Lecture 4: Convolutional Neural Network (CNN)

Areej Alasiry

CIS 6217 – Computer Vision for Data Representation

College of Computer Science, King Khalid University

Outline

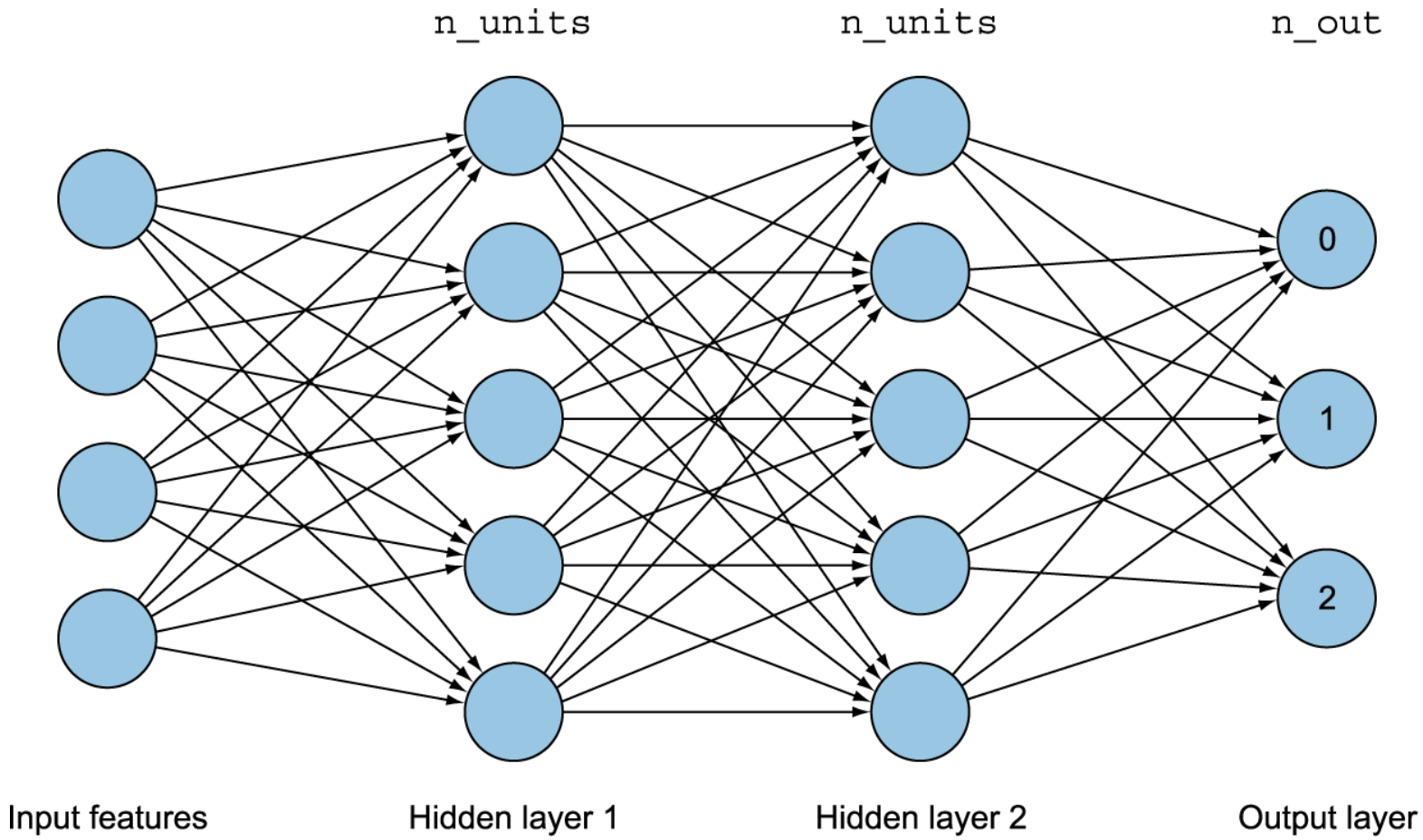
1. Why Not Feed-Forward for Images?
2. CNN Architecture
3. Filters/Kernels & Feature Maps
4. Pooling Layers
5. CNN Building Blocks (Conv, Pool, FC)
6. Lab: Building CNN
7. Summary

Learning Outcomes

- Explain why fully connected NNs are inefficient for images.
- Describe the concepts of local receptive fields, weight sharing, and convolutions.
- Understand filters/kernels and how they extract spatial features.
- Explain the role of pooling layers in reducing dimensions.
- Illustrate a basic CNN architecture for image classification.
- Implement a simple CNN using PyTorch

Why not feed forward for images?

Remember the NN Structure!

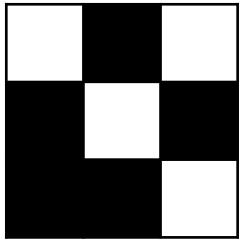


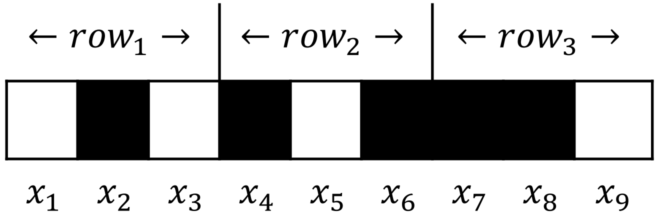
Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Input Layer

28 × 28
= 784 pixels

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	55	87	157	156	187	215	81	0	0	0	0	0
0	0	0	0	0	0	0	0	5	25	54	25	96	140	215	215	251	254	254	254	254	254	163	0	0	0	0	0
0	0	0	0	0	0	0	0	157	254	254	254	0	254	254	254	254	254	254	254	254	241	100	0	0	0	0	0
0	0	0	0	0	0	0	0	156	117	184	214	214	156	117	49	19	19	141	254	241	158	23	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	173	254	246	70	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	247	254	219	58	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	43	203	254	250	121	15	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	43	145	254	254	229	111	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	137	245	254	254	238	40	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	130	254	254	254	254	254	235	106	23	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	55	196	196	114	194	223	254	254	216	23	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	75	245	253	55	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	150	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	200	254	134	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	246	253	59	0	0	0	0	0	0	0	0	0
0	0	0	17	11	18	0	0	0	0	0	0	0	0	67	254	254	170	0	0	0	0	0	0	0	0	0	0
0	0	0	128	254	159	0	0	0	0	0	0	20	140	248	254	254	9	0	0	0	0	0	0	0	0	0	0
0	0	0	154	254	229	80	79	79	161	176	218	254	254	254	104	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	29	203	254	254	254	215	254	254	254	250	95	11	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	80	155	155	156	111	58	58	36	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





$Input = [\quad ,\quad ,\quad ,\quad ,\quad ,\quad ,\quad ,\quad ,\quad]$

Image flattening: The process of transforming Input image into 1-dimensional vector.

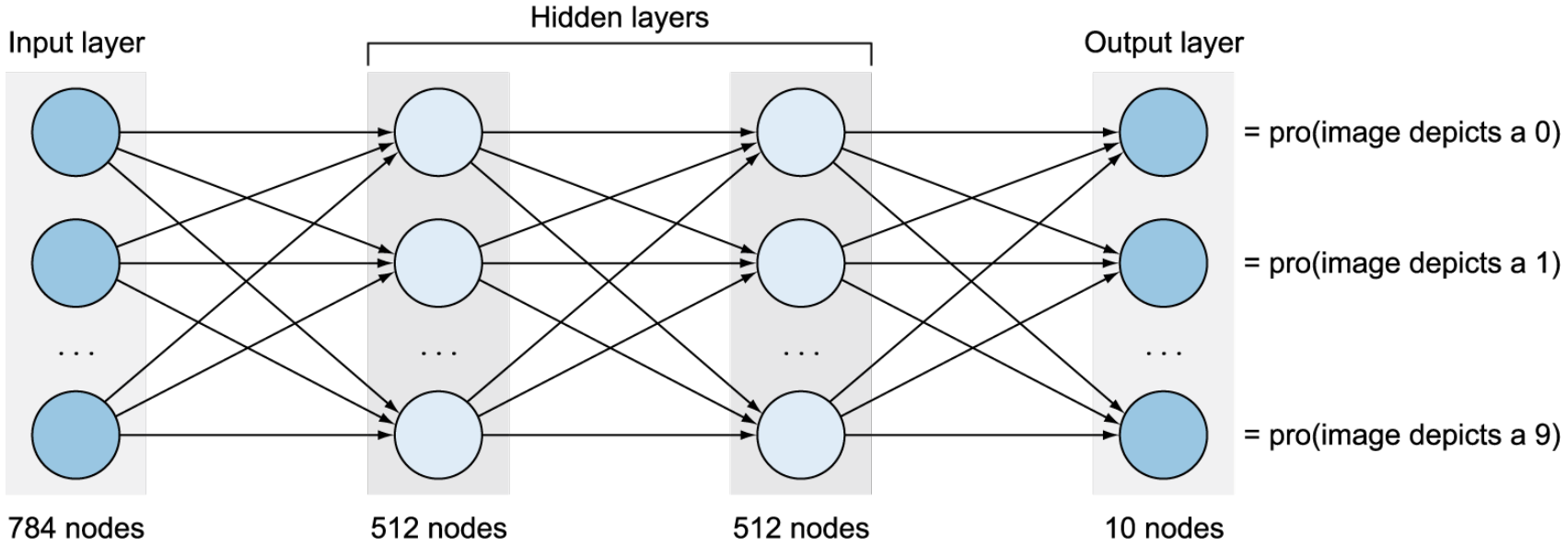
Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Hidden Layers

- Choose the number of layers
- Choose the number of nodes per layer
- Choose activation function!

Output Layer

- In the digit example, how many nodes should be in the output layer?
- We need to identify the number in the image 0,1,2,…,9



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Code

```python
from keras.models import Sequential
from keras.layers import Flatten, Dense

model = Sequential()

model.add( Flatten(input_shape = (28,28) ))

model.add(Dense(512, activation = 'relu'))
model.add(Dense(512, activation = 'relu'))

model.add(Dense(10, activation = 'softmax'))
model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: U
  super().__init__(**kwargs)
```

Model: "sequential"

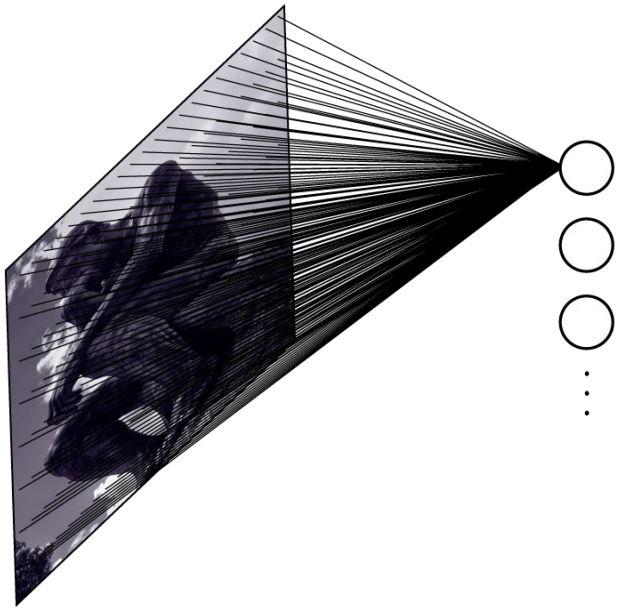
Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 512)	262,656
dense_2 (Dense)	(None, 10)	5,130

Total params: 669,706 (2.55 MB)
Trainable params: 669,706 (2.55 MB)
Non-trainable params: 0 (0.00 B)

MLPs Drawbacks

1. Large number of parameters (complexity)

Fully connected neural net



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

2. Spatial Feature Loss

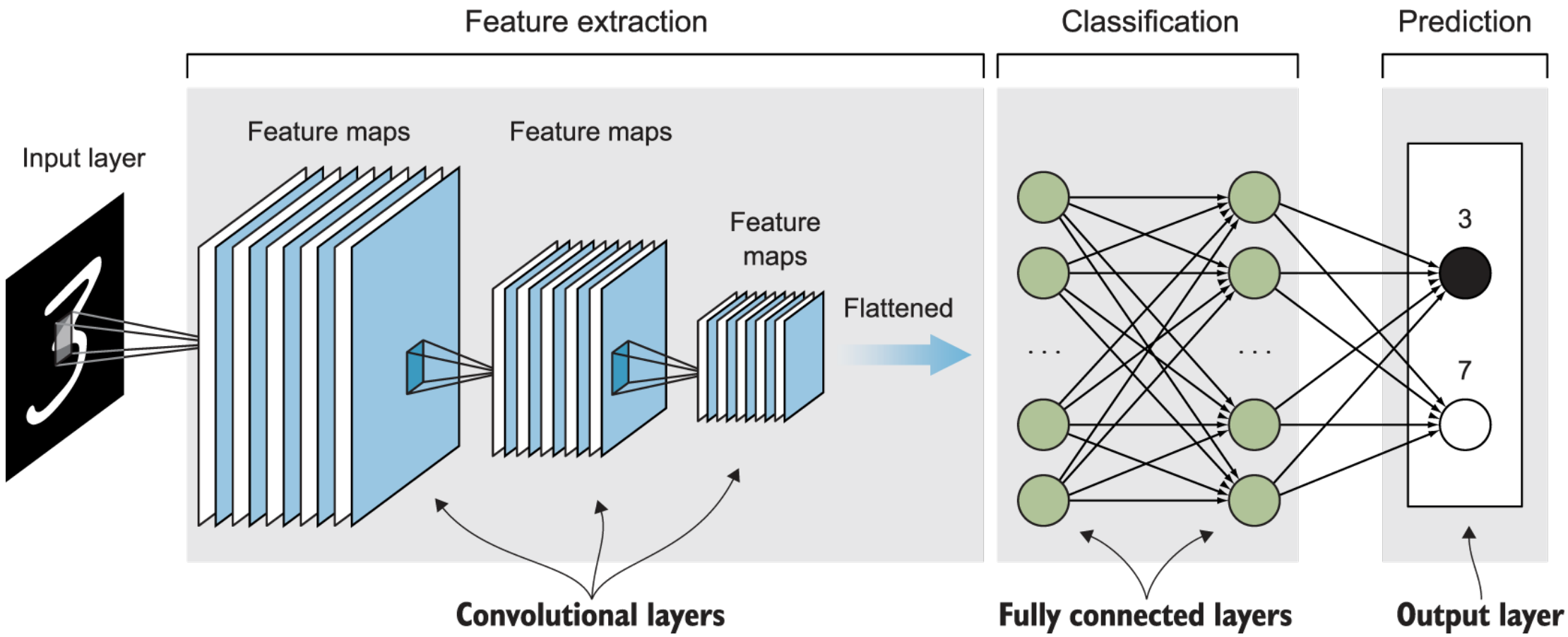
1	1	0
1	1	0
0	0	0

$\leftarrow row_1 \rightarrow$			$\leftarrow row_2 \rightarrow$			$\leftarrow row_3 \rightarrow$		
1	1	0	1	1	0	0	0	0
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9

0	0	0
0	1	1
0	1	1

CNN

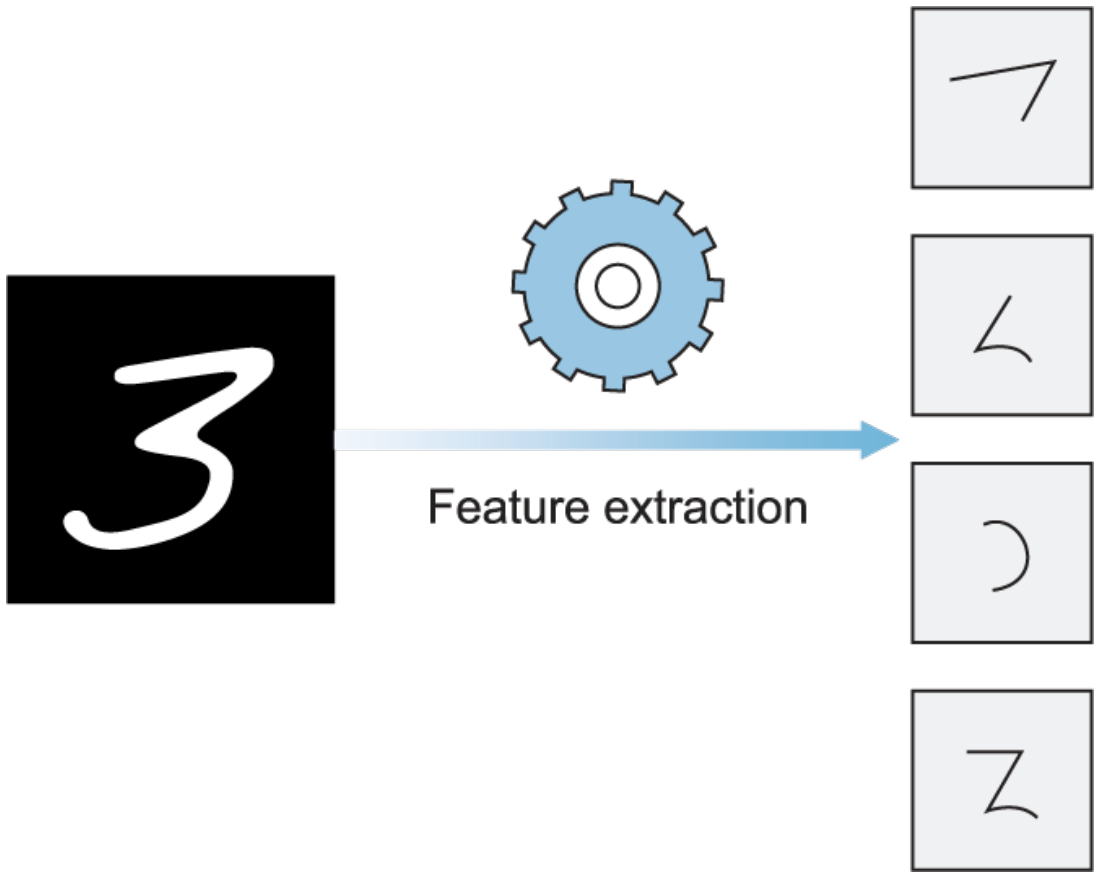
CNN Architecture



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Feature Extraction

- Feature map: one filter applied to the previous layer.
- Mapping of where a specific feature is found in an image

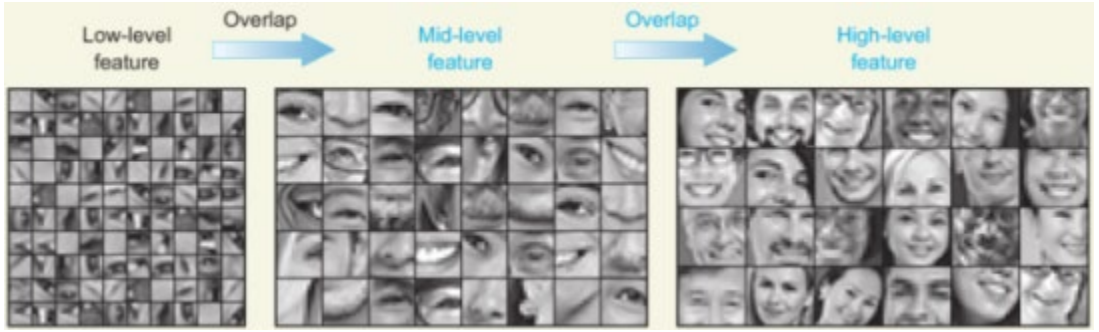


Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Classification

- Fully connected layer for classification
- It is a regular MLP

- How CNN Learns Patterns



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Components of CNN

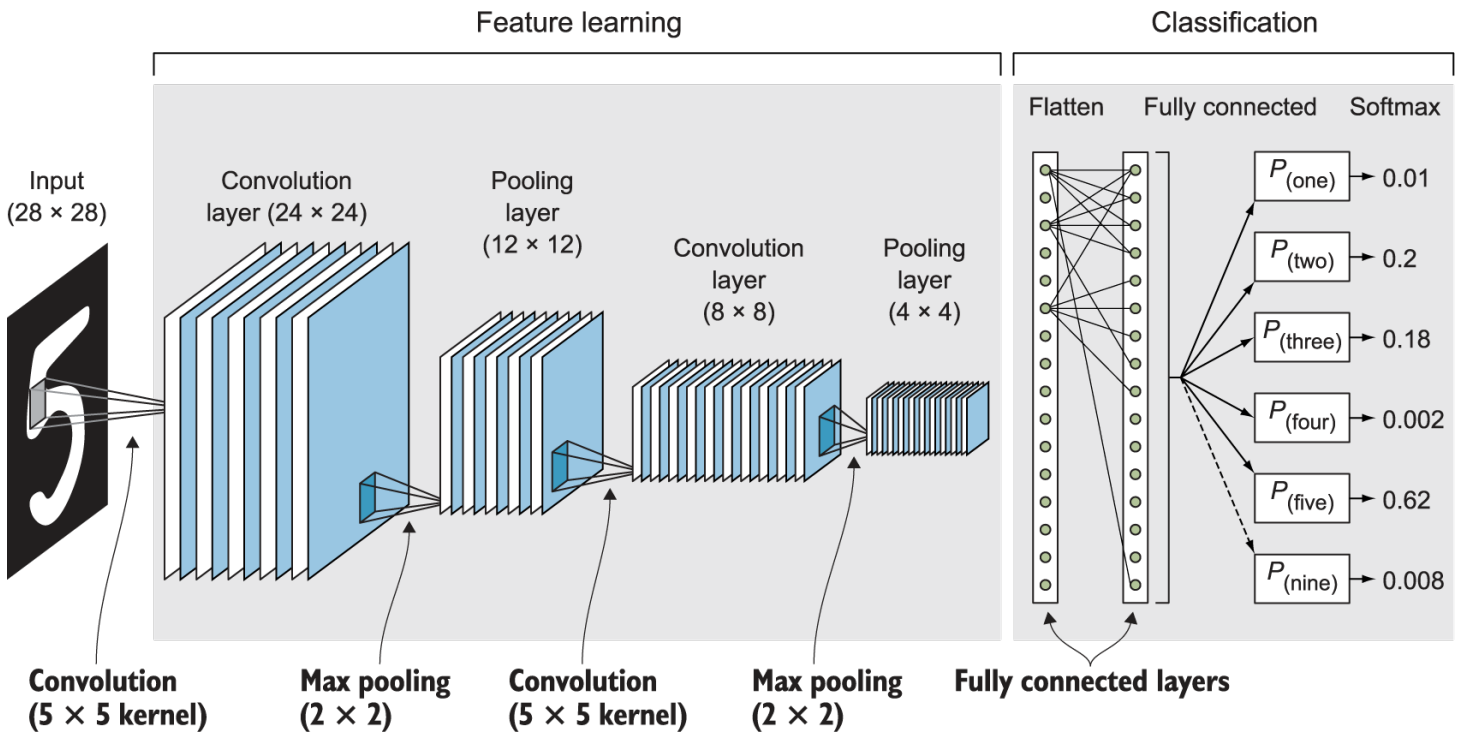
Convolutional layer (CONV)

Pooling layer (POOL)

Fully connected layer (FC)

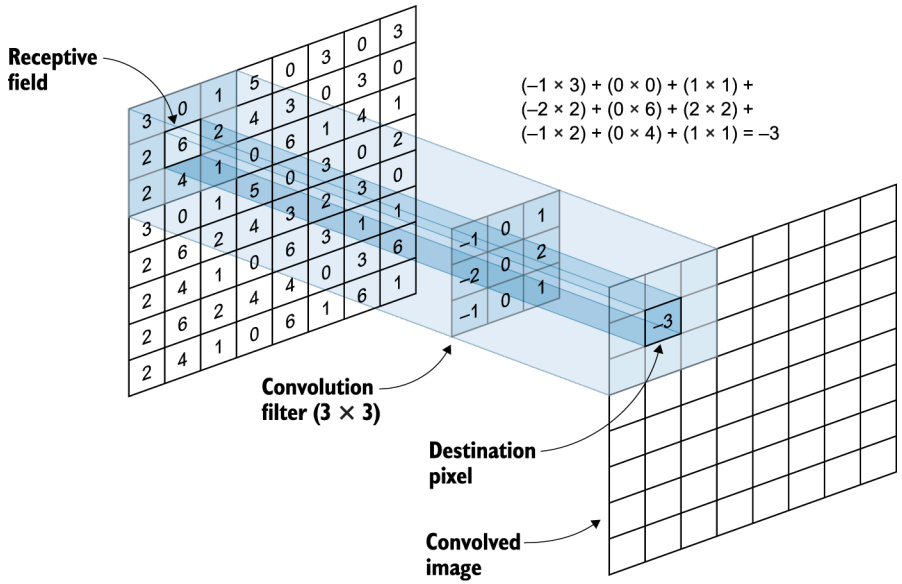
Convolutional Layer

Basic Components of CNN



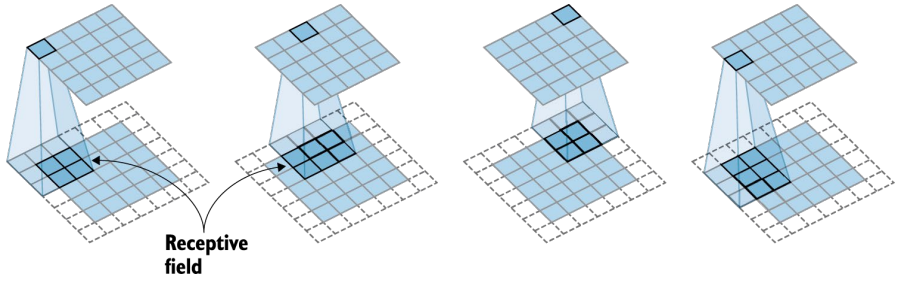
Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Convolutional Layers



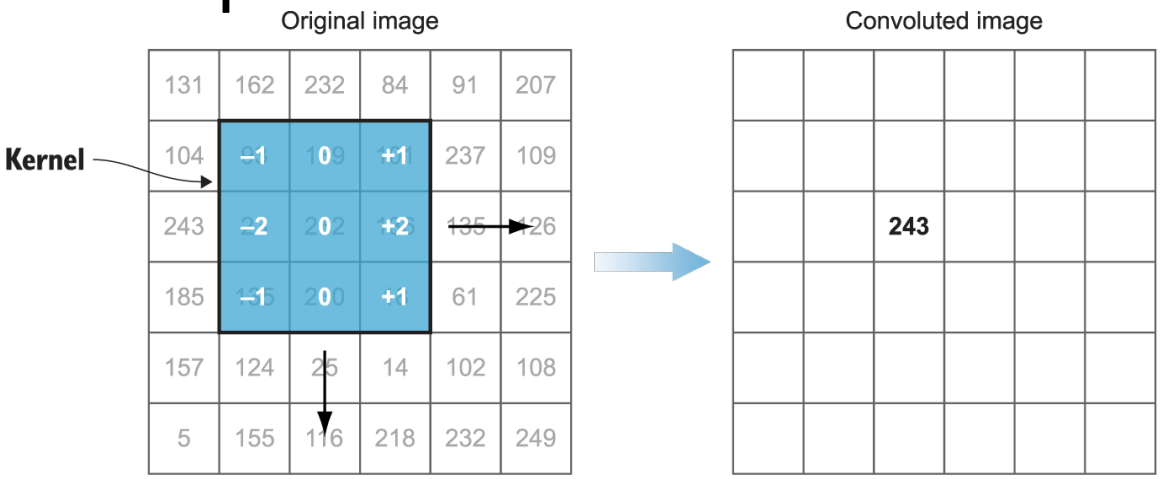
- Kernel is the convolution filter
- It slides over the original image pixel by pixel

- Receptive field: is the area of the image that the filter convolves

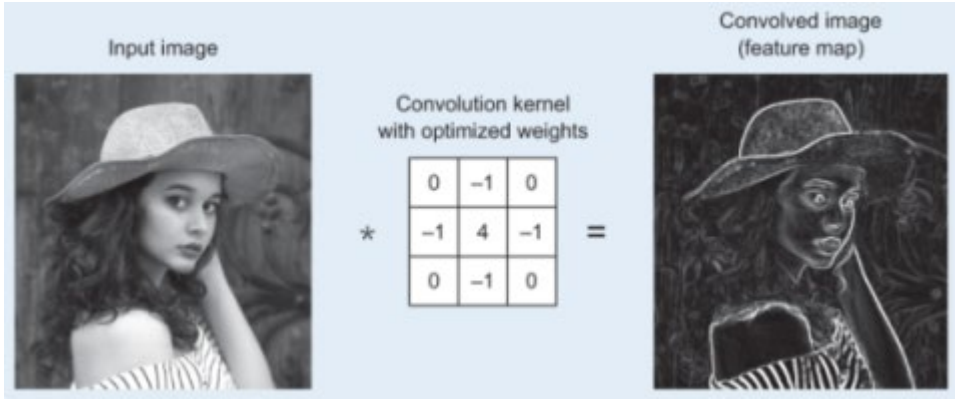


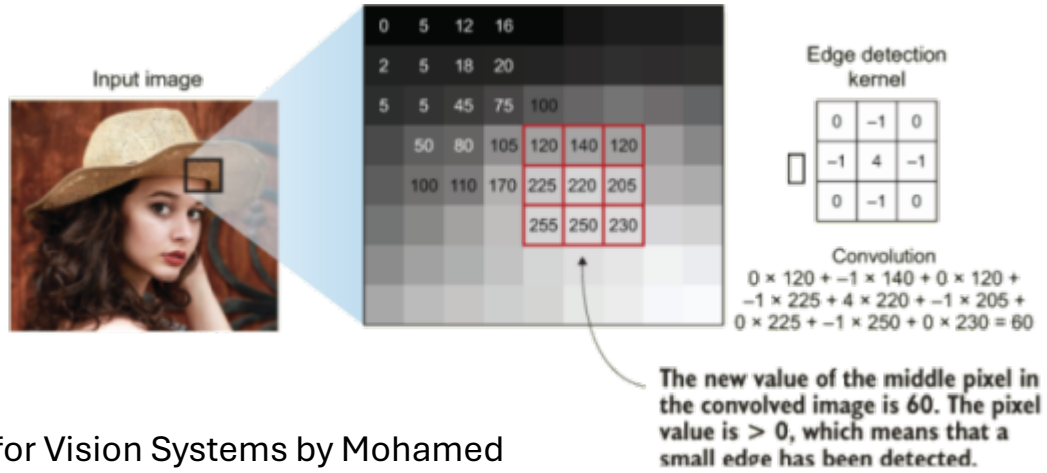
Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Convolutional Operations



- Example: apply edge detection filter

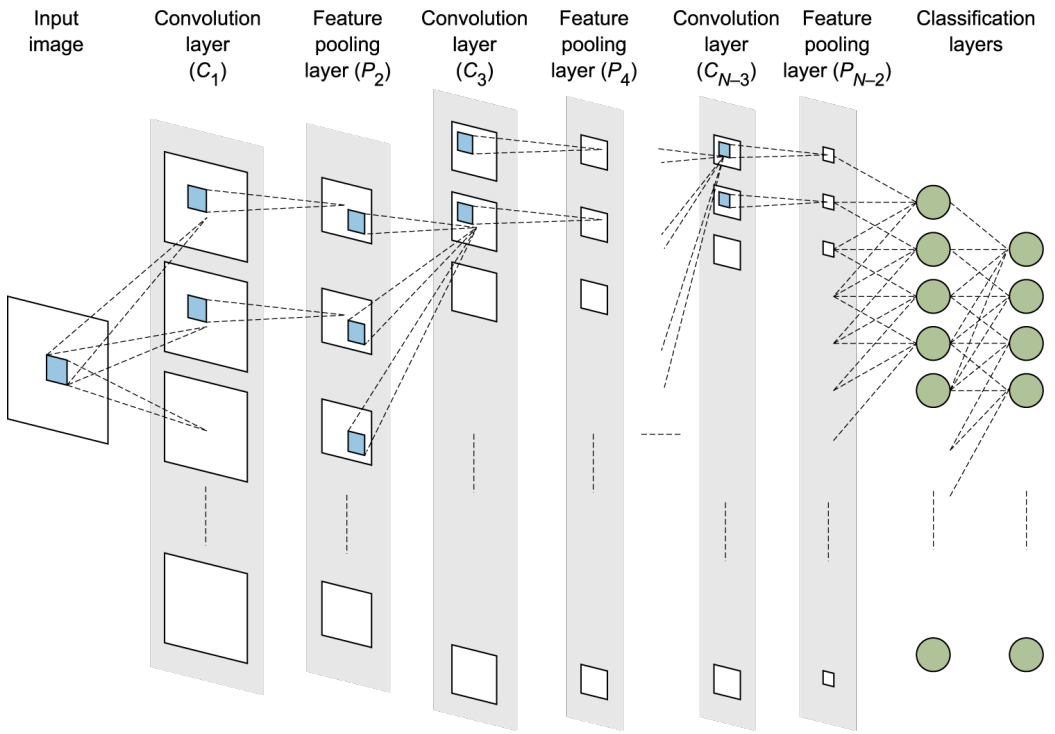




Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Number of filters

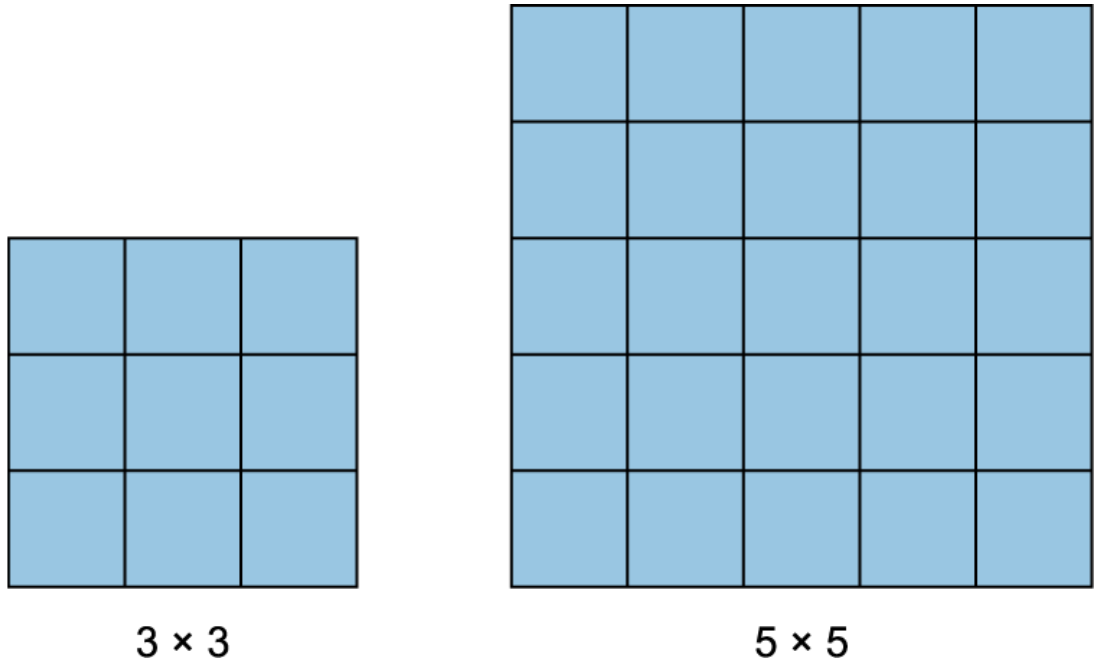
- Each convolutional layer has one or more filters.



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Kernel Size

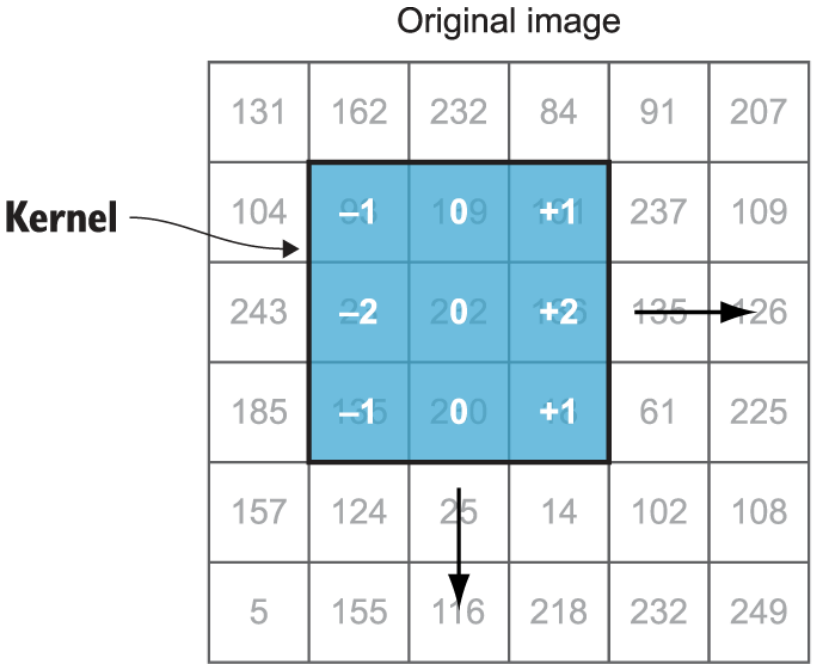
- Dimensions of the convolutional filter

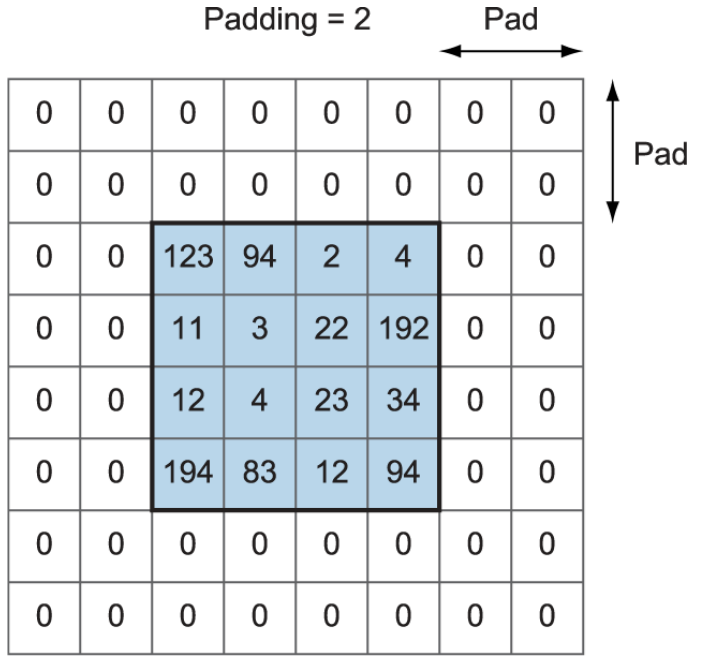


Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Strides and Padding

- *Strides* --The amount by which the filter slides over the image.
- Often called zero-padding because we add zeros around the border of an image



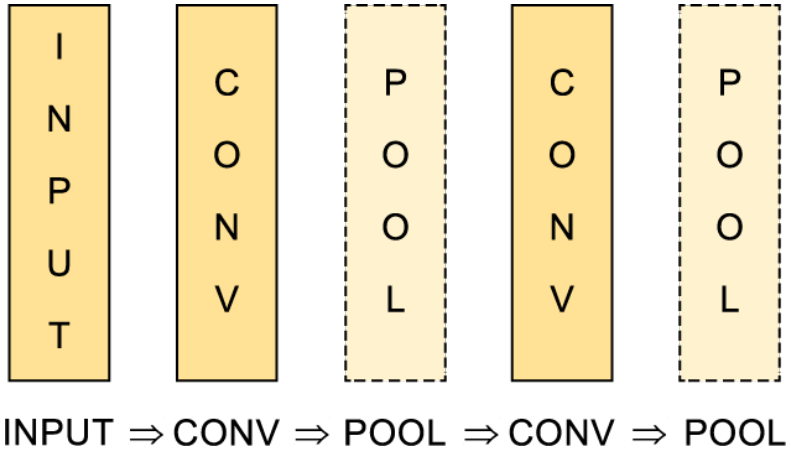


Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Pooling Layer

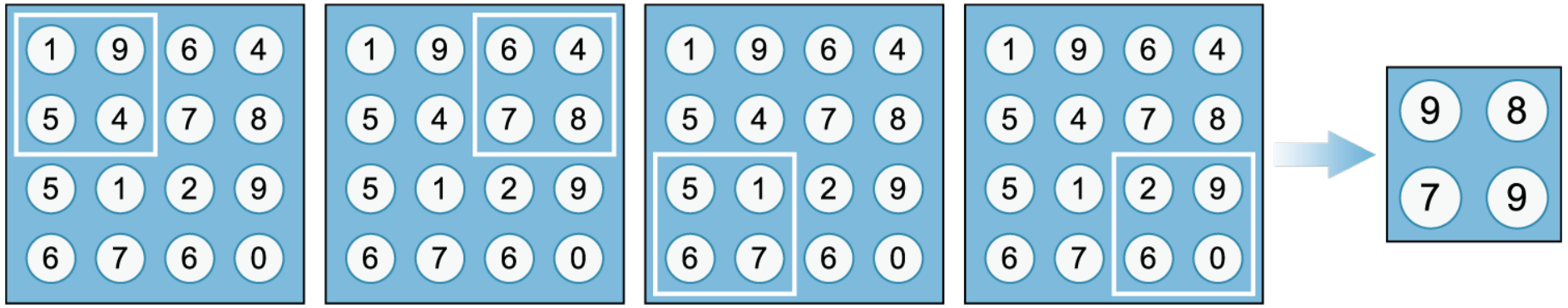
Pooling Layers

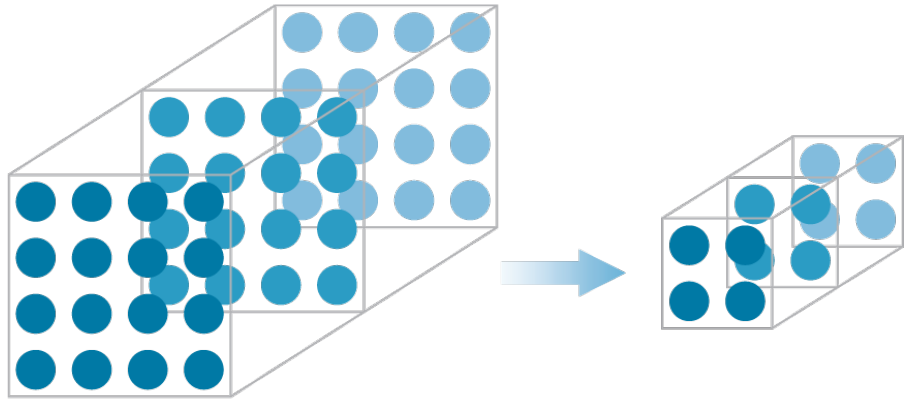
- The goal of the pooling layer is to downsample the feature maps produced by the convolutional layer into a smaller number of parameters, thus reducing computational complexity.



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Max Pooling



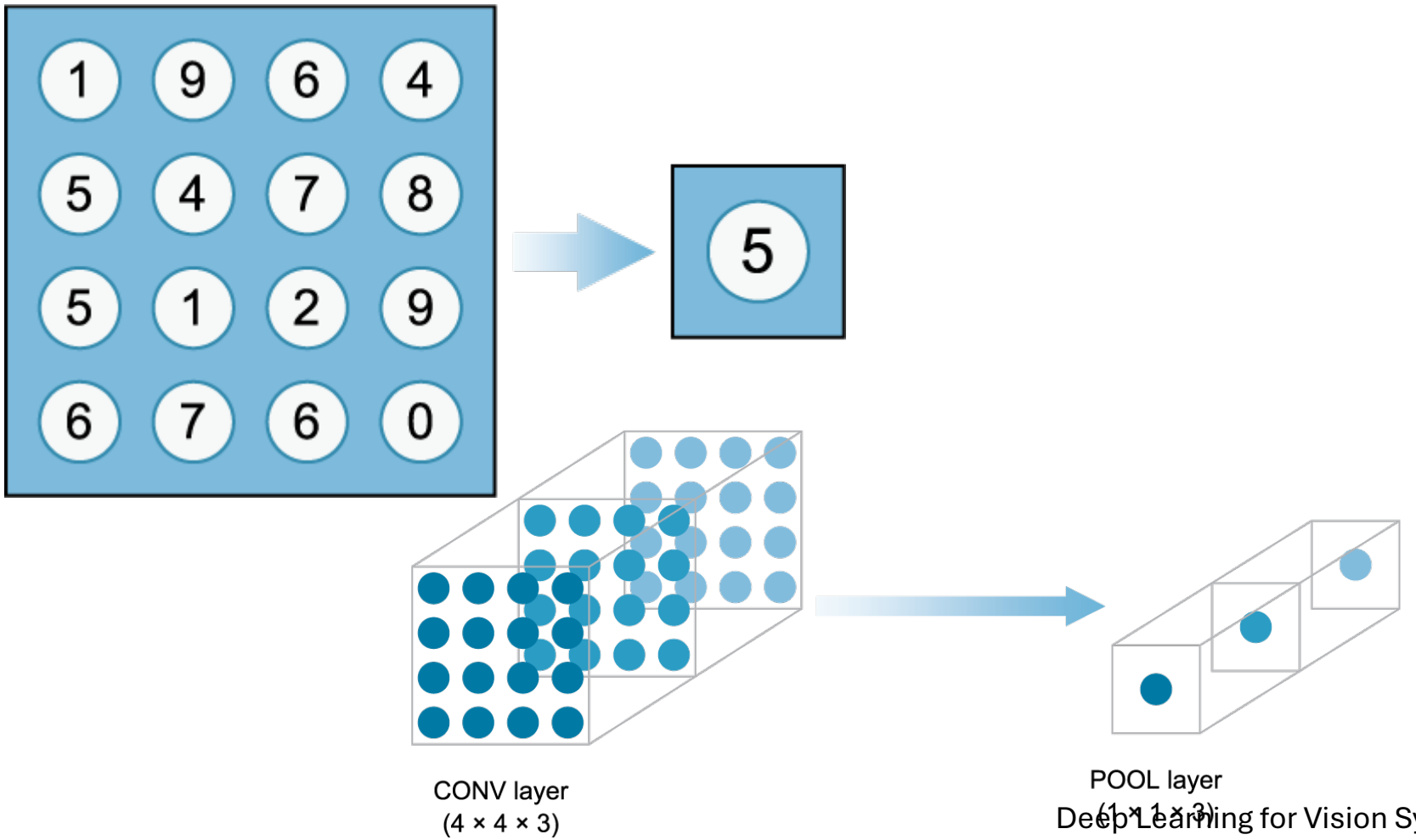


CONV layer
(4 × 4 × 3)

POOL layer
(2 × 2 × 3)

Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Average Pooling

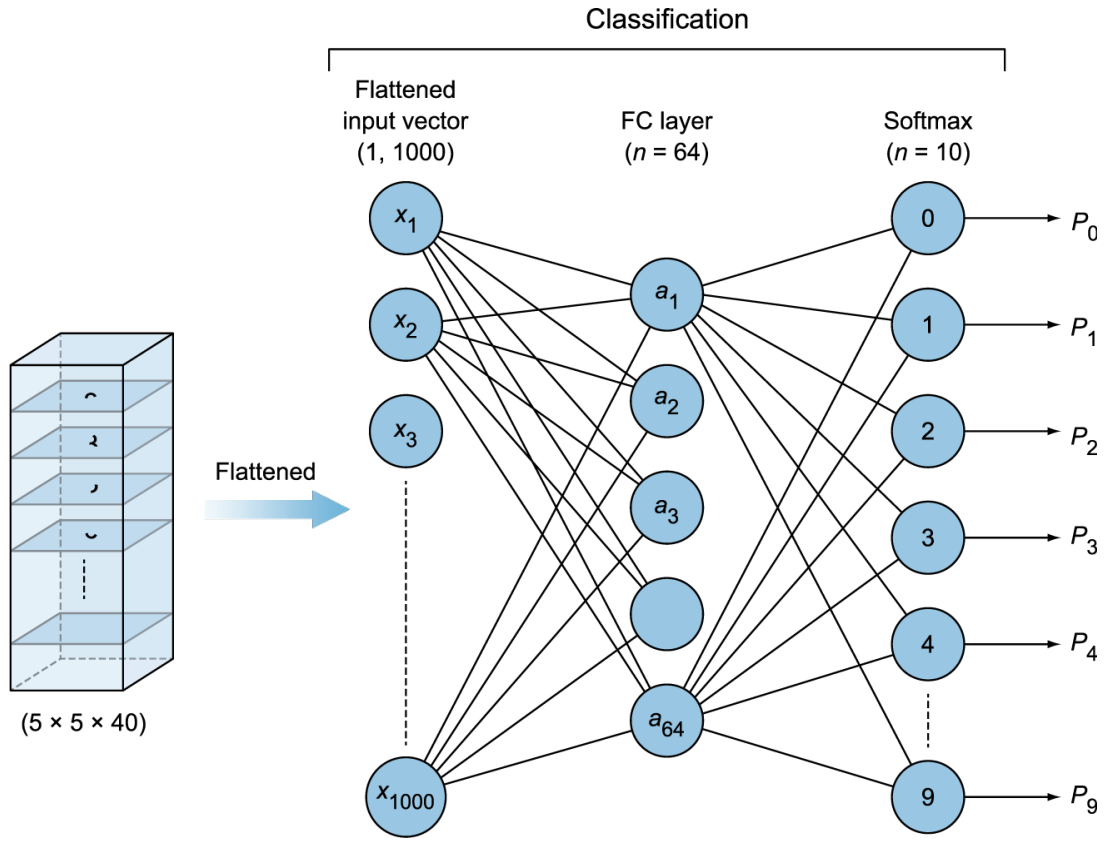


Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Fully Connected

Fully Connected Layer

- Input flattened vector
- Hidden Layer
- Output Layer



Deep Learning for Vision Systems by Mohamed Elgendy (2020)

Lab: Building CNN

```python
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), strides=1, padding='same',
          activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), strides=1, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(64, activation='relu'))

model.add(Dense(10, activation='softmax'))

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 64)	200,768
dense_1 (Dense)	(None, 10)	650

Total params: 220,234 (860.29 KB)
Trainable params: 220,234 (860.29 KB)
Non-trainable params: 0 (0.00 B)

References

- Guide to CNNs for CV – Khan et al. (2018)
- Deep Learning with Python – Chollet (2018)
- Deep Learning in Computer Vision – Awad & Hassaballah (2020)

- Deep Learning for Vision Systems by Mohamed Elgendy (2020)