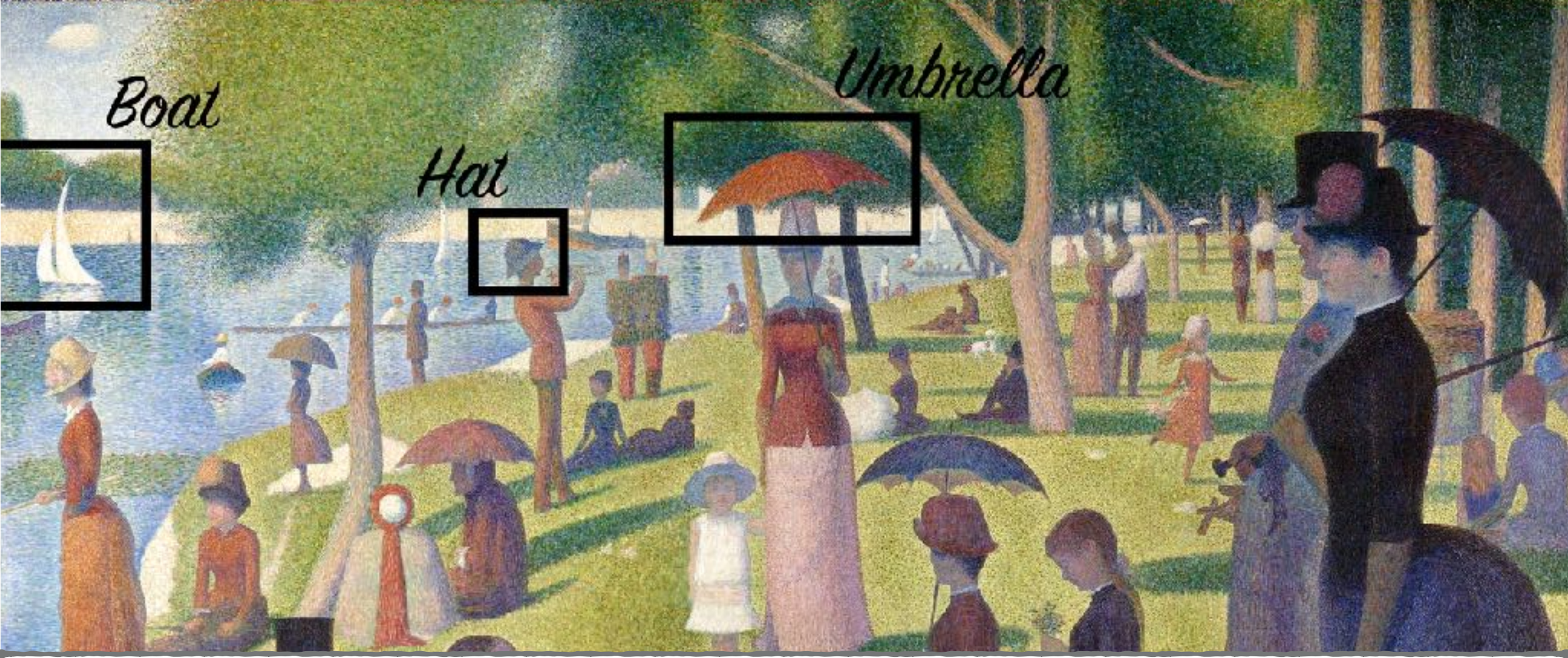
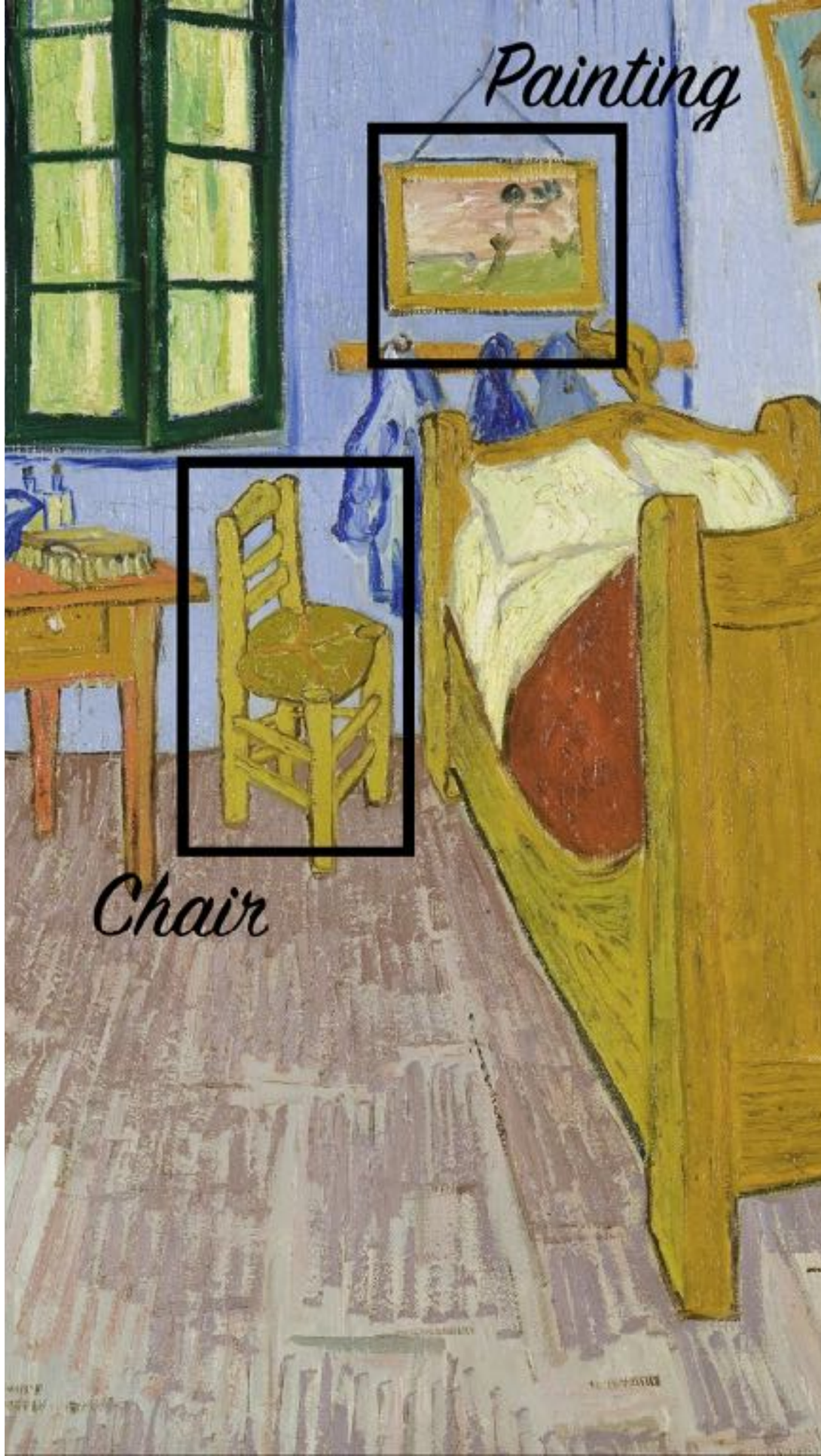


OBJECT DETECTION IN TENSORFLOW

Martina Pugliese
2 May 2019
PyData Edinburgh

SO WHAT IS OBJECT DETECTION?

The little intro



This is an image to **us**:

This is the same image to a **machine**:

```
array([[[151, 240, 254],
        [147, 236, 250],
        [151, 240, 254],
        ...,
        [202,  11,  89],
        [210,  19,  96],
        [206,  16,  90]],

        ...,

       [[206,  40, 112],
        [234,  69, 138],
        [240,  77, 144],
       ...,
        [136,   8,  67],
        [146,  18,  75],
        [152,  25,  80]]], dtype=uint8)
```

Object detection is the task of detecting the pens there and where they are - drawing ***bounding boxes*** *around them*

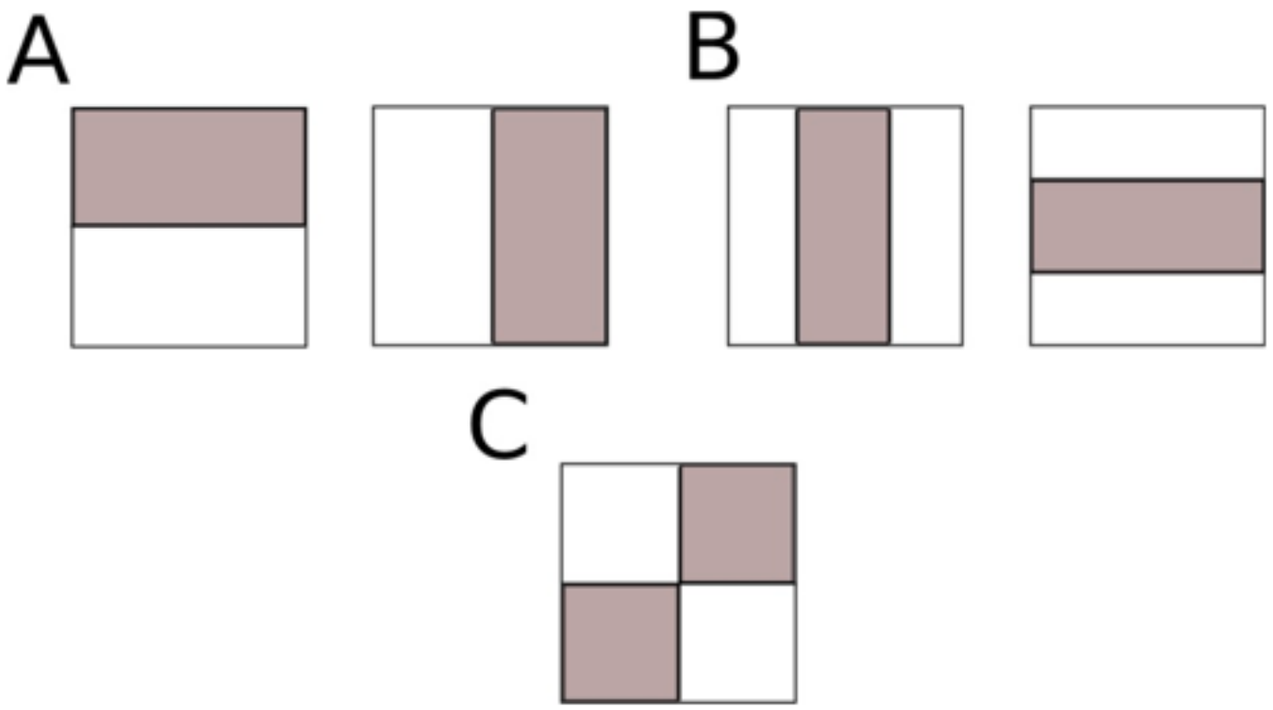
Note that it is a very different problem than a classification (cat or dog)!

METHODS:

NON NN-BASED

You can do many things without deep learning

HAAR CASCADES



- Rectangles are moved over the image, compute sum of intensities
- Comparison to trained images (object/lack of object)
- Train a weak classifier for each feature
- Use AdaBoost to weigh all those
- Put these strong classifiers in a cascade

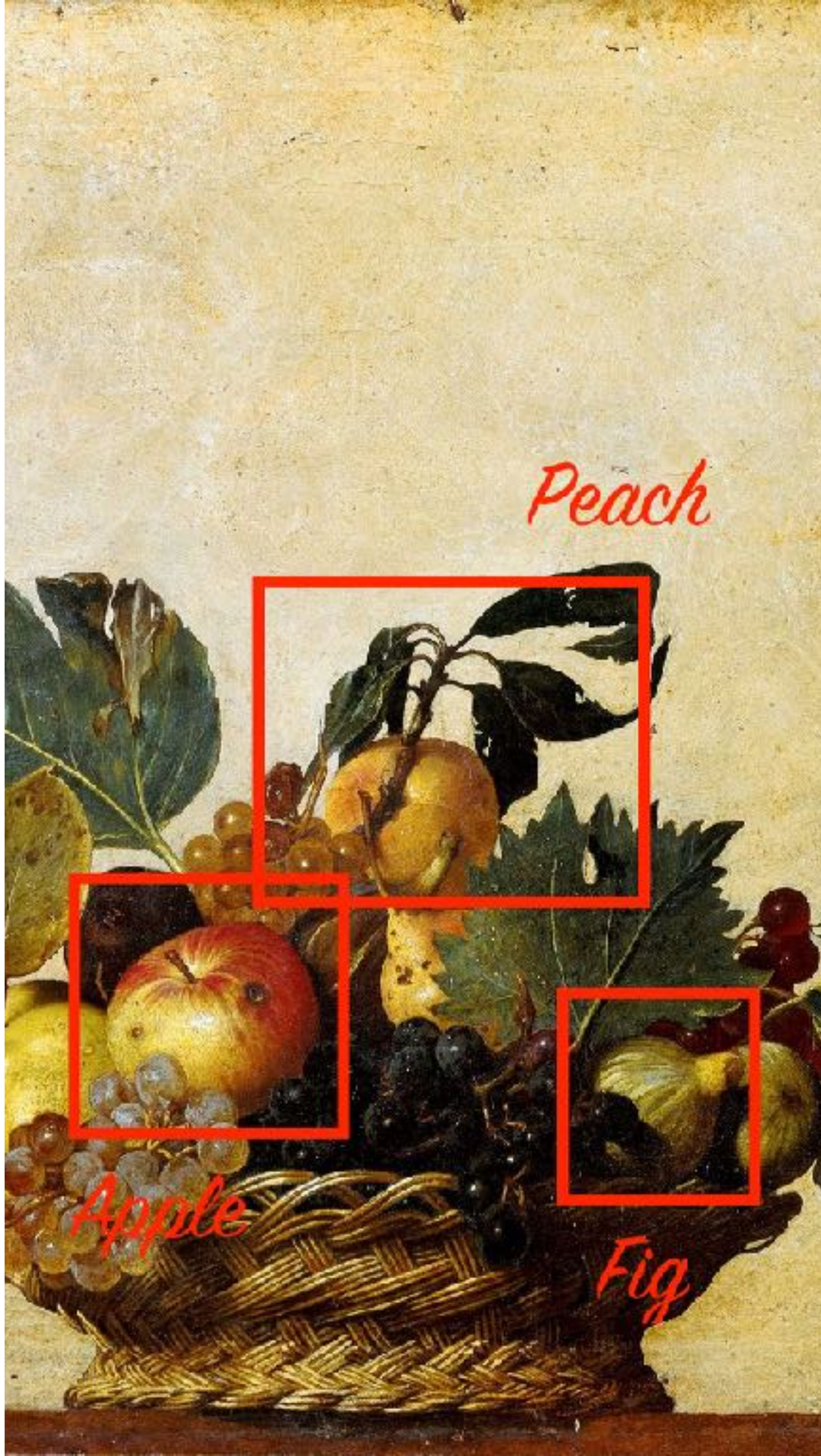
OpenCV comes with pre-trained models for faces, cat faces, plates

You can train it on your own objects

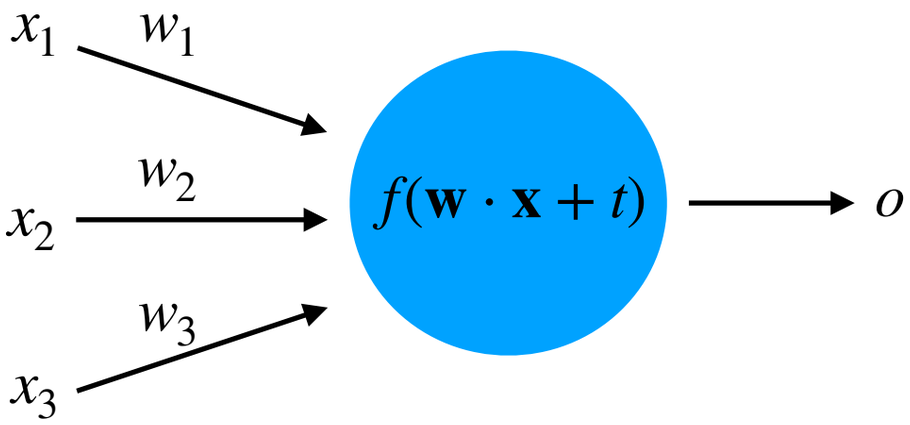
Note there are other methodologies as well

METHODS: DEEP LEARNING ONES

When you really need them



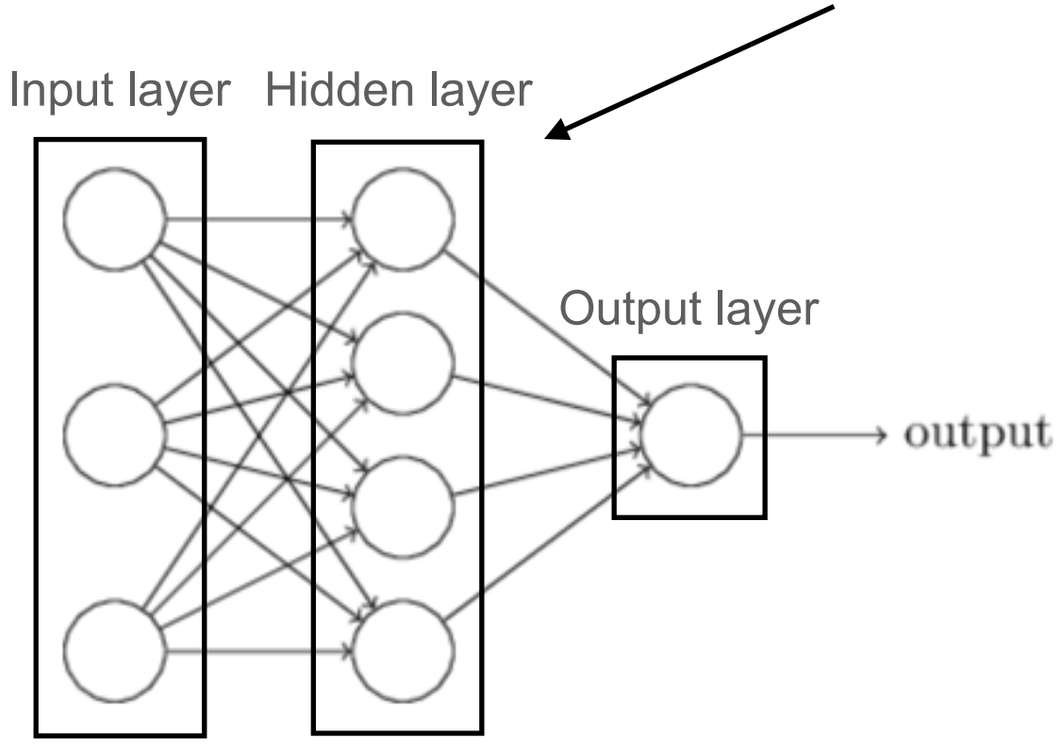
ARTIFICIAL NEURAL NETWORKS (ANNs)



An artificial neuron

- input data and **weights**
- firing **threshold**
- **activation** function

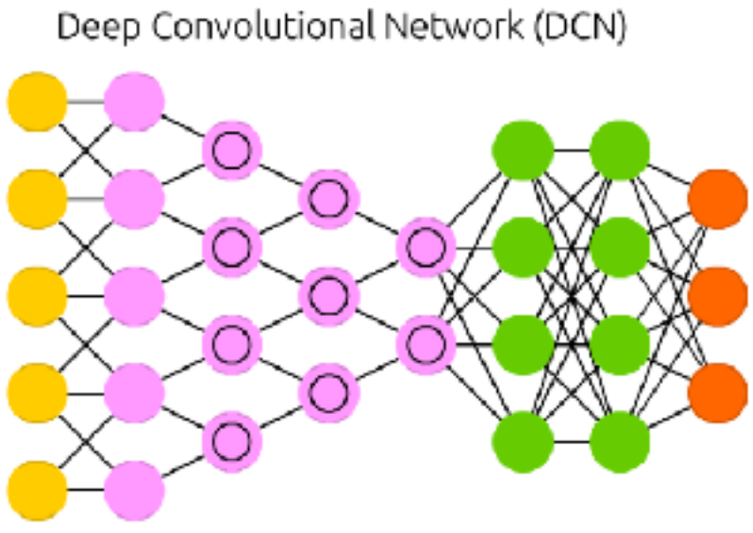
deep means there's several!



An ANN. Img from [1]

- **Weights** are initialised to values - the goal is to adjust them
- Learning: minimise a **loss** function - job of the **optimizer**
- **Backpropagation** improves the weights iteratively -
  - modified by the derivatives of the loss
  - propagated backwards in the net

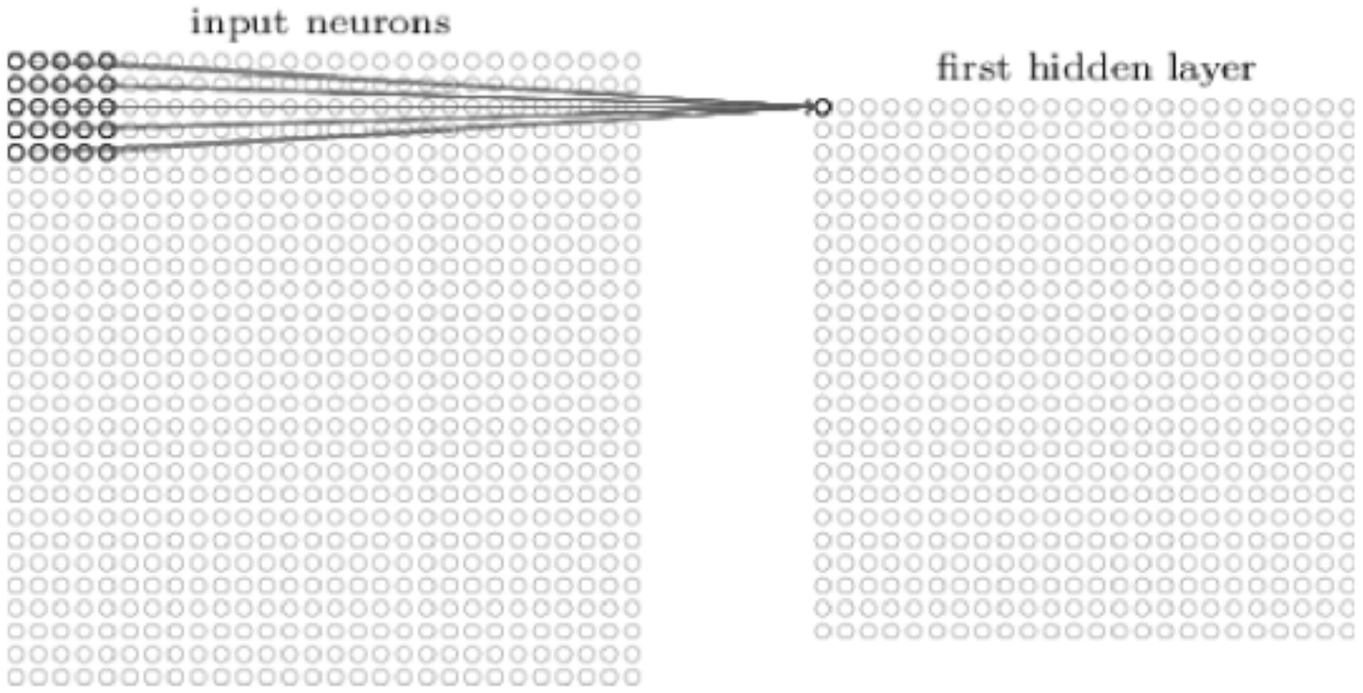
ANNs & VISION



A deep CNN. Image from [4] - the conv layers are highlighted

CNN:

- Some layers are not fully connected
- Output is generated via **convolution** operations
  - every neuron talks to a region of neurons in the layer before
  - a kernel of weights is used to build combinations
  - the combination will pick up when there is agreement
  - the kernel shifts every time
  - a feature map is the output: it encodes shapes
  - shapes are learned hierarchically alongside layers



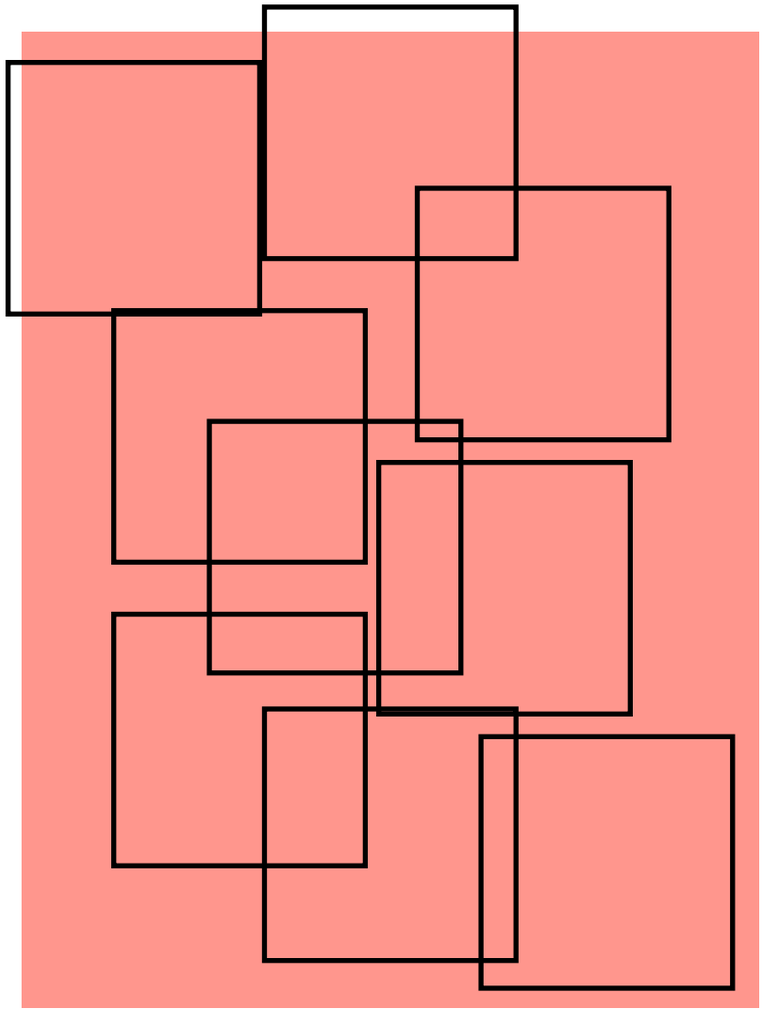
Convolution, image from [1]

The biology

CNNs loosely resemble the functioning of the **visual cortex**, which processed images in batches of pixels (region)

HOW WOULD YOU USE ANNs TO DETECT OBJECTS?

You could think of applying ***a classifier per image region***



- it'd be naive and expensive
- you'd need to consider several regions
- objects overlap
- requires one CNN per region
- objects may have different aspect ratios in different regions

Better methods

- YOLO
- R-CNN
- SSD

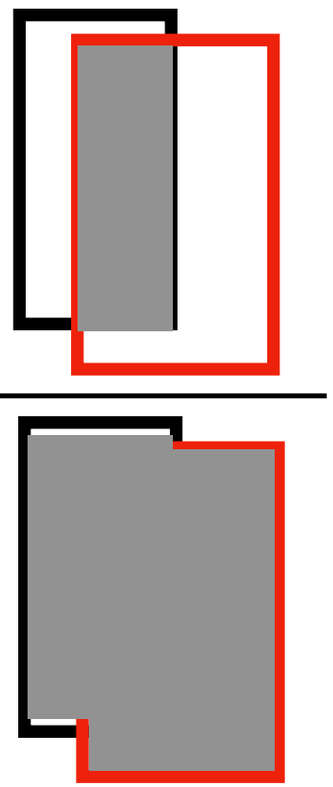
WHEN IS AN OBJECT DETECTED: THE IoU – INTERSECTION OVER UNION

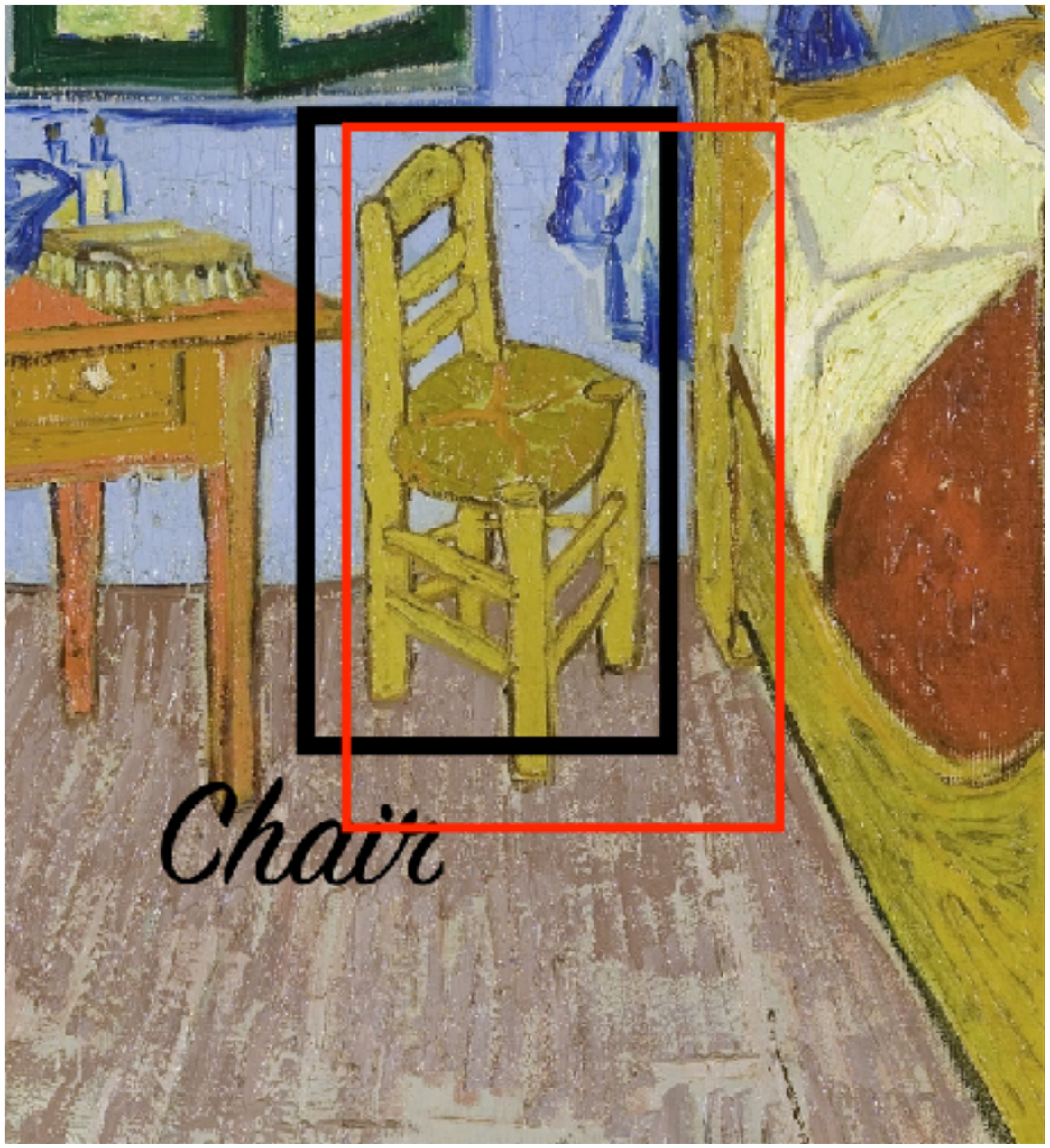
Main metric to assess if the network got an object box right

the box as manually created (training set)

the box as the network found it

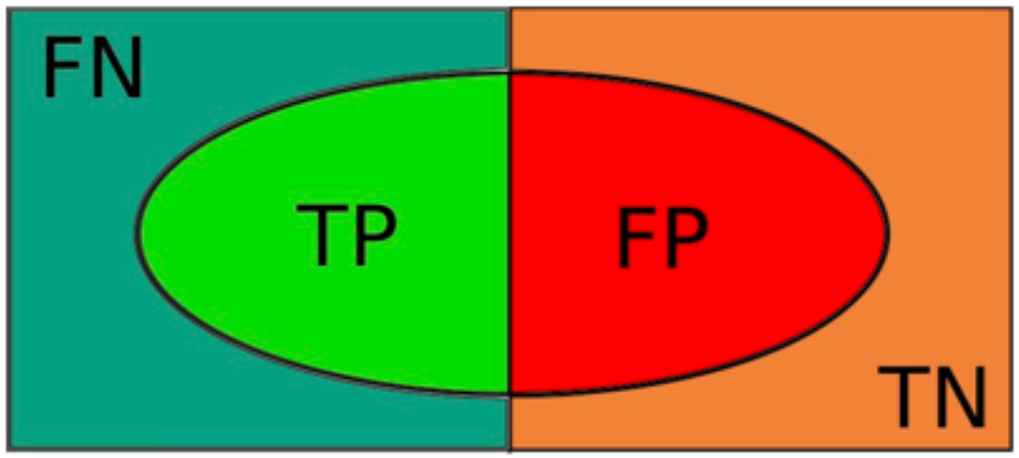
IoU =





usually a IoU of at least 0.5 is considered a detection

ASSESSING A MODEL'S QUALITY – CLASSIFICATION METRICS



positive is your class of interest - binary classification

TP: true positives

FN: false negatives

FP: false positives

FN: false negatives

the fraction of true positives for the class over the total of points belonging to the class, meaning how many of them were detected.

$$p = \frac{TP}{TP + FP}$$

precision - aka fraction of points that are correct for the class (true positive rate), relevancy

[**specificity** is the true negative rate]

$$r = \frac{TP}{TP + FN}$$

recall - aka fraction of points detected in class, completeness

EVALUATING AN OBJECT DETECTOR: THE mAP

mean average precision

$$\text{mAP} = \frac{1}{|C|} \sum_c \frac{TP_c}{TP_c + FP_c}$$

Average of the precisions
for all classes

You'd normally measure the mAP at different values of IoU

and for different boxes sizes

YOU ONLY LOOK ONCE – THE YOLO NETWORK

The year is 2016

- One of the fastest networks for object detection
- It passes over the image just **once**
- It performs a **regression** - predicting the location of the bounding boxes and the class probabilities - for each cell of a grid over the image

- **darknet** is a framework to use YOLO - it comes shipped with pre-trained weights
- it has been ported into TensorFlow as **darkflow**
- it has APIs in **OpenCV**

OTHER "COMPETITOR" NETWORKS

R-CNN

- R is for "region" - works off of a region segmentation
- searches for candidate regions via convolutions (selective search)
- passes them to an SVM classifier for the object
- because of the region selection is quite slow

Fast R-CNN

- Convolution is done once per image
- Still uses selective search
- Still quite slow

Faster R-CNN

- Drops the selective search in favour of a network approach

SSD

- Single-shot multi box detector - one pass overall (localisation/classification)
- Competes with YOLO in speed
- Multibox is the method for extracting candidate bounding boxes

THE SUPER BRIEF TENSORFLOW JOURNEY

Why am I here?
What do I want?

OBJECT DETECTION IN TENSORFLOW

Why am I here?
What do I want?

USING TENSORFLOW FOR OBJECT DETECTION

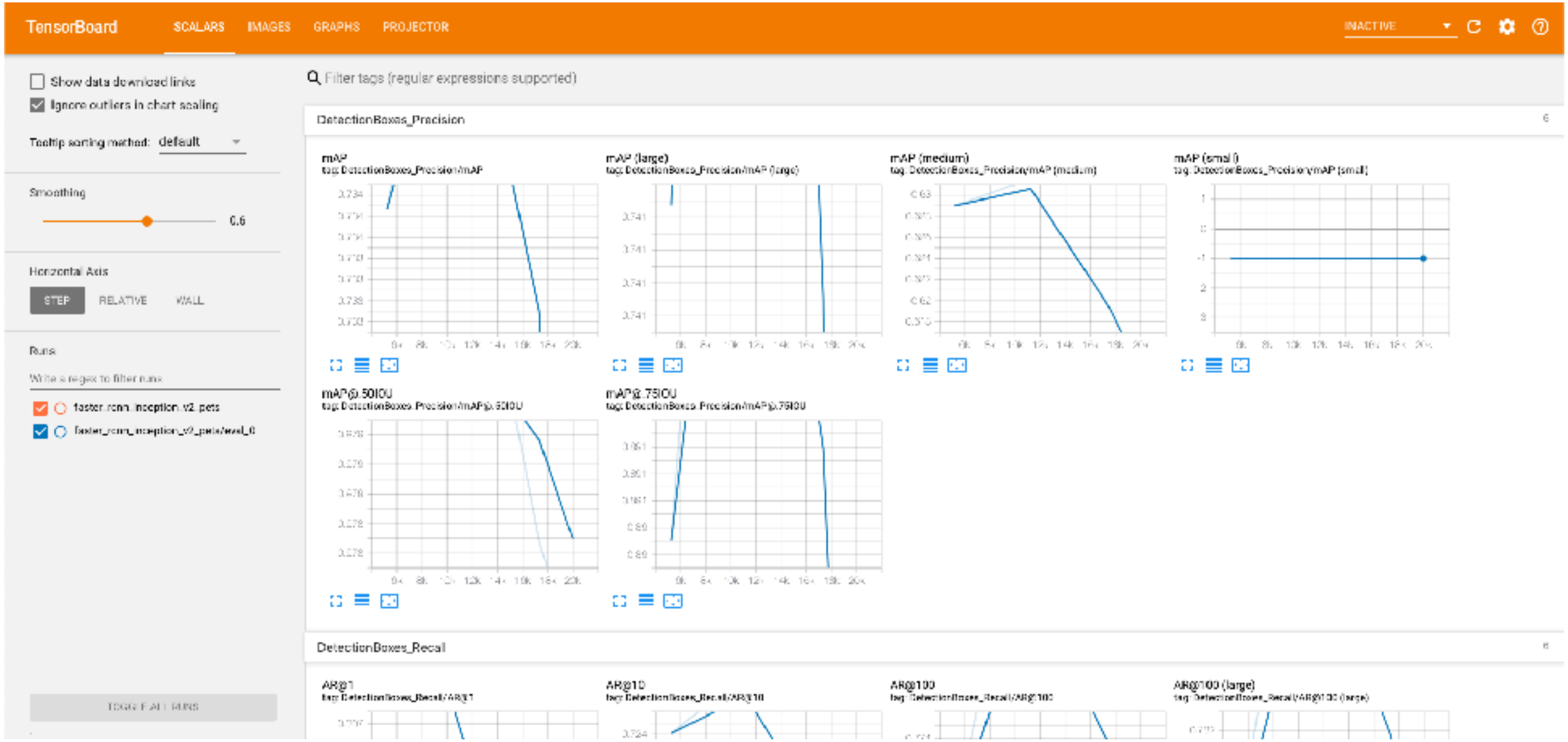
1. Prepare the images and annotate them

2. Configure the job

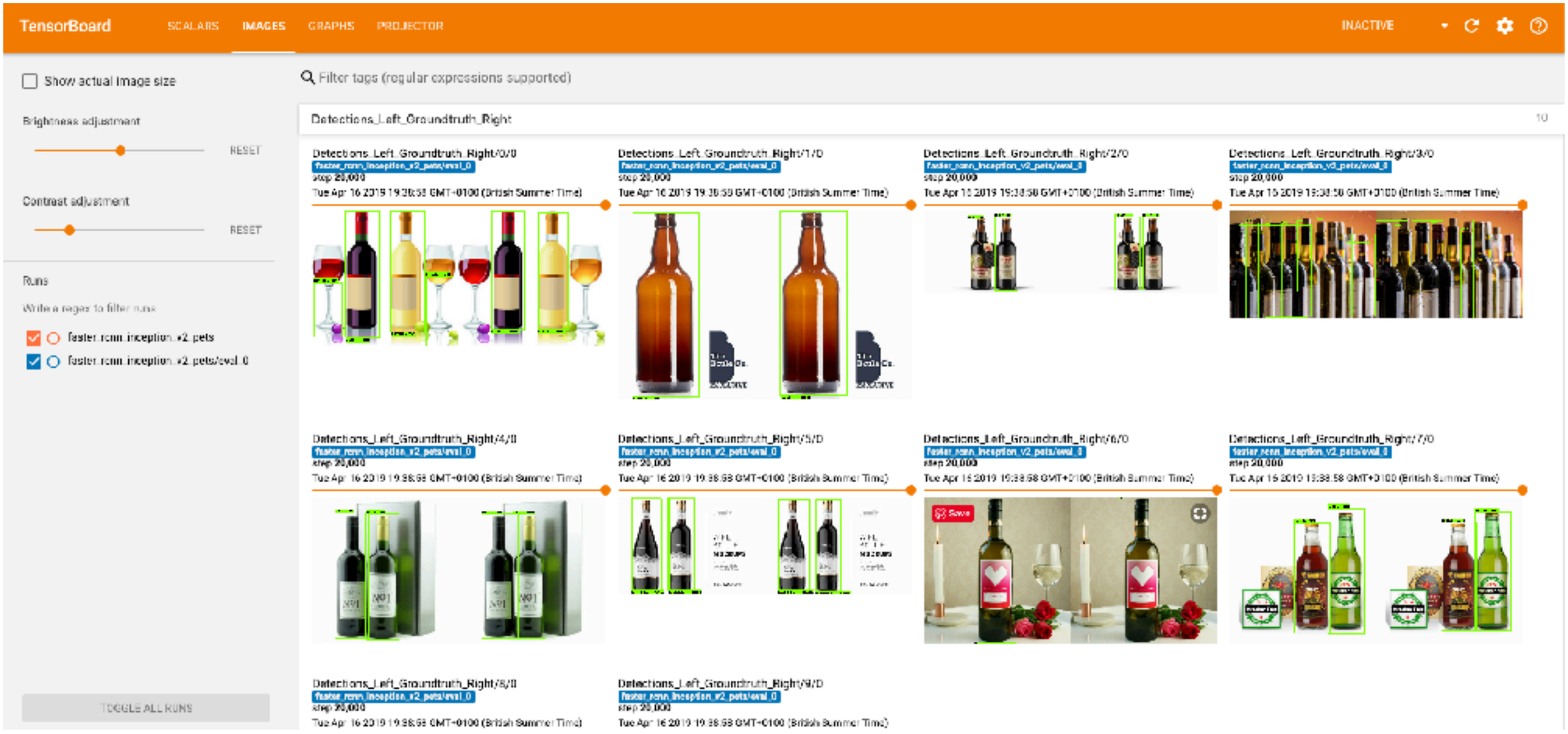
3. Download the checkpoints for transfer learning

4. Use TensorBoard

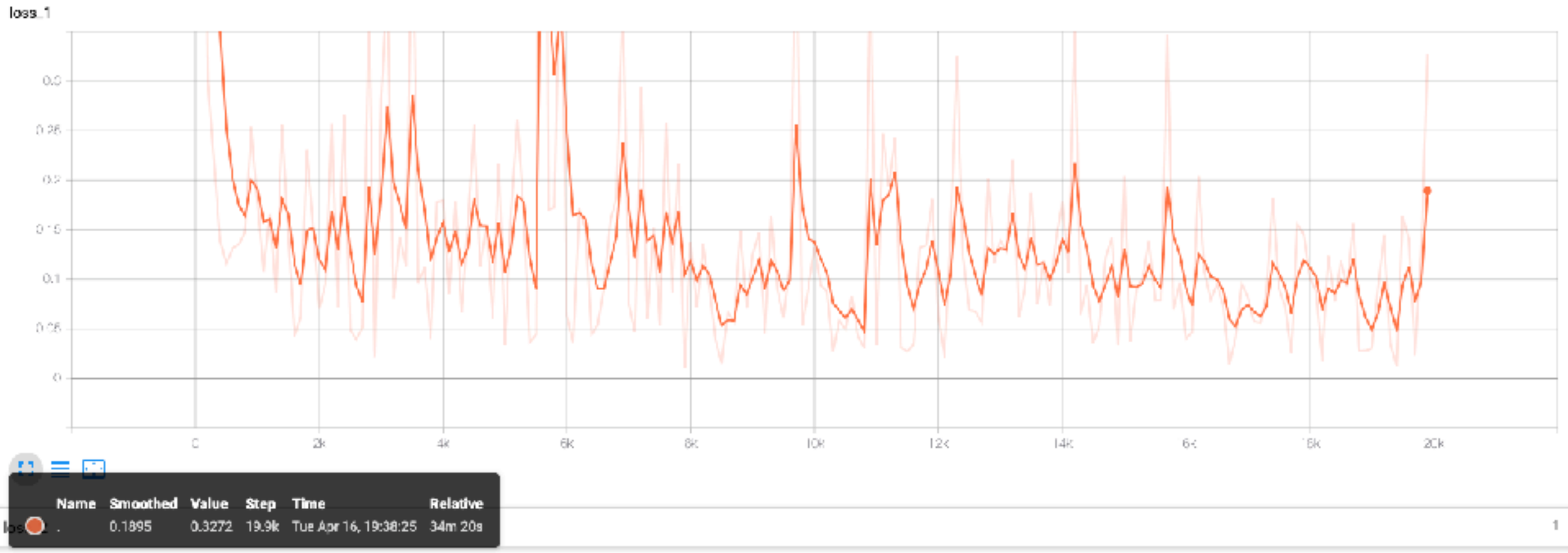
Tensorboard – The performance metrics



Tensorboard – The detected objects on the test set



Tensorboard – The Loss



SOME REFERENCES

Haar Cascades

1. The original Haar Cascades **paper**

ANNs & Deep Learning

1. M Nielsen, **Neural networks and deep learning**, Determination Press, 2015
2. F Chollet, Deep Learning with Python, Manning, 2017
3. **Convolutional Neural Networks for visual recognition**, a Stanford CS class
4. **The neural network zoo**, an article + illustration by F Van Veen at the Asimov Institute

YOLO

1. YOLO original papers: **YOLOv1 paper**, **YOLOv2 paper** and **YOLOv3 paper**
2. A high-level **blog** post about the YOLO paper, introducing its concepts
3. YOLO's **website** where to download pre-trained weights

Other networks for object detection

1. A **blog** on various object detection algorithms
2. The R-CNN original **paper**
3. Faster R-CNN original **paper**
4. A Kharpathy on trying R-CNN back in the day, **here**
5. The SSD original **paper**

In TensorFlow

1. The TensorFlow models **repo**
2. The **labelImg** tool to annotate
3. A **blog** on how to use TensorFlow models for object detection - training a racoon detector

Church

THANKS!

Gondola