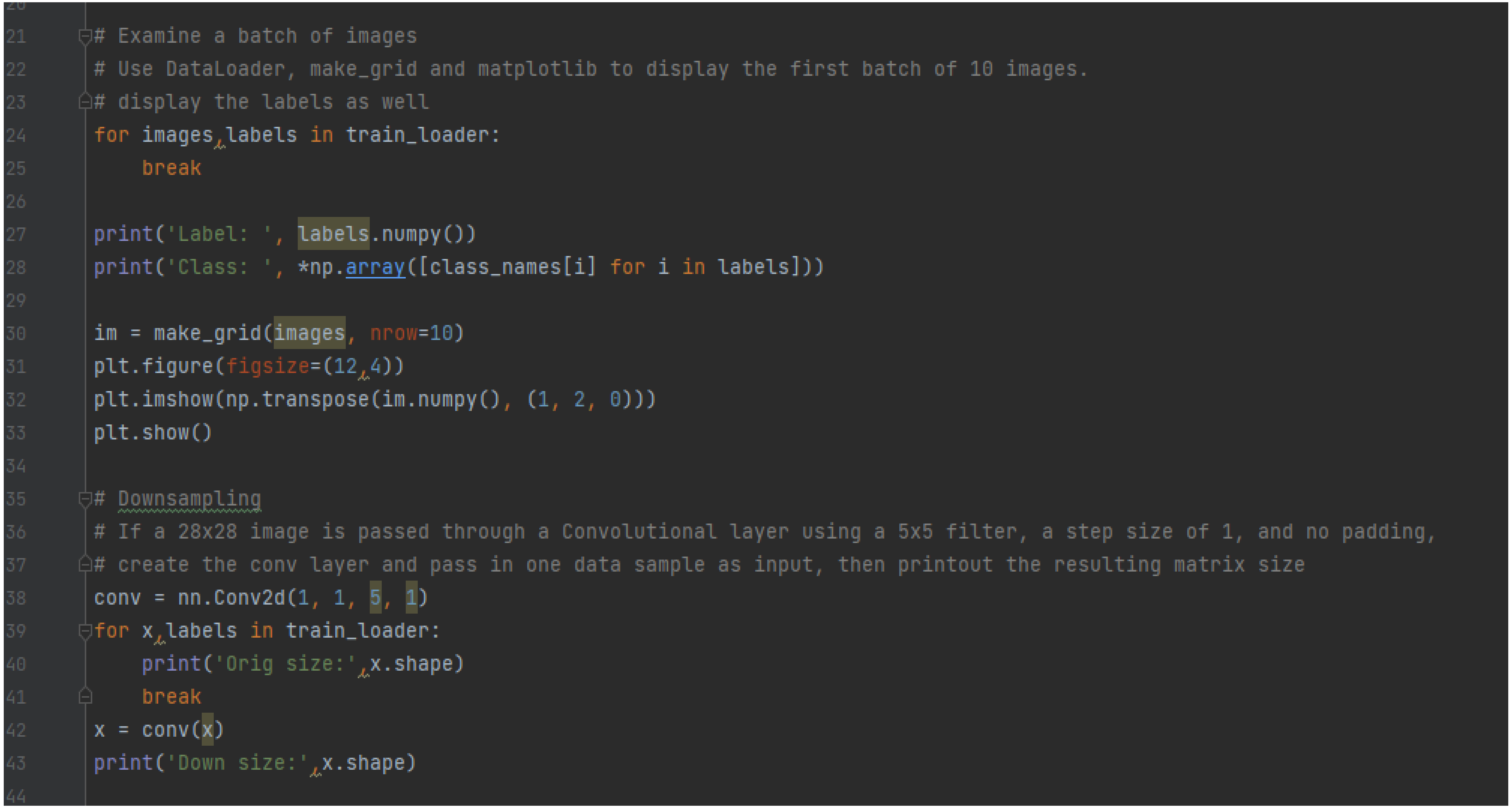
ZeroBox

# Hand-on Real Industrial Machine Learning / AI Project (in Computer Vision)

Zijian Kuang

Dec 18th, 2021

# Assignment 3 Solution



```python
# Examine a batch of images
# Use DataLoader, make_grid and matplotlib to display the first batch of 10 images.
# display the labels as well
for images,labels in train_loader:
    break

print('Label: ', labels.numpy())
print('Class: ', *np.array([class_names[i] for i in labels]))

im = make_grid(images, nrow=10)
plt.figure(figsize=(12,4))
plt.imshow(np.transpose(im.numpy(), (1, 2, 0)))
plt.show()

# Downsampling
# If a 28x28 image is passed through a Convolutional layer using a 5x5 filter, a step size of 1, and no padding,
# create the conv layer and pass in one data sample as input, then printout the resulting matrix size
conv = nn.Conv2d(1, 1, 5, 1)
for x,labels in train_loader:
    print('Orig size:',x.shape)
    break
x = conv(x)
print('Down size:',x.shape)
```

# Agenda

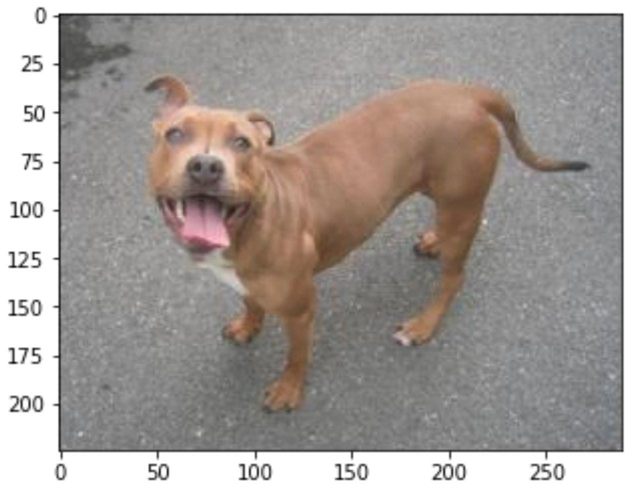
- CNN on Custom Images
- Overview of object detection and YOLO model
- Google Colaboratory
- How to use YOLO object detection to solve real industrial problems
- Prepare real industry product image files provided by Zerobox
- Setup Configurations for YOLO model
- Run Training
- Test Results

# CNN on Custom Images

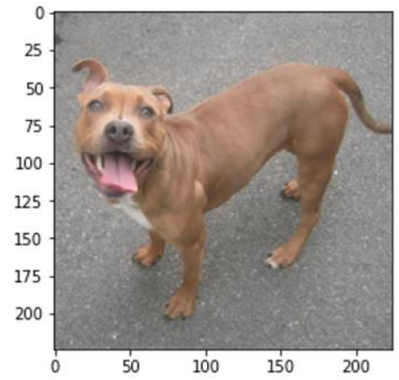
- When working with “real” image data, we need to keep in mind the various preprocessing steps that we can perform on the data.
- Aspect Ratios, Scaling, Normalization, Transforming to Tensor
- The data sets we saw had tens of thousands of images, often our real image data won’t be as large.
- So what approaches can we take?
- **Data Augmentation**: perform a variety of transformations to expand our data set.

# CNN on Custom Images

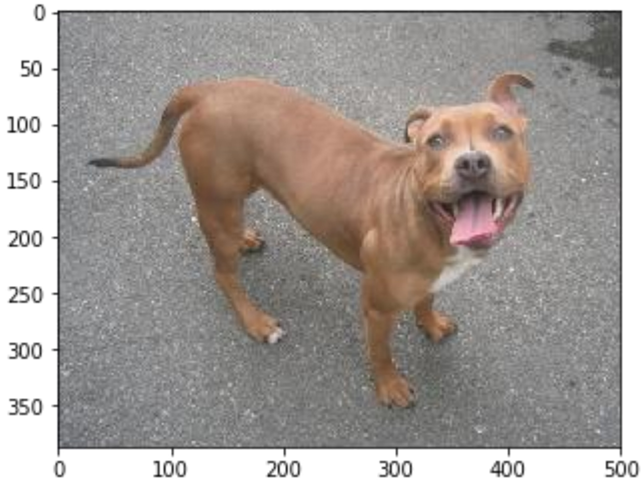
Original Image



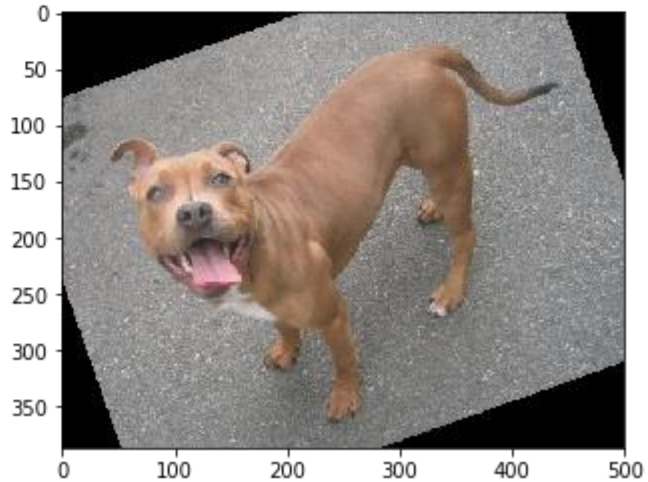
Resize and Crop



Flip



Rotate



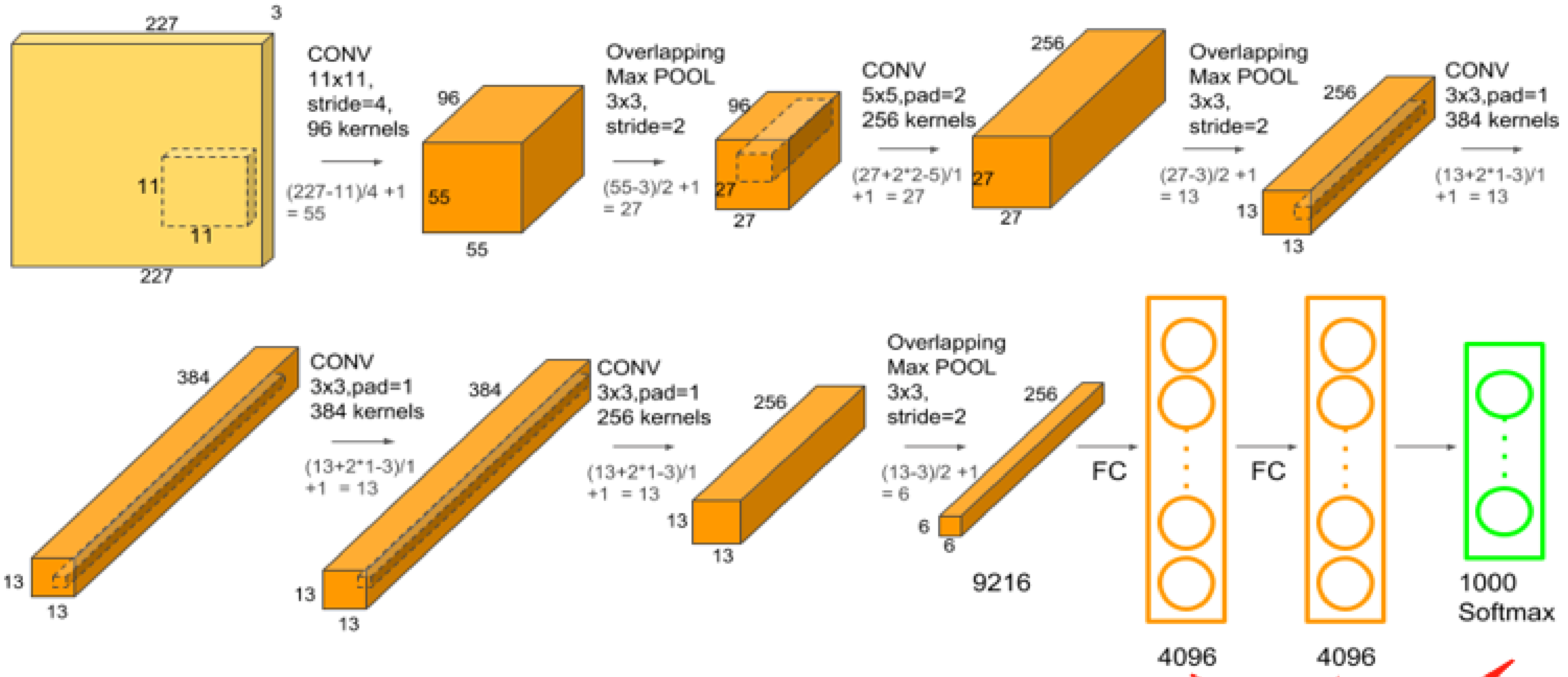
# CNN on Custom Images

- Simple transformations like these allow us to greatly increase the number of data points, as well as make the model more robust to variations on an image.
- Let’s explore how to load in image data and perform transformations.
- Afterwards, we’ll explore how to train on this “real” image data.

# Pre-trained CNN models

- Torchvision has a number of pre-trained models available through torchvision.models
- These have all been trained on the **ImageNet database** of images
- The **ImageNet dataset** is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured. ImageNet contains more than 20,000 categories with a typical category, such as “cat" or “dog"
- All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224.
- **The images have to be loaded in to a range of [0, 1] and then normalized using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].** Ref: https://pytorch.org/vision/stable/models.html
- In our next coding project, we will use the **AlexNet** as an example.

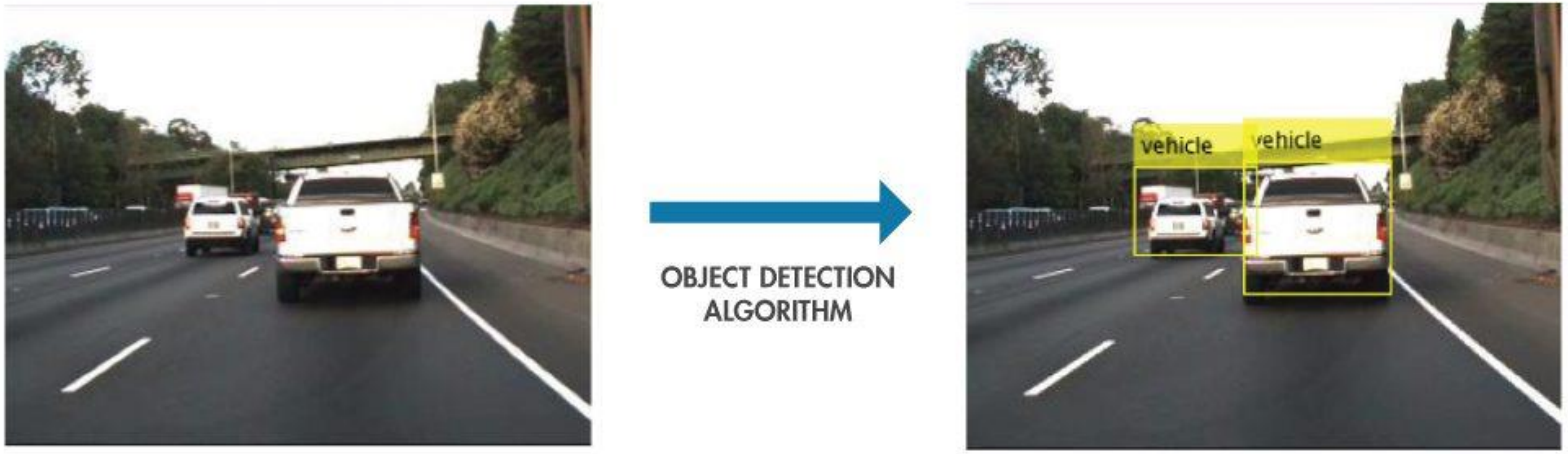
# AlexNet



We only want to refine the last couple FC layers

# Overview of object detection

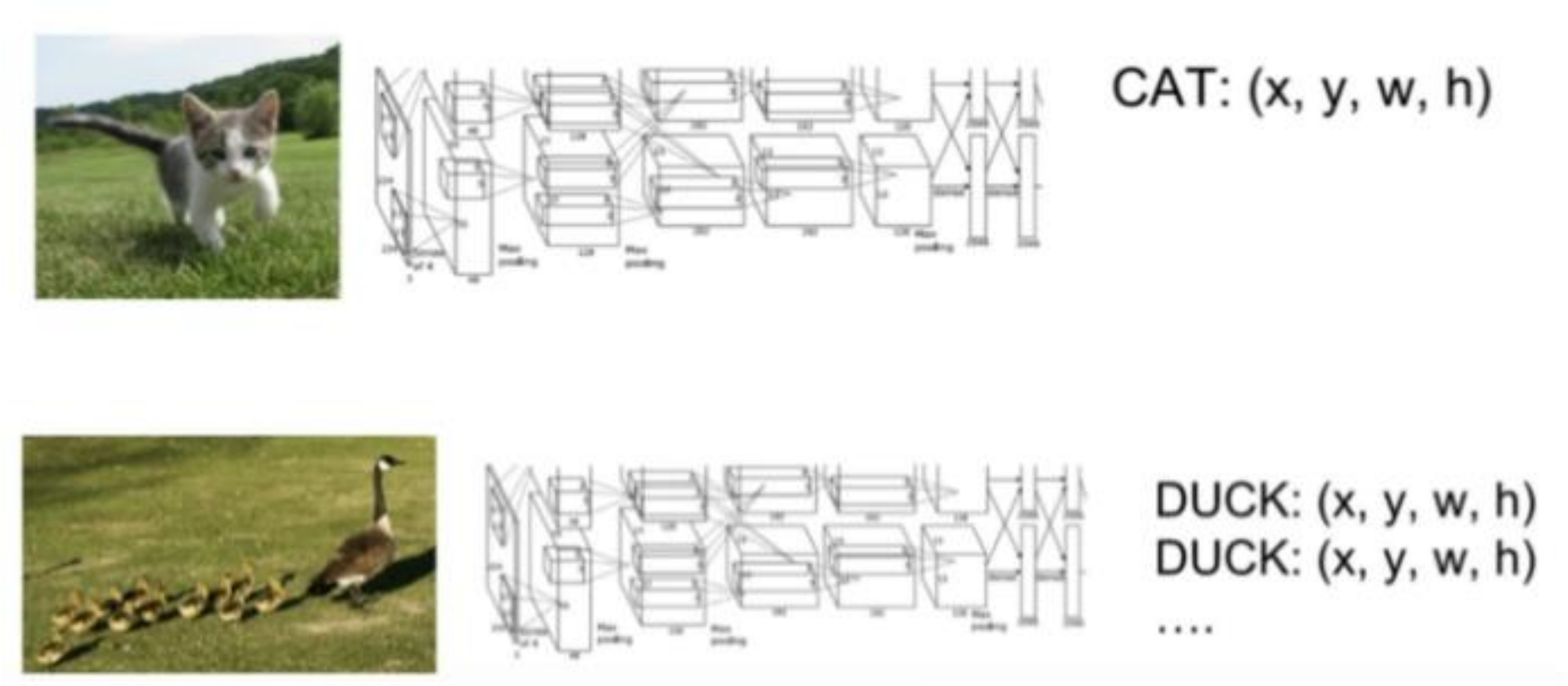
- Object detection is a key technology behind advanced driver assistance systems (ADAS) that enable cars to detect driving lanes or perform pedestrian detection to improve road safety. Object detection is also useful in applications such as video surveillance or image retrieval systems.



https://www.mathworks.com/discovery/object-detection.html

# Overview of object detection

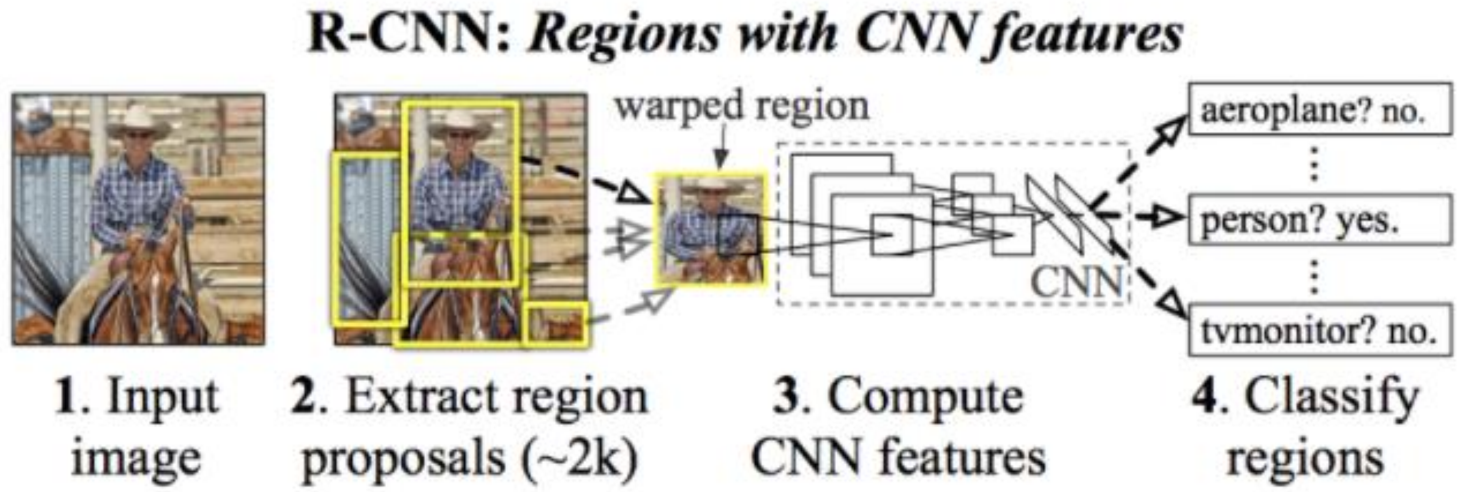
- You can use a variety of techniques to perform object detection. Popular deep learning-based approaches using **convolutional neural networks (CNNs),** such as R-CNN and YOLO v2, automatically learn to detect objects within images.



https://www.mathworks.com/discovery/object-detection.html

# Two-Stage object detection Networks

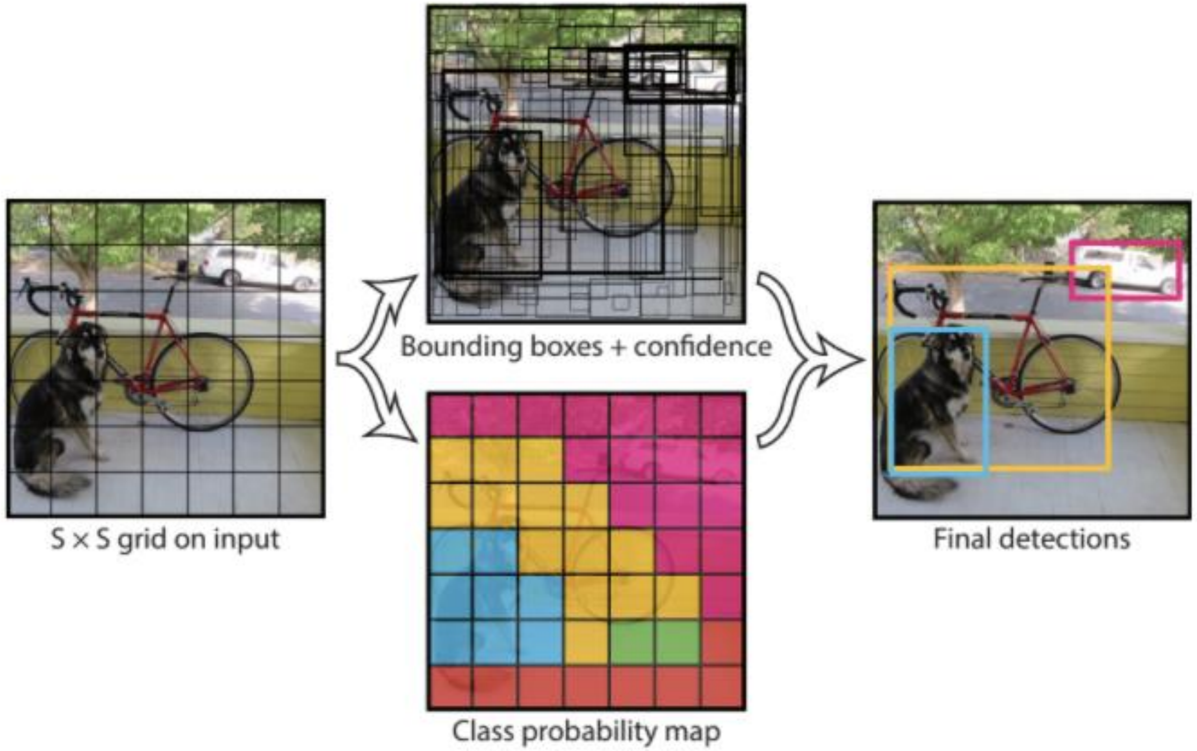
- The initial stage of two-stage networks, such as R-CNN and its variants (Fast RCNN, Mask-RCNN, etc.), identifies region proposals, or subsets of the image that might contain an object. The second stage classifies the objects within the region proposals. Two-stage networks can achieve very accurate object detection results; however, they are typically slow



https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

# Single-Stage object detection Networks

- In single-stage networks, such as YOLO (You Only Look Once), the CNN produces network predictions for regions across the entire image using anchor boxes, and the predictions are decoded to generate the final bounding boxes and the classes for the objects.



https://www.mathworks.com/discovery/object-detection.html

# YOLO

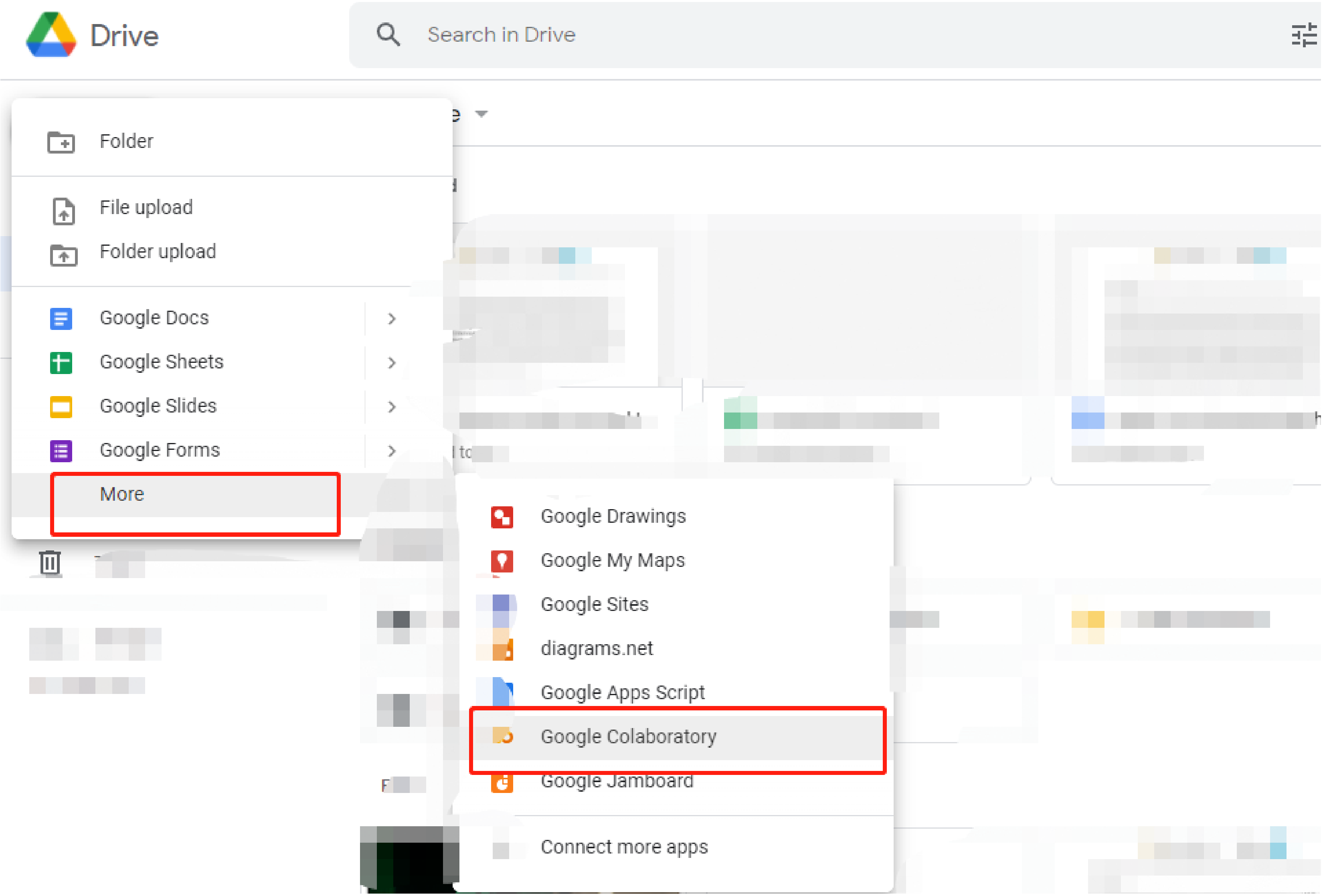
# Fork and Clone YOLOv5

- https://github.com/ultralytics/yolov5

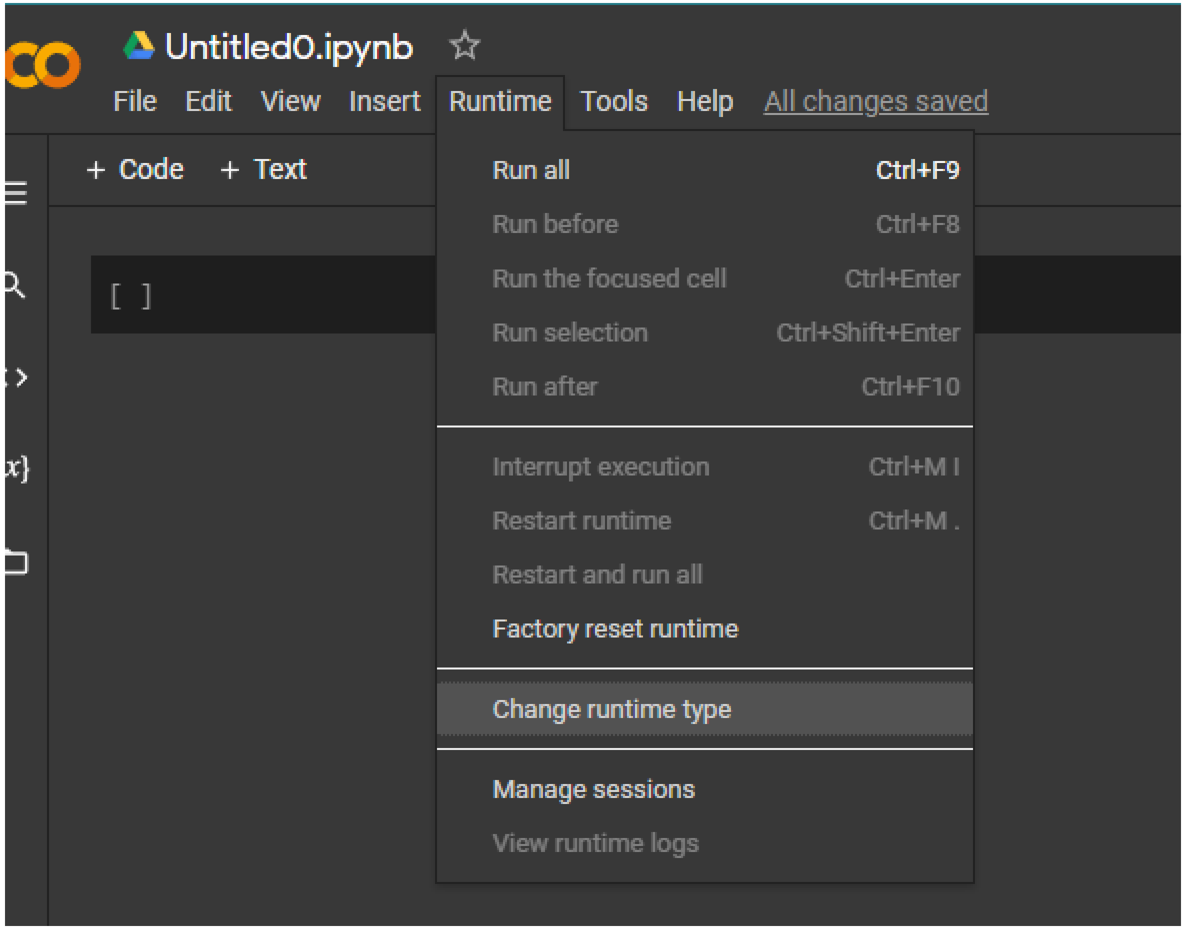
# Google Colab (Jupyter Notebooks)

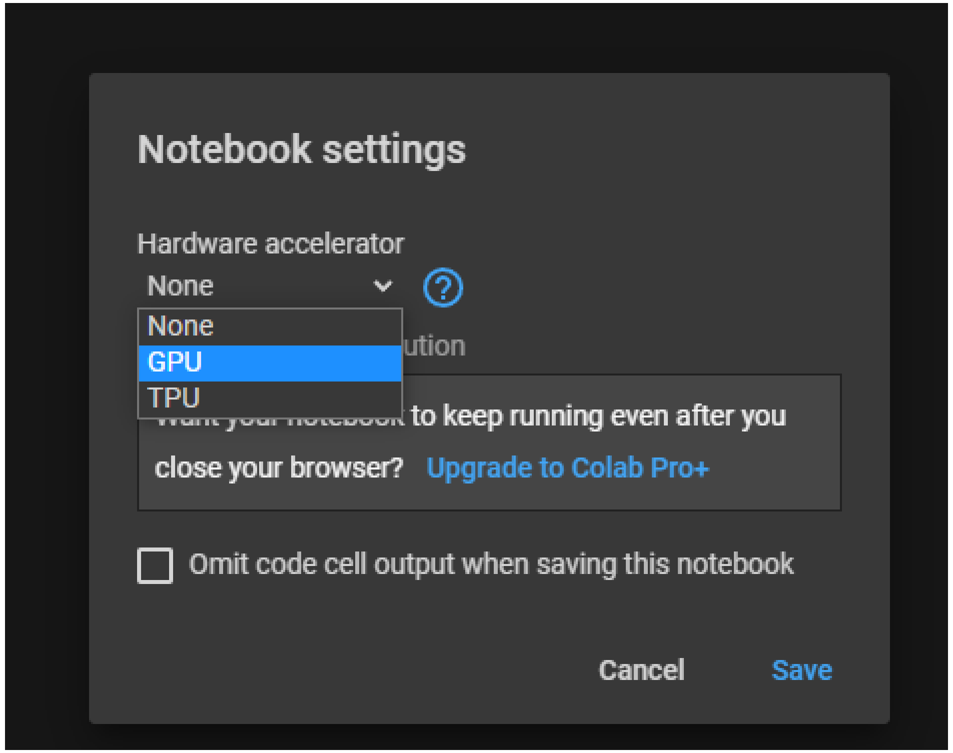
- Hosted by Google
- Access to **GPU** and **TPU** (Tensor Processing Unit)
- Your code files will be stored in Google Drive (seem as other google products such as google doc, etc.)
- Tons of pre-installed libraries for deep learning/machine learning/data sciece (such as numpy, scipy, pytorch, etc.)
- https://colab.research.google.com/?utm_source=scs-index#scrollTo=C4HZx7Gndbrh

# Google Colab (Jupyter Notebooks)



# Google Colab (Jupyter Notebooks)




Notebook settings

Hardware accelerator

None

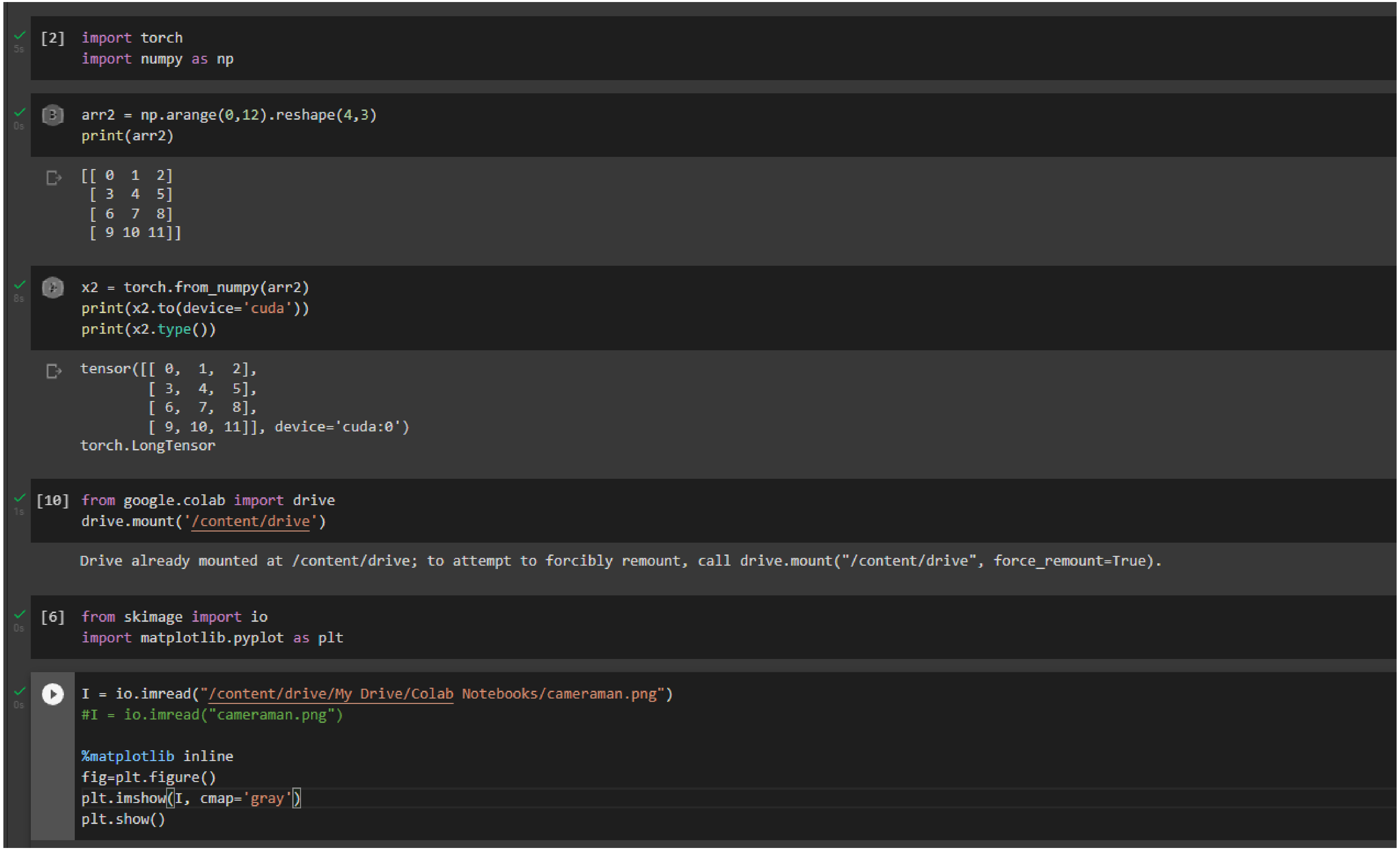
None
GPU
TPU

ution

Want your notebook to keep running even after you close your browser? Upgrade to Colab Pro+

□ Omit code cell output when saving this notebook

# Google Colab Exercise



```
import torch
import numpy as np
```

```
arr2 = np.arange(0,12).reshape(4,3)
print(arr2)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
x2 = torch.from_numpy(arr2)
print(x2.to(device='cuda'))
print(x2.type())
```

```
tensor([[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11]], device='cuda:0')
torch.LongTensor
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

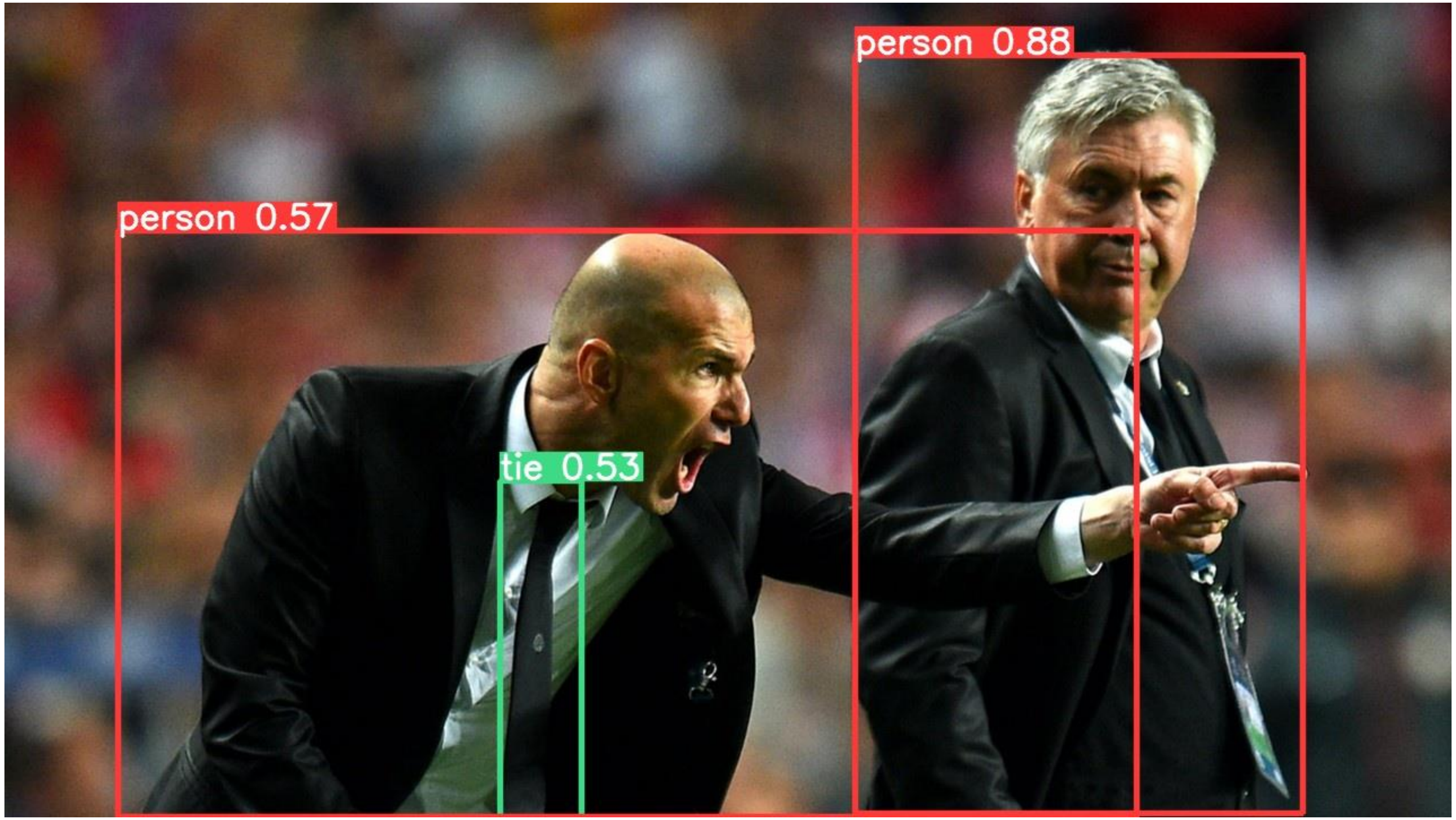
```
from skimage import io
import matplotlib.pyplot as plt
```

```
I = io.imread("/content/drive/My Drive/Colab Notebooks/cameraman.png")
#I = io.imread("cameraman.png")

%matplotlib inline
fig=plt.figure()
plt.imshow(I, cmap='gray')
plt.show()
```

# Google Colab YOLOv5

- https://github.com/ultralytics/yolov5

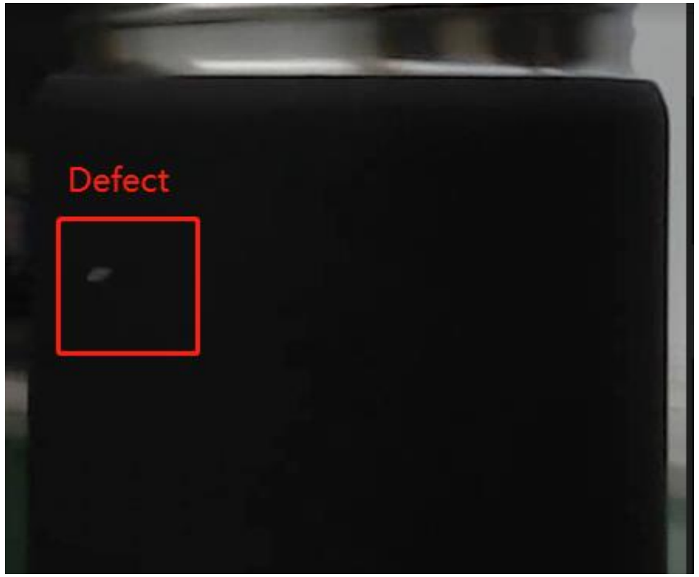


# Bottle Surface Defect Detection (Zerobox)

- Visual defect detection is critical to ensure the quality of most products. However, the majority of small-medium manufacturers still rely on tedious and error-prone human manual inspection.

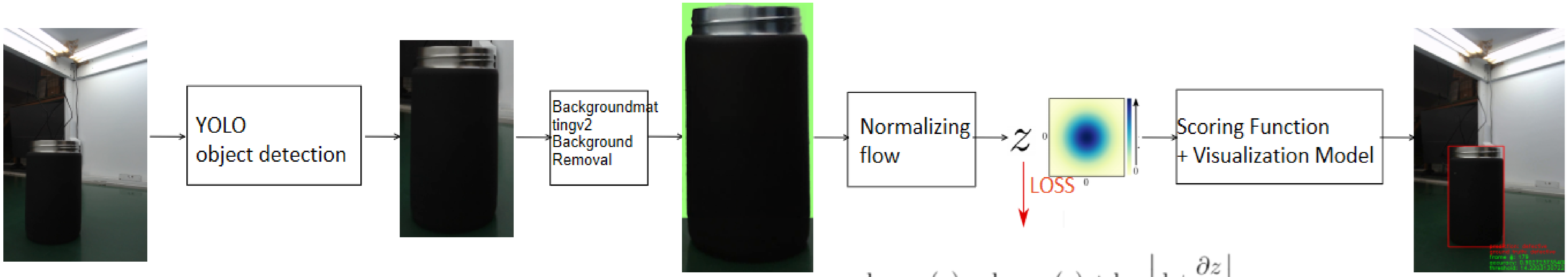
# Bottle Surface Defect Detection (Zerobox)

- Real Industrial Raw Data – Video clips and pictures collected from the factory assembly line



# Bottle Surface Defect Detection (Zerobox)

- One potential solution:



YOLO object detection

Backgroundmatting v2 Background Removal

Normalizing flow

$z$

LOSS

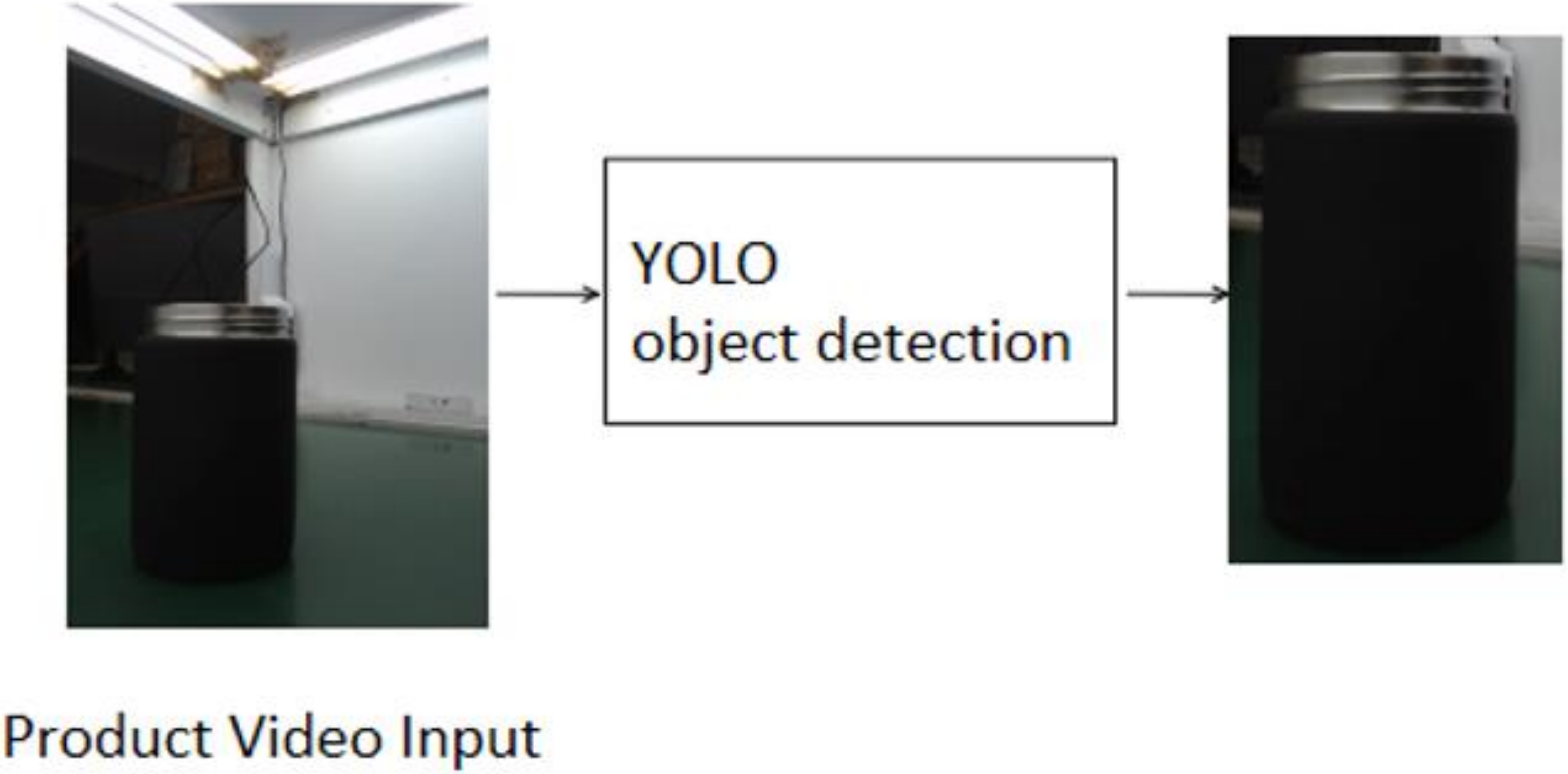
Scoring Function + Visualization Model

$$\log p_Y(y) = \log p_Z(z) + \log \left| \det \frac{\partial z}{\partial y} \right|$$

Product Video Input

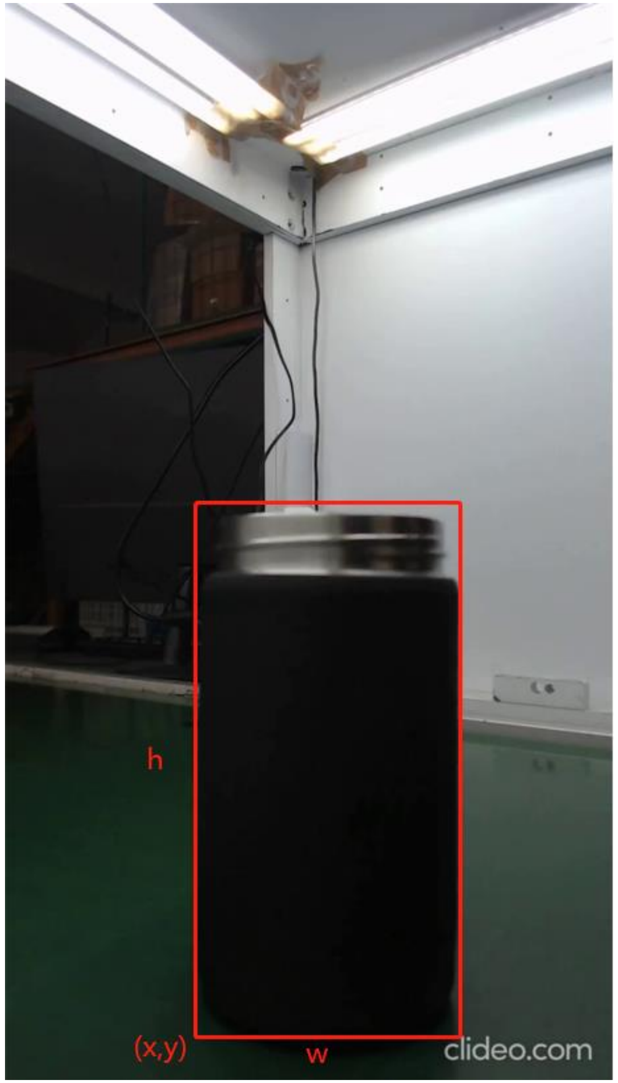
Visualized Video Output

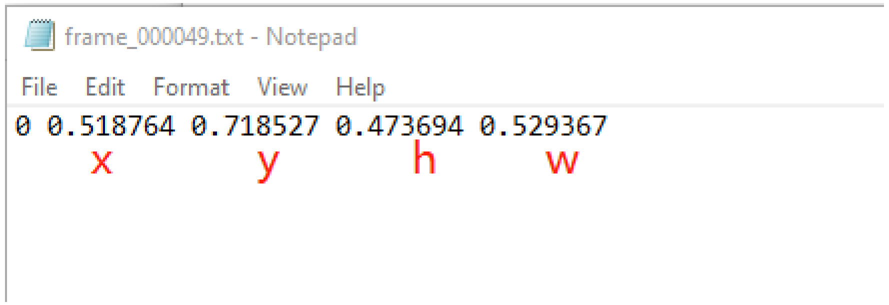
# Using YOLO to perform object detection



# Creating dataset using image annotation tool

Original Raw Data



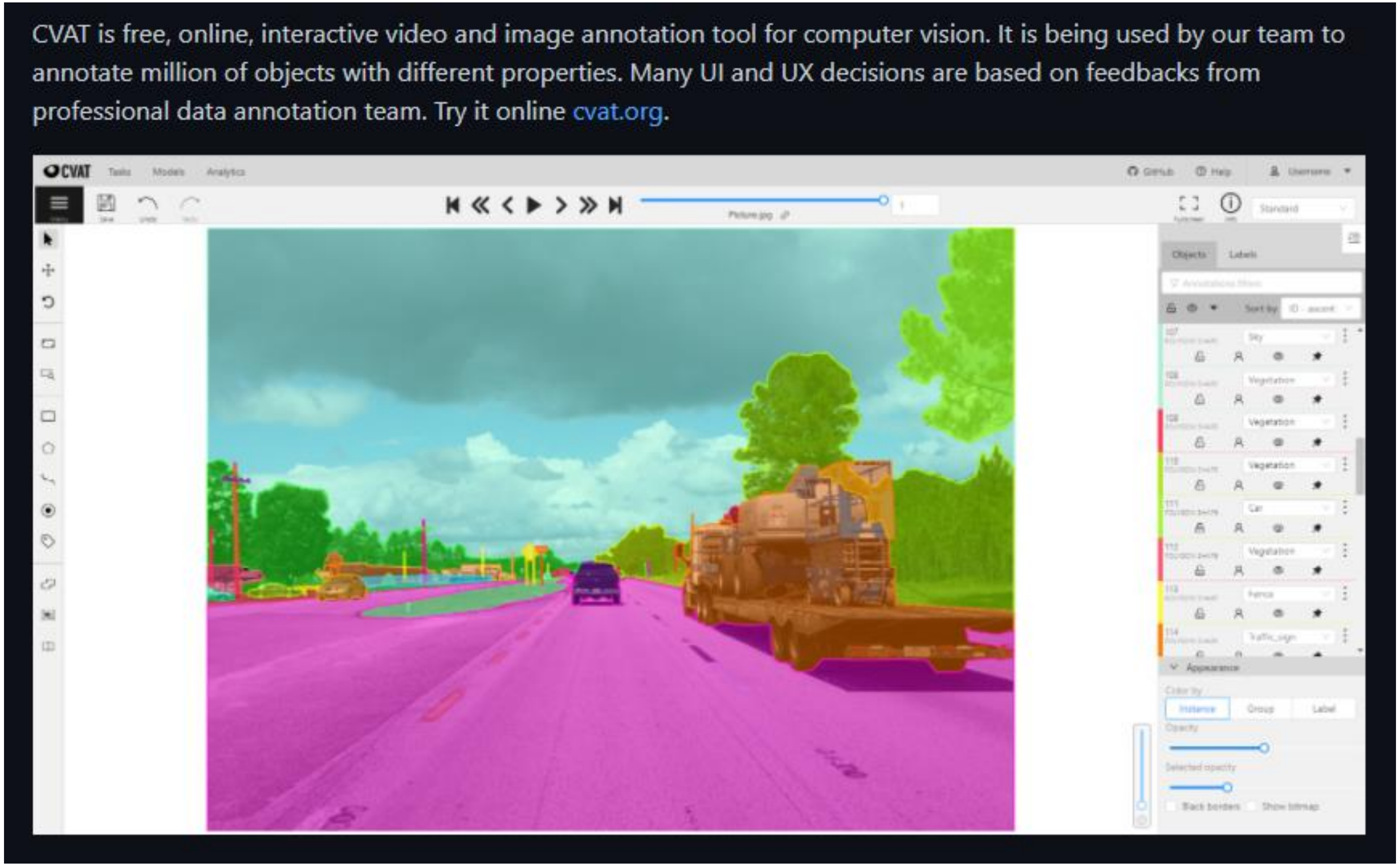


```
0 0.518764 0.718527 0.473694 0.529367
```

Images + bounding box annotations for training/testing

# Creating dataset using image annotation tool

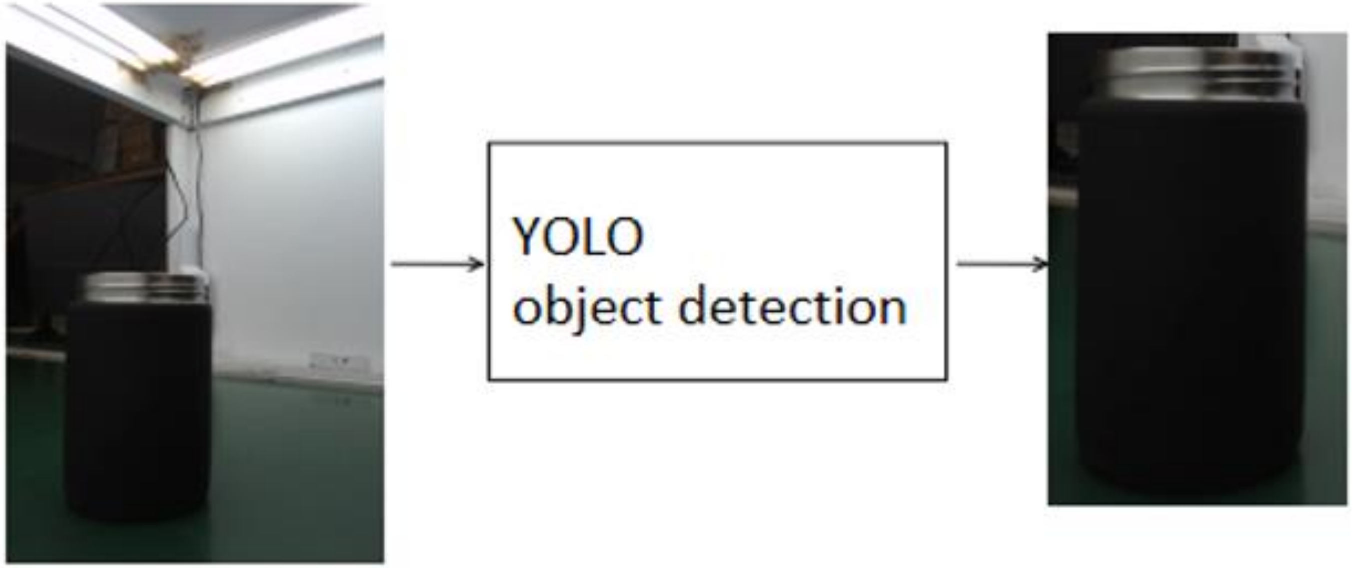
- https://github.com/openvinotoolkit/cvat



CVAT is free, online, interactive video and image annotation tool for computer vision. It is being used by our team to annotate million of objects with different properties. Many UI and UX decisions are based on feedbacks from professional data annotation team. Try it online cvat.org.

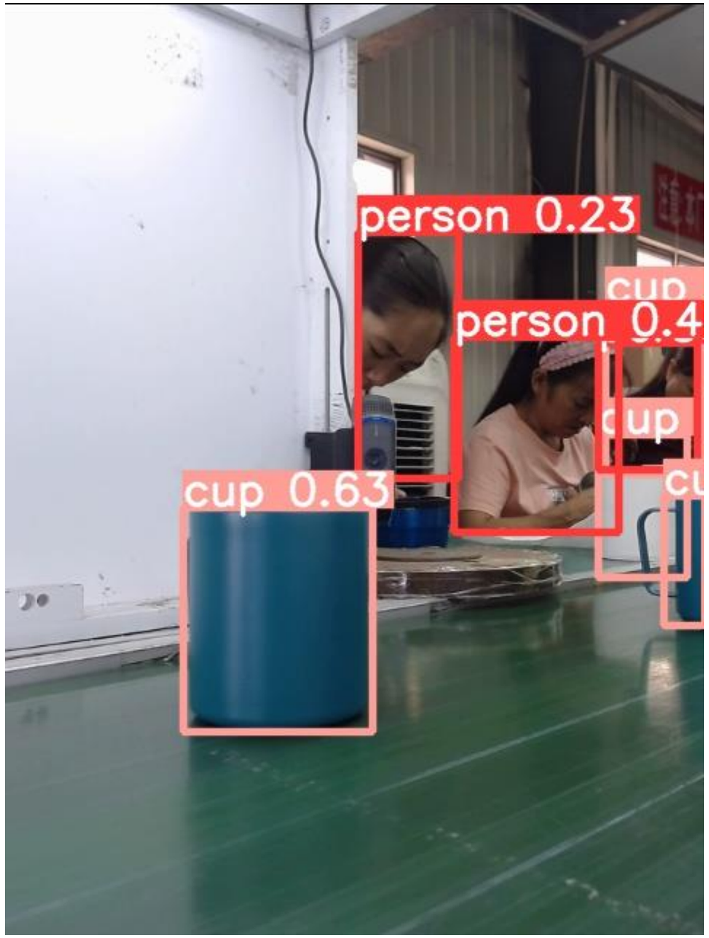
# Train an object detection model using our custom data

- https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data



YOLO
object detection

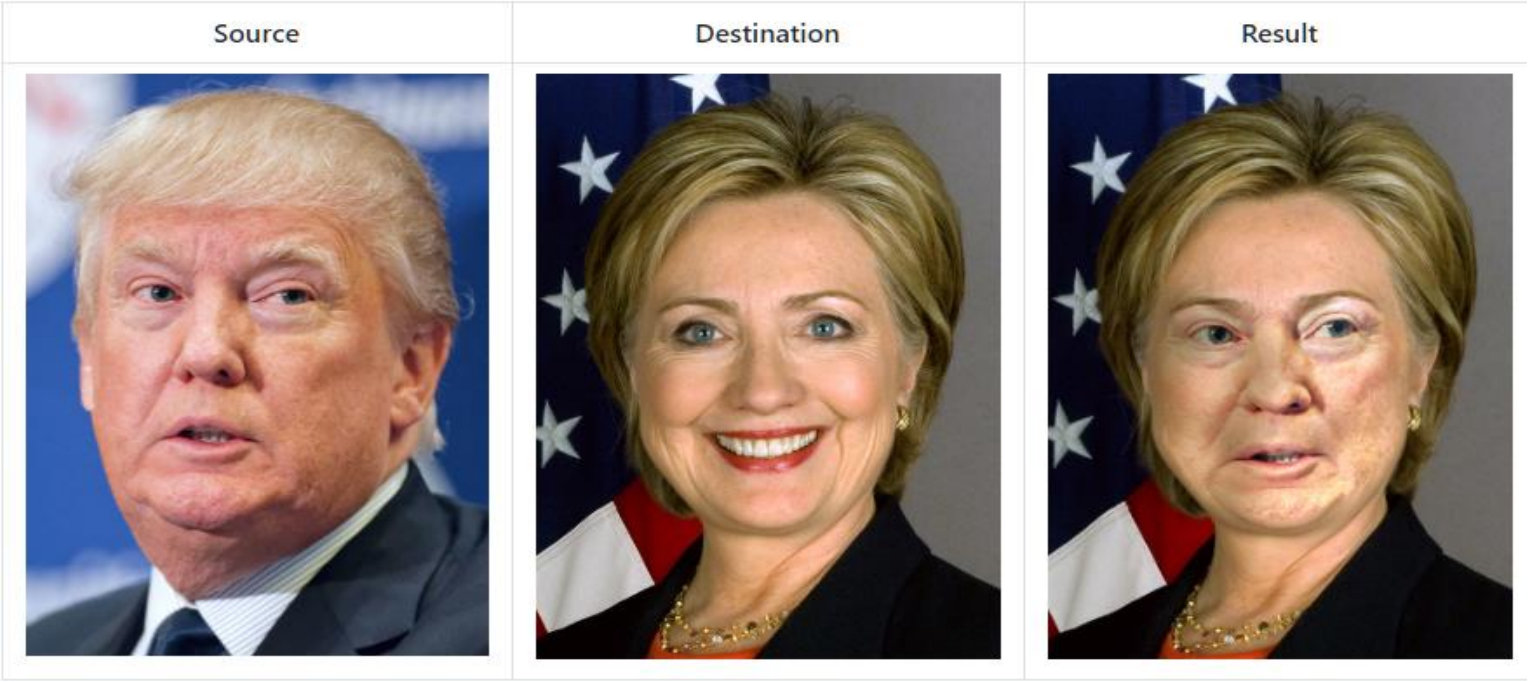
Product Video Input



# Final Assignment - Deepfake with Synthetic Face Image

- Objective: Recently, **Deepfake** and **Generative Adversarial Network (GAN)** are two hot topics in Deep Learning field. In this final project, we would like you to build an application to bridge these two techniques. We want you to use the Deepfake technique to replace the face in the destination image with any **synthetic human faces generated by the GAN**.

- Details: There are lots of different open source codes for DeepFake and Human face GAN. You can do some exploring and experiments to pick any existing implementations which you prefer to use. Remember to put proper references in your project README file.



# Final Assignment - Deepfake with Synthetic Face Image

- Extra resource: https://github.com/datamllab/awesome-deepfakes-materials
- In this project, instead of using a source face image that is from a real person, we want you to first use Human face GAN (such as https://github.com/NVlabs/stylegan2) to generate a **synthetic face**. Then use it as a source face for doing face swapping on destination image.
- You need to write an application in python and the output image can be downscaled. e.g. 256x256, 128x128.

# Overview

- How to manage profile and activities in GitHub/LinkedIn/AngelList
- Where to search and find Machine learning related jobs
- How to prepare interview (tech interview, e.g. where to practice data structure/algorithm test; system design test and organizational behavior questions)