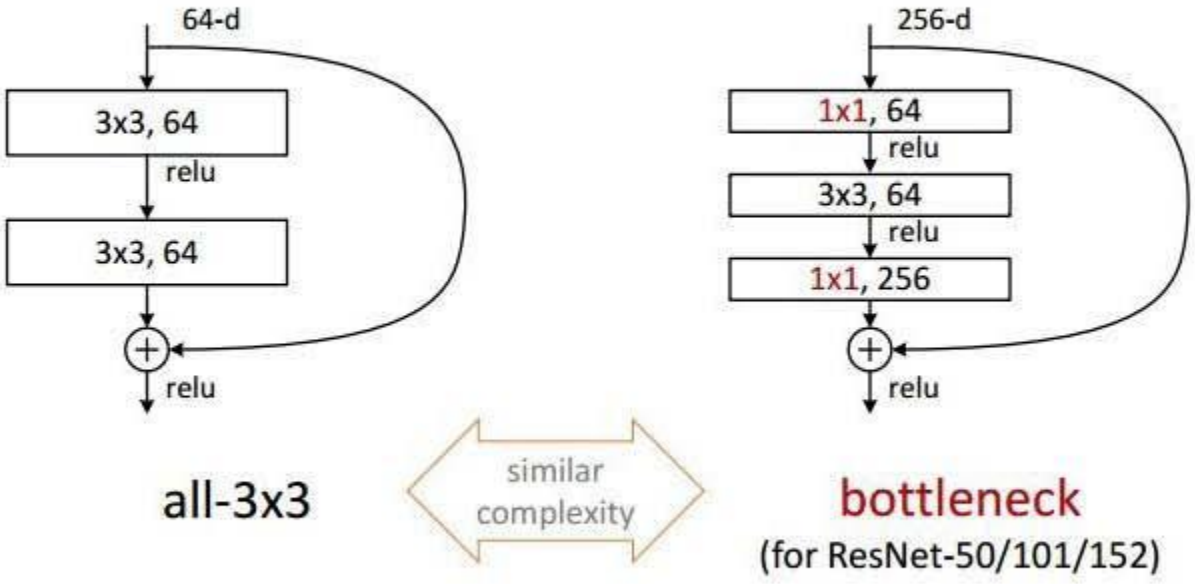
Case Study: ResNet *[He et al. 2015]*



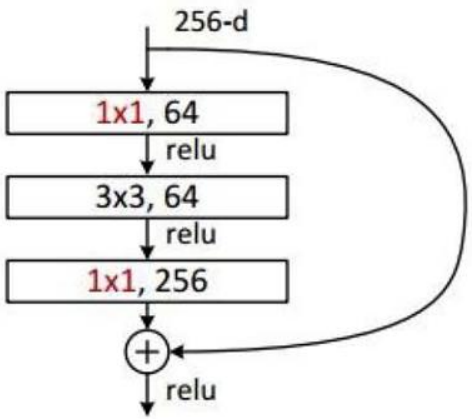
Case Study: ResNet *[He et al. 2015]*

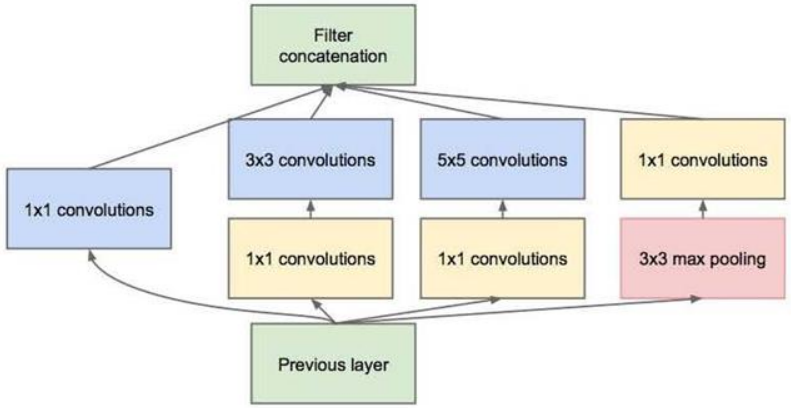


Case Study: ResNet *[He et al. 2015]*



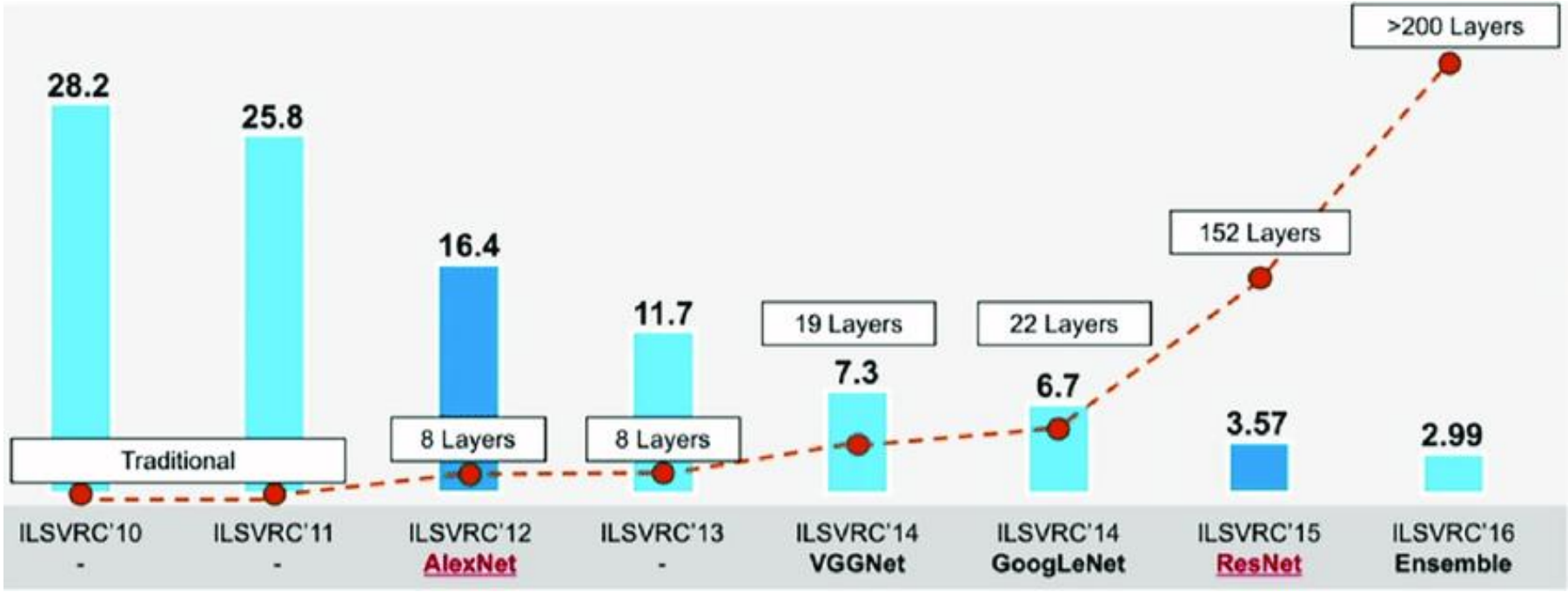
Case Study: ResNet *[He et al. 2015]*





(this trick is also used in GoogLeNet)

CNN Top 5 Error vs. Number of Layers

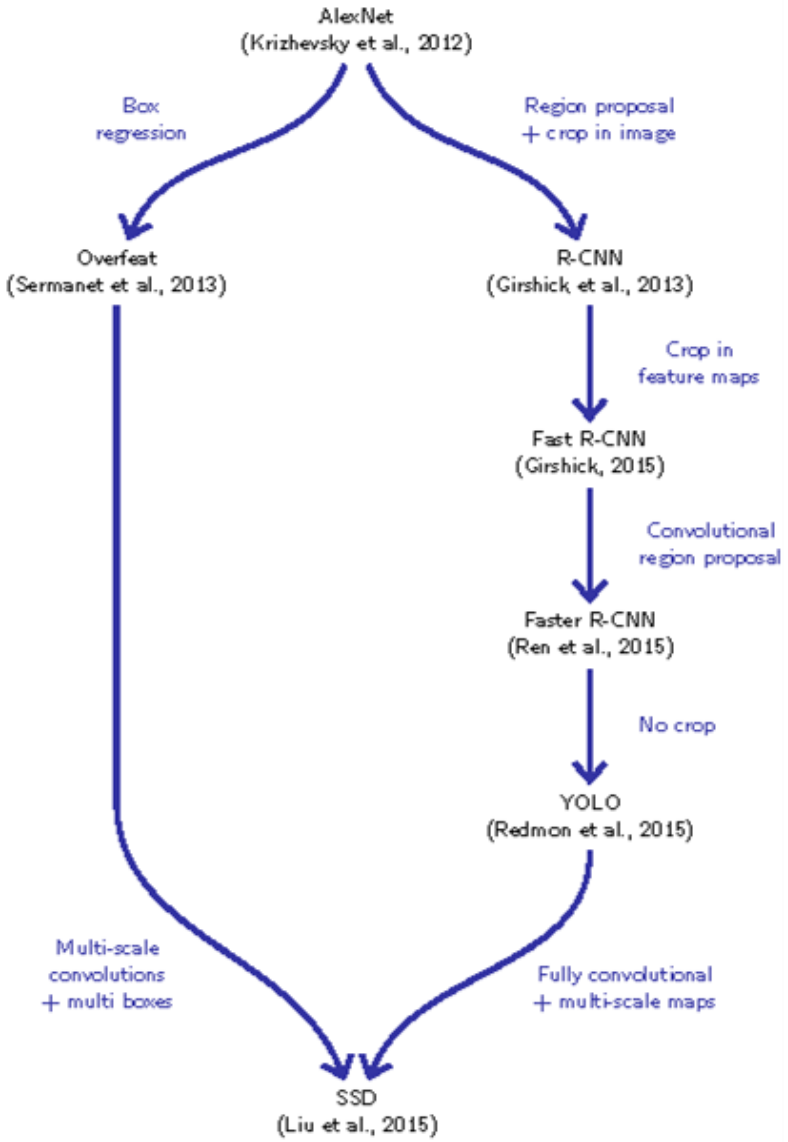


https://www.researchgate.net/figure/Recent-ConvNets-proposed-in-ILSVRC_fig1_338797371

Object Detection

Survey of Object Detection

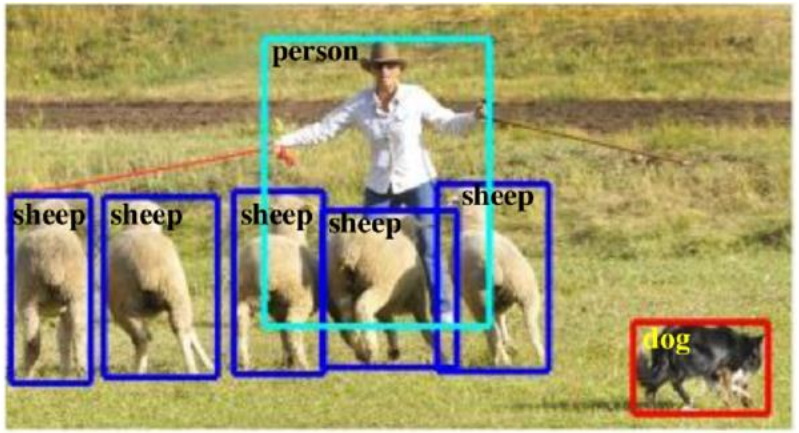
- Deep Learning for Generic Object Detection: A Survey Li Liu et al. 2018
  - Region Proposals
  - Region CNN
  - Fast RCNN
  - Faster RCNN
  - Yolo
  - SSD
- 3D Object Detection



From Classification to Object Detection

- 20 class classification
- 20 + 4 i.e. we will detect only 1 bbox per image

(a) Object Classification



(b) Generic Object Detection (Bounding Box)

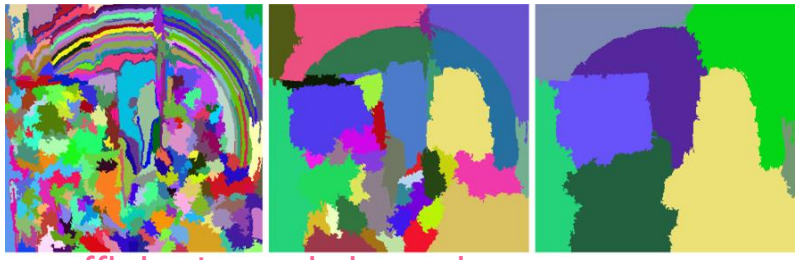
Class

Class
(x,y,w,h)

Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

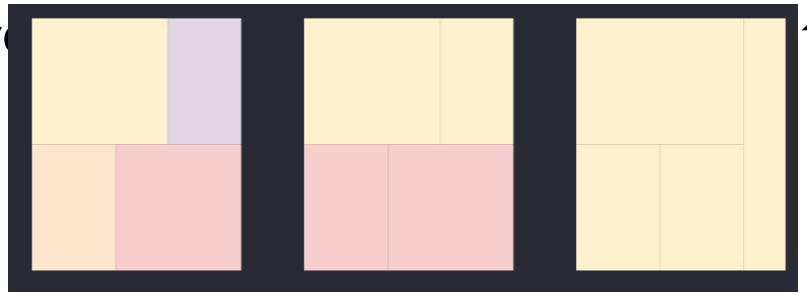
Region Proposals: Selective Search

- 1. Generate Initial Sub-segmentation

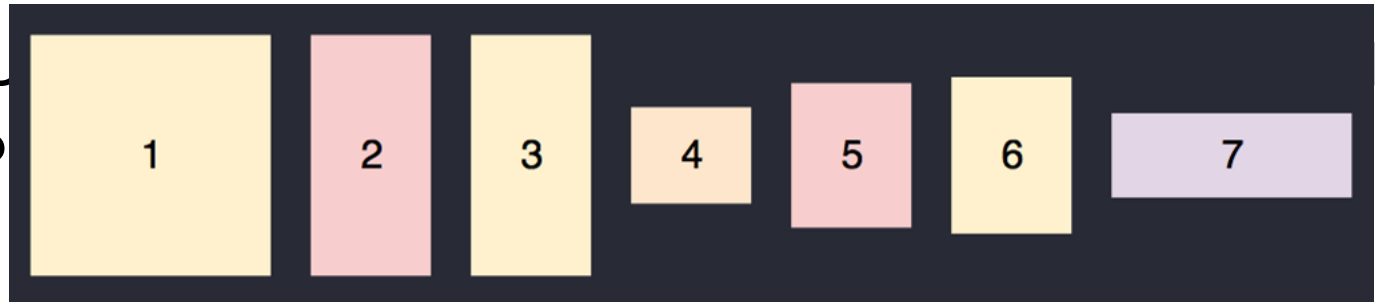


efficient graph-based segmentation

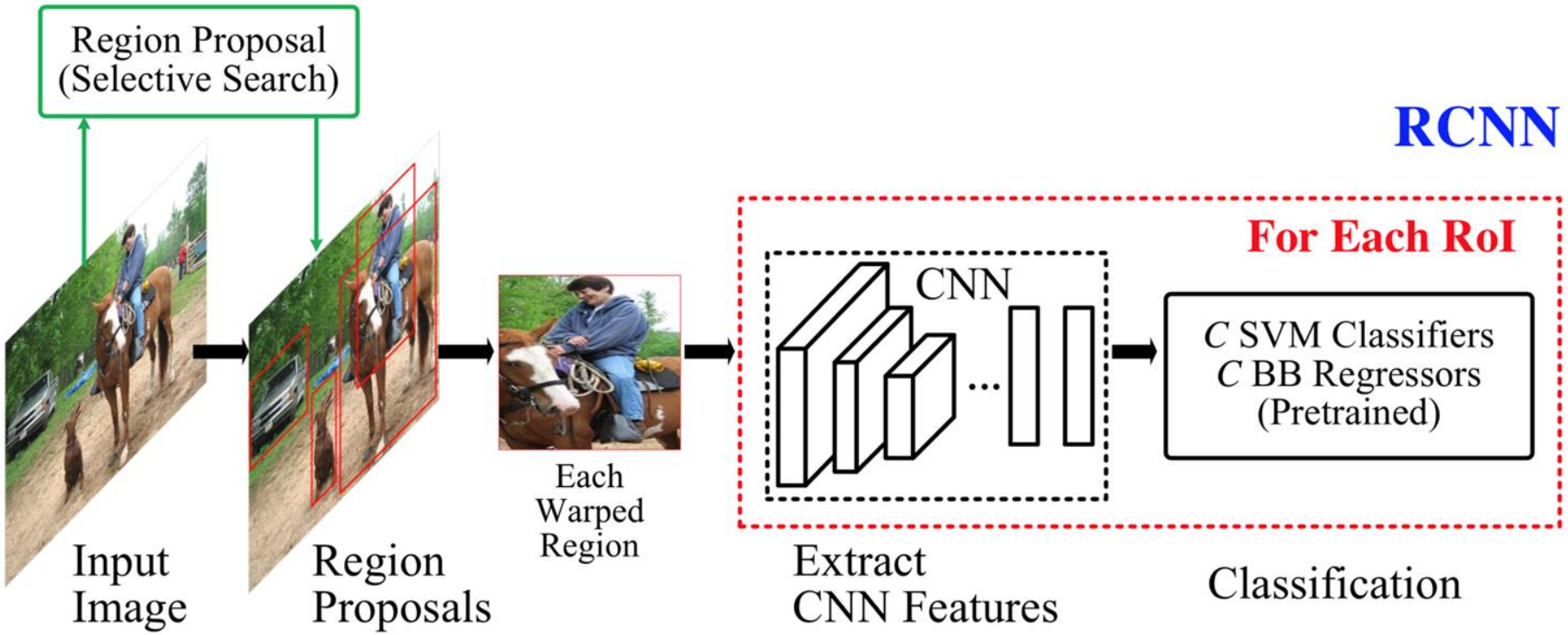
- 2. Combine similar r texture, size)



- 3. U ate region pro



Region CNN



Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

Region CNN

Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Region CNN

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

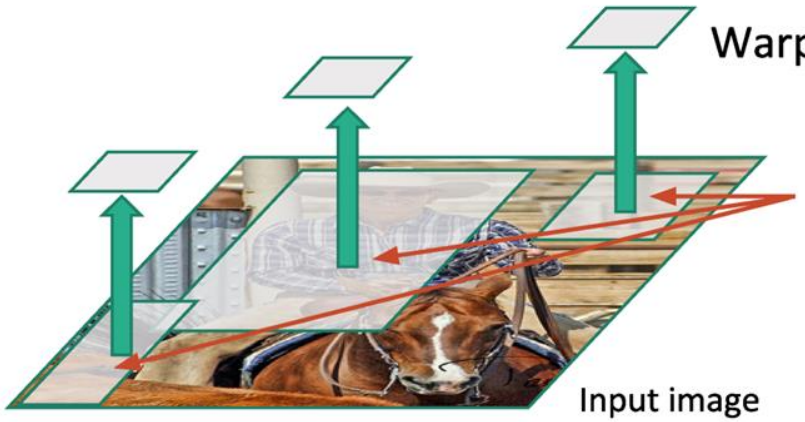
Region CNN

Input image

Regions of Interest (RoI) from a proposal method (~2k)

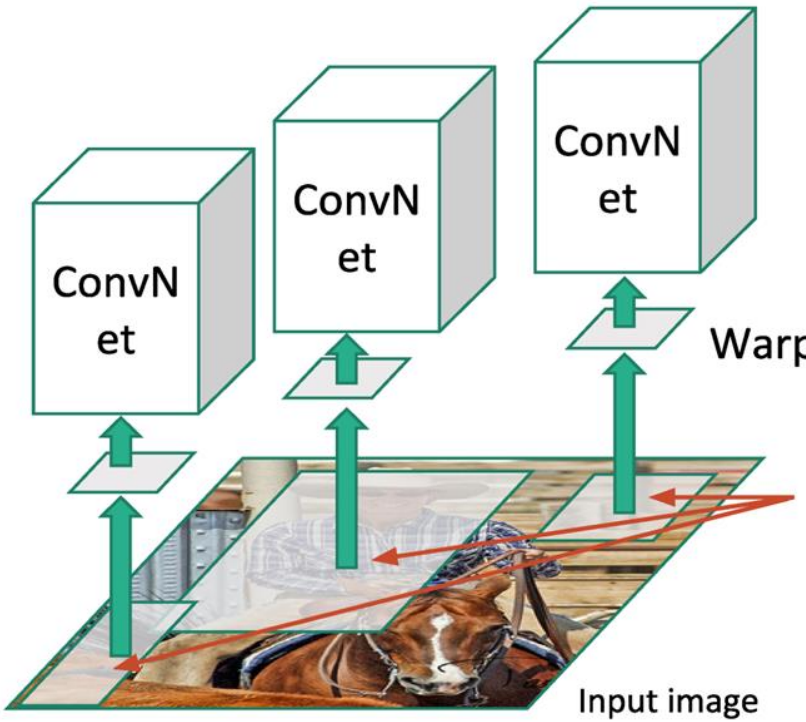
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Region CNN



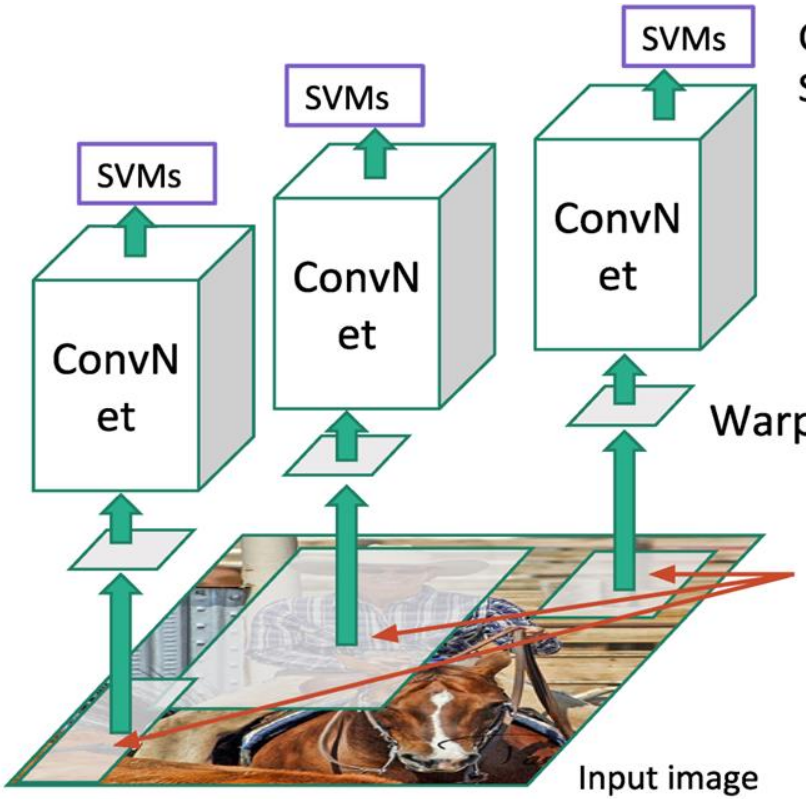
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Region CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Region CNN



Classify regions with SVMs

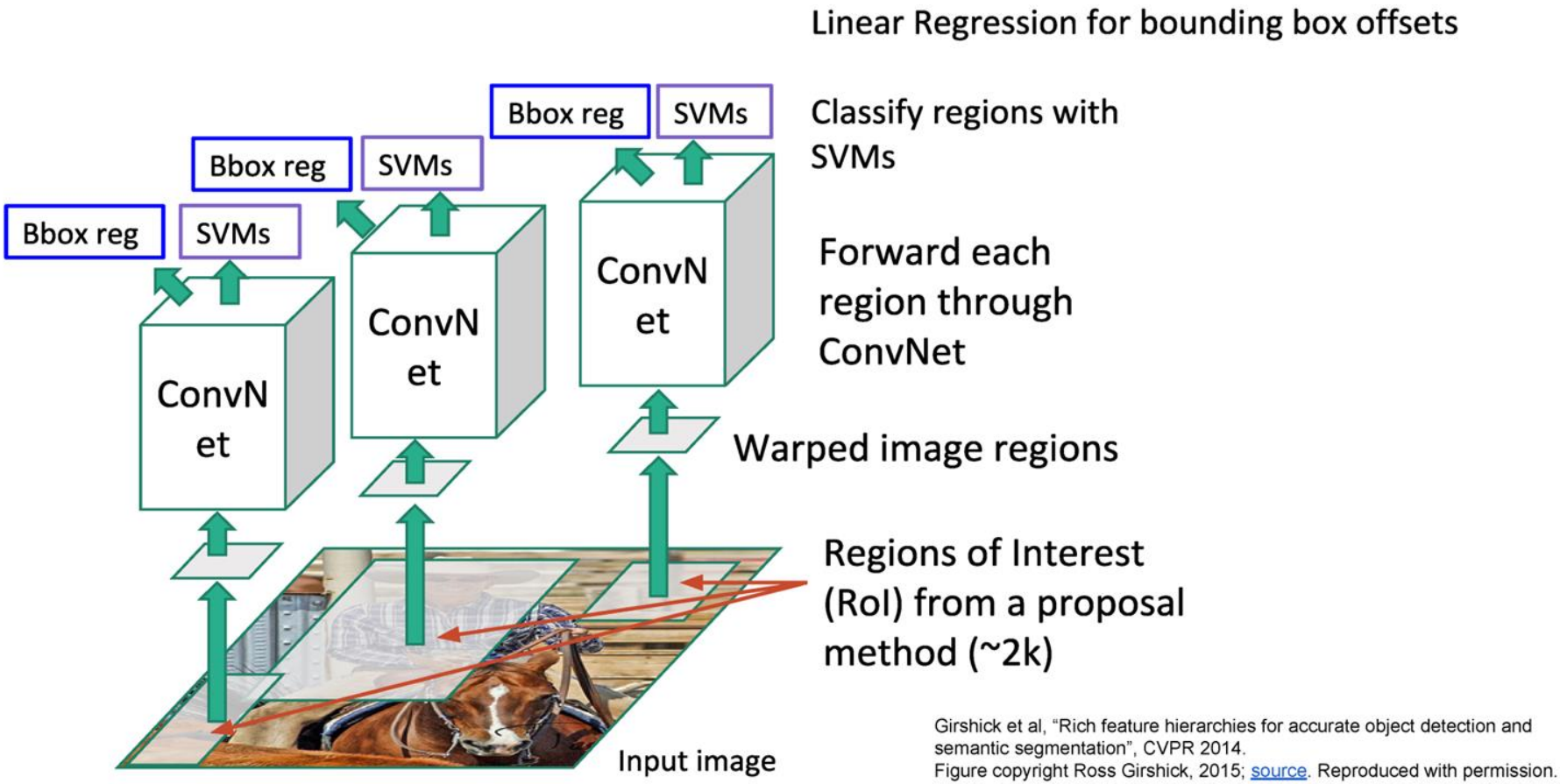
Forward each region through ConvNet

Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

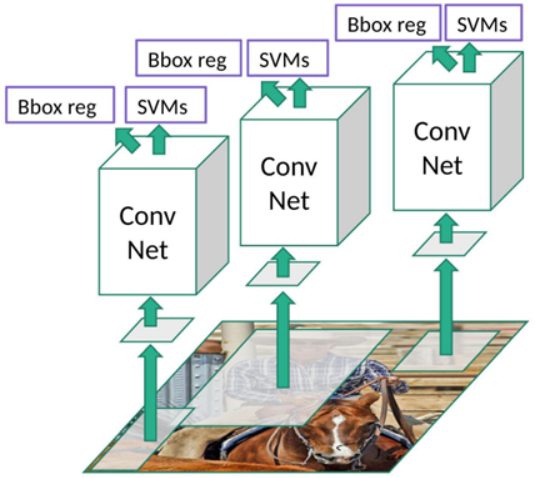
Region CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

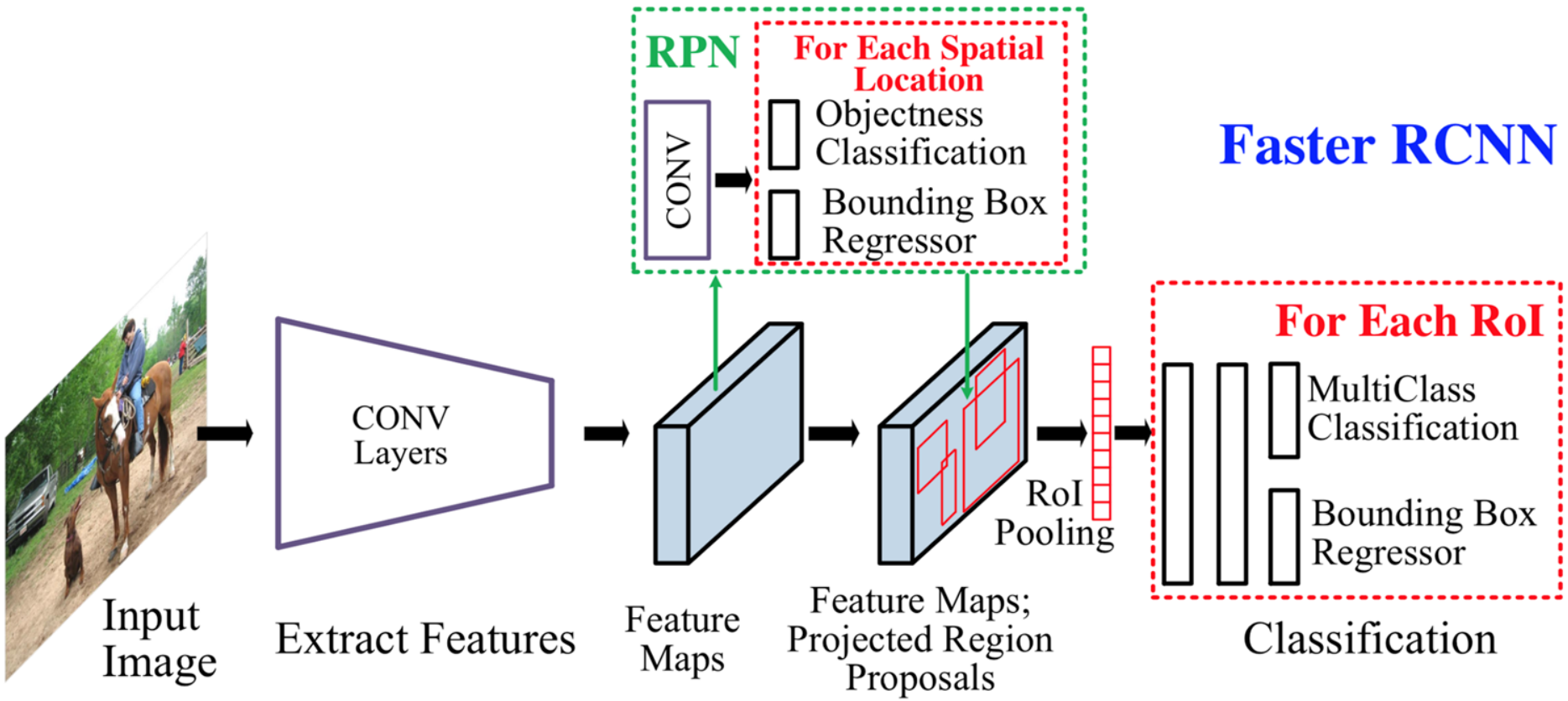
Region CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

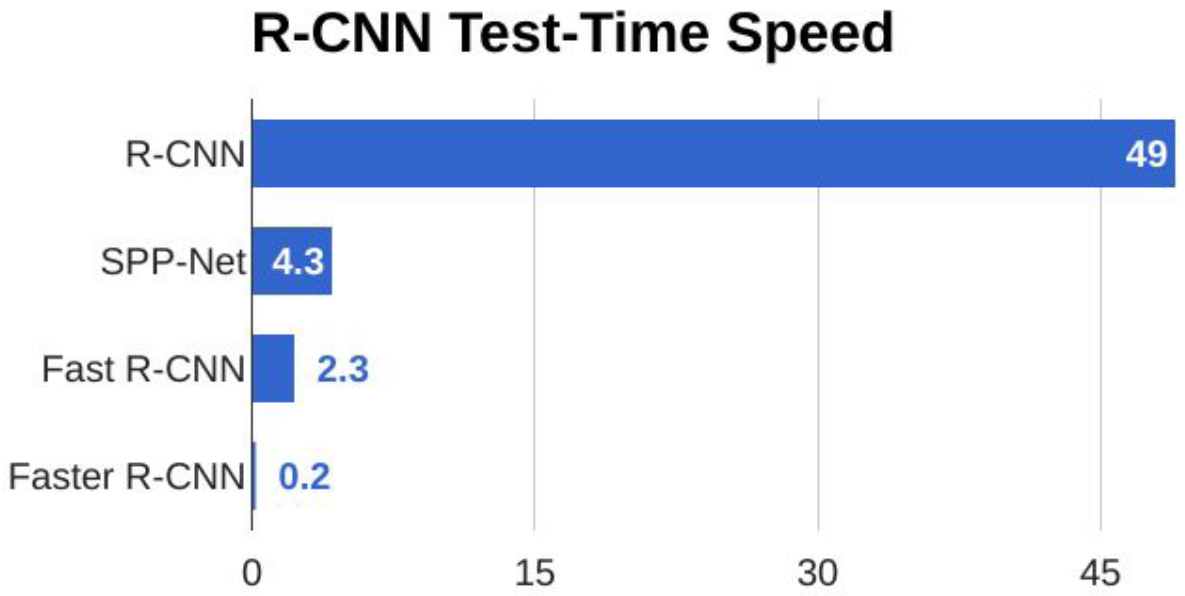
Faster RCNN



Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

Faster RCNN

Make CNN do proposals!



Pre Deep Era, RCNN, Fast RCNN, Faster RCNN

	Candidate Box Selection	Feature Extraction	Classification
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN			
Fast RCNN			
Faster RCNN			

Pre Deep Era, RCNN, Fast RCNN, Faster RCNN

	Candidate Box Selection	Feature Extraction	Classification
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN			
Faster RCNN			

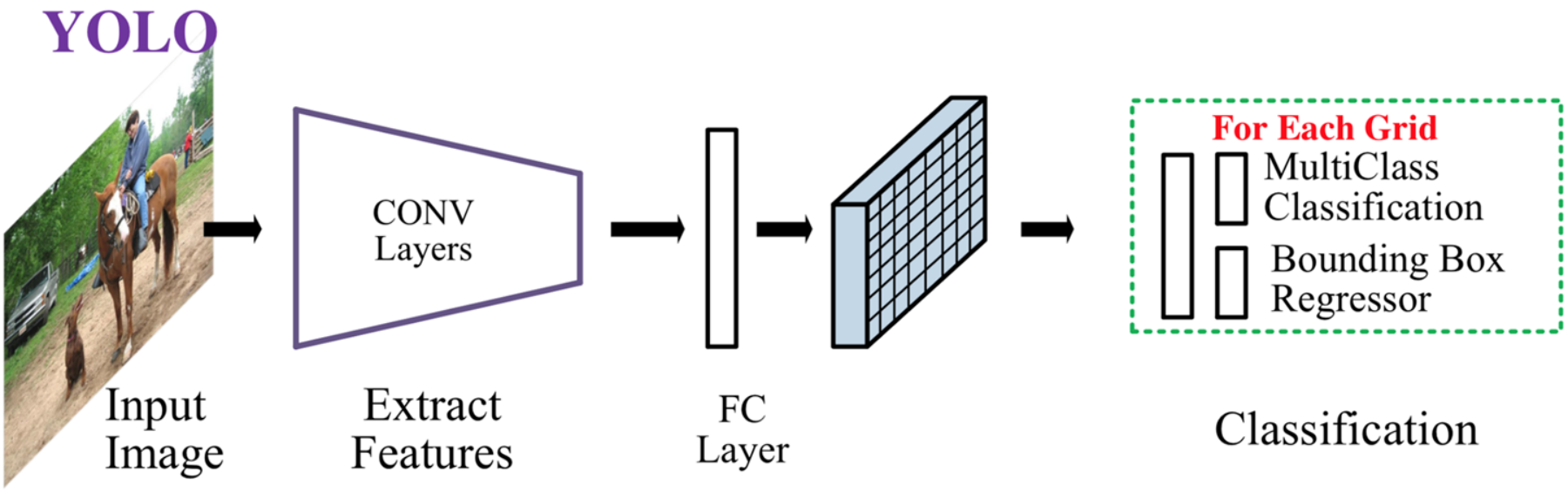
Pre Deep Era, RCNN, Fast RCNN, Faster RCNN

	Candidate Box Selection	**Feature Extraction**	**Classification**
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN	Region Proposal	Deep	
Faster RCNN			

Pre Deep Era, RCNN, Fast RCNN, Faster RCNN

	Candidate Box Selection	Feature Extraction	Classification
Pre Deep Era	Exhaustive	Hand-crafted (*e.g.* HOG)	Linear
RCNN	Region Proposal	Deep	Linear
Fast RCNN	Region Proposal	Deep	
Faster RCNN	Deep	Deep	

YOLO: You Only Look Once



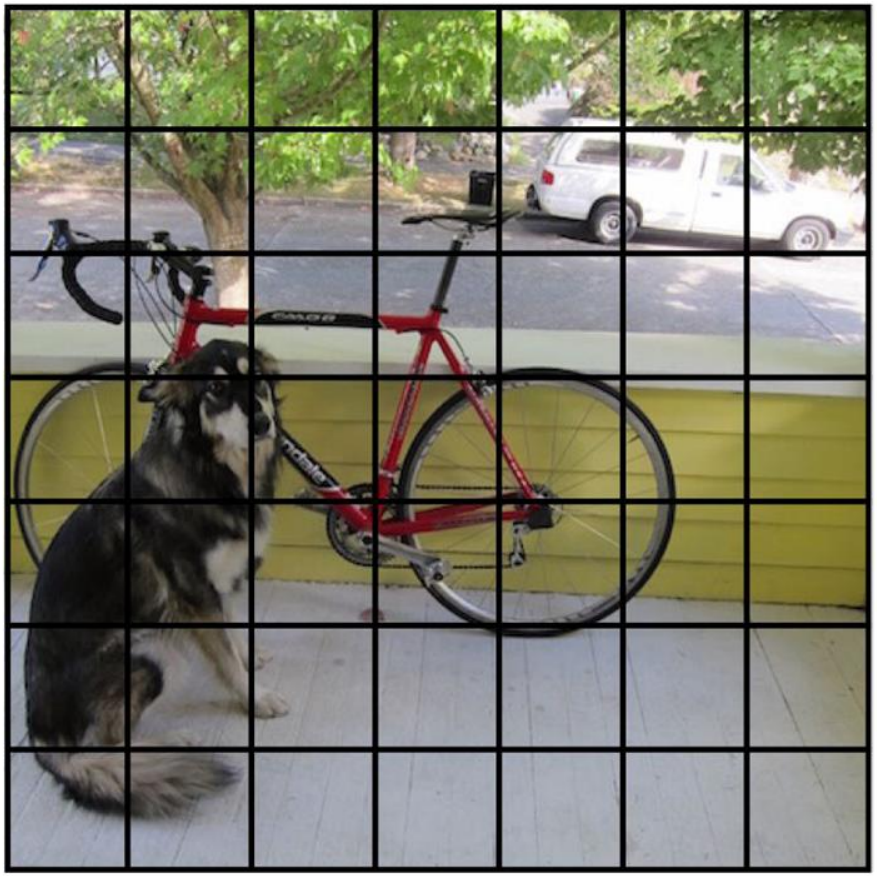
Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

YOLO: You Only Look Once

- Detection Procedure

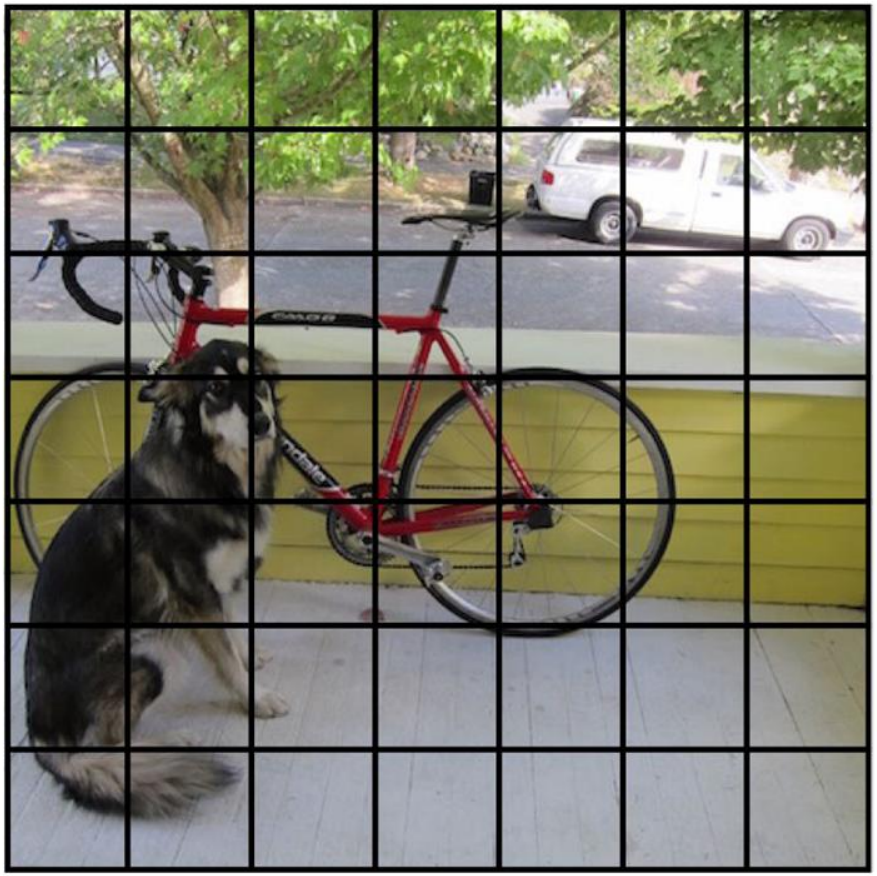
YOLO: You Only Look Once

We split the image into an S*S grid



YOLO: You Only Look Once

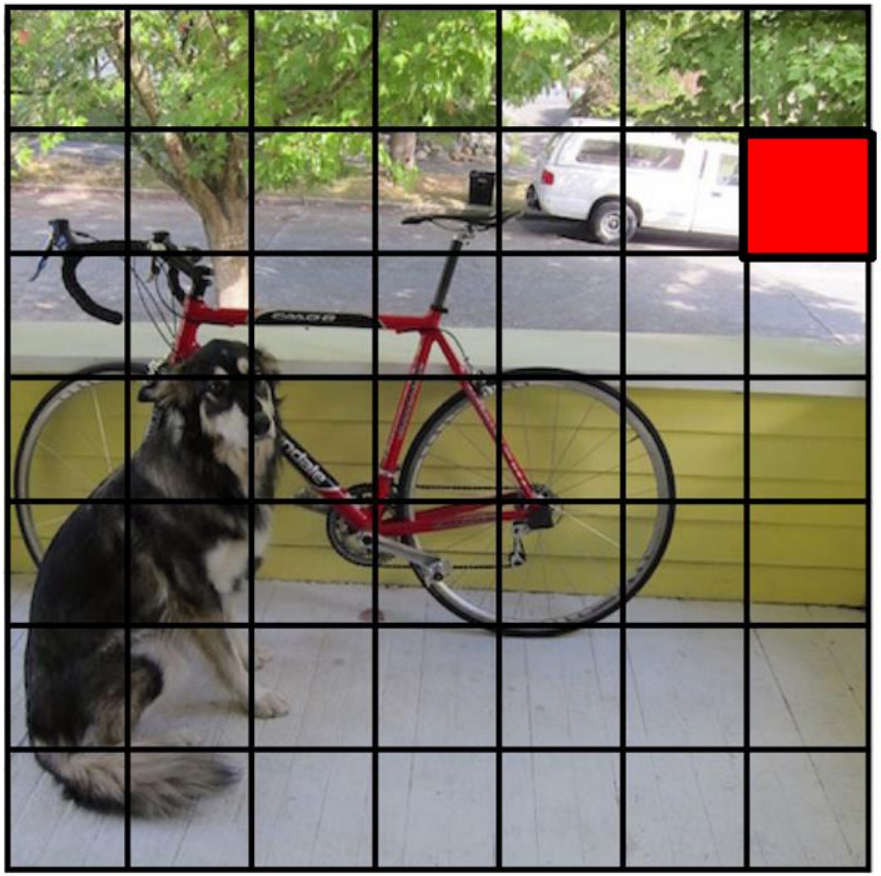
We split the image into an S*S grid



7*7 grid

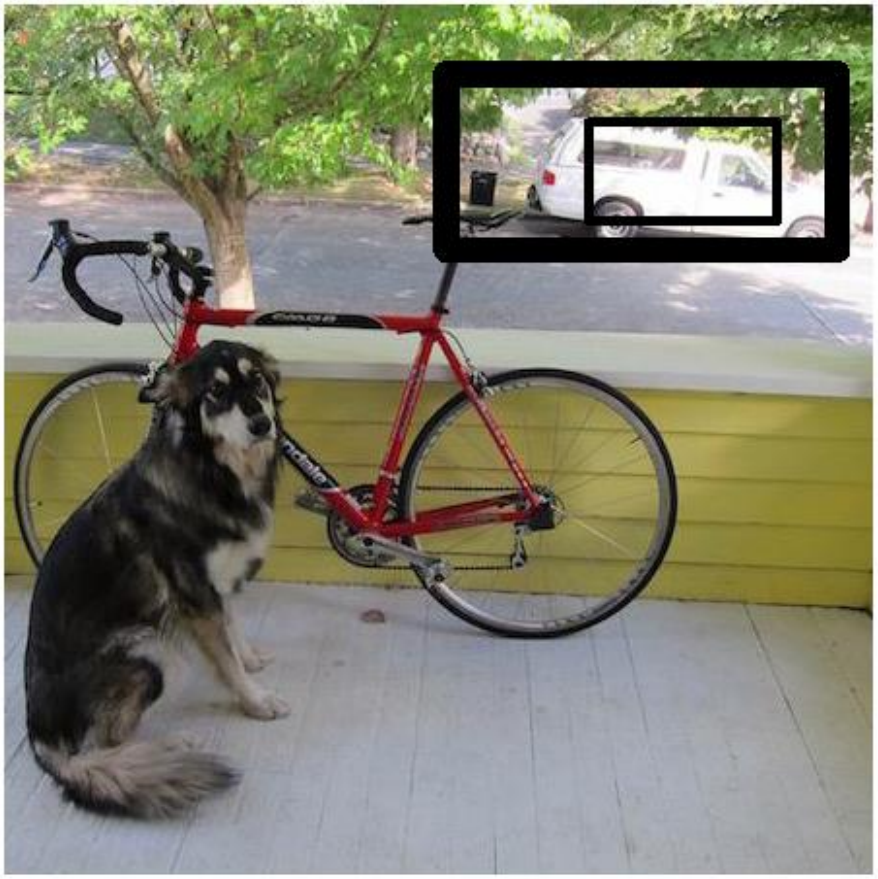
YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

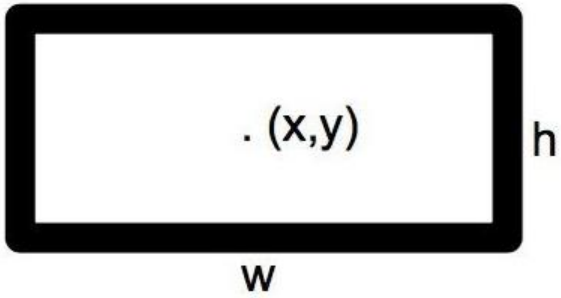


YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

B = 2

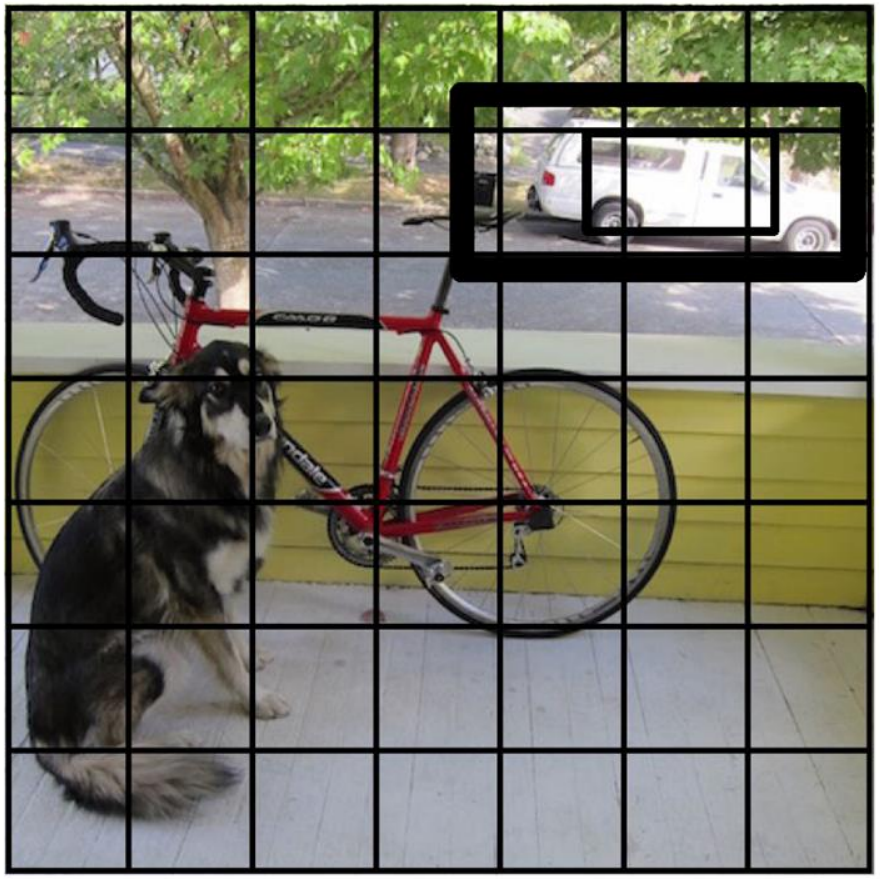
each box predict:



P(Object): probability that the box contains an object

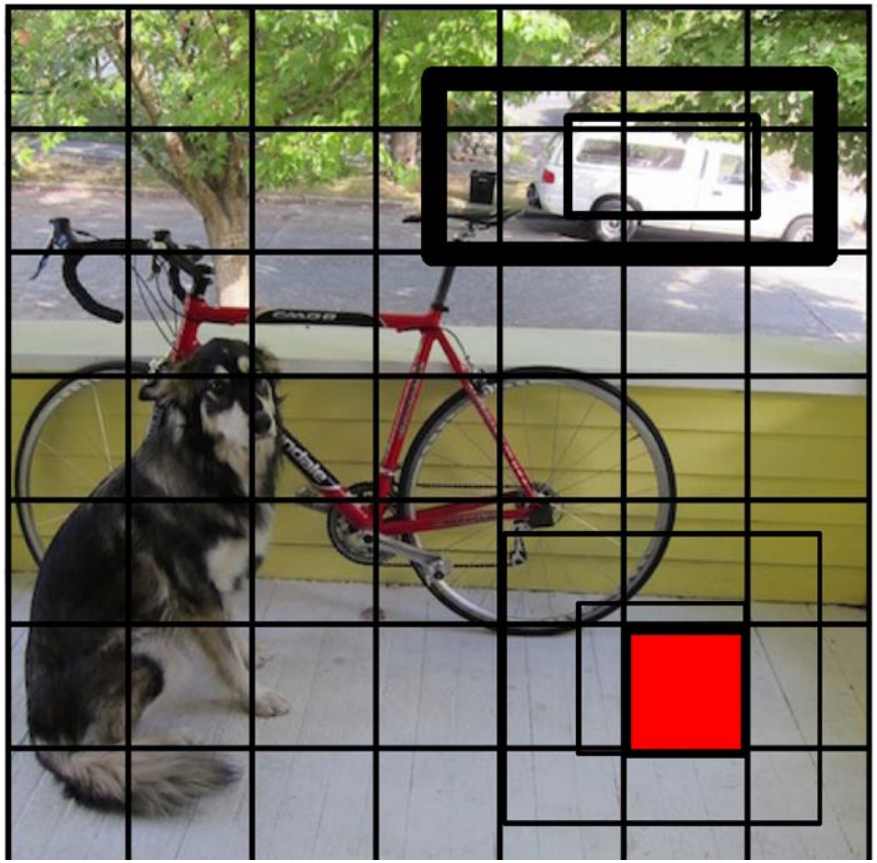
YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

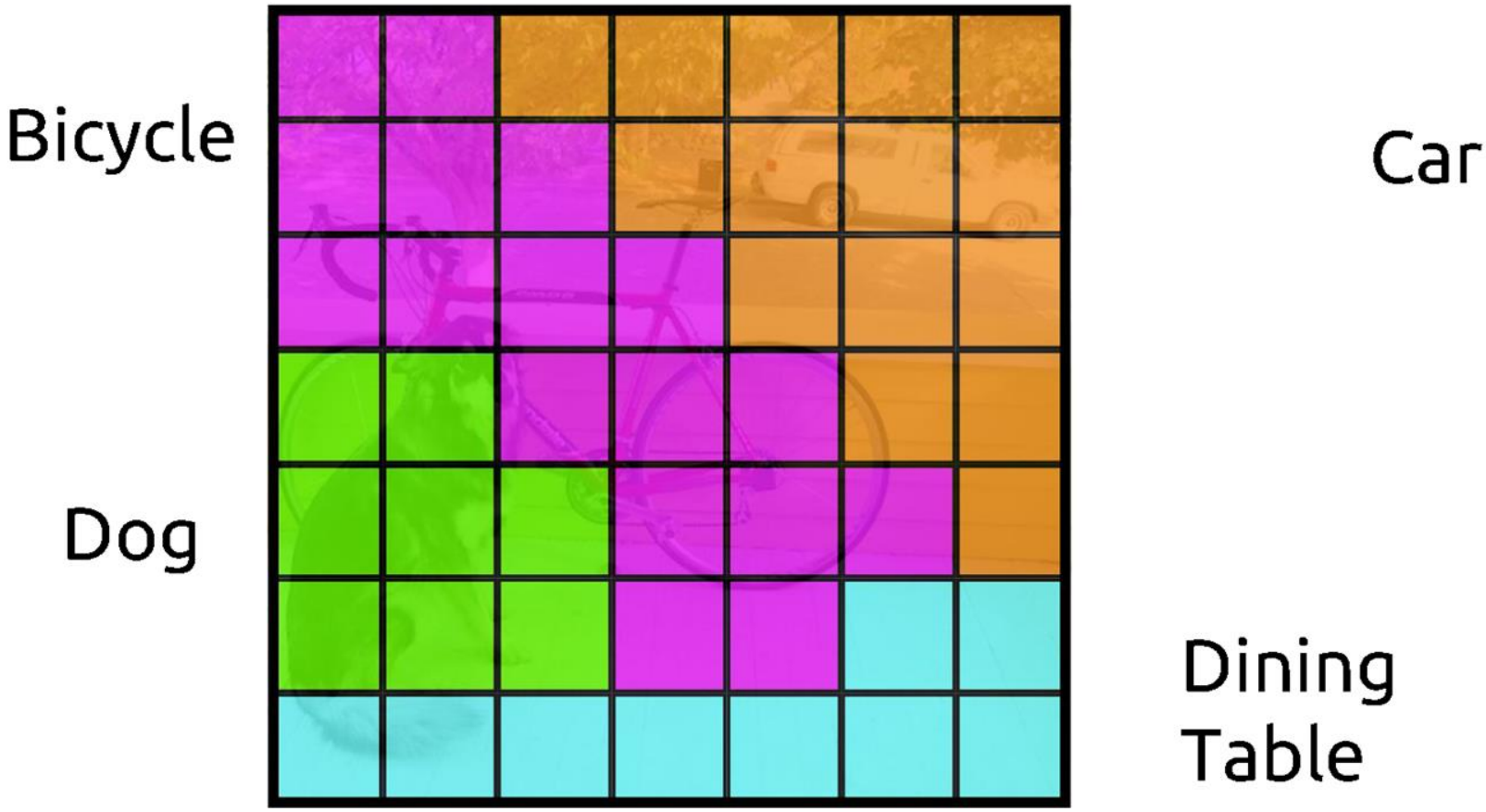


YOLO: You Only Look Once

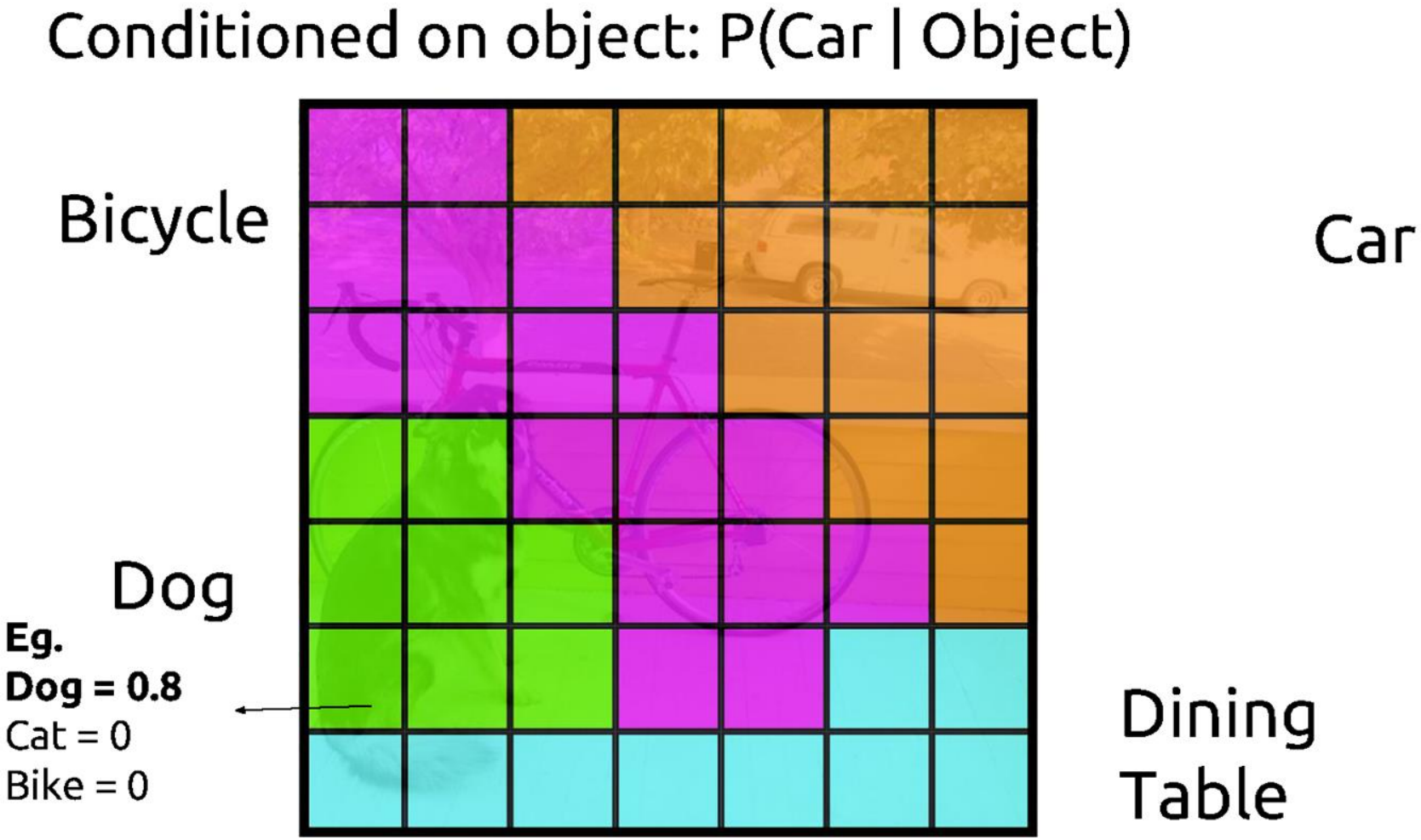
Each cell predicts boxes and confidences: P(Object)

YOLO: You Only Look Once

Each cell also predicts a class probability.

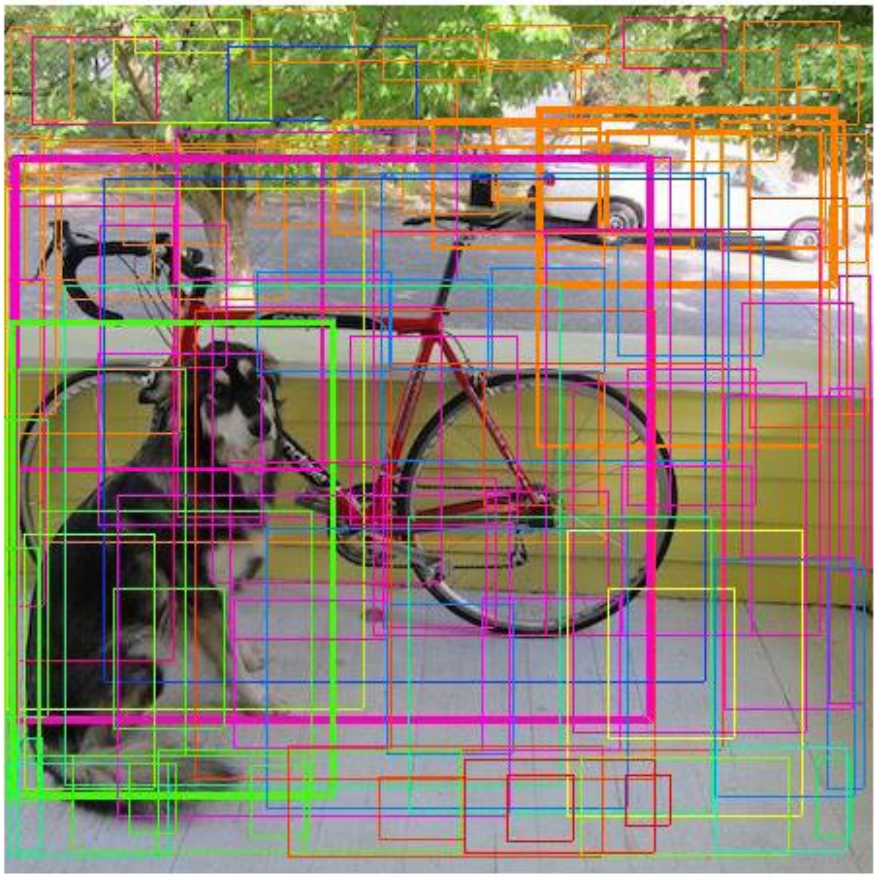


YOLO: You Only Look Once



YOLO: You Only Look Once

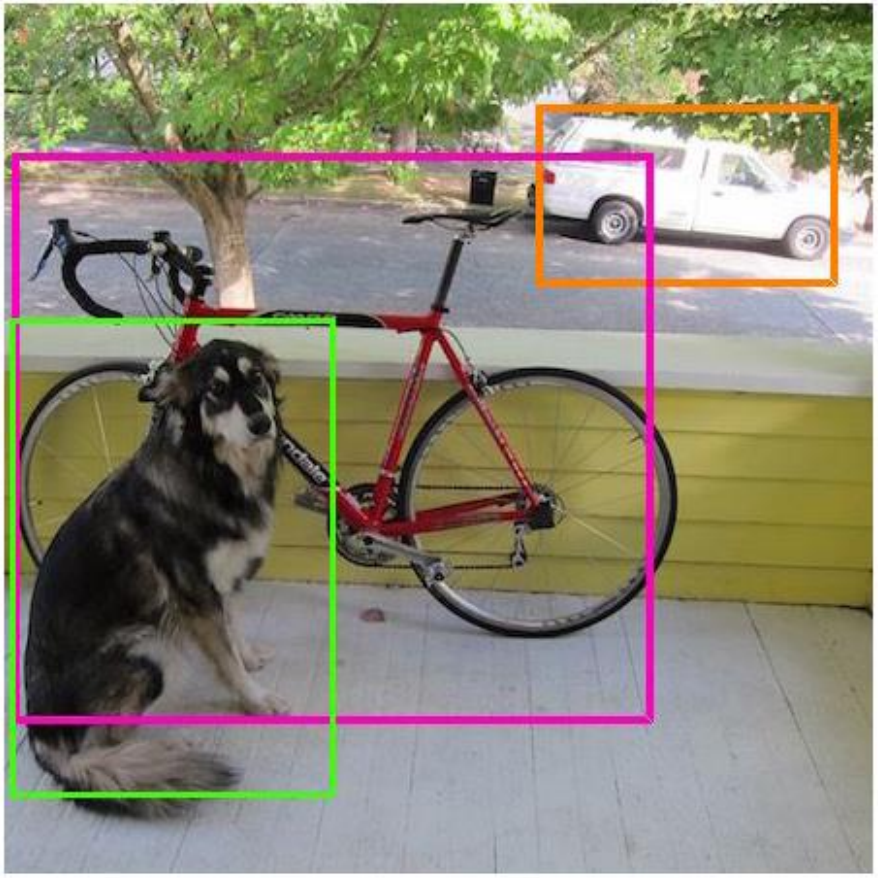
Then we combine the box and class predictions.



P(class|Object) * P(Object)
=P(class)

YOLO: You Only Look Once

Finally we do threshold detections and NMS

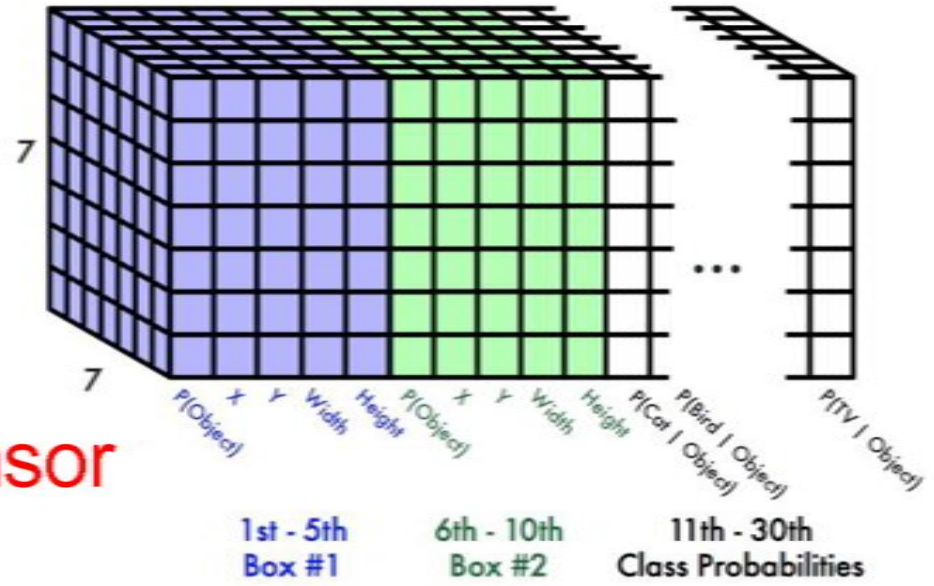


YOLO: You Only Look Once

https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_44

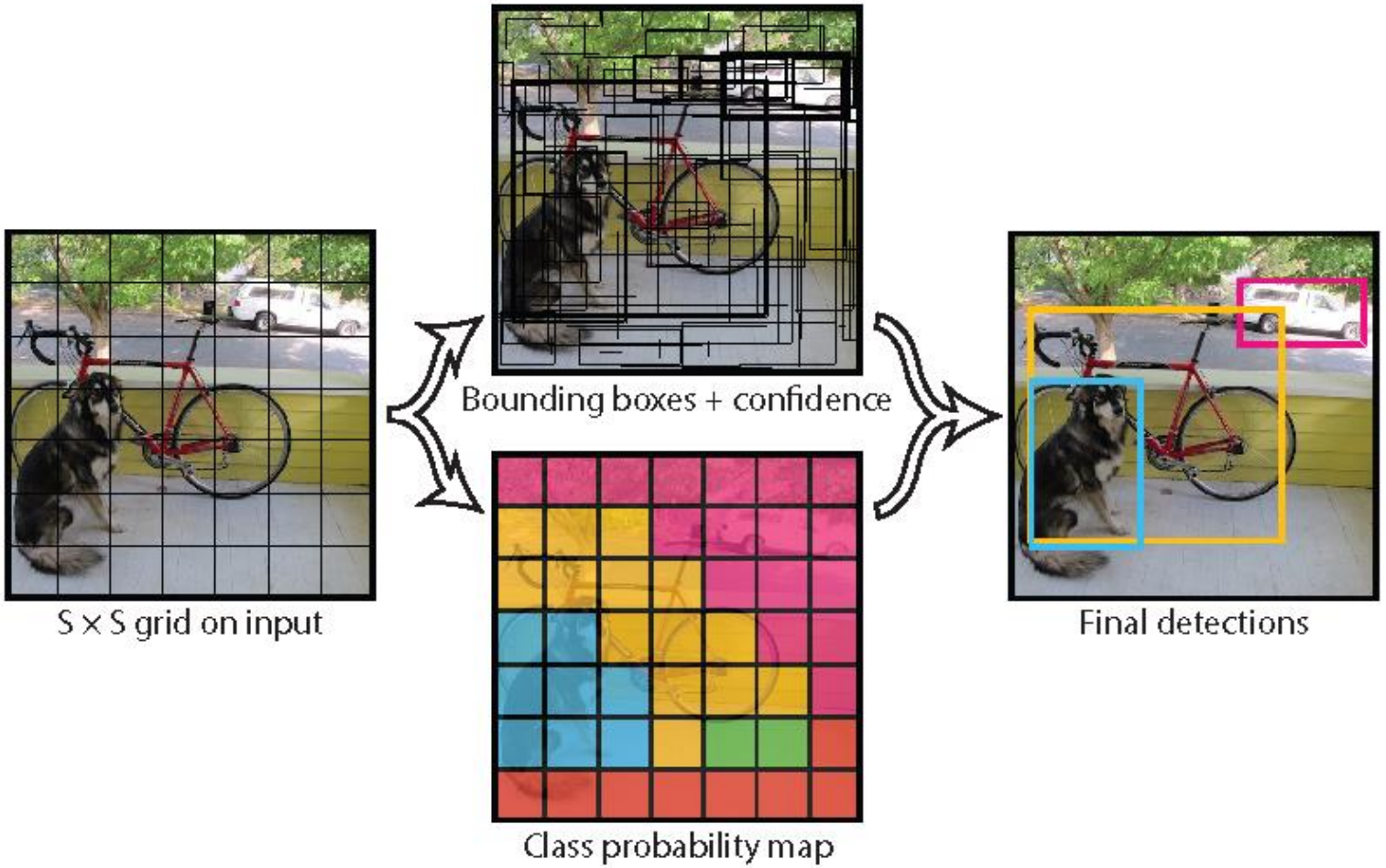
Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



S * S * (B * 5 + C) tensor

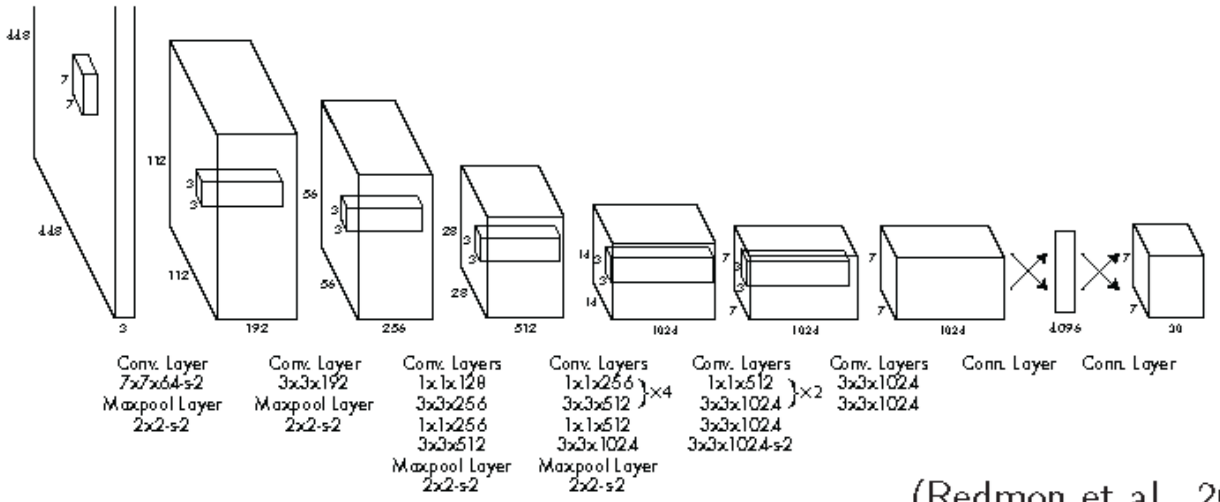
YOLO: You Only Look Once



(Redmon et al., 2015)

YOLO: You Only Look Once

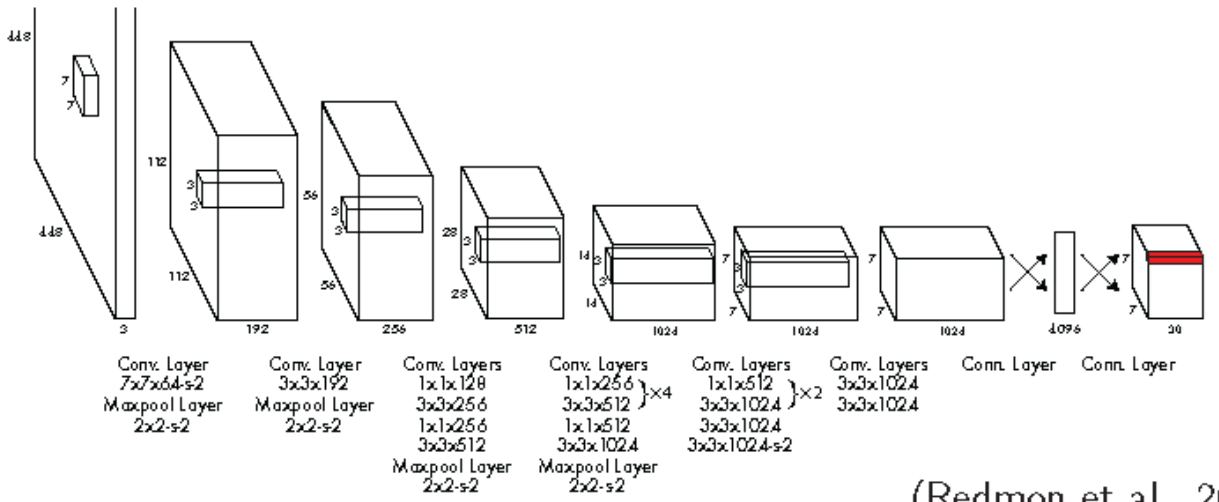
The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$



(Redmon et al., 2015)

YOLO: You Only Look Once

The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector

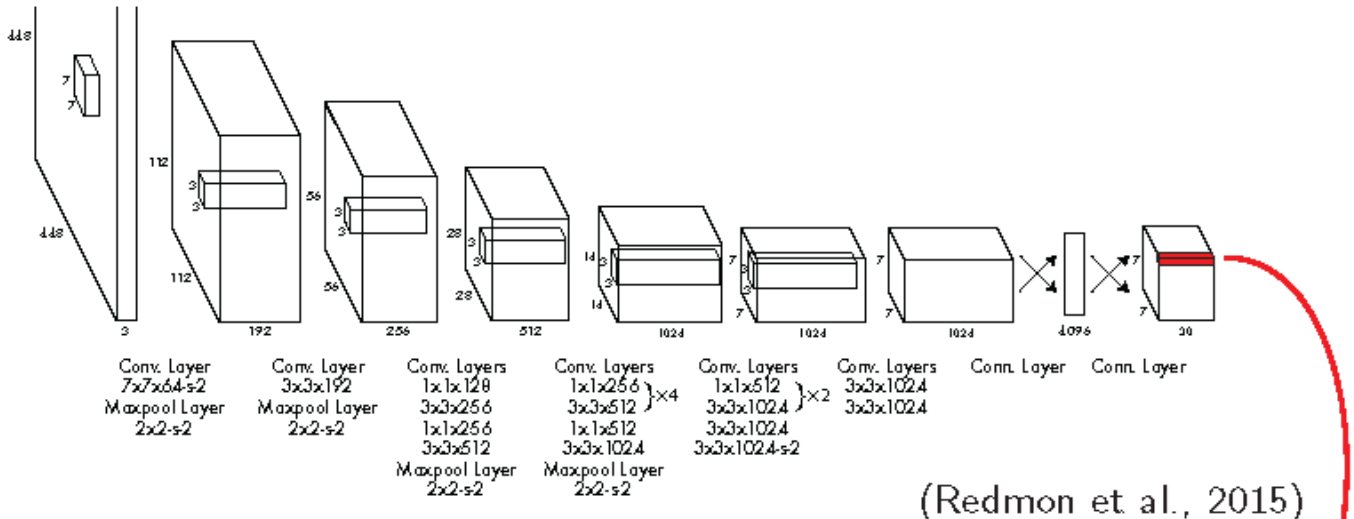


(Redmon et al., 2015)

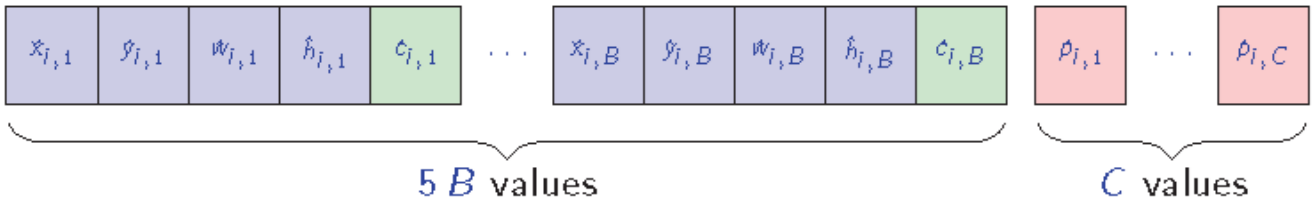
YOLO: You Only Look Once

The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector:

- $B = 2$ bounding boxes coordinates and confidence,
- $C = 20$ class probabilities, corresponding to the classes of Pascal VOC.



(Redmon et al., 2015)



References

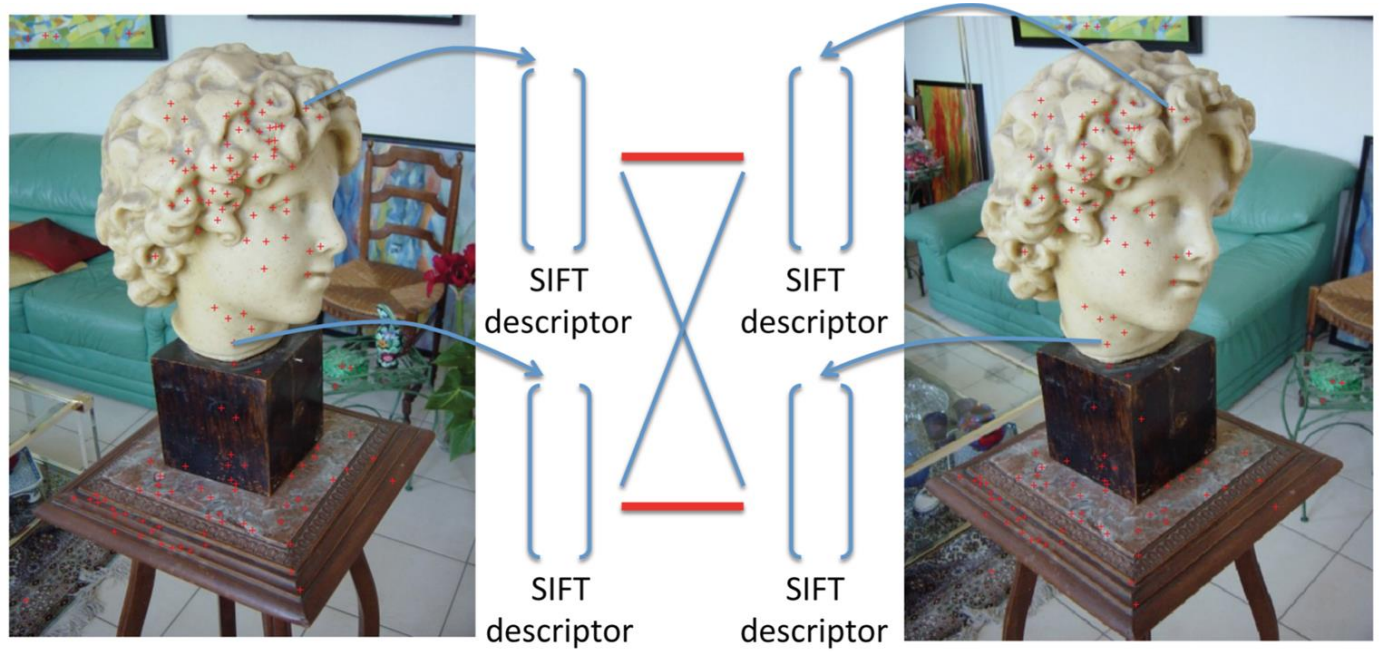
- https://medium.com/@2017csm1006/forward-and-backpropagation-in-convolutional-neural-network-4dfa96d7b37e
- http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/
- https://grzegorzgwardys.wordpress.com/2016/04/22/8/
- https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199
- https://computing.ece.vt.edu/~f15ece6504/slides/L5_cnn_backprop_notes.pdf
- https://www.youtube.com/watch?v=BvrWiL2fd0M
- http://scs.ryerson.ca/~aharley/vis/conv/
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

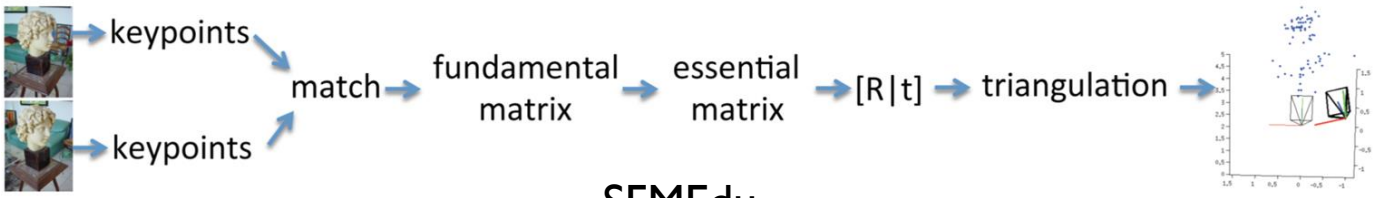
Optional Videos

- https://youtu.be/xh050sJFeQY
- https://youtu.be/3iCRY1IQAwk

Next: 3D Computer Vision

- Motion
- Rigid Structure from Motion
- Stereo





SFMEdu