DATASCI207-005/007 Applied Machine Learning

Vilena Livinsky, PhD(c)

School of Information, UC Berkeley

Week 10: 03/12/2025 & 03/13/2025

Today's Agenda

- Convolutional Neural Networks
- Walkthroughs:
  - CNN intro
  - CNN use-case (revisiting Drug Reviews)

Final Project: Step 2 (Reminder)

Form a group

- 3-4 people, max 4
- NO silos or groups of 2
- Groups can only be composed of you and your colleagues in your section

Inform me & the class of your formed group in Slack

- Include names of group members
- Due date: 01/24/2025 EOD

General Plan:

- ~~Step 1: form a group~~
- Step 2: submit your group's question to answer/ goal + dataset
- Step 3: baseline presentation
- Step 4: final presentation

Dates

- Step 2: 03/13/2025 EOD
- Step 3: 04/03/2025
- Step 4: 04/17/2025

Final Project: Logistics/ Due Dates (Reminder)

Final Project Timeline/ Deliverables

- Step1: See groups on Slack
- **Step 2: Select dataset** and identify a leading **question/goal** for your use-case
  - notify via email of dataset + question selection for you group (vlivinsky@ischool.berkeley.edu)
  - Due date: 03/13/2025 EOD
- **Step 3: Baseline** group presentation (10 mins)
  - Due date: 04/03/2025
- **Step 4: Final** Project Presentations (15 mins)
  - Due date: 04/17/2025
- For past project examples refer to: Cornelia Paulik

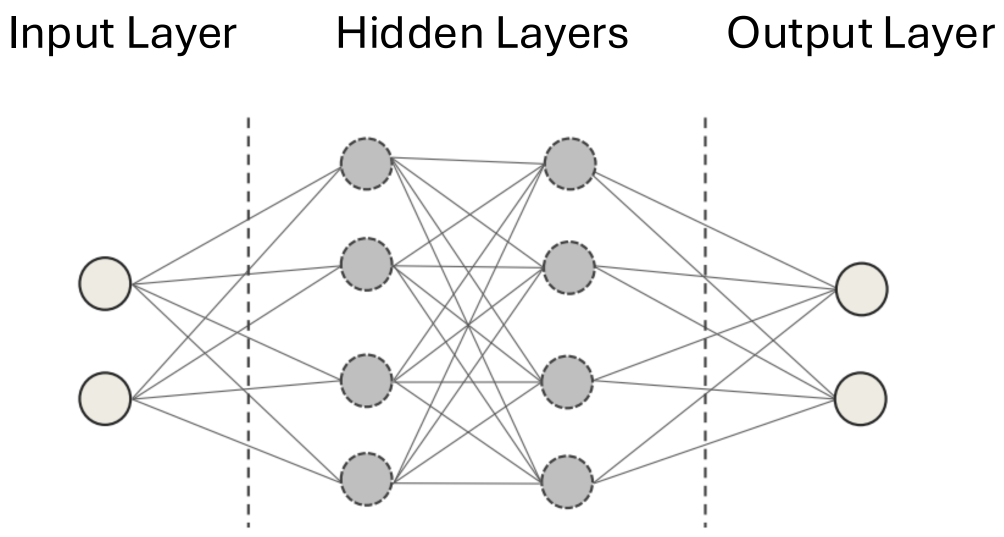
Make sure your **baseline presentation** slides include:

- Title, Authors
- What is the **question** you will be working on? Why is it interesting?
- What is the **data** you will be using? Include the data source, size of dataset, main features to be used. Please also include summary statistics of your data.
- What prediction **algorithms** do you plan to use? Please describe them in detail.
- How will you **evaluate** your results? Please describe your chosen performance metrices and/or statistical tests in detail.

Refer to bcourses home page for **final presentation** guidelines and grading

- Note that the final project grade is individual and is based on each member's contribution
- Final project team member reviews to be submitted at end of class—a survey will be sent

Neural Nets: Dense Networks vs. ConvNets



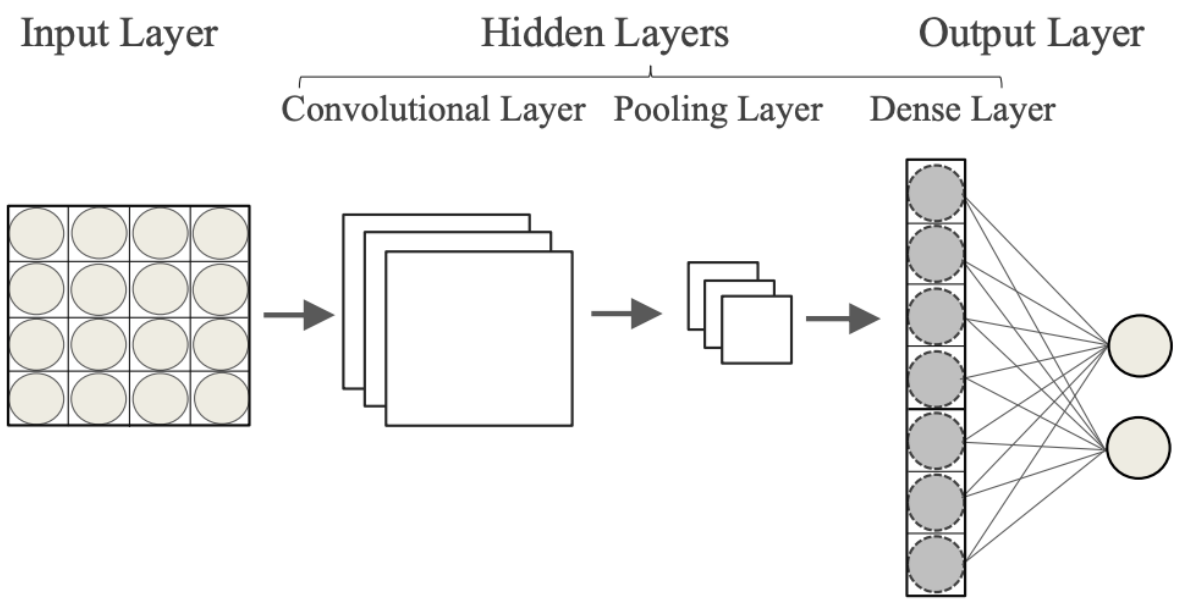


Image Credit: Nazari, F., & Yan, W. (2021). Convolutional versus dense neural networks: Comparing the two neural networks performance in predicting building operational energy use based on the building shape. *arXiv preprint arXiv:2108.12929*.

Neural Networks

Dense

- Learning global patterns
- Ex.:
  - MNIST dataset:
  - learning patterns with all pixels

Convolutional

- Learning local patterns
  - Small windows of inputs

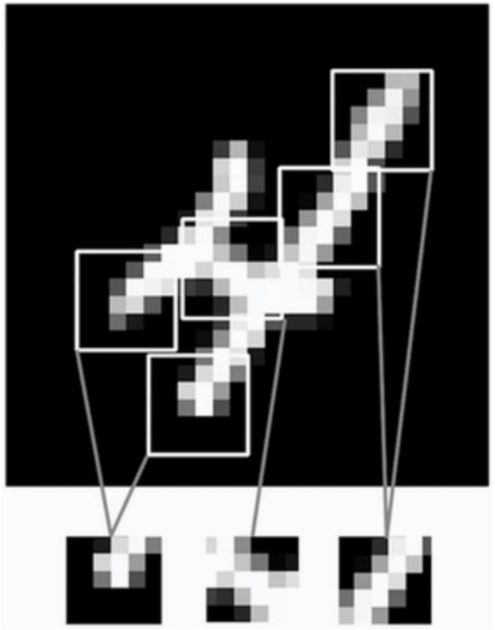


Image Credit: Deep Learning with Python, Second Edition, Francois Chollet

Convolutional Layer: Input & Response Map

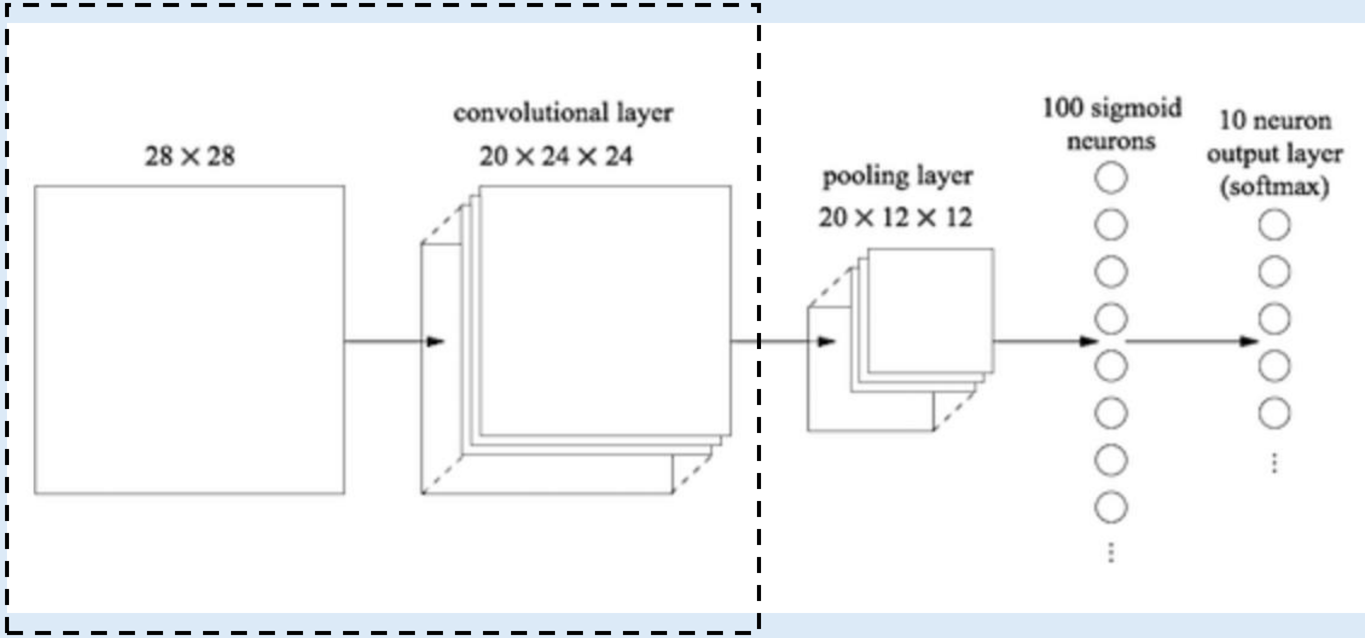


Image Credit: Neural Networks and Deep Learning, Michael Nielsen, Dec 2019

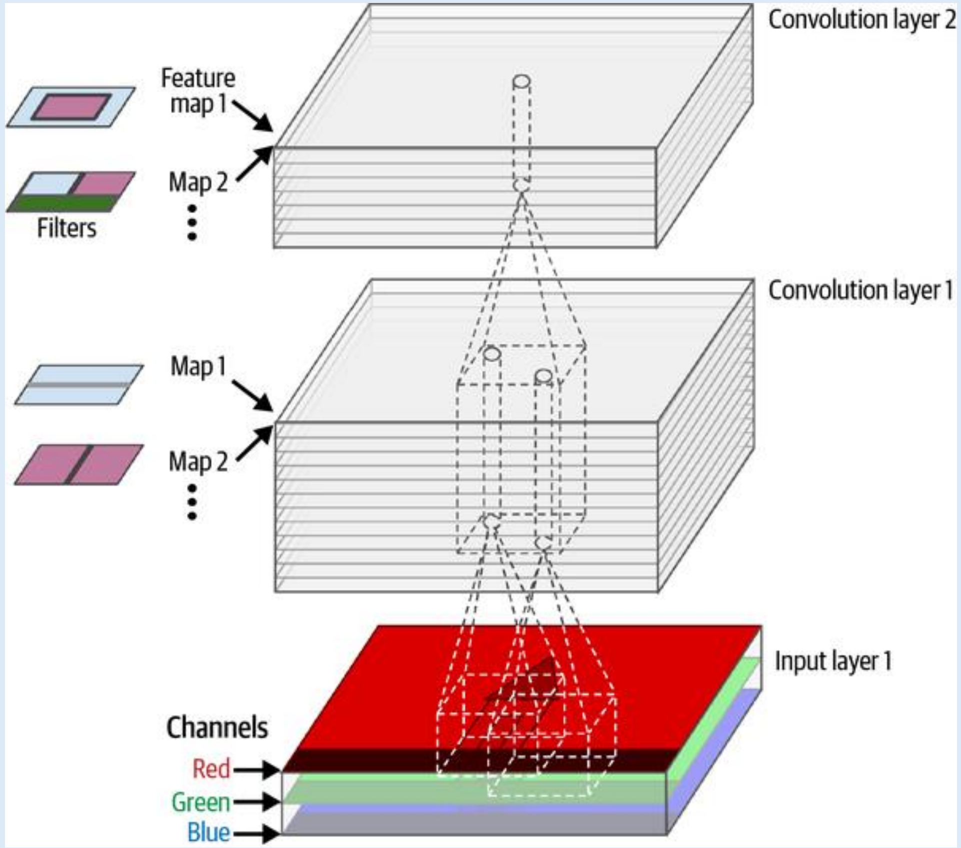
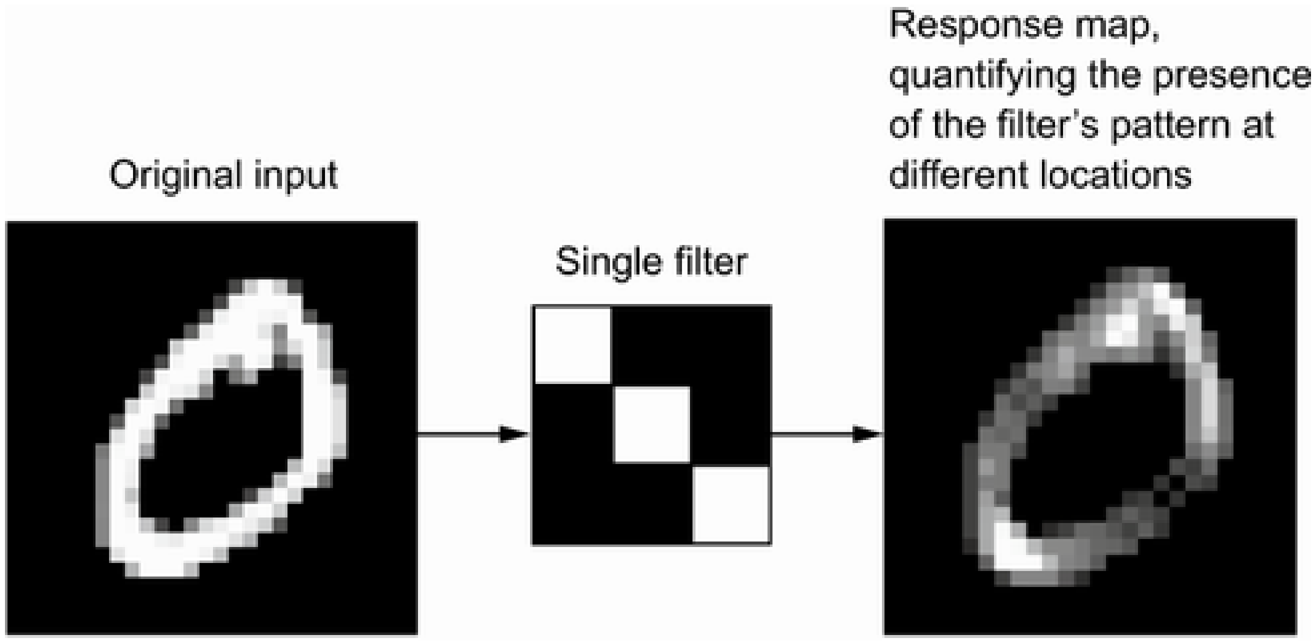


Image Credit: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition, Aurélien Géron

The Workings of a Response Map

- a 2D map of the presence of a pattern at different locations in an input
  - i.e. looking for a specific spatial feature in the input



Source: Deep Learning with Python, Second Edition, Francois Chollet

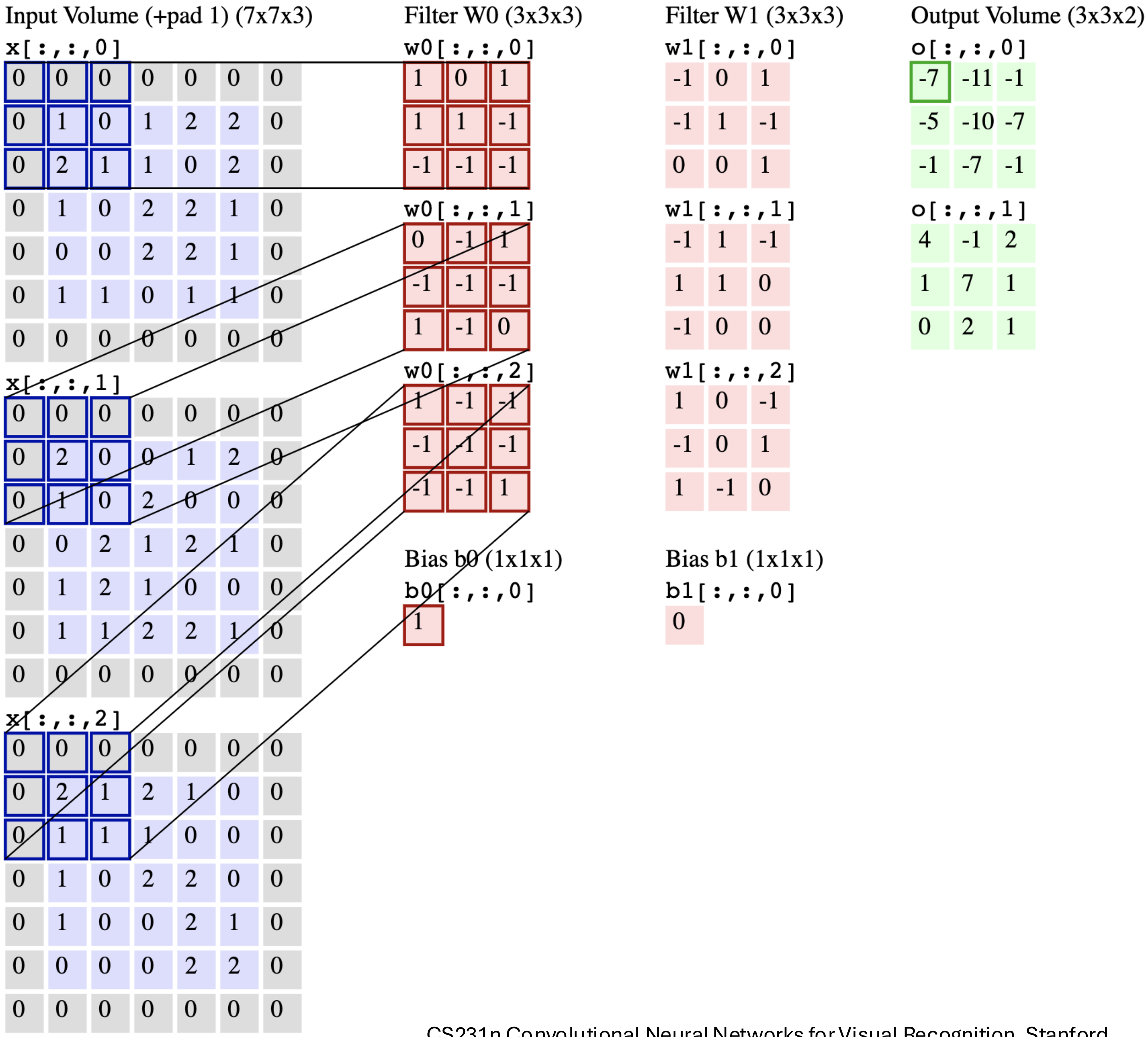
Convolutional Layers

Learning spatial features

Note: "Parameter sharing"

- We constrain the neurons in each depth slice to use the same weights and bias

    - An animated version of the still image (right) can be found at:
    - https://cs231n.github.io/convolutional-networks/

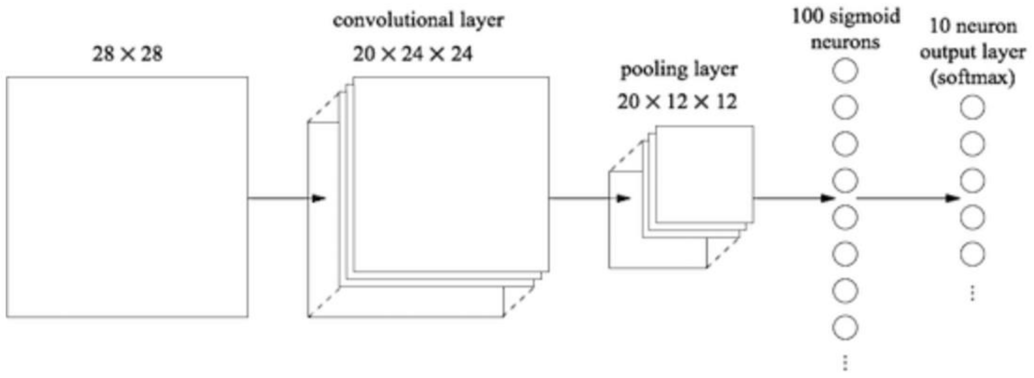


ConvNets:
Properties

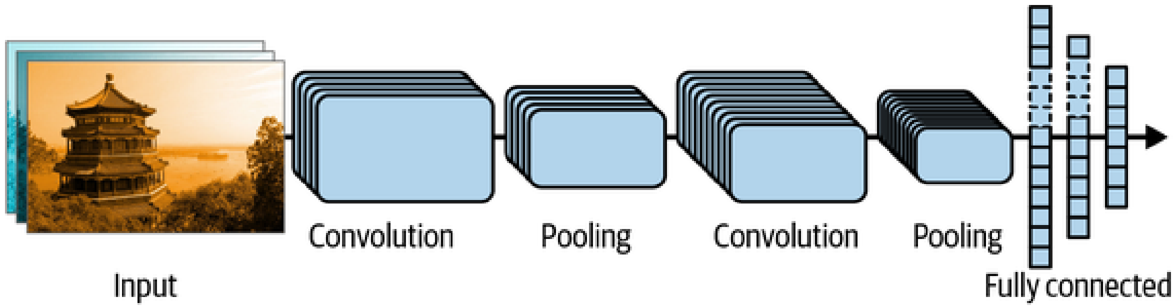
Patterns learned are **translation-invariant**

Spatial **hierarchies** of patterns are learned

ConvNets: Architecture



Source: Neural Networks and Deep Learning, Michael Nielsen, Dec 2019



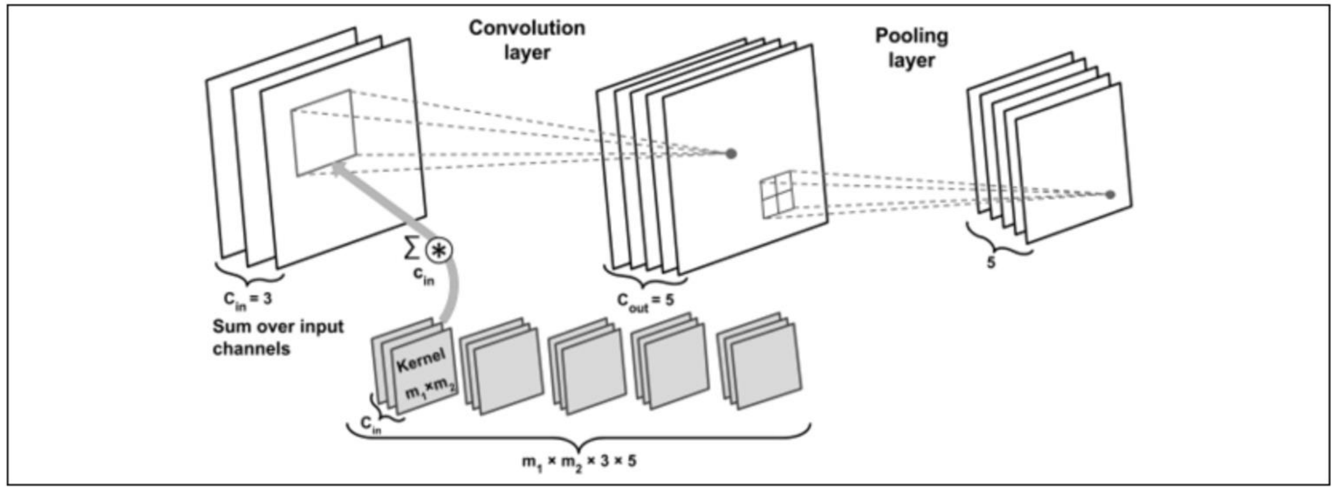
Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition, Aurélien Géron

- Lower layers usually identify features in small areas of the images
- Higher layers combine the lower-level features into larger features

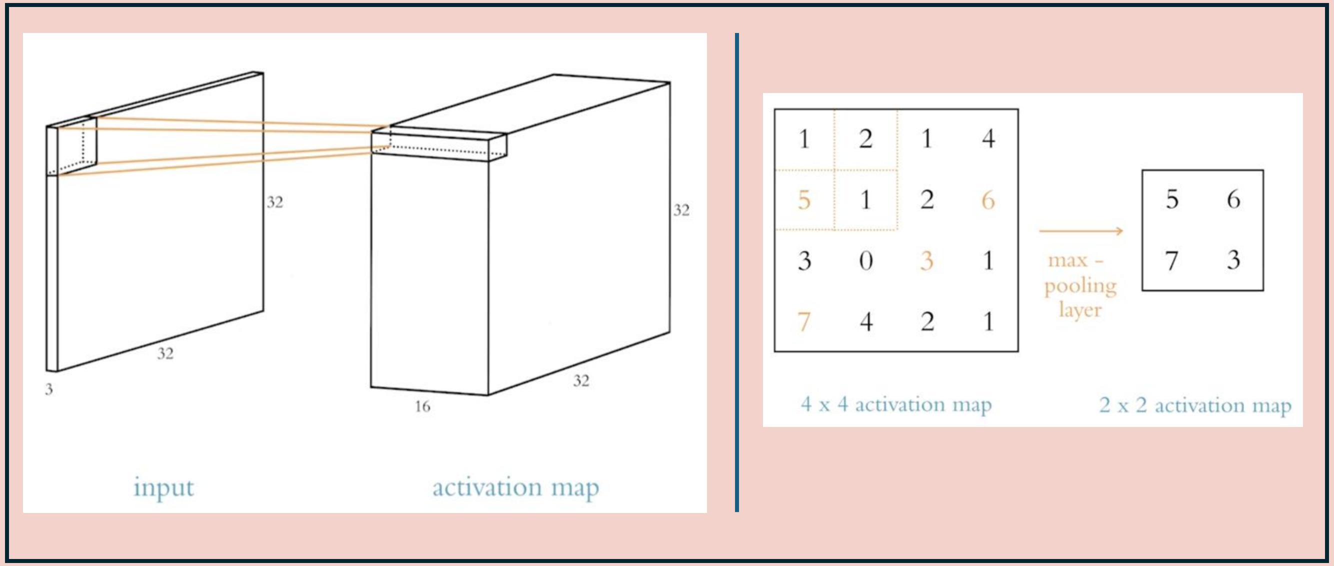
Max-Pooling

- Reduces activation maps spatially (leaves depth intact)
  - downsamples feature maps, much like strided convolutions
  - Result:
    - Reduces parameter count
    - Reduces complexity
    - (+) speeds up computation
    - (+) combats overfitting
- Note:
  - Conv. Layer:
  - transforming local patches via a <u>learned</u> linear transformation (the convolution kernel)

  - Pooling Layer:
  - transformation via a hardcoded max/avg tensor operation
    - Max vs. Avg-Pooling?



Source: Python Machine Learning - Third Edition, Sebastian Raschka, Vahid Mirjalili



Source: Deep Learning with TensorFlow, Keras, and PyTorch, Jon Krohn

Consider

1. Calculate parameter count for: *conv2d & conv2d_1*
2. Which layer/s has/have learnable parameters?
3. What does an output map from a convolutional layer indicate? (Qualitatively/Conceptually? Quantitatively?)

```python
tf.keras.backend.clear_session()
tf.random.set_seed(42)
np.random.seed(42)

model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, kernel_size=3, padding="same",
                           activation="relu", kernel_initializer="he_normal"),
    tf.keras.layers.Conv2D(64, kernel_size=3, padding="same",
                           activation="relu", kernel_initializer="he_normal"),
    tf.keras.layers.MaxPool2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(128, activation="relu",
                          kernel_initializer="he_normal"),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation="softmax")
])

model.build(input_shape=(None, 28, 28, 1))
model.compile(loss="sparse_categorical_crossentropy", optimizer="nadam",
              metrics=["accuracy"])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18,496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dropout (Dropout)	(None, 12544)	0
dense (Dense)	(None, 128)	1,605,760
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 1,625,866 (6.20 MB)
Trainable params: 1,625,866 (6.20 MB)
Non-trainable params: 0 (0.00 B)

Appendix

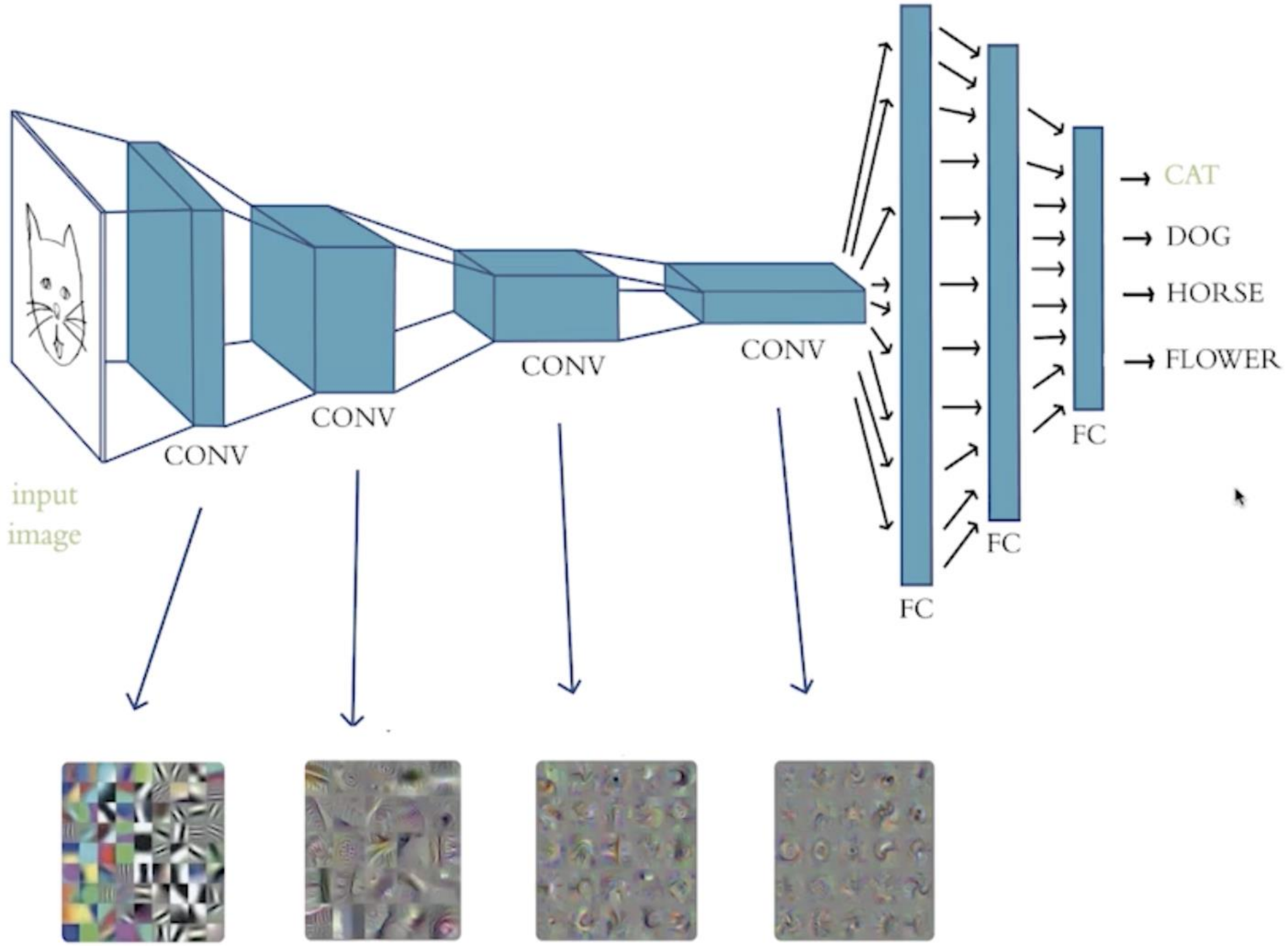
Size Before and After Convolutions

Feature map size:

$$\underset{\text{output width}}{O} = \frac{\overset{\text{input width}}{W} - \overset{\text{kernel width}}{K} + 2\overset{\text{padding}}{P}}{\underset{\text{stride}}{S}} + 1$$

Source: Sebastian Raschka, Intro to Deep Learning

AlexNet: Krizhevsky et al., 2012



Deep Learning with TensorFlow, Keras, and PyTorch, Jon Krohn

Ex. AlexNet: Transfer Learning

- Using AlexNet to learn spatial attributes characteristic of a typical vs a non-typical population
  - https://www.researchgate.net/publication/364806159_Early_Detection_of_Autism_in_Children_Using_Transfer_Learning