Co-funded by the Erasmus+ Programme of the European Union

DeCAIR

Deep Computer Vision Using Convolutional Neural Networks

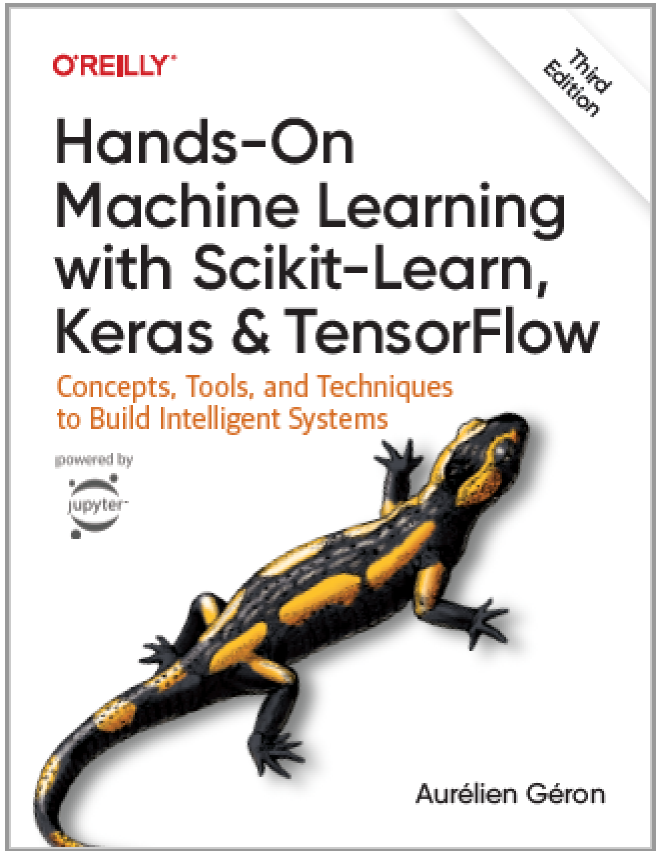
Prof. Gheith Abandah

Developing Curricula for Artificial Intelligence and Robotics (DeCAIR)
618535-EPP-1-2020-1-JO-EPPKA2-CBHE-JP

Reference

- Chapter 14: **Deep Computer Vision Using Convolutional Neural Networks**

- Aurélien Géron, **Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow**, O'Reilly, 3rd Edition, 2022
  - Material: https://github.com/ageron/handson-ml3



Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise

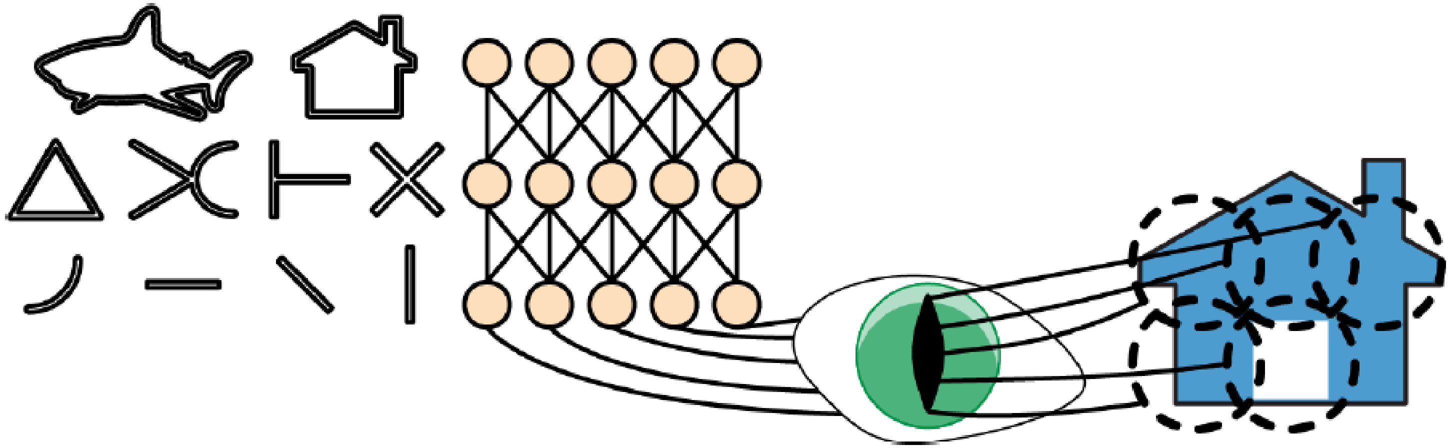
1. Introduction

- YouTube Video: **Convolutional Neural Networks (CNNs) explained** from Deeplizard

https://youtu.be/YRhxdVk_sIs

1. Introduction

- **Convolutional neural networks (CNNs)** emerged from the study of the brain's **visual cortex**.
- Many neurons in the visual cortex have a small **local receptive field**.

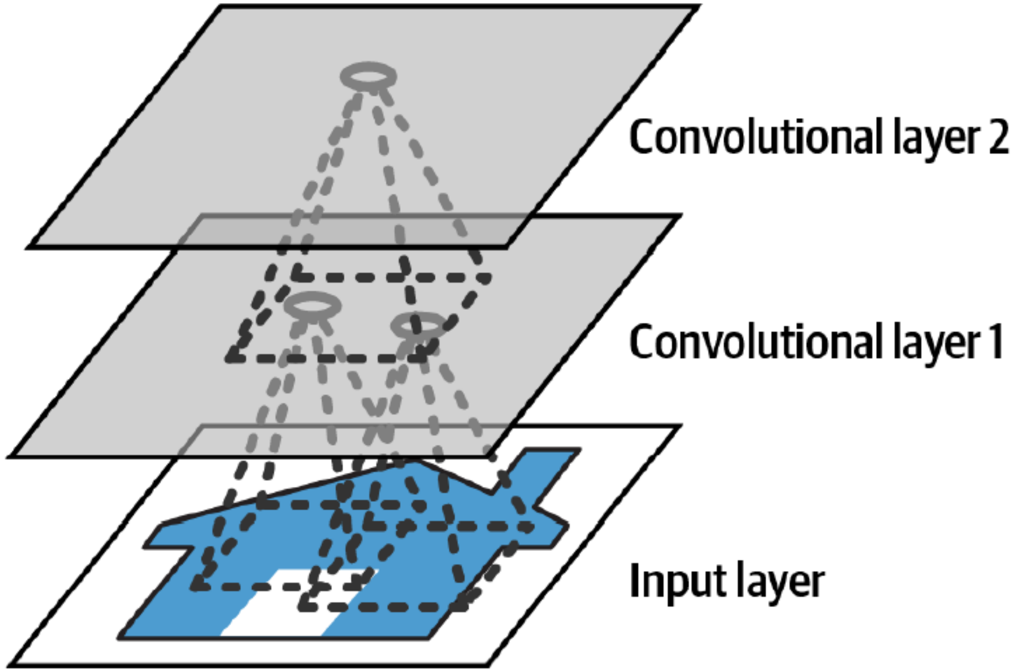


Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise

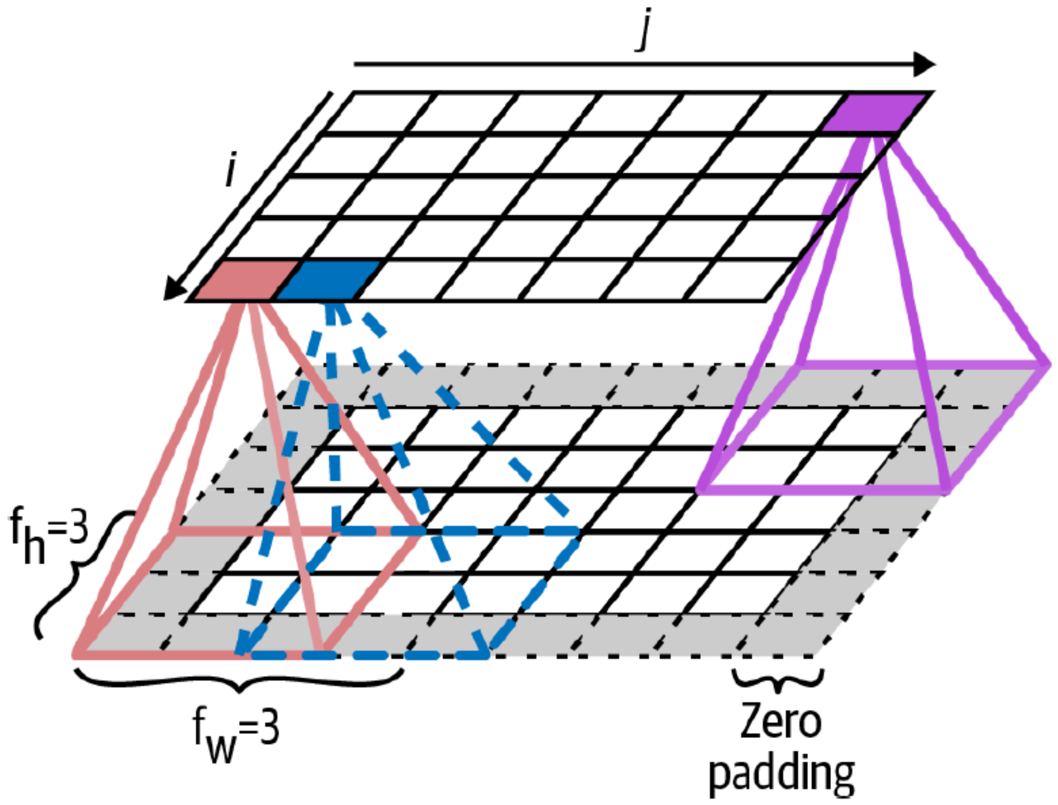
2. Convolutional Layer

- **Neurons** in one layer are not connected to every single pixel/neuron in the previous layer, but only to pixels/neurons in their **receptive fields**.
- This architecture allows the network to concentrate on **low-level features** in one layer, then assemble them into **higher-level features** in the next layer.
- Each layer is represented in **2D**.



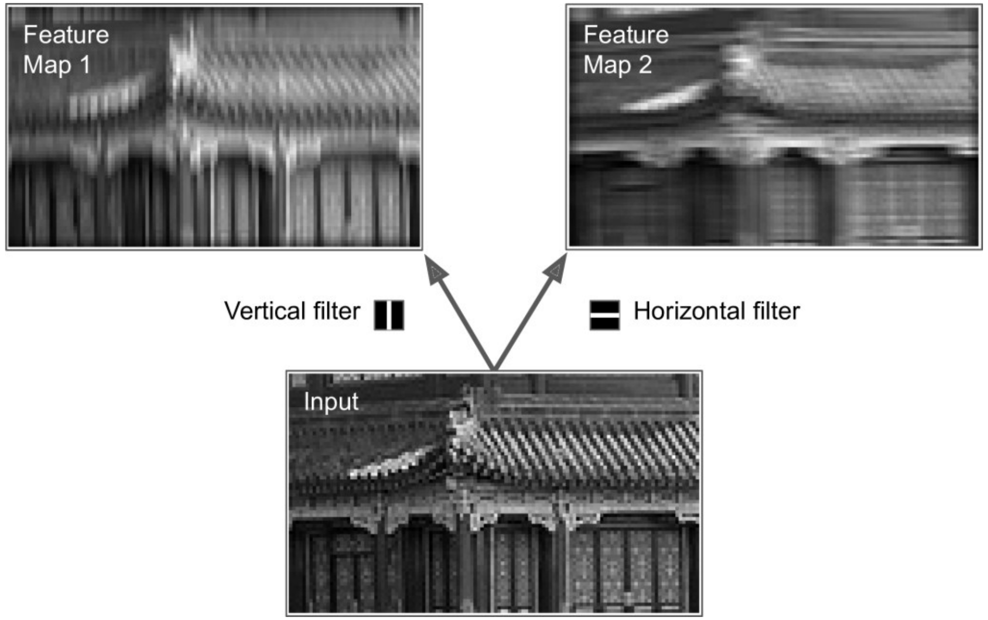
2. Convolutional Layer

- f_h and f_w are the height and width of the receptive field.
- **Zero padding**: In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs.



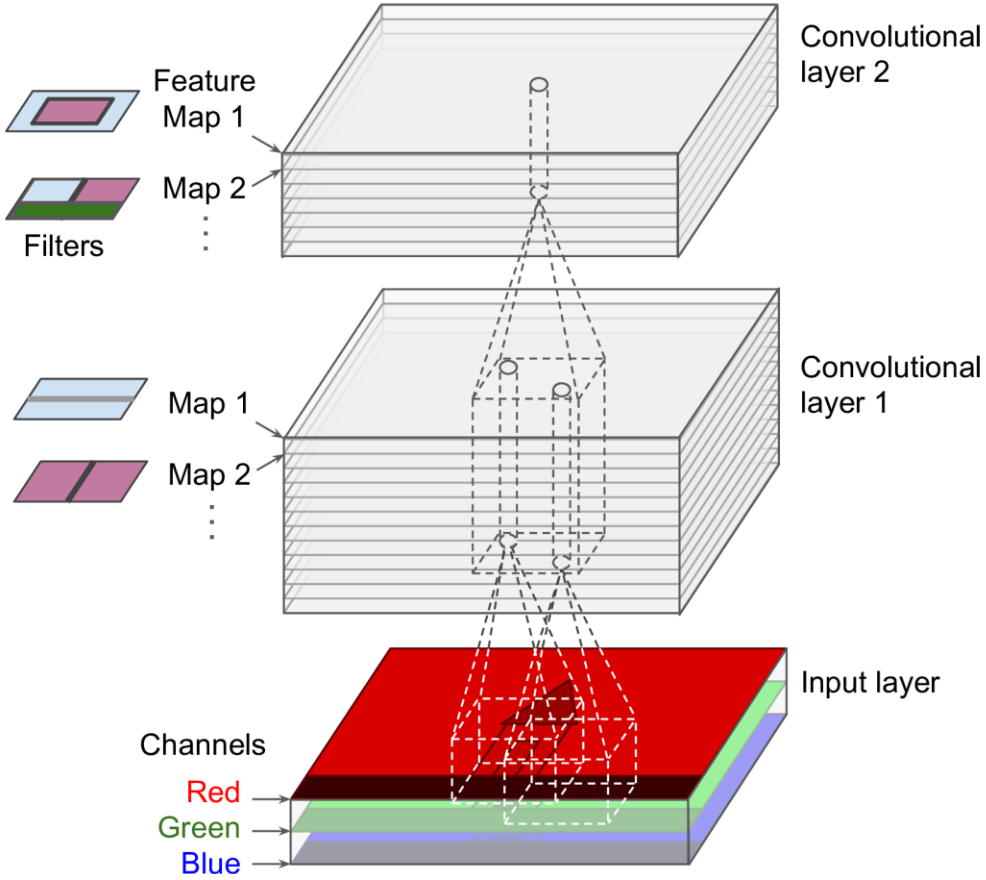
2.1 Filters

- A neuron's weights can be represented as a small image the size of the receptive field, called **filters**.
- When all neurons in a layer use the same line filters, we get the **feature maps** on the top.



2.2 Stacking Feature Maps

- In reality, each layer is **3D** composed of **several feature maps** of equal sizes.
- **Within** one feature map, all neurons **share** the same parameters, but **different** feature maps may have **different** parameters.
- Once the CNN has learned to **recognize a pattern in one location**, it can recognize it in any other location.

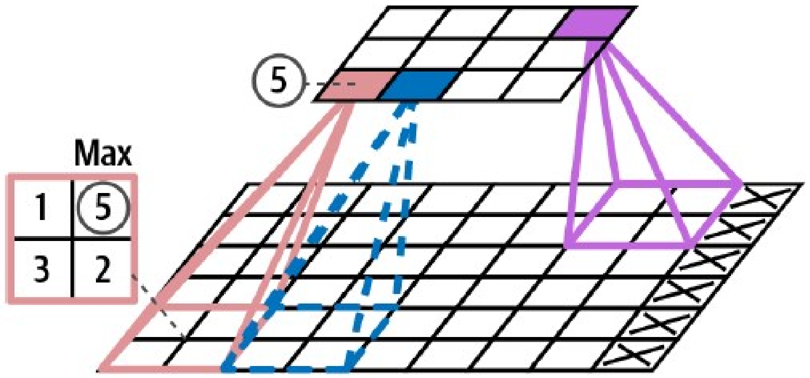


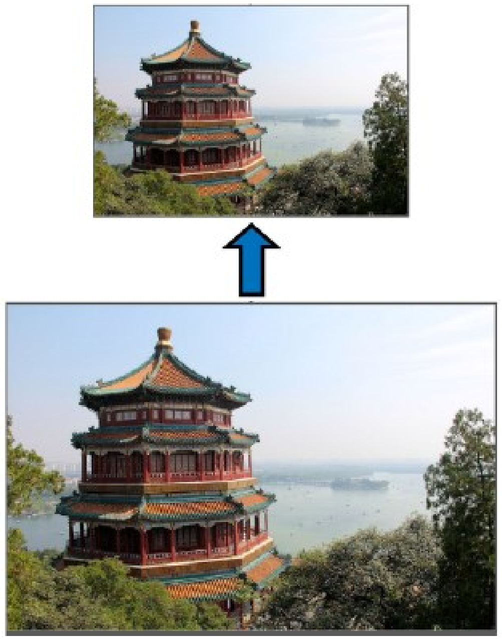
Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise

3. Pooling Layer

- Its goal is to **subsample** (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters.
- It aggregates the inputs using **max** or **mean**.



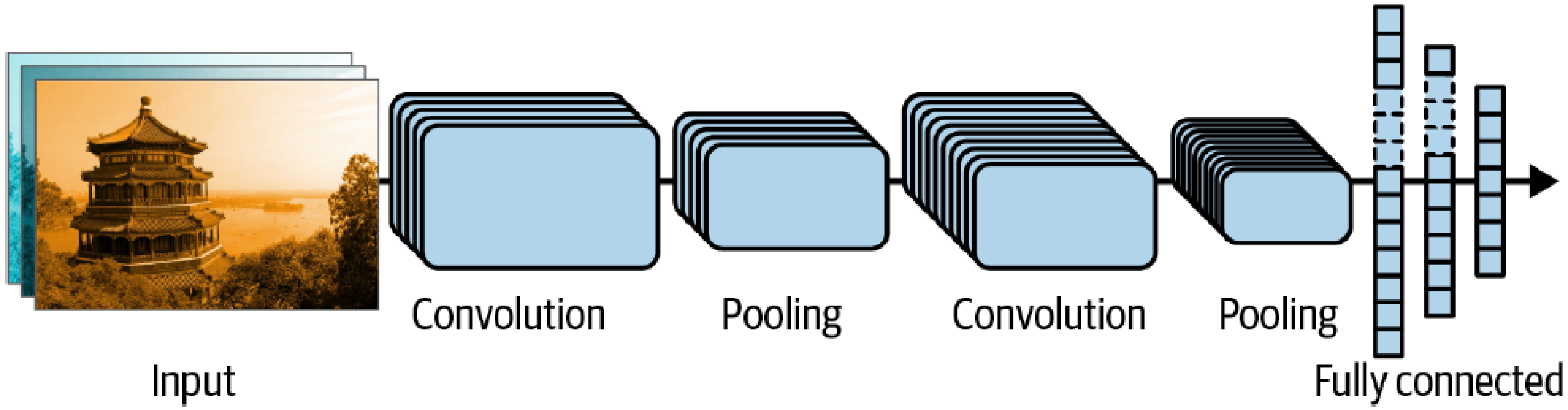


Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise

4. CNN Architectures

- **Stack** few **convolutional layers** (each one generally followed by a **ReLU** layer), then a **pooling** layer, then another few convolutional layers, then another pooling layer, and so on. The image gets **smaller and smaller**, but it also gets **deeper and deeper**. At the end, a **dense** NN is added.



Outline

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise

5. Example – Fashion MNIST

```
model = keras.models.Sequential([
    keras.layers.Conv2D(64, 7, activation="relu", padding="same",
        input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(128, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.Conv2D(256, 3, activation="relu", padding="same"),
    keras.layers.MaxPooling2D(2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation="relu"),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation="softmax")
])
```

Filter size

Feature maps

2×2 window and stride 2

6. Exercise

14.9. Build your own CNN from scratch and try to achieve the highest possible accuracy on **MNIST**.

Summary

1. Introduction
2. Convolutional layer
    1. Filters
    2. Stacking feature maps
3. Pooling layer
4. CNN architectures
5. Example – Fashion MNIST
6. Exercise