
# 3546 – Deep Learning

## Module 3: Convolutional Networks

# Course Syllabus

| Module | Topic | Deliverables |
|---|---|---|
| 1 | Course Intro + Review | Term Project Released |
| 2 | Model Tuning | Assignment 1 Released |
| 3 | Convolutional Networks | |
| 4 | Deep Computer Vision | Assignment 1 Due, A2 Released |
| 5 | Recurrent Neural Networks | |
| 6 | Natural Language Processing | |
| 7 | Deep Models for Text | Assignment 2 Due, A3 Released |
| 8 | Representational Learning & Variational Methods | |
| 9 | Deep Generative Models | Assignment 3 Due, A4 Released |
| 10 | Speech and Music Recognition & Synthesis | |
| 11 | Term Project Presentations A | Term Project Due |
| 12 | Term Project Presentations B | Assignment 4 Due |


# Learning Outcomes for this Module

- Understand the theory and motivation behind Convolutional Neural Networks (CNNs).
- Be able to build and train a basic CNN using Keras.
- Gain exposure to some of the most important CNN architectures of recent years.
- Be able to apply transfer learning to re-use the learned features from an existing model for a new target task.

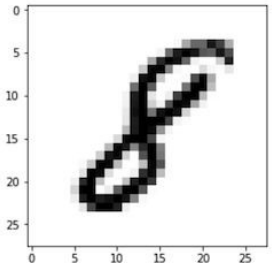
# Topics for this Module

- **3.1** Introduction to Convolutional Networks
- **3.2** Building CNNs with Keras (Notebook)
- **3.3** Data Preprocessing Techniques (Notebook)
- **3.4** Milestone CNN Architectures
- **3.5** Transfer Learning with CNNs (Notebook)
- **3.6** Resources and Wrap-Up

Module 3 – Section 1

# Intro to Convolutional Networks

# Why Not Dense Layers?



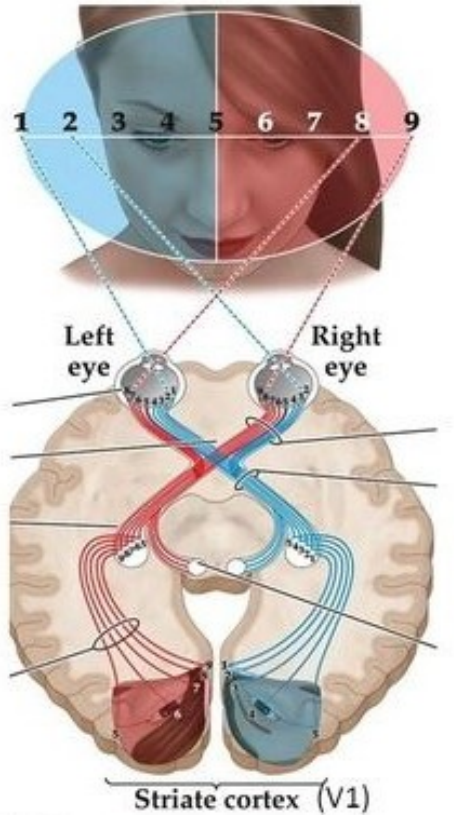
Simple MNIST digits problem…

…versus what a self-driving car must keep track of.

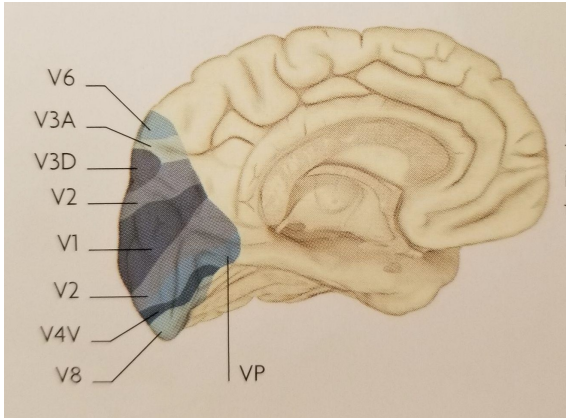
- Fully-connected (i.e. “Dense”) layers **do not scale well** to deep networks and larger images.
  - For a 100 x 100 pixel image, a single layer with 1000 neurons already has 10 million weights.

- The large number of tunable parameters can lead to **severe overfitting**.
  - Even digital noise on a few pixels could cause a misclassification.

# Motivations from Biology




**The Visual Cortex**



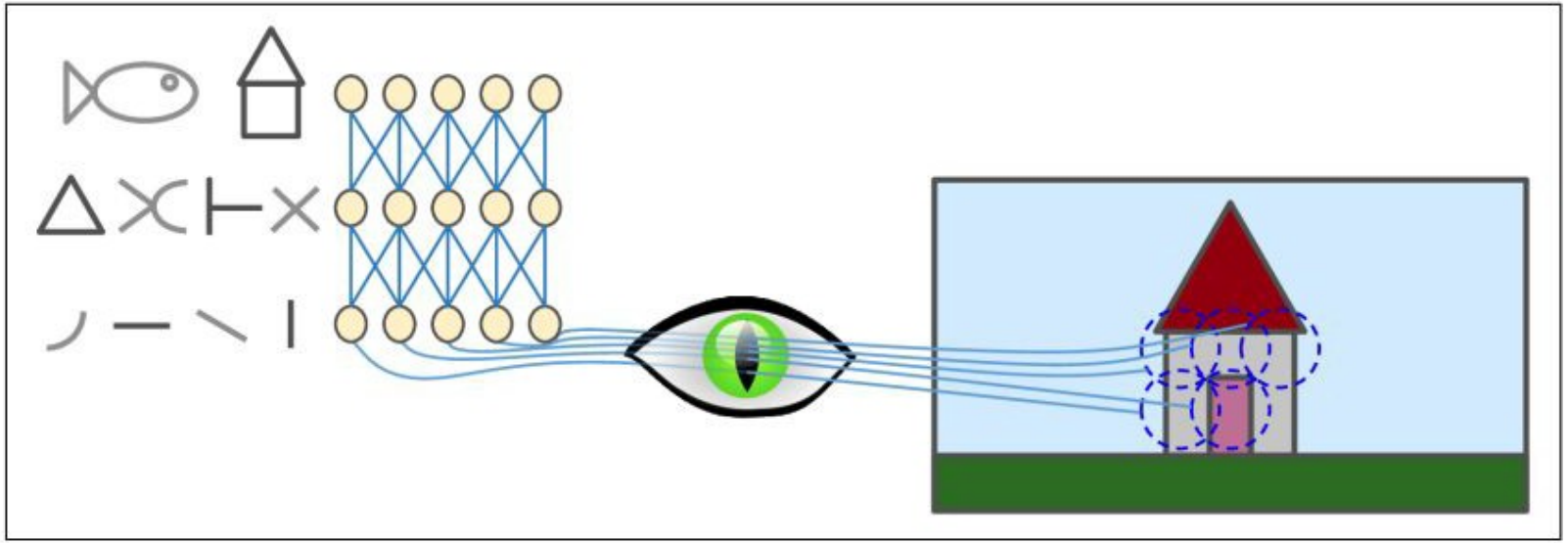

Img Source: The Human Brain Book, 2nd Ed., by Rita Carter, DK Publishing, (2009)

| Area | Function |
|---|---|
| V1 | Responds to visual stimuli |
| V2 | Responds to complex shapes |
| V3A, V3D, VP | Registers angles and symmetry; combines motion and direction. |
| V4D, V4V | Responds to colour, orientation, form, and movement. |
| … | … |



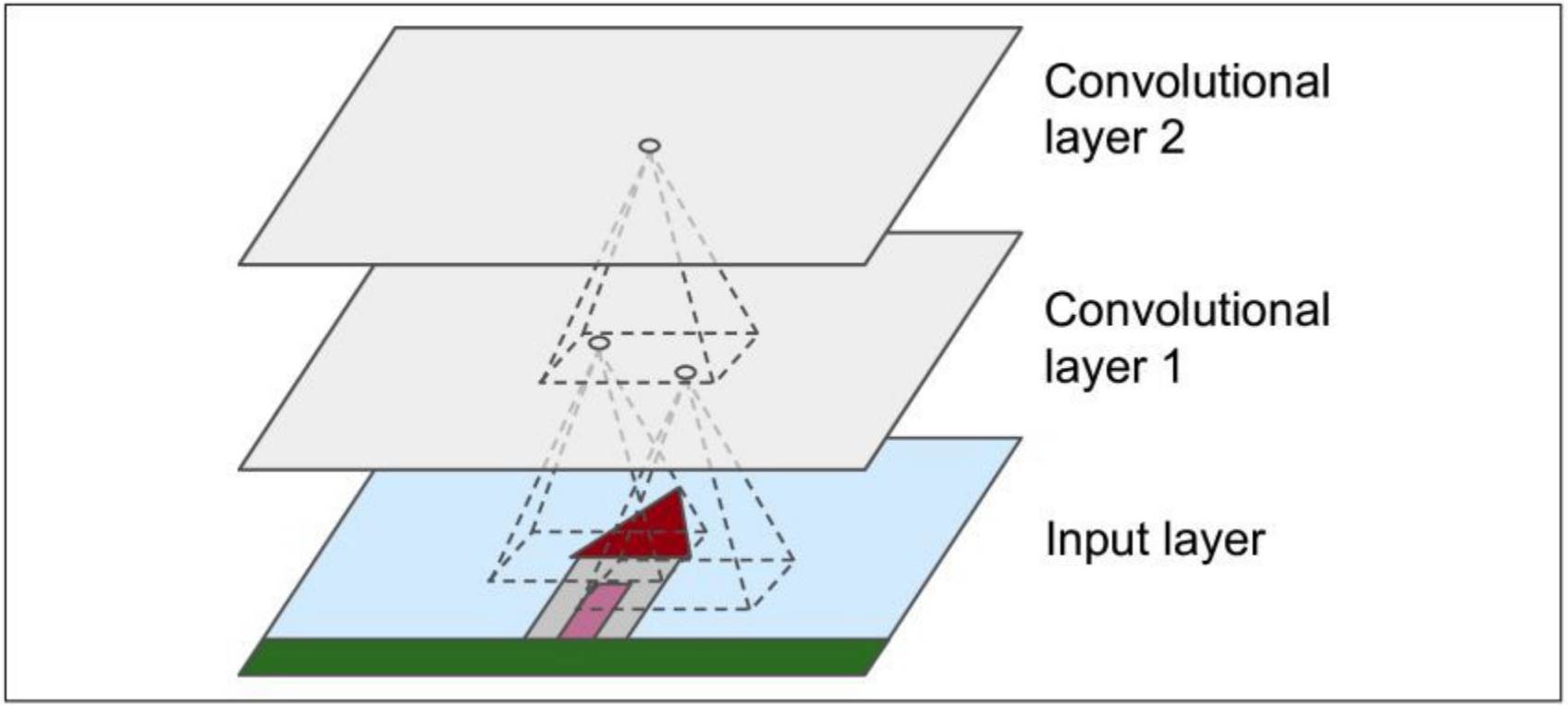
Source: https://neurologism.com/2018/03/22/in-what-format-is-information-stored-

# Local Receptive Fields



Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 447 (2019).

# Convolutional Layers




Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 448 (2019).

# Convolution

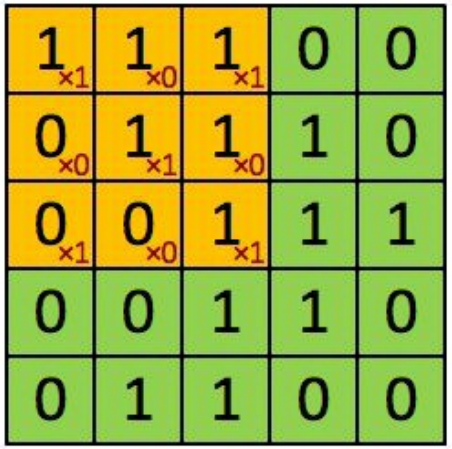
The “discrete” 2D convolution of **X** and **Y**:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} * \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

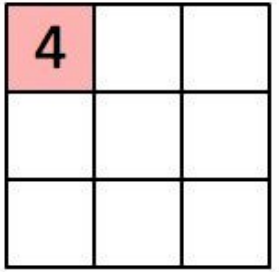
$$= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{(m-i)(n-j)} y_{(1+i)(1+j)}$$

In our case:

- **X** is the image
- **Y** is the filter (weights)



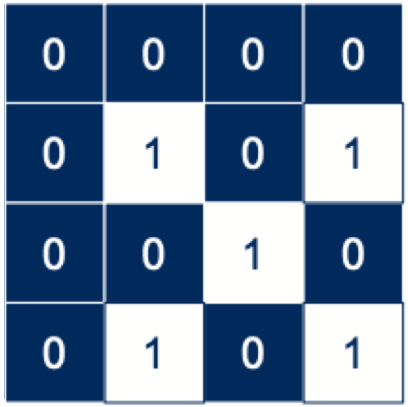

Image



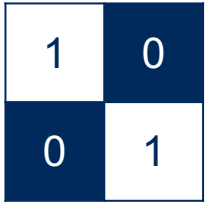

Convolved Feature

# Weights as Filters

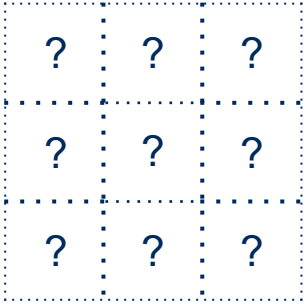
**Input**




**Weights**

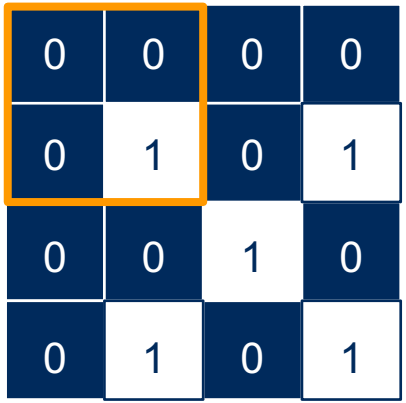



**Output**

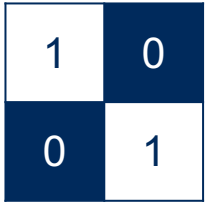


# Weights as Filters

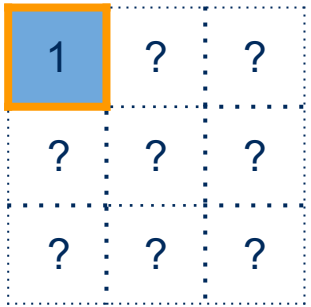
**Input**




**Weights**



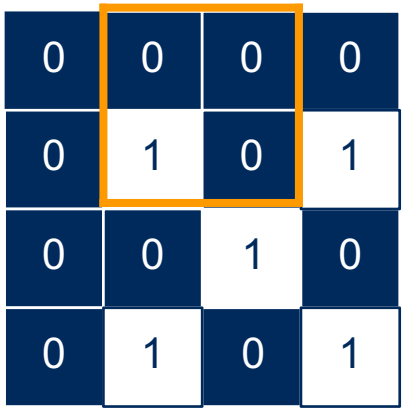

**Output**



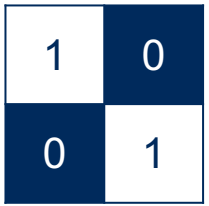

(0 * 1) + (0 * 0) +
(0 * 0) + (1 * 1) = 1

# Weights as Filters

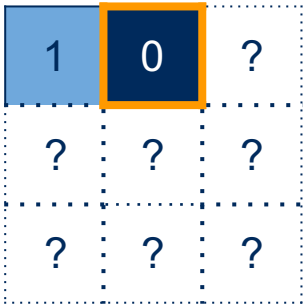
**Input**




**Weights**




**Output**

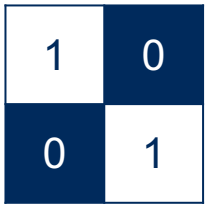



(0 * 1) + (0 * 0) +
(1 * 0) + (0 * 1) = 0

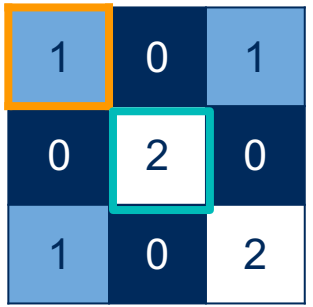
# Weights as Filters

**Input**


**Weights**




**Output**

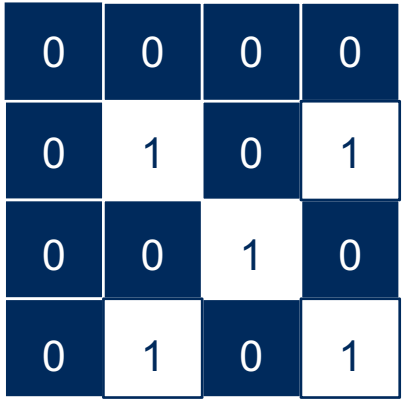



(0 * 1) + (0 * 0) +
(0 * 0) + (1 * 1) = 1

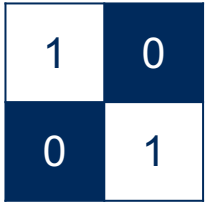
(1 * 1) + (0 * 0) +
(0 * 0) + (1 * 1) = 2

# Weights as Filters

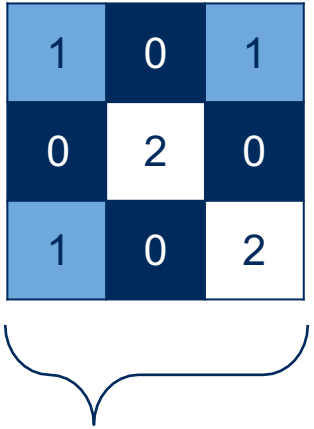
**Input**




**Weights**

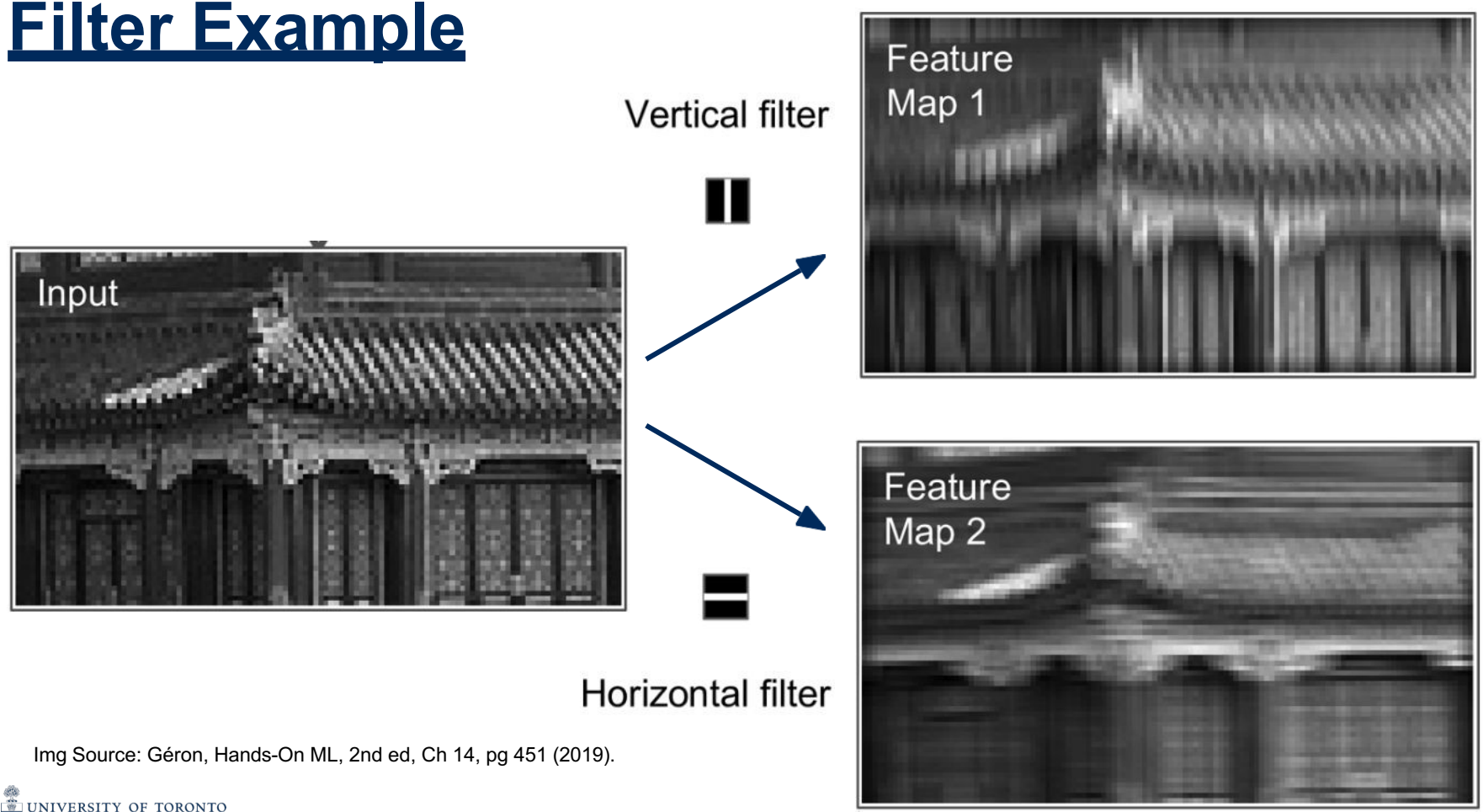



**Output**



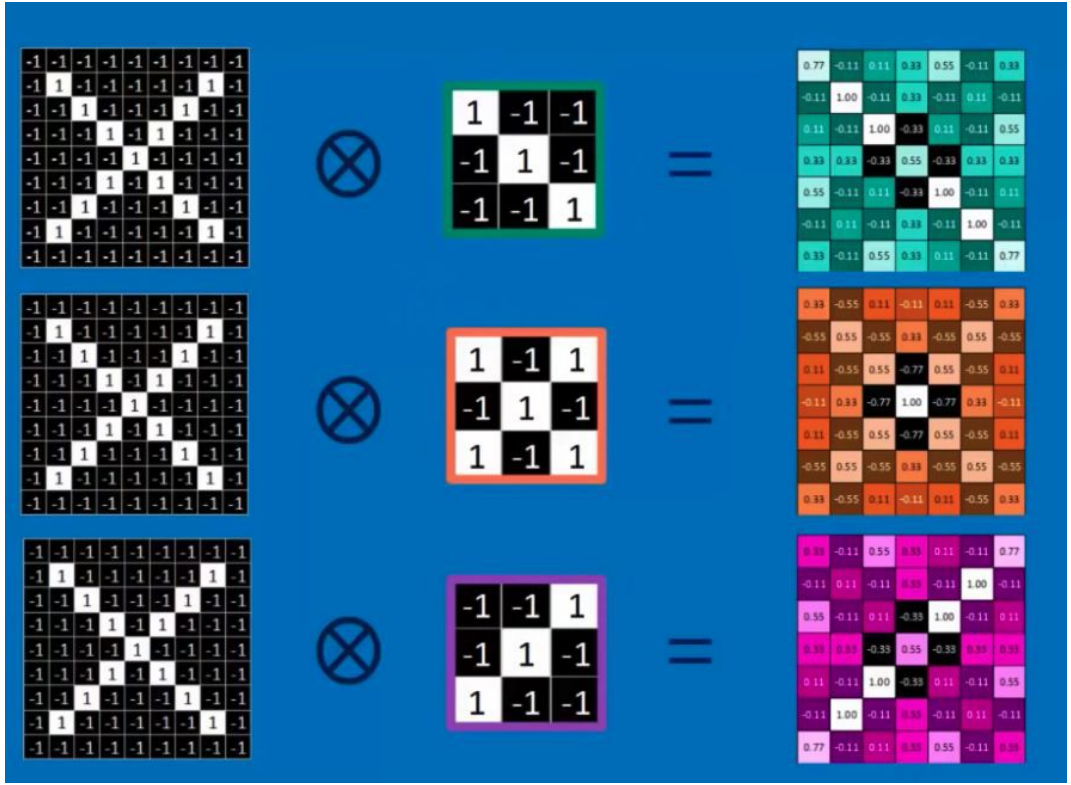

Emphasizes where a ‘match’ to the filter was detected.

# Filter Example




Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 451 (2019).

# Applying Multiple Filters in a Layer



Multiple output **feature maps** (or 'channels') add a **depth** dimension to our layers.

Img Source: "How Convolutional Neural Networks work" by Brandon Rohrer, https://www.youtube.com/watch?v=FmpDIaiMIeA

# Stacking Feature Maps

**Layer Weights Tensor Shape**

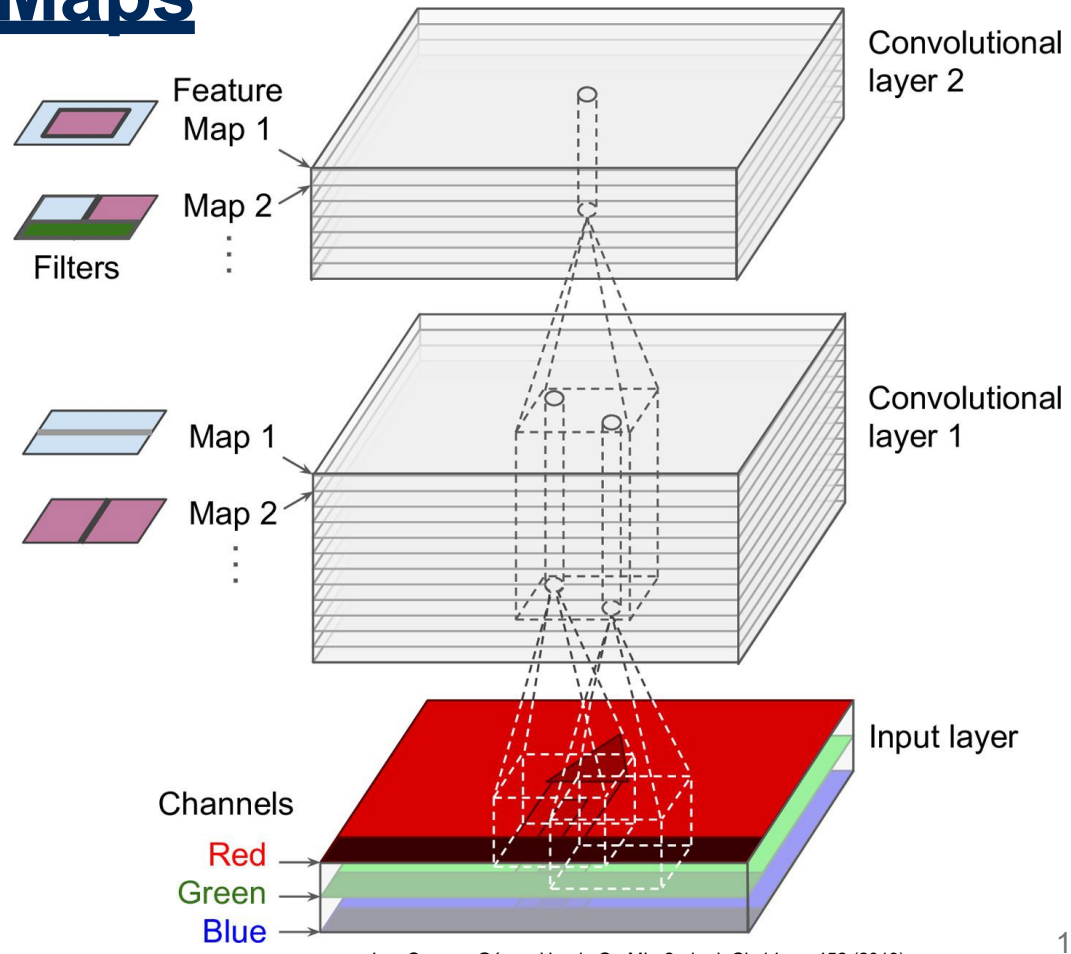
$$(f_h, f_w, k_{\text{in}}, k_{\text{out}})$$

$f_h$ = receptive field height, pixels
$f_w$ = receptive field width, pixels
$k_{in}$ = num input channels
$k_{out}$ = num filters in layer

**Input Tensor Shape**

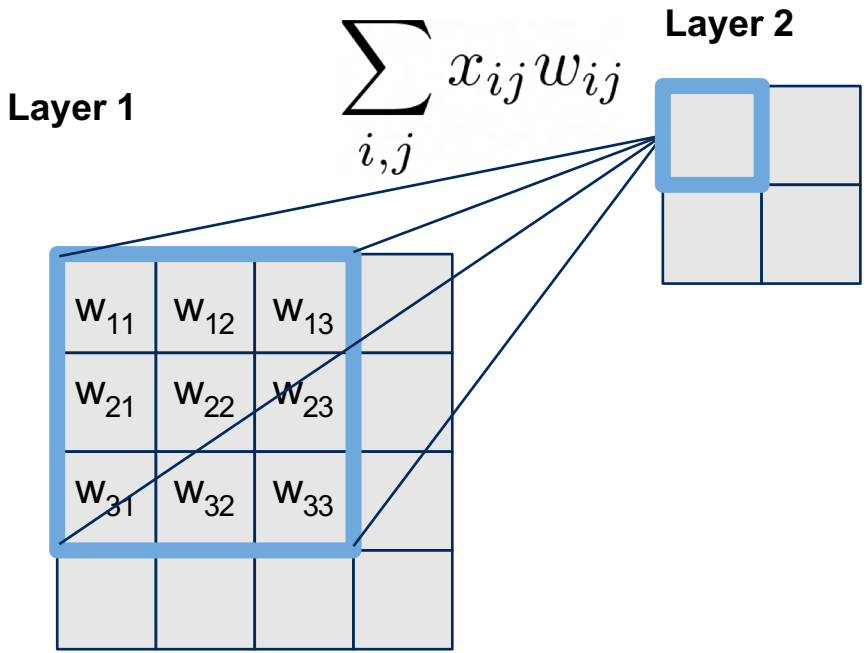
$$(n, h, w, c)$$

n = minibatch size
h = image height, pixels
w = image width, pixels
c = num channels (e.g. 3 for RGB)

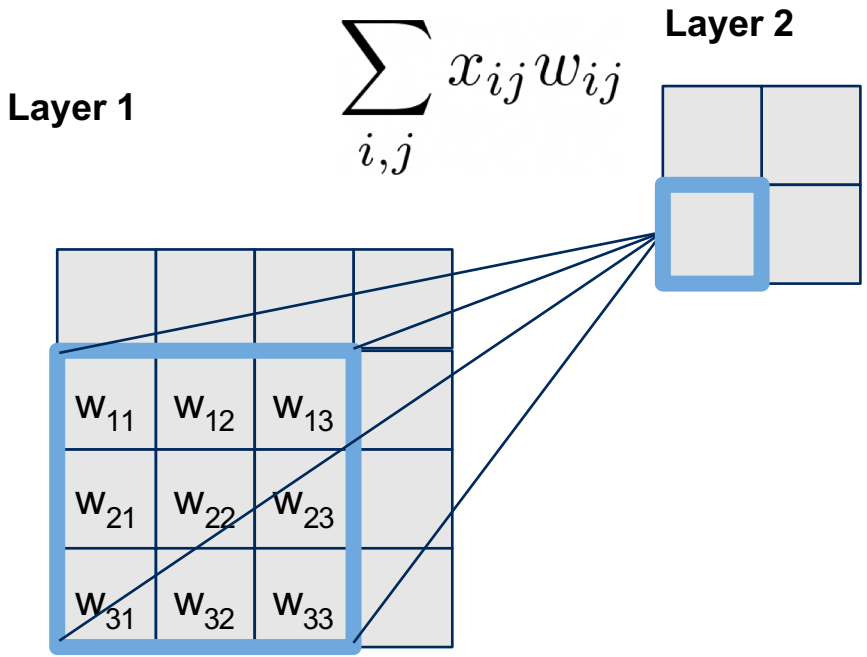



Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 452 (2019).

# Shared Weights

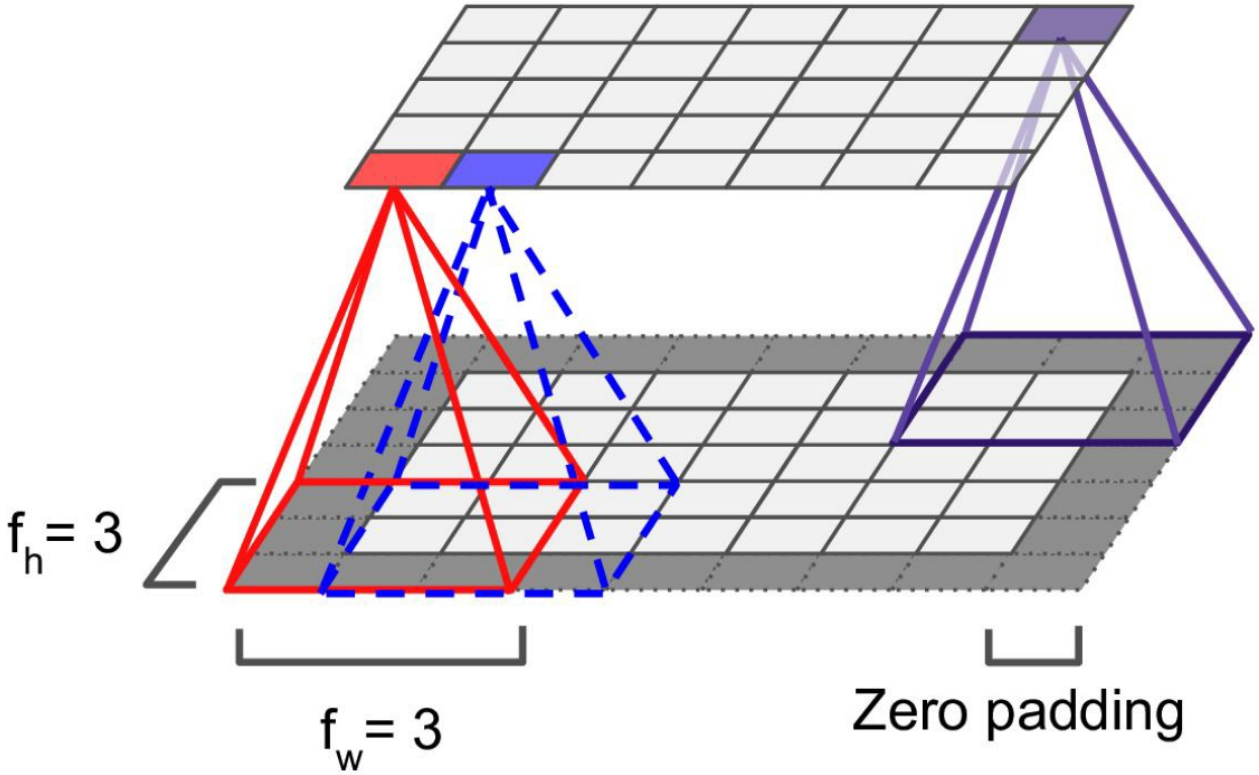


# Shared Weights




Same weights!

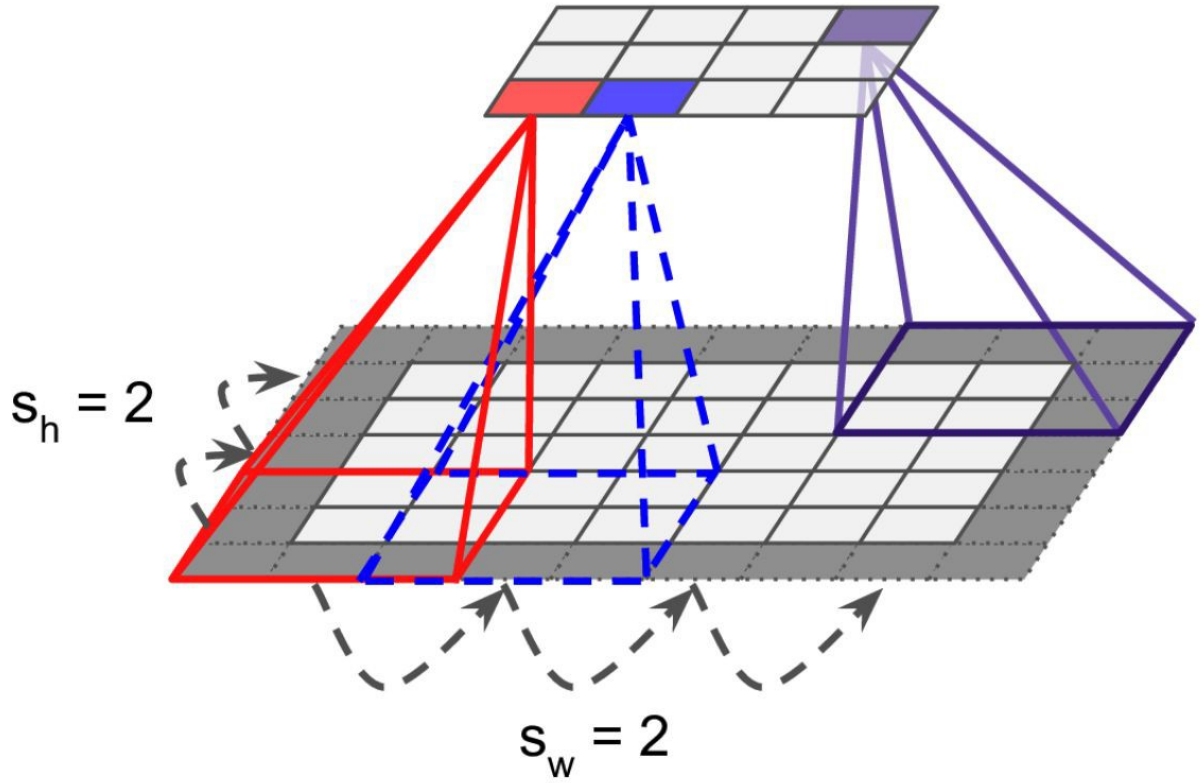
# Controlling Layer Size: Padding






Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 452 (2019).

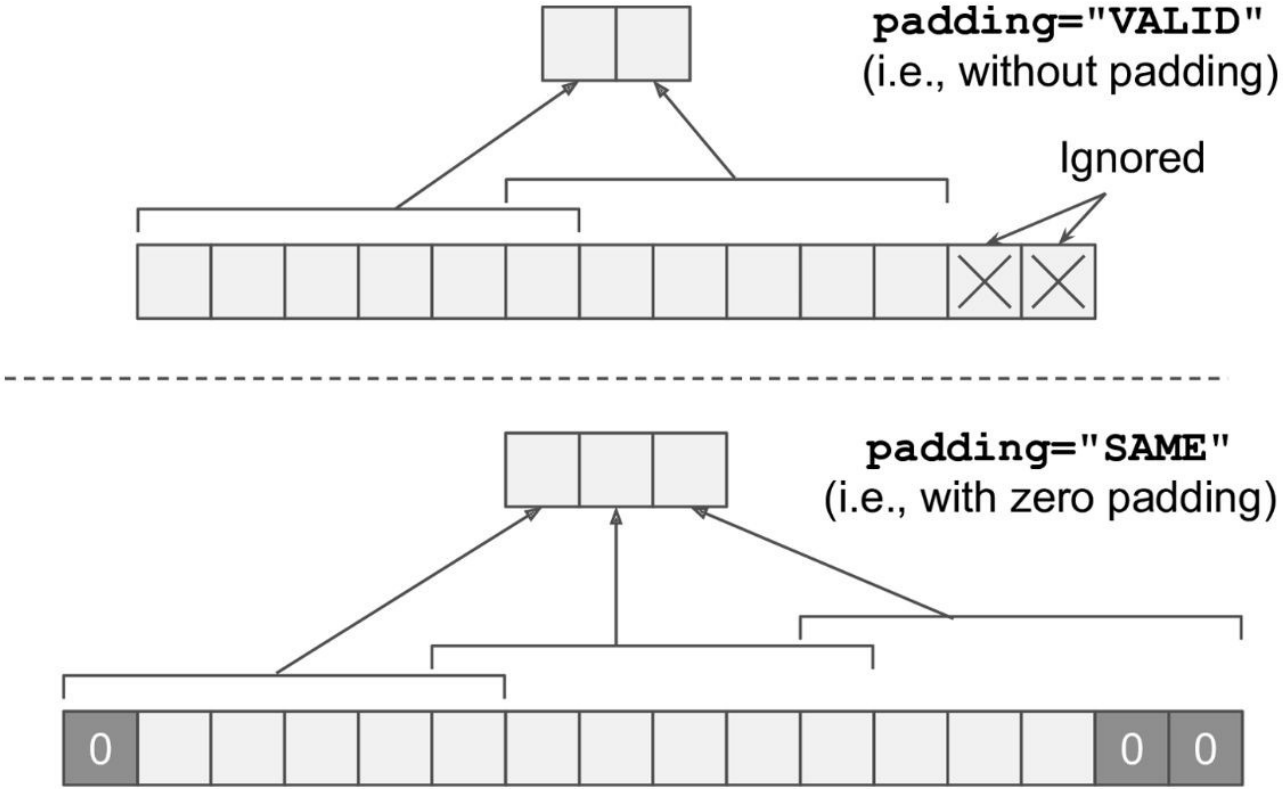
# Controlling Layer Size: Stride






Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 450 (2019).

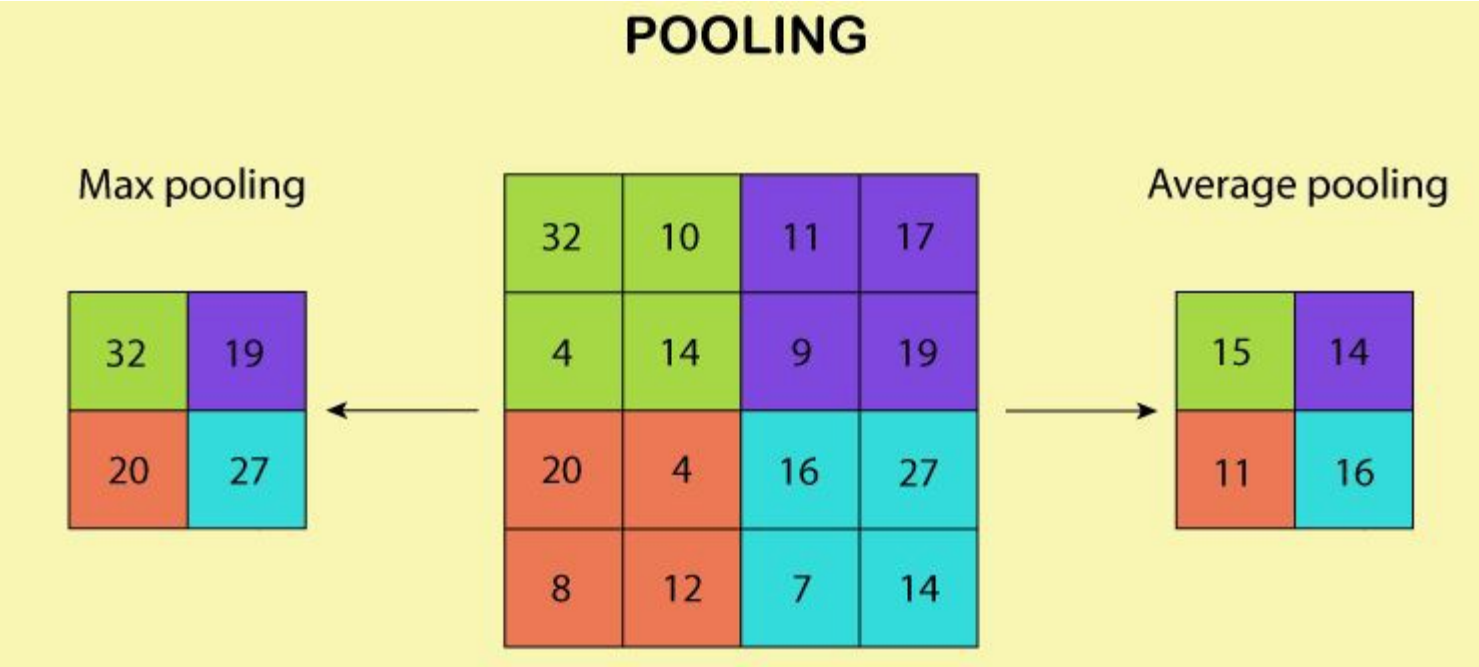
# “Same” vs. “Valid” Padding





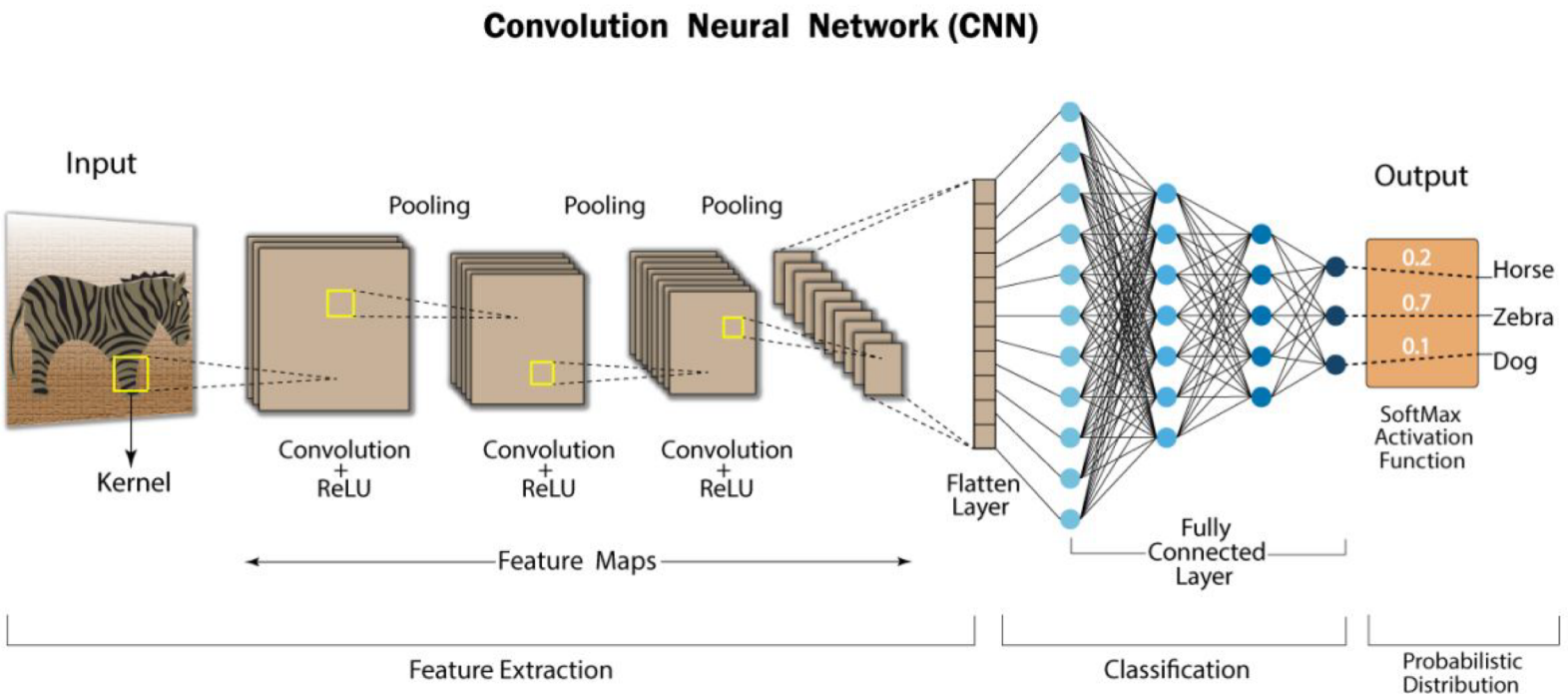

Img Source: Géron, Hands-On ML, 2nd ed, Ch 14, pg 450 (2019).

# Pooling Layers



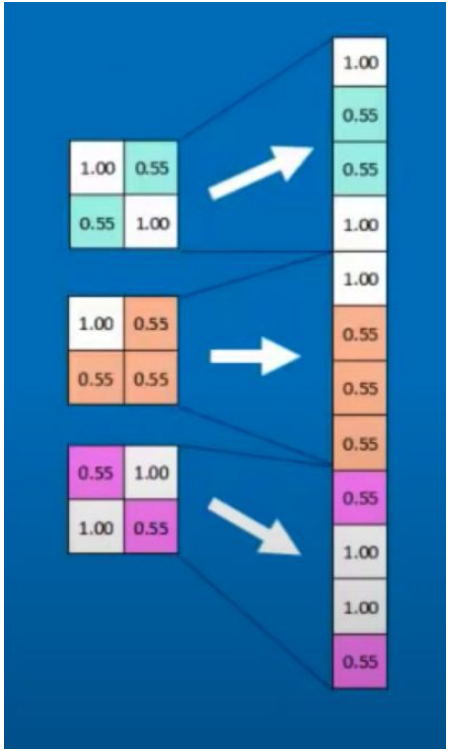

Img source: https://developersbreach.com/convolution-neural-network-deep-learning/

# Putting It All Together




Img source: https://developersbreach.com/convolution-neural-network-deep-learning/

# Details of “Flatten” Step




Img Source: “How Convolutional Neural Networks work” by Brandon Rohrer, https://www.youtube.com/watch?v=FmpDIaiMIeA

## Module 3 – Section 2

# Building CNNs with Keras

→ *See Interactive Notebook!*

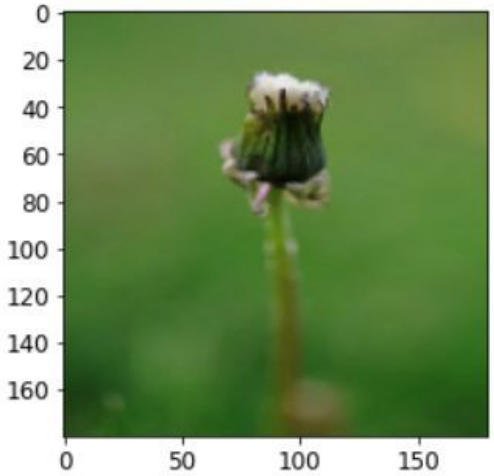
## Module 3 – Section 3

# Data Preprocessing Techniques

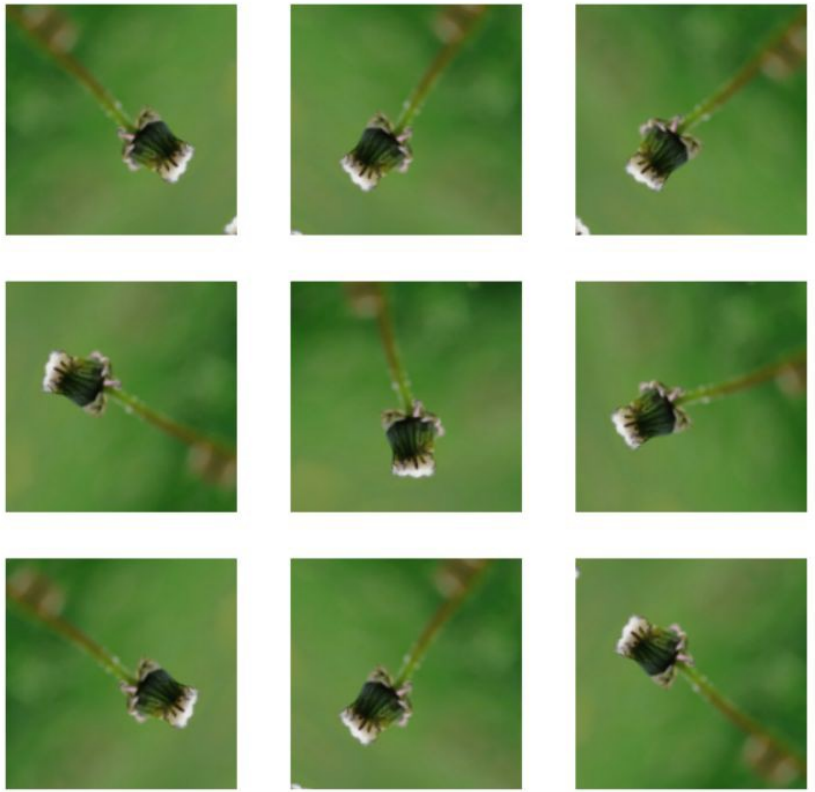
→ *See Interactive Notebook!*

# Data Augmentation

Original image:




Example: randomized rotations



Module 3 – Section 4

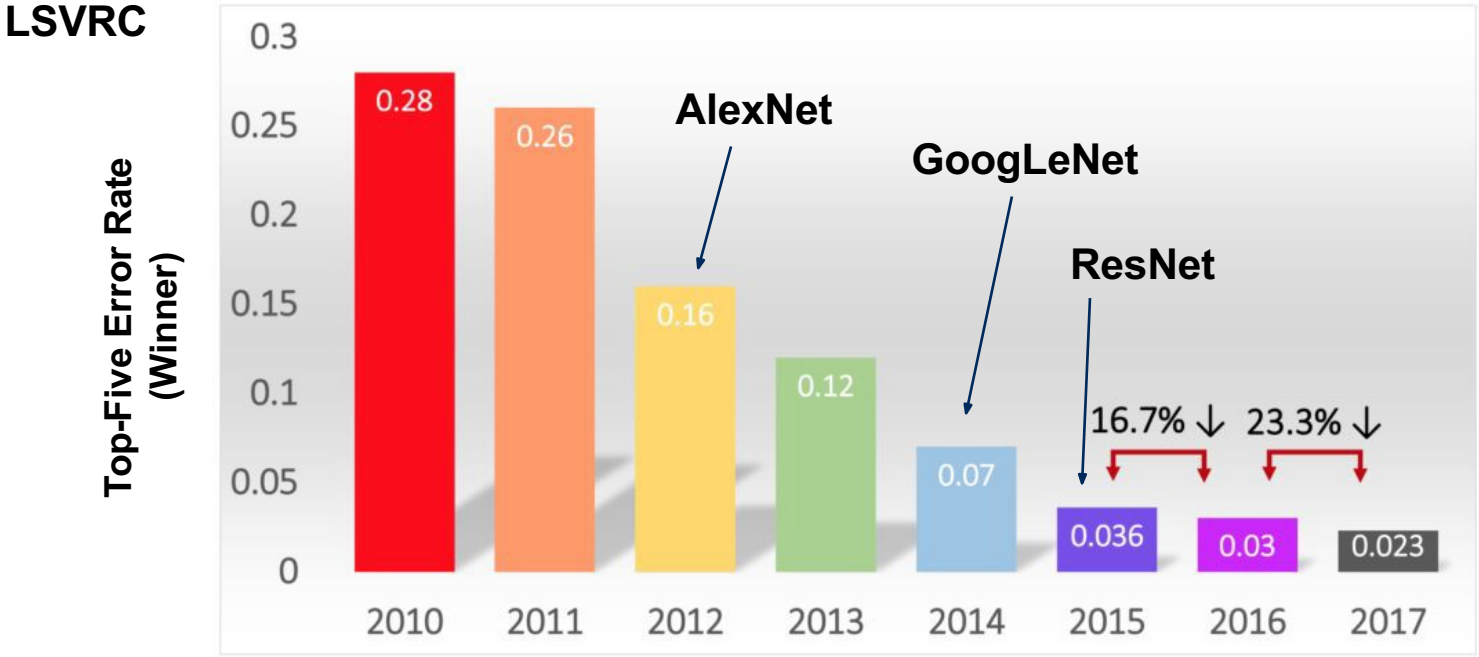
# Milestone CNN Architectures

# LSVRC ImageNet Competition

- Tests both object detection and object localization.
- Exact criteria vary year-to-year.
- Typically 1000 object categories, 150k photographs (1.2M in entire dataset).
- Goal is to achieve the lowest “Top Five Error Rate” (TFER)
    - TFER refers to the percent of cases where the correct class is not among the top-five highest predicted classes.

https://www.image-net.org/challenges/LSVRC/

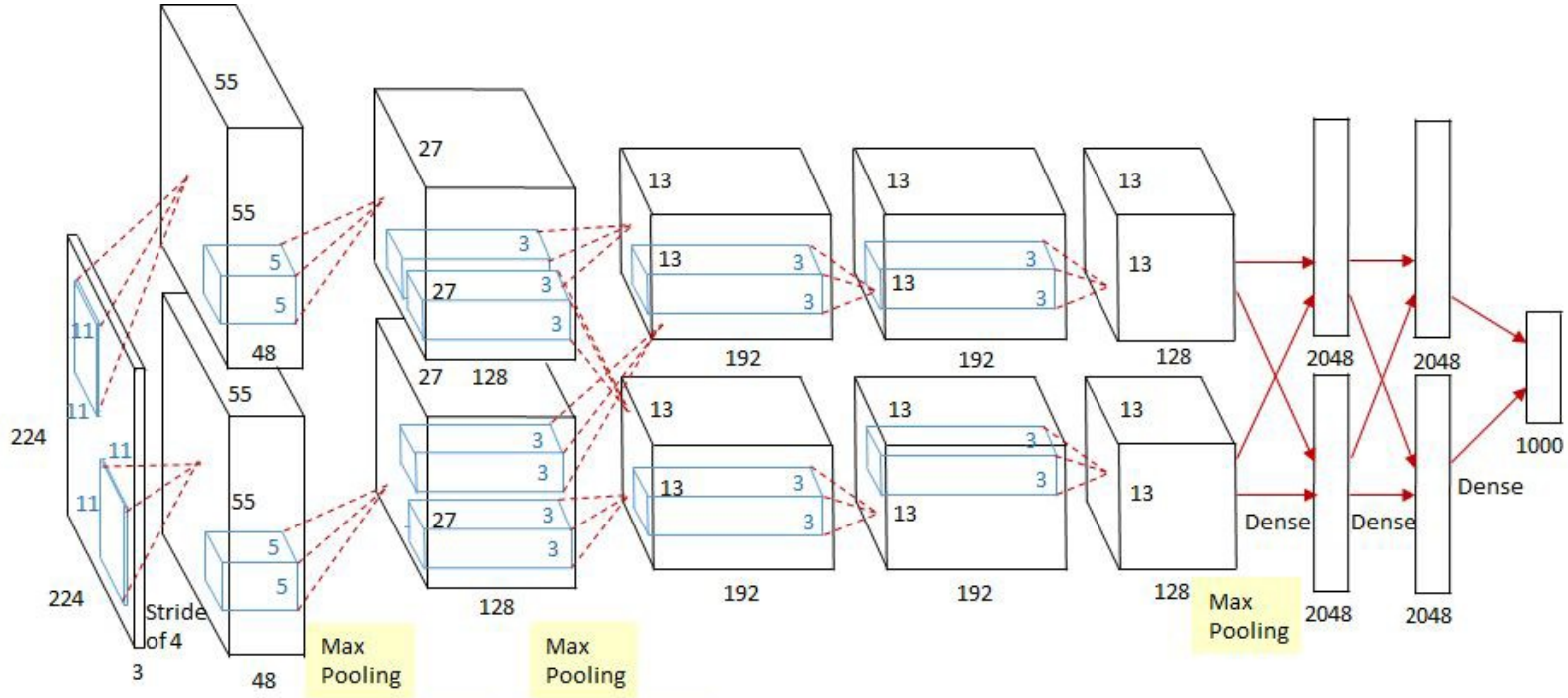
# Evolution of Model Performance




Img Source: https://www.kaggle.com/getting-started/149448

# AlexNet (2012)

16% TFER

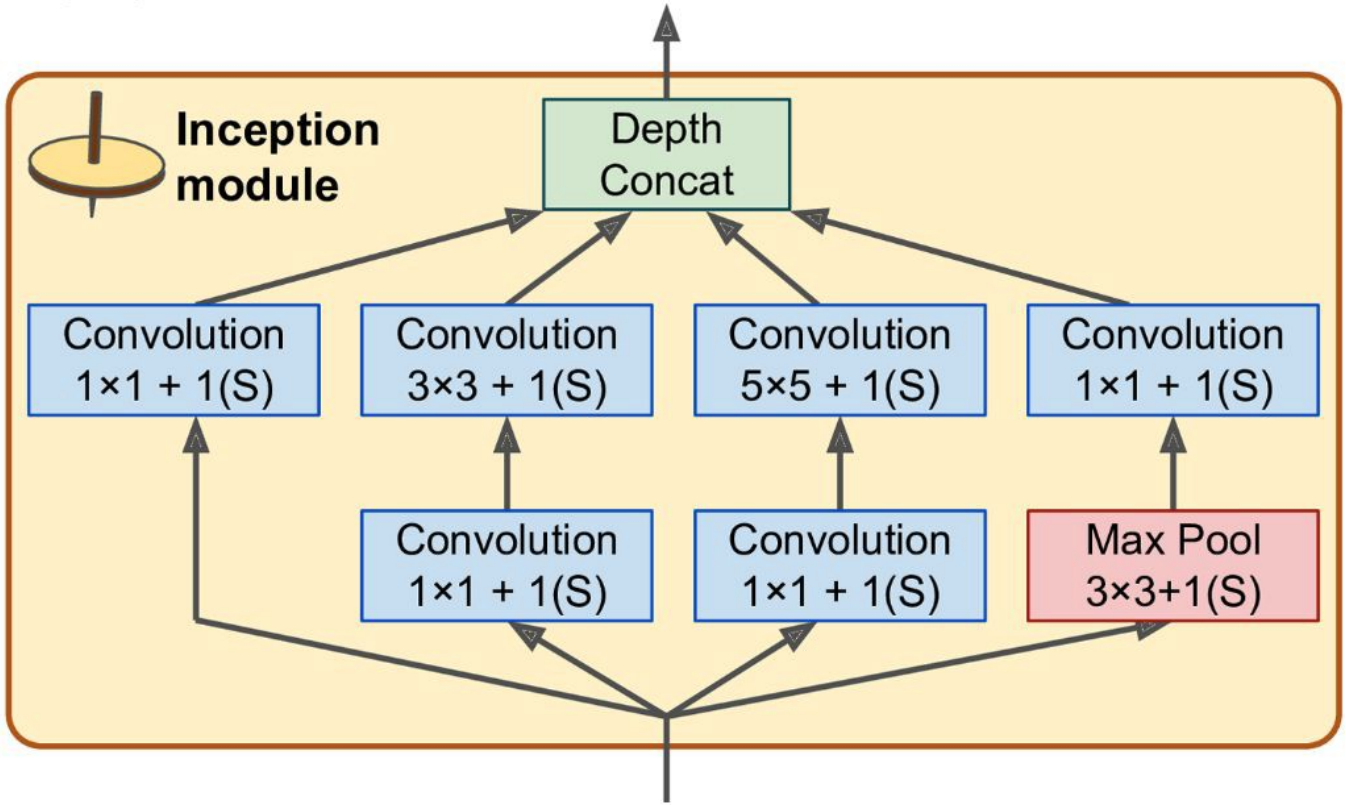



Depth-wise normalization: strongly-activated neurons inhibit neurons in the same position in adjacent channels. Keeps feature maps from being too similar, encouraging specialization.

# GoogLeNet (2014)

7% TFER

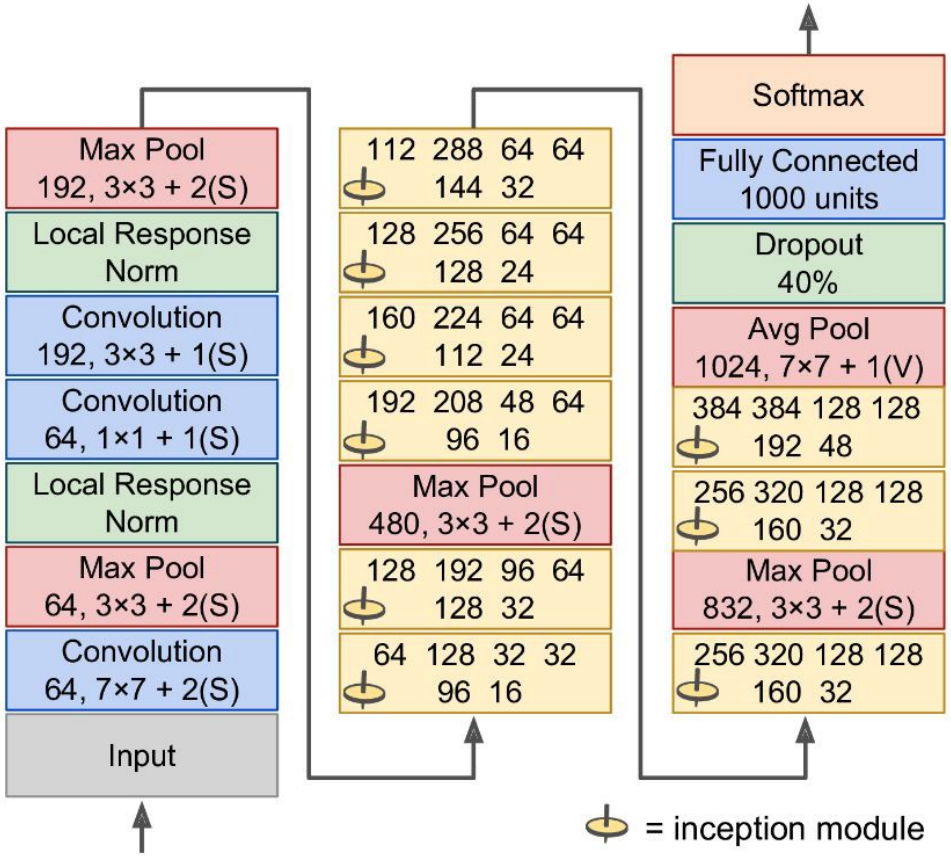
- Deeper network comprised of sub-networks.
- Different kernel sizes capture different scales of features.
- 1x1 conv layers reduce the depth dimensionality
- 1x1 conv layer + regular conv layer sweeps a two-layer NN across their inputs = more complex feature representations.



# GoogLeNet (2014)

7% TFER

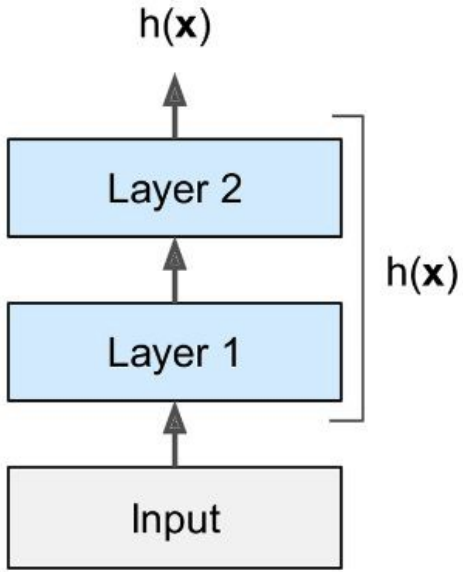
- First few layers perform dimensionality reduction.
- Uses Local Response Norm from AlexNet.
- Then several inception layers mixed with max pool layers.

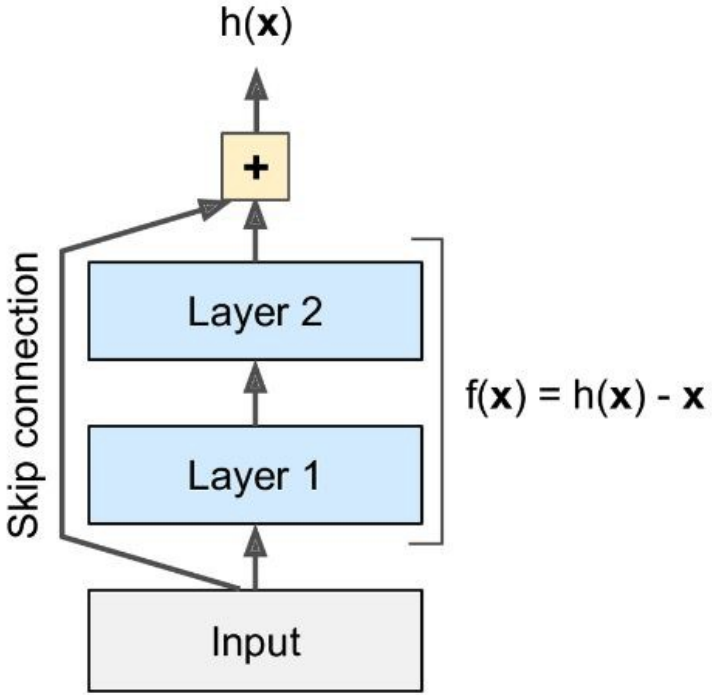


# ResNet (2015)

3.6% TFER

- On initialization, layer outputs are typically zero by default.
- Skip-connections means each layer's output now defaults to the input (identity transform).
- Deeper layers can start training immediately without waiting for input to flow from shallower layers.

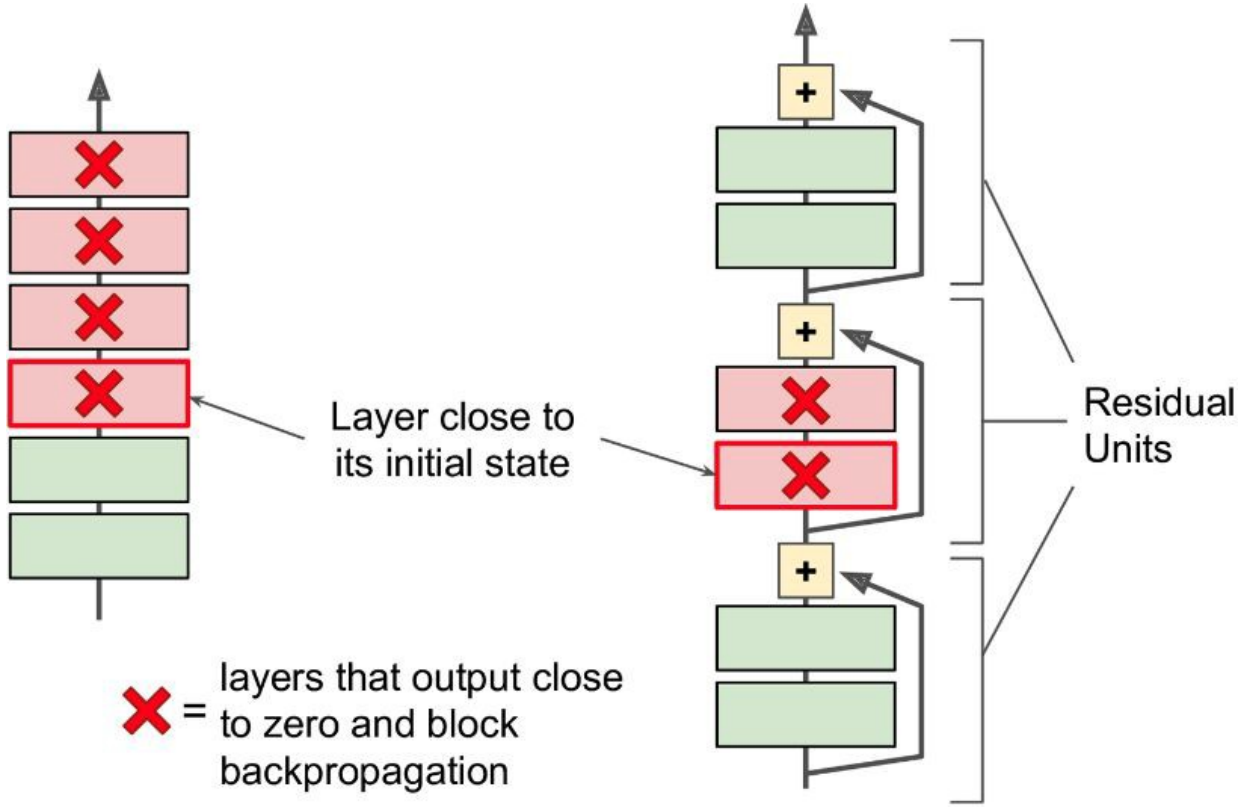


# ResNet (2015)

3.6% TFER

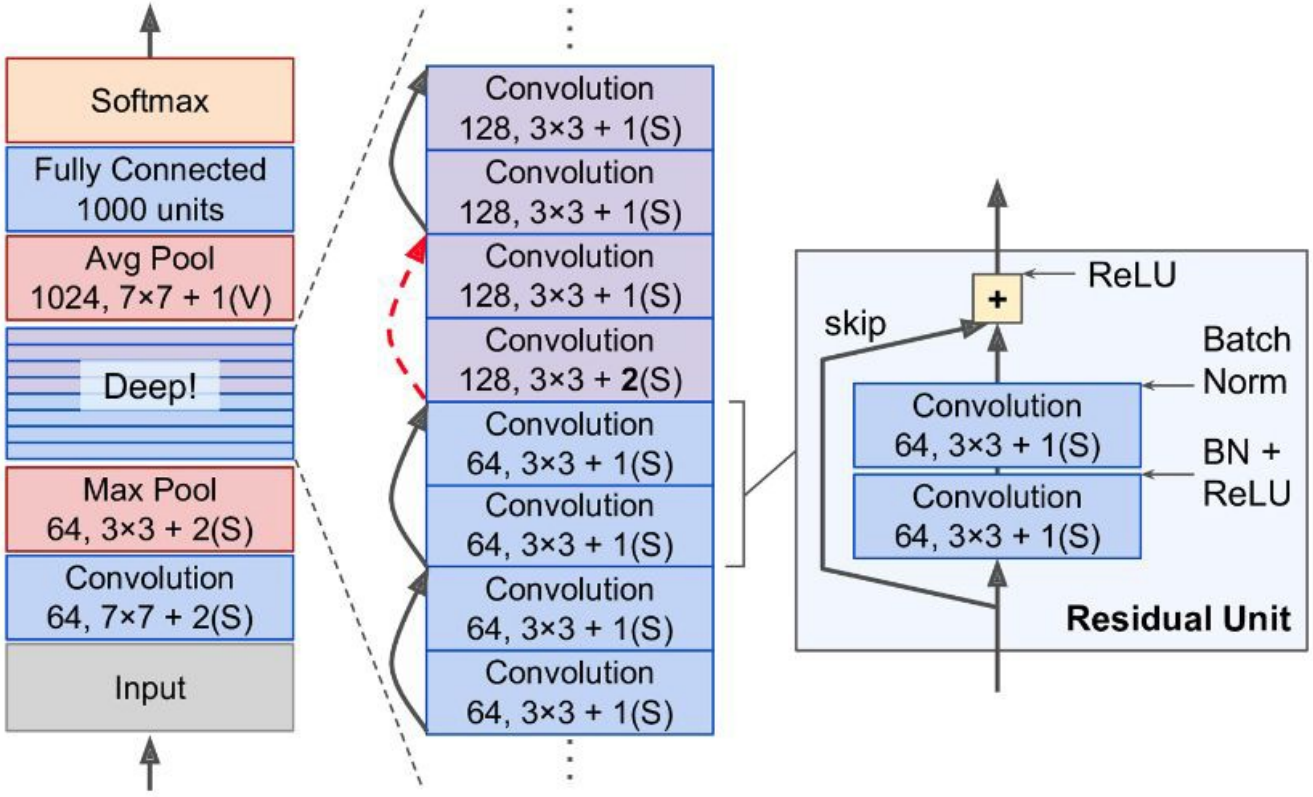
- Skip connections prevent slowly-training layers from blocking the flow of gradients to the rest.
- Helps mitigate vanishing gradients.
- Makes it easier to train deeper networks.



# ResNet (2015)

3.6% TFER

- Typical ResNet architectures can be thousands of layers deep.
- Criticism: not all these layers are actually useful. Sometimes you end up with ‘dead’ layers that never trained. So ‘effective depth’ is much smaller.



## Module 3 – Section 5

# Transfer Learning

→ *See Interactive Notebook!*

## Module 3 – Section 6

# Resources and Wrap-up

# Homework

- Complete Assignment 1 (due by start of Module 4 webinar).
- Review Module 3 notebook.
- Read Géron Chapter 14 (Deep Computer Vision).

# Next Class

- We'll be covering further applications of CNNs for computer vision: segmentation, object detection, etc.
- A preview of the jupyter notebook will be made available.

# Any questions?

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

# Thank You

Thank you for choosing the University of Toronto School of Continuing Studies