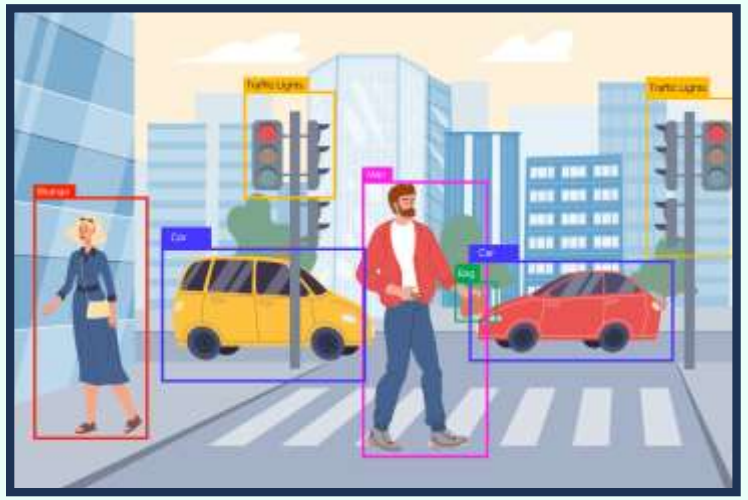
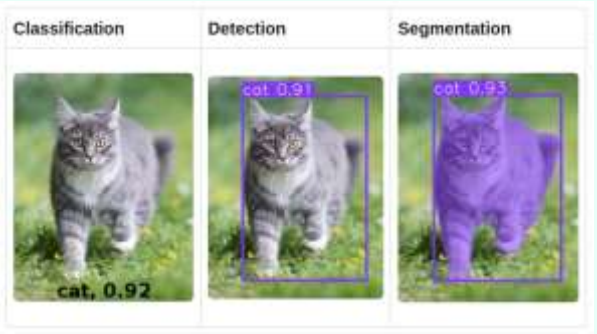
Making Your First Object Detection Model

Henry Robbins



What is Object Detection?

What is Object Detection?



- Computer vision task that involves identifying and locating objects
  - Used in many cutting-edge applications including self-driving cars, robotics, and surveillance
- Encapsulates many different algorithms and model types
  - Today I will talk about one in particular: the YOLO model
- For YOLO we have several choices of models depending on which task we wish to complete
  - Classification, detection

What is Object Detection? (contd.)

- Today I will talk about using and training a detection model
  - We will use it to detect, classify, and locate
- Can use one picture or many pictures (video)
- Technical Information:
  - Today I will be using the YOLOv8 model
    - Now we have YOLOv9
  - YOLO is a Convolutional Neural Network (CNN) which was written and runs on the pytorch framework
  - YOLO is offered through the Ultralytics API which is often used in python

Small Note on CNNs

- A popular type of feed-forward neural net
  - This means that each neuron in a layer is connected to each neuron on the next layer
  - CNN allows for convolutional layers which is more practical
  - Also means data is prone to “overfitting” to your data
    - Will talk about how to prevent this
- Inspired by biology and very similar to the visual cortex in animals
- Uses little preprocessing

More Info. On YOLOv8

- Several different ‘types’ of models to choose from (more on this later)
- Each ‘type’ allows us to choose our balance of speed vs. accuracy
- The model will exist as a *.pt file
- All YOLOv8 models are pretrained on the COCO dataset
- This allows us to play around with computer vision but limits our use cases in practicality

Using the Model

- Once you import the *.pt model, how can you leverage it?
- When inputting images into the model, it spits out information
- You can use this information how you wish in your code to do as you wish
- Returns bounding boxes with coordinates, confidence, and class labels
- Now lets see how to do it…

How Can *I* Do it?

Step 1.

Take many pictures

Take pictures

- This step is self explanatory
- Use whatever picture taking device you wish to take pictures
- In this case bigger is better
  - Taking more pictures in different environments will help you build a better model

Step 2.

Annotate these images

Annotate Images

- For this there is many options
  - Make sure it is in the format of whatever model you will train
- For me in this case we will be training a YOLOv8 model
- I used Roboflow to annotate my images
  - Allow easy ability to annotate, label, and export
  - Don’t know if the website itself is open source, but when you upload images, it makes you open-source licenses all the images you choose to upload and annotate

Step 3.

Export these images

Export Images

- Export these images into the machine you wish to train your model on
- I will use colab
  - I unfortunately do not have access to a supercomputer
- Roboflow will allow many ways to export these images, either via zip file or through a script
- However you do it *make sure the files are there*

Step 4.

Code.

Code

- Not too much code
  - ‘only three lines’

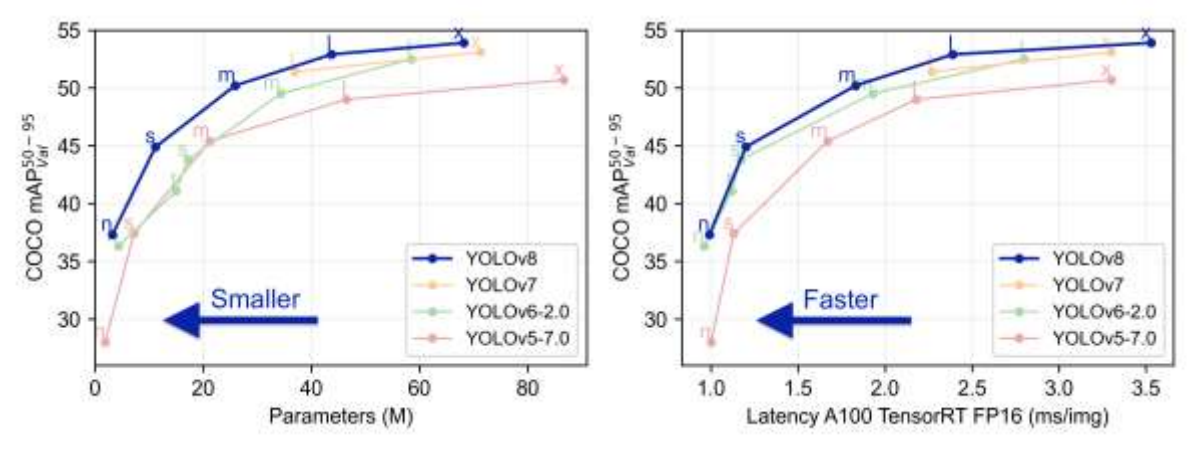
  Link

Step 5.

Understand the code

Understand the code

- Easy to copy and paste the code from stack overflow
  - But what does it mean???
- Import model
  - Choose what kind of model



Understand the code (contd.)

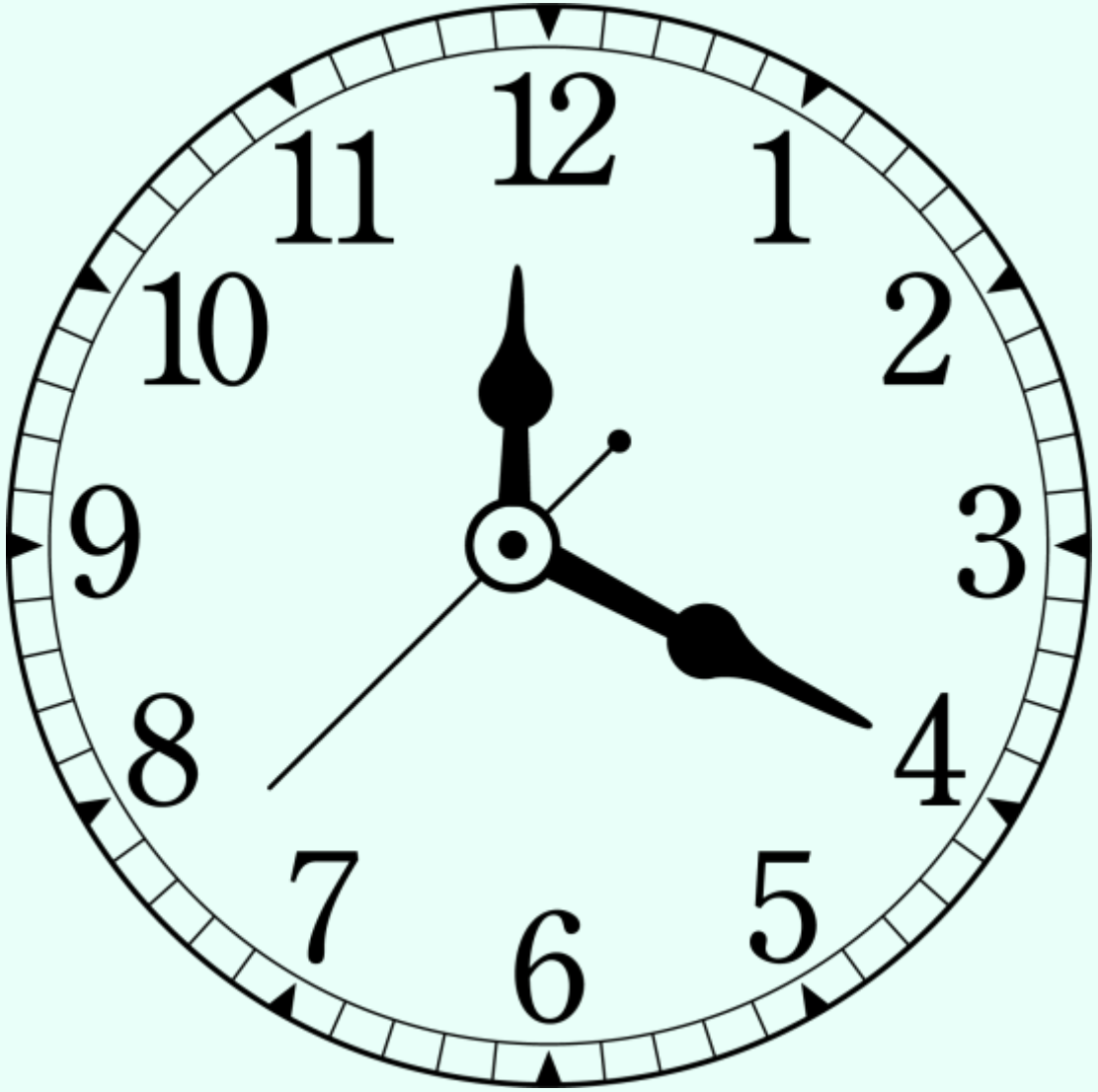
- Train the model
  - Parameters – start off small
- data=('path to data.yaml') ← only required option – check file
- epochs=100 ← The number of training cycles (100 is default)
- name=\model

Understand the code (contd.)

- Imgsz=640 ← Image size, all images resized to this size (default)
- batch=8 ← batch size of training before models parameters are changed can use -1 for autoBatching (default is 16)
- MANY more check them out HERE

Step 6.

Wait


12
1
2
3
4
5
6
7
8
9
10
11

Step 7.

Understand the results

Understanding the Graphs

- More Information

Step 8.

The great beyond…

I Trained My Model. Now what?

- Next step is to use the information from the model on real life data to complete what you wish to do
- How?
  - More code…
- We now have a *.pt file we can use to predict objects locations in an image

Now what? (contd.)

- Need access to webcam with drivers that work on your kernel
  - Can also just use prerecorded pictures and videos – less fun
- Besides that, simply need some special lines of code to put the box on the screen

- This allows the coordinates of the detected object and what *kind* of object we see in frame
  - Maybe more than 1

Now what? (contd.)

- Information will be returned in a results tuple
- First index is the info we want
- This will contain the bounding boxes, and information you need to complete your tasks in your projects