Object detection using torchvision

Greg Teichert
CSCAR Consultant

Outline

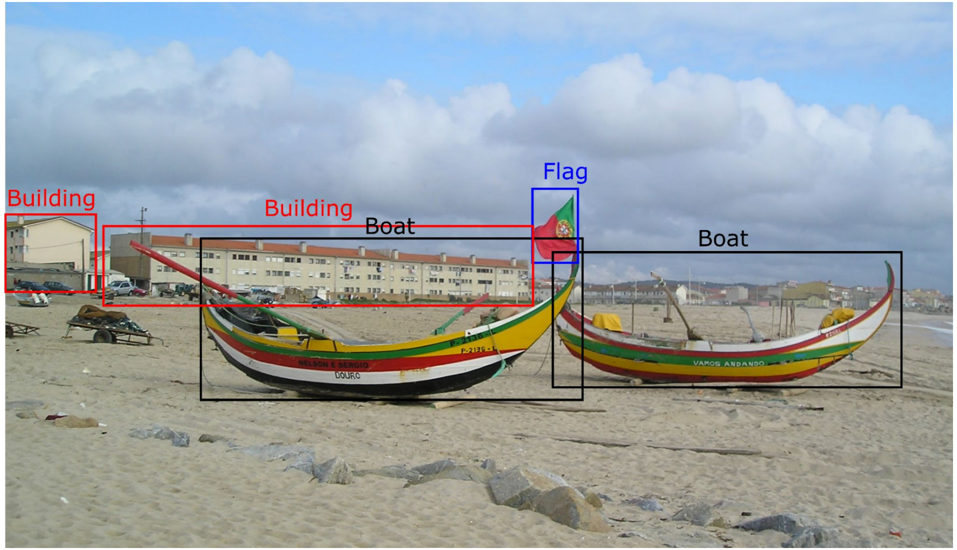
- Object detection
- CNNs
- Transfer learning
- Object detection models (two-stage vs. single stage)
- PyTorch, torchvision
- Object labeling (bounding boxes)
  - Matlab
  - LabelImg (open-source: github.com/tzutalin/labelImg)
- torchvision code example

Object Detection

Two parts:

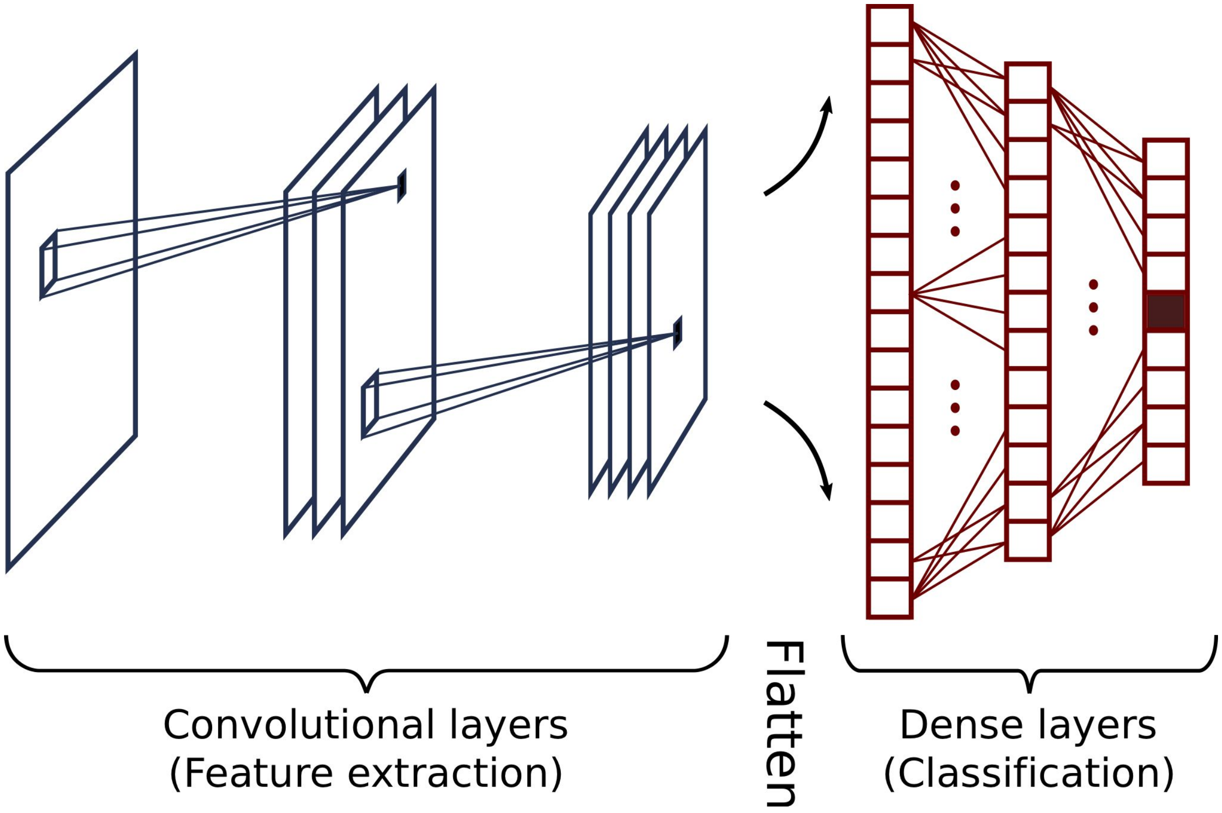
1. Box (potentially important) objects in an image.
2. Classify the objects in the boxes (throw out those that aren't of interest; i.e. don't match a category).

Some detection networks perform both in a single step; some do them in two steps.



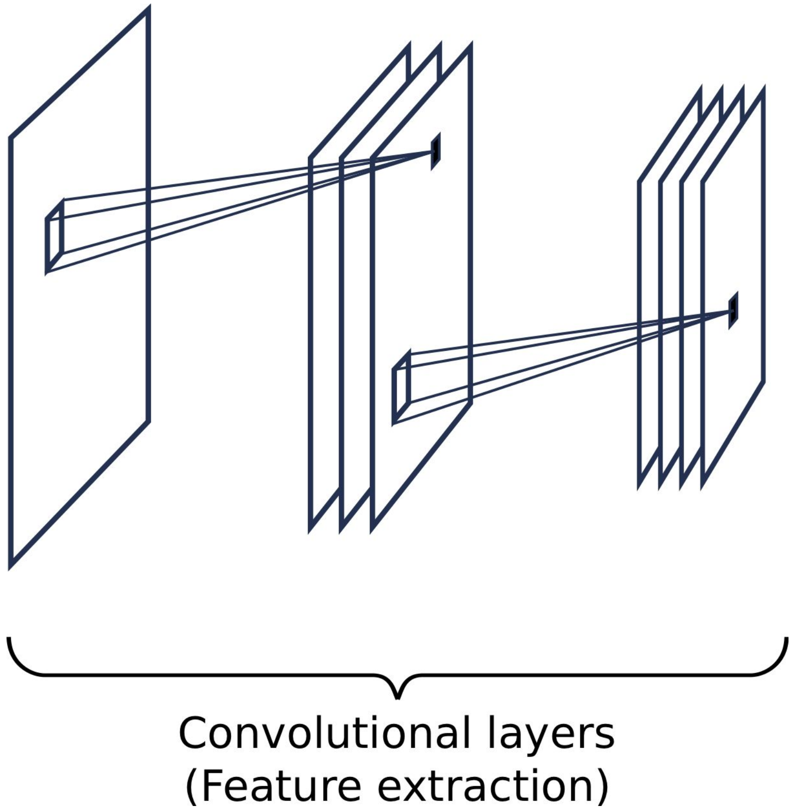
Convolutional Neural Network (CNN)

First,
a review of
CNNs for
image
classification



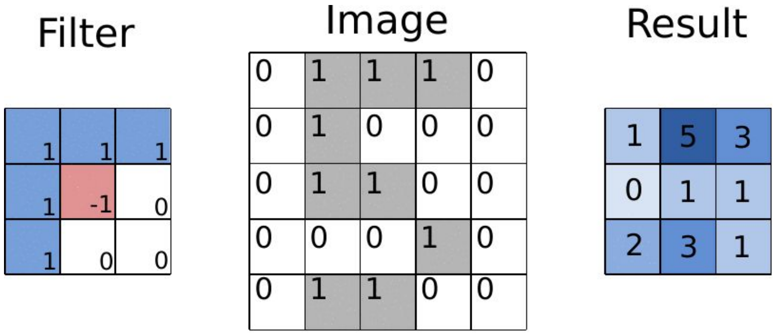
Convolutional Neural Network (CNN)

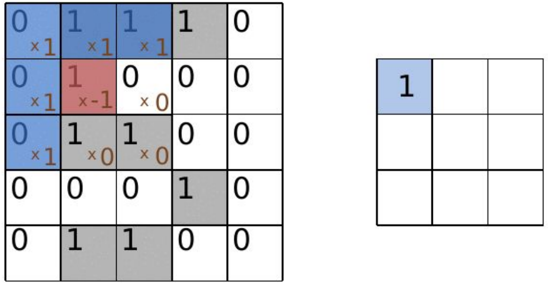
A convolutional neural network is made up (in part) of convolutional layers.



Convolutional Neural Network (CNN)

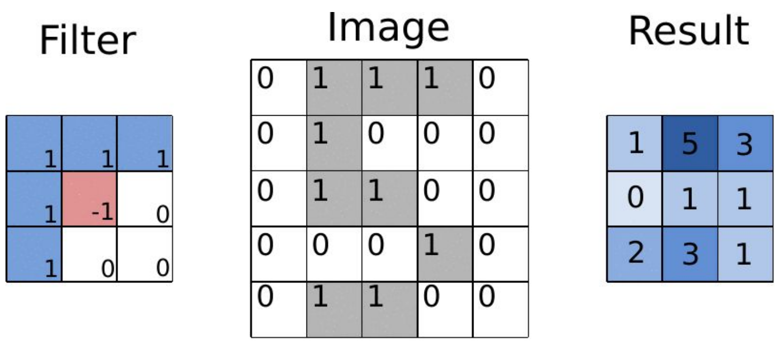
In a convolutional layer, a set of "filters" scan across the input image (or across the output of a previous layer).

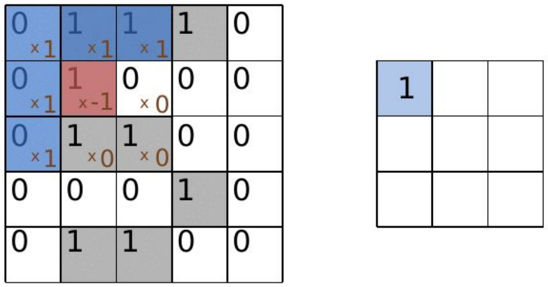




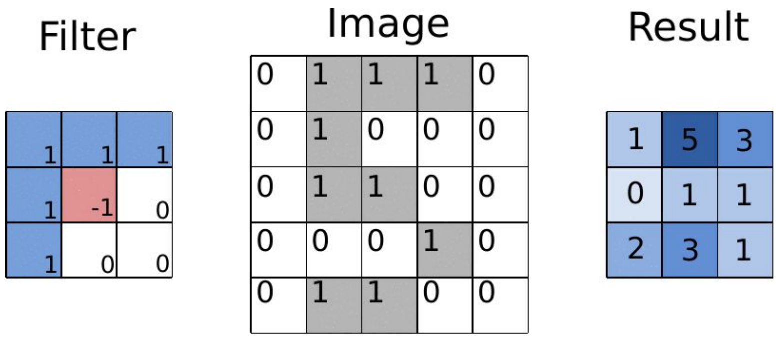
Convolutional Neural Network (CNN)

Filters correspond to visual features in the image (e.g. corners, diagonal lines, etc.)

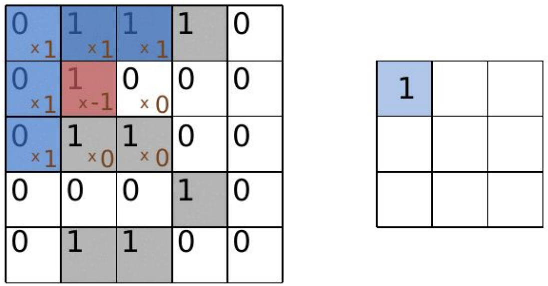




Convolutional Neural Network (CNN)

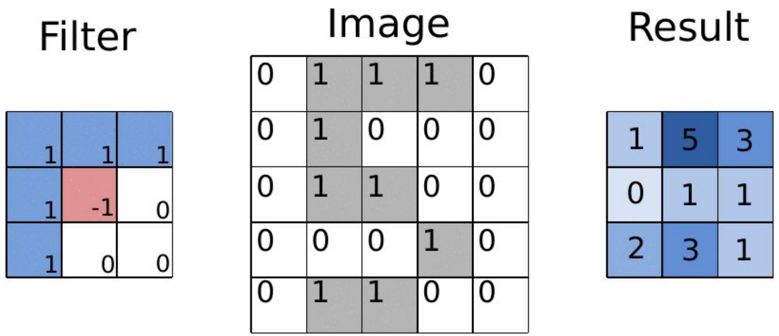


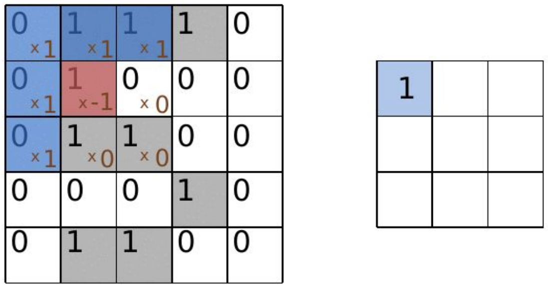
A high value shows a good match between the filter and the image at that location.



Convolutional Neural Network (CNN)

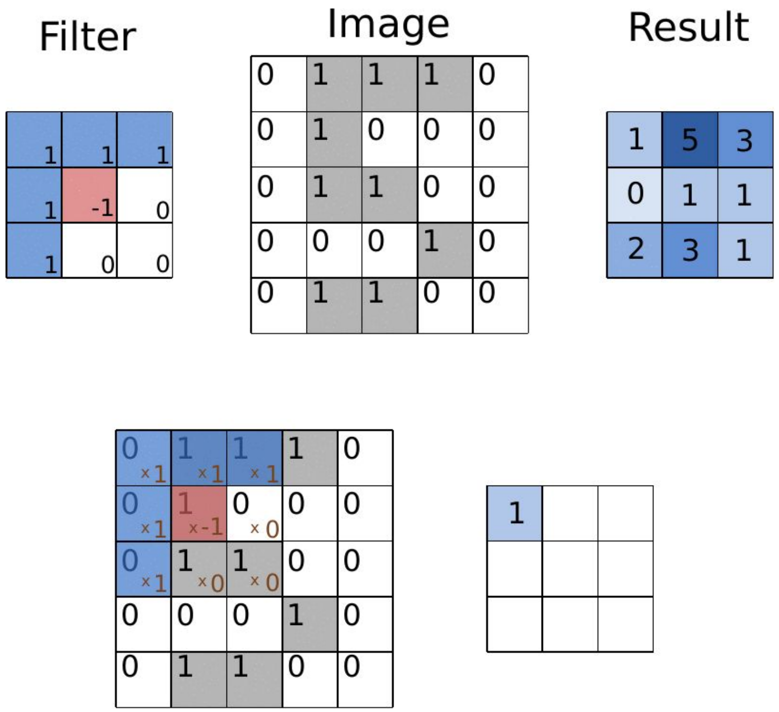
Convolutional layers can be “stacked” to capture more complex data.





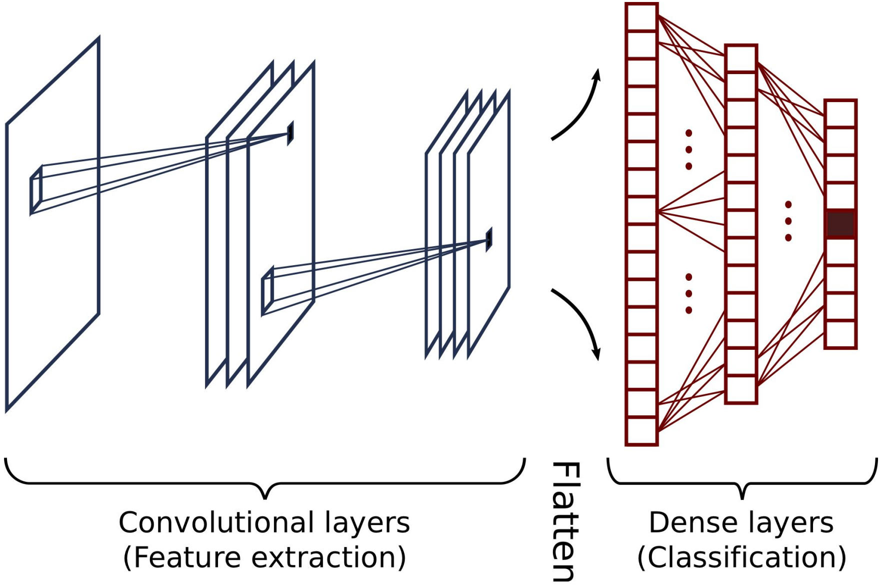
Convolutional Neural Network (CNN)

The filters are usually initialized randomly, then trained to pick out the most relevant features.



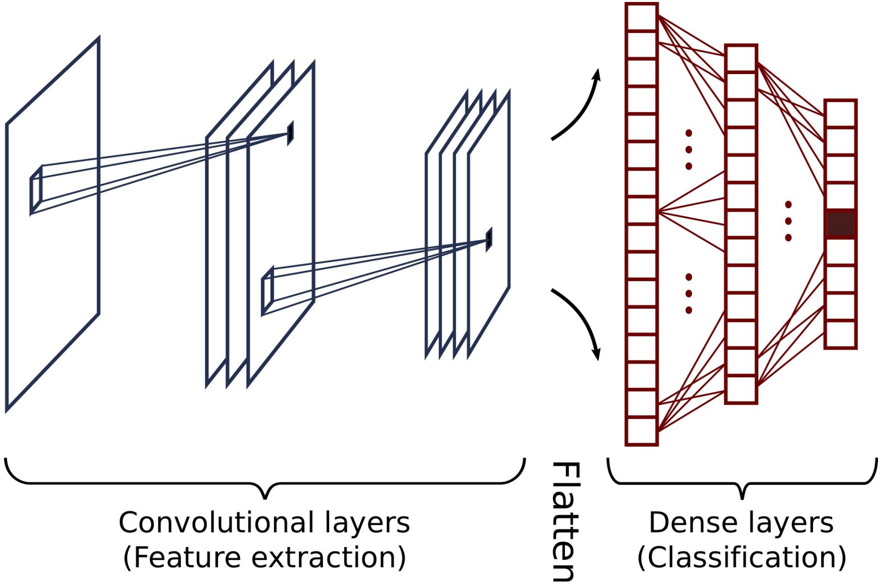
Convolutional Neural Network (CNN)

Those features can be flattened into a single 1D vector, and fed into the dense layers for classification.



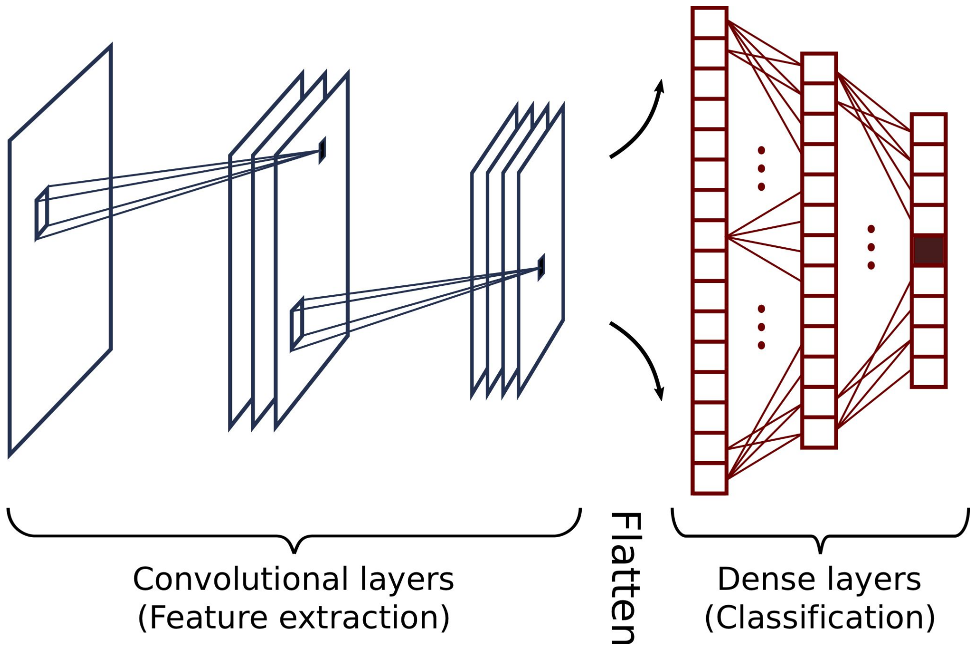
Convolutional Neural Network (CNN)

See our other workshop on Deep Learning to understand more about CNNs:
https://github.com/greght/Workshop-Keras-DNN



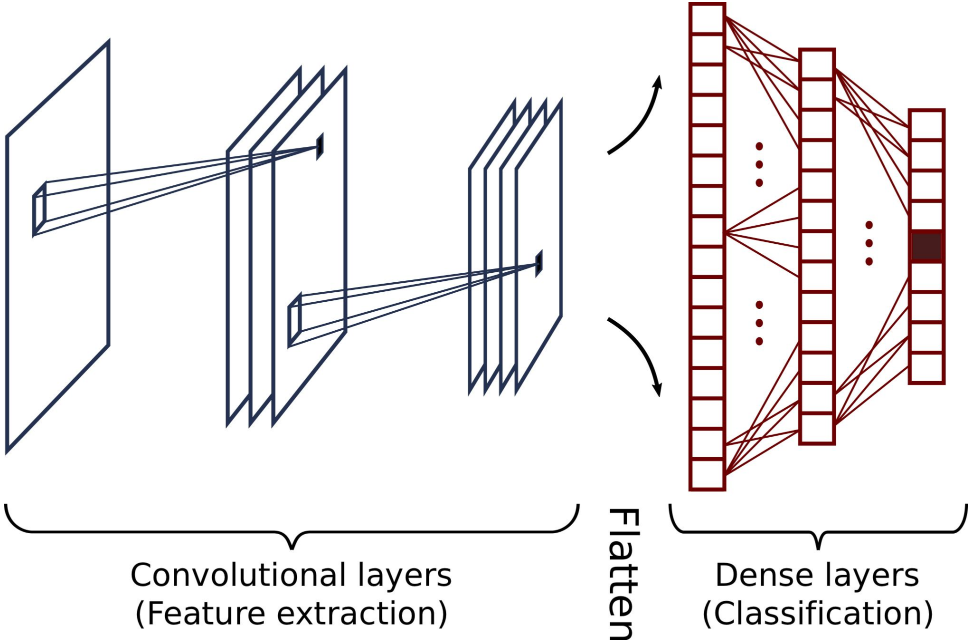
Transfer learning

Transfer learning can speed up the training process by applying “knowledge” learned by a different neural network to a new training set.



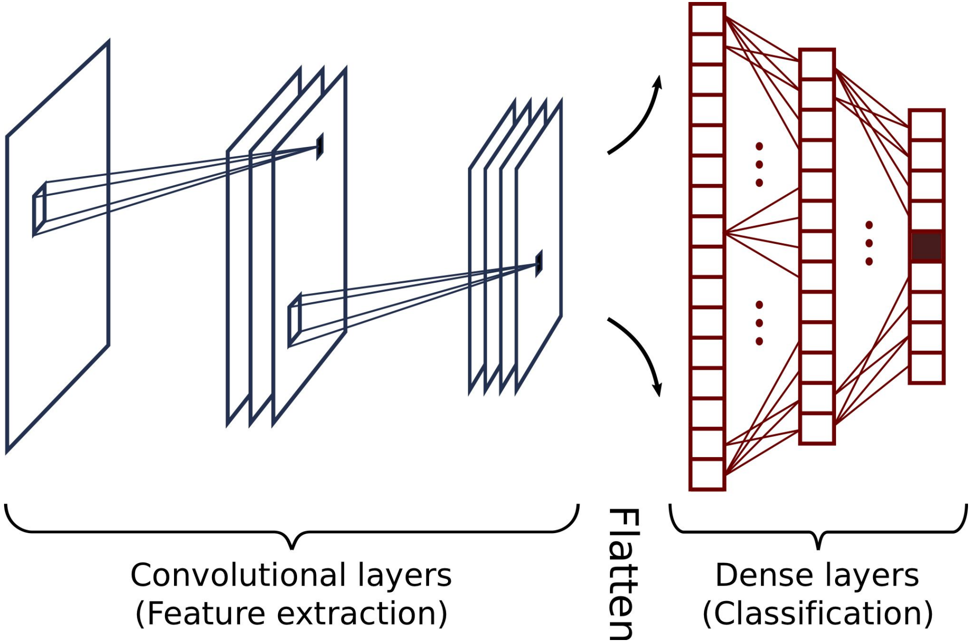
Transfer learning

Take the trained *filters* from a network trained on a different training set (convolutional layers).



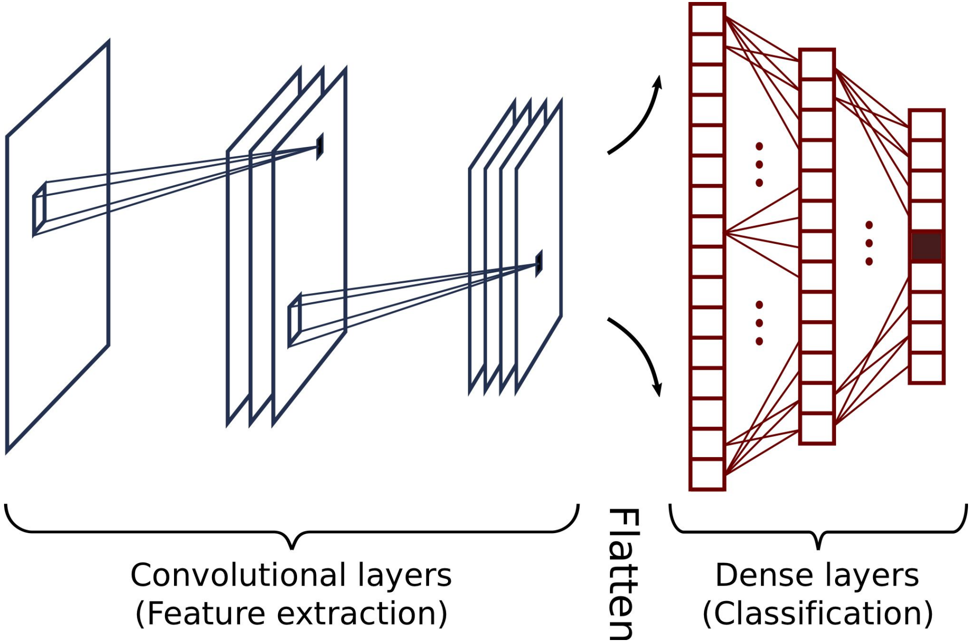
Transfer learning

Freeze the values of the filters while performing training on the weights for the final, classification layers.



Transfer learning

One can then *unfreeze* the filters if fine-tuning is desired. Usually a very small learning rate is used.

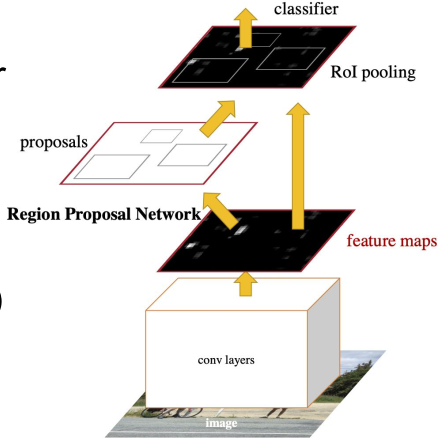


Example object detection models

(Based on this blog post: https://machinelearningmastery.com/object-recognition-with-deep-learning/ and the papers)

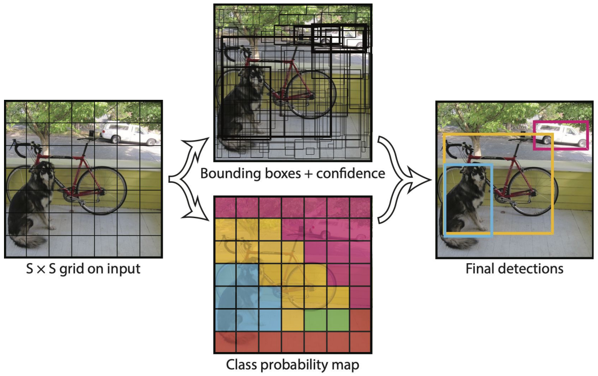
R-CNN family (R-CNN, Fast R-CNN, Faster R-CNN):

- Two parts: Region proposal + classification
- Faster R-CNN models use a convolutional network for the proposed regions and for the classification, combined into a single network (see arxiv.org/abs/1506.01497)



YOLO (You Only Look Once):

- Uses a single neural network to predict bounding boxes *and* classifications (see arxiv.org/abs/1506.02640)
- Generally faster than R-CNN type models but less accurate



PyTorch and torchvision

PyTorch is a widely used and versatile machine learning library developed by Facebook. It is designed to be user-friendly and provide distributed computing.

PyTorch includes the torchaudio, torchtext, torchvision libraries.

Torchvision includes models, functions, and datasets related to computer vision.

Creating training data (boxing/labeling)

There are a range of mask creation tools that can be used.

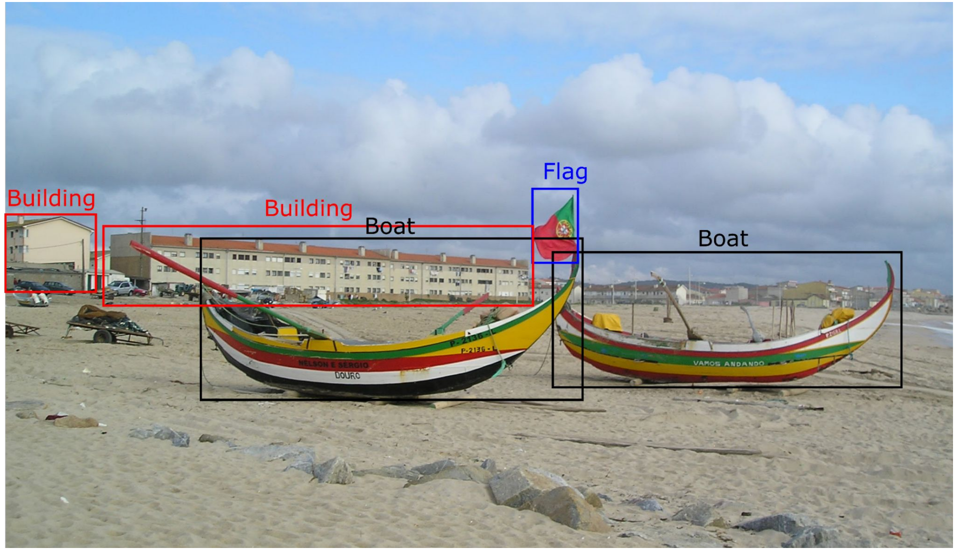
One open source & free tool is here:

https://github.com/tzutalin/labelImg

We will use Matlab.

(Go to https://midesktop.umich.edu/)

Dataset:
https://www.cis.upenn.edu/~jshi/ped_html/PennFudanPed.zip



Torchvision code example

- Fill-in-the-blank notebook: https://colab.research.google.com/github/greght/Workshop-Torchvision-Object-Detection/blob/main/torchvision_object_detection_fill_in_the_blank.ipynb

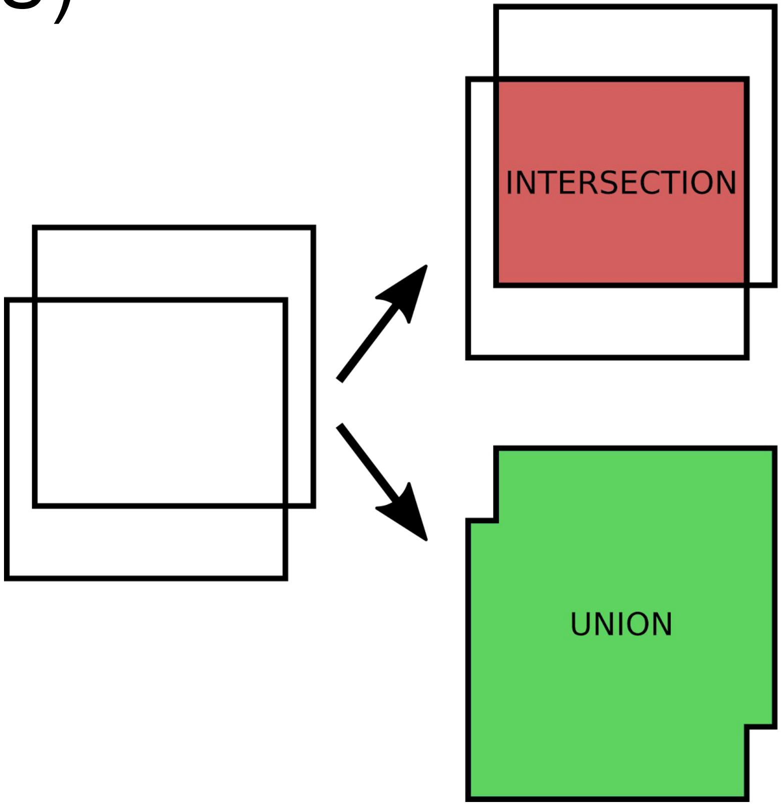
- Complete notebook: https://colab.research.google.com/github/greght/Workshop-Torchvision-Object-Detection/blob/main/torchvision_object_detection.ipynb

Intersection-over-Union (IOU)

IOU is a measure of how well two rectangles (detections, in this case) match up.

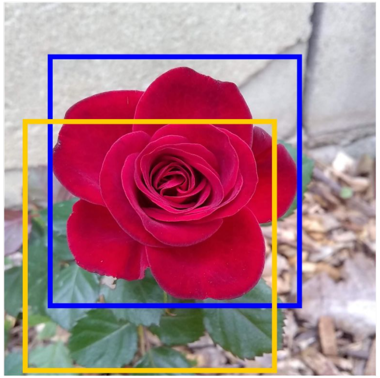
$$\mathrm{IOU} = \frac{\mathrm{INTERSECTION}}{\mathrm{UNION}}$$

Values range from 0 (no overlap) to 1 (perfect match)

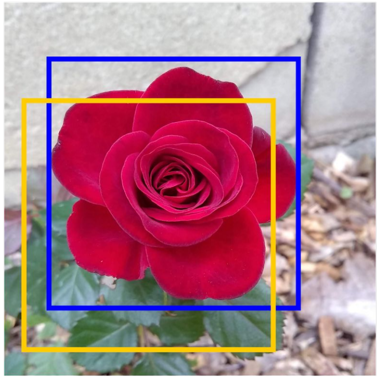


Intersection-over-Union (IOU)

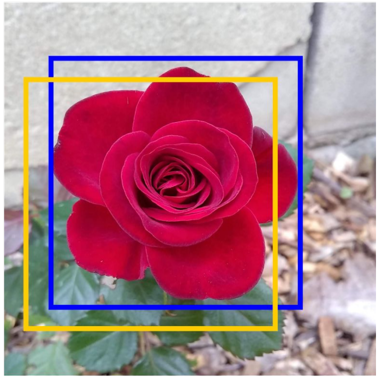
IOU = 0.5

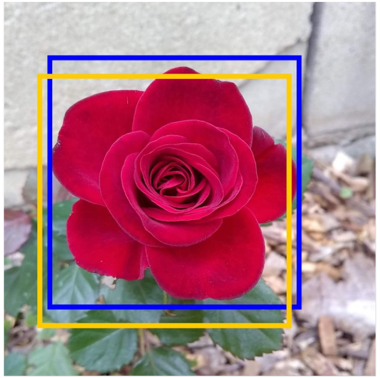


IOU = 0.6

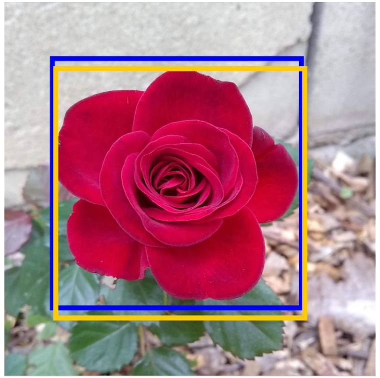


IOU = 0.7





IOU = 0.8



IOU = 0.9