Computer vision:
deep-learning image recognition

Materials for this section:

- stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks
- https://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns

Deep-learning: automatic feature extraction

https://github.com/rasbt/stat453-deep-learning-ss20

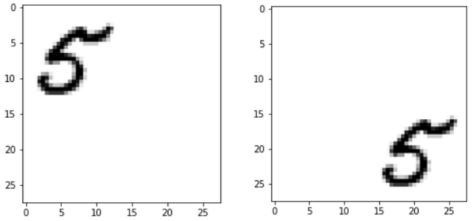
Why is image classification so hard?

Different lighting, contrast, viewpoints, etc.

Image Source: twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

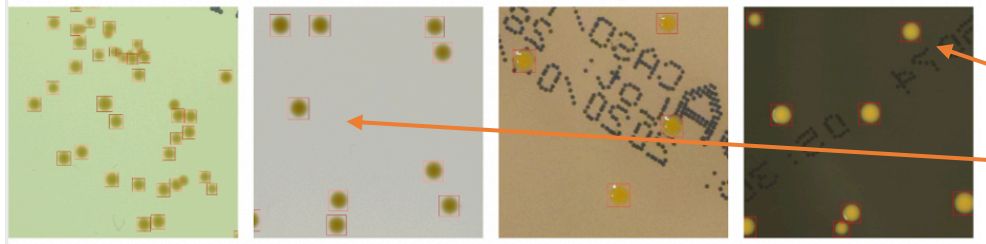
Or even simple translation



- Extracted feature vector should be **independent** from specific image conditions/setups

= days or weeks to properly tune parameters (really! ;)

Feature Engineering
Manual Extraction+Selection



Same objects in very different lightning conditions

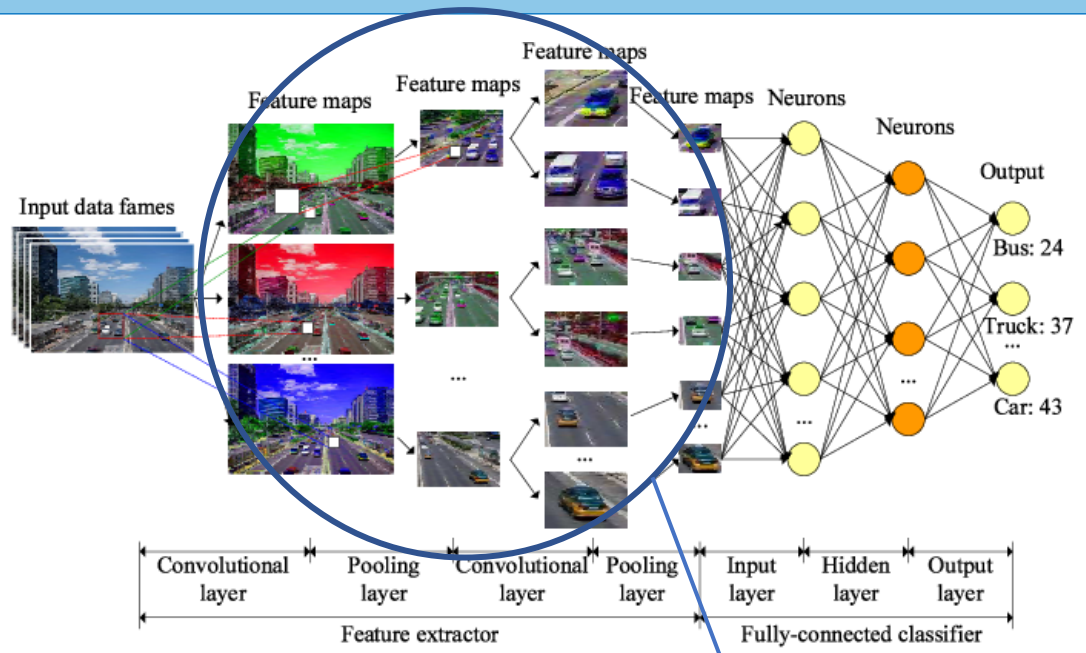


Fig. 3. Example of the CNN model for vehicle classification.

Deep learning do the same **automatically**!!!

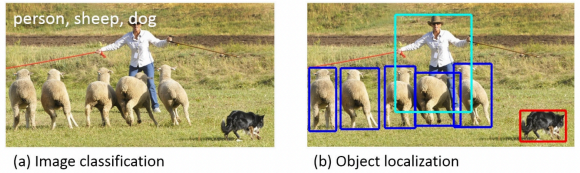
- Price: we need huge datasets (10-100 × larger)
- In CNNs we naturally loose irrelevant information (by applying properly optimized convolutions)

Deep-learning image recognition: datasets

Popular datasets to train models towards detection/segmentation/classification

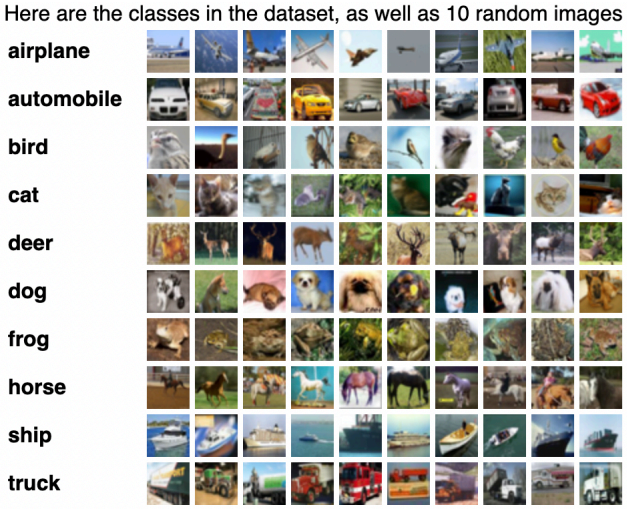
- ImageNet
- CIFAR-10 (much simpler)
- COCO (object localization + segmentation masks)
- Pascal VOC
- CityScapes

...



COCO: 330K images, 1.5 million object instances:

- segmentation mask for each **instance**
- instances may overlap



CIFAR:
10 classes only

CityScapes: dense (panoptic) instance segmentations

Object detection

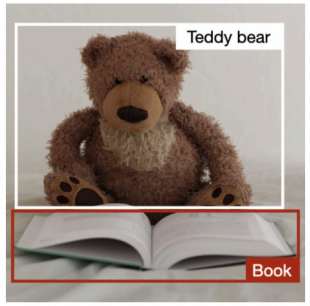
stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Image classification

- Classifies a picture
- Predicts probability of object

Traditional CNN

Detection



- Detects up to several objects in a picture
- Predicts probabilities of objects and where they are located

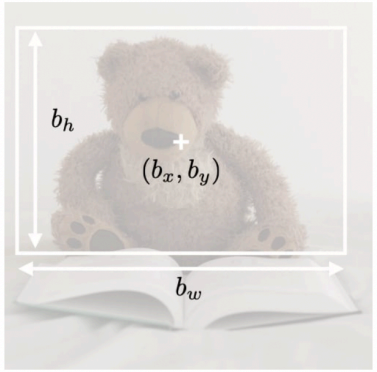
YOLO, R-CNN

Detection:

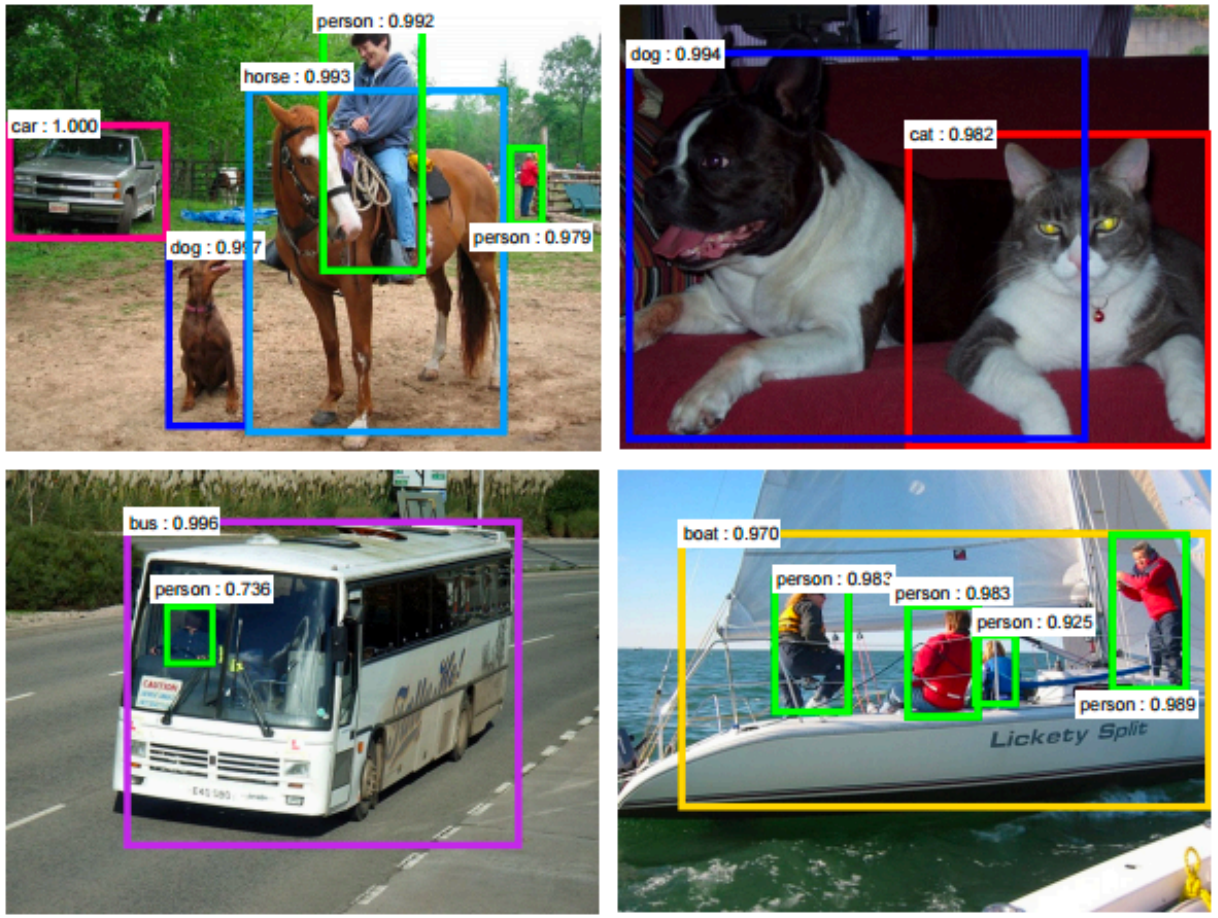
- predicts *probabilities* of objects and where they are located
- mark locations using *bounding boxes*
- assign class for each object

Bounding box detection

- Detects the part of the image where the object is located



Box of center (b_x, b_y), height b_h and width b_w

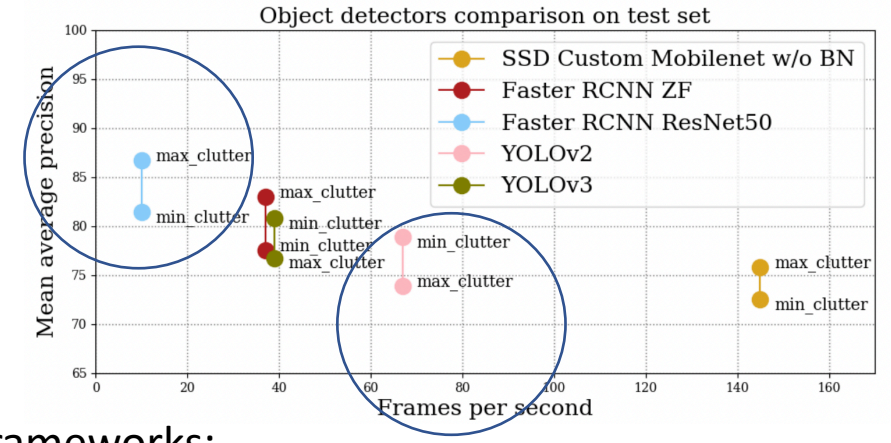


stats.stackexchange.com/questions/298389/probability-scores-threshold-usually-used-in-deep-learning-for-an-object-recogni

Detector networks

Two types of detectors:

- **Single-stage** (faster), e.g.
  - SSD
  - YOLO
  - RetinaNet
- **Two-stage** (more accurate), e.g.
  - R-CNN (Fast, Faster, Cascade…)



Popular frameworks:

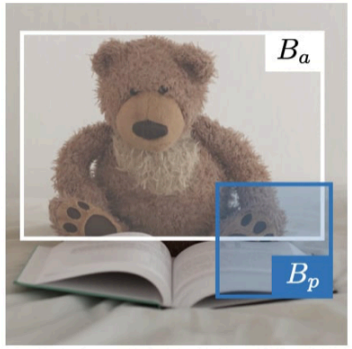
- MMDetection
- Detectron (Facebook)

Fundamental detection metric: IoU

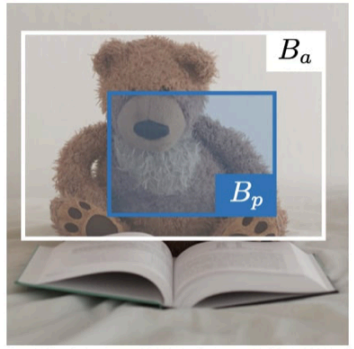
□ **Intersection over Union** — Intersection over Union, also known as IoU, is a function that quantifies how correctly positioned a predicted bounding box B_p is over the actual bounding box B_a. It is defined as:

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

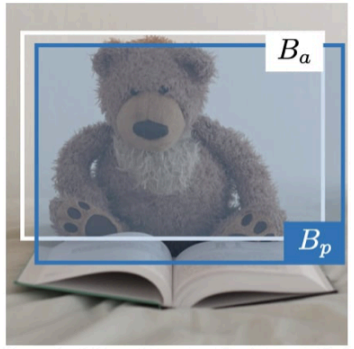
Remark: we always have $\text{IoU} \in [0, 1]$. *By convention, a predicted bounding box* B_p *is considered as being reasonably good if* $\text{IoU}(B_p, B_a) \geqslant 0.5$.



$\text{IoU}(B_p, B_a) = 0.1$



$\text{IoU}(B_p, B_a) = 0.5$



$\text{IoU}(B_p, B_a) = 0.9$

stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Common detectors

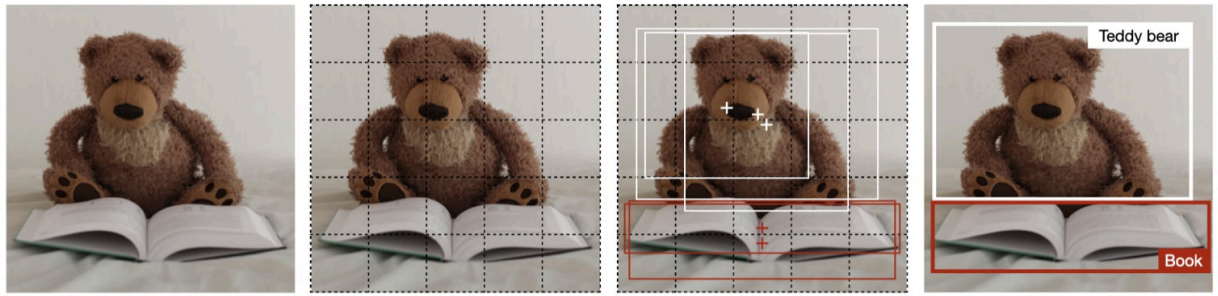
YOLO — You Only Look Once (YOLO) is an object detection algorithm that performs the following steps:

- Step 1: Divide the input image into a $G \times G$ grid.
- Step 2: For each grid cell, run a CNN that predicts y of the following form:

$$y = [\underbrace{p_c, b_x, b_y, b_h, b_w, c_1, c_2, ..., c_p, ...}_{\text{repeated } k \text{ times}}]^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

where p_c is the probability of detecting an object, b_x, b_y, b_h, b_w are the properties of the detected bouding box, $c_1, ..., c_p$ is a one-hot representation of which of the p classes were detected, and k is the number of anchor boxes.

- Step 3: Run the non-max suppression algorithm to remove any potential duplicate overlapping bounding boxes.



stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Each grid cell is responsible for predicting a number of bounding boxes, with H × W that may be different from cell size

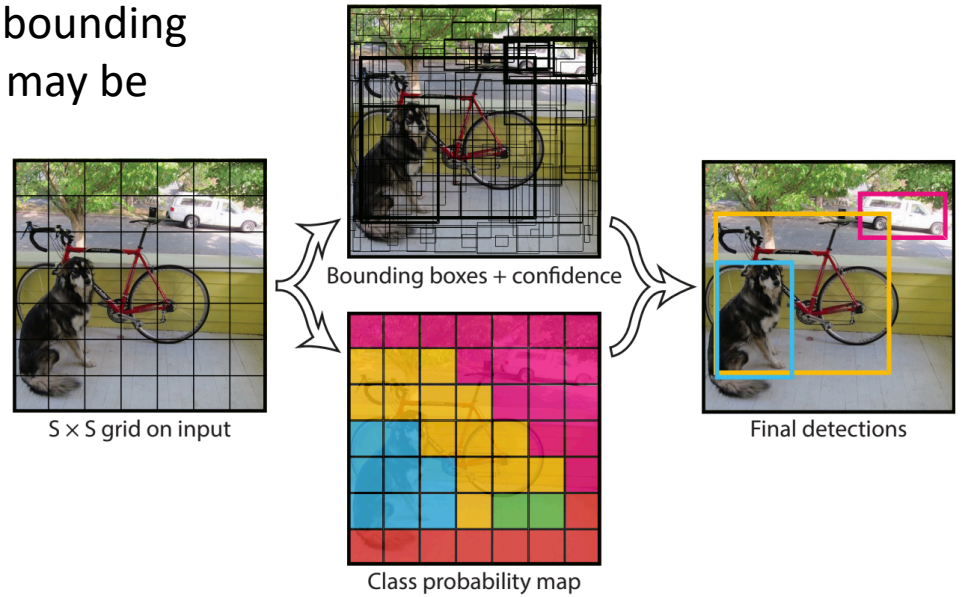


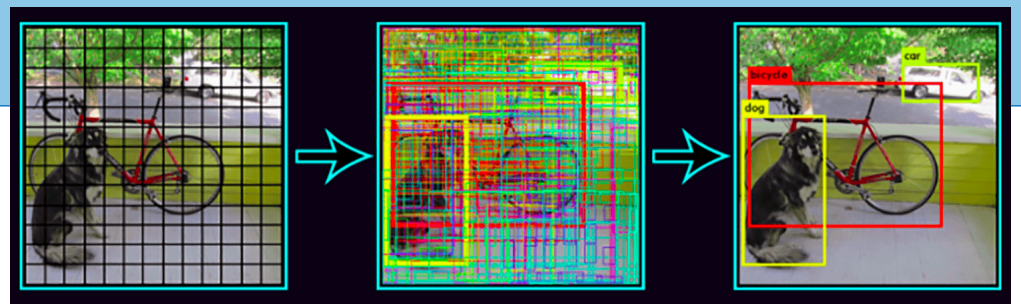
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

- drawback: less accurate, especially for small objects
- albeit YOLOv3/v4 is more precise

https://arxiv.org/abs/1506.02640

https://arxiv.org/abs/1804.02767

Non-Maximum Suppression



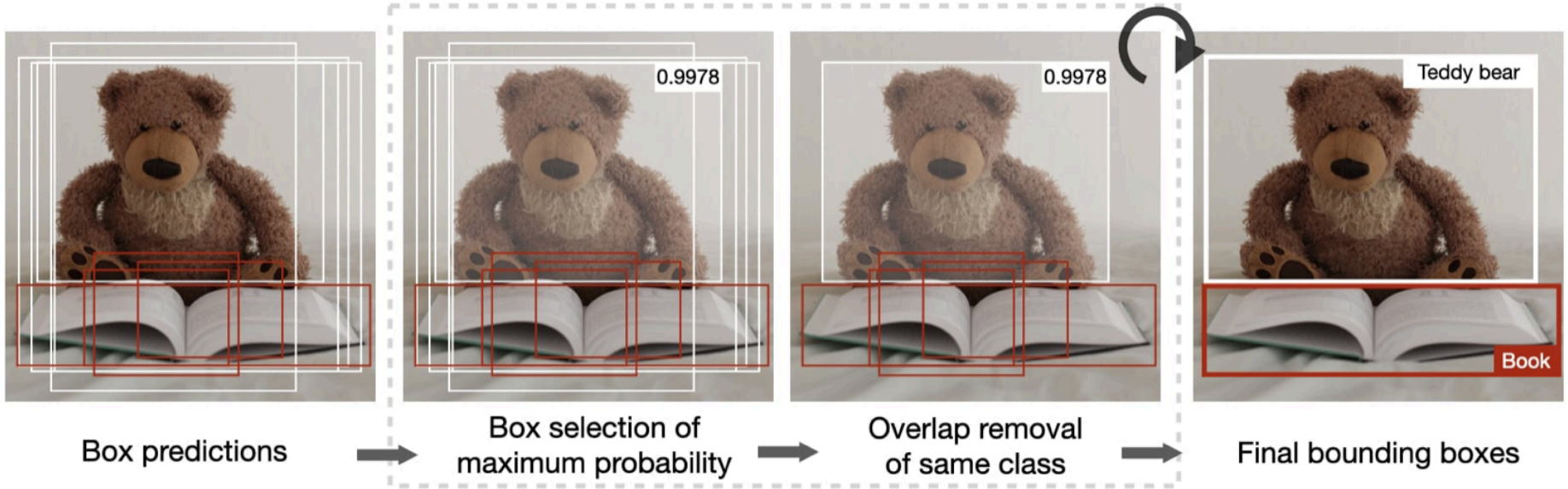
How to get rid of excessive bounding boxes?
-- **NMS**

□ **Non-max suppression** — The non-max suppression technique aims at removing duplicate overlapping bounding boxes of a same object by selecting the most representative ones. After having removed all boxes having a probability prediction lower than 0.6, the following steps are repeated while there are boxes remaining:

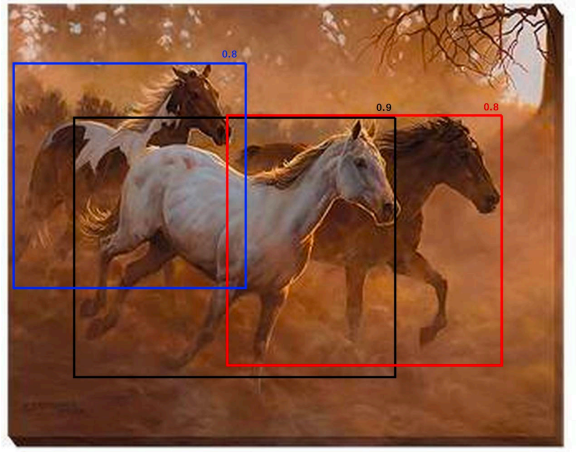
For a given class,

- Step 1: Pick the box with the largest prediction probability.
- Step 2: Discard any box having an $\mathrm{IoU} \geqslant 0.5$ with the previous box.

NMS threshold



What if objects overlap
-- **soft NMS**



https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

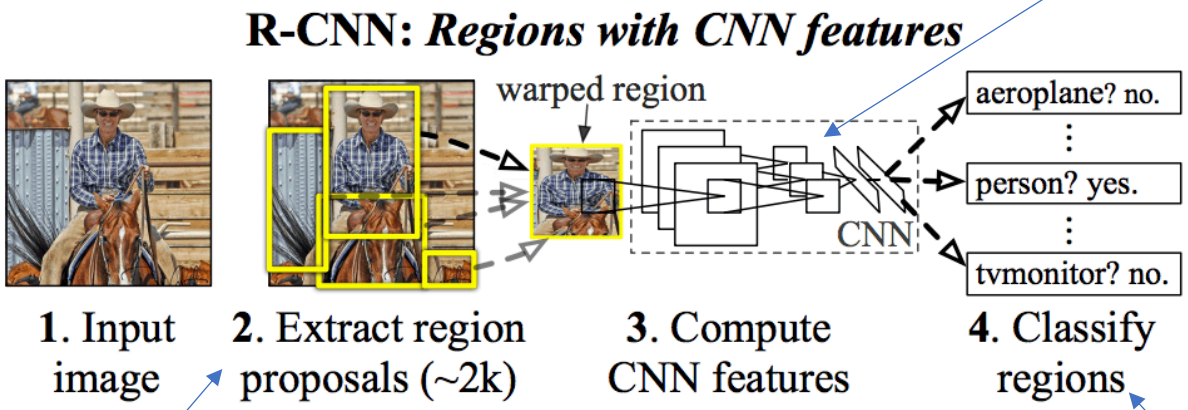
stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#

Common detectors

Region-based convolutional neural networks: **R-CNN**s

Original R-CNN architecture

https://arxiv.org/abs/1311.2524



feature extractor, e.g. AlexNet

generate region proposals using traditional methods, e.g. segmentation

Classify regions with SVM basing on vector of features

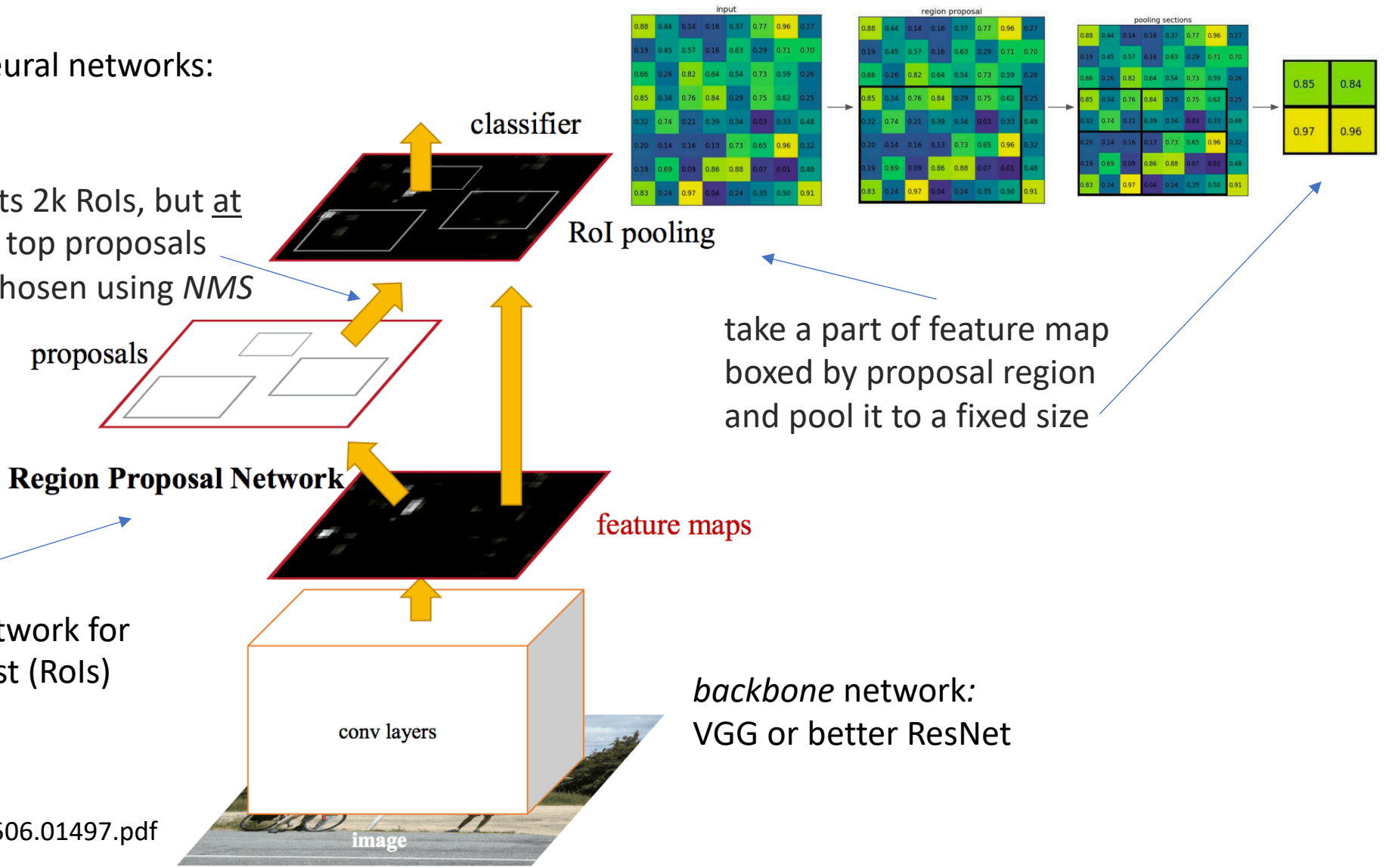
Drawbacks:

- needs to prepare 3 separate modules
- training/evaluating deep CNN on each of proposal regions separately makes it very slow

Common detectors

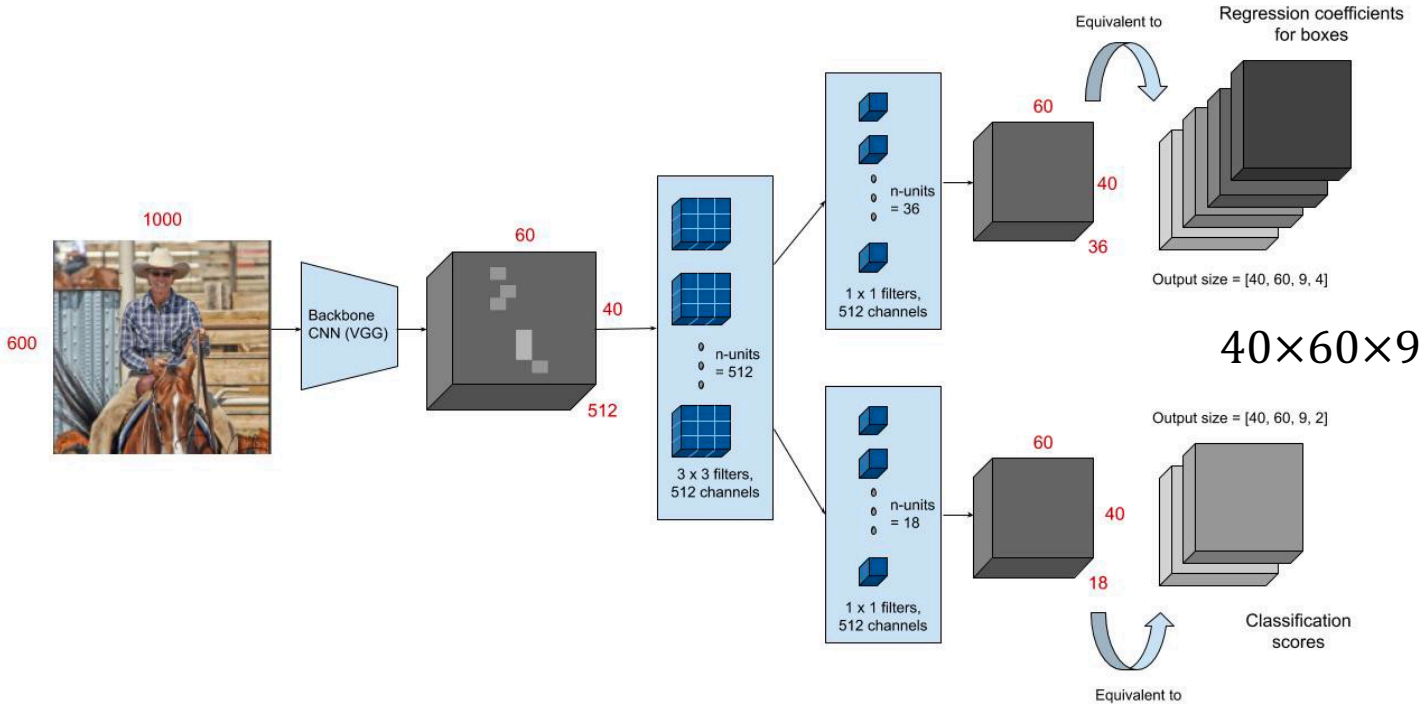
Region proposal convolutional neural networks:
Current approach: **Faster R-CNN**

- end-to-end approach
- much faster than original R-CNN



https://arxiv.org/pdf/1506.01497.pdf

Region Proposal Network



towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

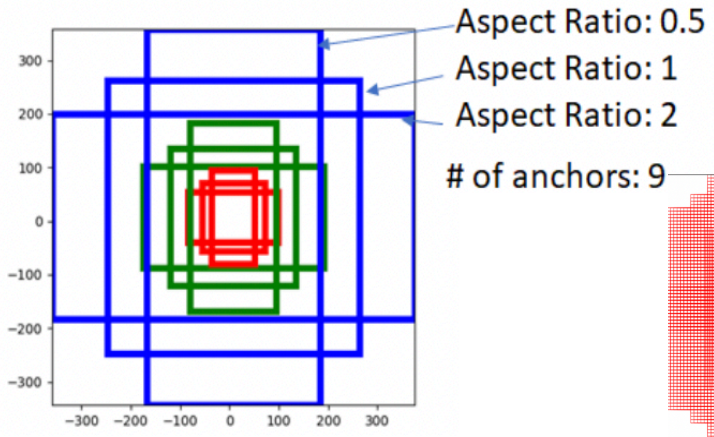
$40 \times 60 \times 9 = 20\text{k}$ anchors → NMS → 2k refined anchors

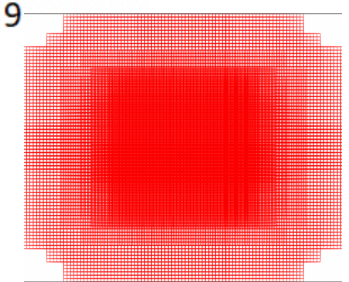
The goal of the RPN is to select promising *anchors*

- to distinguish interesting regions on image (RoIs)
- then generate good proposal bounding boxes: (by improving position and size of anchors)

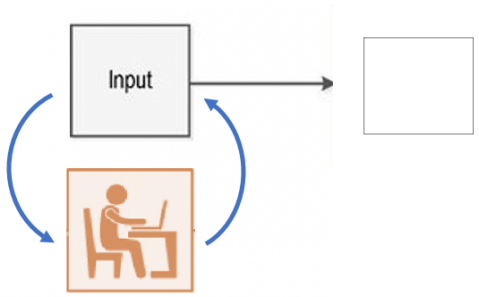
- anchors – proposal bounding boxes placed at every point of feature map:

- Set of aspect ratios (0.5, 1, 2)
- Stride length (downscaling performed by resnet head: 16)
- Anchor Scales (8, 16, 32)



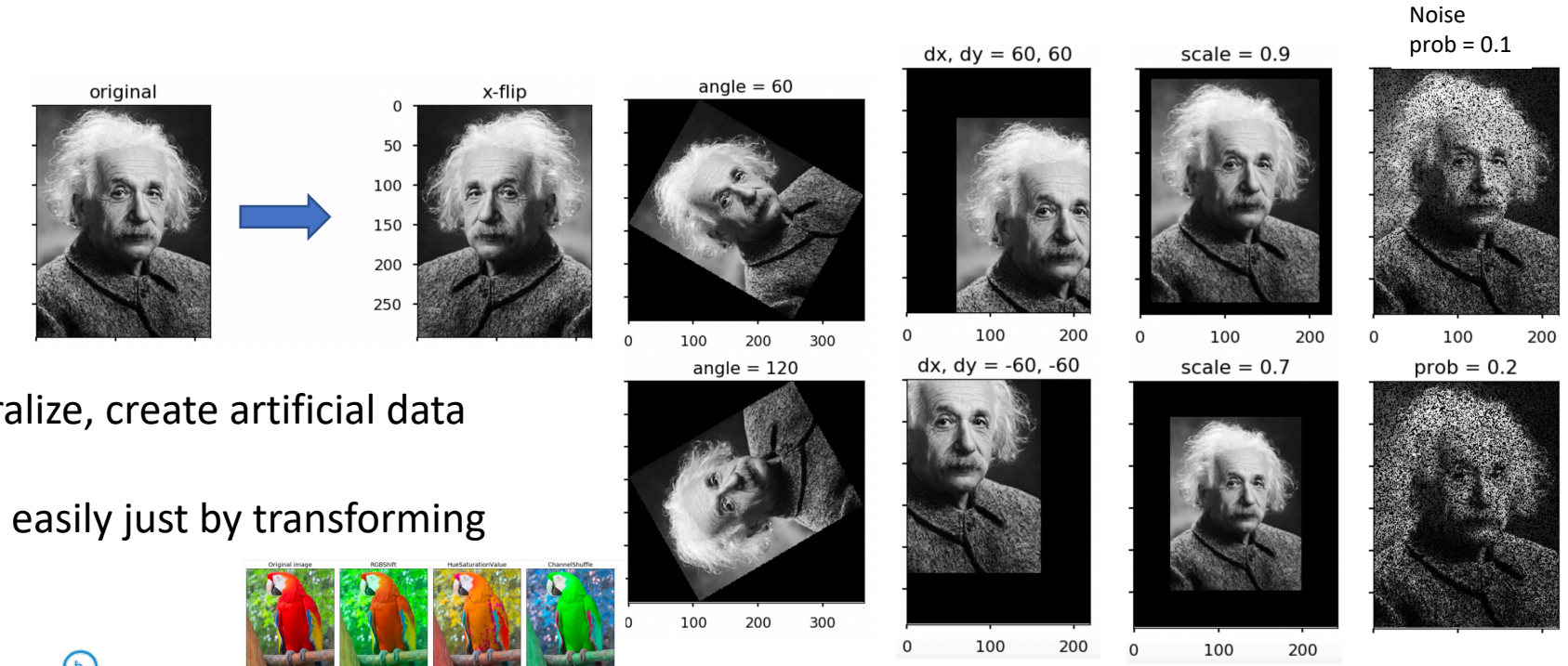


Dataset augmentation



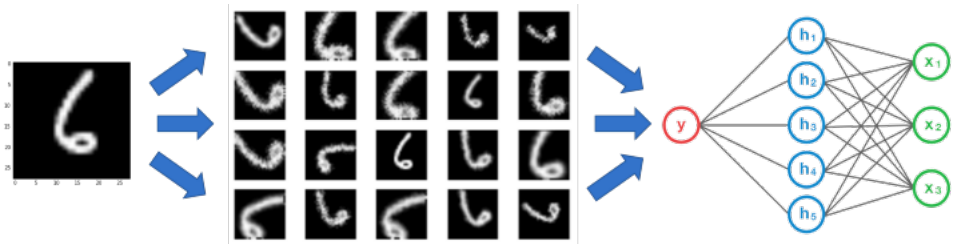
- **dataset augmentation**

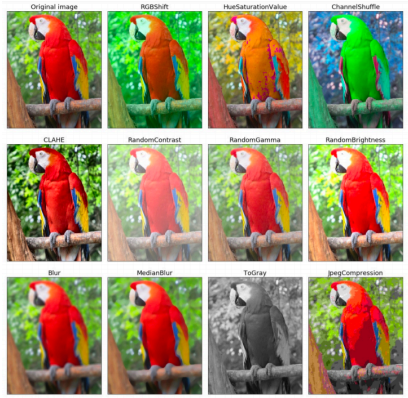
Dataset augmentation is a particularly effective technique for a specific classification problem: object recognition



Idea:

- to improve model ability to generalize, create artificial data and add it to the training set
- we can generate new (X, y) pairs easily just by transforming the X inputs in our training set.





nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

- simple geometrical operations on training images can often greatly improve generalization
- also adding some distractors or noise may help

tomaszgolan.github.io/introduction_to_machine_learning