Deep Learning

Data mining: Advanced Aspects

Siham Tabik
University of Granada
siham@ugr.es

Schedule

Deep Learning		
February 13th	February 14th	February 20th
Deep Neural Networks. • Convolutional Neural Networks (CNNs) • Classification & detection • Two case studies	**Tensorflow.** • Get familiar • Build our first CNN model	**Recurrent Neural Networks.** • Some theory • an example using tensorflow • One case study

Logistics

- Forum for debates, …
  https://github.com/SihamTabik/Patio_DL/issues
- Textbook “Deep Learning with Tensorflow”. Please, download the pdf from PRADO.

Convolutional Neural Networks

Siham Tabik
University of Granada
siham@ugr.es

Outline

1. What are ANN and CNNs?
2. How do CNNs work?
   - Convolution layers, pooling layers, FC layers, gradient, Backpropagation
3. Transfer learning
4. Data augmentation
5. One case study of CNN-classification model
6. CNNs for Detection
7. Two case studies

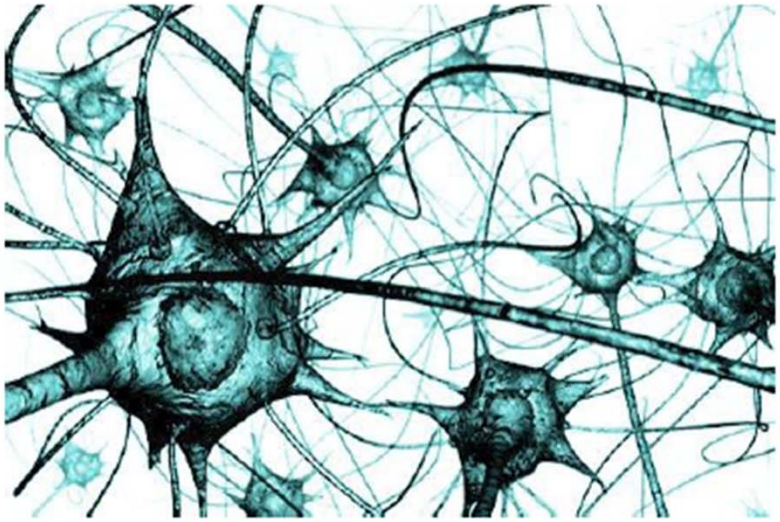
Outline

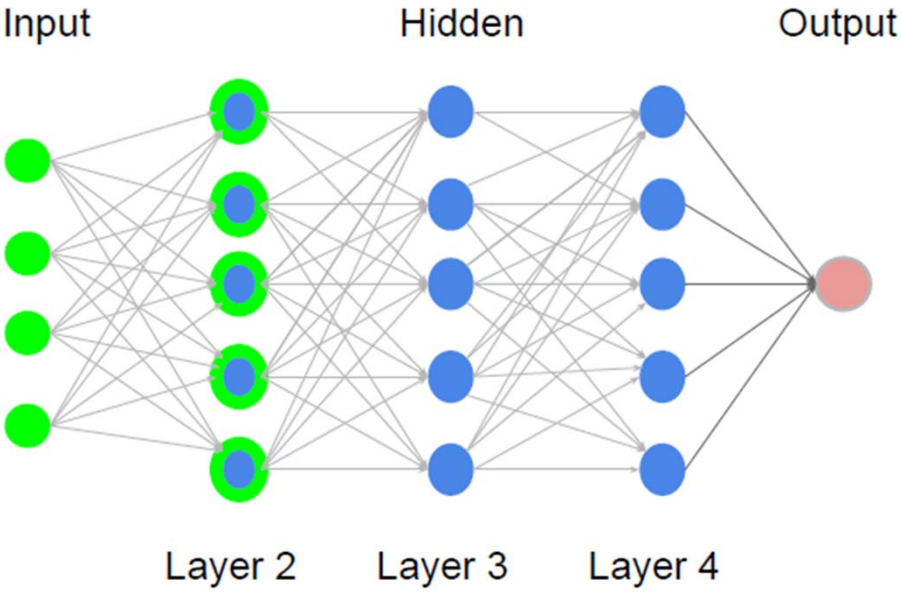
1. What are ANN and CNNs?
2. How do CNNs work?
   - Convolution layers, pooling layers, FC layers, gradient, Backpropagation
3. Transfer learning
4. Data augmentation
5. One case study of CNN-classification model
6. CNNs for Detection
7. Two case studies

Artificial Neural Networks

Machine learning algorithms

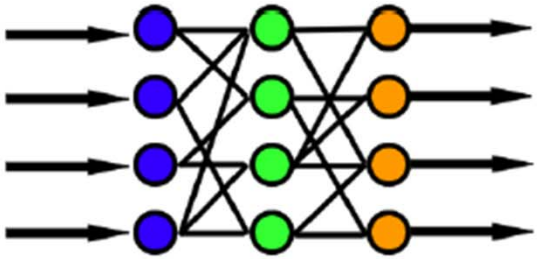
Learn and predict on data



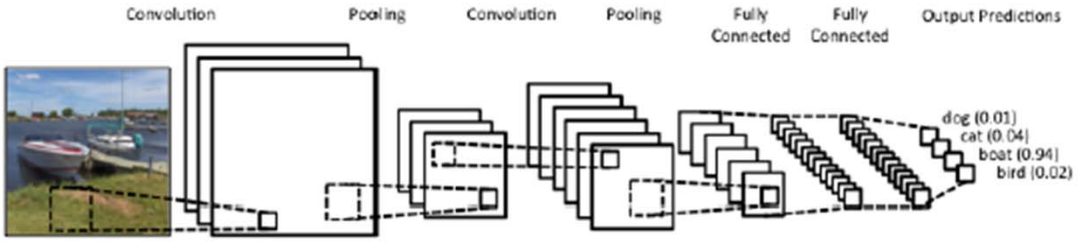


Artificial Neural Networks

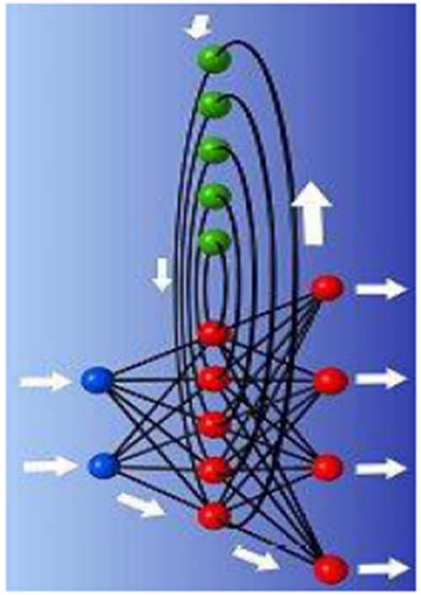
Feed Forward Nets (FNN)



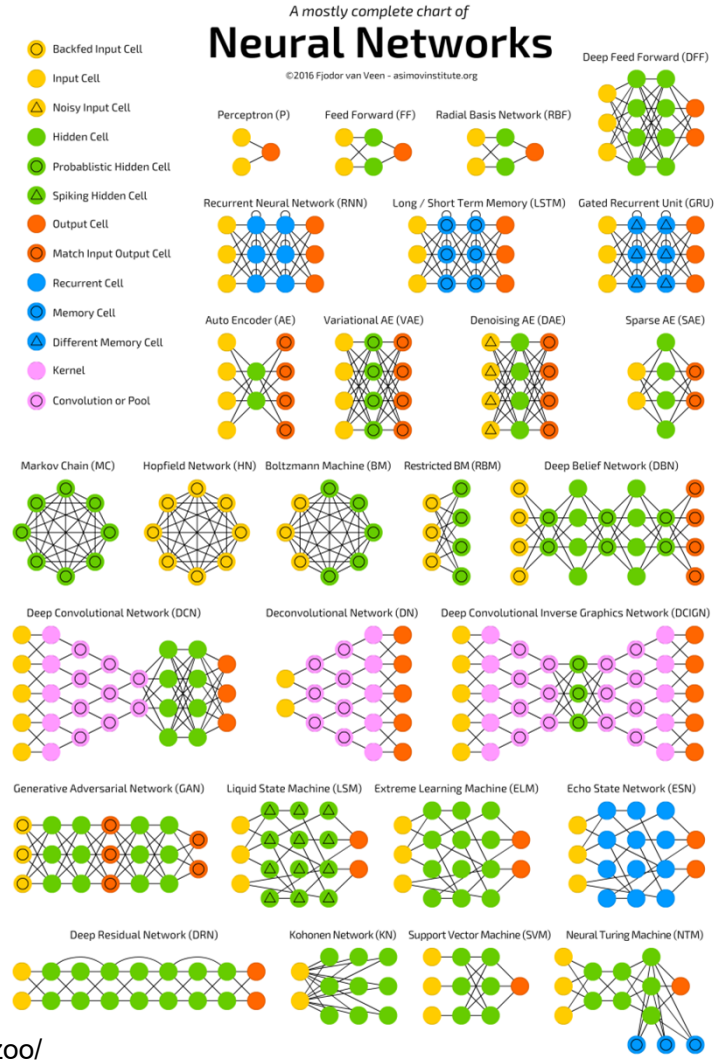
Convolutional Nets (CNN)



Recurrent Nets (RNN)

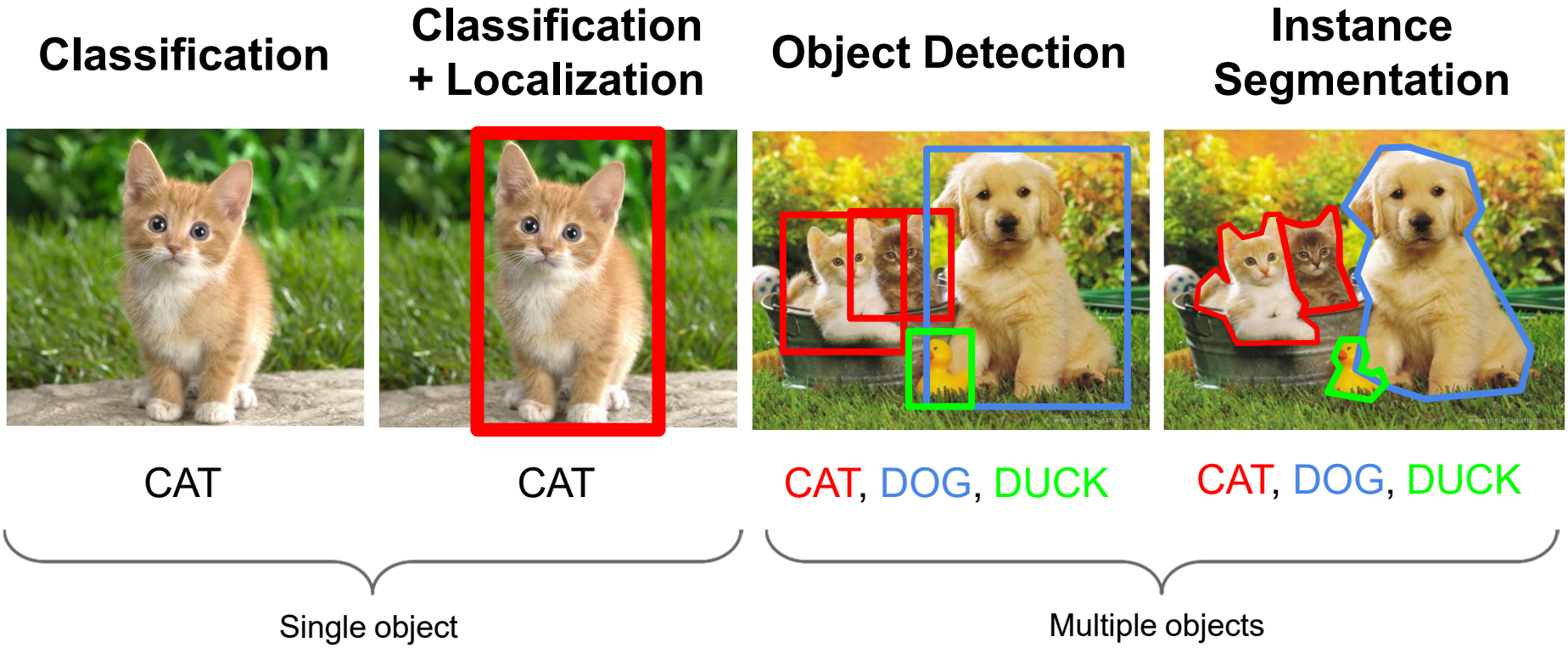


... and others

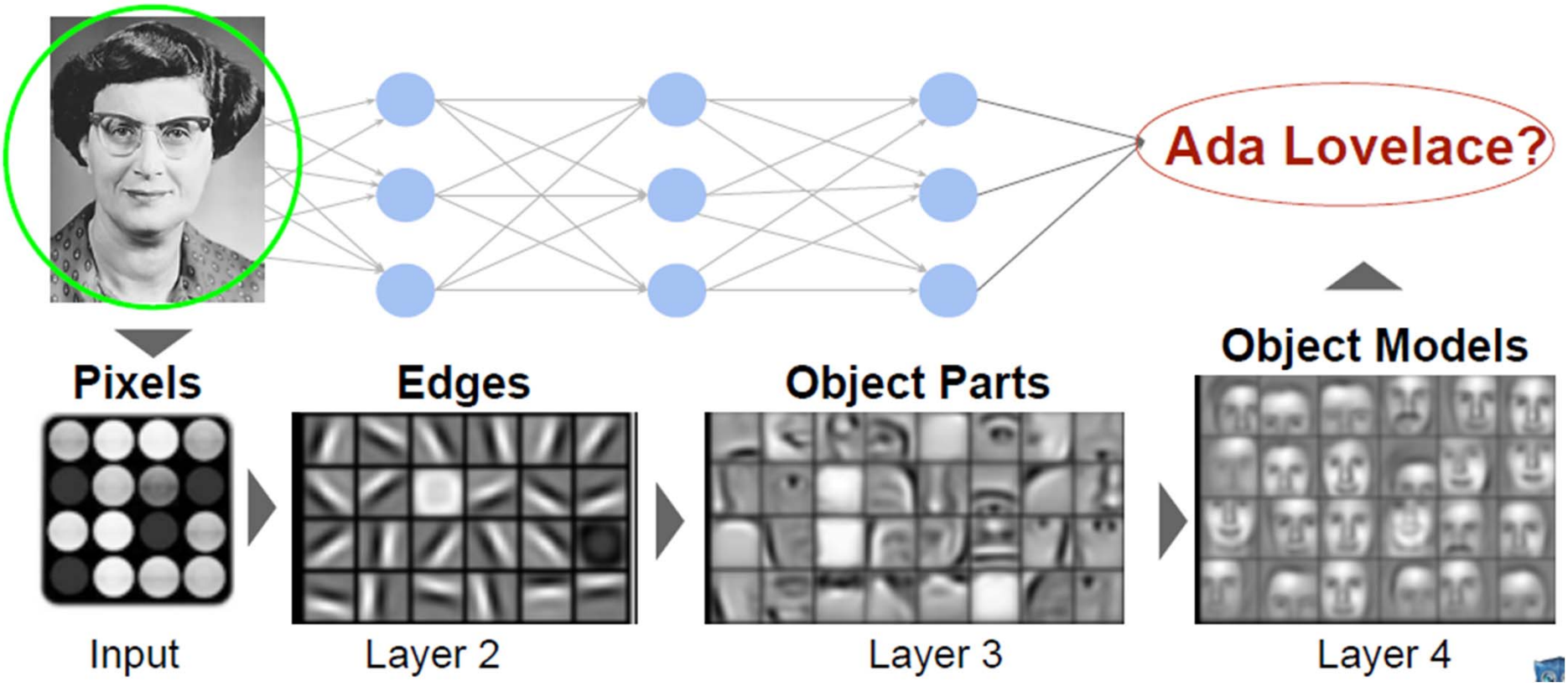


Source:
http://www.asimovinstitute.org/neural-network-zoo/

By the way, what is image classification?



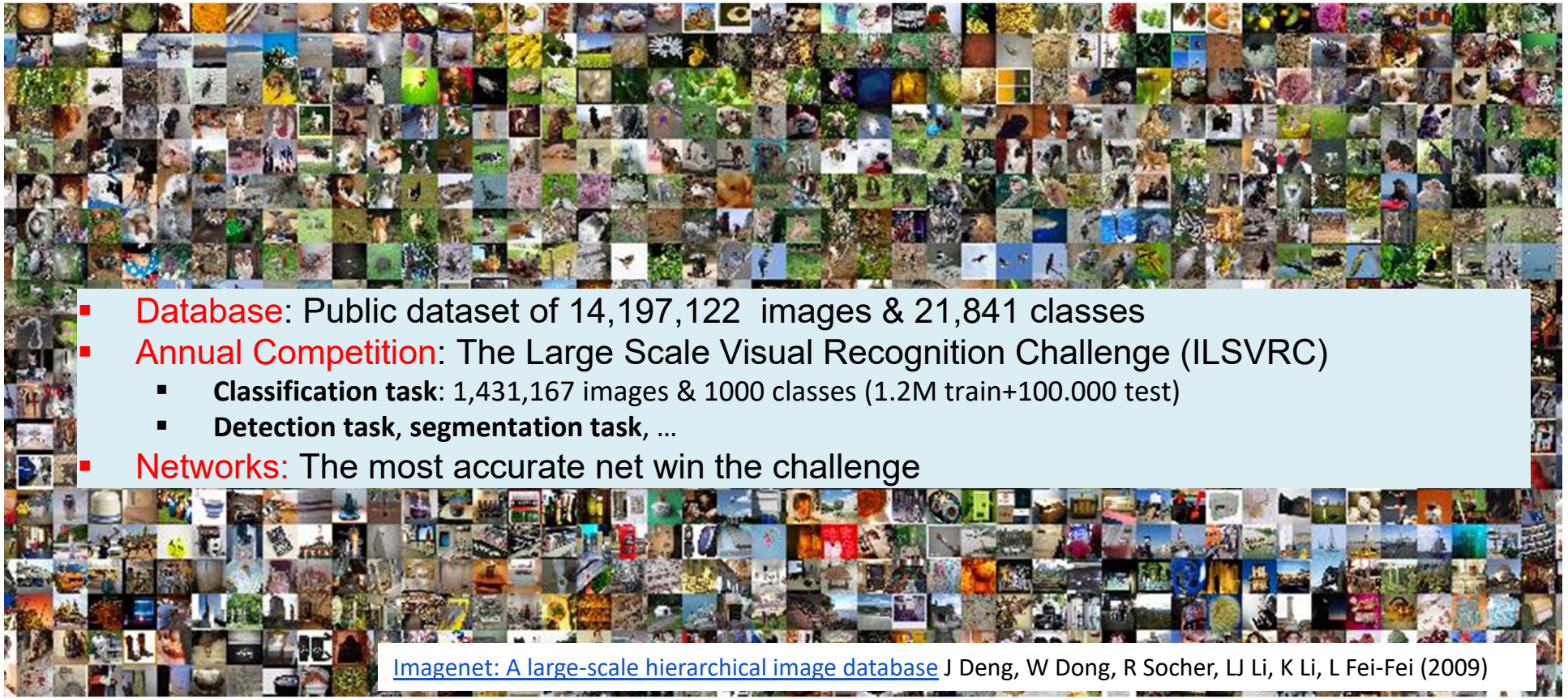
Convolutional Neural Networks



PASCAL Visual Object Classes Challenge (2005-2012)

- Database: Public dataset of 10,103 images& 20 object classes
- Annual Competition: PASCAL VOC Challenge
- Networks: The most accurate net wins the challenge

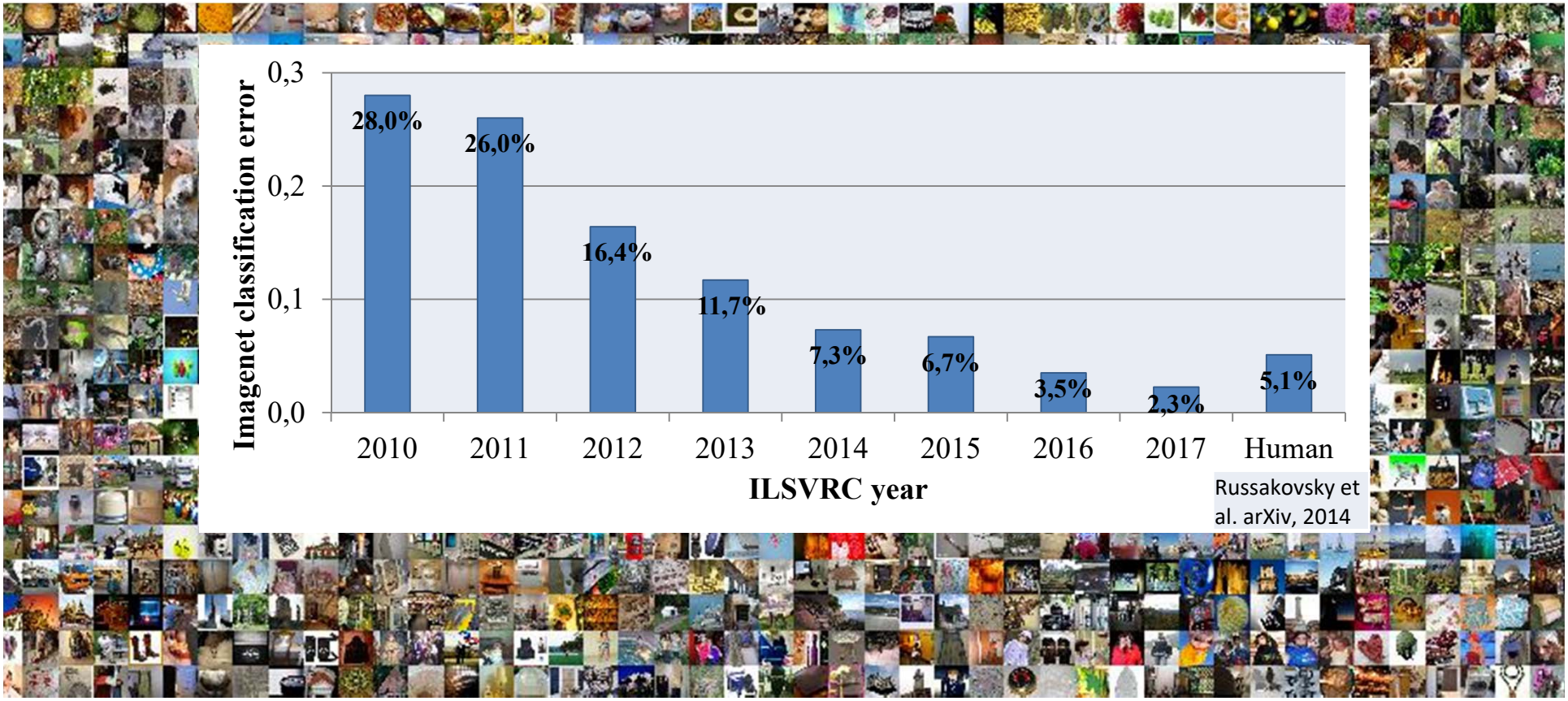
IMAGENET image classification challenge



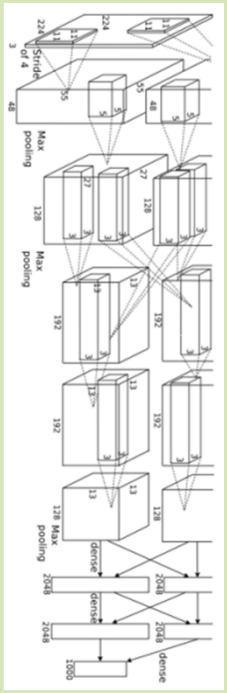
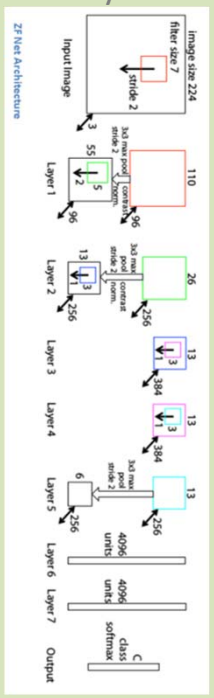
- Database: Public dataset of 14,197,122 images & 21,841 classes
- Annual Competition: The Large Scale Visual Recognition Challenge (ILSVRC)
  - **Classification task**: 1,431,167 images & 1000 classes (1.2M train+100.000 test)
  - **Detection task**, **segmentation task**, …
- Networks: The most accurate net win the challenge

Imagenet: A large-scale hierarchical image database J Deng, W Dong, R Socher, LJ Li, K Li, L Fei-Fei (2009)

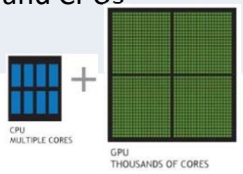
IMAGENET image classification challenge



IMAGENET image classification challenge

Year	**2012**	**2013**	**2014**		**2015**	**2016**
	AlexNet	ZFnet	GoogLeNet	VGG	ResNet	Inception&ResNet
	8 layers	8 layers	19 layers	22 layers	152 layers	
						
	Krizhevsky et al. NIPS	Zeiler and Fergus	Frizhevsky et al. arxiv &	simonyan et al. arxiv	He et al CVPR	

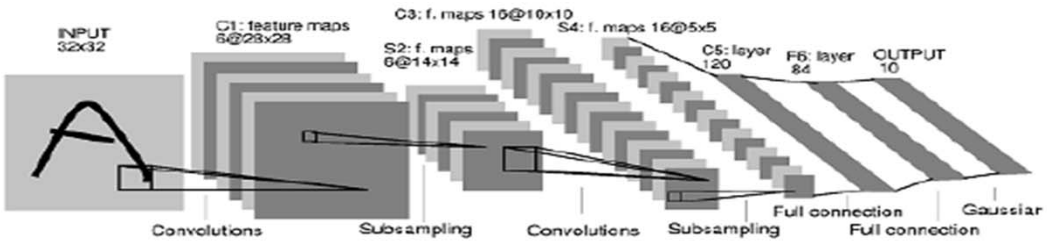
Evolution of CNNs

	1975	1985	1998	2009	2012
CNNs	**1st implementation of Conv. & pooling layers** by K. Fukushima	**1st CNN trained with Backpropagation**	**1st CNN trained with backp.** on MNIST: **LeNet** (0.8% error) by Y. LeCun		Won the ILSVRC classification challenge
Learning Algorithms	Costly	Backpropagation	Backpropagation	Backpropagation	Backpropagation
Database	Small	Small	MNIST	1st massive dataset **ImageNet** 	
Hardware	Limited capacity	Limited capacity	Limited capacity	Powerful GPUs and CPUs 	

CNNs architecture

1998

LeCun et al.



# of transistors

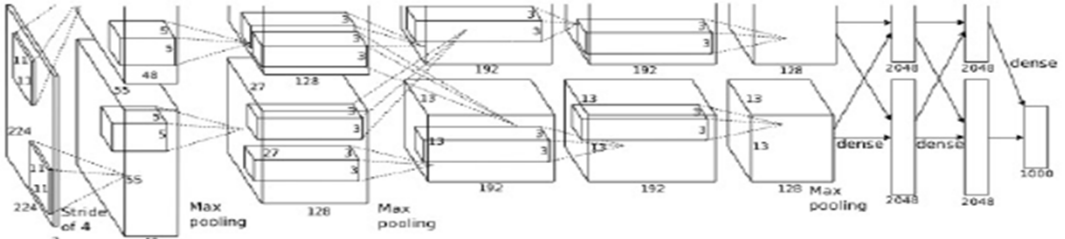
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

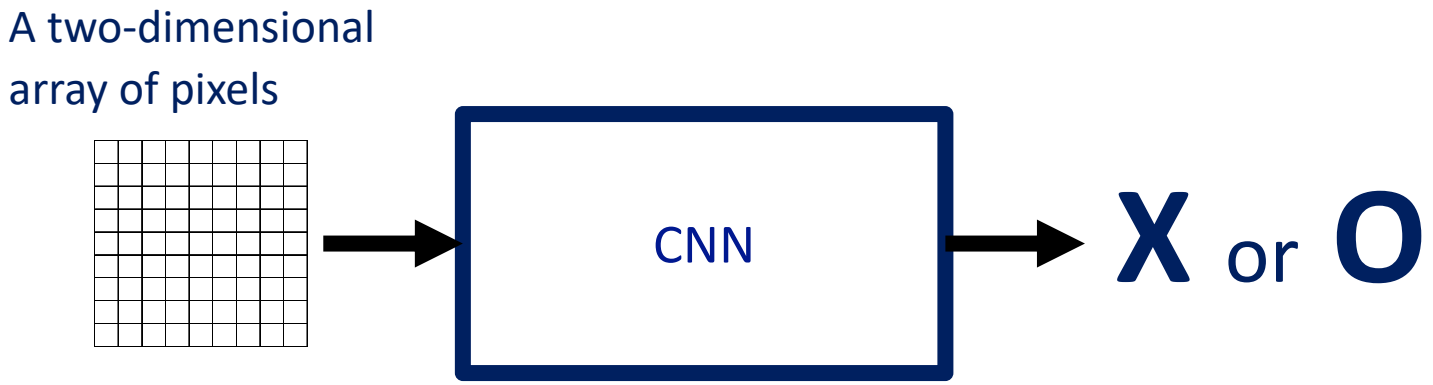
10^{14} IMAGENET

Outline

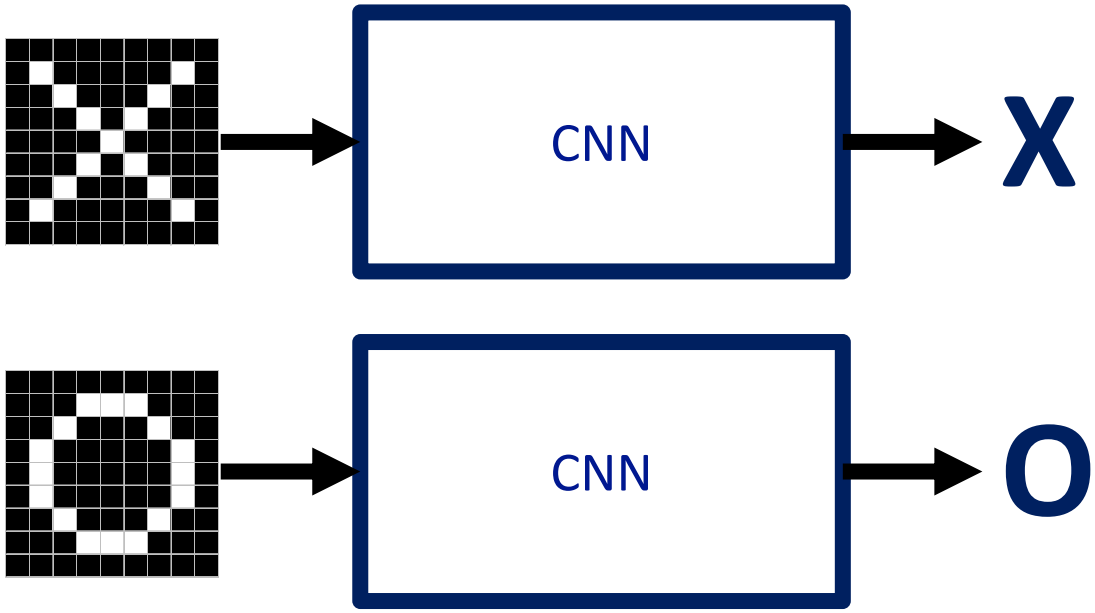
1. What are ANN?
2. What are CNNs?
3. How CNNs work?
   - Convolution layers, pooling layers, FC layers, Gradient, Backpropagation
4. Transfer learning
5. Data augmentation
6. One case study
7. CNNs for Detection
8. Two case studies

How CNNs work?

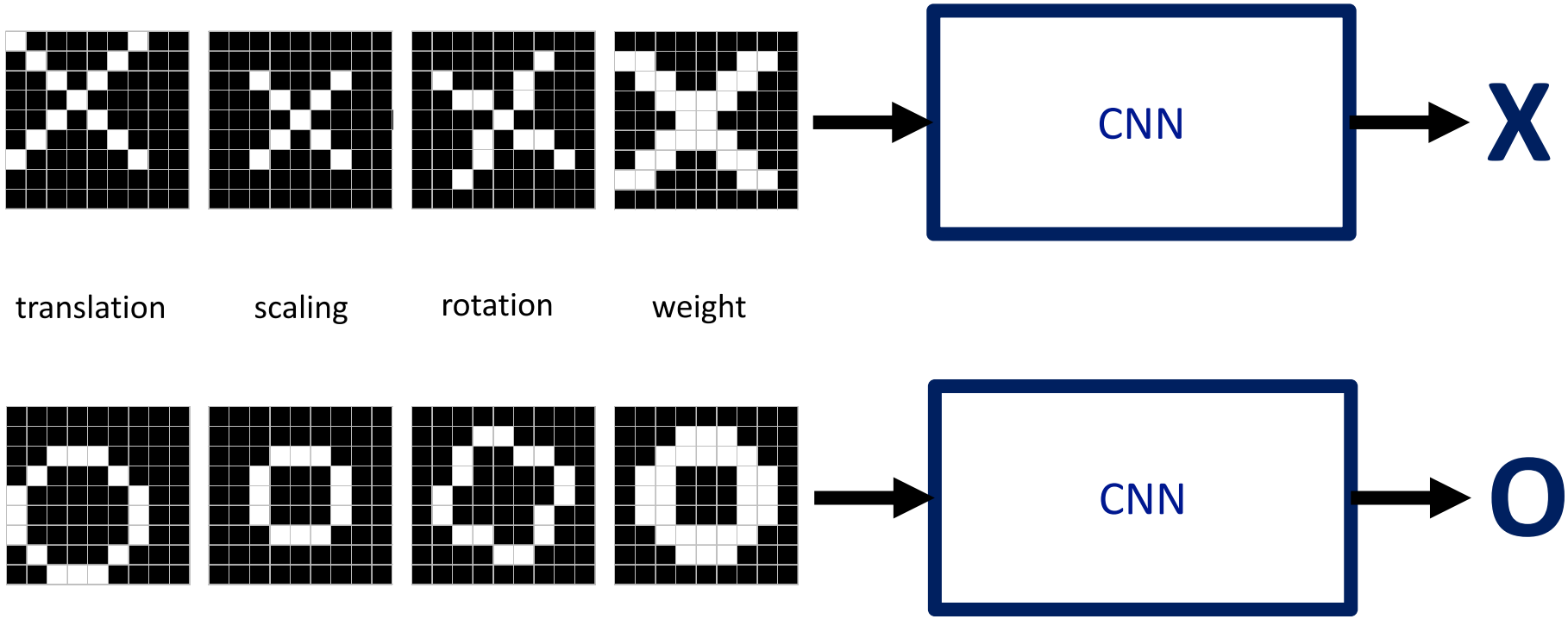
A toy CNN: says whether a picture is of an X or an O



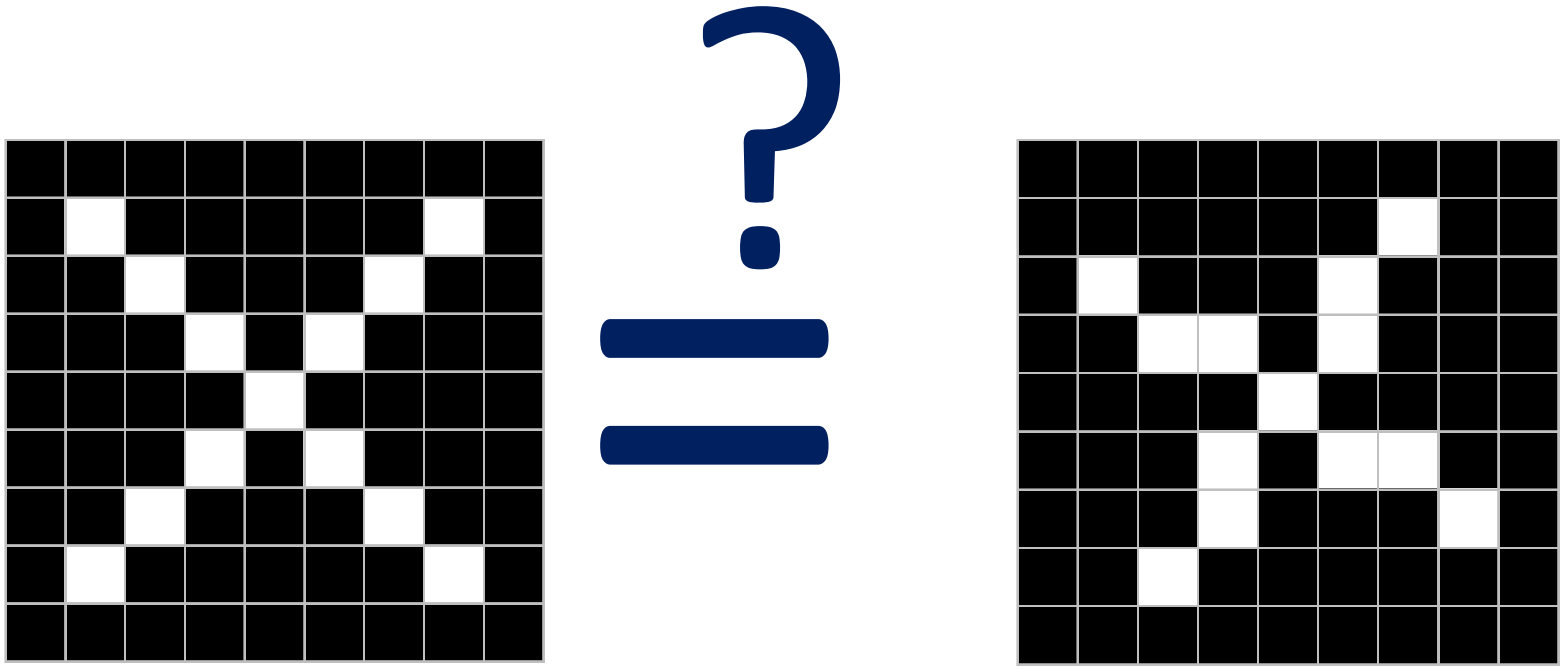
For example



Trickier cases



Deciding is hard



Deciding is hard

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

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

What computers see

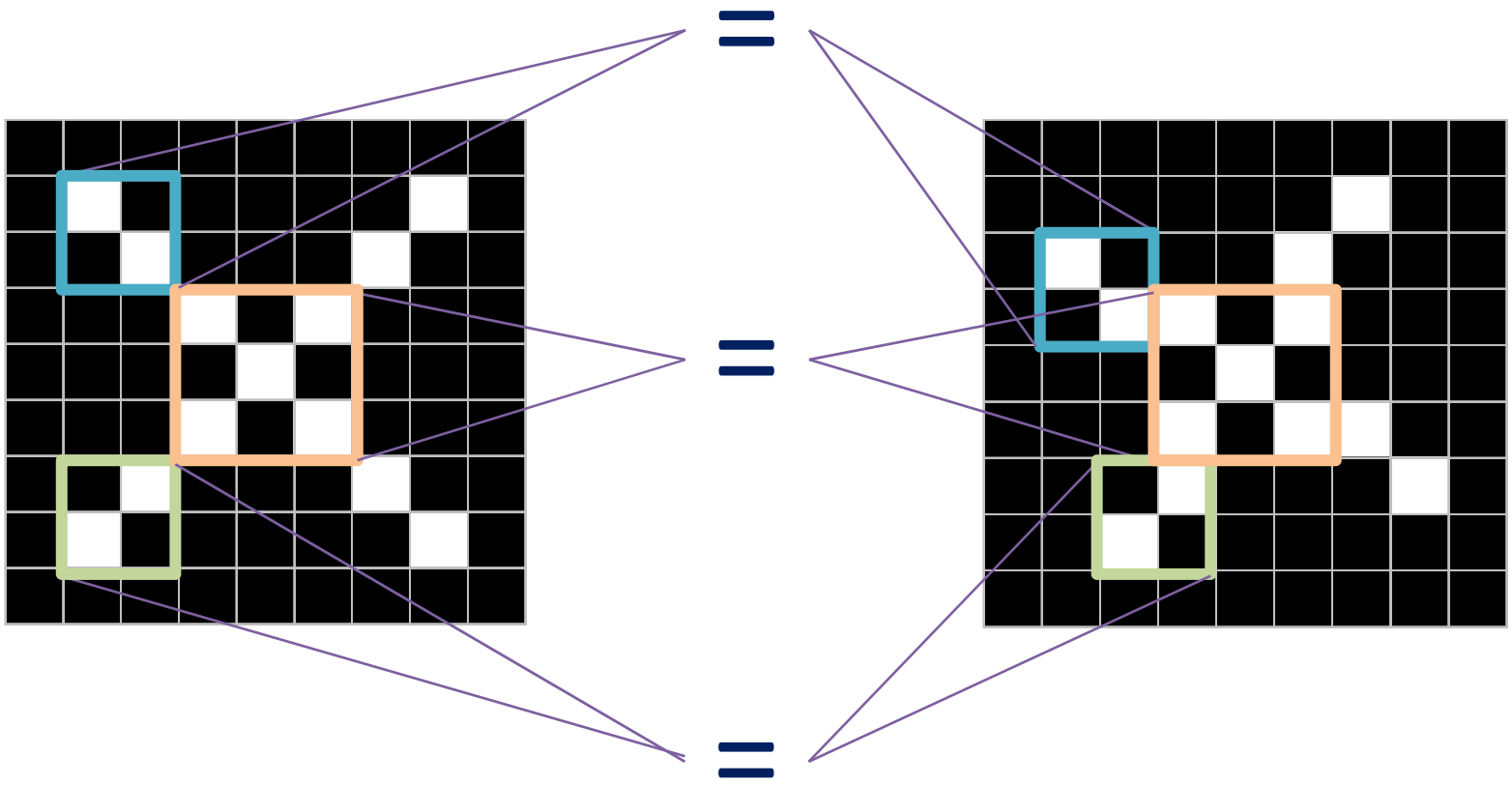
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

Computers are literal

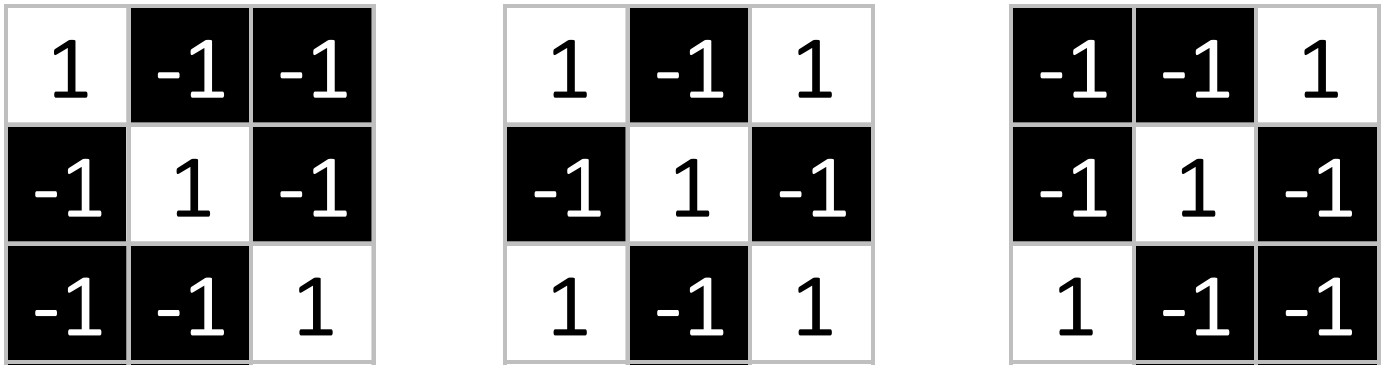
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

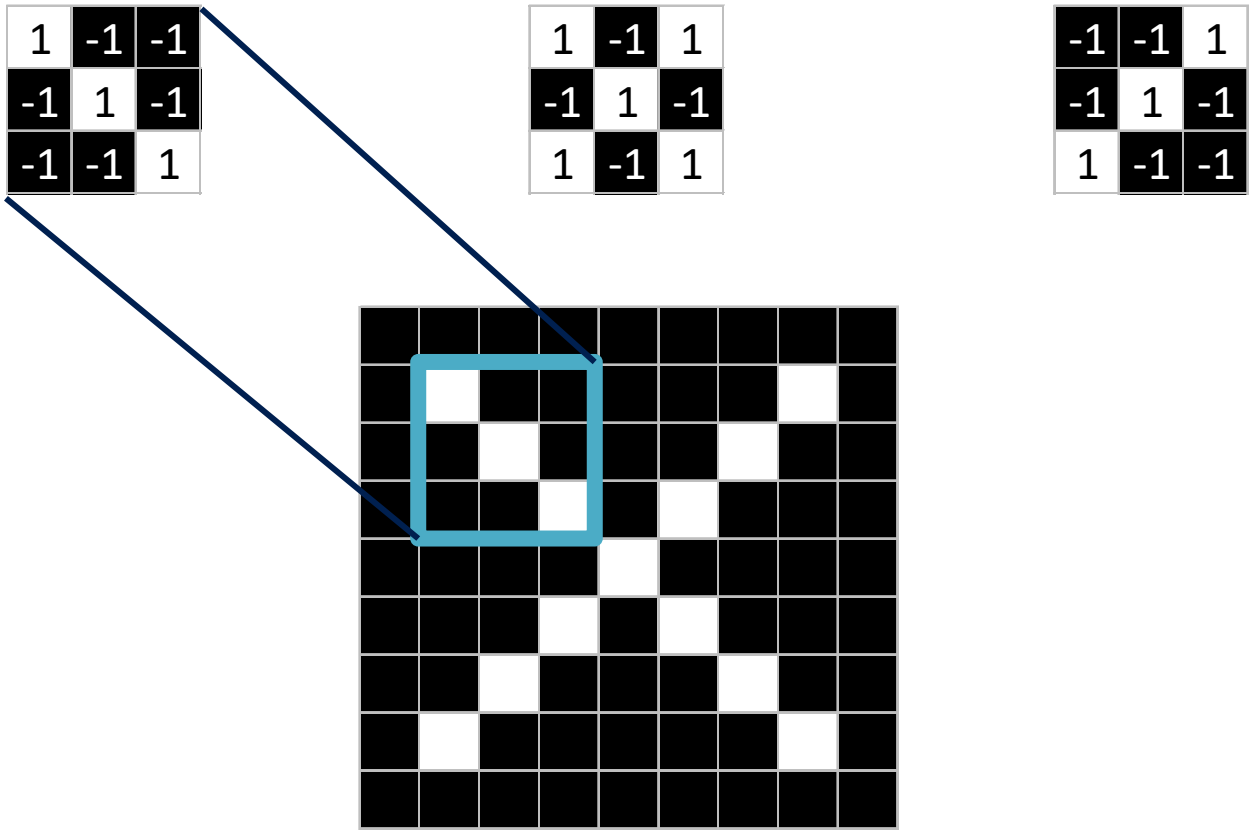
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

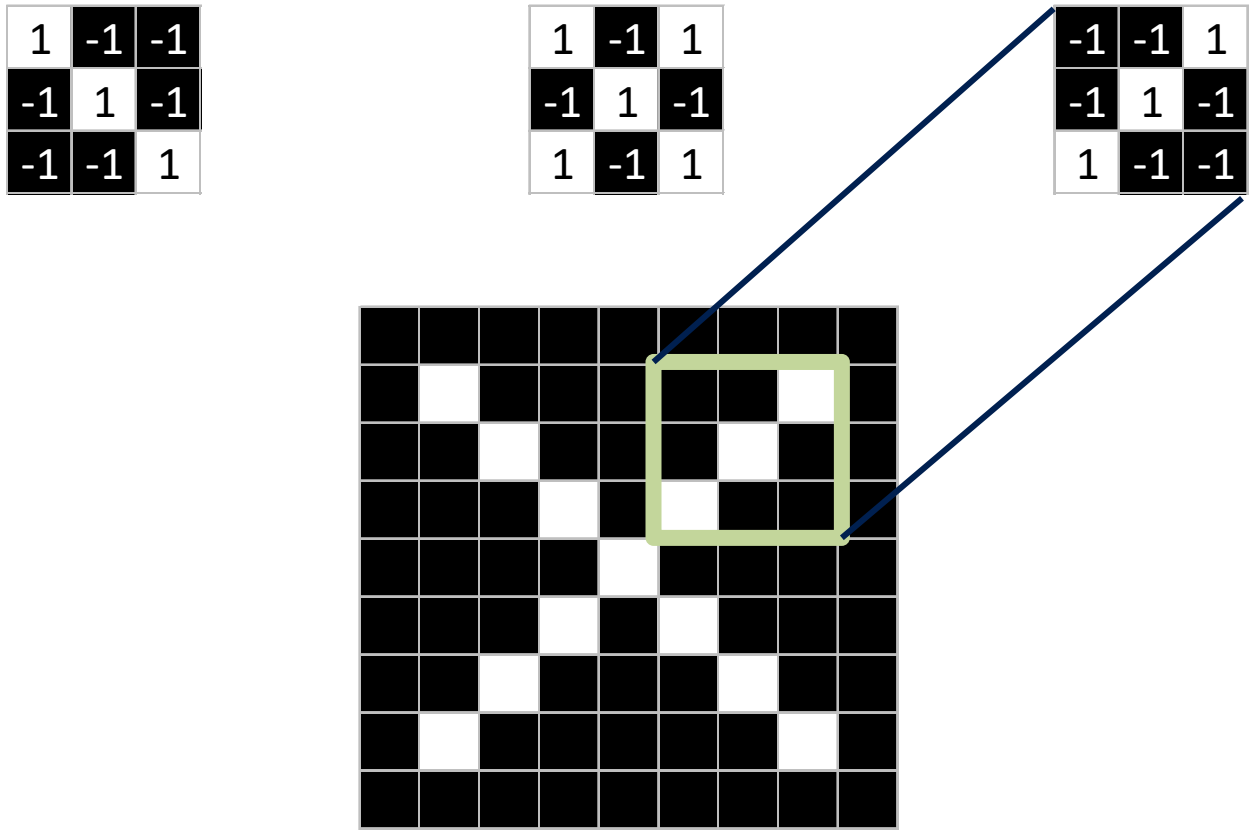
CNNs match pieces of the image



Features match pieces of the image



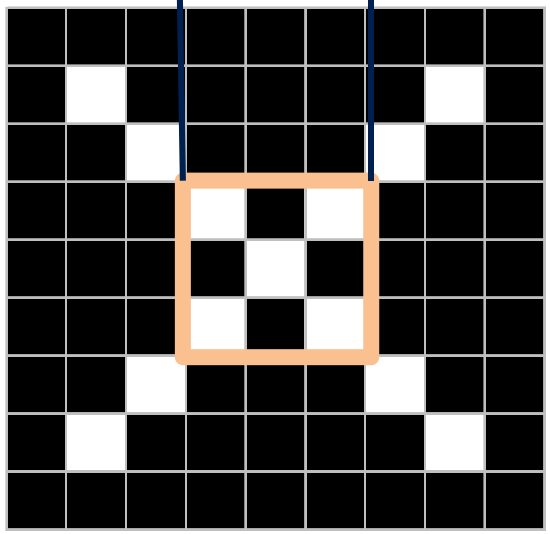

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

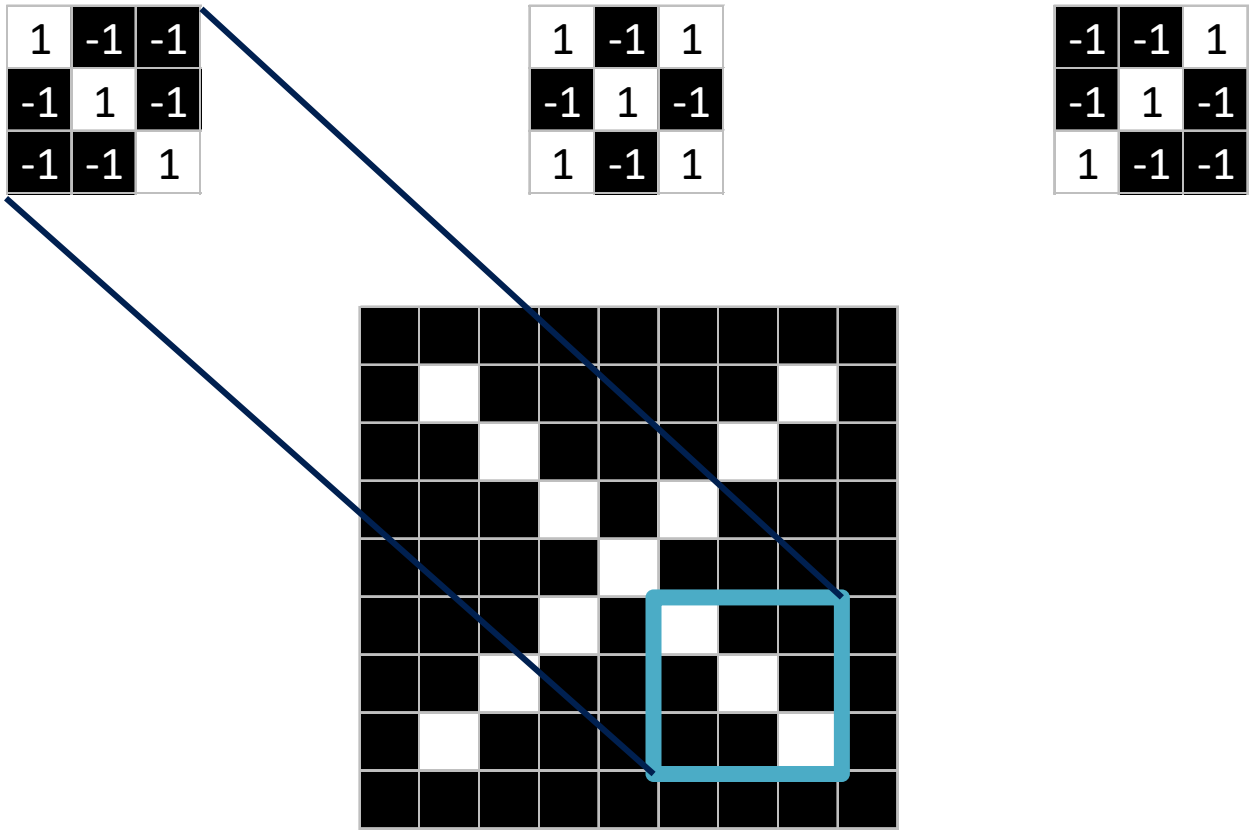

1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

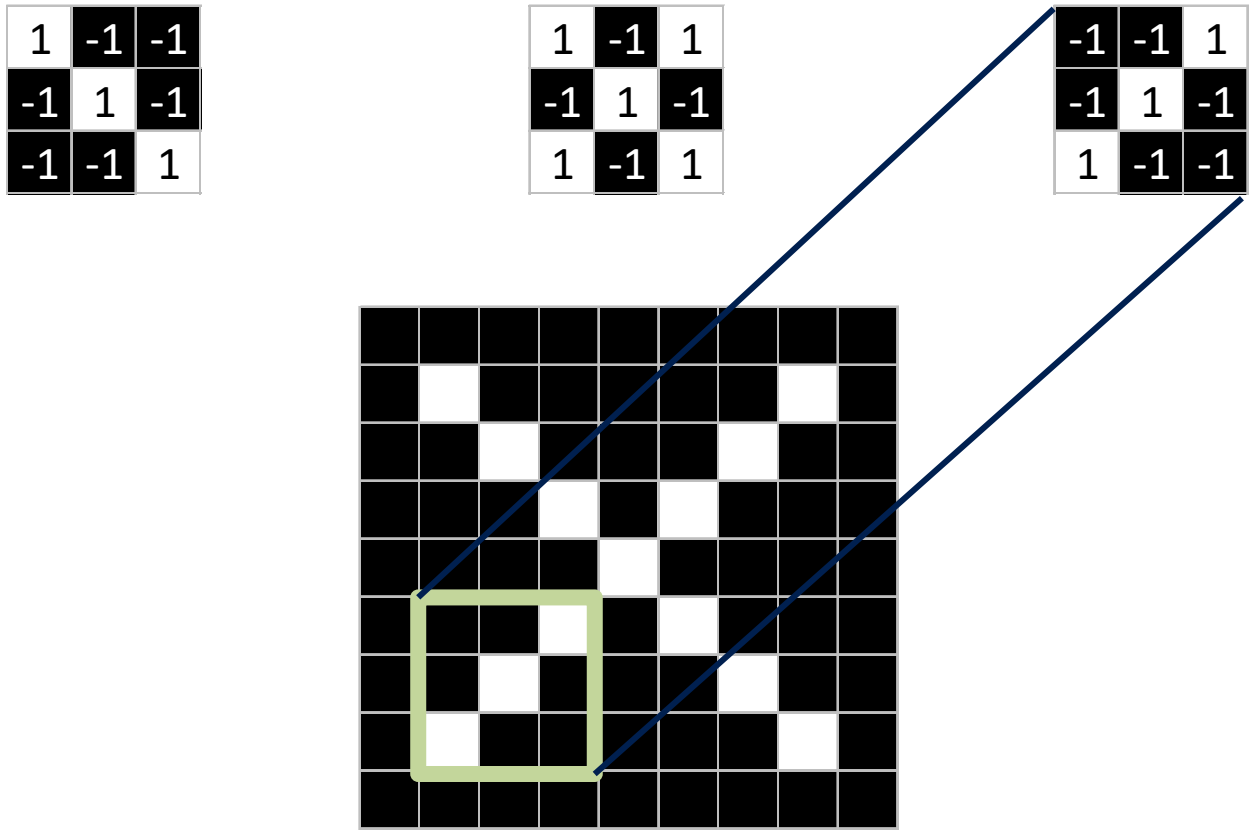
1	-1	-1
-1	1	-1
-1	-1	1

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

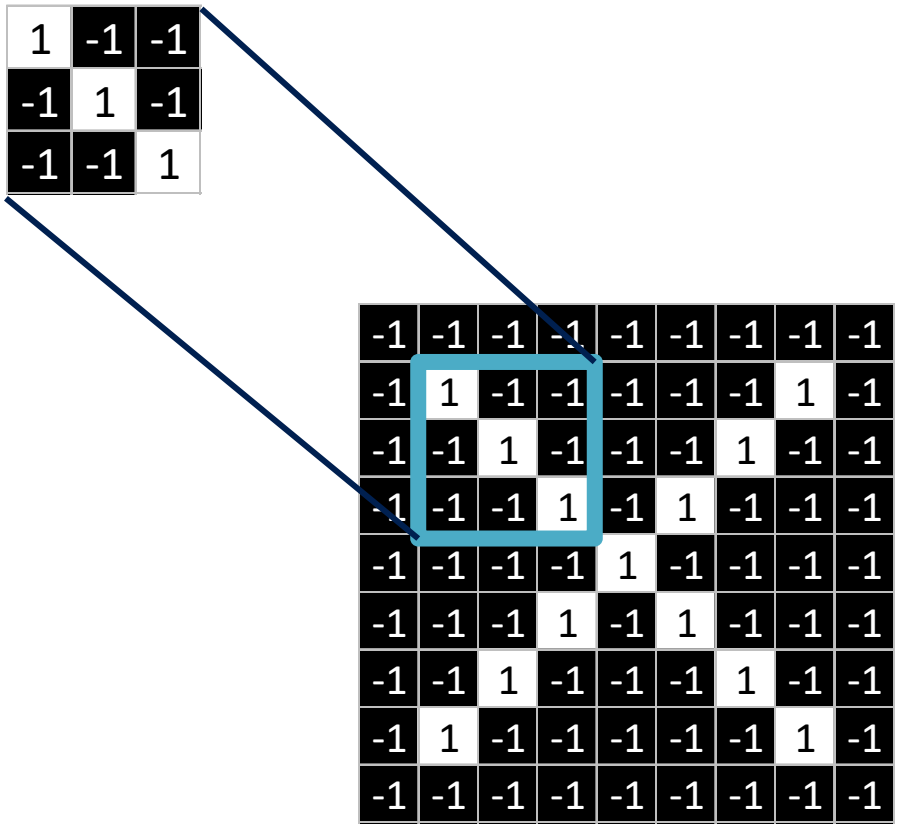
-1	-1	1
-1	1	-1
1	-1	-1




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


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

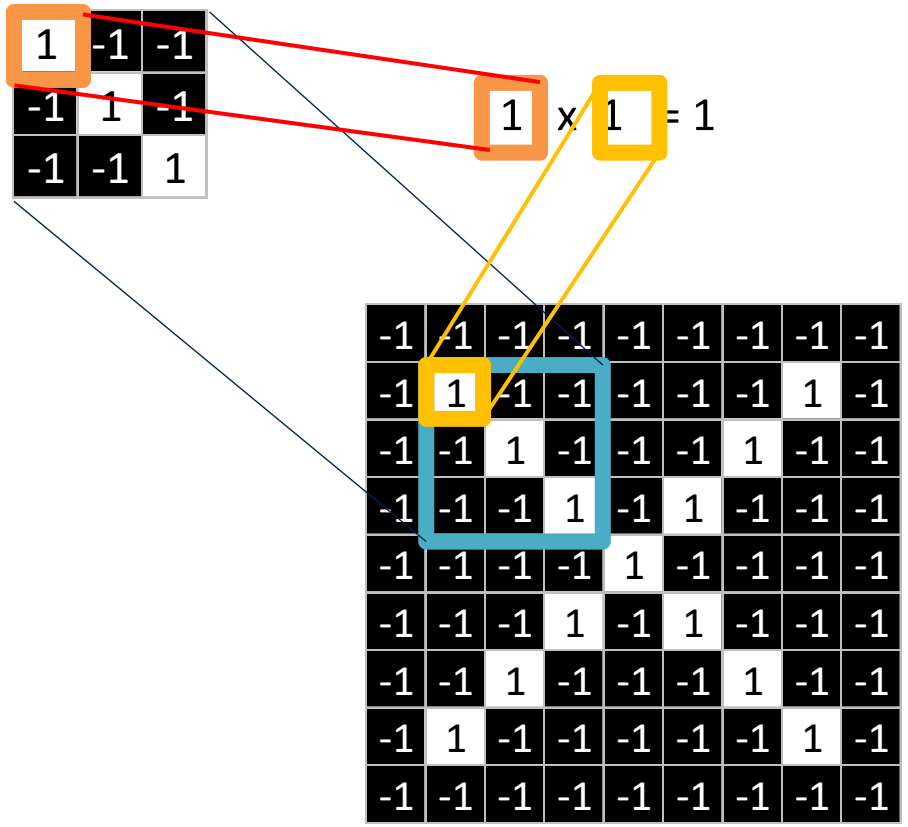
Filtering: The math behind the match



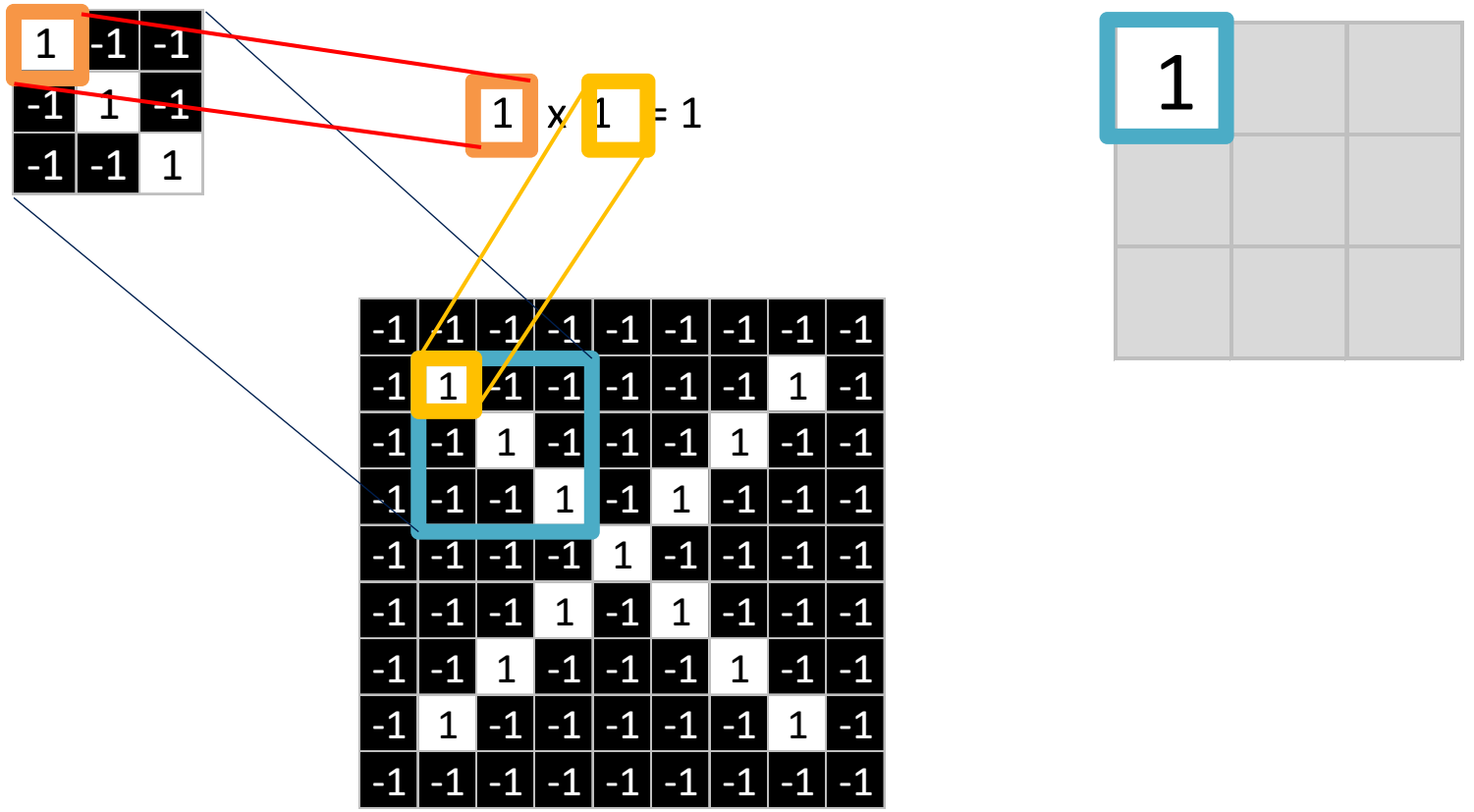
Filtering: The math behind the match

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.

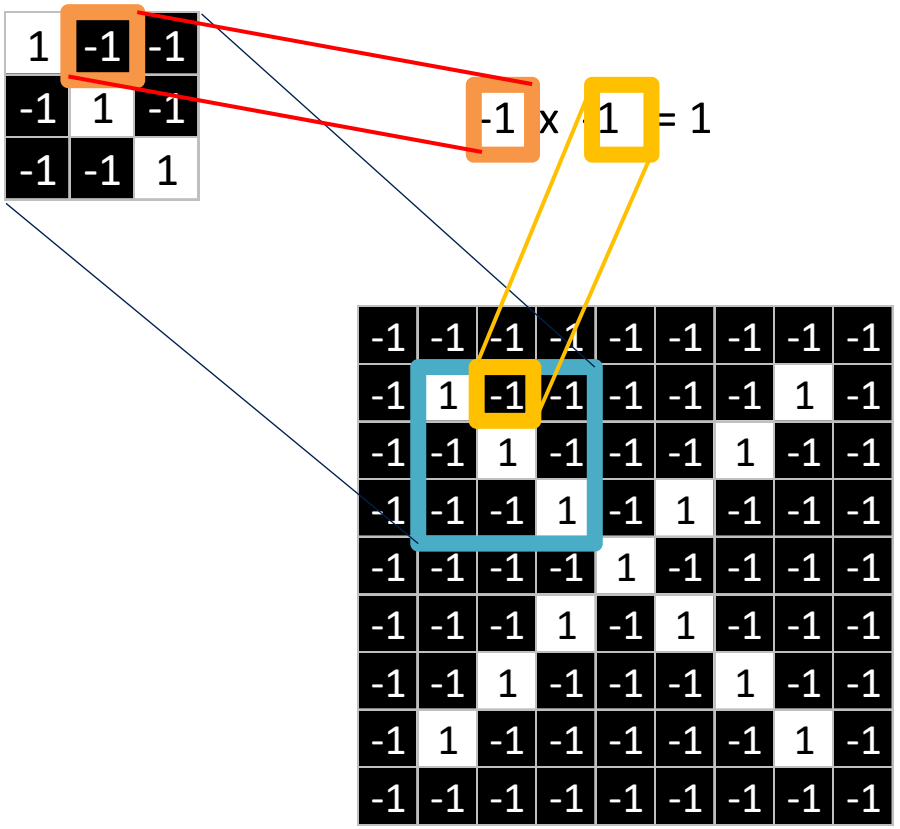
Filtering: The math behind the match

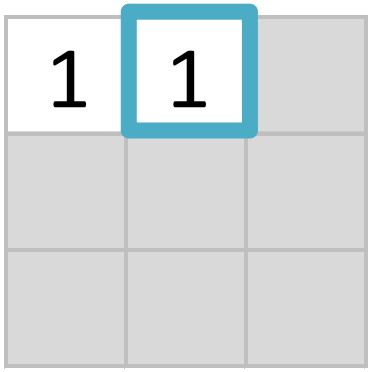


Filtering: The math behind the match

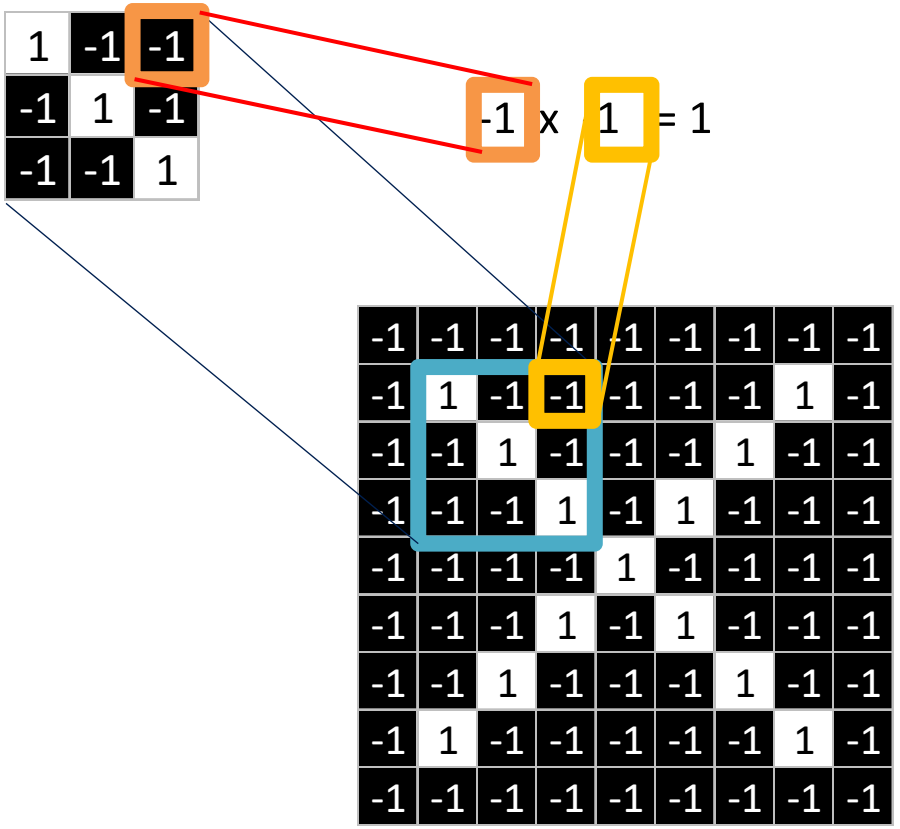


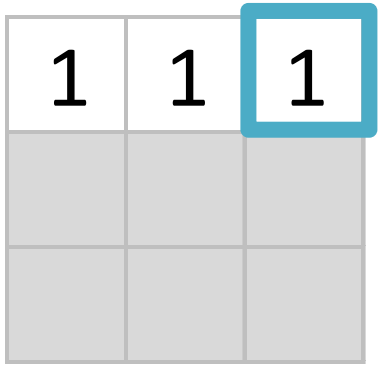
Filtering: The math behind the match



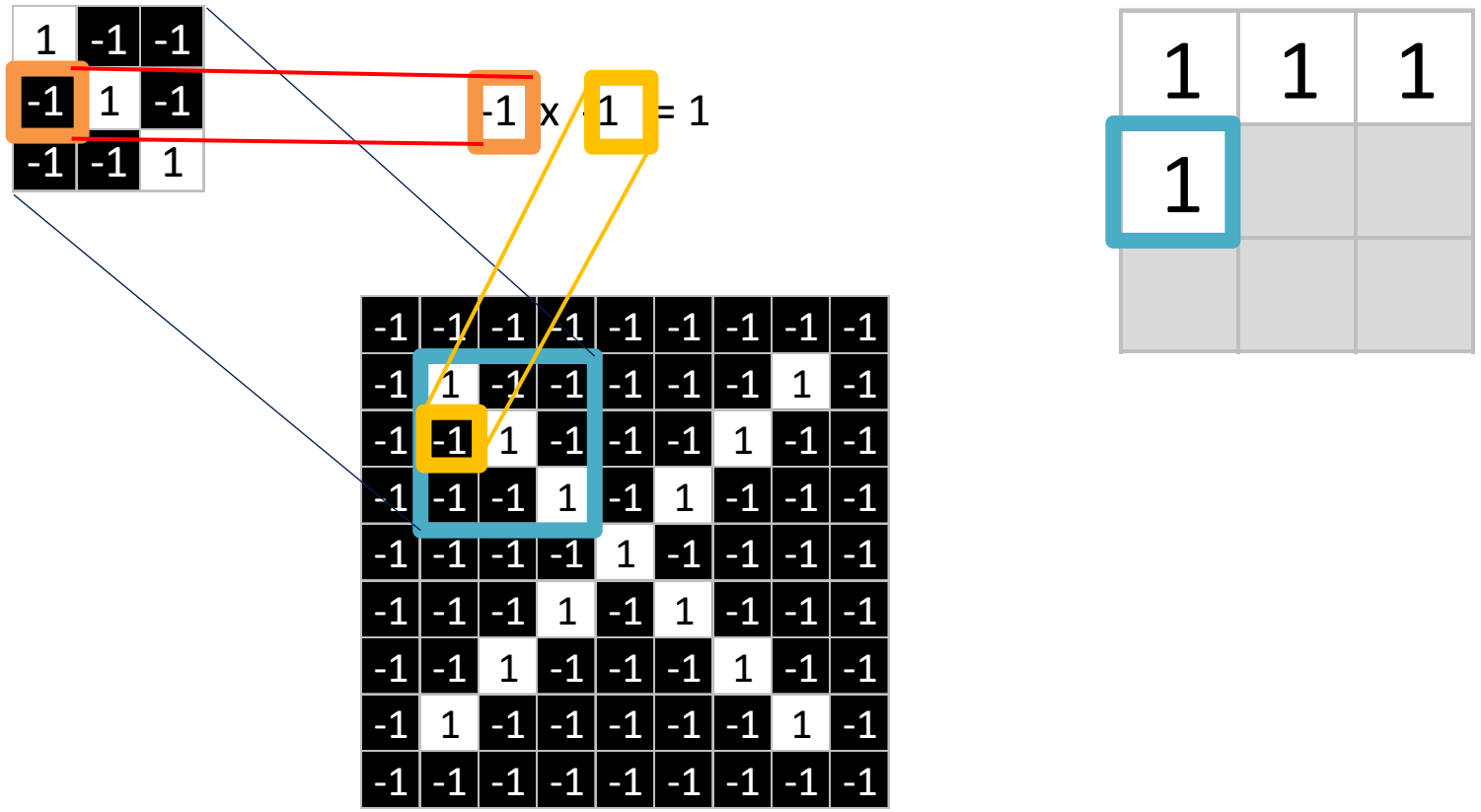


Filtering: The math behind the match

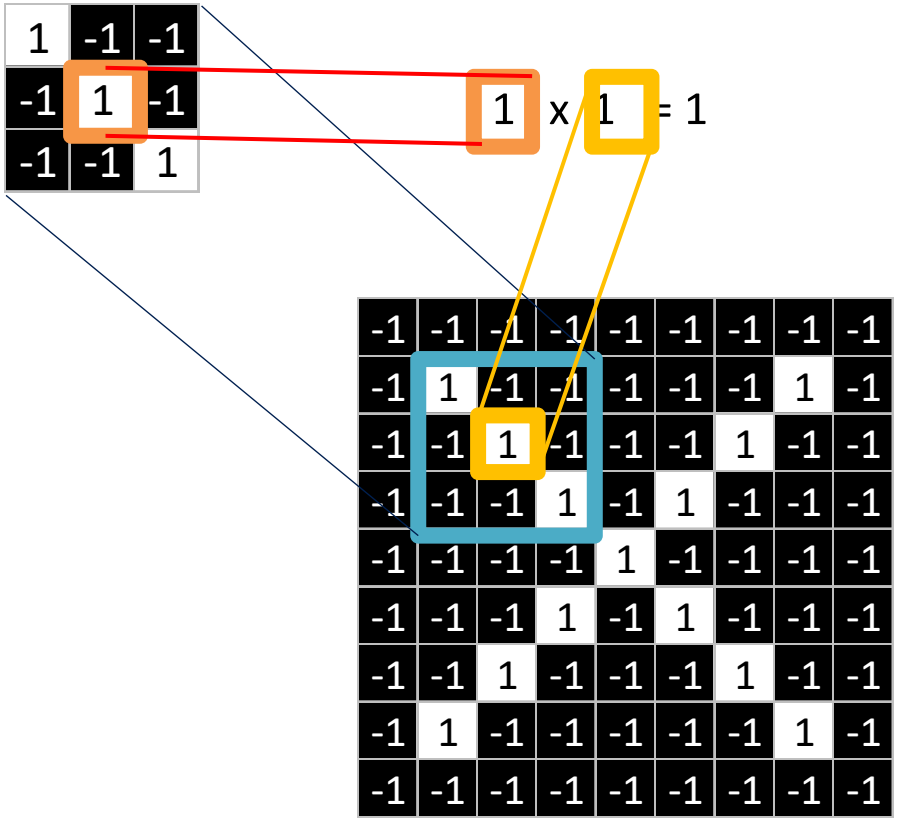


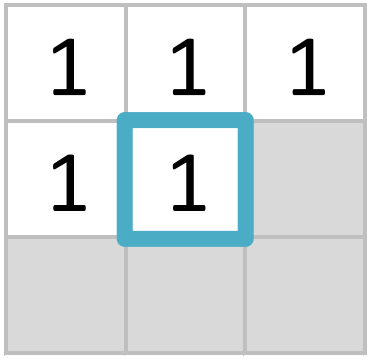


Filtering: The math behind the match

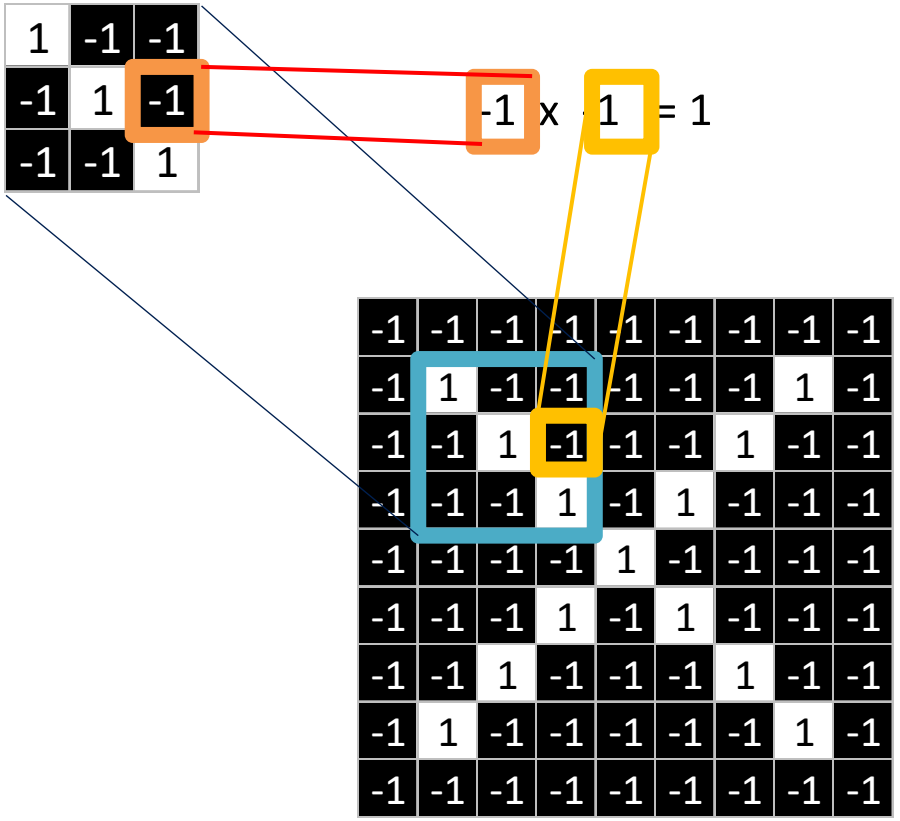


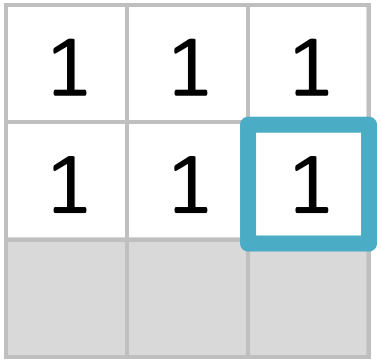
Filtering: The math behind the match



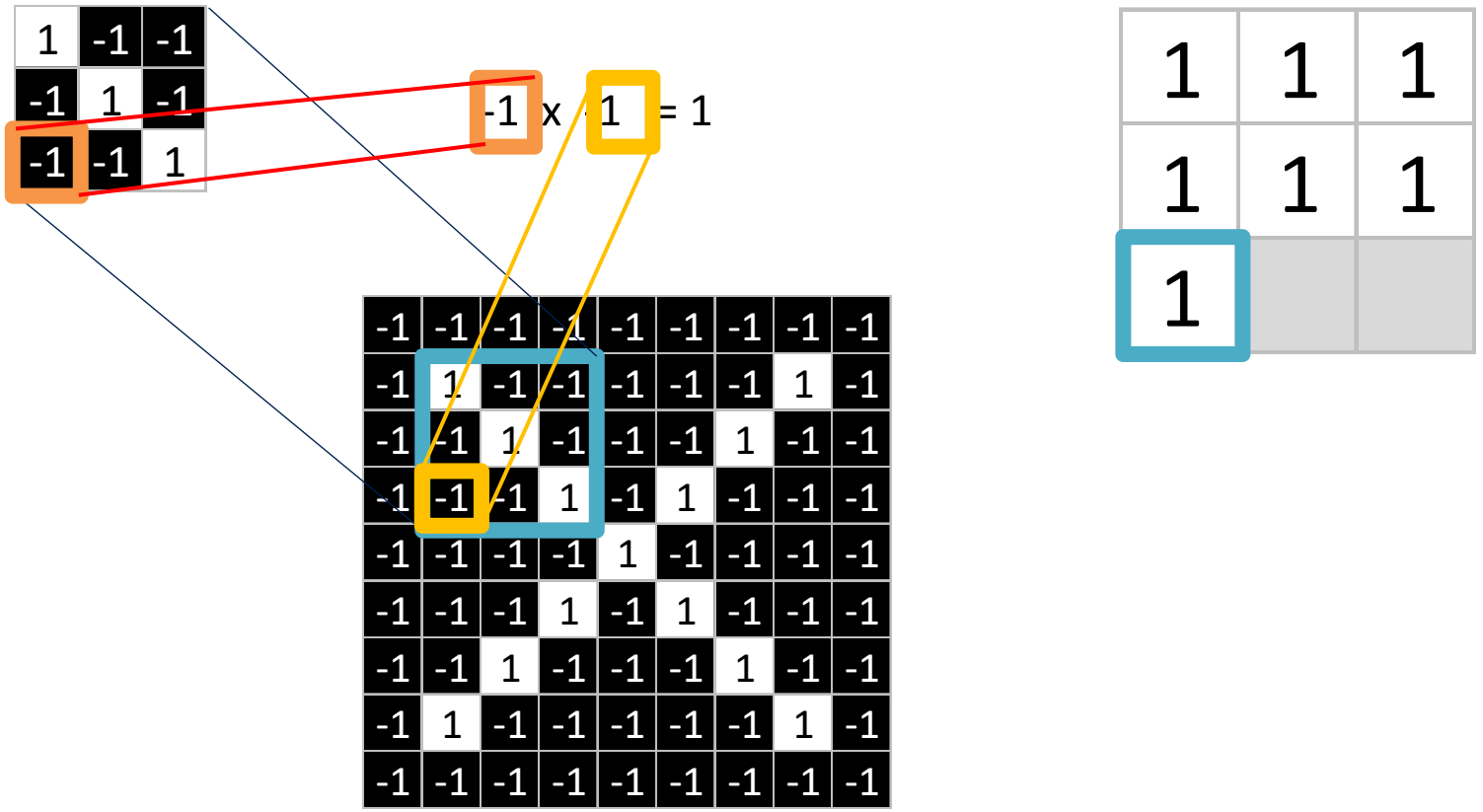


Filtering: The math behind the match

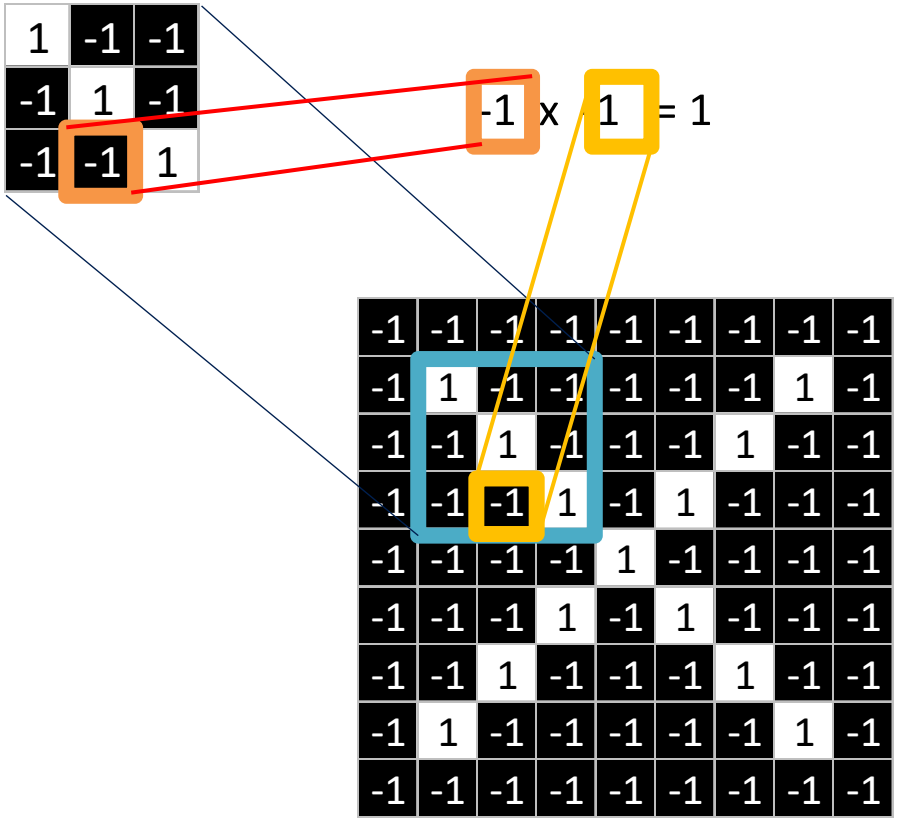


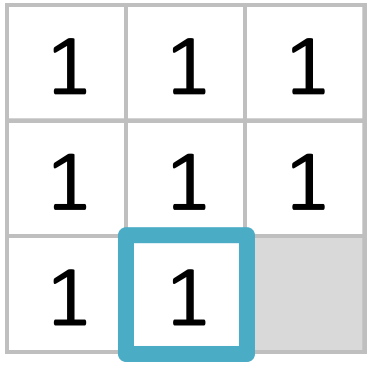


Filtering: The math behind the match

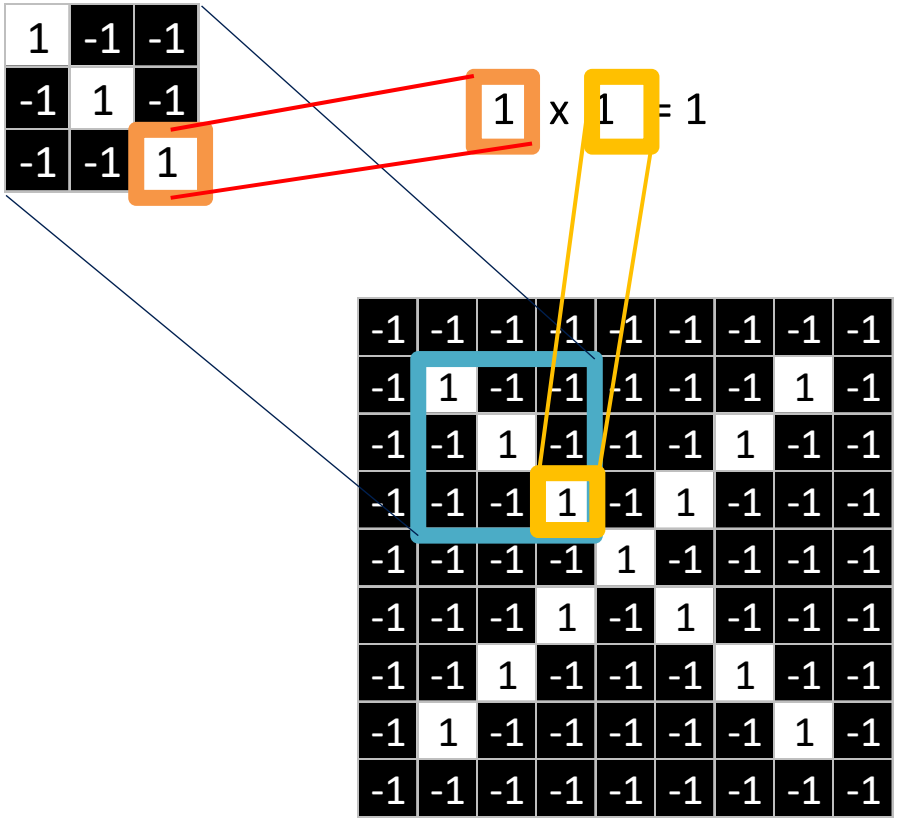


Filtering: The math behind the match



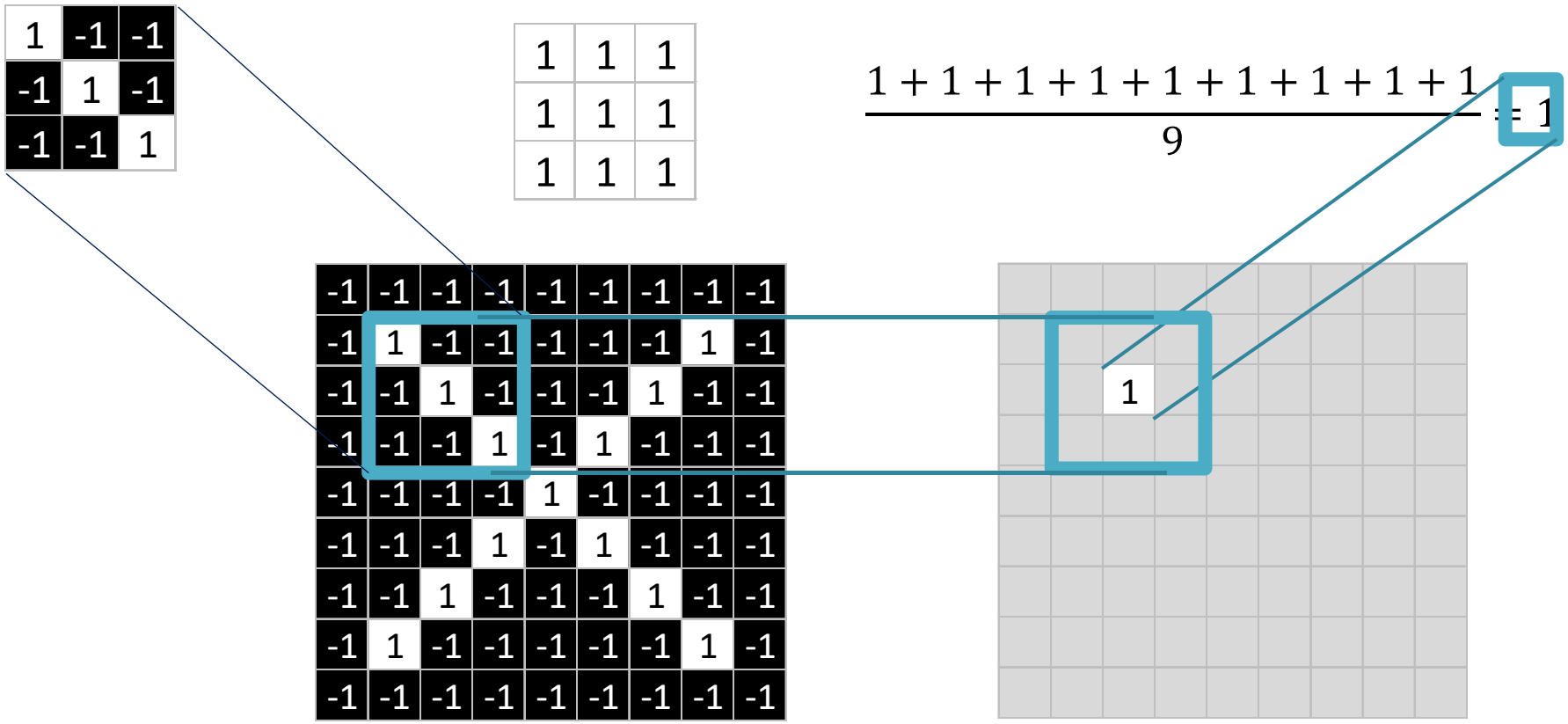


Filtering: The math behind the match

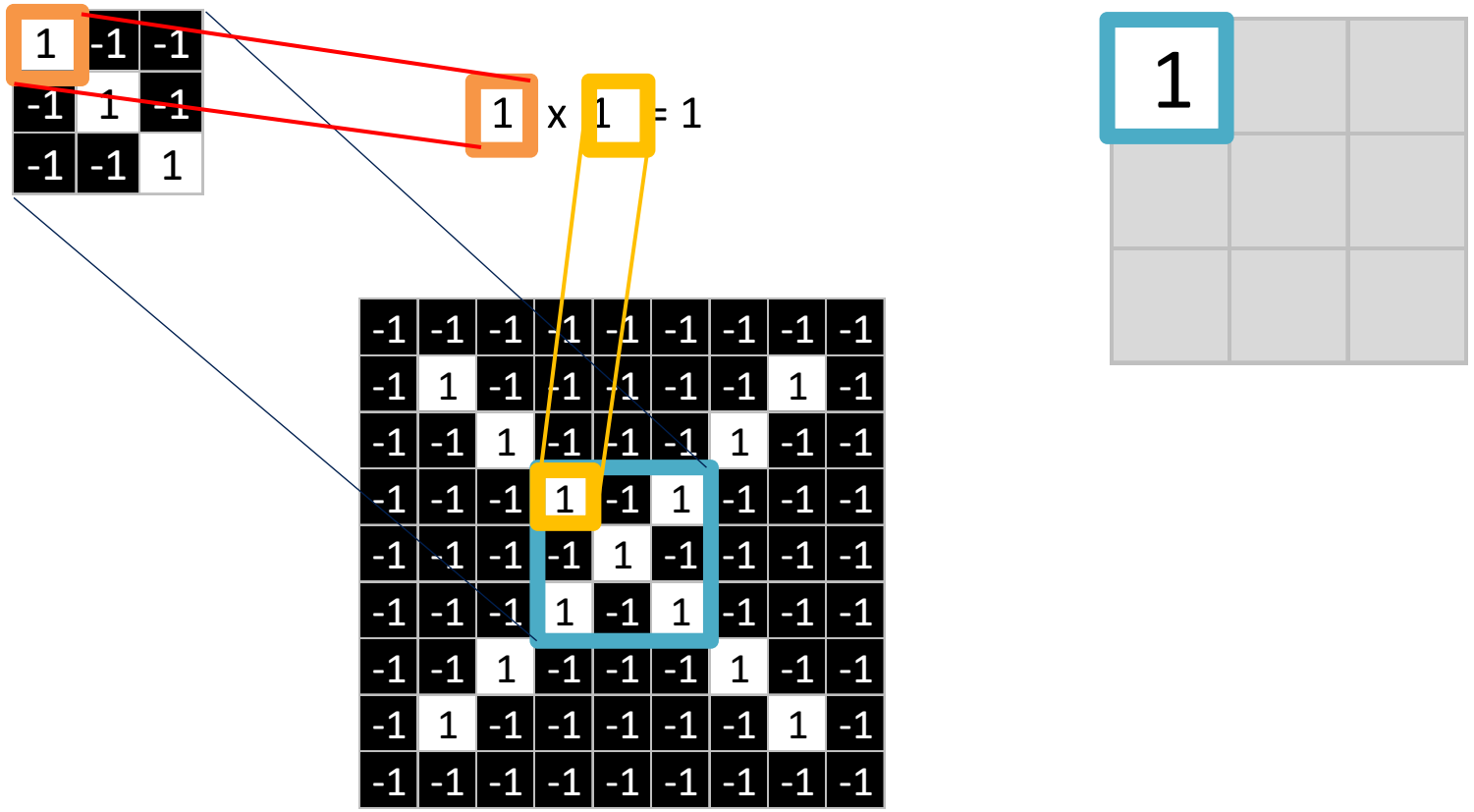


1	1	1
1	1	1
1	1	1

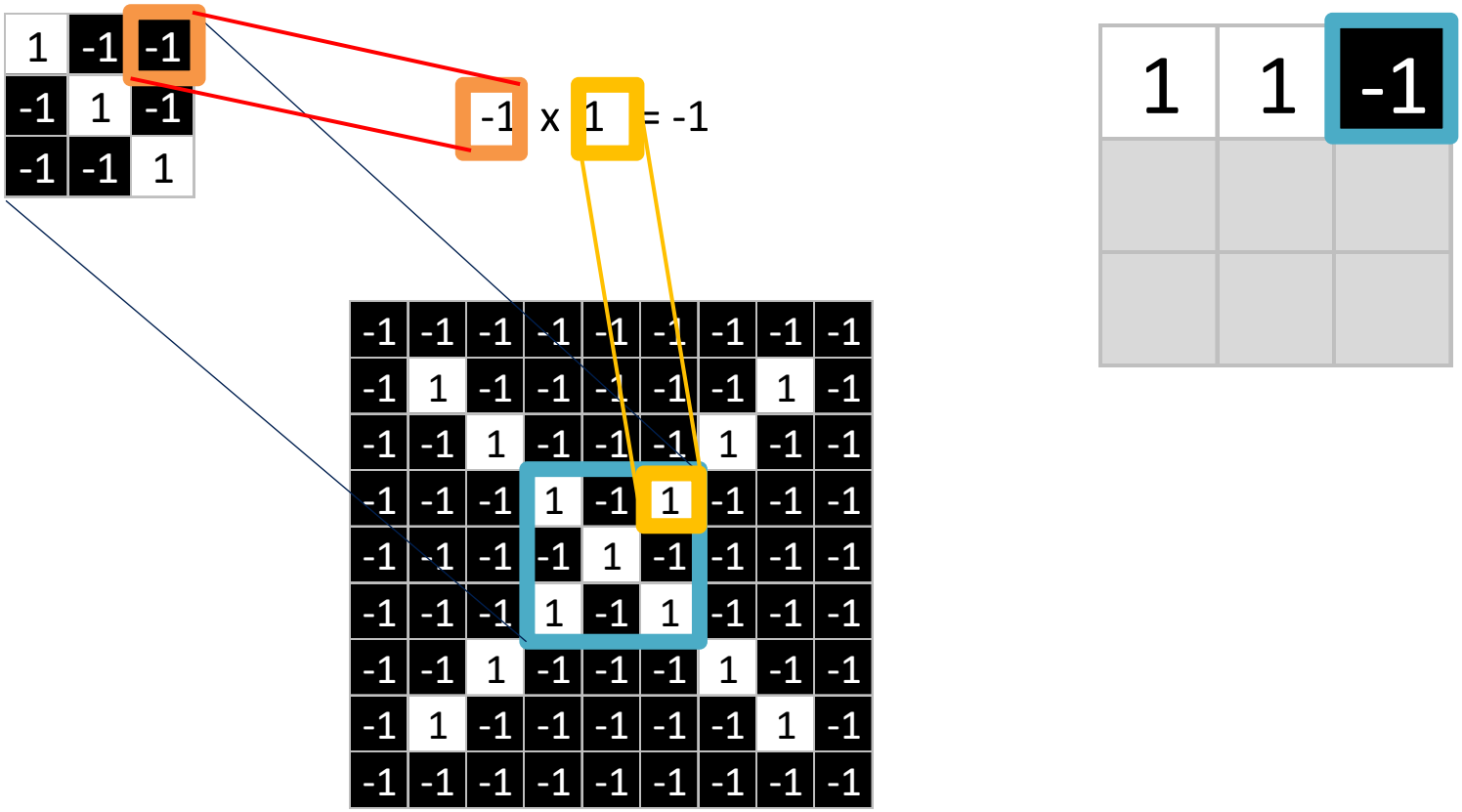
Filtering: The math behind the match



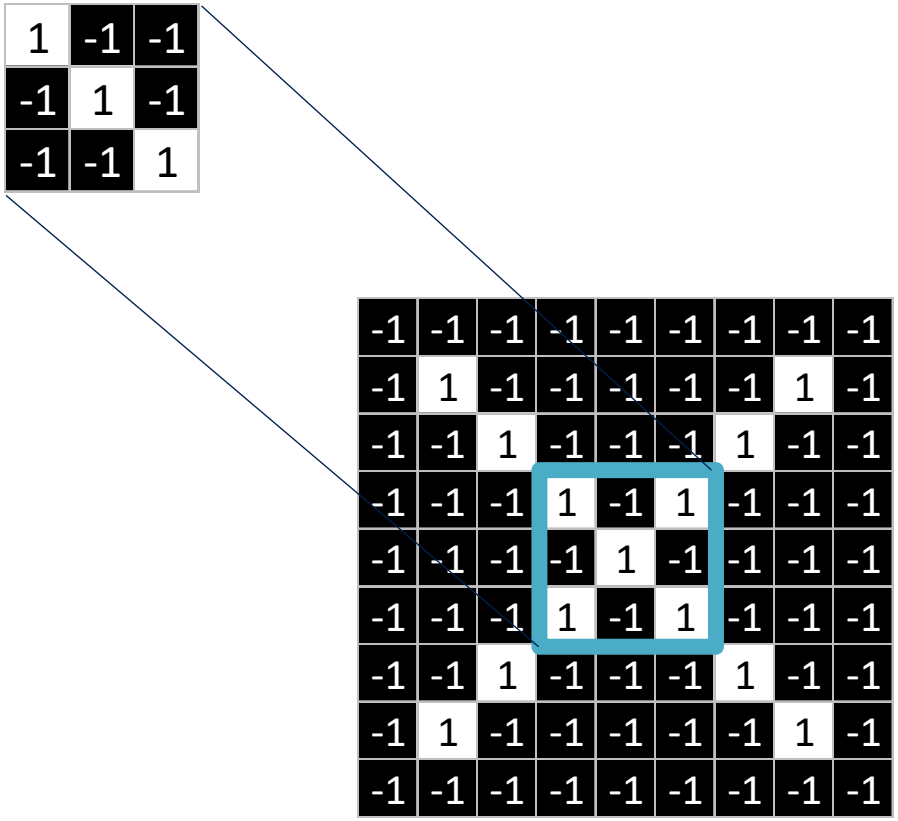
Filtering: The math behind the match



Filtering: The math behind the match

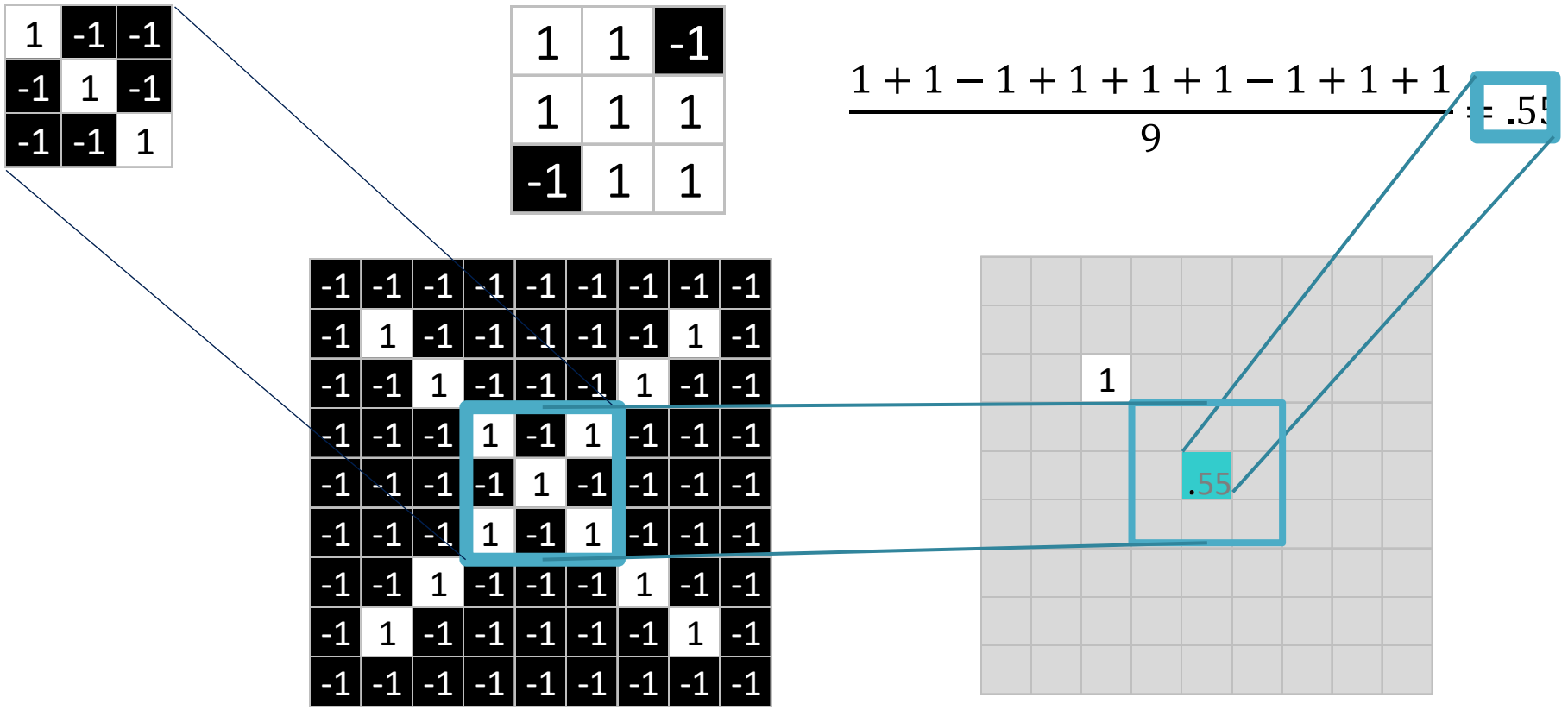


Filtering: The math behind the match



1	1	-1
1	1	1
-1	1	1

Filtering: The math behind the match



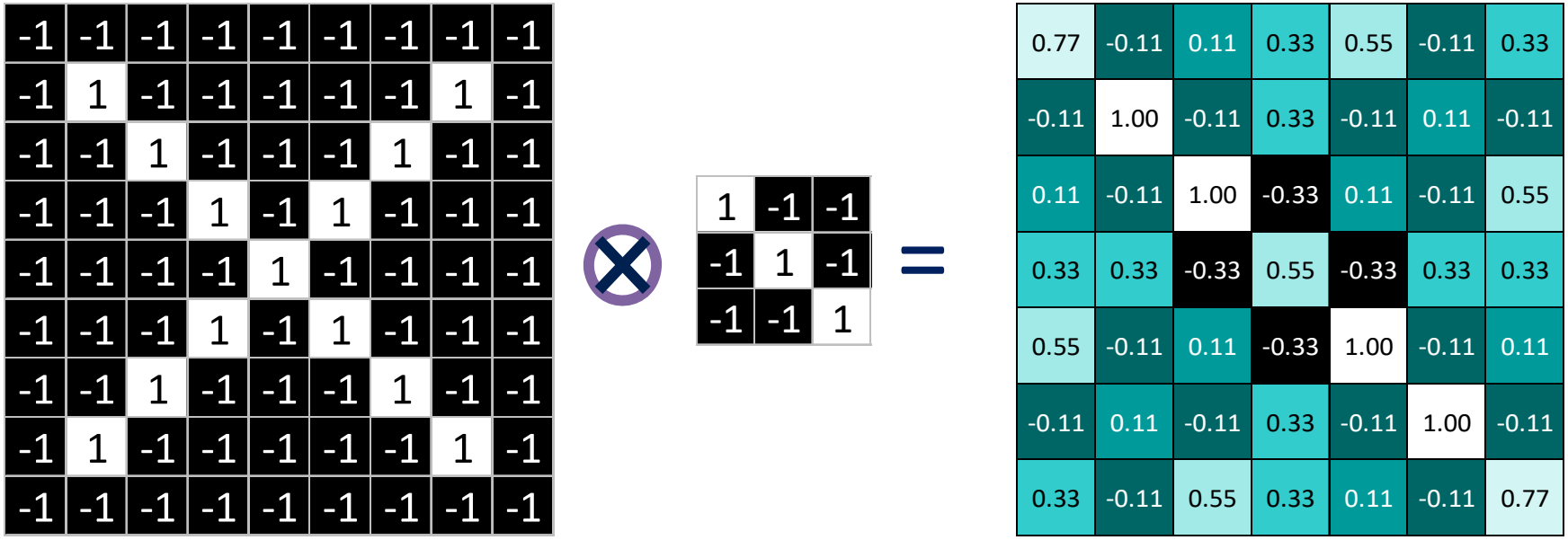
Convolution: Trying every possible match

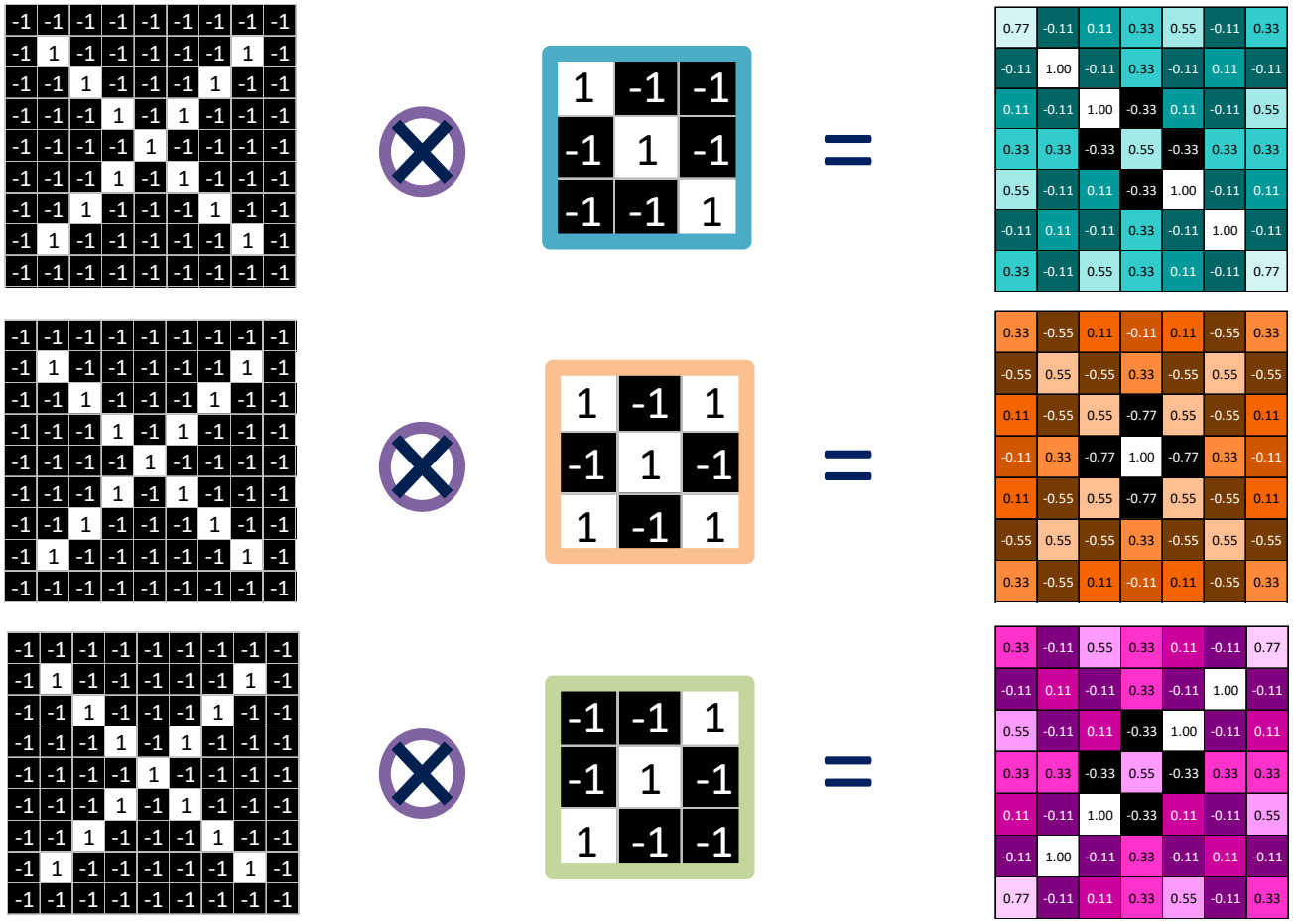
1	-1	-1
-1	1	-1
-1	-1	1

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

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

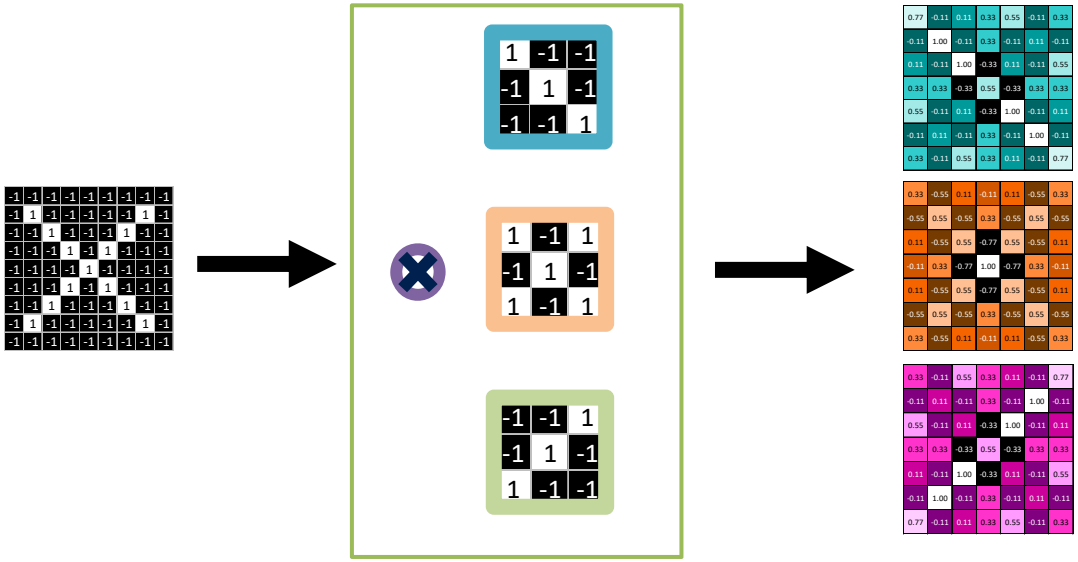
Convolution: Trying every possible match




-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 1
-1 1 -1
1 -1 1
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
-0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 1
-1 1 -1
1 -1 -1
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33

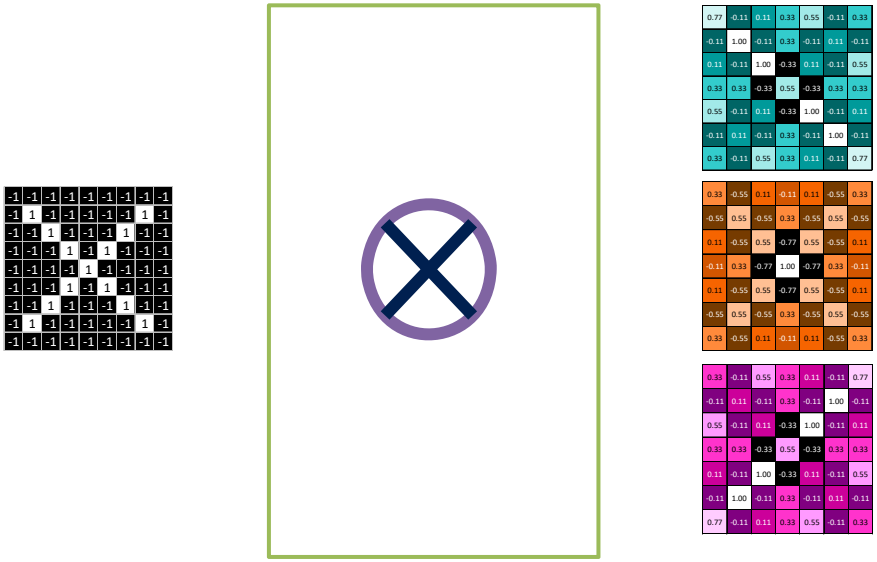
Convolution layer

One image becomes a stack of filtered images



Convolution layer

One image becomes a stack of filtered images



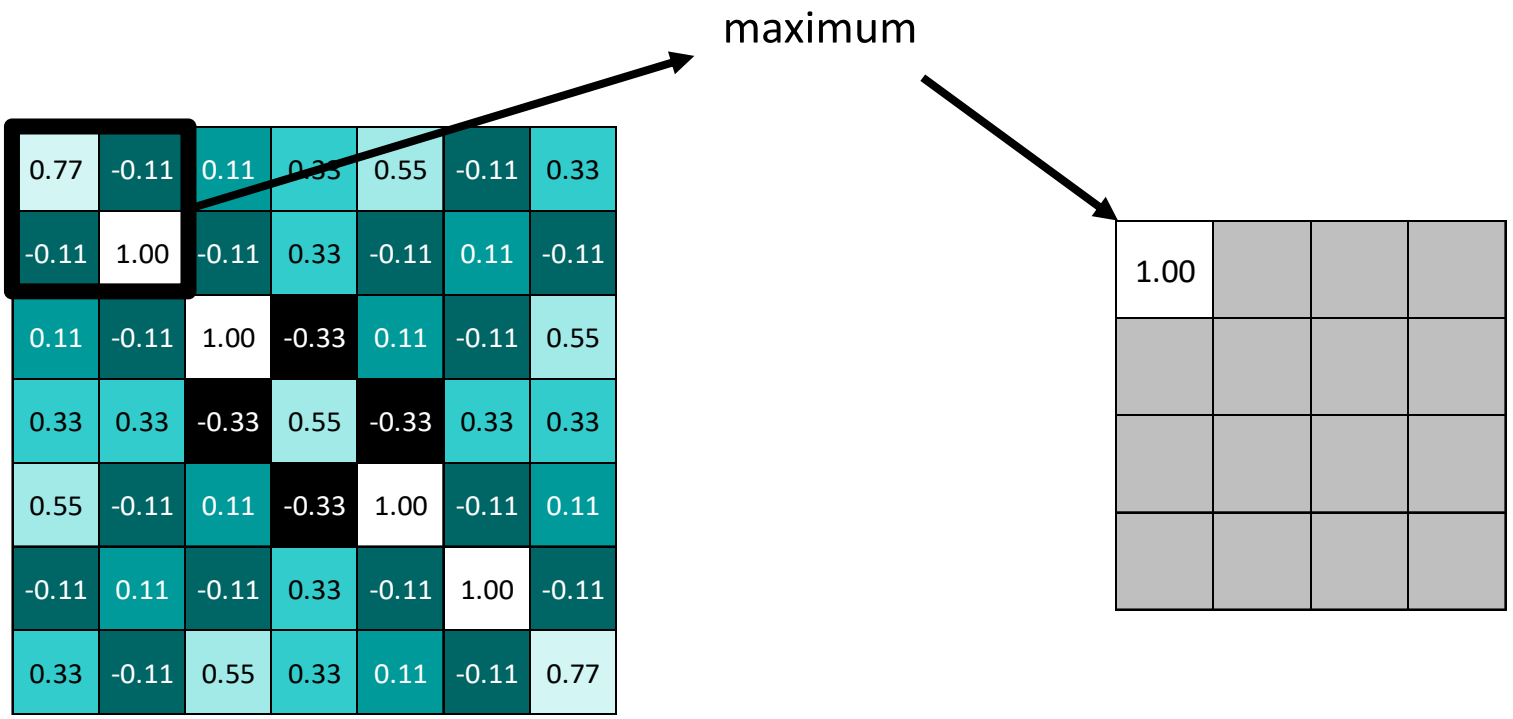
Pooling: Shrinking the image stack

Achieves a more abstract representation

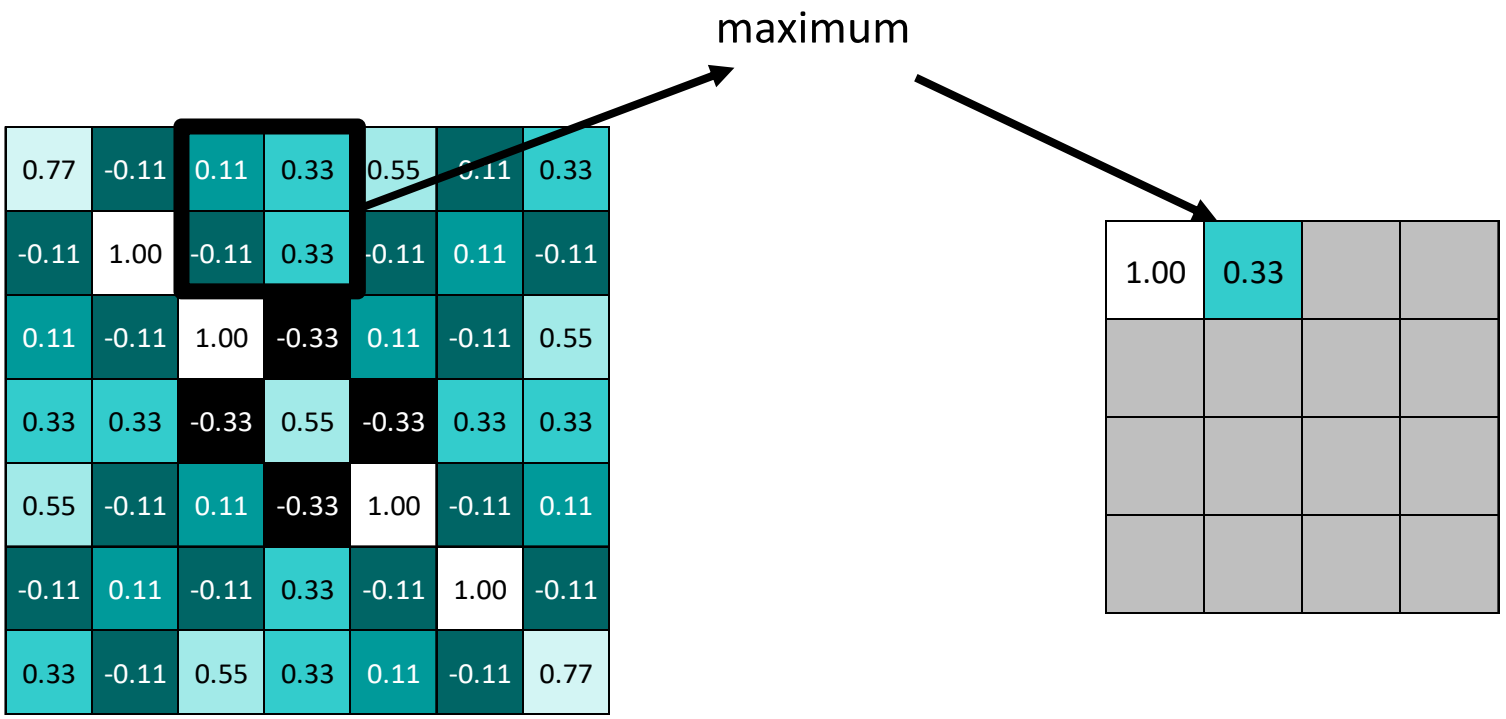
Improves invariance to geometric transformation

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

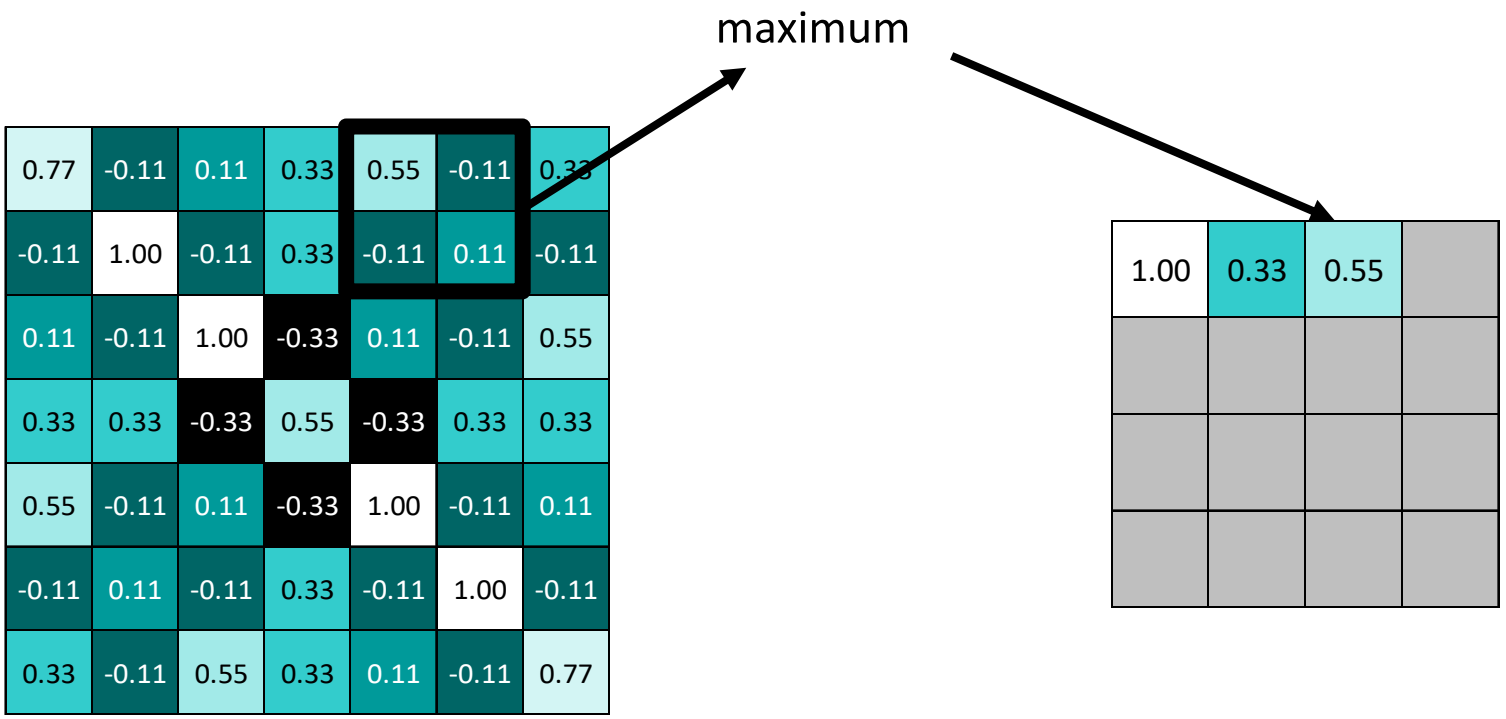
Pooling


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

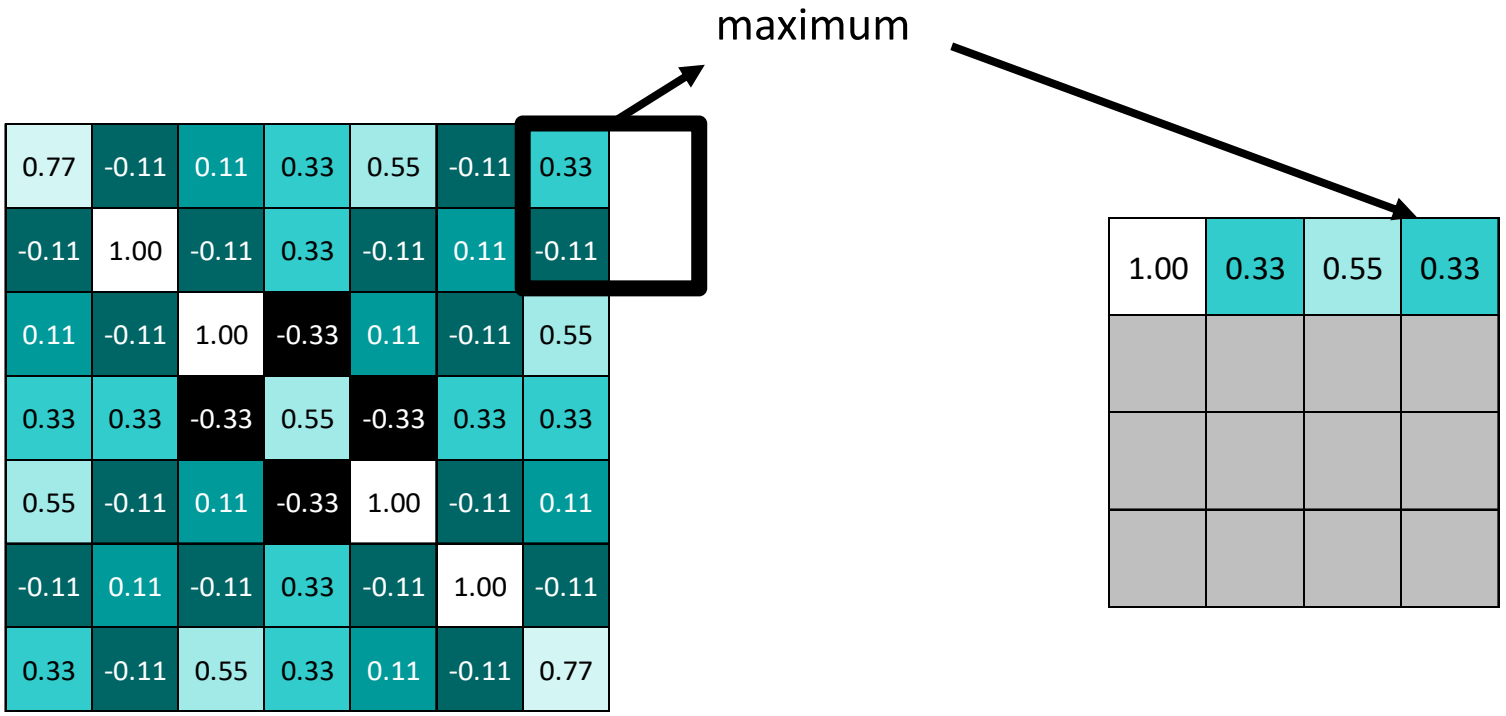
Pooling


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

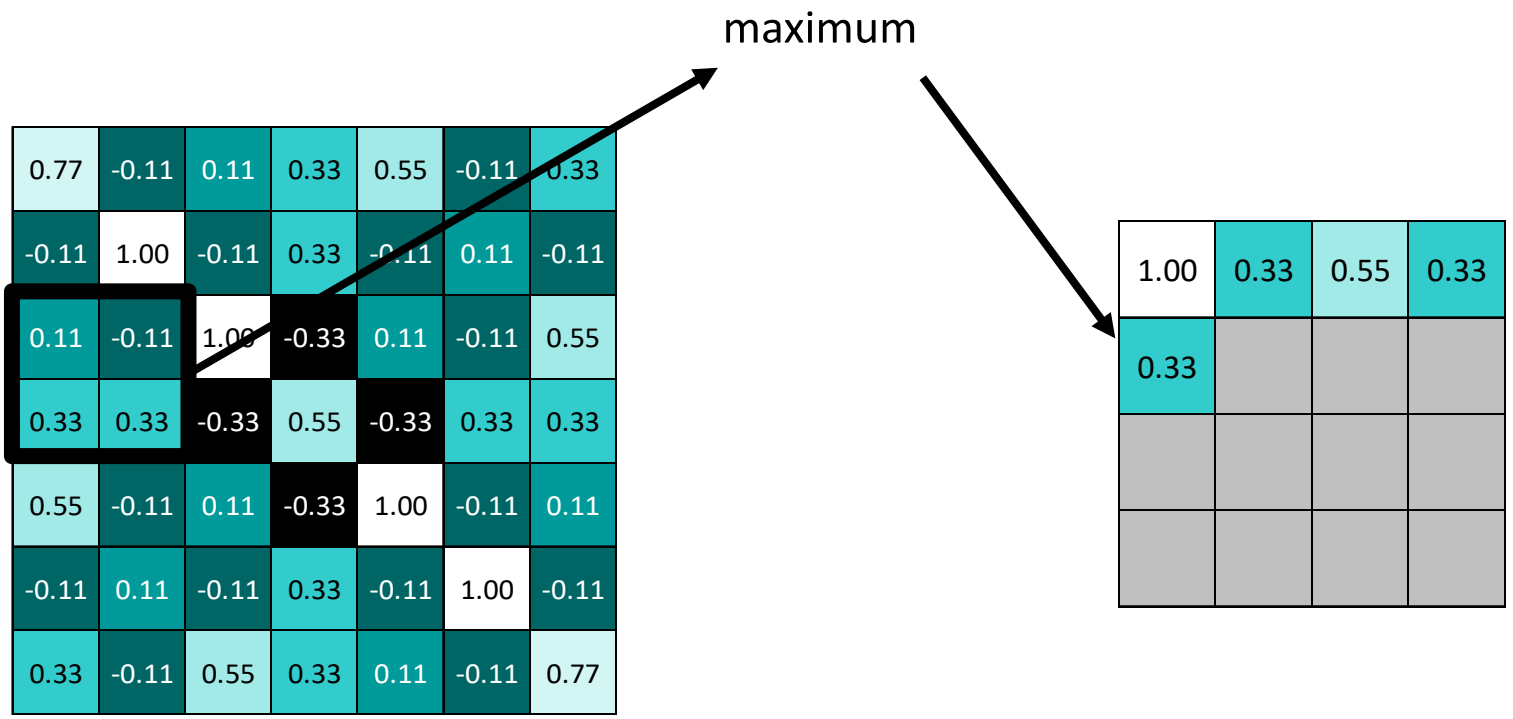
Pooling


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

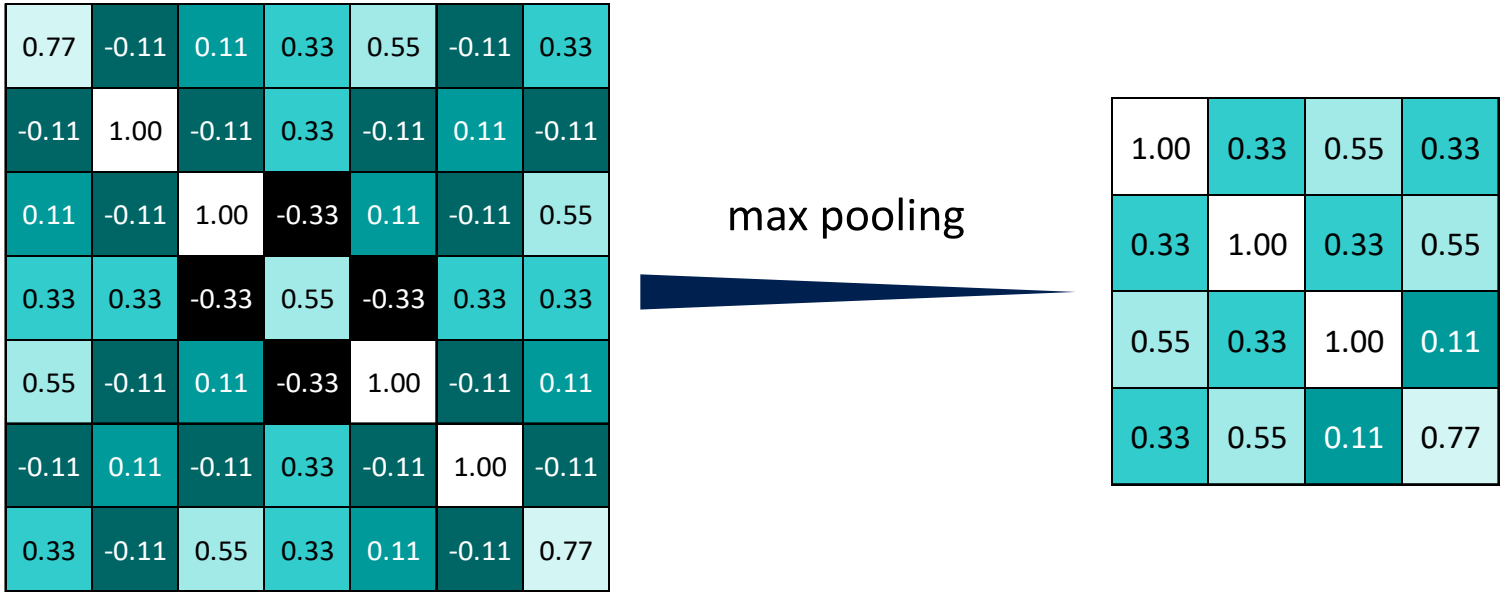
Pooling

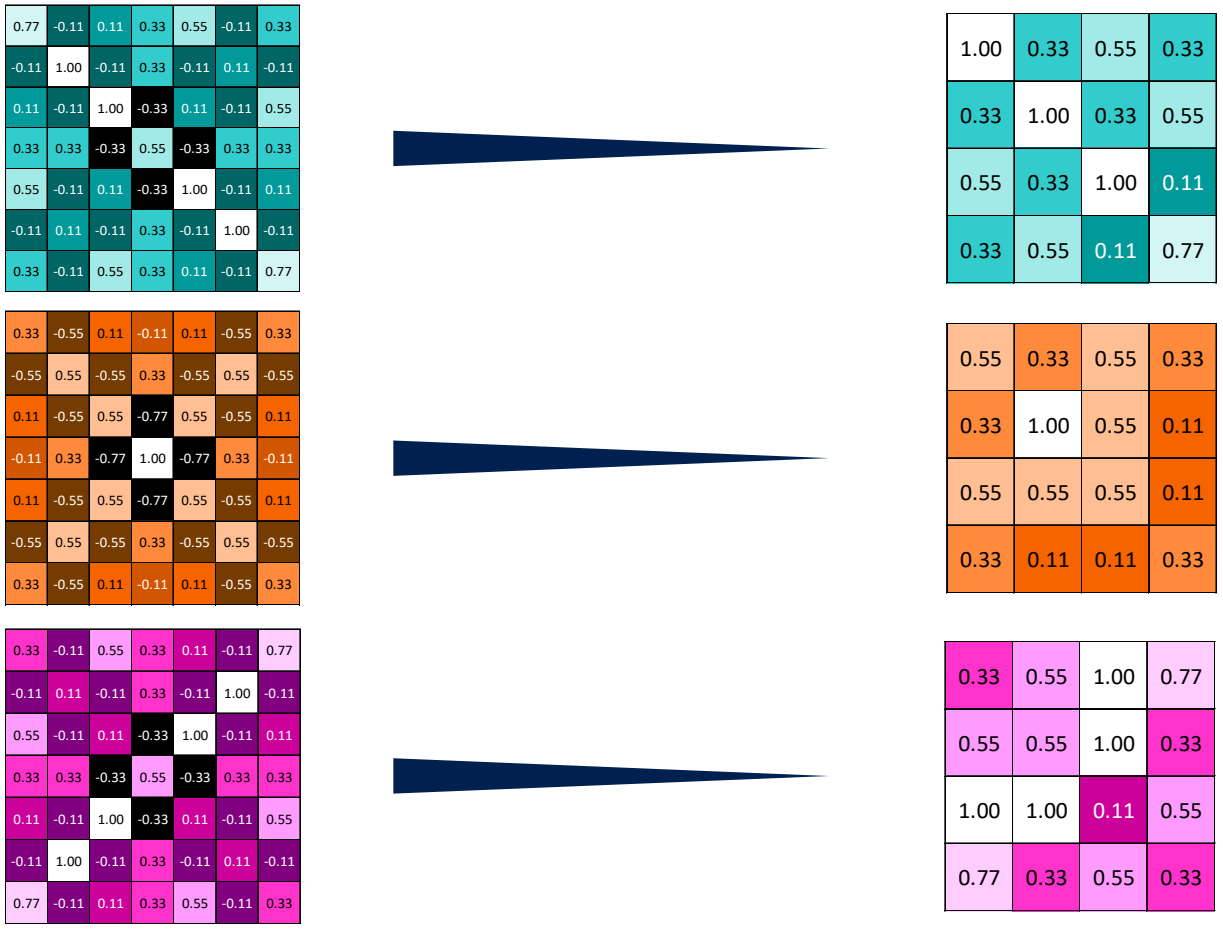

maximum
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
1.00 0.33 0.55 0.33

Pooling


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

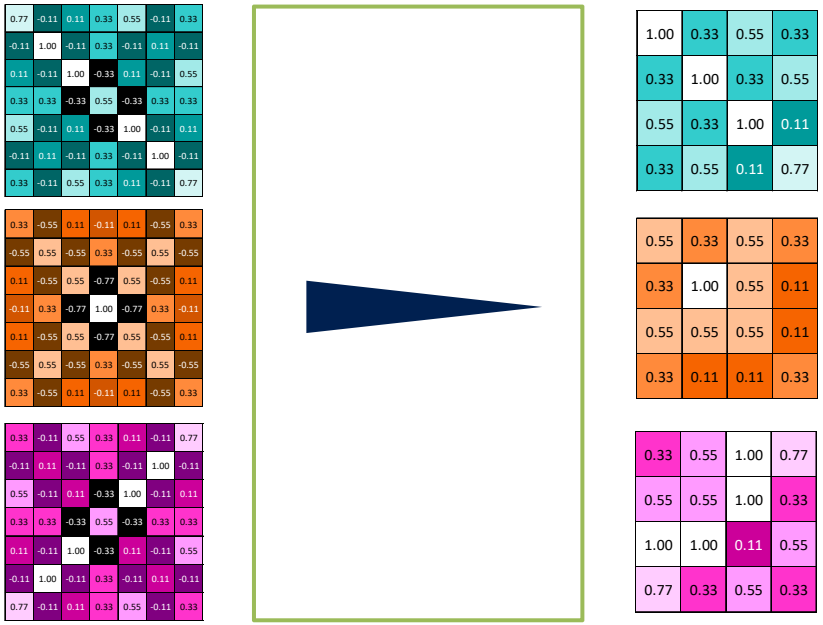
Pooling


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
max pooling
1.00 0.33 0.55 0.33
0.33 1.00 0.33 0.55
0.55 0.33 1.00 0.11
0.33 0.55 0.11 0.77


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

Pooling layer

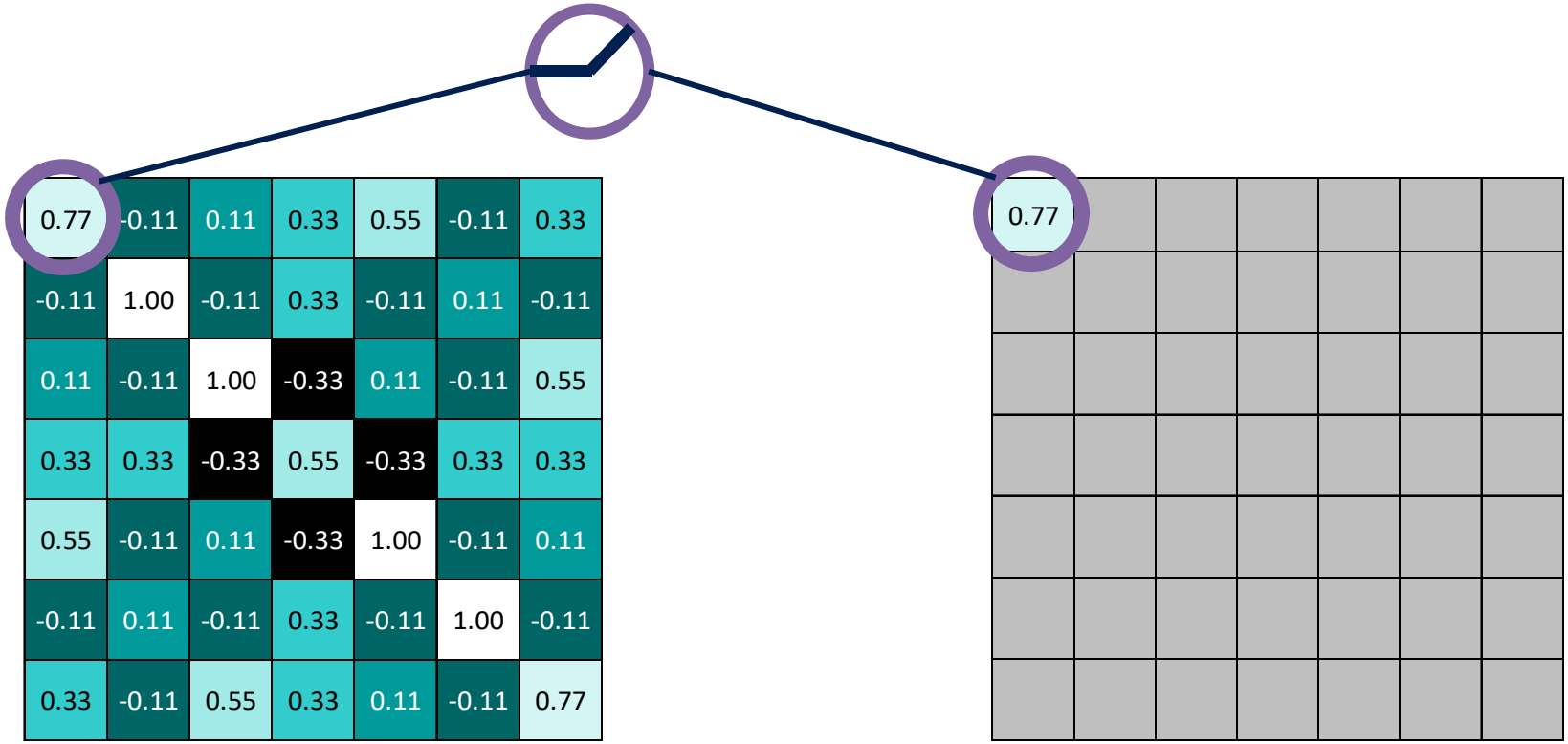
A stack of images becomes a stack of smaller images.



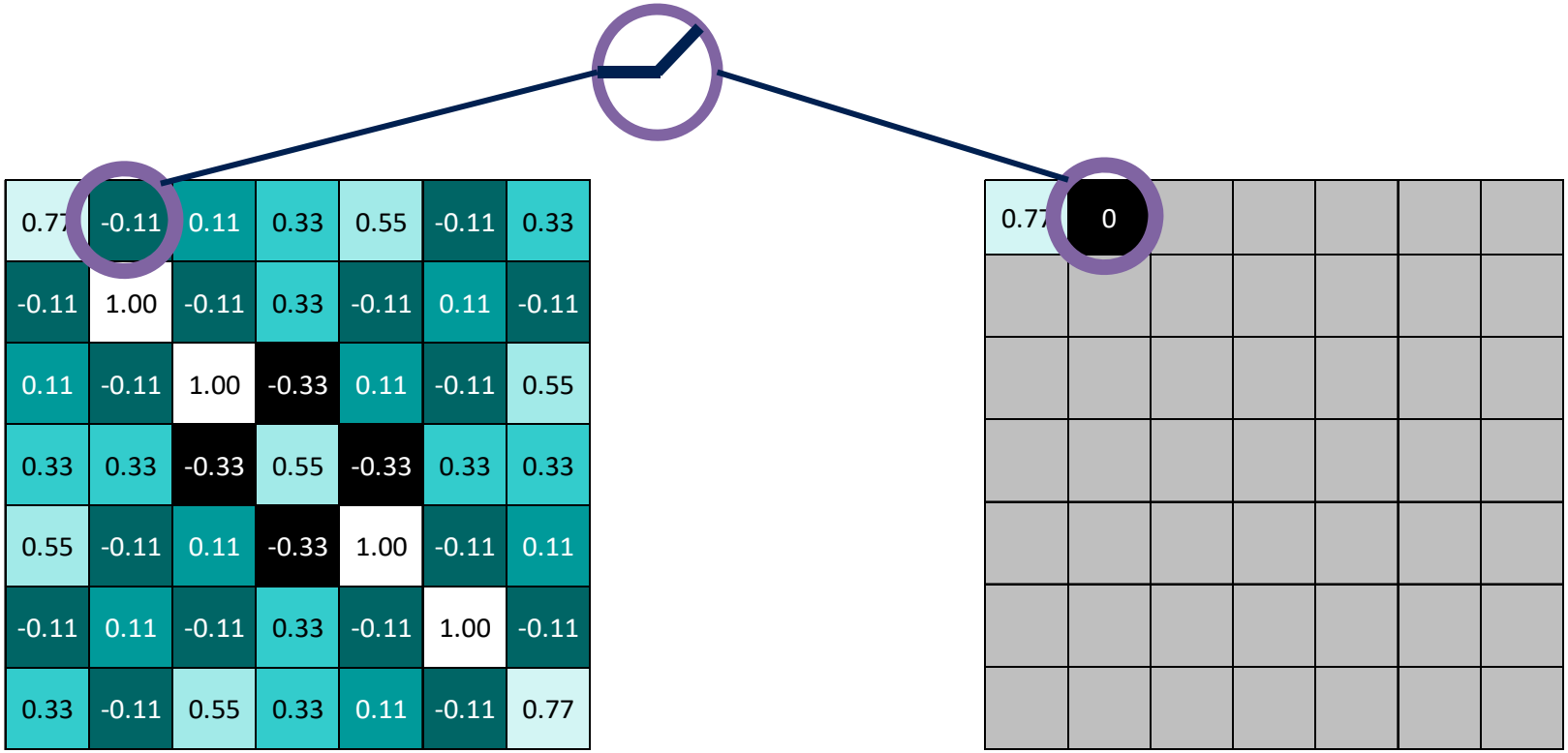
Normalization

- Keep the math from breaking by tweaking each of the values just a bit.
- Change everything negative to zero.

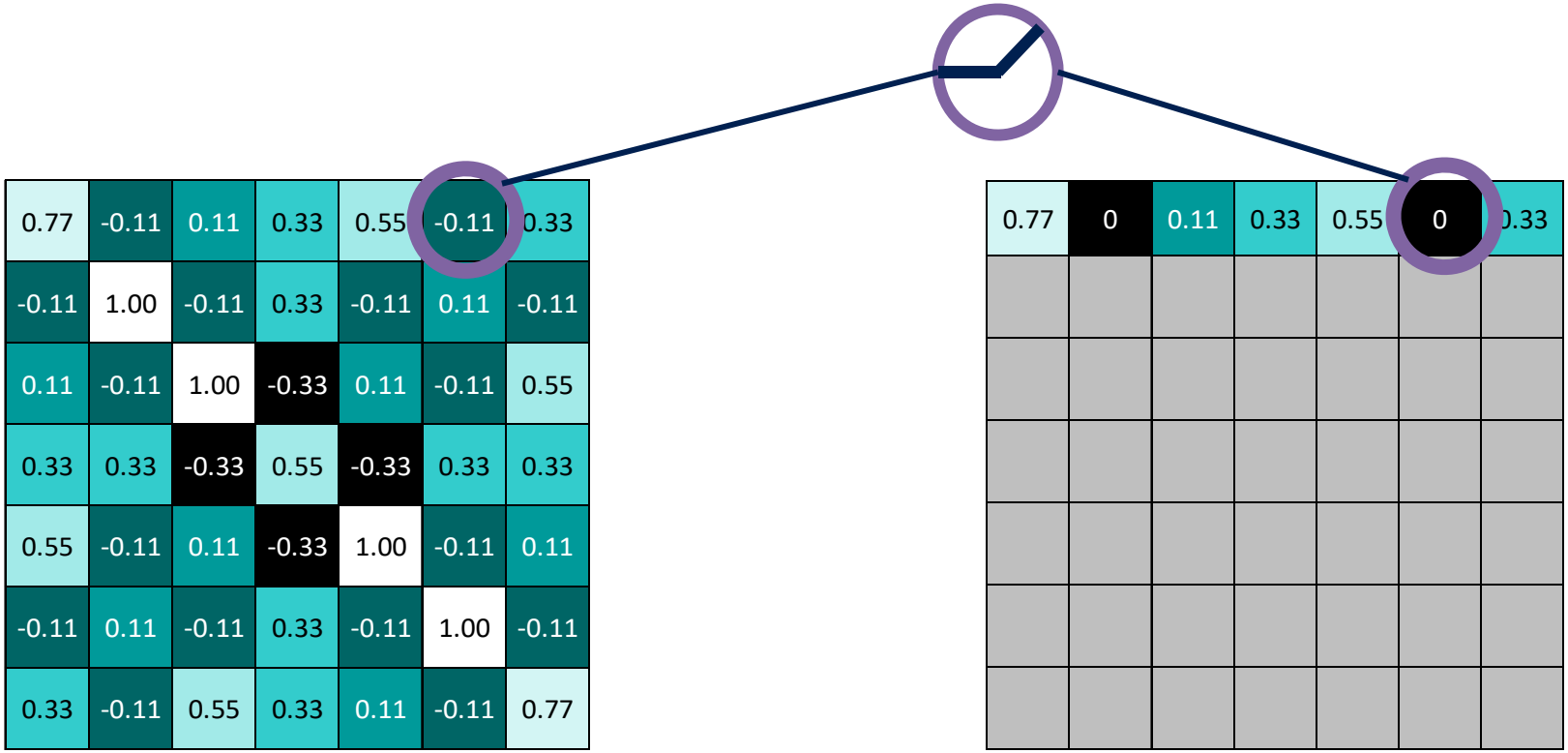
Rectified Linear Units (ReLUs)



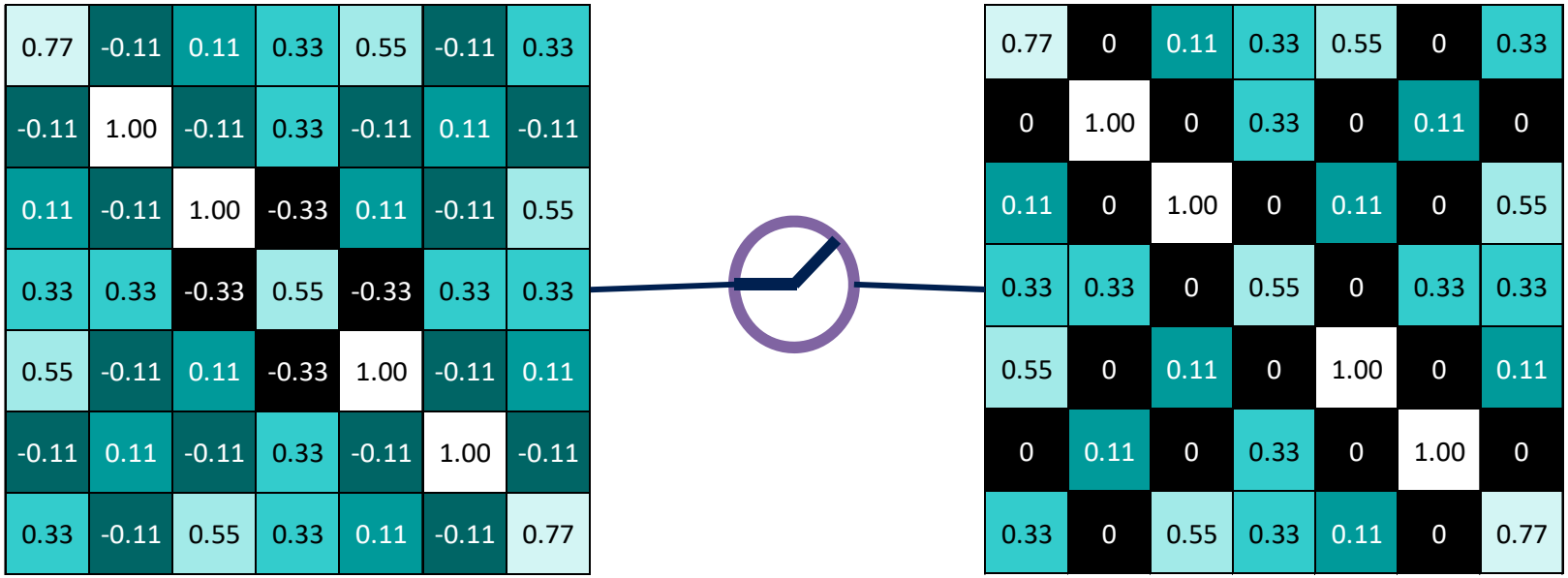
Rectified Linear Units (ReLUs)



Rectified Linear Units (ReLUs)

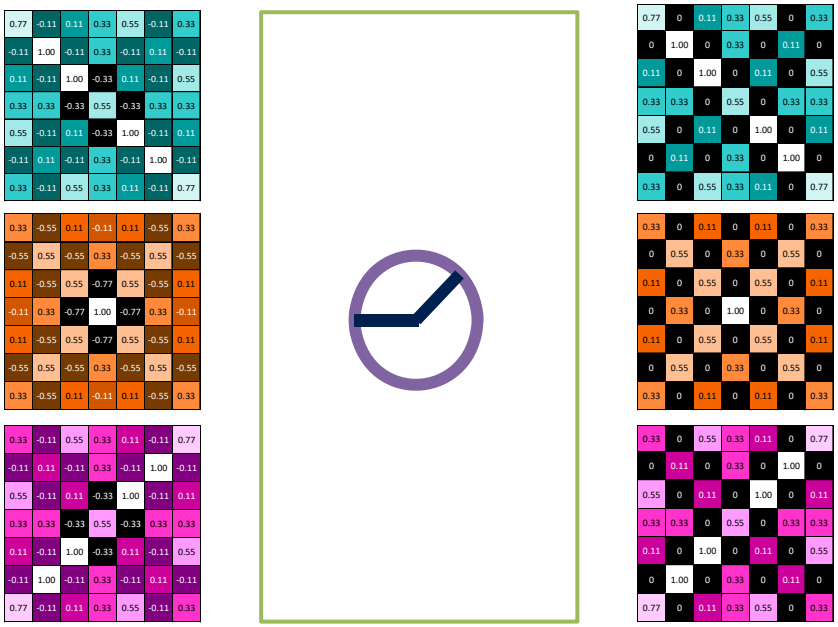


Rectified Linear Units (ReLUs)



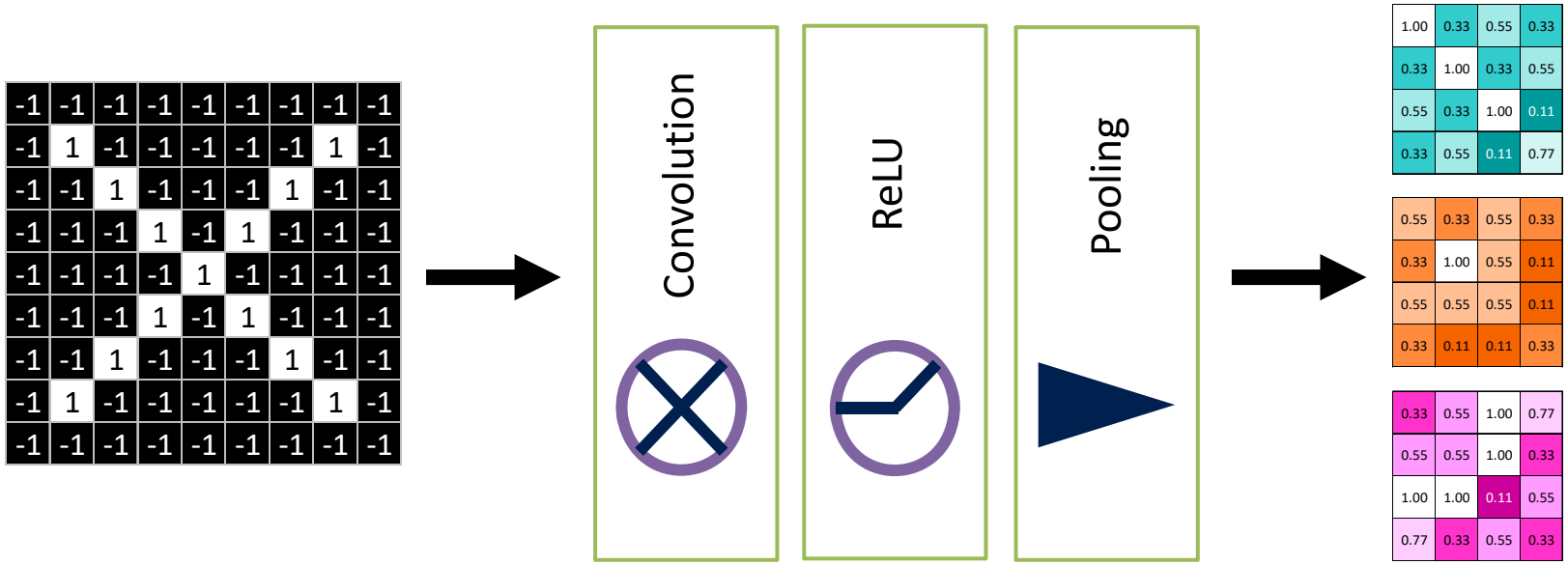
ReLU layer

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



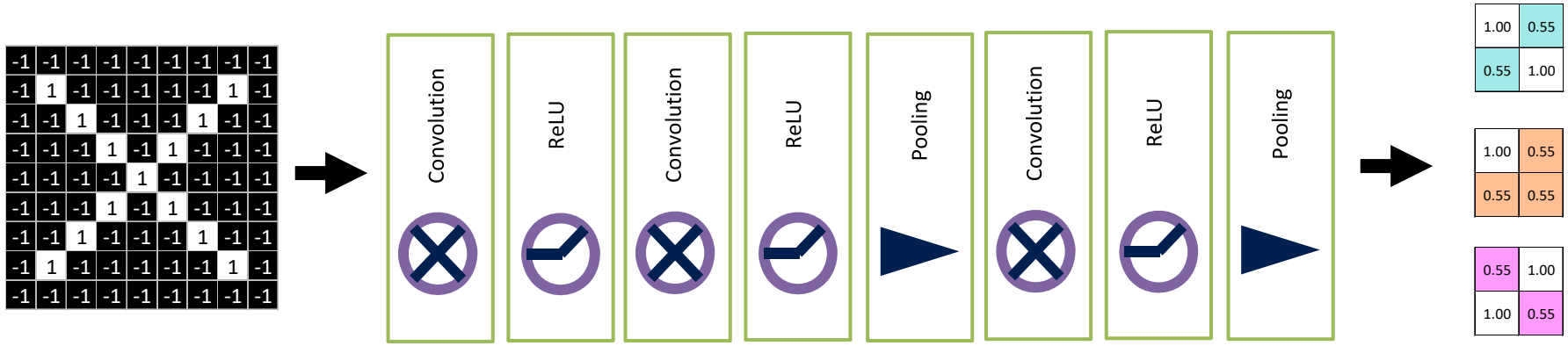
Layers get stacked

The output of one becomes the input of the next.



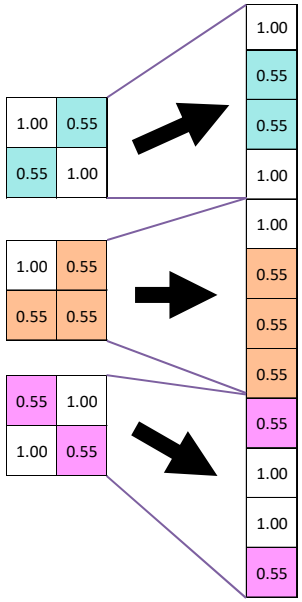
Deep stacking

Layers can be repeated several (or many) times.



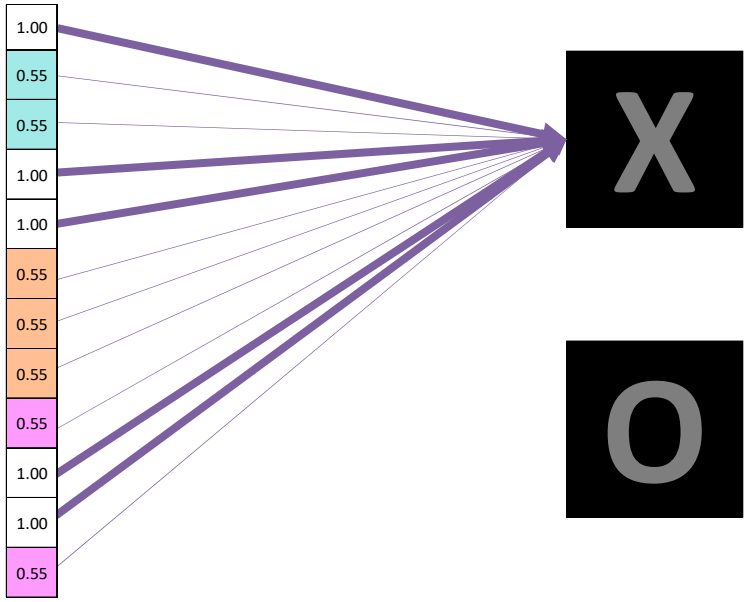
Fully connected layer

Every value gets a vote



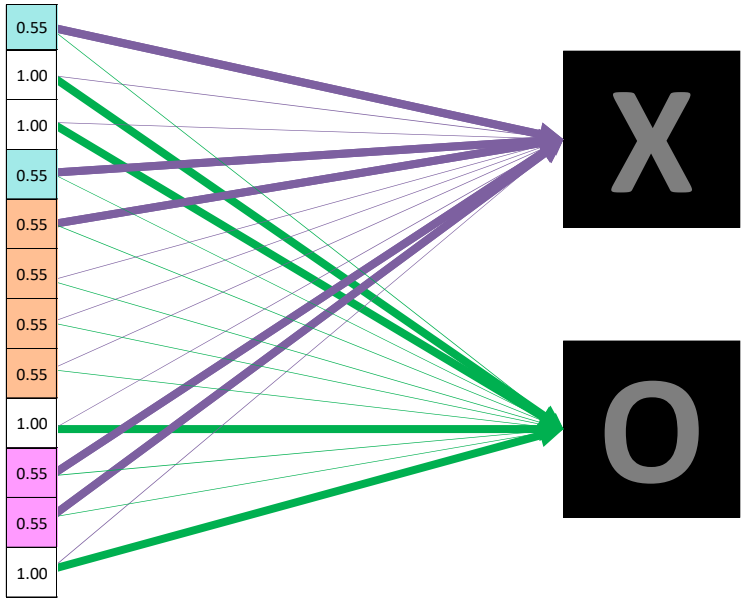
Fully connected layer

Vote depends on how strongly a value predicts X or O



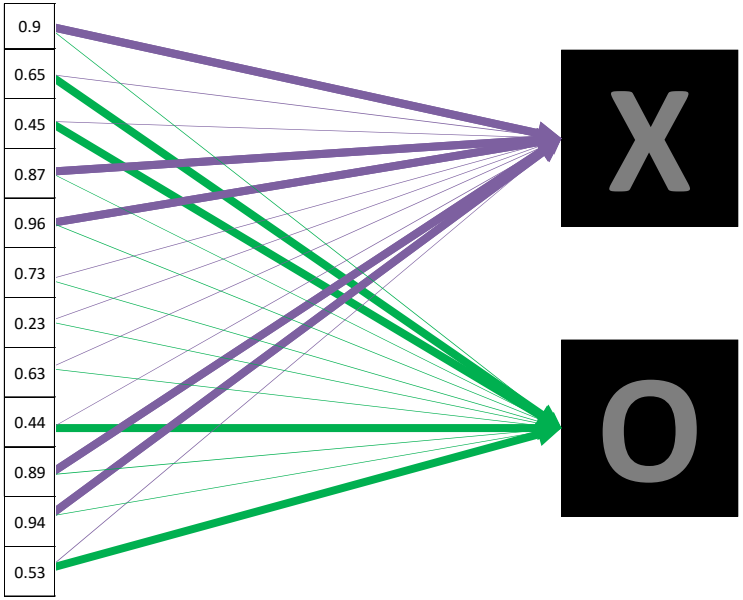
Fully connected layer

Vote depends on how strongly a value predicts X or O



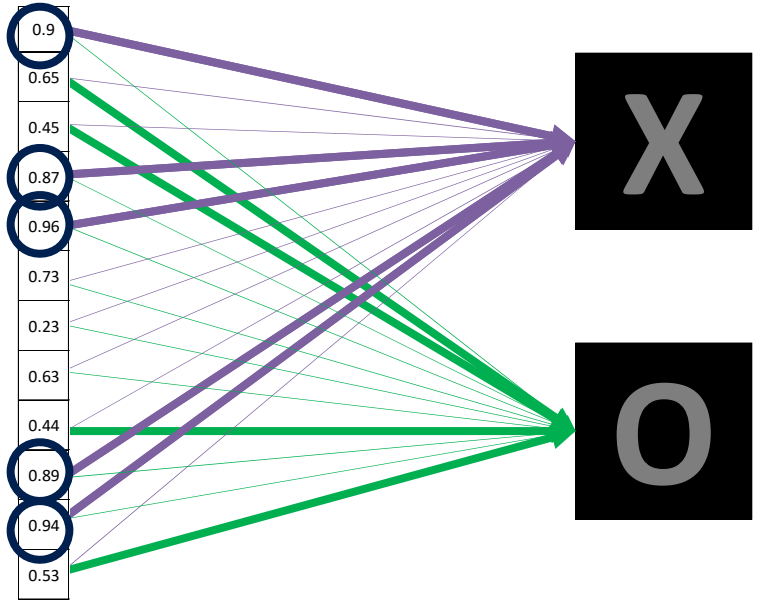
Fully connected layer

Future values vote on X or O



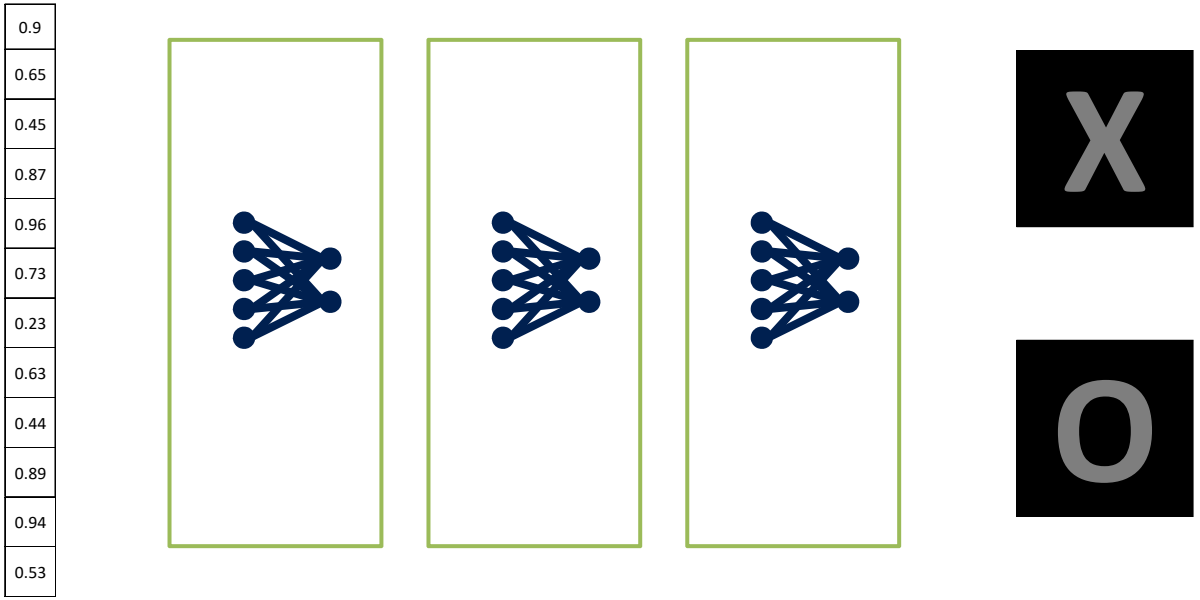
Fully connected layer

Future values vote on X or O



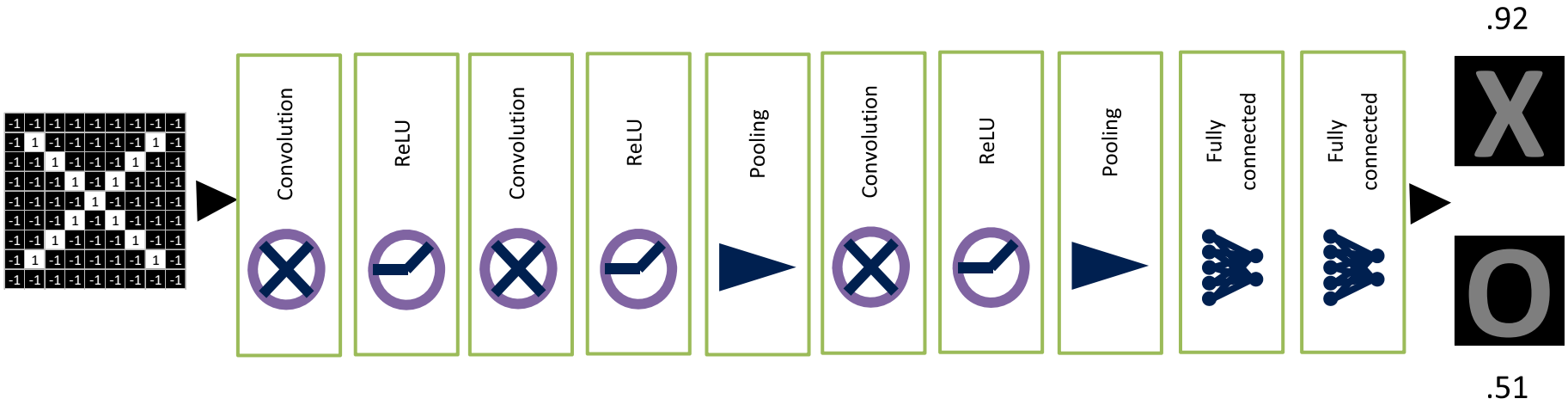
Fully connected layer

These can also be stacked.



Putting it all together

A set of pixels becomes a set of votes.



Learning

Q: Where do all the magic numbers come from?

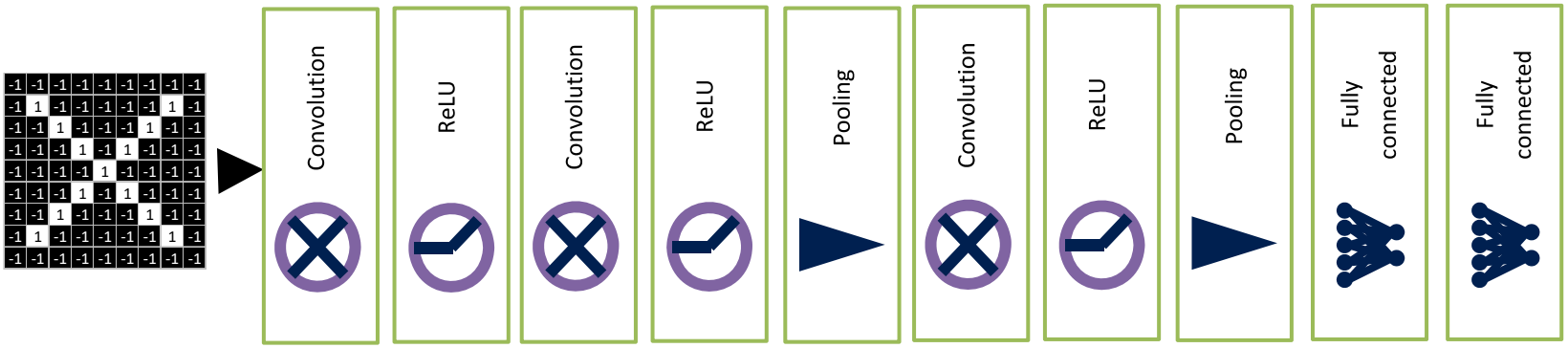
Features in convolutional layers

Voting weights in fully connected layers

A: Backpropagation

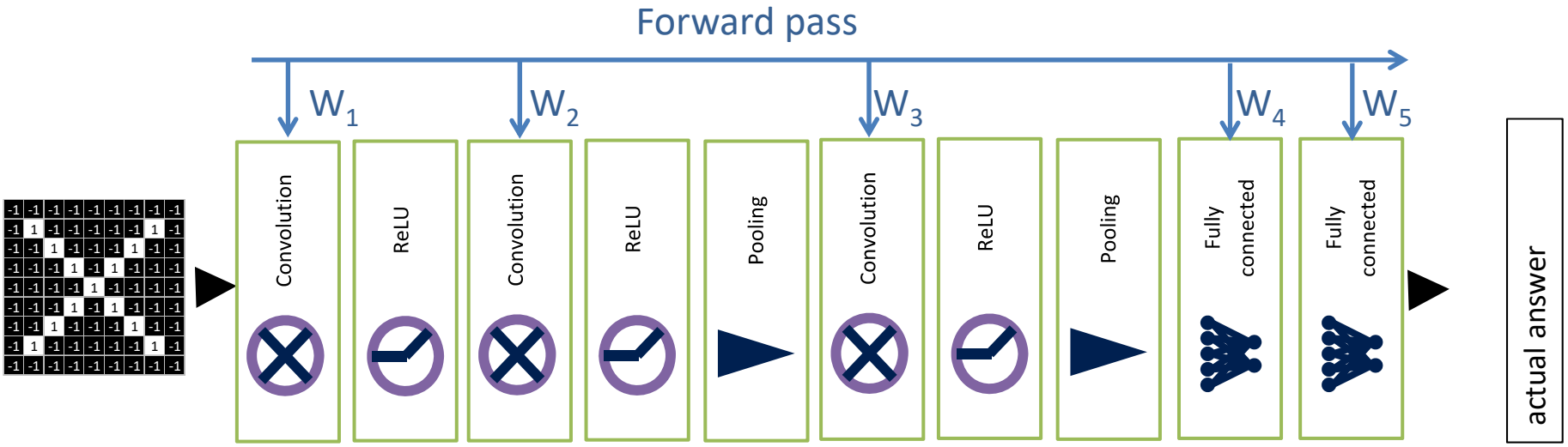
Backpropagation

In the first iteration Wi are set randomly



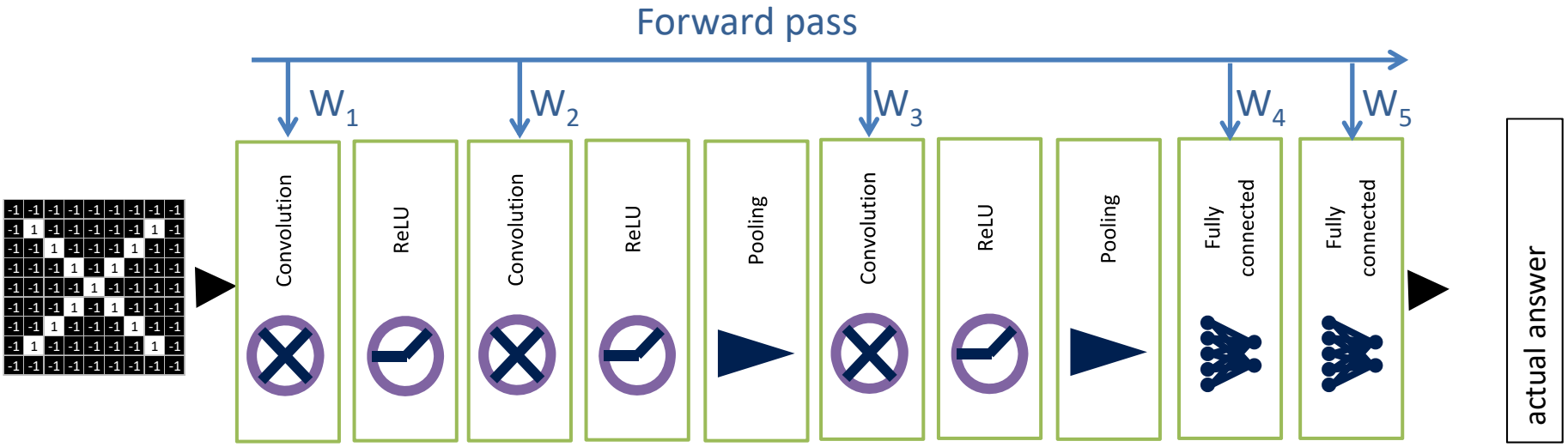
Backpropagation

In the first iteration Wi are set randomly



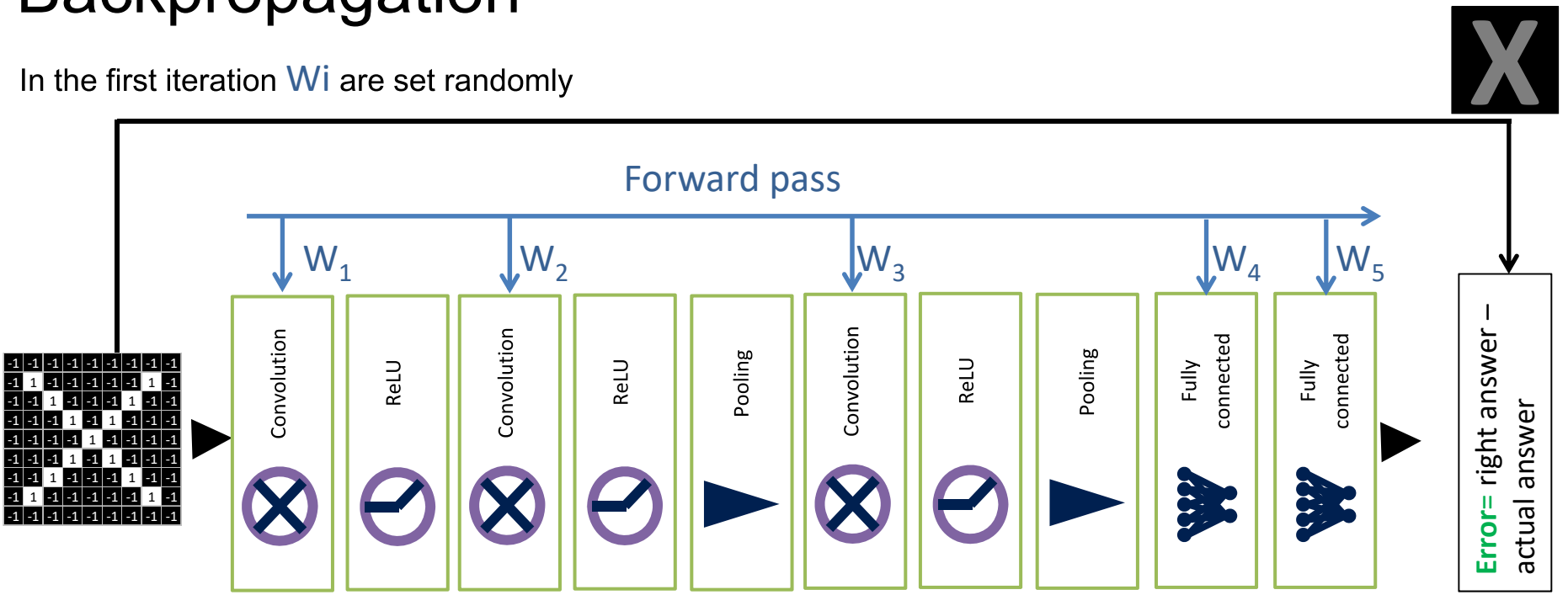
Backpropagation

In the first iteration Wi are set randomly



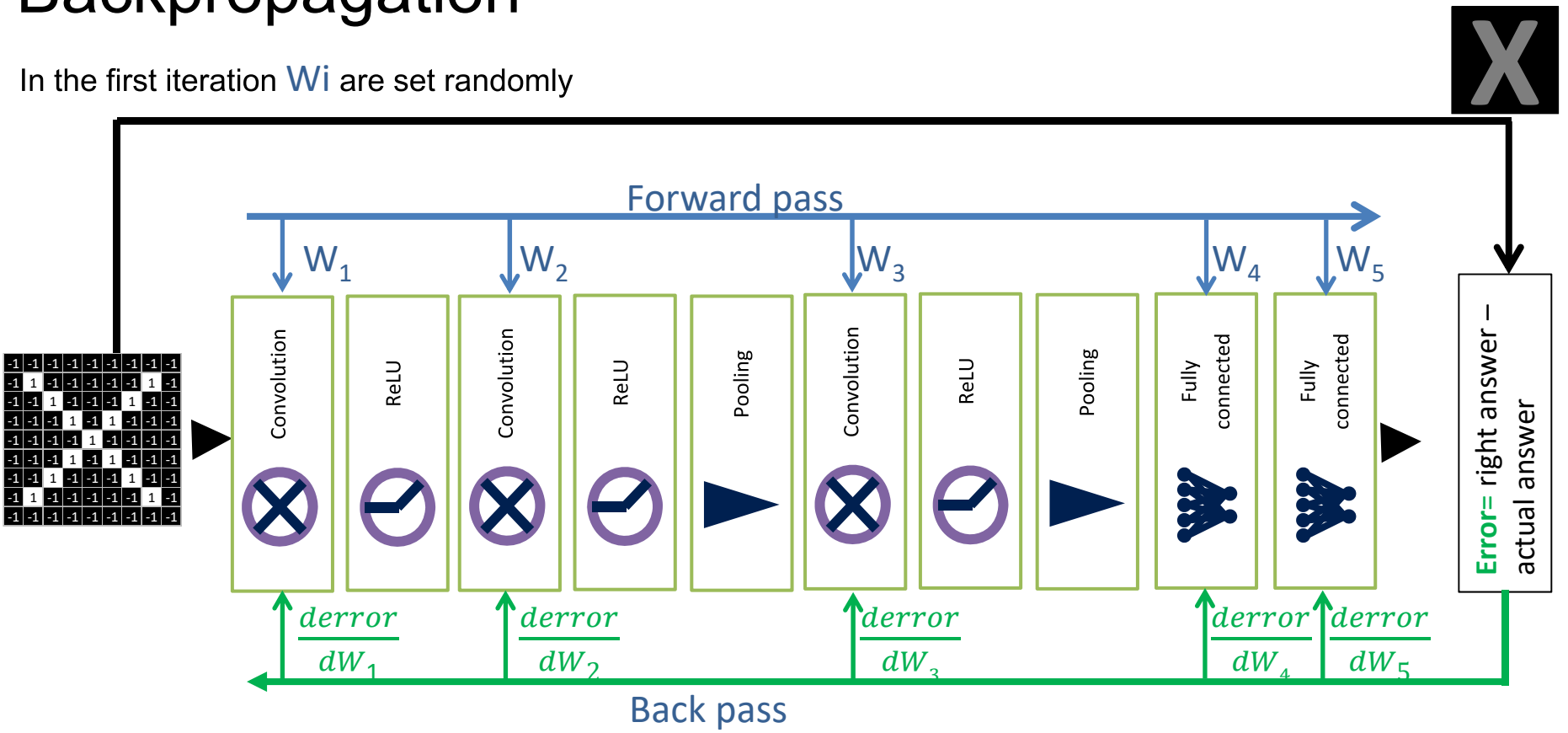
Backpropagation

In the first iteration Wi are set randomly



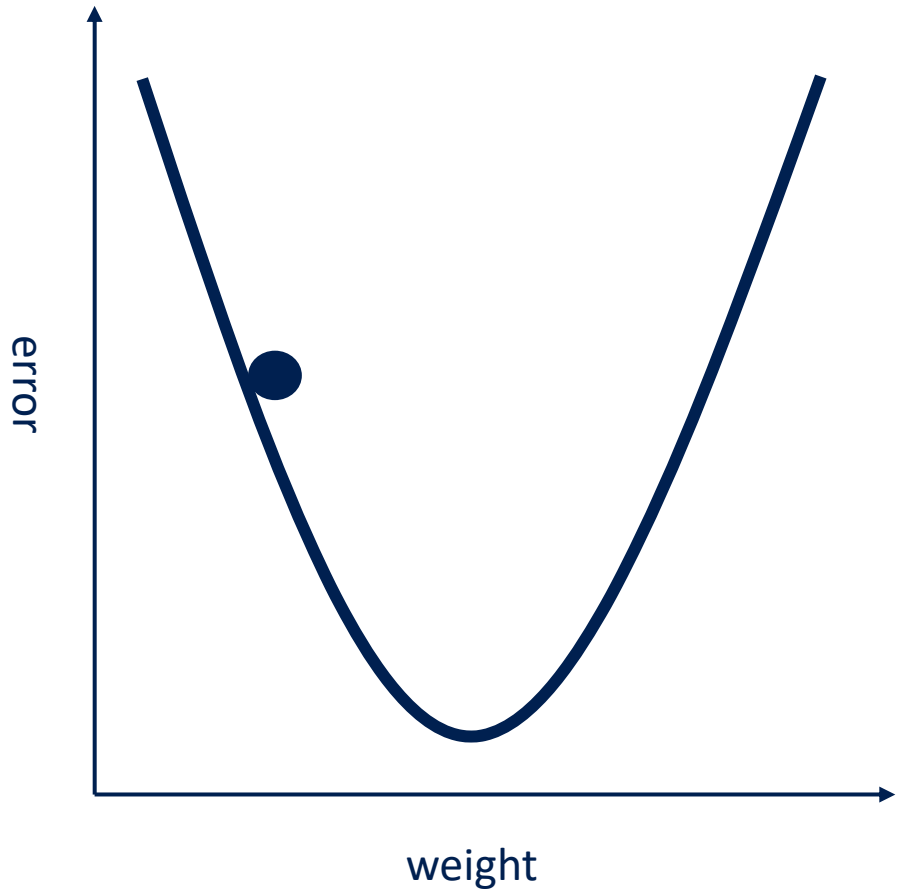
Backpropagation

In the first iteration Wi are set randomly



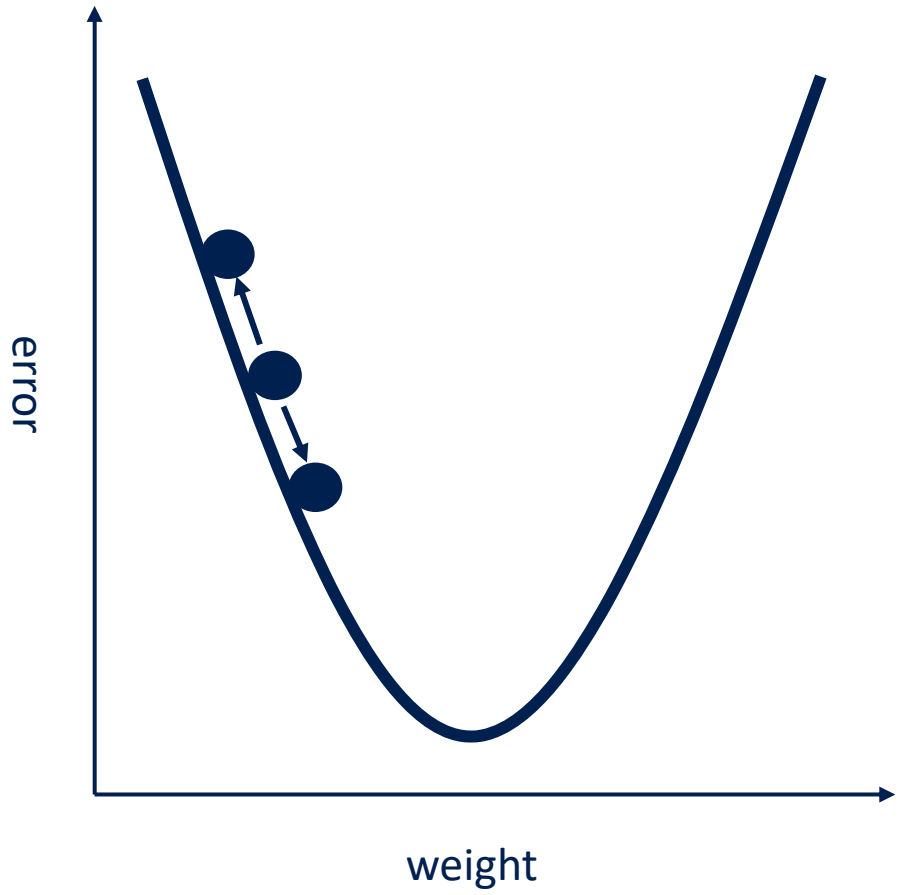
Gradient descent

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



Gradient descent

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



Limitations

- CNNs require large amount of data to get high accuracies

Practical solutions

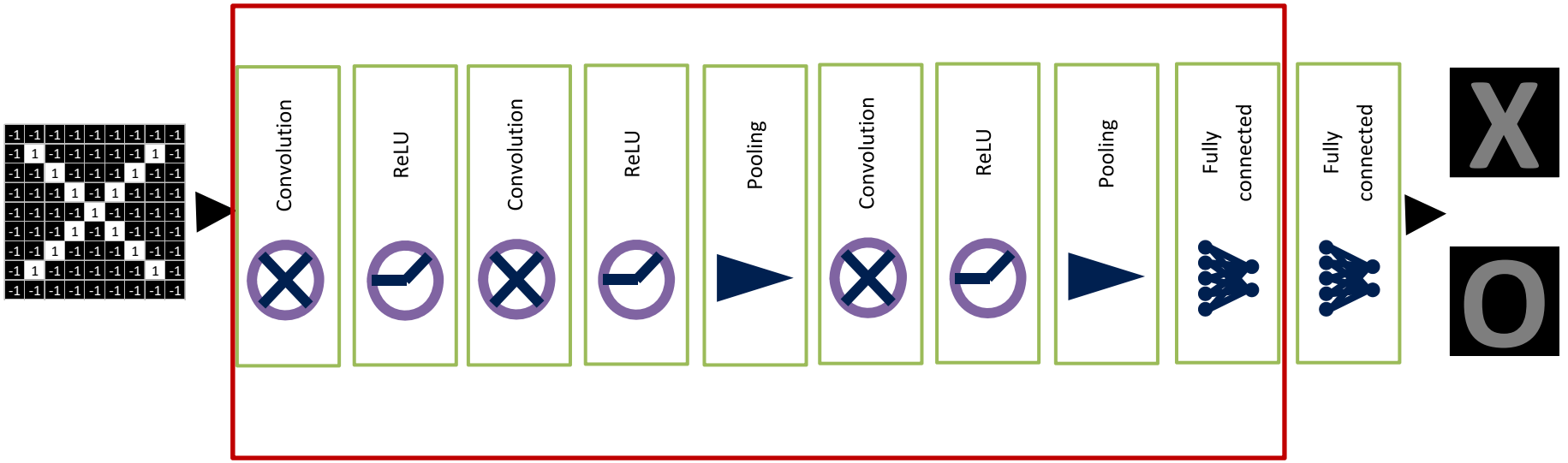
- Transfer learning
- Data augmentation

Outline

1. What are ANN?
2. What are CNNs?
3. How do CNNs work?
   - Convolution layers, pooling layers, FC layers, Backpropagation
4. Transfer learning
5. Data augmentation
6. One case study
7. CNNs for Detection
8. Two case studies

Transfer Learning

- Given a new problem & new data
- Instead of training from scratch, train only the last layers



Outline

1. What are ANN?
2. What are CNNs?
3. How CNNs work?
   - Convolution layers, pooling layers, FC layers, Backpropagation
4. Transfer learning
5. Data augmentation
6. Three case studies

Data augmentation

- Increase the training dataset volume artificially using transformations
- Objective: Improve model robustness

Original : Scaling Centring translation rotation …. etc

Outline

1. What are ANN?
2. What are CNNs?
3. How CNNs work?
   - Convolution layers, pooling layers, FC layers, Backpropagation
4. Transfer learning
5. Data augmentation
6. One case study
7. CNNs for Detection
8. Two case studies

Case study 1: Data-augumentation for CNNs using MNIST

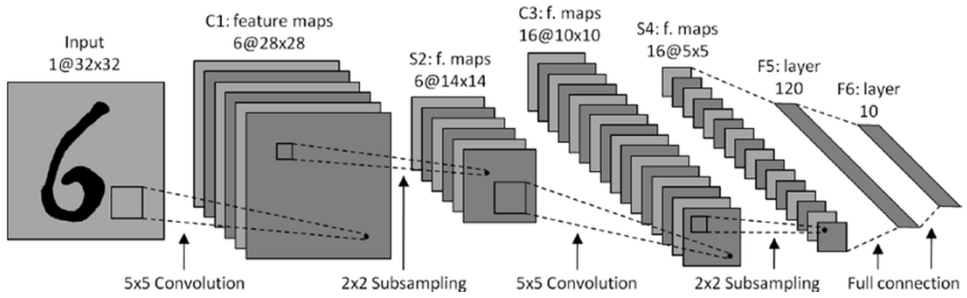
- Objective: Analyze the benefit of data-augmentation and ensembles on CNNs
- Methodology:
  - MNIST (60.000 train + 30.000 test) & 10 classes
  - Three CNNs: LeNet, Network3, Dropconnect

A snapshot of image pre-processing for convolutional neural networks: case study of MNIST S Tabik, D Peralta, A Herrera-Poyatos, F Herrera. International Journal of Computational Intelligence Systems 10, 555–568

Case study 1: Data-augumentation for CNNs using MNIST

Three Lenet like CNNs:

1. LeNet: 2 Conv + 2FC(ReLU)
2. Network3: 2 Conv + 3FC (ReLU+Dropout)
3. DropConnect: 2 Conv + 2FC (ReLU+Dropconnect)



Data-augmentation techniques

Dataset	Combination	# of training instances
1	Original	60,000
2	Centering	60,000
3	Elastic	300,000
4	Translation	300,000
5	Rotation	300,000
6	Elastic-centering	300,000
7	Rotation-centering	300,000
8	Translation-elastic	1,500,000
9	Translation-rotation	1,500,000
10	Rotation-elastic	1,500,000
11	Rotation-elastic-centering	1,500,000
12	Elastic-elastic	1,500,000

Case study 1: Data-augumentation for CNNs using MNIST

Test-set accuracies ($\frac{correct\ predictions}{total\ test\ samples}$)

	LeNet (10,000 iter.)				LeNet (50,000 iter.)			
Dataset	**Average**	**Best**	**Epochs**	**Time(s)**	**Average**	**Best**	**Epochs**	**Time(s)**
Original	99.08%	99.18%	10.67	267.91	99.05%	99.21%	213.33	1070.29
Centered	98.85%	99.06%	10.67	203.52	98.95 %	98.09%	213.33	926.38
Elastic	99.09%	99.19%	2.13	232.75	99.36%	99.44%	42.67	1065.38
Translation	99.09%	99.32%	2.13	268.75	99.30%	99.41%	42.67	1065.38
Rotation	99.05%	99.10%	2.13	268.03	99.25%	99.37%	42.67	1065.38
Elastic-centered	[5]99.17%	99.26%	2.13	267.20	99.27%	99.36%	42.67	925.51
Rotation-centered	98.90%	99.07%	2.13	232.73	99.19%	99.33%	42.67	950.38
Translation-elastic	[4]99.18%	99.32%	0.43	267.43	[5]99.39%	99.54%	8.53	1050.38
Translation-rotation	99.16%	99.40%	0.43	267.41	[3]99.40%	97.55%	8.53	1045.38
Rotation-elastic	[1]**99.31%**	99.39%	0.43	268.14	[1]**99.47%**	**99.57%**	8.53	1046.25
Rotation-elastic-centered	[3]99.19%	99.24%	0.43	232.30	[2]99.43%	99.52%	8.53	925.68
Elastic-elastic	[2]99.27%	**99.45%**	0.43	268.10	[4]99.40%	99.50%	8.53	1047.64

Case study 1: Data-augumentation for CNNs using MNIST

Test-set accuracies

	Network3(10 epochs)			Network3(20 epochs)		
Dataset	**Average**	**Best**	**Time(s)**	**Average**	**Best**	**Time(s)**
Original	99.01%	99.07%	124.45	99.25%	99.25%	205,21
Centered	98.73%	98.80%	118.32	98.97%	99.01%	196.92
Elastic	99.49%	99.54%	656,85	[3]99.61%	99.67%	1200,33
Translation	[5]99.49%	99.55%	631.53	[4]99.59%	99.63%	1228,71
Rotation	99.44%	99.50%	636.25	99.44%	99.50%	1256,95
Elastic-centered	99.32%	99.39%	566.44	99.57%	99.60%	1109,43
Rotation-centered	98.88%	98.94%	569.04	99.31%	99.32%	1167,32
Translation-elastic	[4]99.54%	99.57%	3647.78	[5]99.58%	99.63%	7111,65
Translation-rotation	[3]99.57%	99.61%	3650.66	99.58%	99.60%	7149,25
Rotation-elastic	[2]99.62%	**99.67%**	3642,85	[2]99.67%	99.69%	6996,23
Rotation-elastic-centered	99.43%	99.51%	3054,43	99.51%	99.52%	6908,70
Elastic-elastic	[1]**99.65%**	99.66%	3607.32	[1]**99.67 %**	**99.70%**	7189,16

Case study 1: Data-augumentation for CNNs using MNIST

Test-set accuracies

	DropConnet(100 epochs)			DropConnet(200 epochs)		
Dataset	**Average**	**Best**	**Time(s)**	**Average**	**Best**	**Time(s)**
Original	98,32%	98,83%	7803.43	98.98%	98,99%	18748.53
Centered	95.35%	94,46%	6659.31	95.13%	98,85%	18635.54
Elastic	99.33 %	99,35%	7512.25	99.36%	99,36%	18606.15
Translation	[5]99.43%	99,46%	7736.41	[5]99.47%	99,47%	18710.45
Rotation	99.18%	99,29%	7151.73	99.37%	99,47%	18729.29
Elastic-centered	96.58%	96,69%	6969.89	97.08%	97,09%	18661.80
Rotation-centered	98.30%	98,41%	6974.23	98.55%	98,63%	18668.05
Translation-elastic	99.40%	99,57%	7162.37	[3]99.58%	99,58%	18745.93
Translation-rotation	[2]99.57%	99,59%	7410.32	[1]**99.69%**	**99,69%**	18772.40
Rotation-elastic	[3]99.54%	**99,60%**	7397.40	[4]99.56%	99,56%	18724.38
Rotation-elastic-centered	[4]99.47%	99,49%	7803.73	99,44%	99,46%	18220.50
Elastic-elastic	[1]**99,58%**	99,59%	7911.30	[2]99,59%	99,61%	18712.22

Case study 1: Data-augumentation for CNNs using MNIST

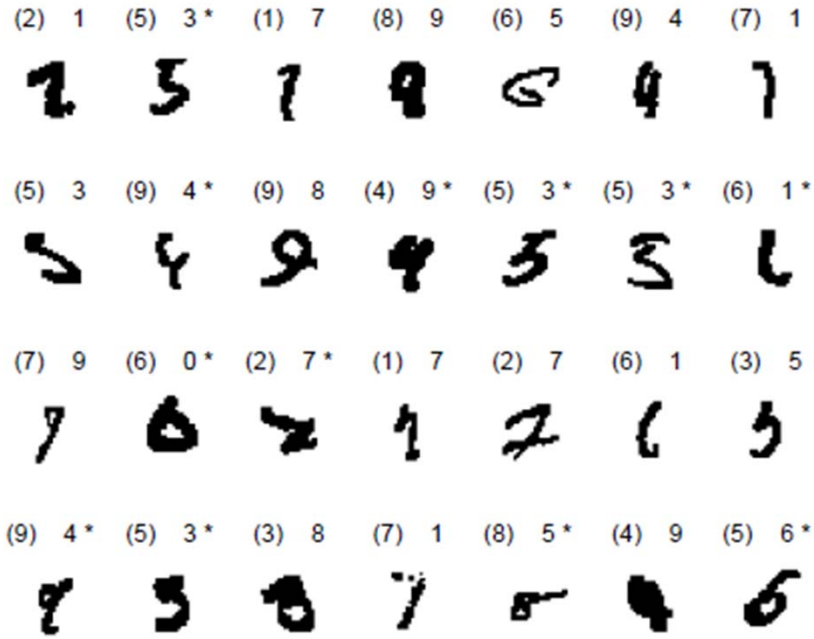
Results Esembles using the most voted strategy

	LeNet(500 neurons)		**Network3**		**DropConnect**	
	10,000 iter	**50,000 iter**	**10 epochs**	**20 epochs**	**100 epochs**	**200 epochs**
Ensemble-5	99,55%	**99,57%**	**99,72%**	**99,69%**	**99,72%**	99,66%
Ensemble-3	99,43%	**99,54%**	99,69%	99,67%	99,69%	**99,68%**

Error : 0.28% versus state of the art ensemble 0.23%

Case study 1: Data-augumentation for CNNs using MNIST

Results:The 28 misclassified characters

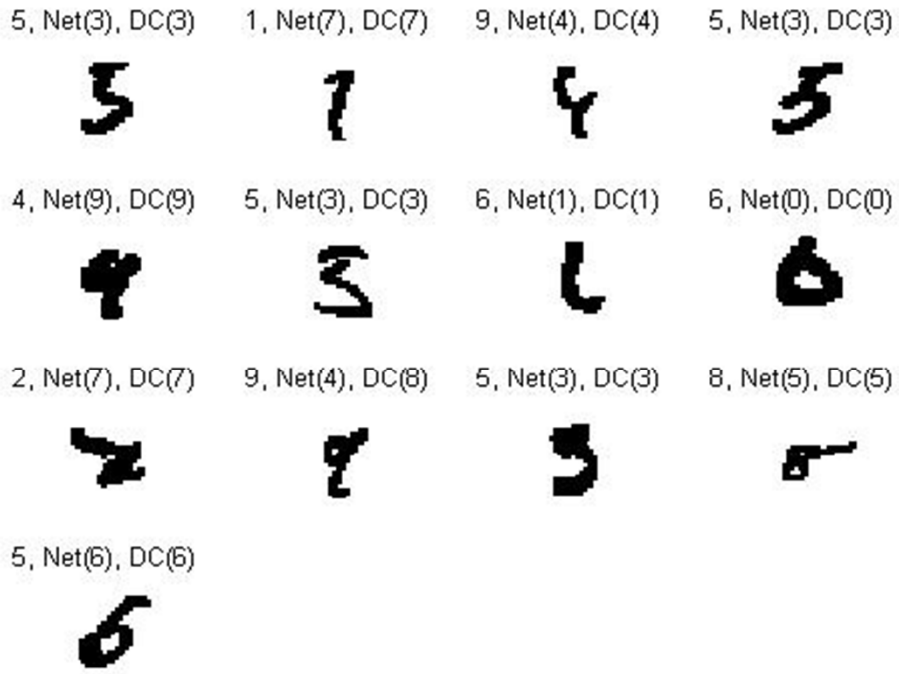


ensemble-5 (Network3)

ensemble-5 (DropConnect)

Case study 1: Data-augumentation for CNNs using MNIST

The 13 handwritten digits misclassified by ensemble-5 of DropConnet and Network3



Detection with CNNs

Siham Tabik

siham@ugr.es

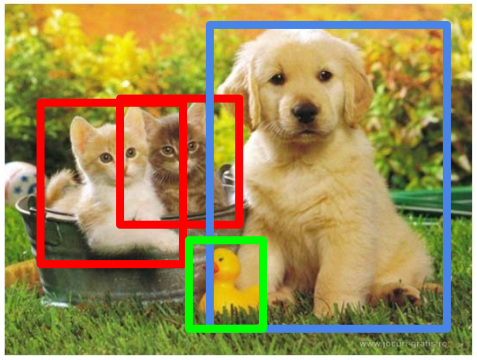
Outline

- What is detection?
- How detection models work?
- The state-of-the art detection models
- Case study 1
- Case study 2

By the way, what is object detection?

Three classes: cat, dog and duck.
Given a test image:

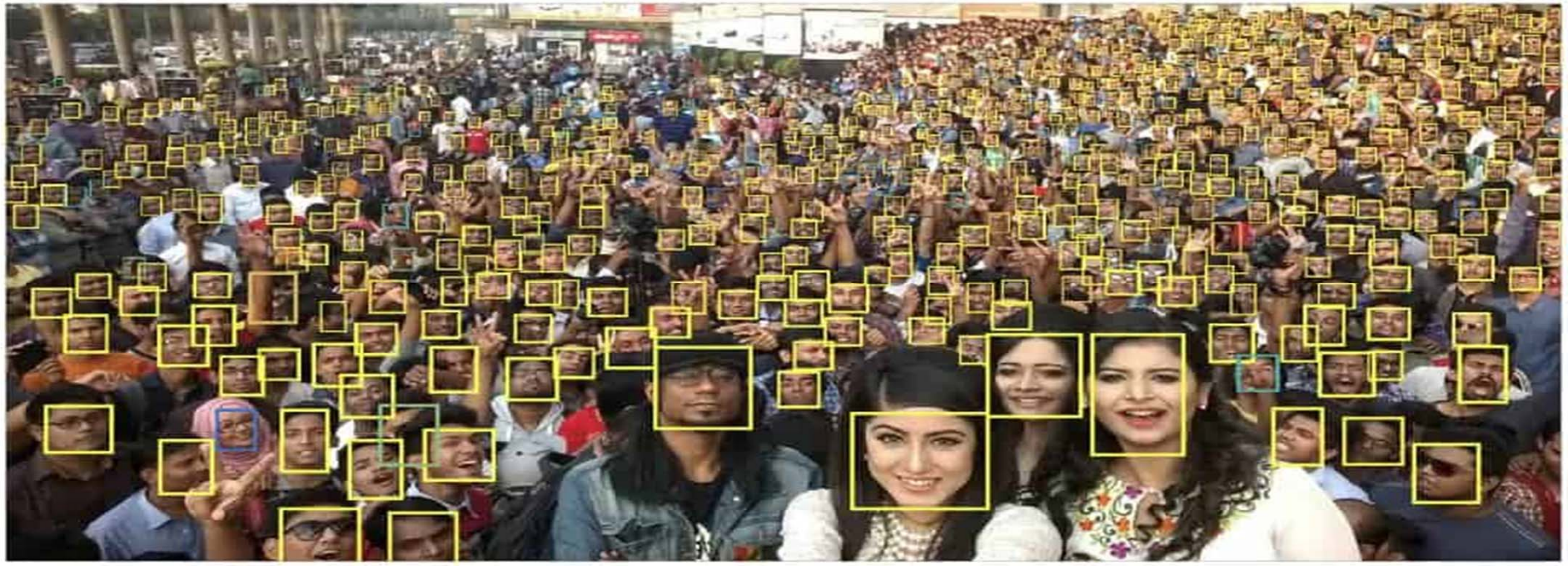
Object Detection



CAT, DOG, DUCK

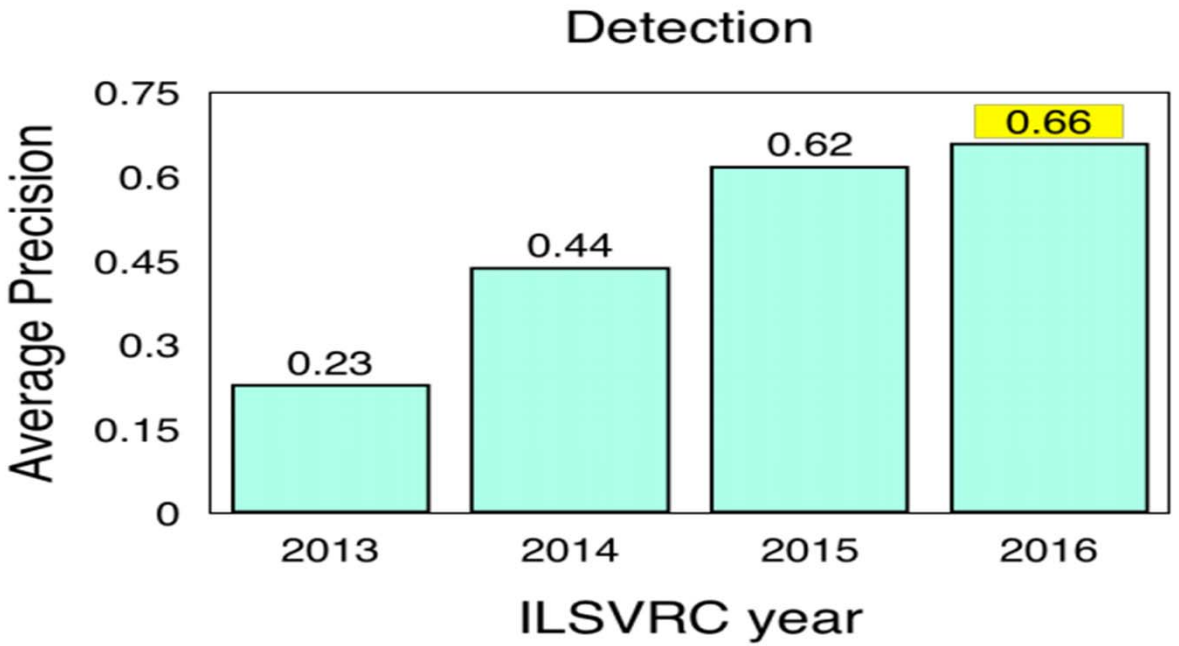
Object detection

An example of face detection: Object Detection of a single class.



Results of the ILSVRC object detection task over four years

Database: 200 object classes 527,982 images



How object detection works?

- Object detection methods reformulate the detection problem into a classification problem
- Combine a selective research method with a classification model

Detection as Classification

CAT? NO

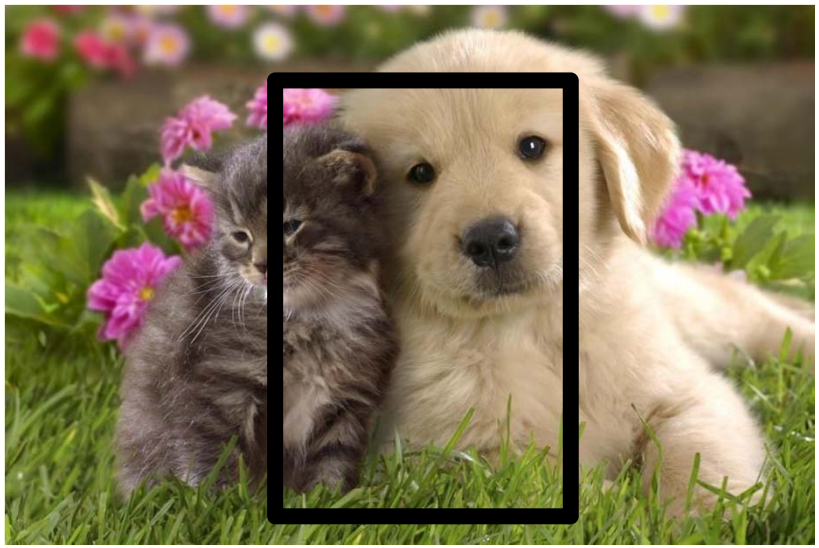
DOG? NO

Detection as Classification

CAT? YES!

DOG? NO

Detection as Classification



CAT? NO

DOG? NO

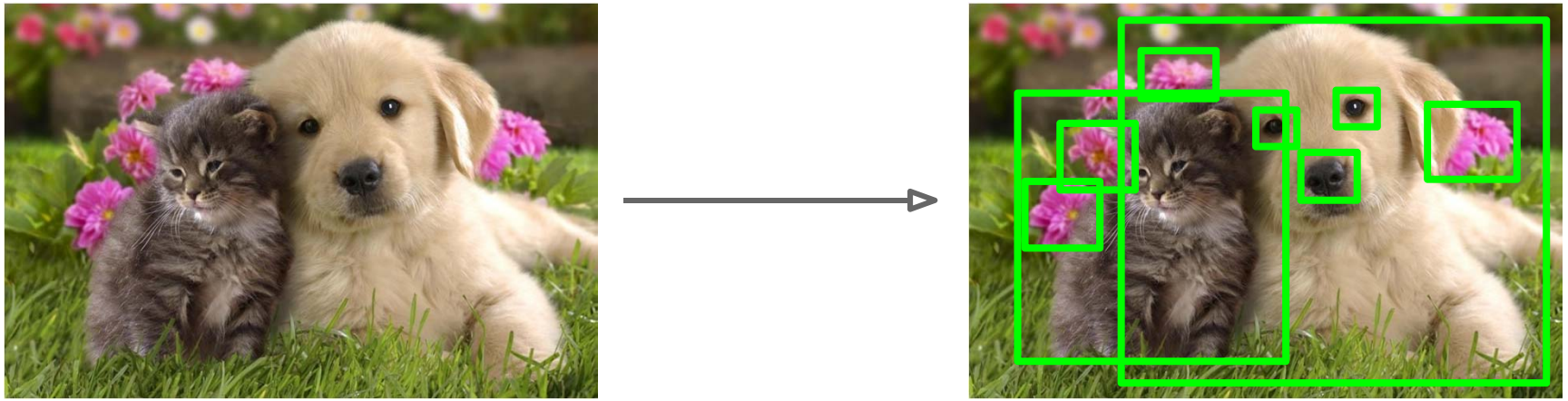
Detection as Classification

Problem: Need to test many positions and scales, and use a computationally demanding classifier (CNN)

Solution: Only look at a tiny subset of possible positions

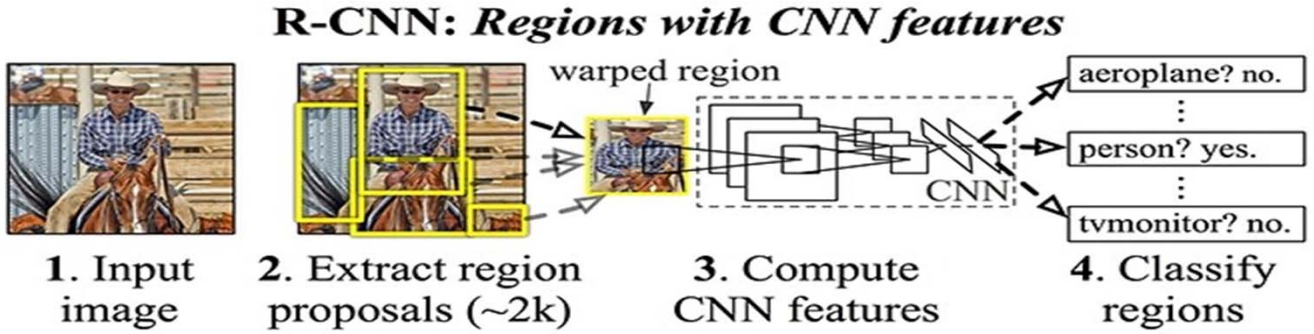
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



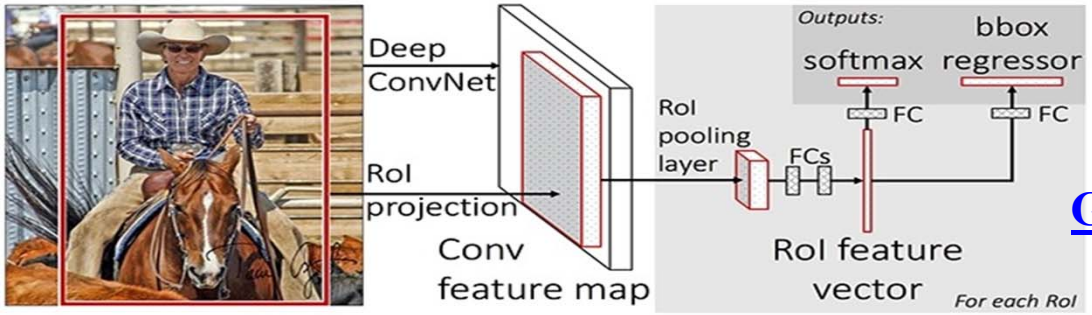
R-CNN (RP+CNN)

- The first CNN based detection model was R-CNN
- Combines: VGG and Region Proposals
- Boosted the object detection challenge by almost 50%.

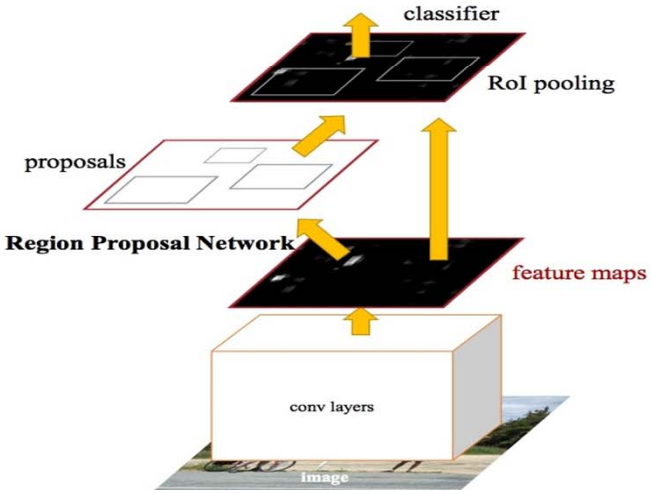


Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." 2014.

Fast R-CNN and Faster R-CNN

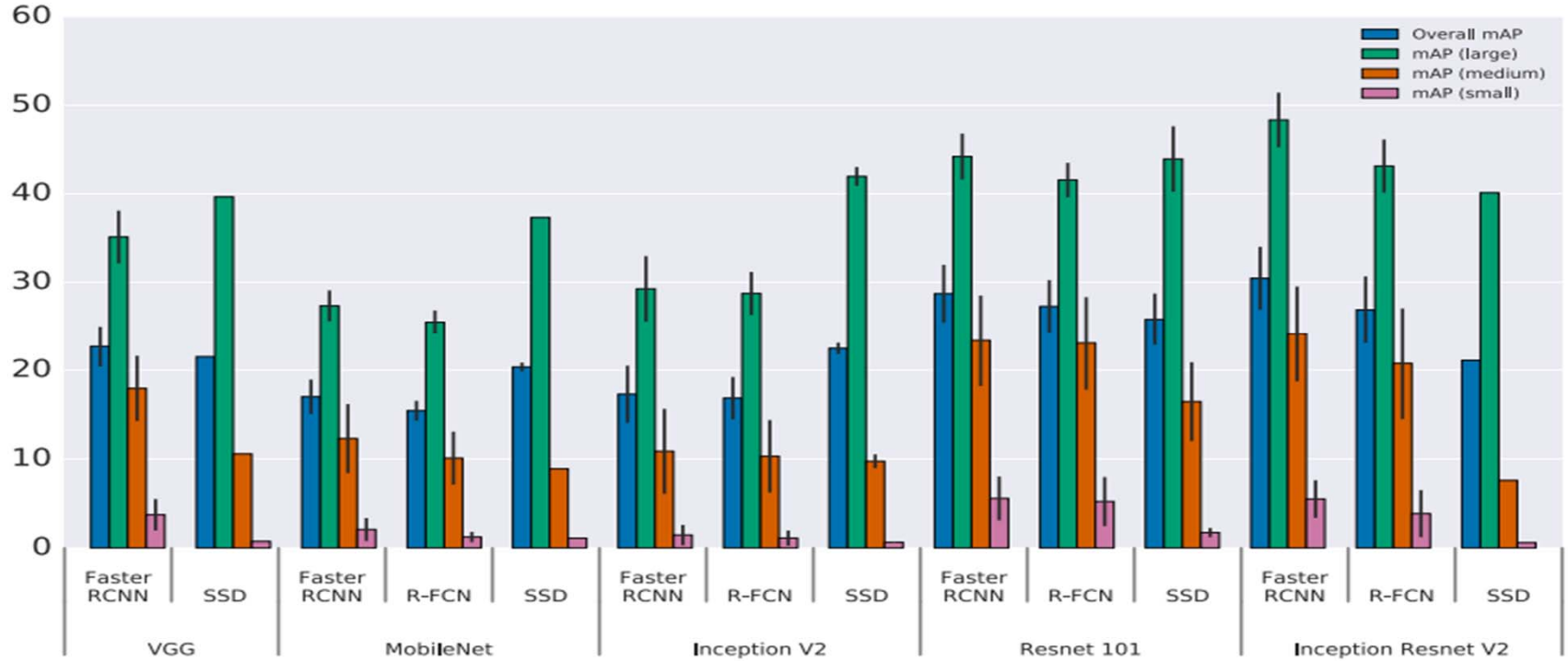


[Girshick, Ross. "Fast R-CNN" 2015.]



[Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." 2015.]

Faster R-CNN, SSD, R-FCN

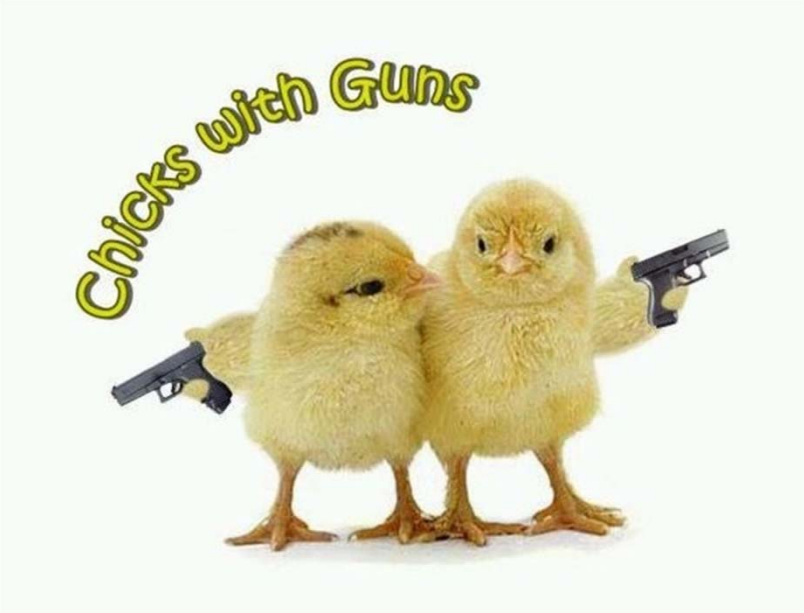


Outline

- What is detection?
- How detection models work?
- The state-of-the art detection models
- Case study 1
- Case study 2

Case study 3: Handgun alarm detection in videos

- Objective: Develop a fast and accurate pistol detection model in videos
- Methodology:
  - New database
  - Two models:
    - "VGG+ Sliding Window"
    - "VGG+ Region Proposal (Faster-RCNN)"

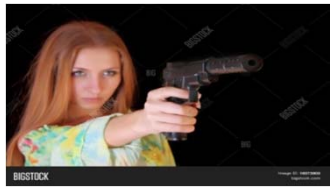


Automatic Handgun Detection Alarm in Videos Using Deep Learning R Olmos, S Tabik, F Herrera. Neurocomputing, 275, 66-72

Case study 3: Handgun alarm detection in videos

A new database

Pistol class



Negative class

Results on the same dataset

Database	#TP	#FN	#TN	#FP	Precision	Recall	F1_measure
For Sliding window	97	207	298	6	**94.17%**	31.91%	47.67%
For Faster RCNN	304	0	247	57	81.21%	**100.00%**	**91.43%**

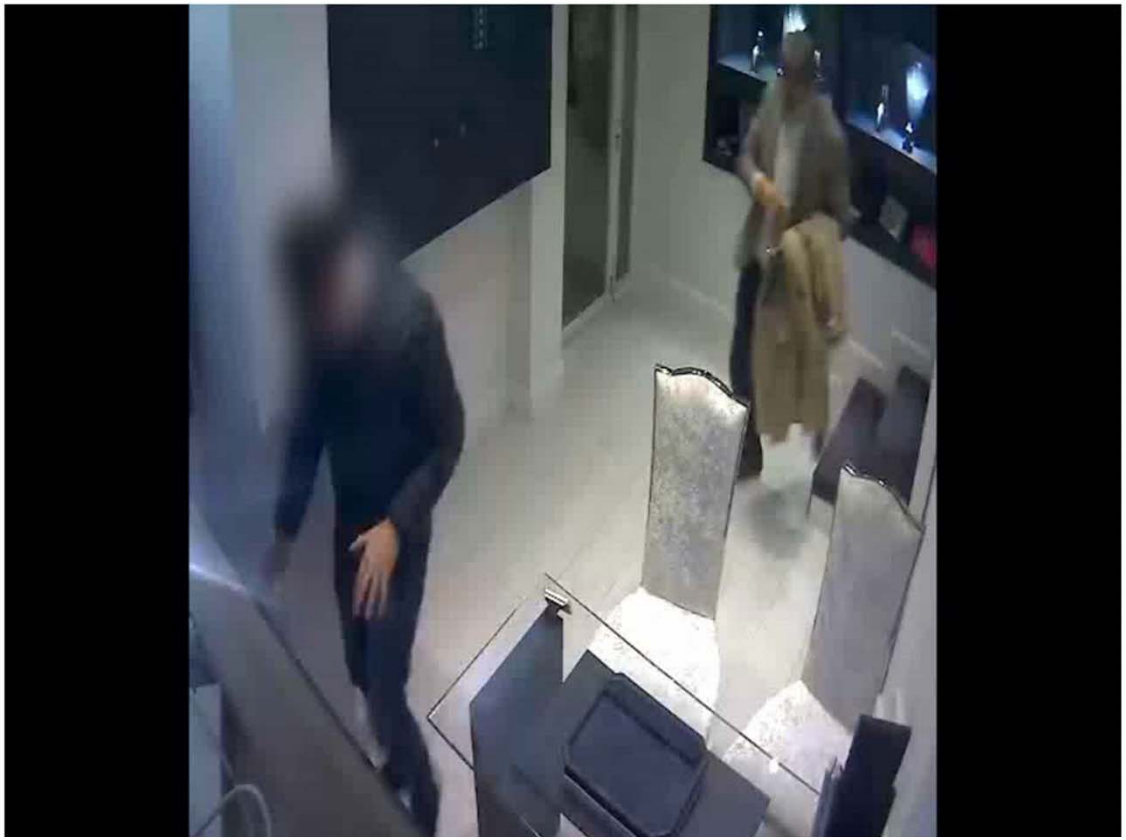
Results in images

Results on pictures

Results on youtube videos

Results on Skyfall

Results on real surveillance videos



Outline

- What is detection?
- How detection models work?
- The state-of-the art detection models
- Case study 1
- Case study 2

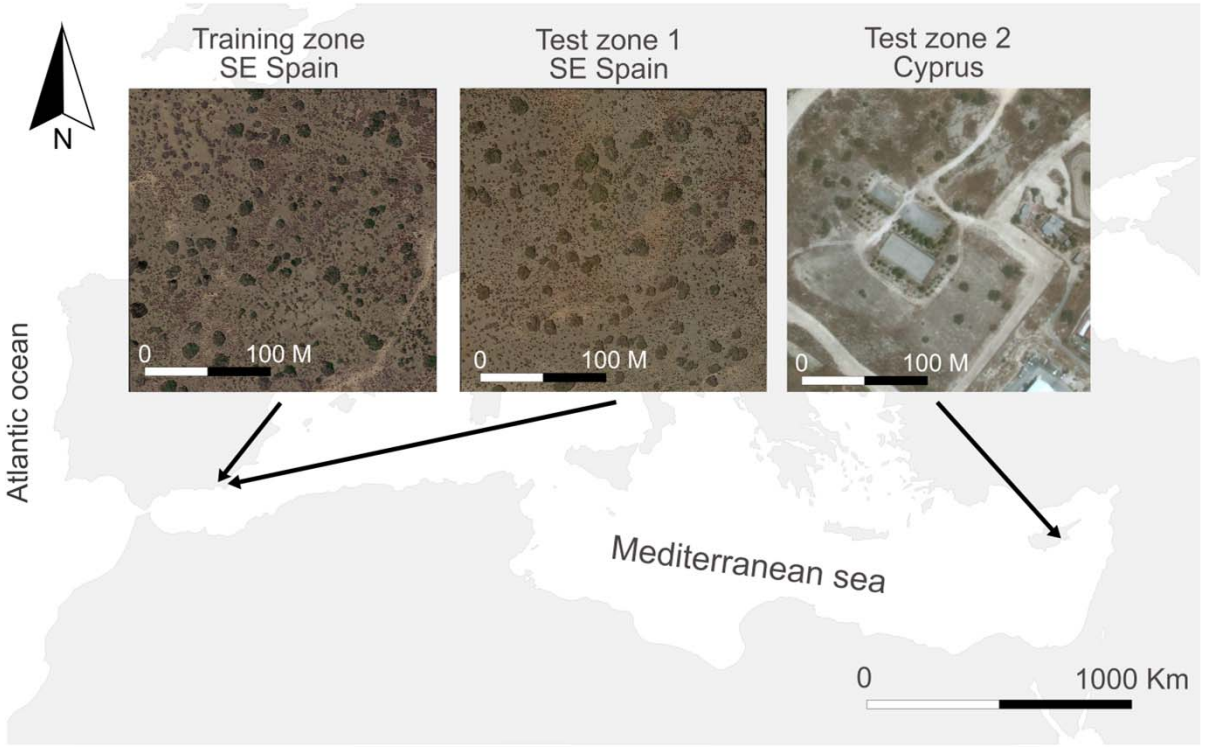
Case study 2: Protected plant species detection using Google-earth imagery

- Objective: *Ziziphus Lotus* detection in GE images
- Methodology:
  - Design new database to train CNNs
  - Three models: "CNNs+ Sliding Window" "CNNs+ Preprocessing" "OBIA"

Deep-Learning Convolutional Neural Networks for scattered shrub detection with Google Earth Imagery E Guirado, S Tabik, D Alcaraz-Segura, J Cabello, F Herrera. Remote Sensing 9(12): 1220 (2017)

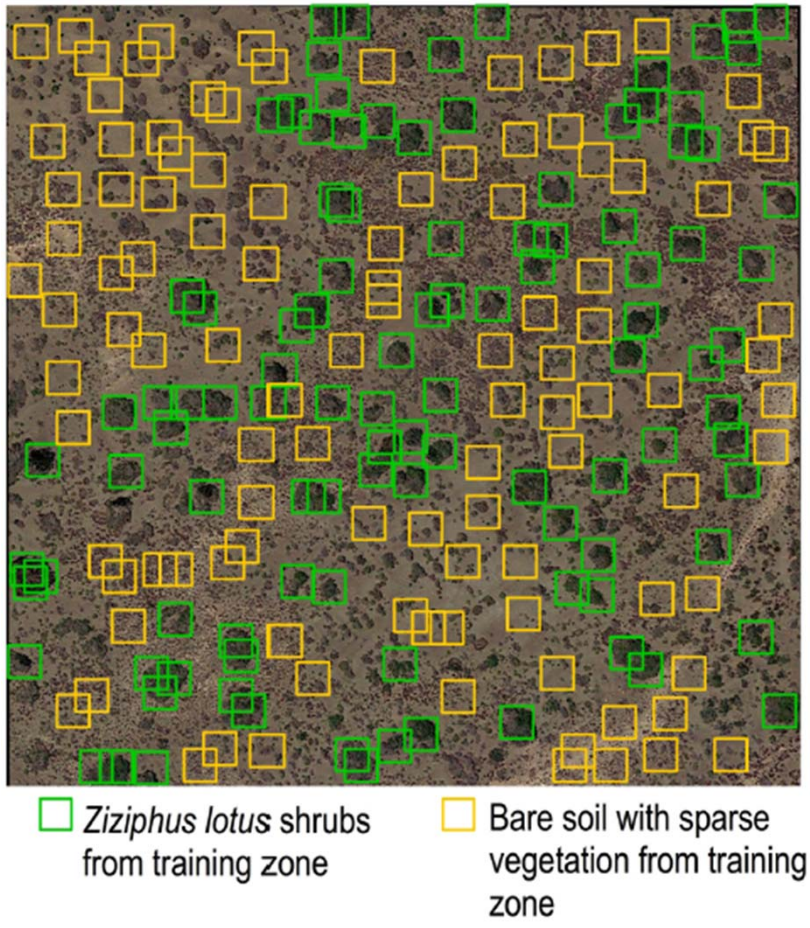
Case study 2: Protected plant species detection using Google-earth imagery

- Design new database to train CNNs

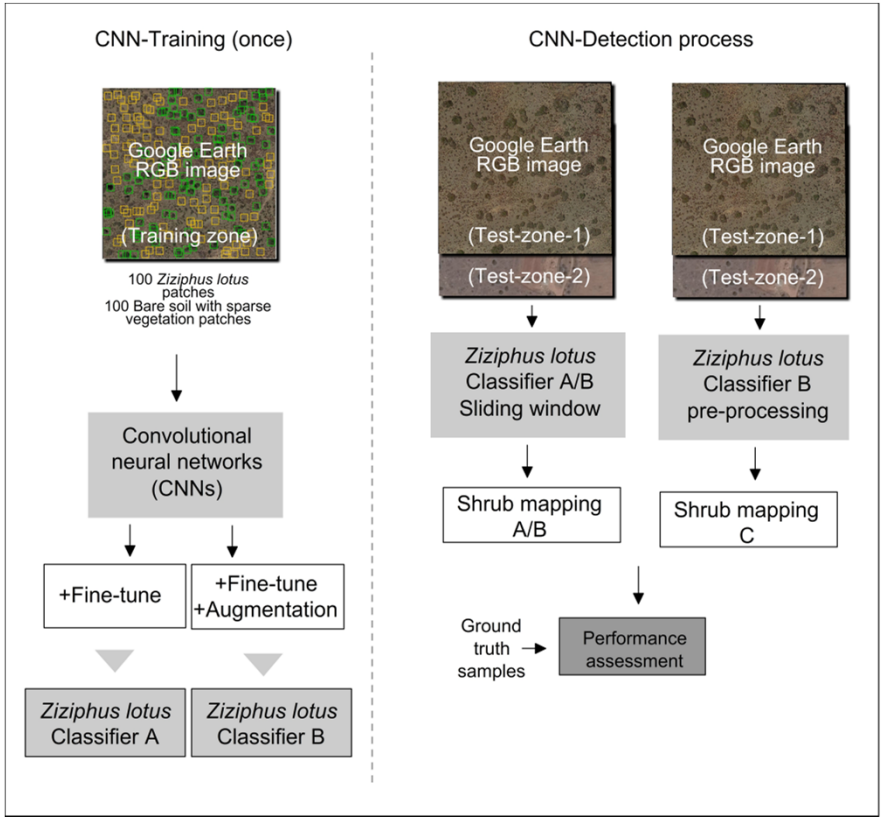


Case study 2: Protected plant species detection using Google-earth imagery

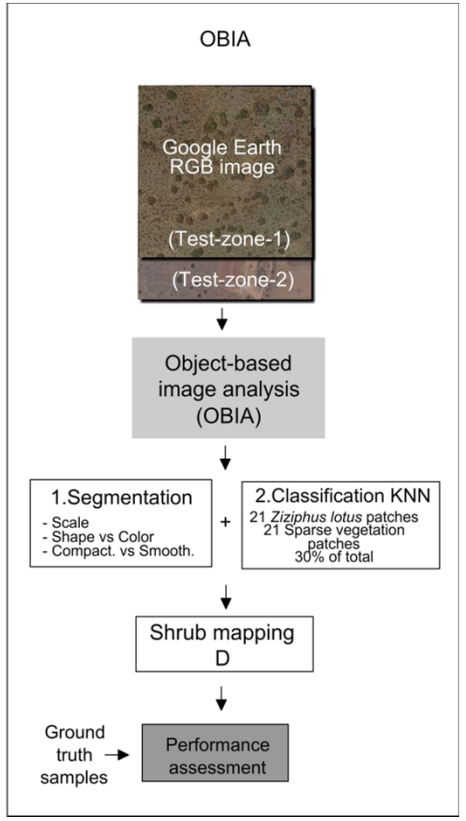
- Database for training the CNNs
- A two-class problem



Case study 2: Protected plant species detection using Google-earth imagery



VS



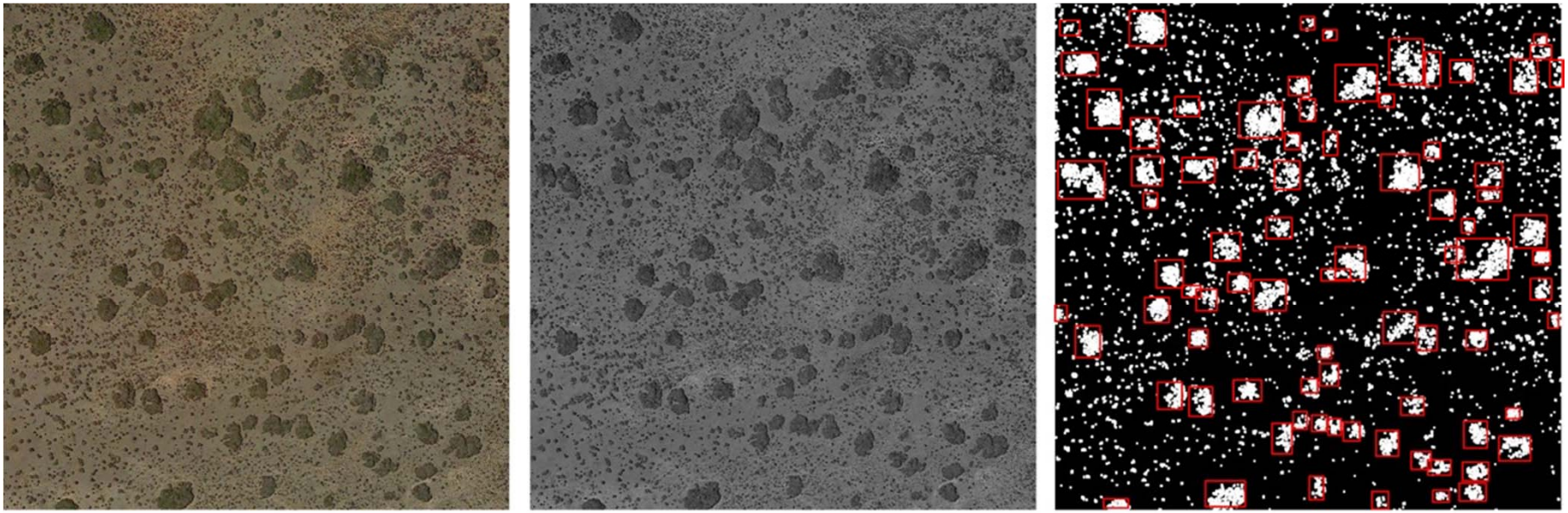
$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives},$$

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives},$$

$$F1\ measure = 2 \times \frac{precision \times recall}{precision + recall}$$

Case study 2: Protected plant species detection using Google-earth imagery

Preprocessing:



1. RGB image to PAN image

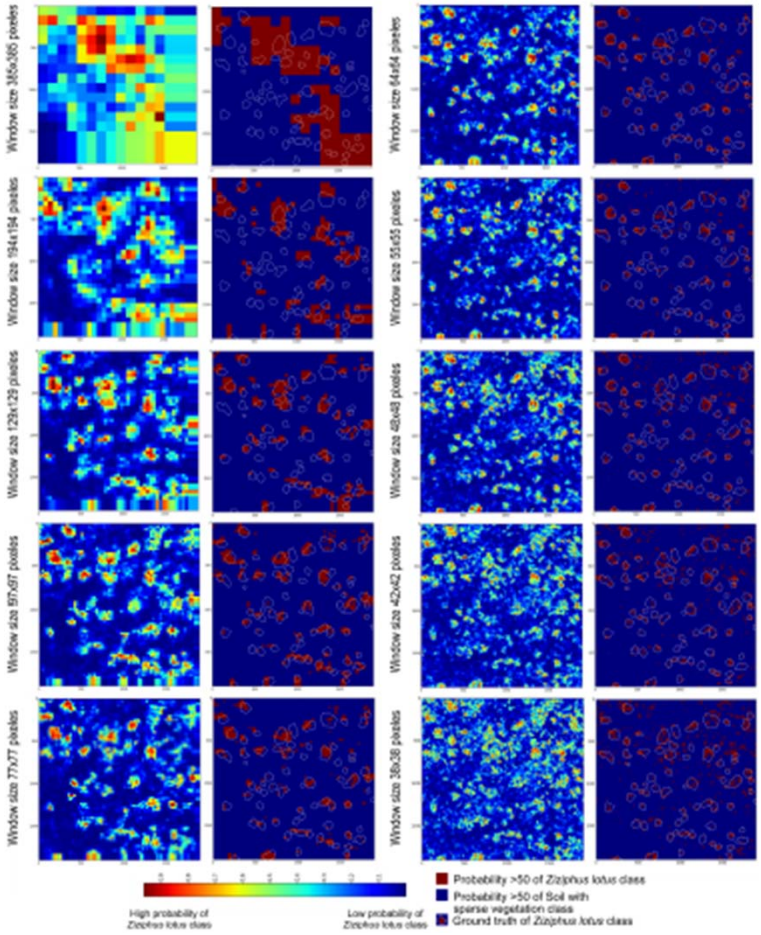
2. PAN image to gray threshold image

3. Gray threshold image to contour threshold candidate patches

(Pixel value<100)

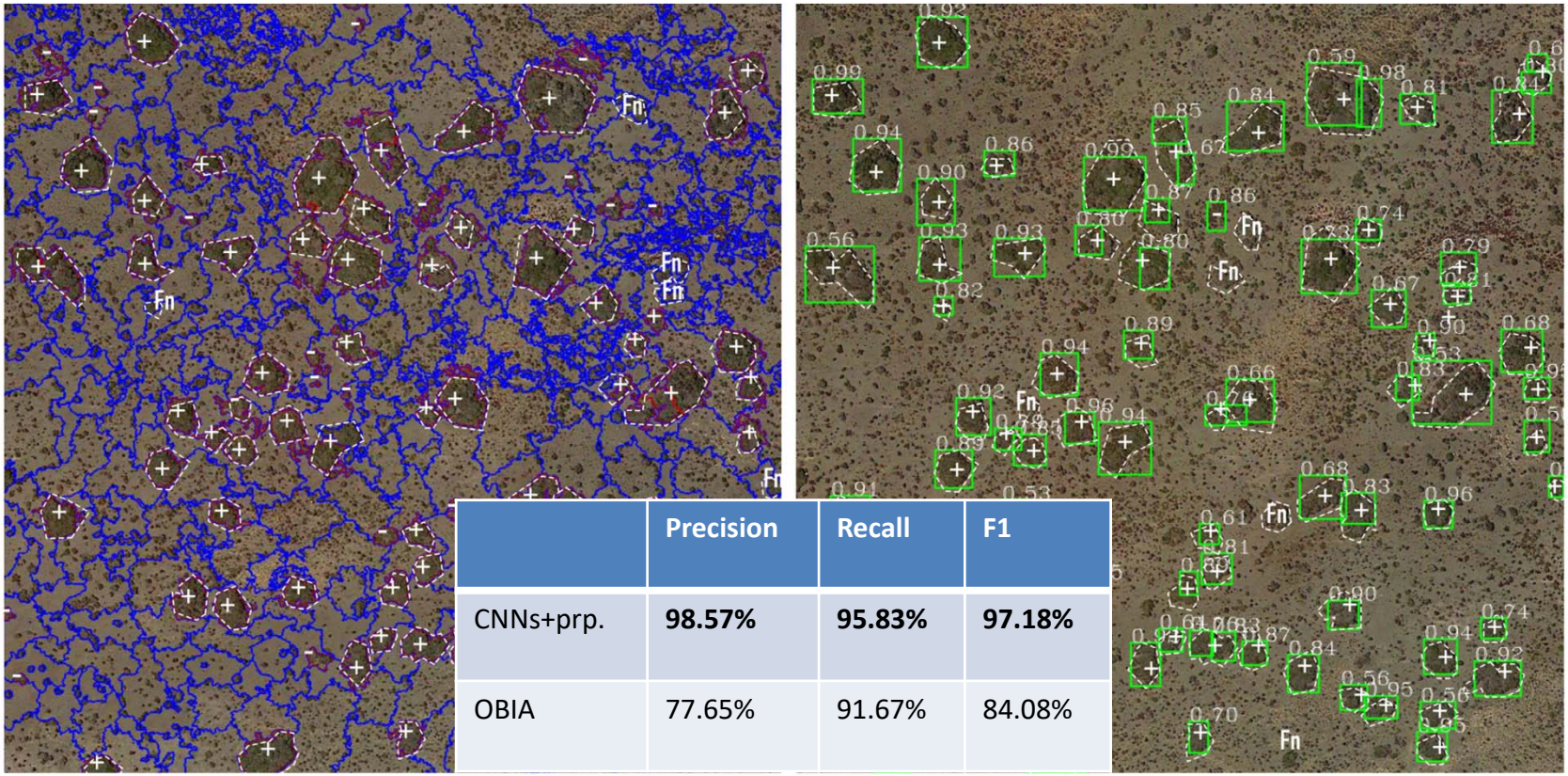
Case study 2: Protected plant species detection using Google-earth imagery

- Results (GoogLeNet+sliding window)
- It takes: 291 min



Case study 2: Protected plant species detection using Google-earth imagery

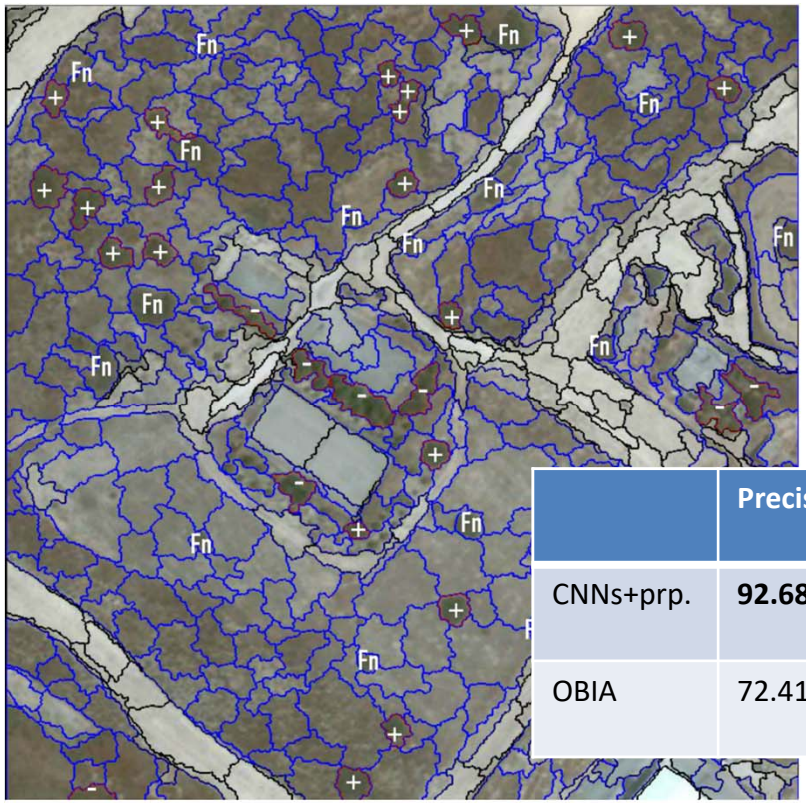
Results: OBIA(12 hours) vs CNNs(35.4seconds)

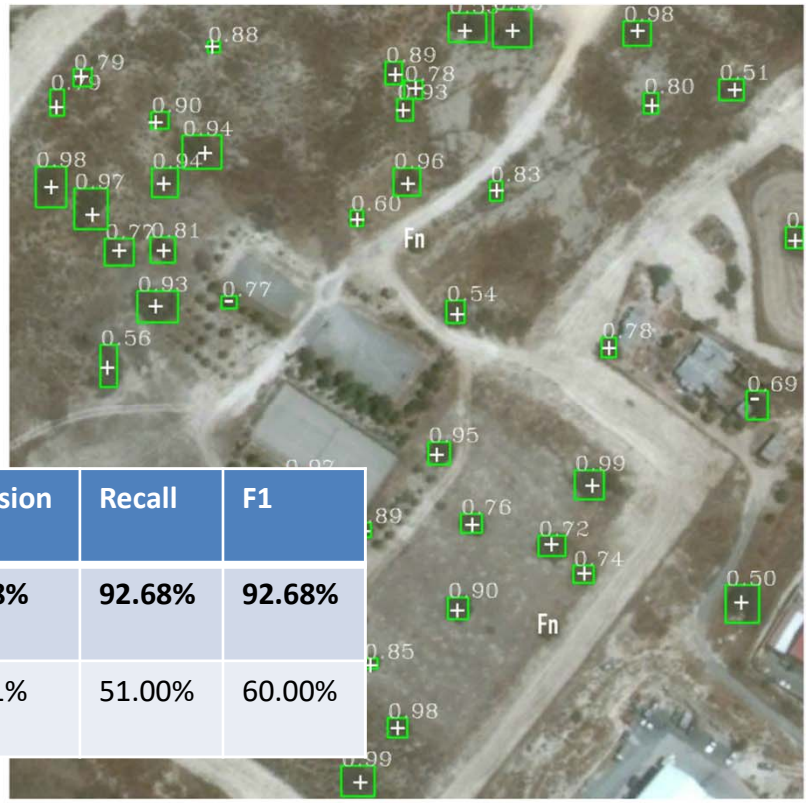


	Precision	Recall	F1
CNNs+prp.	**98.57%**	**95.83%**	**97.18%**
OBIA	77.65%	91.67%	84.08%

Case study 2: Protected plant species detection using Google-earth imagery

Results: OBIA(12 hours) vs CNNs(24.1seconds)





	Precision	Recall	F1
CNNs+prp.	**92.68%**	**92.68%**	**92.68%**
OBIA	72.41%	51.00%	60.00%