Introduction to Deep Learning. Convolutional Neural Networks.

Elli Valla

PhD student and junior researcher at

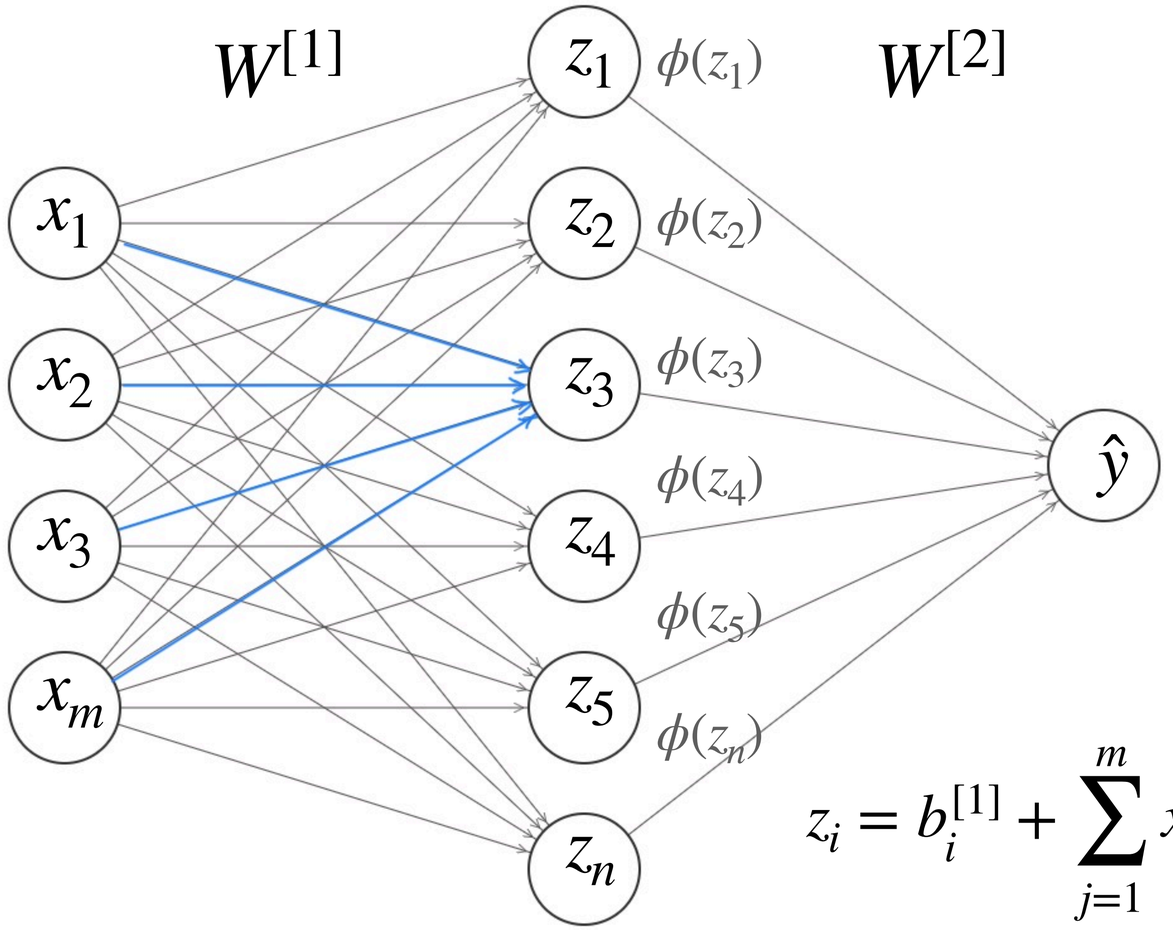
Department of Software Science
TalTech University

elli.valla@taltech.ee

Plan for today:

1. Simple Neural Network recap
2. Deep Learning Background and Applications
3. Convolutional Neural Network
    - 3.1 Convolutional layers
        - 3.1.1 - 3.1.3 Padding, Strides, Pooling
    - 3.2 Architectures
4. MATLAB Deep Learning ToolBox examples

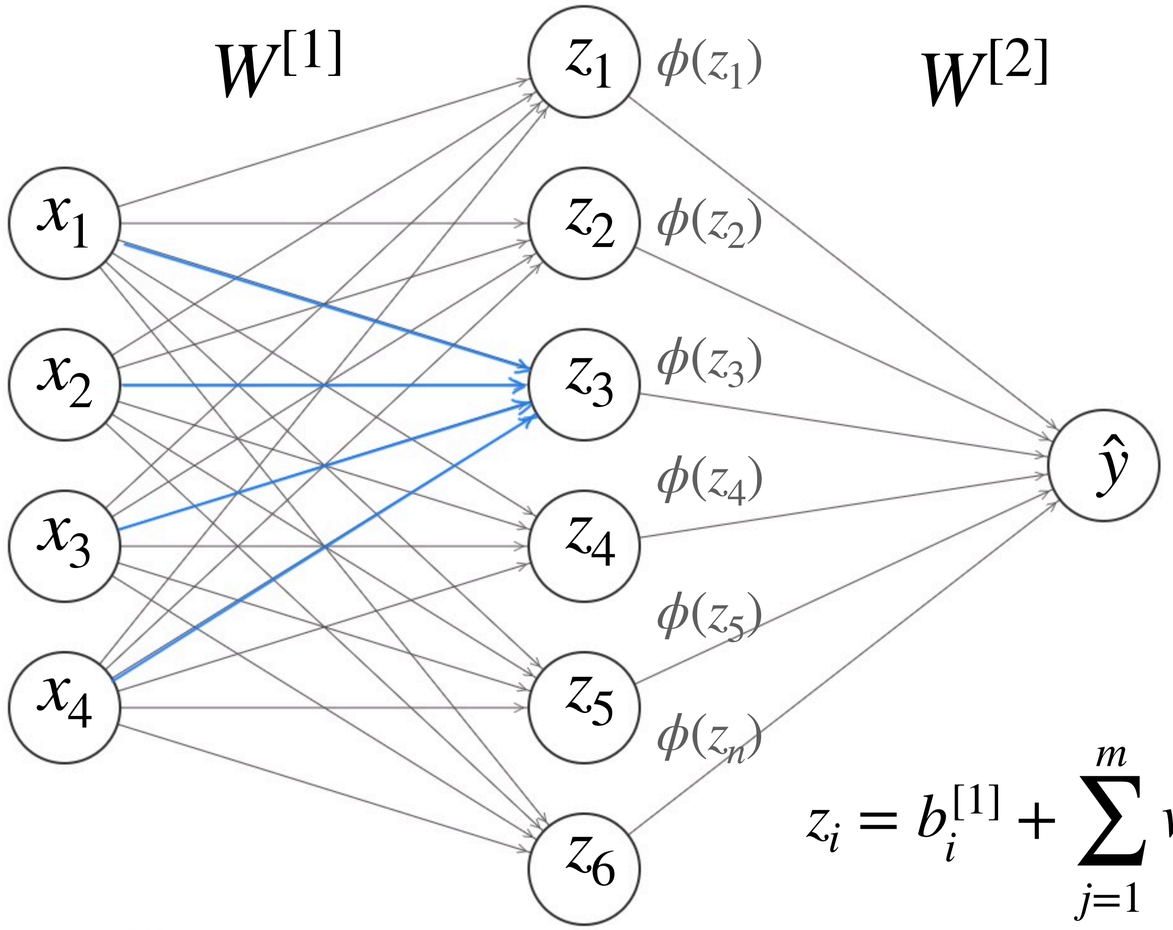
1. Simple Neural Network recap



$$\hat{y} = b_i^{[2]} + \sum_{j=1}^{n} \phi(z_j) w_{j,i}^{[2]}$$

$$z_i = b_i^{[1]} + \sum_{j=1}^{m} x_j w_{j,i}^{[1]}$$

1. Simple Neural Network recap



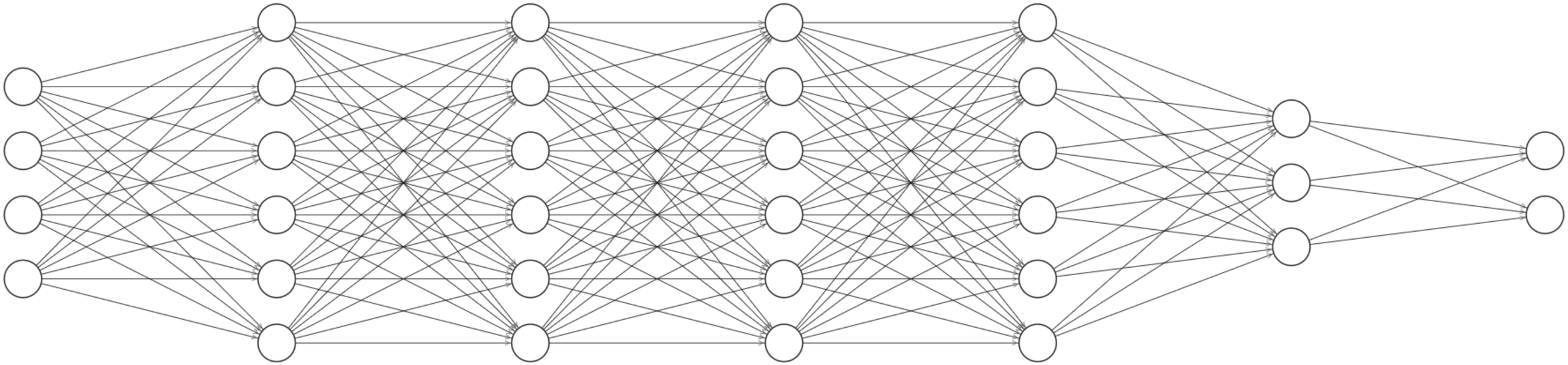
$$\hat{y} = b^{[2]} + \sum_{j=1}^{6} w_j^{[2]} \phi(z_j)$$

$$z_i = b_i^{[1]} + \sum_{j=1}^{m} w_{j,i}^{[1]} x_j$$

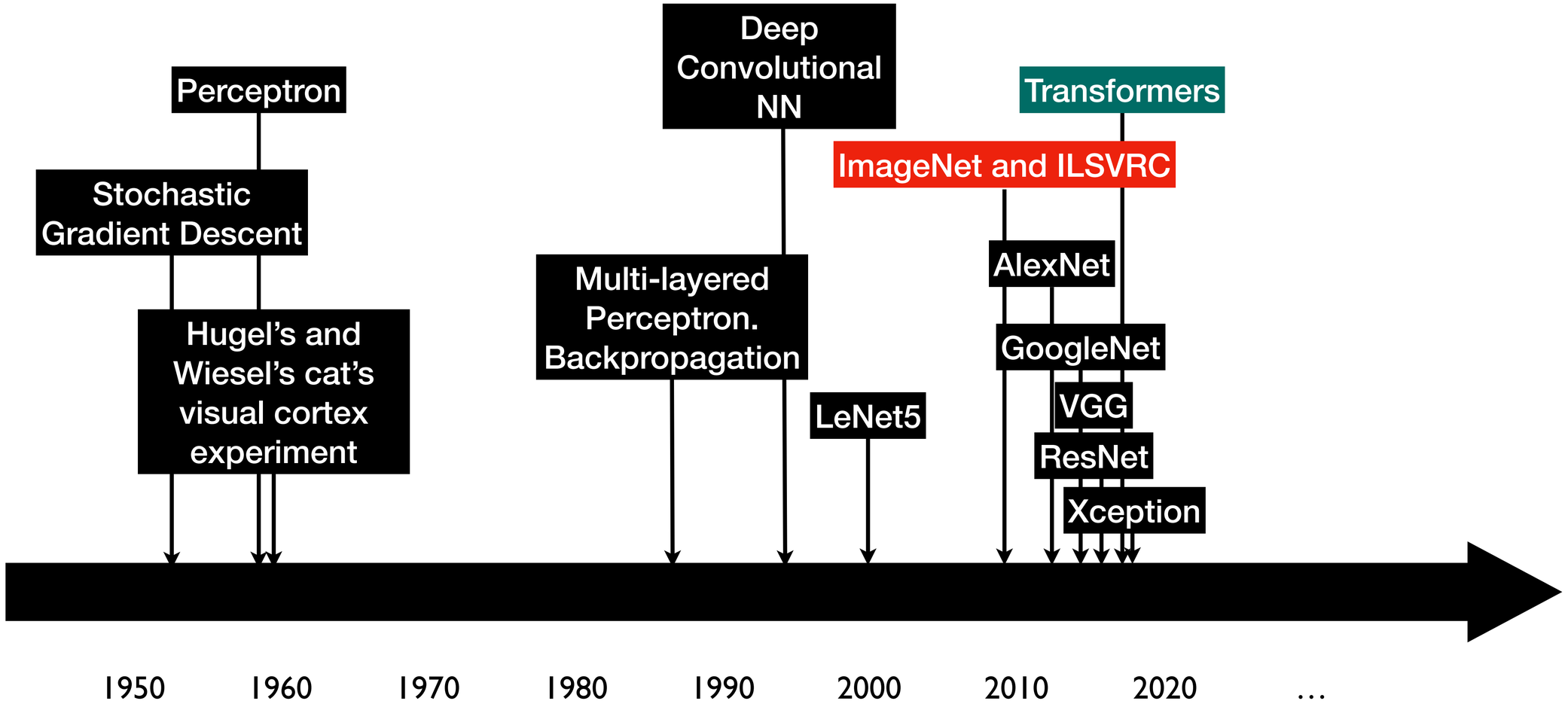
$$z_3 = b_3^{[1]} + \sum_{j=1}^{4} w_{j,3}^{[1]} x_j =$$

$$= b_3^{[1]} + w_{1,3}^{[1]} x_1 + w_{2,3}^{[1]} x_2 + w_{3,3}^{[1]} x_3 + w_{4,3}^{[1]} x_4$$

2. Deep Learning Background and Applications



2. Deep Learning Background and Applications



Why now?

1. Big Data
   - ImageNet, CIFAR10, MNIST
   - Data collection and storage
2. Hardware
   - Graphics Processing Units (GPUs)
3. Software

2. Deep Learning Background and Applications

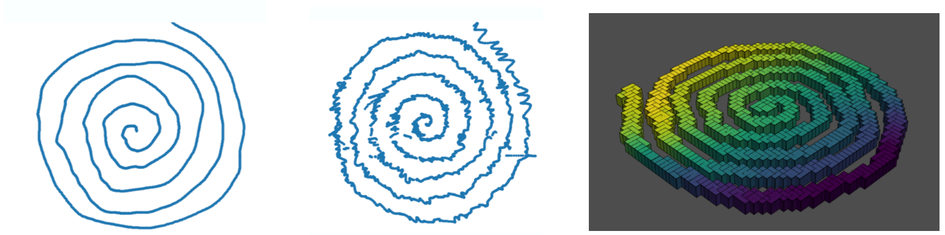
Robotics

2. Deep Learning Background and Applications

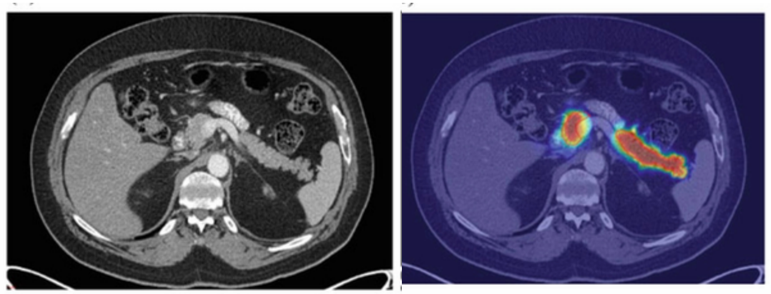
Autonomous Driving

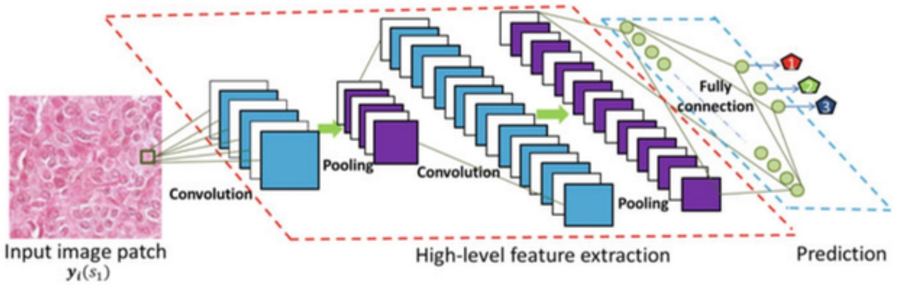
2. Deep Learning Background and Applications

Biology & Medicine

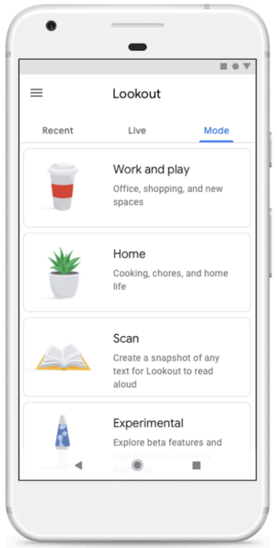


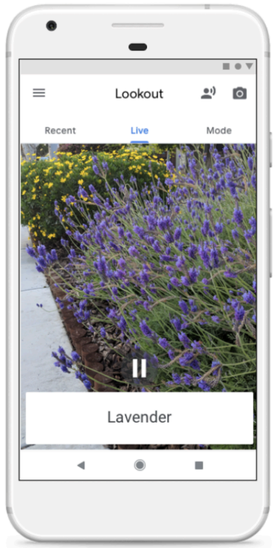
Parkinson's disease diagnostics

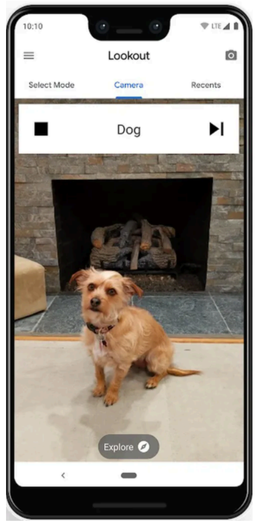




Accessibility



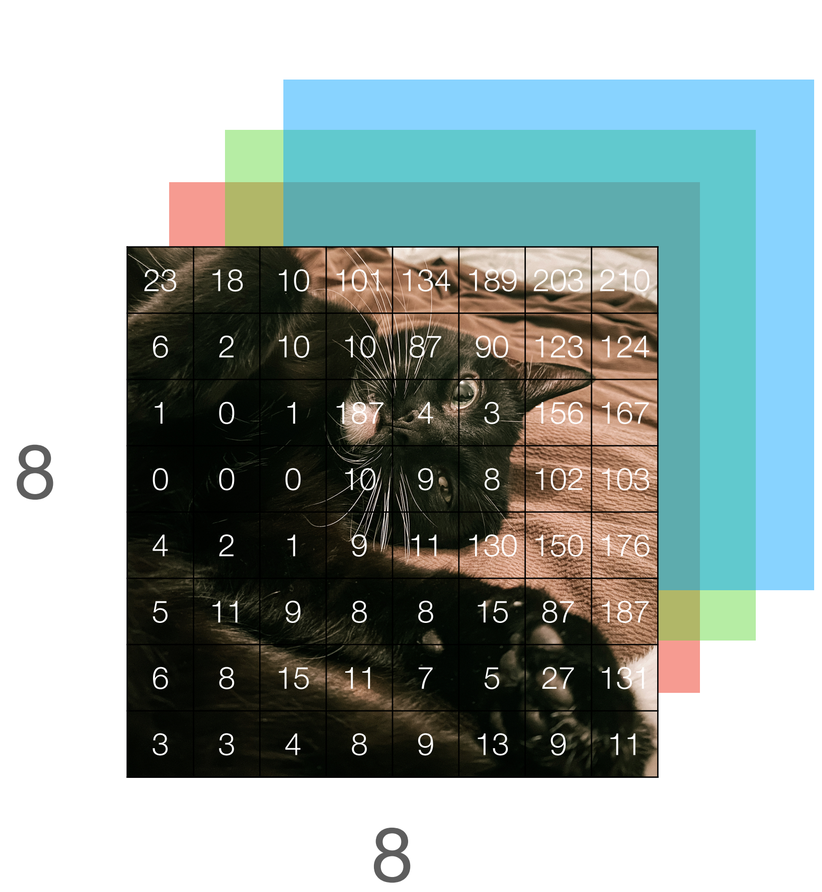




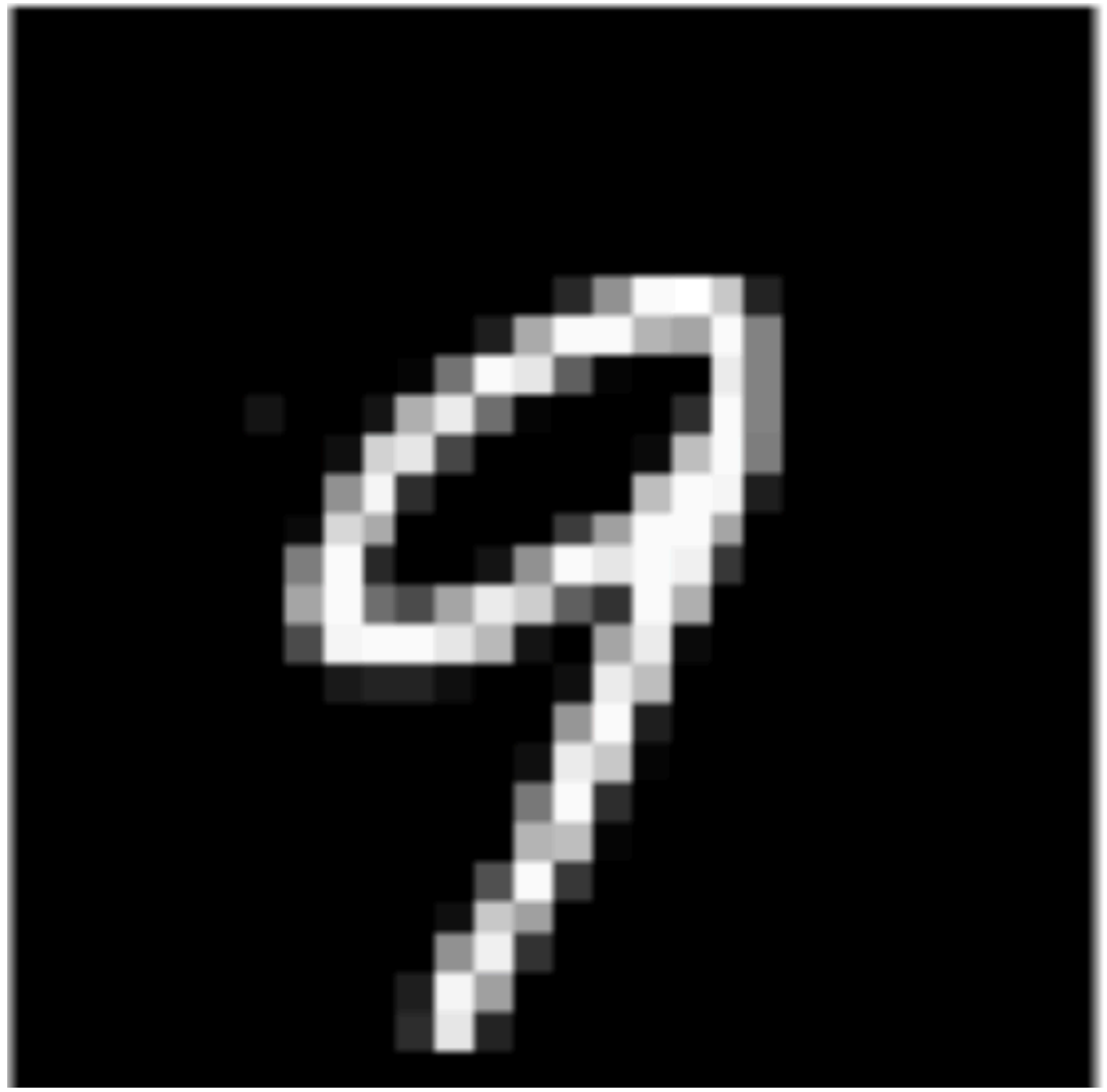
"Deep Learning and Convolutional Neural Networks for Medical Image Computing", Le Lu et al, Springer 2017

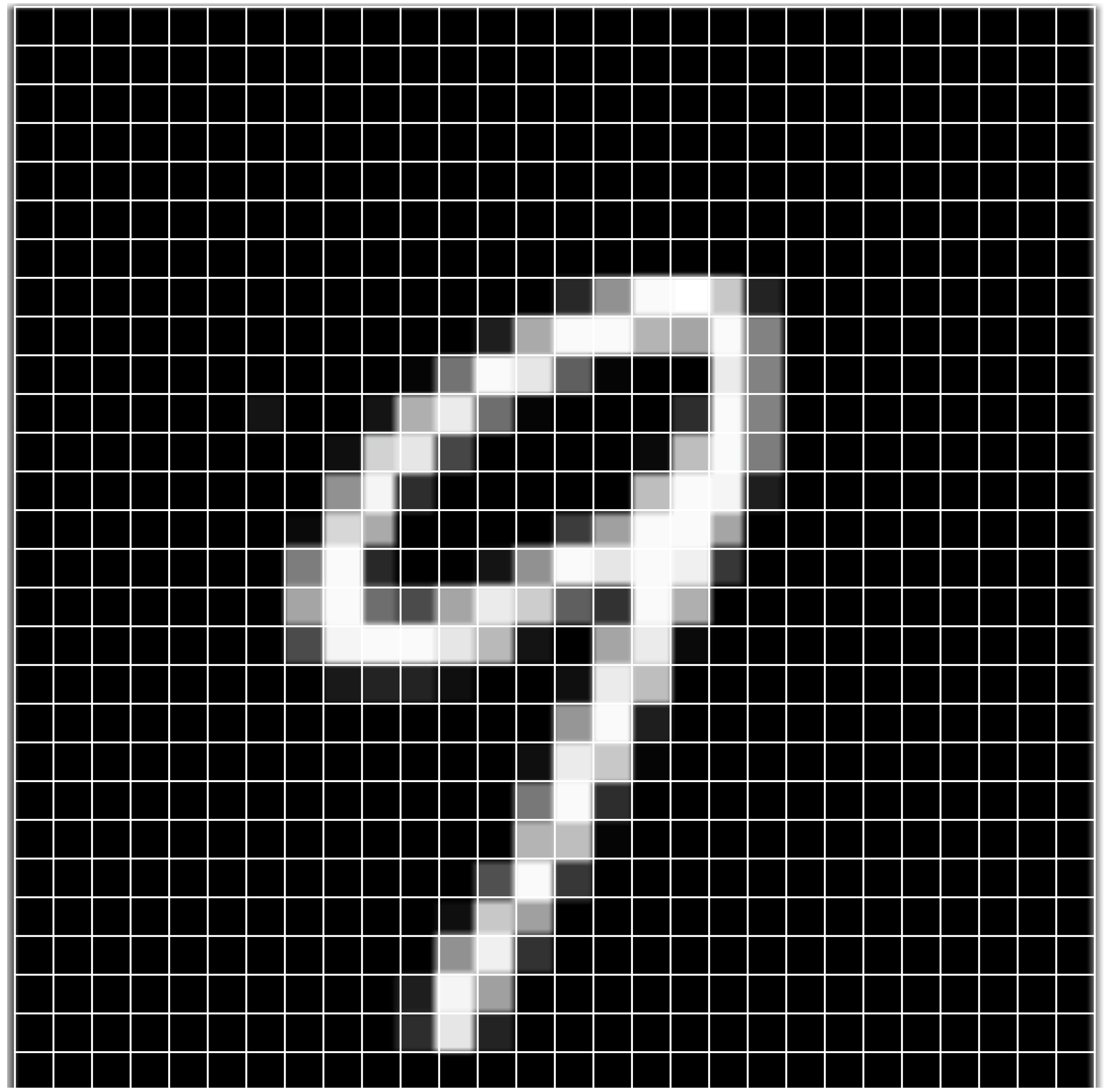
How do computers "see"?

- An image is just a matrix (tensor) of numbers [0, 255].
- In this example it's 8x8x3



How do computers "see"?

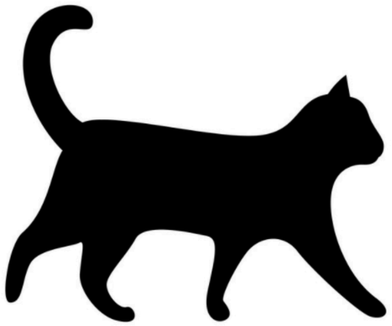


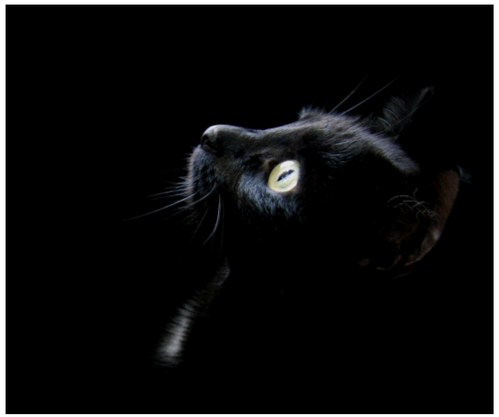


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	215	255	255	250	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	210	255	255	199	199	255	210	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	210	255	249	253	0	0	0	255	210	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	270	255	0	0	0	0	0	0	255	210	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	250	255	0	0	0	0	0	0	0	255	210	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	255	255	0	0	0	0	0	0	0	255	255	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	199	0	0	0	0	0	0	0	255	255	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	210	255	0	0	0	0	0	0	0	255	255	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	199	255	130	70	190	255	250	130	20	255	230	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	120	255	255	255	250	230	0	0	230	250	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	250	230	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	190	255	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	251	230	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	190	255	5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	190	230	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	130	255	30	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	20	240	190	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	190	255	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	255	230	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Can we extract features manually?

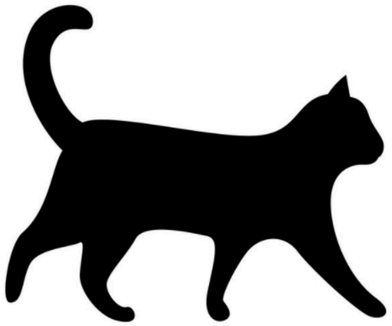
Define features!

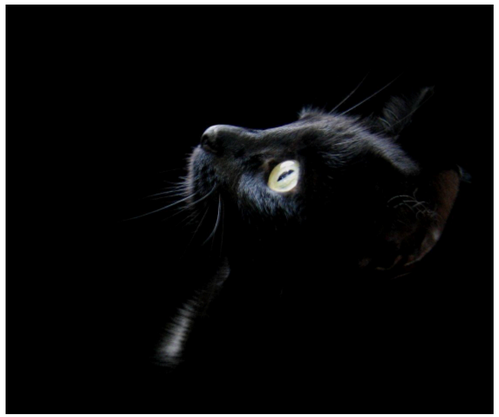




Can we extract features manually?

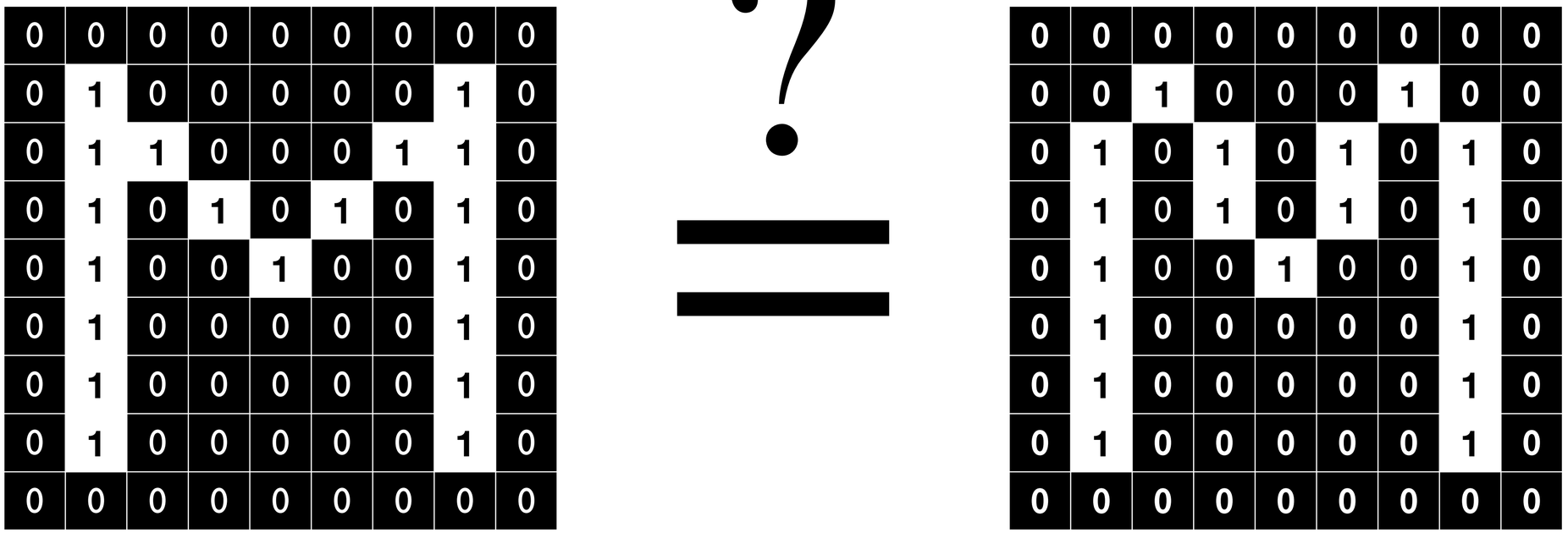
Define features!



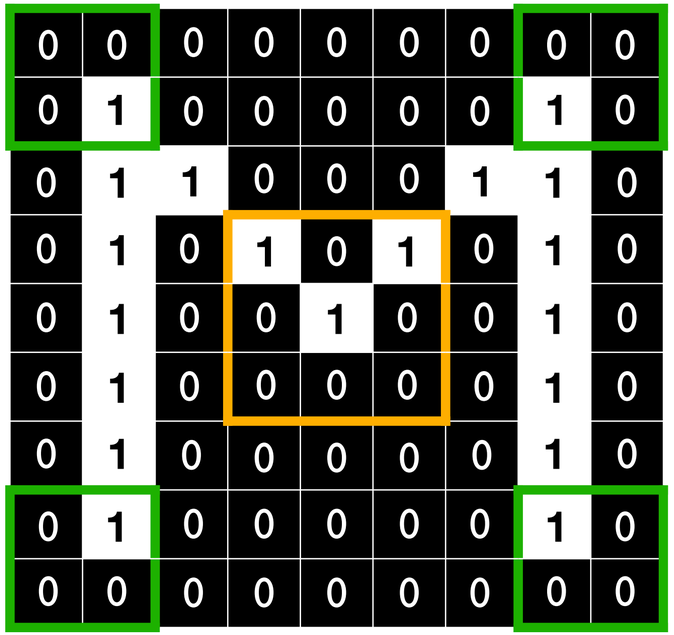


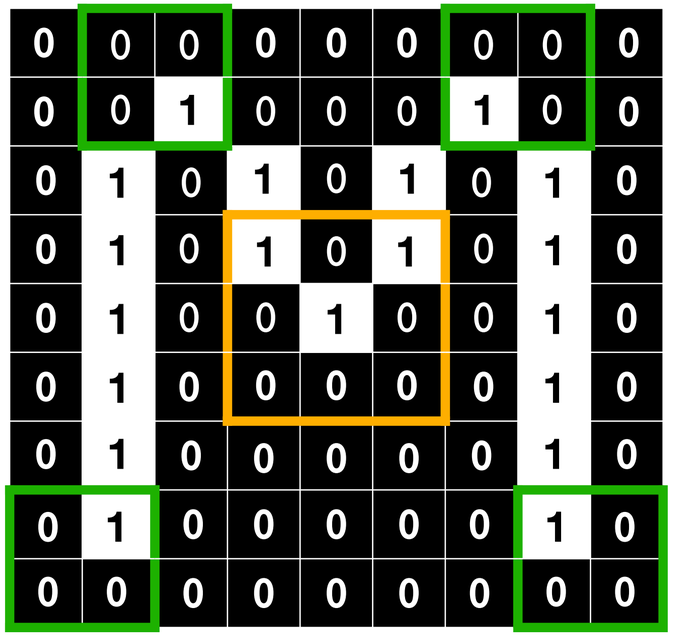
- Lighting conditions
- Deformation
- Intra-class variations
- Scale variations
- …

Algorithm needs to be invariant to all of these variations.

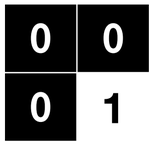

0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0
0 1 1 0 0 0 1 1 0
0 1 0 1 0 1 0 1 0
0 1 0 0 1 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0
?
=
0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 0
0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0
0 1 0 0 1 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0

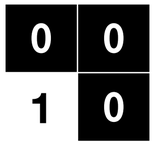
Features of the letter "M"

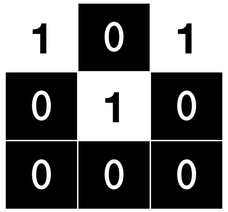


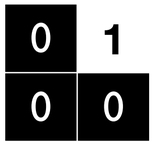


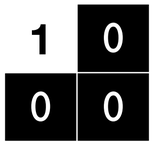
Filters to detect letter "M":



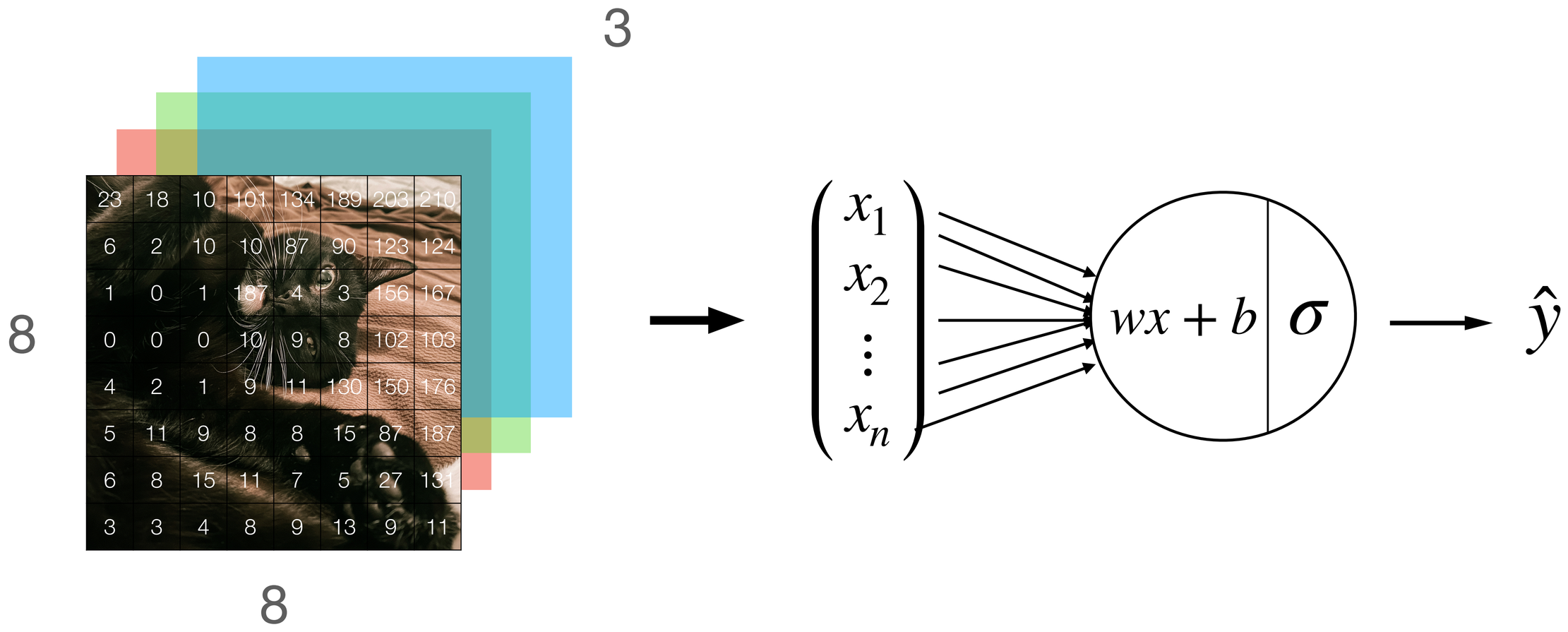








3. Convolutional Neural Networks. Why convolution?



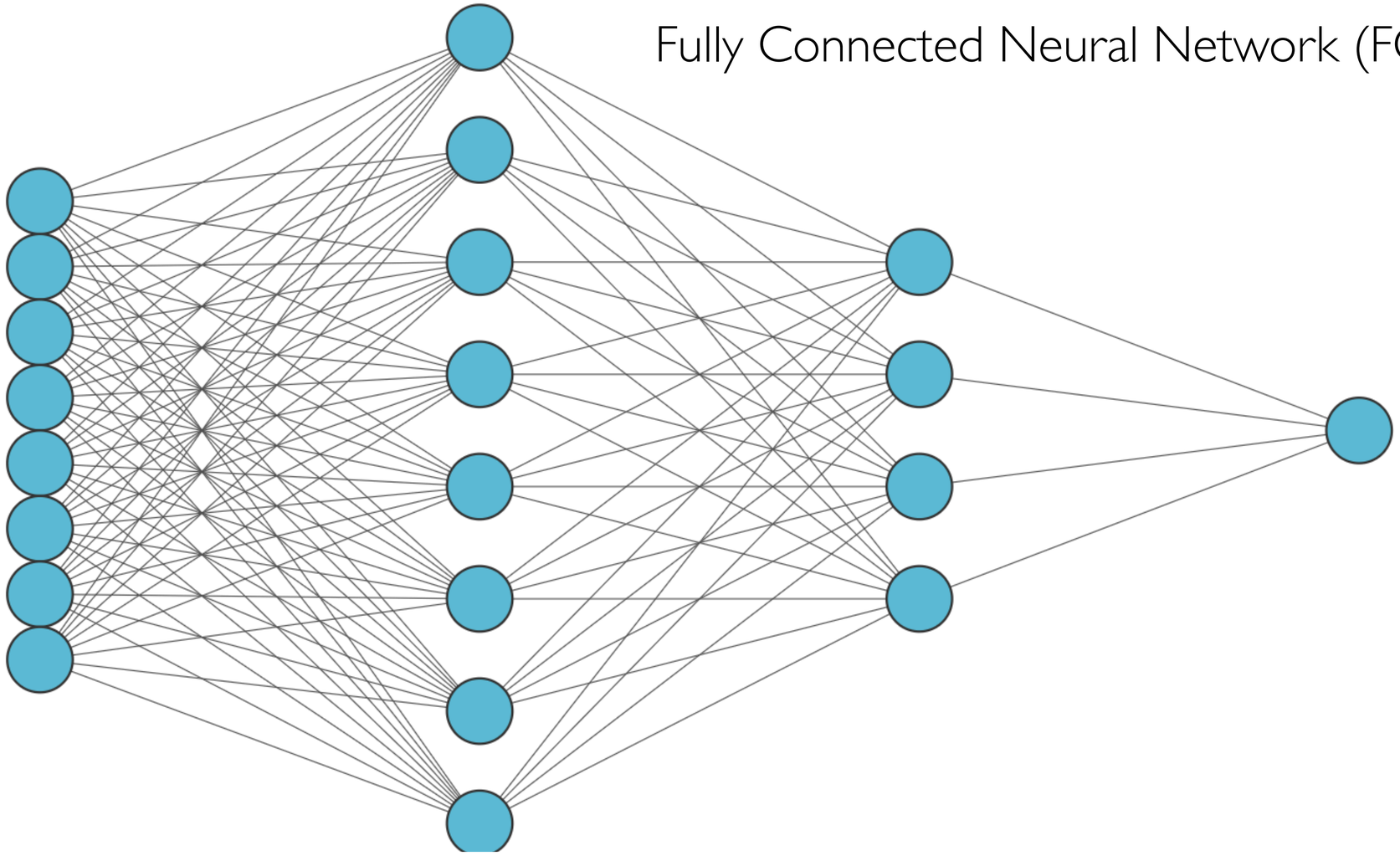
Why convolution?

1

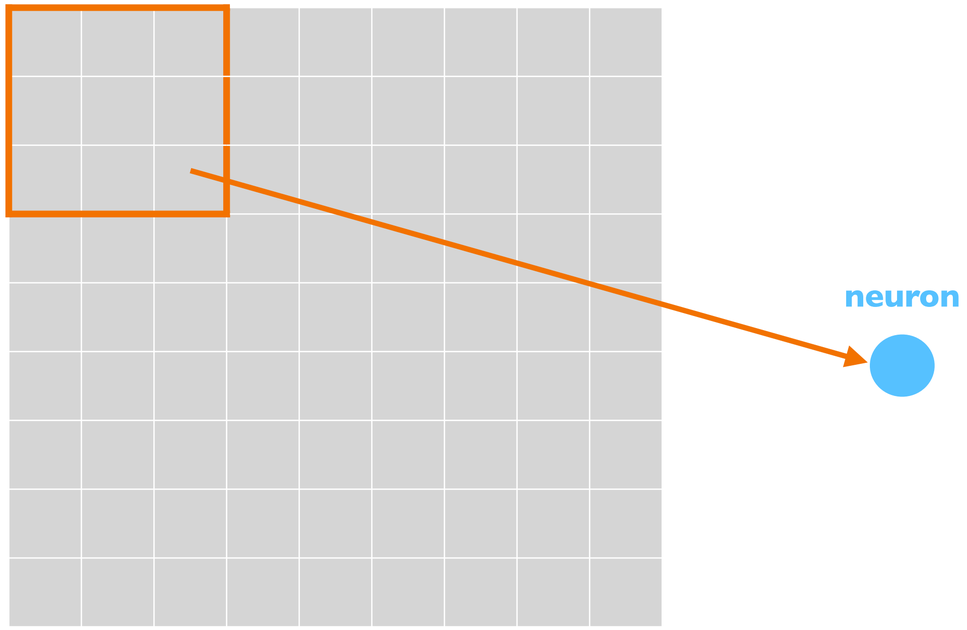
No spatial information

2

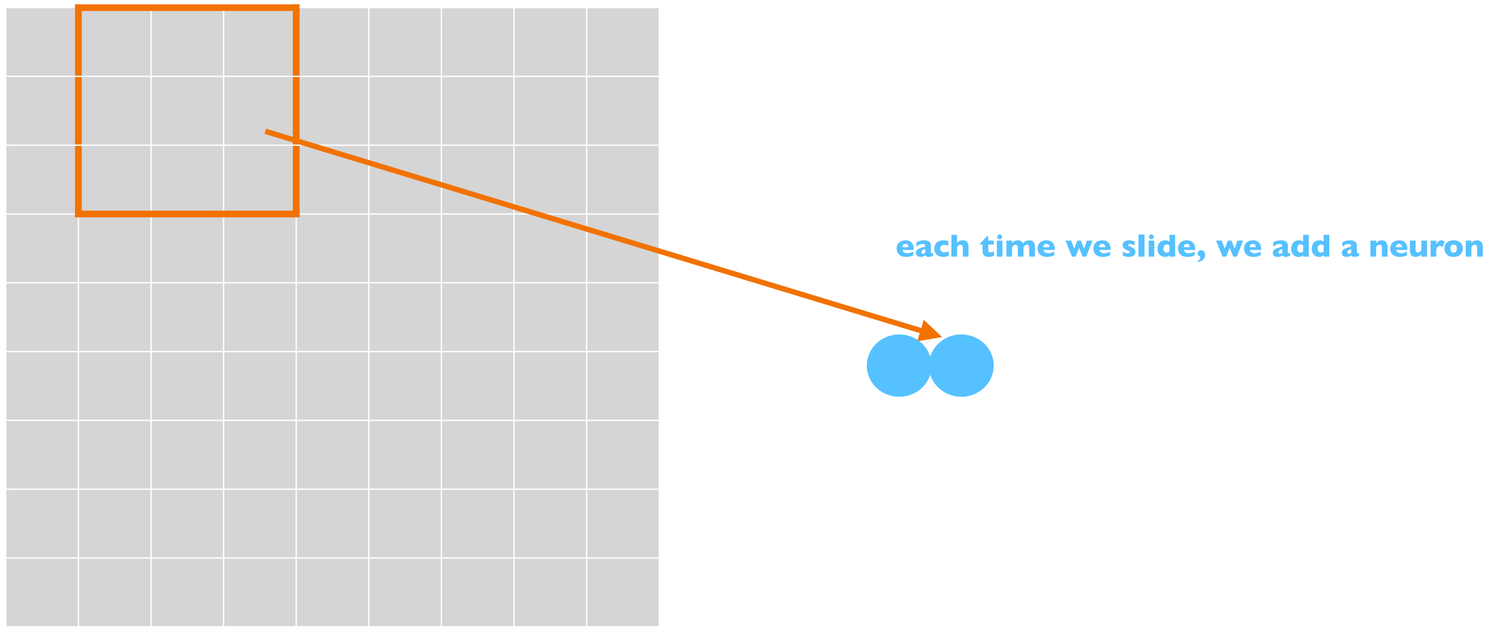
Too many parameters



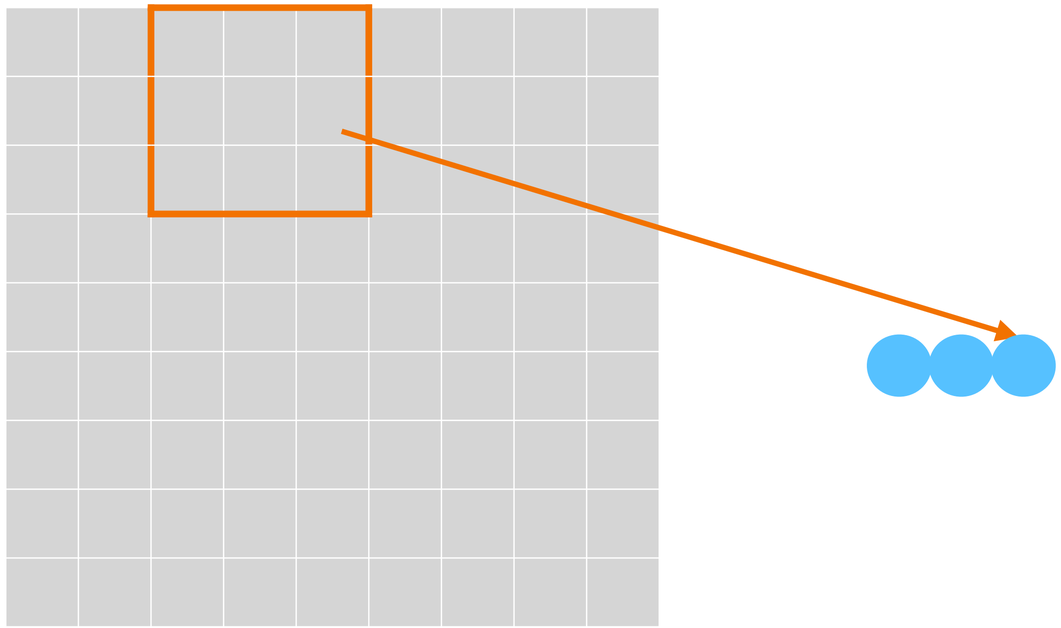
Input: array of pixel -> 2D image



Input: array of pixel -> 2D image



Input: array of pixel -> 2D image



Output size:

$$\frac{n+2p-f}{s}+1 \times \frac{n+2p-f}{s}+1$$

$n \times n$ *image*

$f \times f$ *filter*

p – *padding*

s – *stride*

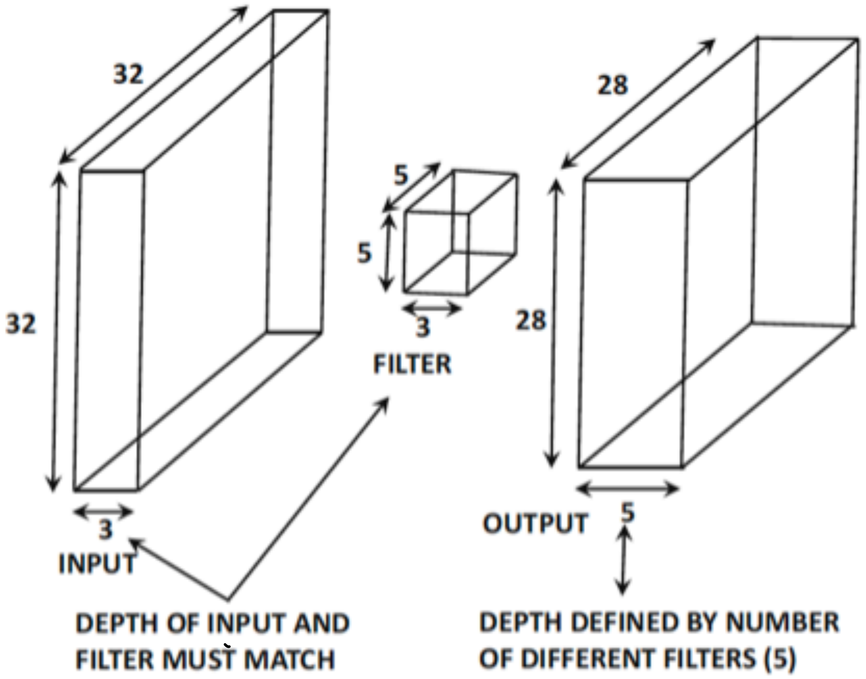
FEATURE MAP

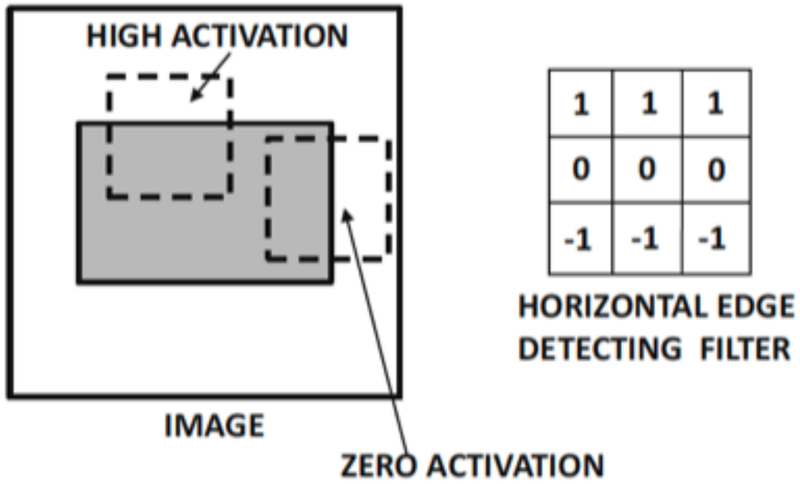
Filter (or kernel)

The goal is to learn these weights!

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

- Reduces the amount of weights.
- Adds spatial information.





1	1	1
0	0	0
-1	-1	-1

HORIZONTAL EDGE
DETECTING FILTER

$$h_{ijp}^{(q+1)} = \sum_{r=1}^{F_q} \sum_{s=1}^{F_q} \sum_{k=1}^{d_q} w_{rsk}^{(p,q)} h_{i+r-1,j+s-1,k}^{(q)} \qquad \forall i \in \{1 \ldots, L_q - F_q + 1\}$$

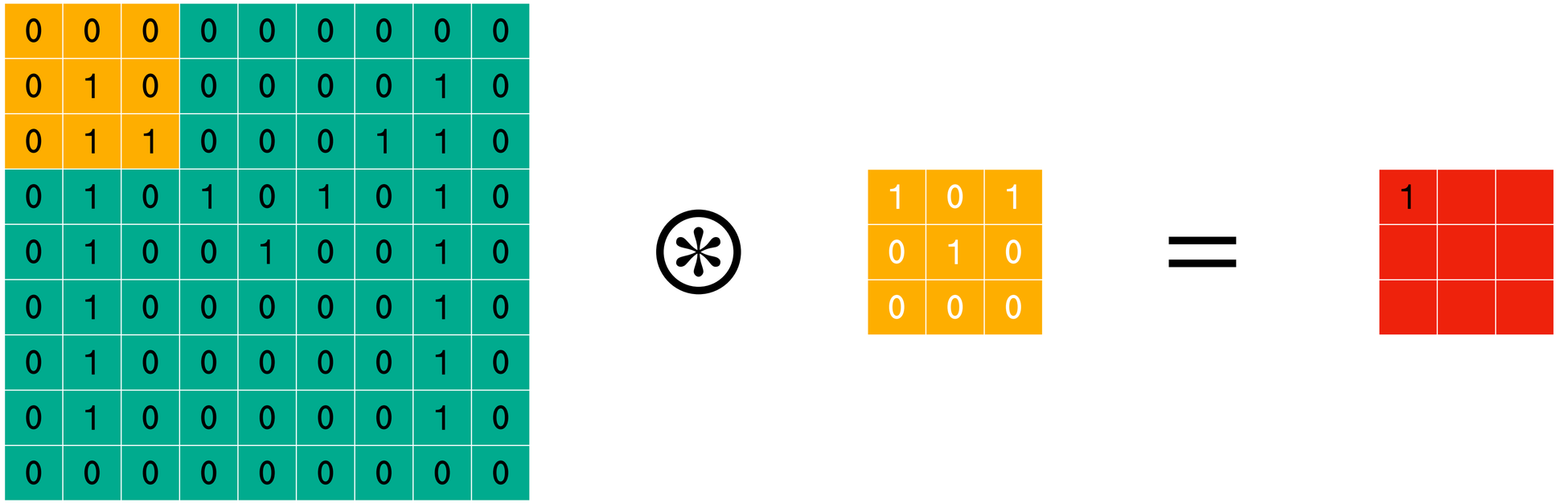
$$\forall j \in \{1 \ldots B_q - F_q + 1\}$$
$$\forall p \in \{1 \ldots d_{q+1}\}$$

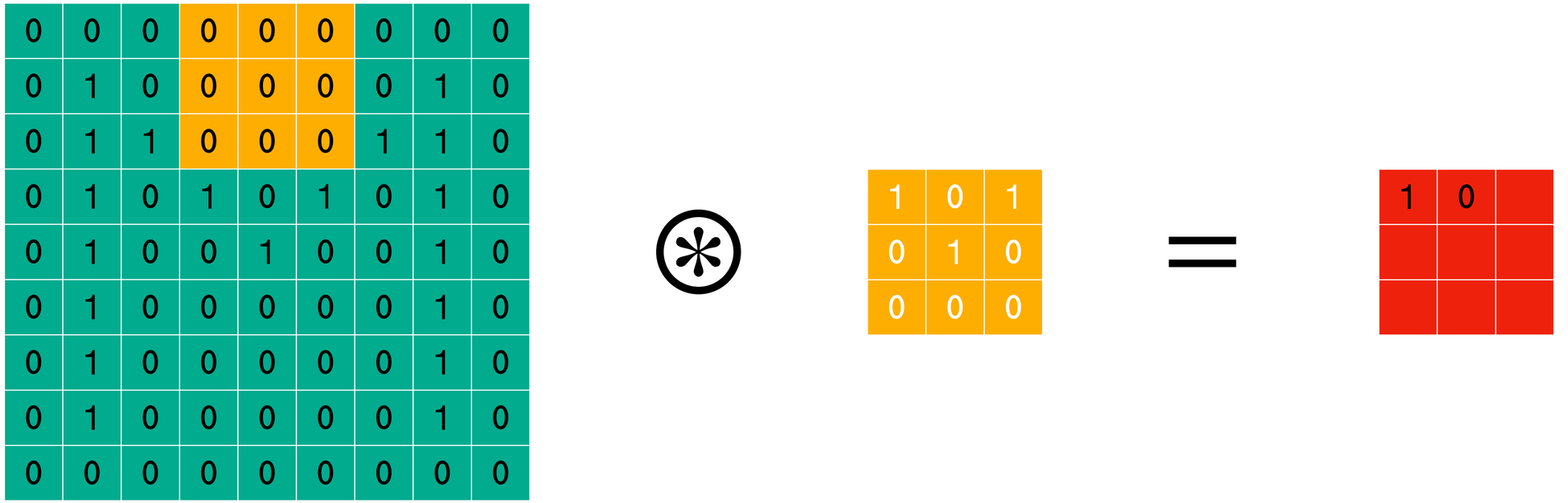
Charu C. Aggarwal, "Neural Networks and Deep Learning", Springer 2018

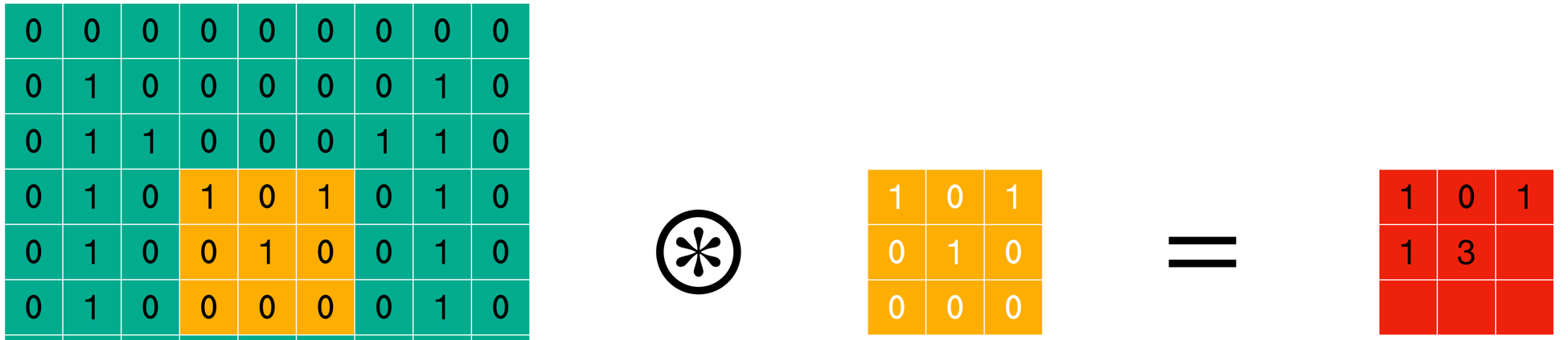
- slide the filter over the input
- element-wise multiply
- add the outputs

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

1	0	1
0	1	0
0	0	0


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


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


0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0
0 1 1 0 0 0 1 1 0
0 1 0 1 0 1 0 1 0
0 1 0 0 1 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0
1 0 1
0 1 0
0 0 0
1 0 1
1 3

input 9x9

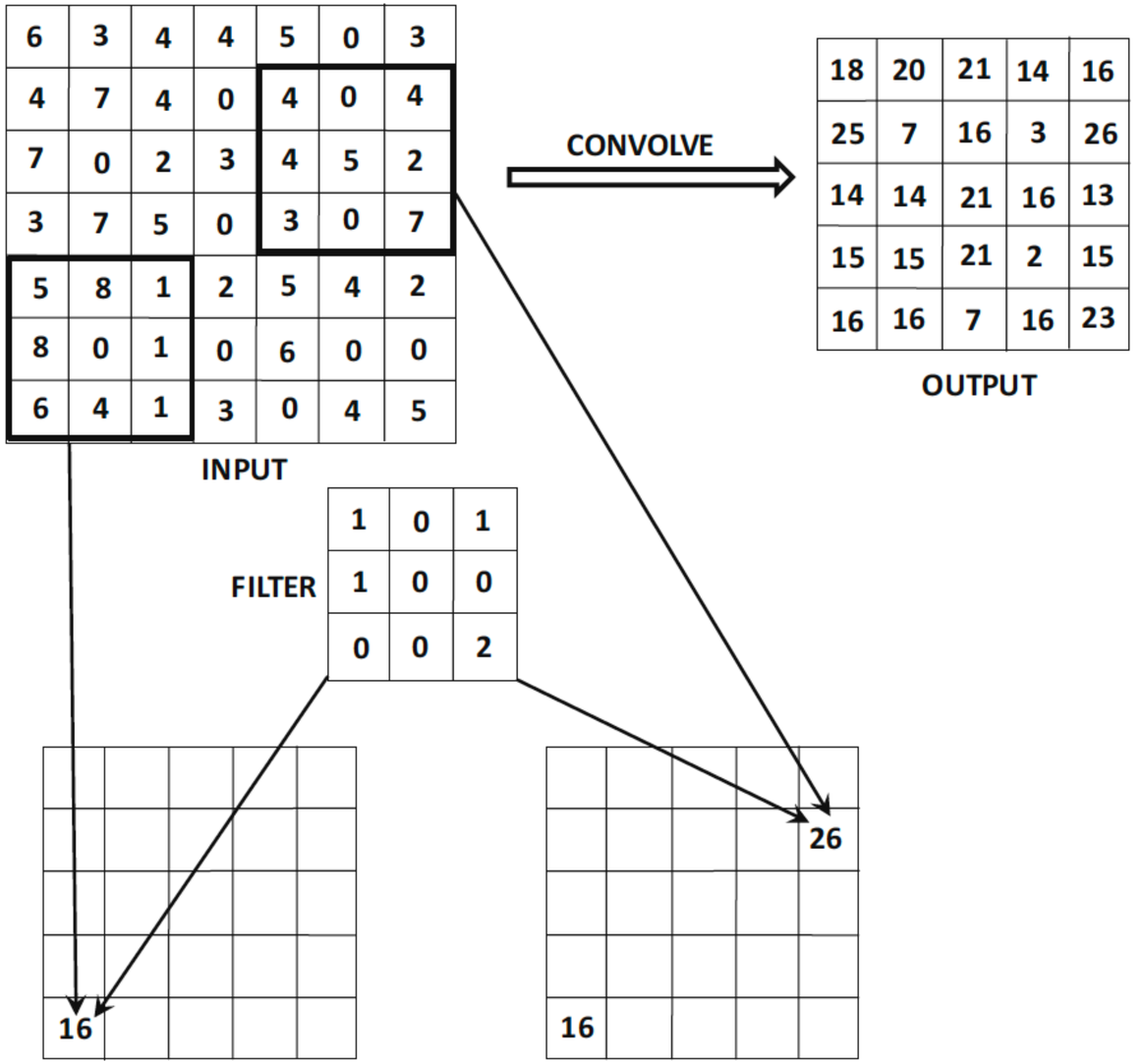
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	1	1	0
0	1	0	1	0	1	0	1	0
0	1	0	0	1	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0

filter 3x3

1	0	1
0	1	0
0	0	0

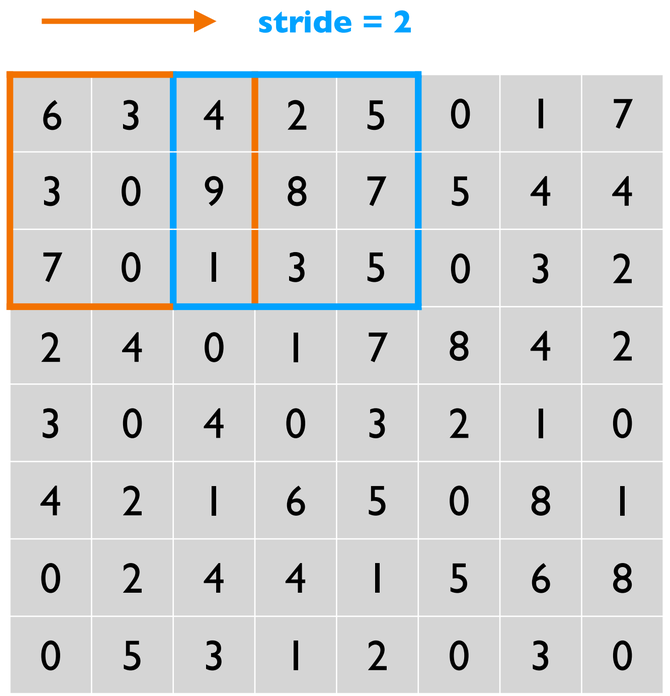
feature map 3x3

1	0	1
1	3	1
1	0	1



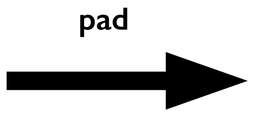
Charu C. Aggarwal, "Neural Networks and Deep Learning", Springer 2018

3.1.1 Strides


stride = 2
6 3 4 2 5 0 1 7
3 0 9 8 7 5 4 4
7 0 1 3 5 0 3 2
2 4 0 1 7 8 4 2
3 0 4 0 3 2 1 0
4 2 1 6 5 0 8 1
0 2 4 4 1 5 6 8
0 5 3 1 2 0 3 0

3.1.2 Padding

6	3	4	2	5	0	1	7
3	0	9	8	7	5	4	4
7	0	1	3	5	0	3	2
2	4	0	1	7	8	4	2
3	0	4	0	3	2	1	0
4	2	1	6	5	0	8	1
0	2	4	4	1	5	6	8
0	5	3	1	2	0	3	0



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	6	3	4	2	5	0	1	7	0	0
0	0	3	0	9	8	7	5	4	4	0	0
0	0	7	0	1	3	5	0	3	2	0	0
0	0	2	4	0	1	7	8	4	2	0	0
0	0	3	0	4	0	3	2	1	0	0	0
0	0	4	2	1	6	5	0	8	1	0	0
0	0	0	2	4	4	1	5	6	8	0	0
0	0	0	5	3	1	2	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

"Valid" - no padding is applied
"Same" - padded so that the output size is the same as the input size

3.1.3 Pooling

MaxPooling

6	3	4	2	5	0	1
3	0	9	8	7	5	4
7	0	1	3	5	0	3
2	4	0	1	7	8	4
3	0	4	0	3	2	1
4	2	1	6	5	0	8
0	2	4	4	1	5	6

3x3 pooling
stride = 2

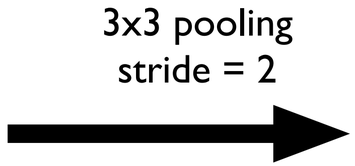
9	9	7
7		

"Neural Networks and Deep Learning", Charu C. Aggarwal, Springer 2018

3.1.3 Pooling

MaxPooling

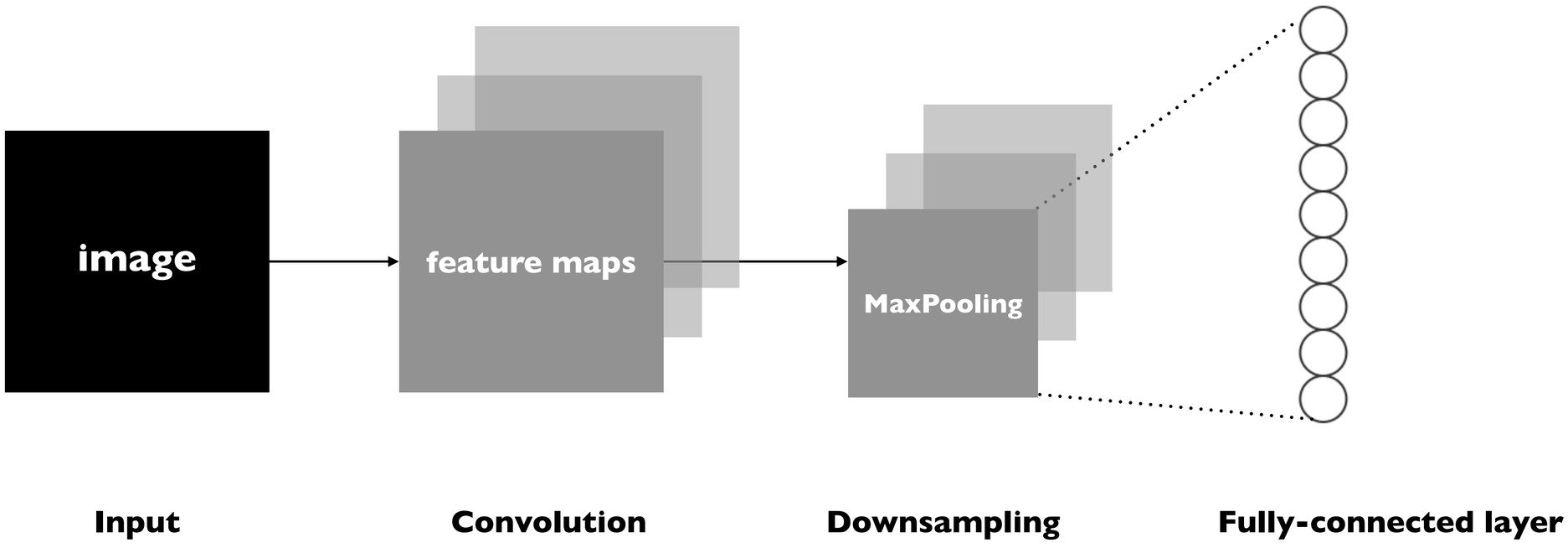
6	3	4	2	5	0	1
3	0	9	8	7	5	4
7	0	1	3	5	0	3
2	4	0	1	2	8	4
3	0	4	0	3	2	1
4	2	1	6	5	0	8
0	2	4	4	1	5	6



9	9	7
7	5	8
4	6	8

"Neural Networks and Deep Learning", Charu C. Aggarwal, Springer 2018

3.2 Architectures



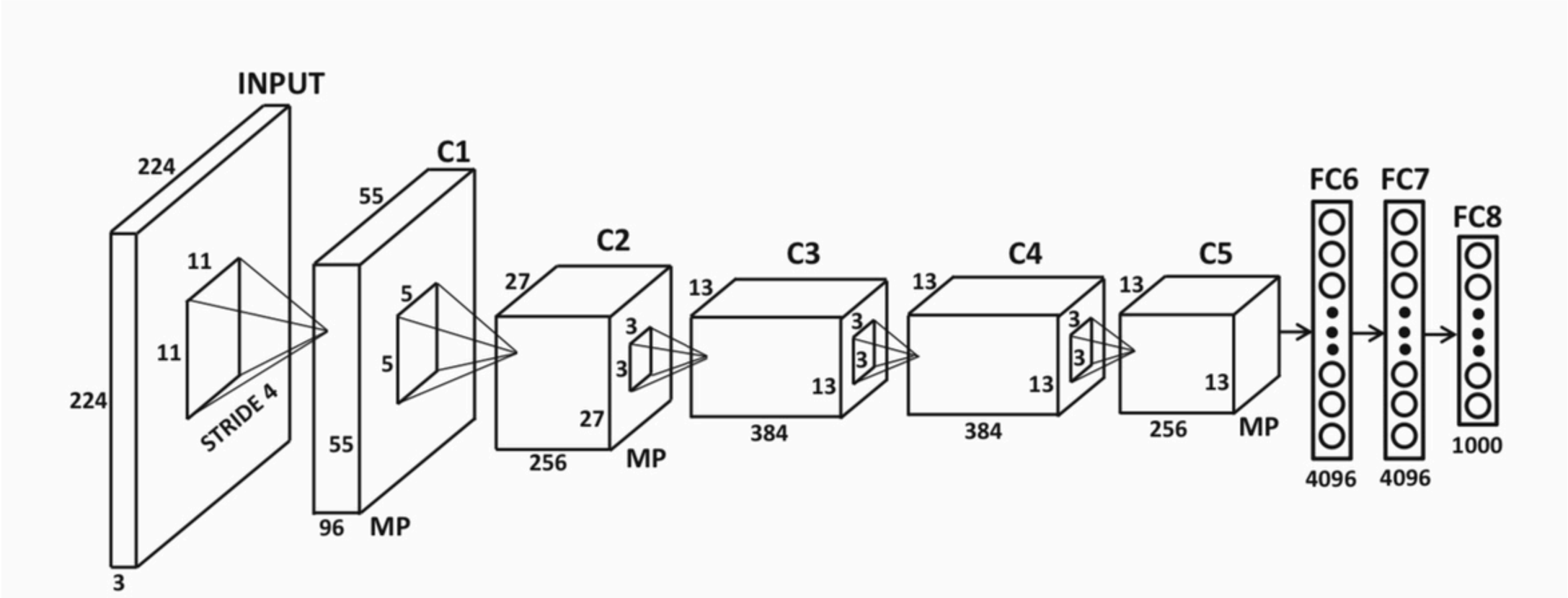
Convolution: apply filters to generate feature maps.

Non-linearity: most common choice ReLU.

Downsampling: most common choice MaxPooling.

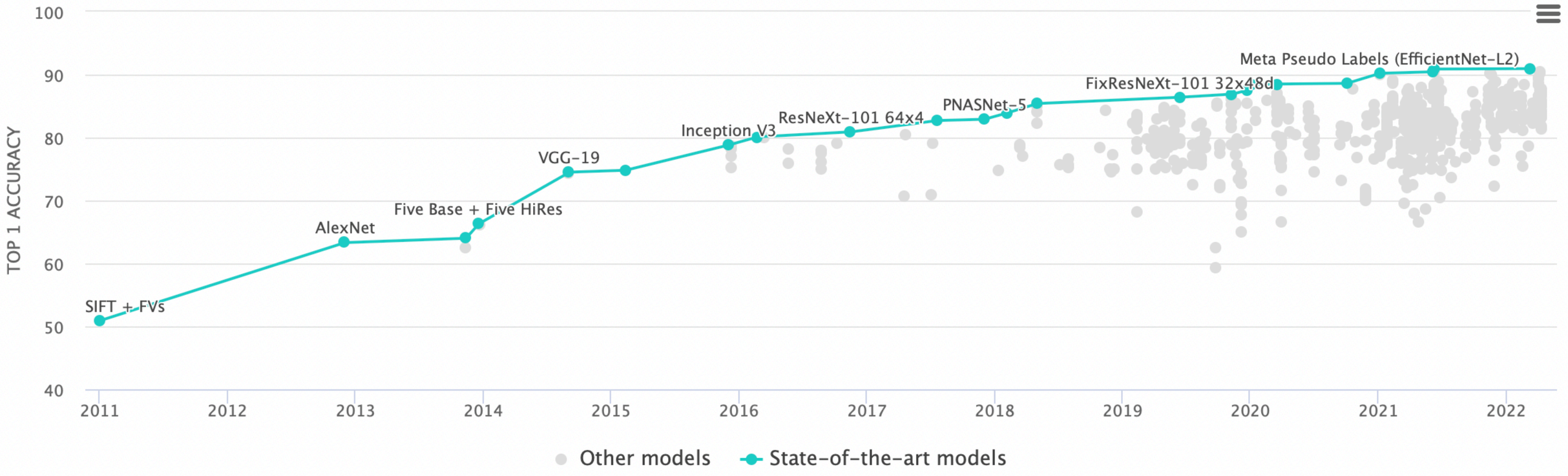
Training: learning weights of filters in convolutional layers.

3.2.2 AlexNet



Charu C. Aggarwal, "Neural Networks and Deep Learning", Springer 2018

Alex Kryzhevsky et al "Imagenet classification with deep convolutional neural networks", 2012



source: https://paperswithcode.com/sota/image-classification-on-imagenet

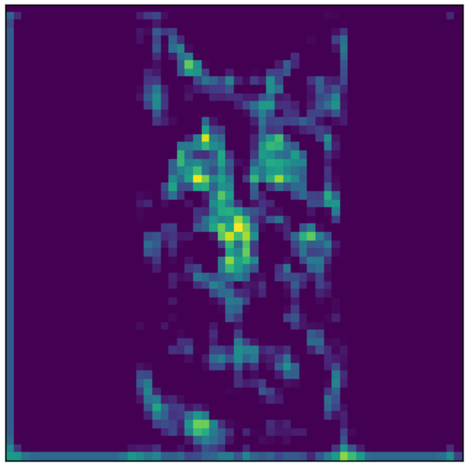
4. Code example

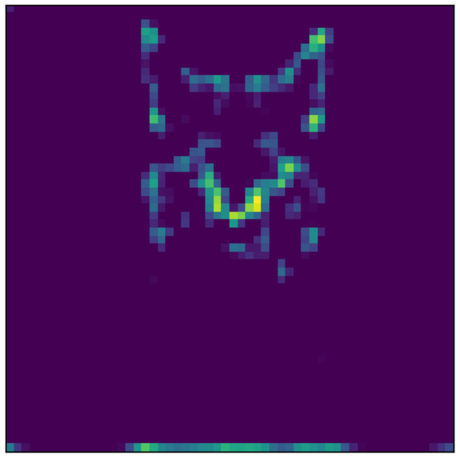
Keras

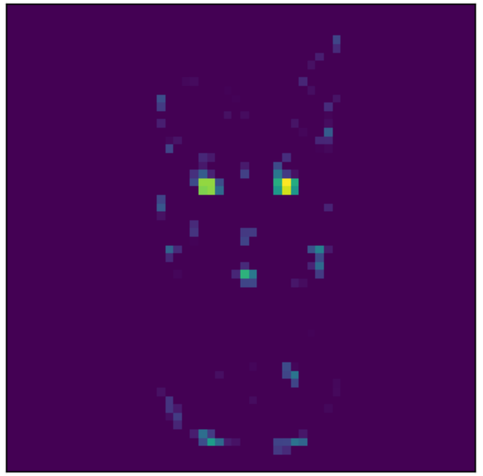
TensorFlow

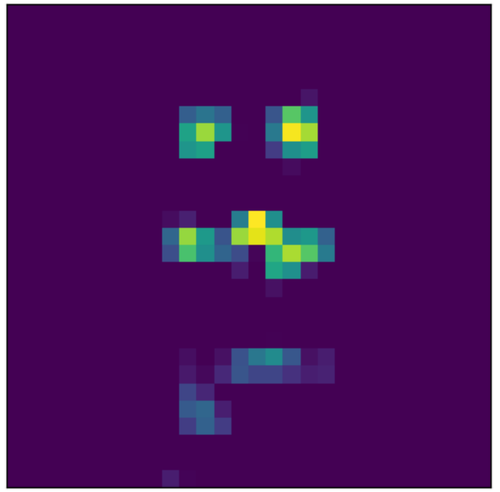
```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4), activation='relu', padding='same',
                        input_shape=(227, 227, 1)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
    keras.layers.Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=384, kernel_size=(1, 1), strides=(1, 1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='same'),
    keras.layers.Flatten(),
    keras.layers.Dense(4096, activation='relu', input_dim=(227, 227, 1)),
    keras.layers.Dropout(0.4),
    keras.layers.Dense(4096, activation='relu'),
    keras.layers.Dropout(0.4),
    keras.layers.Dense(1000, activation='relu'),
    keras.layers.Dropout(0.4),
    keras.layers.Dense(10, activation='softmax')
])
```

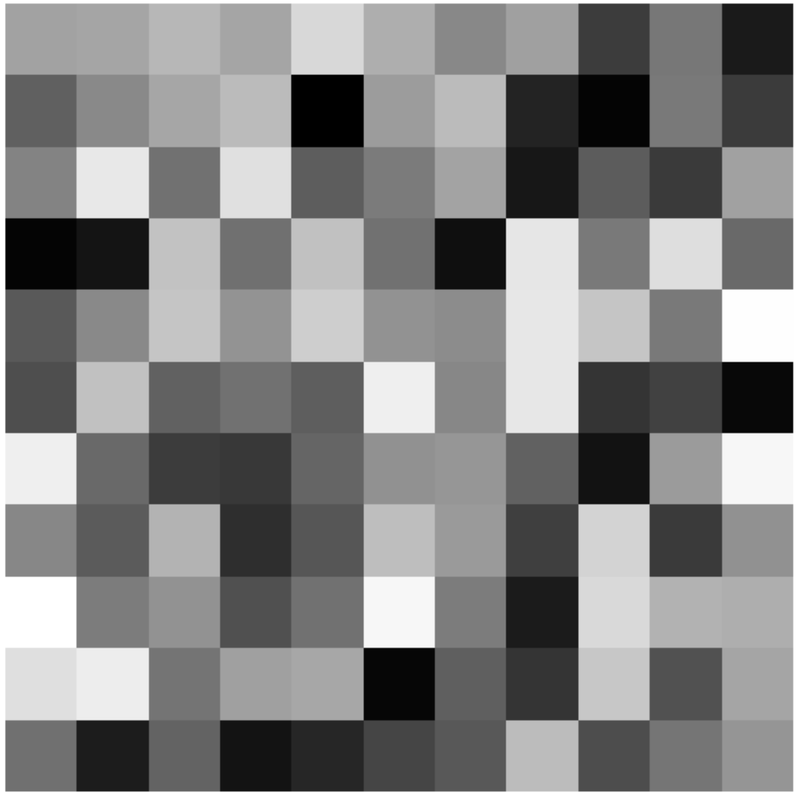
Hierarchical Feature Engineering

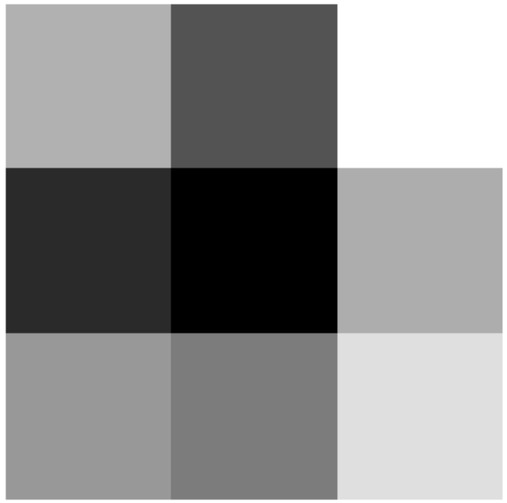


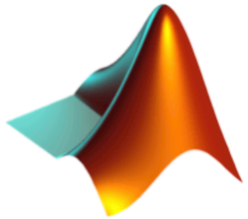












Practice!

Example 1

Thank you for your attention!