AGENDA

GENERAL IDEA
The origin of CNNs

CONVOLUTION
How does a CNN work?

CNN ARCHITECTURE
Example VGGnet

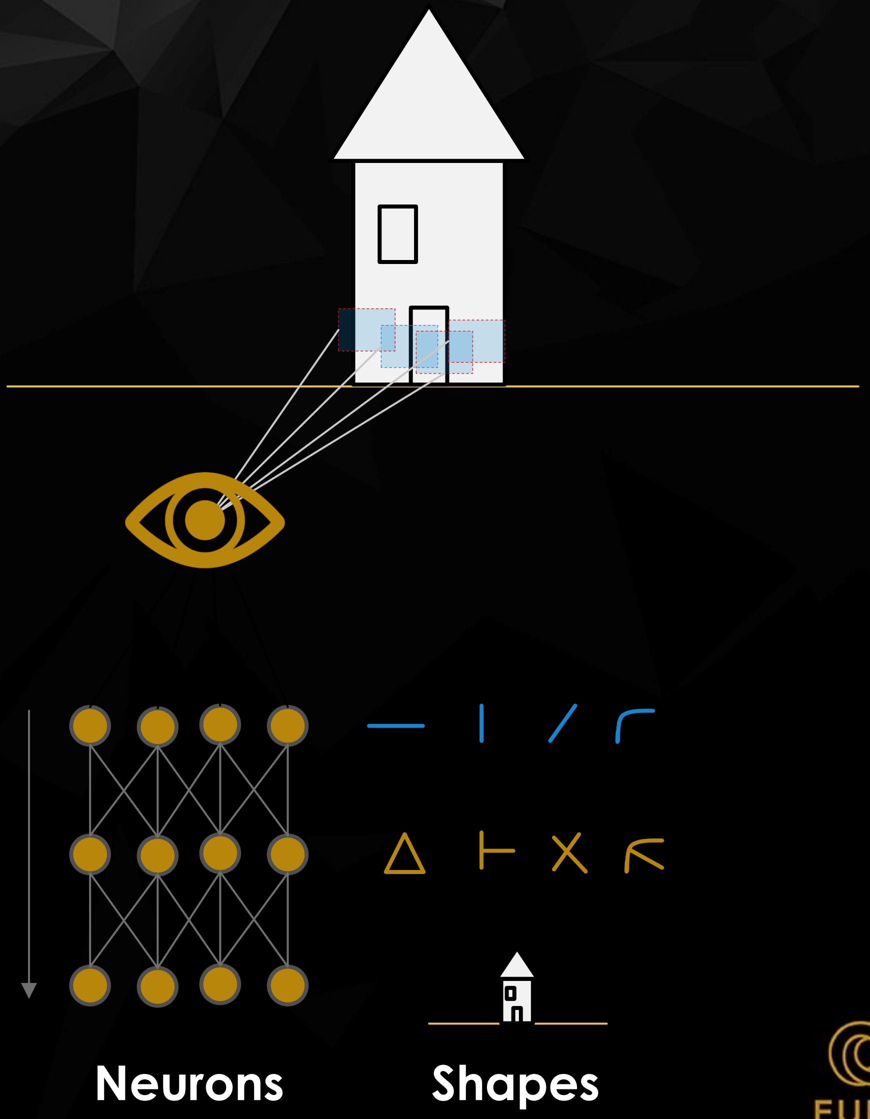
EURO AUSTRIA

GENERAL IDEA

THE ORIGIN OF CNNS

THE ORIGIN OF CNNS

- USED FOR IMAGE RECOGNITION SINCE 1980S
- **INSPIRED BY THE BRAIN'S VISUAL CORTEX**
  - NEURONS IN VISUAL CORTEX HAVE A **SMALL LOCAL RECEPTIVE FIELD**
  - RECEPTIVE FIELDS OF DIFFERENT NEURONS **OVERLAP**
  - TOGETHER THEY TILE THE **WHOLE VISUAL FIELD**
  - SOME NEURONS ONLY REACT TO **SPECIFIC SHAPES**
  - SOME NEURONS REACT TO MORE **COMPLEX SHAPES FROM LOWER LEVELS**
- POWERFUL ARCHITECTURE OF LOWER AND HIGHER-LEVEL NEURONS TO DETECT COMPLEX PATTERNS



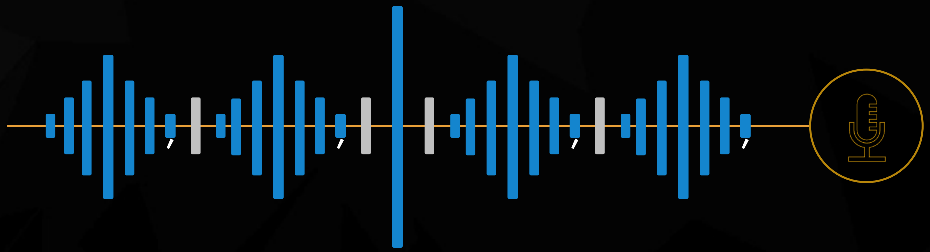
Source: Géron (2019)

EURO AUSTRIA

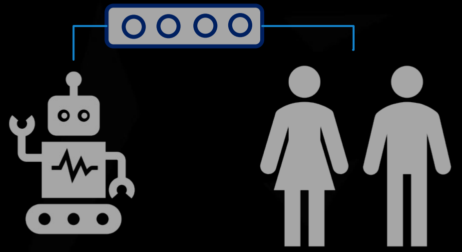
FIELDS OF APPLICATIONS

- IMAGE DETECTION

- VOICE RECOGNITION



- NATURAL LANGUAGE PROCESSING (NLP)



CONVOLUTION

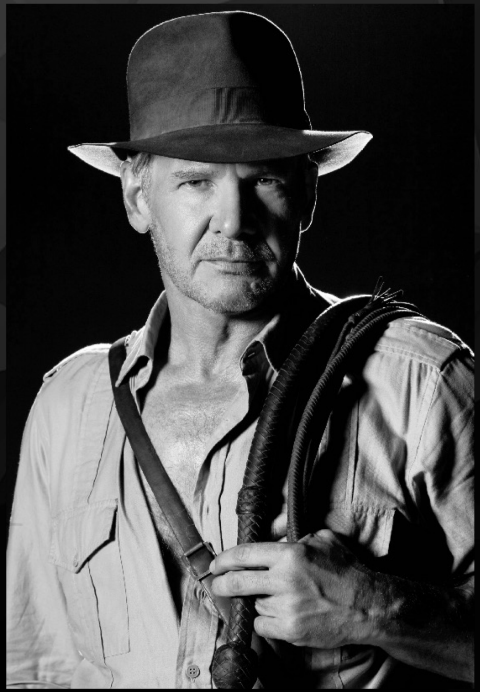
HOW A CNN WORKS

EURO AUSTRIA

HOW DOES A CNN WORK?

Image preprocessing → Convolution → Filters → Pooling → Activation

WHAT DOES A COMPUTER SEE?



Input image
3600 x 2400

Resized image
30 x 30

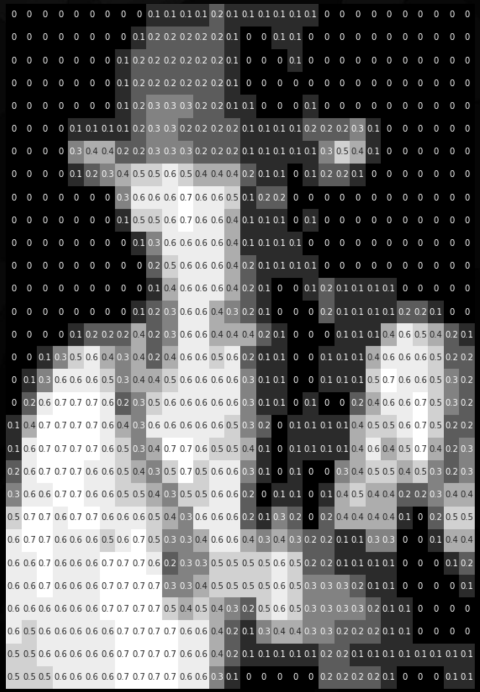
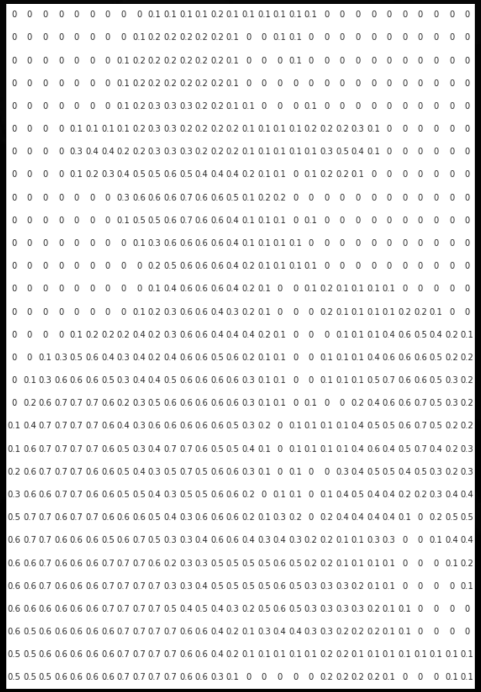


Image as matrix
of numbers [0, 1]

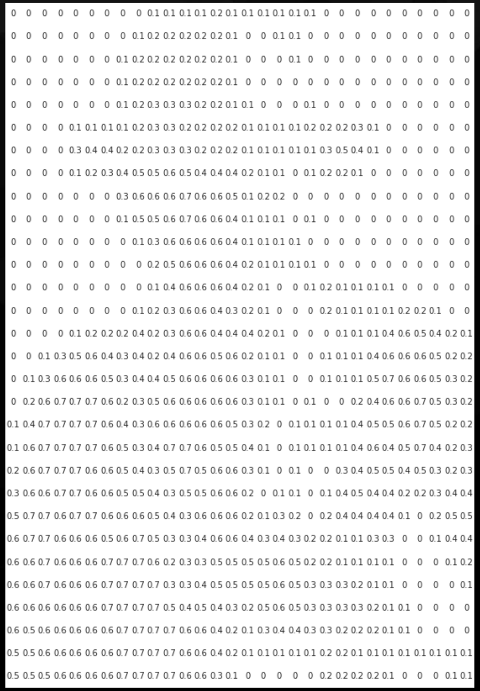


Matrix of numbers
[0, 1]
(900 values)

Source:
https://www.youtube.com/watch?v=iaSUYvmCekI

TASK IN COMPUTER VISION

Input image



Matrix of numbers

Classificatio
n

Harrison Ford [0.8]

Sean Connery [0.1]

Roger Moore [0.05]

Tom Cruise [0.05]

Prediction

WHY NOT SIMPLY USE A DEEP NETWORK WITH FULLY CONNECTED LAYERS?

Indie resized to 100 x 100

The picture has 10.000 pixels

With a **1.000 neuron** input layer …

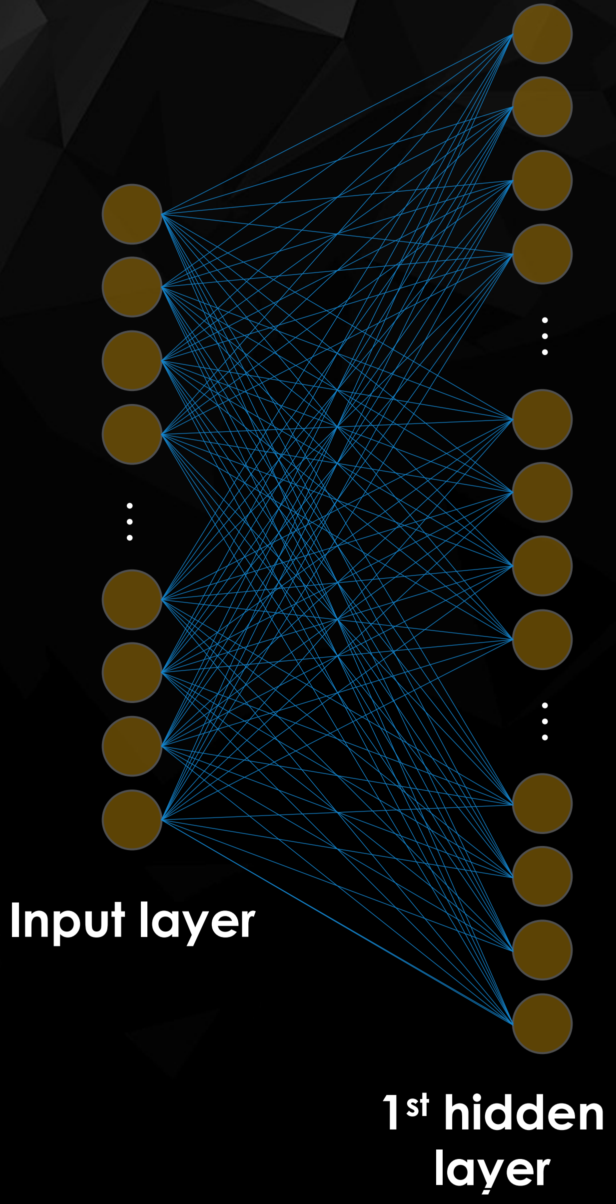
and a **fully connected** 1st hidden layer, …

this first operation amounts to a total of **10 million** connections (weights, parameters).

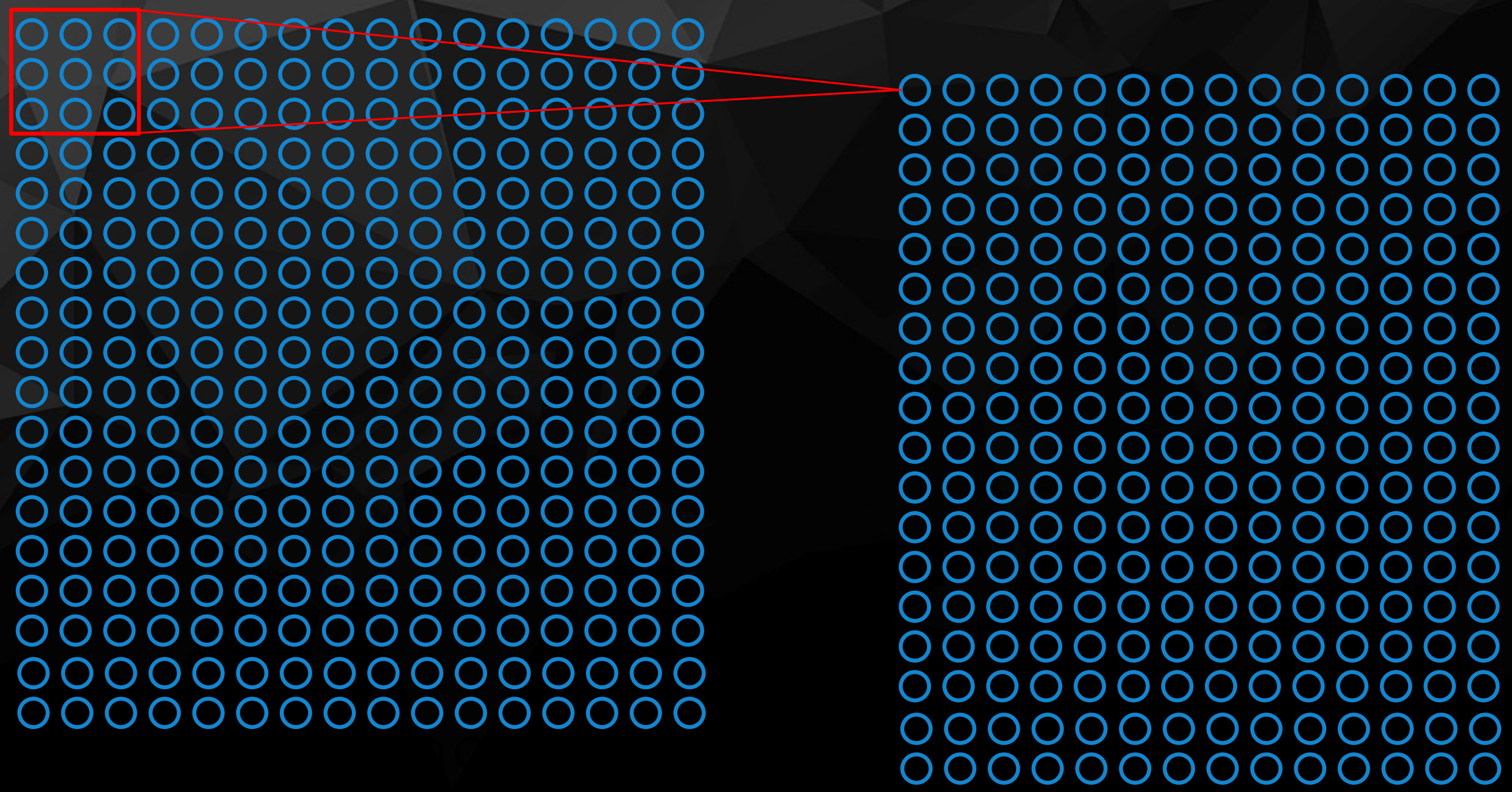
And that is just the **1st** layer!

For **large images,** a deep neural network breaks down.

AND we do not capture the spatial information of the pixels



THE CONVOLUTION LAYER – USING SPATIAL STRUCTURE



Input image
1 neuron for 1 pixel

Hidden layer
1 neuron for all pixels
in **receptive field** (filter)

Idea: connect smaller sections of the input image to respective neuron in the hidden layer.

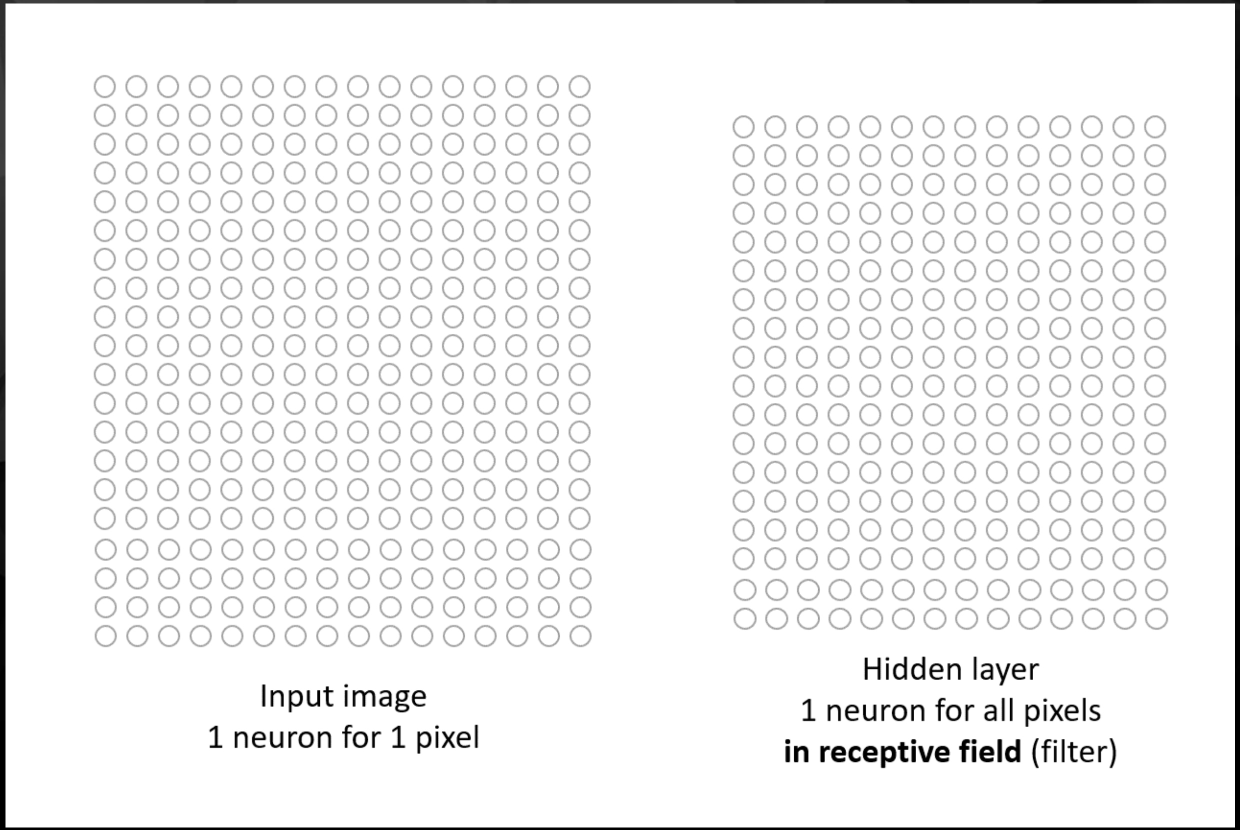
Receptive field (**FILTER**) marks a specific area in the input image.

Use a **sliding window** to define the connections.

The GOAL: How to ***weight*** the **FILTER** to detect particular features in the image?

Convolution

CONVOLUTION



- Apply a **set of weights** – a **filter** – to extract local features
- Use **multiple filters** to extract different features
- **Spatially share parameters** of each filter

ELEMENT-WISE MULTIPLY AND ADD THE OUTPUTS

$$\begin{bmatrix} 1 & 3 \\ 5 & 2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = 19$$

Part of input image

Filter

$$(1 * 1) + (3 * 2) + (5 * 2) + (2 * 1) = \mathbf{19}$$

Filters

APPLICATION OF A FILTER

Convolutional layer: Connection between neurons and only those pixels within their ***receptive field***.

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

Image

$\otimes$

1	0	1
0	1	0
1	0	1

Filter

$=$

4	**3**	**4**
2	**4**	**3**
2	**3**	**4**

Feature map

$$(1*1)+(1*0)+(1*1)+(0*0)+(1*1)+(1*0)+(0*1)+(0*0)+(1*1)=\color{red}{4}$$

$$(1*1)+(1*0)+(0*1)+(1*0)+(1*1)+(1*0)+(0*1)+(1*0)+(1*1)=\color{red}{3}$$

Filters

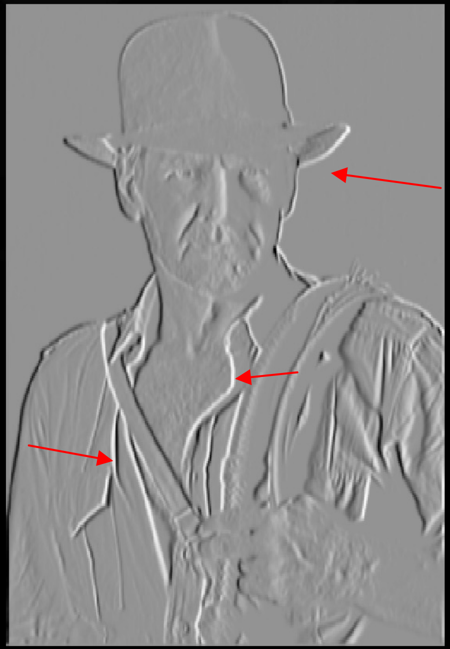
APPLYING DIFFERENT FILTERS

Horizontal edge detection

-1	-2	-1
0	0	0
1	2	1

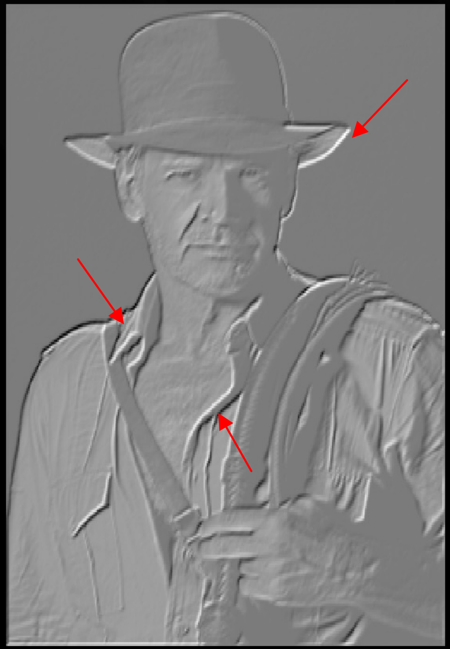
Vertical edge detection

-1	0	1
-2	0	2
-1	0	1



Mixed edge detection

0	-2	0
-2	1	2
0	2	0



Filters

PADDING

Adding additional space to ***preserve the same height and width of previous layer.***

Zero padding

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

Image

$\otimes$

1	0	1
0	1	0
1	0	1

Filter

$=$

2	2	3	1	1
1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
0	2	2	1	1

Feature map

Also, different kinds of paddings possible (i.e., One padding).

Highlight pixels at the edges of image.

Filters

USING A LARGER STRIDE

The shift from one receptive field to the next one is called ***stride***.

Stride = 2

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

Image

$\otimes$

1	0	1
0	1	0
1	0	1

Filter

$=$

4	**4**
2	**4**

Feature map

Connect larger input to smaller layer.

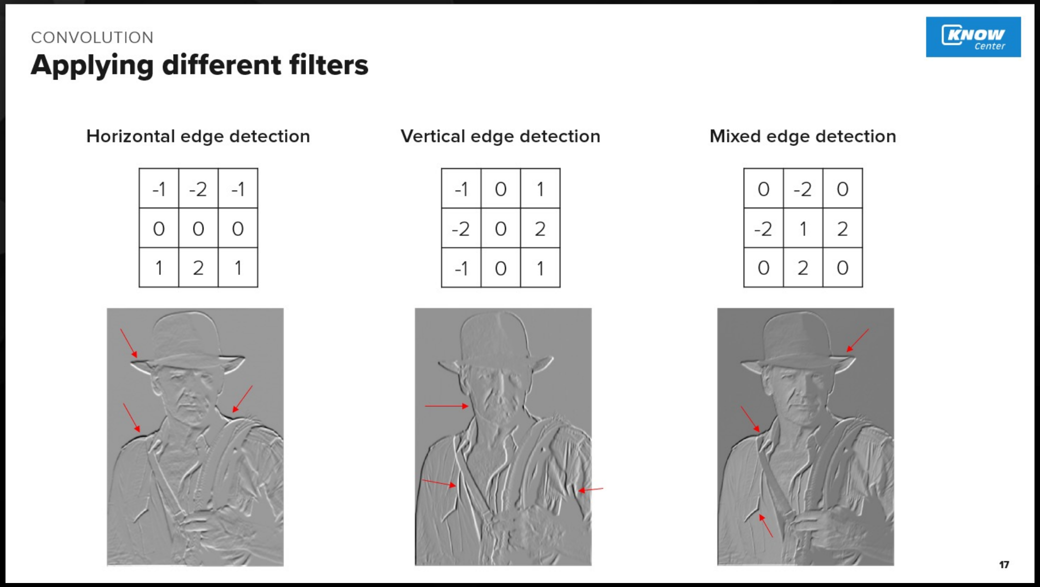
Reduction of the model's computational complexity.

Filters

EURO AUSTRIA

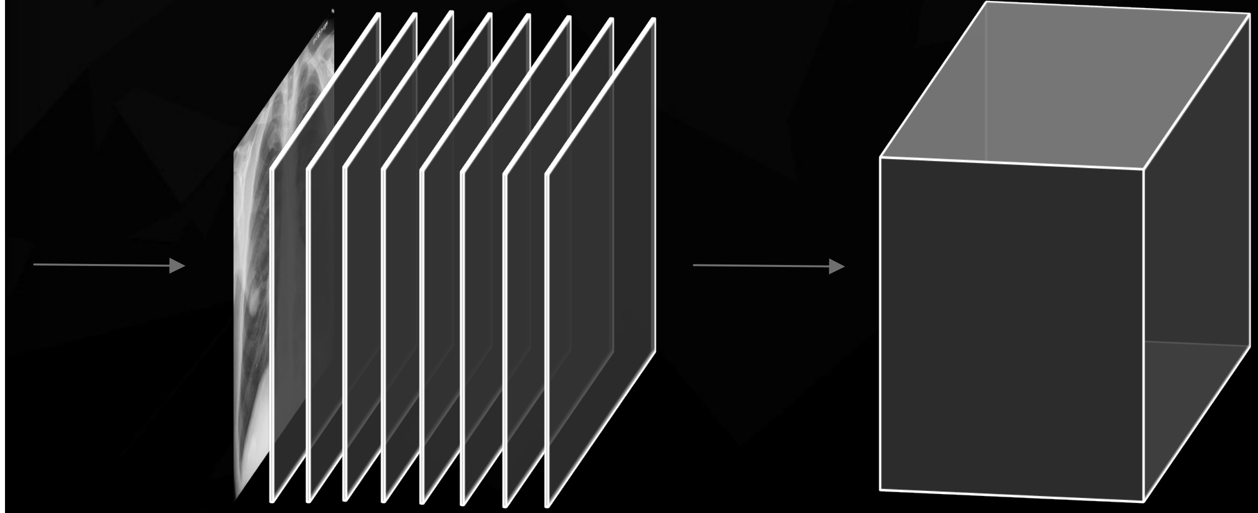
STACKING MULTIPLE FEATURE MAPS

Applying different filters …



… results in different feature maps.

- Convolutional layers with multiple feature maps
- Representation in 3D



-1	-2	-1
0	0	0
1	2	1

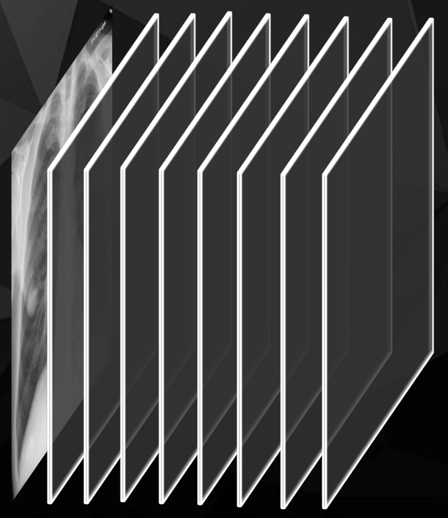
-1	0	1
-2	0	2
-1	0	1

0	-2	0
-2	1	2
0	2	0

…

Filters

HOW DOES A CNN LEARN THE FILTERS?



-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

0	-2	0
-2	1	2
0	2	0

…

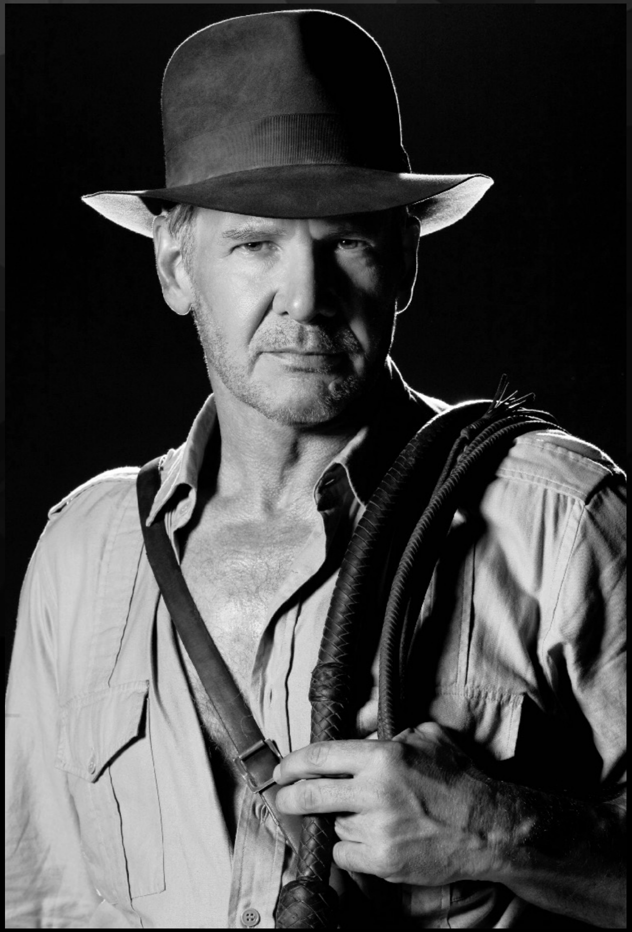
How are all these filters determined or learned?

- NUMBER AND SIZE OF FILTERS IN EACH CONVOLUTION LAYER ARE SET **BY DESIGN**
- FILTERS INITIALIZED AT RANDOM AND THEN LEARNED (FWD PASS, BACKWARD PROP)
- CONVOLUTIONAL LAYER LEARNS MOST USEFUL FILTERS **AUTOMATICALLY** DURING TRAINING FOR **ITS TASK**
- LAYERS AFTER THIS WILL LEARN TO **COMBINE** THEM TO MORE **COMPLEX PATTERNS**

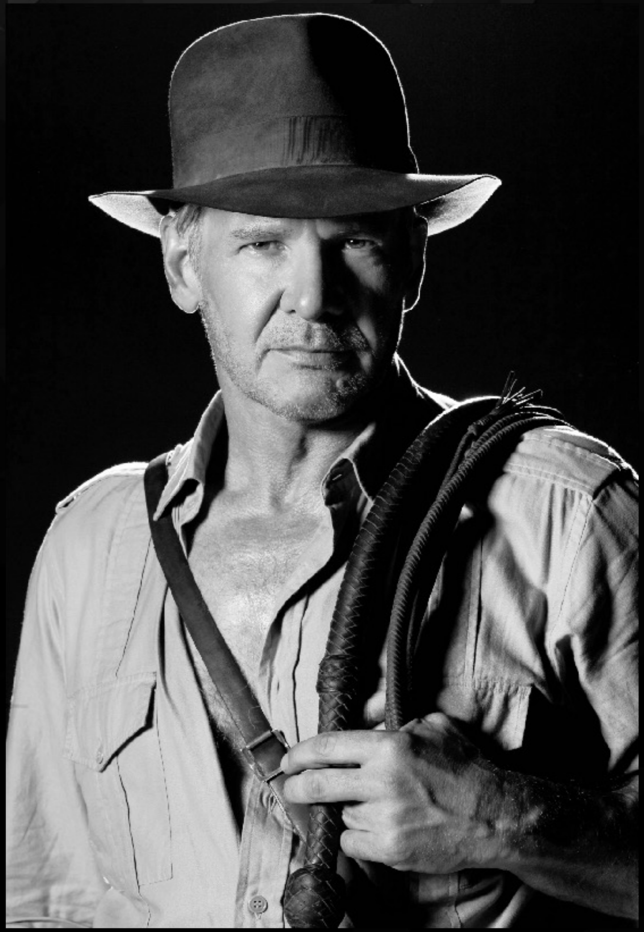
EURO AUSTRIA

CAN YOU SPOT THE DIFFERENCE?

Image subsampled by
POOLING



Full sized image



EURO AUSTRIA

Pooling

POOLING LAYERS

Subsample (i.e., **shrink**) the input image to reduce the computational load (memory usage, number of parameters)

Stride = 2
Filter = 3x3

1	1	2	0	4
0	1	7	1	0
0	8	1	1	1
9	0	1	3	5
0	4	1	0	0

Image

Max pooling

8	**7**
9	**5**

Feature map

Set filter size, stride and padding as for convolution layer.

No weights attached.

The layer aggregates input with an aggregation function (i.e., max or mean).

Pooling

ACTIVATION BY NON-LINEARITY

- APPLY AFTER EVERY CONVOLUTION OPERATION
- RECTIFIED LINEAR UNIT (RELU)
- $f(x) = \text{MAX}(0, x)$
- PIXEL-BY-PIXEL OPERATION THAT REPLACES ALL NEGATIVE VALUES BY **ZERO**

Vertical edges activated

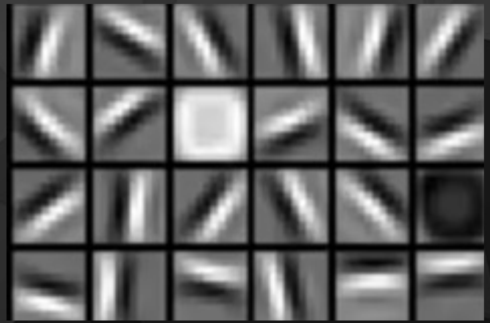
EURO AUSTRIA

Activation

CNN ARCHITECTURE

EXAMPLE VGGNET

FROM BASIC TO DETAILED FEATURES

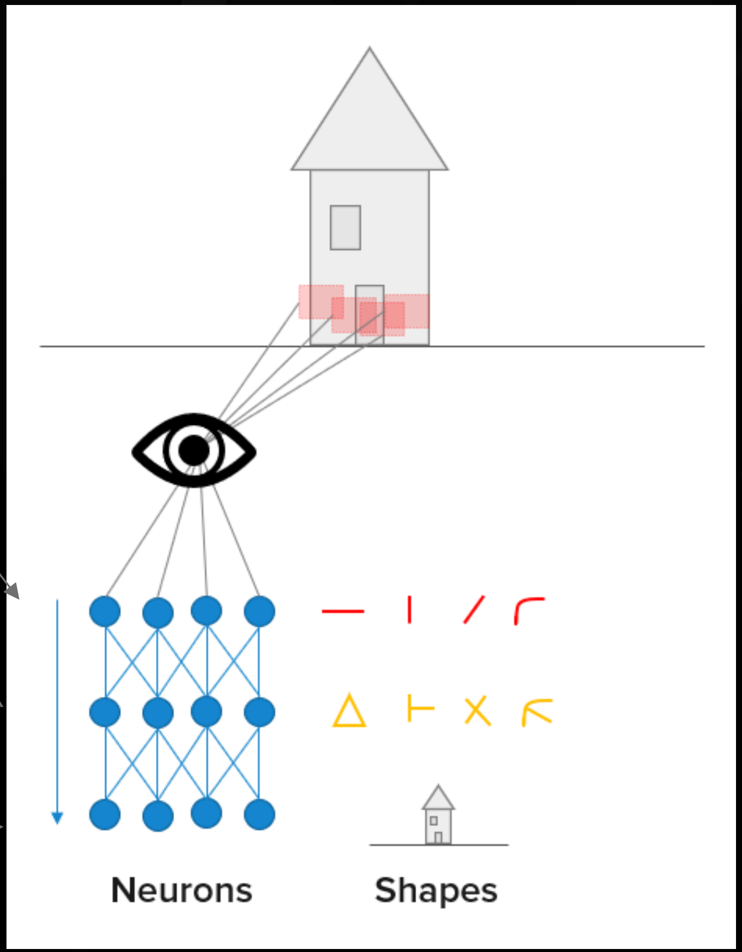


Low level features
(1st Conv Layers)

Mid level features
(...)

High level features
(Last Conv Layers)

Remember the visual cortex?



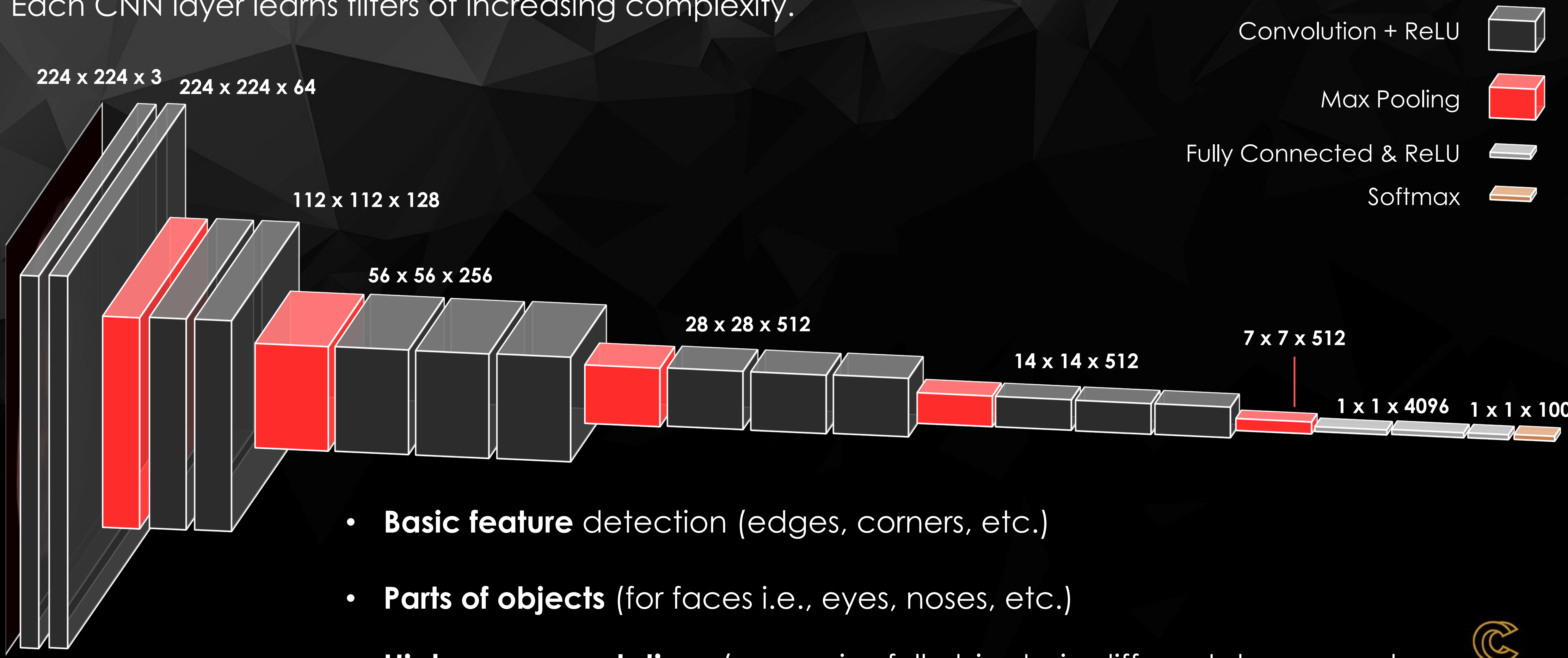
VGGNET

- Invented by Simonyan and Zisserman from Visual Geometry Group (VGG) at University of Oxford in 2014 [1]
- Large Scale Visual Recognition
- Fixed filter size of 3×3 and the stride of 1
- Different versions (VGG16, VGG19, etc.)
- **Why?** Reduce the # of parameters in the CONV layers and improve on training time.

[1] K. S. a. A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in International Conference on Learning Representations (ICLR), San Diego, 2015.

VGGNET 16

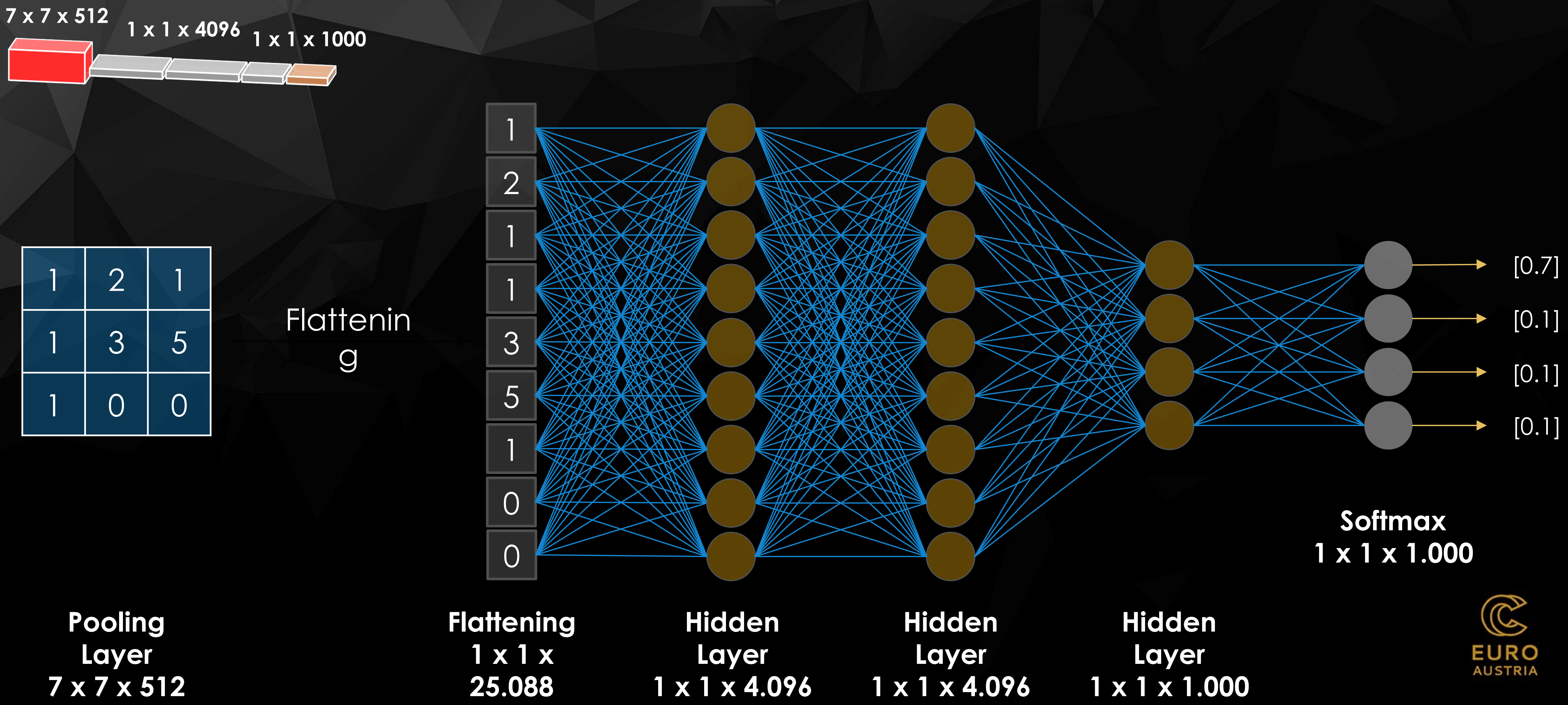
Each CNN layer learns filters of increasing complexity.



- **Basic feature** detection (edges, corners, etc.)
- **Parts of objects** (for faces i.e., eyes, noses, etc.)
- **Higher representations** (recognize full objects, in different shapes and positions)

EURO AUSTRIA

A CLOSER LOOK ON FINAL PREDICTION



EURO AUSTRIA