Training Custom object detection model

on google colab Using transfer learning

# STEPS

1. Gather & label images
2. Set up environment & install necessary package
3. Generate TFRecords
4. Select & download pre-trained model
5. Configure training pipeline
6. Training
7. Export trained model

## **Running on Google Colab**

## Gather & label images

1. Gather images

* Harvest images online or your own photos

1. Label image

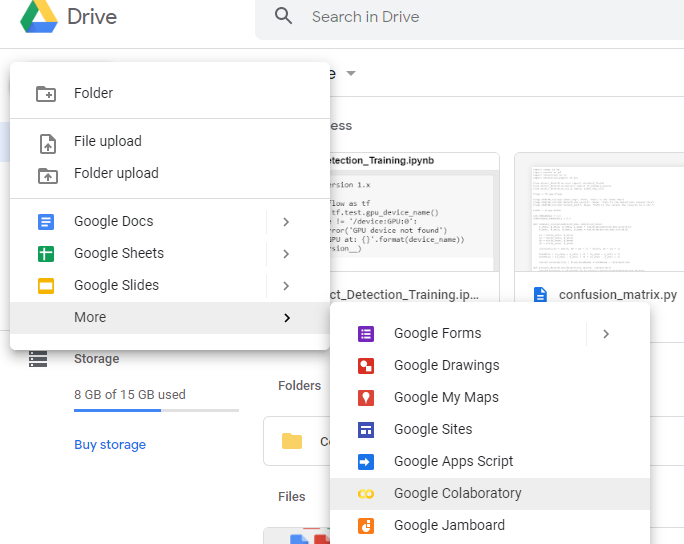
* Label the image gathered
  1. Install labelimg <https://medium.com/@sanghuynh_73086/how-to-install-labelimg-in-windows-with-anaconda-c659b27f0f>
  2. Open labelimg
  3. Start labeling



The labelimg will turn the image you labeled into an .xml file

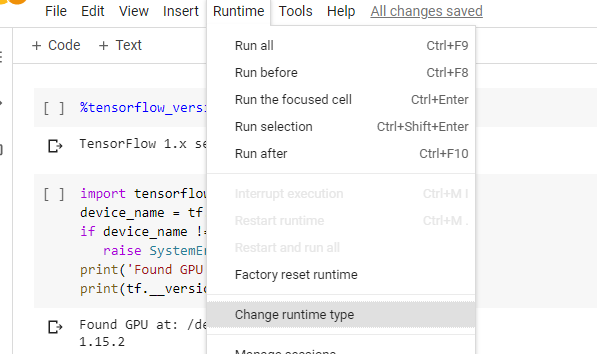
## Set up environment & install necessary package

1. Create Google Colab Notebook

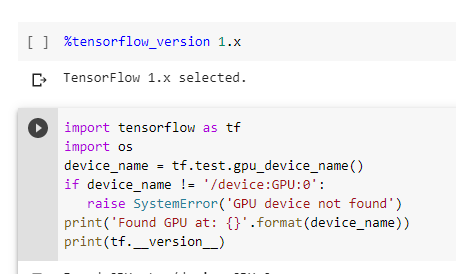


1. Go to Runtime -> Change runtime type -> Hardware Accelerator -> GPU

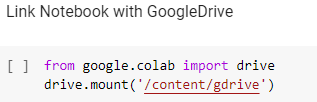
If the GPU usage limit of google colab is reached , try to use different account or create a new google account to use to GPU provided by Google.



1. Import package



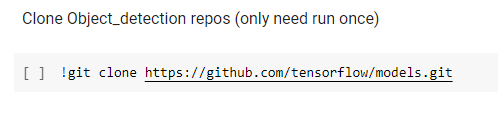
1. Mount GDrive to Google Colab Notebook



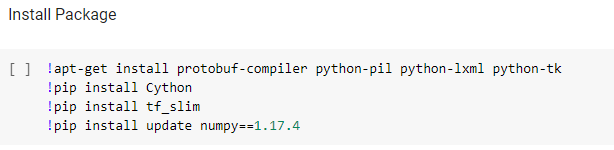
1. Create and change directory in GDrive



1. Clone Tensorflow Object\_Detection repos

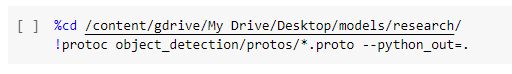


1. Install package



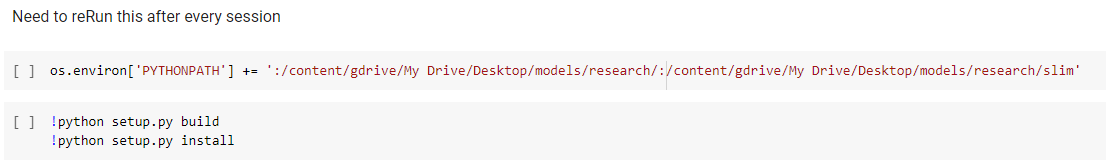
You probably need to install tensorflow-gpu if u have not installed it.

1. Extract python files from proto files in object\_detection/protos directory



1. Set environment

If the setup.py is in the slim folder , cd into slim folder and run setup.py

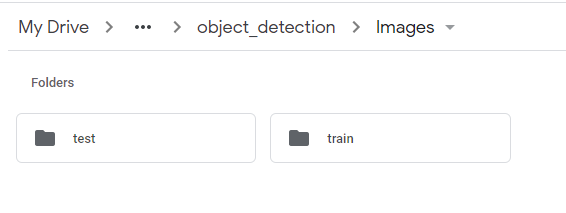


## GENERATE tfrecord

Process

* Split the image (with .xml) into train and test folder
  + Directory object\_detection/images/train & object\_detection/images/test
* Convert .xml into .csv
* Convert .csv into .record

1. Split the image (train: 80% test: 20%)



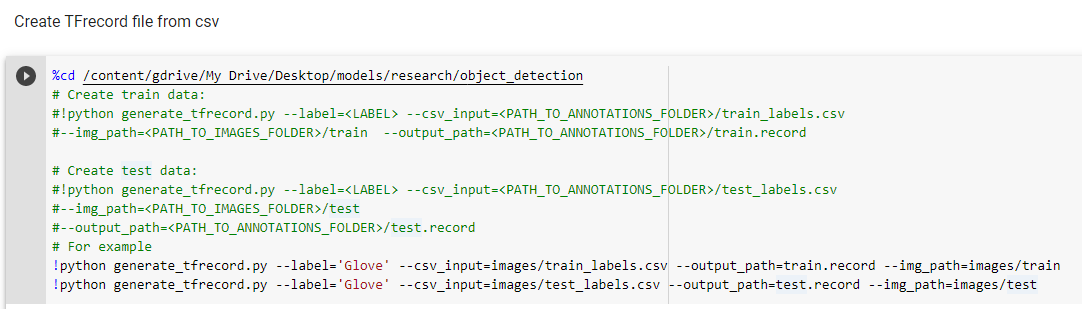
1. Convert .xml into .csv

Text

Description automatically generated with medium confidence

1. Convert .csv into .record

The generate\_tfrecord.py can be obtained at here: <https://github.com/ElectroNath/-Training-an-Object-Detection-Model-with-TensorFlow-API-using-Google-COLAB/blob/master/generate_tfrecord.py>



## Download pre-trained model

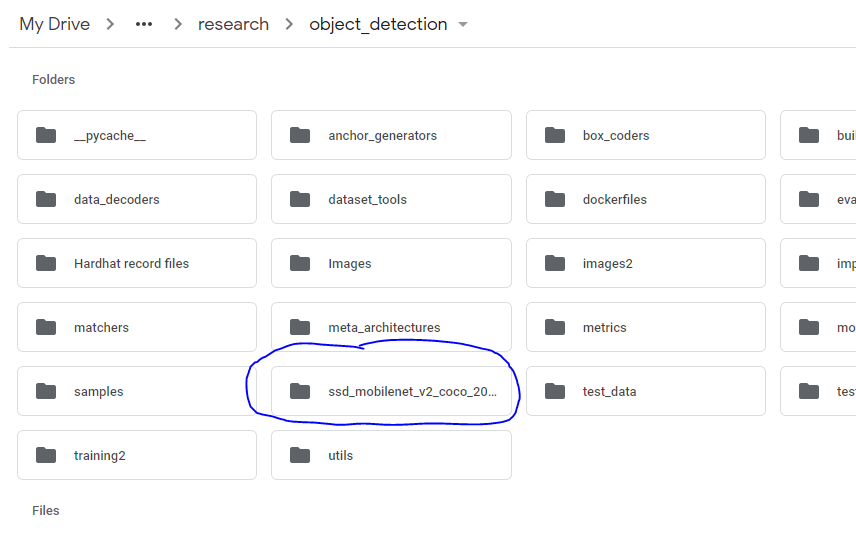
Pre-trained Model can be downloaded from this link <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md>

Table

Description automatically generated with medium confidence

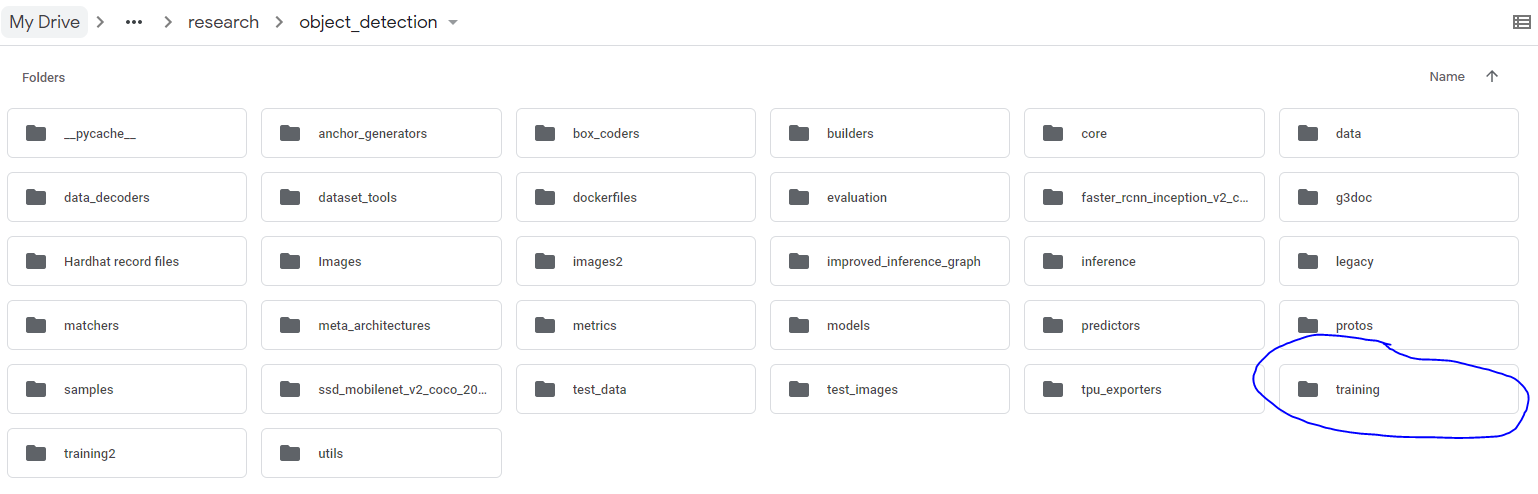
Speed(ms) is the how fast can the model runs and mAP is the accuracy of the model.

Extract and place it into object\_detection folder.

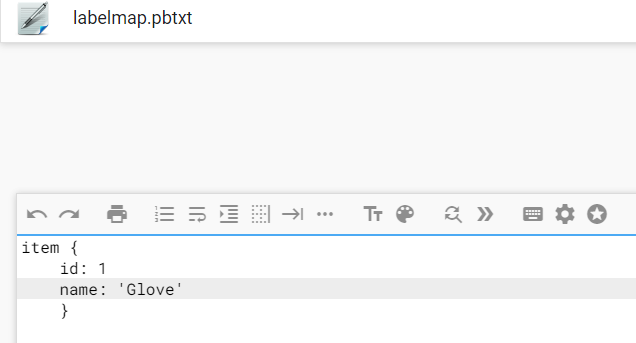


## Configure training pipeline

1. Create a folder inside object\_detection folder called training



1. Create label.pbtxt file inside training folder



1. Get the corresponding sample config file from object\_detection/samples/configs, in my case ssd\_mobilenet\_v2\_coco and place it inside the training folder.

Amend the contents inside

Line 9: change the number of classes to number of objects you want to detect (1 in our case).

(Depend on how many items do you have in the pbtxt file, if you want to test person with helmet and person without helmet , set it as 2 .)

Line 156: change fine\_tune\_checkpoint to the path of the model.ckpt file

Take Note: there are two types of checkpoint file name which are model.ckpt or ckpt-0 , depending on which is your checkpoint file name.

fine\_tune\_checkpoint:  
"/Users/<username>/projects/tensorflow/models/research/object\_detection/faster\_rcnn\_inception\_v2\_coco\_2018\_01\_28/model.ckpt"

Line 175: change input\_path to the path of the train.record file:

input\_path:  
"/Users/<username>/projects/tensorflow/models/research/object\_detection/train.record

Line 189: change input\_path to the path of the test.records file:

input\_path:  
"/Users/<username>/projects/tensorflow/models/research/object\_detection/test.record

Line 177, 191: change label\_map\_path to the path of label map file:

label\_map\_path:  
"/Users/<username>/projects/tensorflow/models/research/object\_detection/training/labelmap.pbtxt

Line 181: change num\_example to the number of images in your test folder.

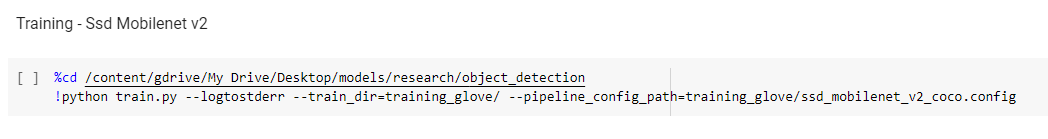
num\_examples: 10

## Training

To view training progress, open tensorboard



* Run train.py to commence training (train.py found in legacy folder)



Note that if the training only trains for a few steps, do change the batch\_size in line 141 of ssd\_mobilenet\_v2\_coco.config:

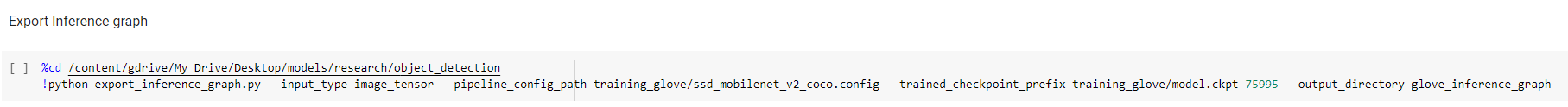
* You can also run

Text

Description automatically generated

Note that you can experiment and change the batch size value, but the confirmed batch sizes that work are 2, 4, 8, 16 because any higher may not work due to Colab’s RAM limitation

## Export inferenced graph



Once the inference graph is created, go to the output directory and save frozen\_inference\_graph.pb as shown below

Table

Description automatically generated with medium confidence

## USING MODEL WITH OPENCV

It can be achieved by following this guide <https://jeanvitor.com/tensorflow-object-detecion-opencv/>

I followed the guide using anaconda prompt

1. Create a conda environment (optional)
2. Conda install opencv-contrib-python
3. Create folder
4. Within the folder you need:
   1. frozen\_inference\_graph.pb
   2. ssd\_mobilenet\_v2\_coco.config (your config file that was used to train your model, under the “training” folder)
   3. tf\_text\_graph\_common.py
   4. tf\_text\_graph\_(model).py, tf\_text\_graph\_ssd.py in my case
5. Enter python tf\_text\_graph\_ssd.py --input=”frozen\_inference\_graph.pb” --output=”graph.pbtxt” --config=”ssd\_mobilenet\_v2\_coco.config”

**To note:** Once you get the 'graph.pbtxt' you must find & replace all “AddV2” to “Add” inside.

## References

<https://towardsdatascience.com/training-tensorflow-object-detection-api-with-custom-dataset-for-working-in-javascript-and-vue-js-6634e0f33e03>

<https://medium.com/analytics-vidhya/training-an-object-detection-model-with-tensorflow-api-using-google-colab-4f9a688d5e8b>

<https://www.youtube.com/watch?v=Rgpfk6eYxJA&t=111s>

Link to my colab notebook:

<https://colab.research.google.com/drive/1IVwXUcQQSF_695kWHQAsUFgMQ0qQsdoM?usp=sharing>

## **Running on local Computer With Tensorflow 2**

* Please make sure the python version is compatible to the TensorFlow version. For instance, python 3.9 supports TensorFlow 2.5 or later.

Graphical user interface, text, application

Description automatically generated

* Other than python and TensorFlow, CUDA and CUDNN is needed if you are using Nvidia GPU because they will speed up the process significantly.
* You can get the CUDA and CUDNN with the links:

[CUDA Toolkit Archive | NVIDIA Developer](https://developer.nvidia.com/cuda-toolkit-archive)

[cuDNN Archive | NVIDIA Developer](https://developer.nvidia.com/rdp/cudnn-archive)

* Please make the version of CUDA and CUDNN are compatible.
* After downloading CUDA and CUDNN , copy the files in the CUDNN into CUDA folder. You can go to your CUDA folder with this path：C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6
* If the files are in the CUDNN bin folder, copy them and paste into CUDA bin folder. Repeat the same process to ‘include’ and ‘lib’ folder.

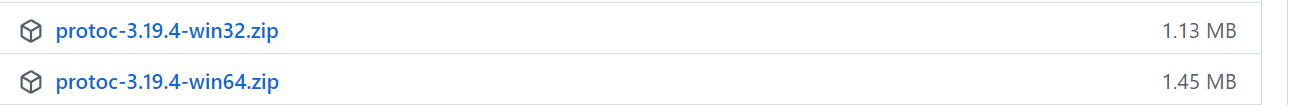
A picture containing text, electronics, black, screenshot

Description automatically generated

## Install protoc

* TensorFlow graphs are represented in protocol buffer text format. We need this library in order to use the TensorFlow Object Detection Library. Otherwise , there would be tons of errors coming out.

[Releases · protocolbuffers/protobuf (github.com)](https://github.com/protocolbuffers/protobuf/releases)



* Go to the bottom of the link above , click on the zip file that we need.

NOTE : if your laptop’s CPU is 32-bit version , click and download on the win32 version zip file. Otherwise , download the win64 zip file.

* Unzip and add the folder containing the files to your environment path
  + CTRL+S to seach ‘edit the system environment variable’
  + Select environment variables
  + Select path
  + Select New and fill in the path.

## How to check 32-bit or 64-bit Windows

Graphical user interface, application

Description automatically generated 

* Press CTRL+S and search ‘ System Information’ , find ‘System Type’ on the left column.

## Create Virtual Environment

* Aim to create a virtual environment is to create an isolated environment with its own dependencies so that the dependencies would not affect other projects. Imagine virtual environment is an empty room , you create and enter the empty room and adding whatever you need for the project. Hence , the things in this room would not affect other rooms.
* Enter the command below to create your virtual environment.

NOTE : the commands are entered in the terminal.

python -m venv env

* for my case : python -m venv tfod
* Env – this is the name of your virtual environment

## Activate virtual environment

* Copy the path to the ‘activate’ file . For example , Graphical user interface, text

  Description automatically generated
* And enter the command ‘source C:/tfod/Scripts/activate’ to activate virtual environment. You can see there is the virtual environment name with bracket (env) in the terminal.



Graphical user interface

Description automatically generated with medium confidence

## Installation of dependencies

Enter the commands below the install required dependencies in your virtual environment. To check what is the current dependencies in your virtual environment , you can enter ‘pip list’ to check.

python -m pip install --upgrade pip

pip install ipykernel

python -m ipykernel install --user --name=”name of your virtual environment”

pip install tensorflow

## Install Object Detection API

This process is to install the object detection API that we need to train our model.

cd models/research

protoc object\_detection/protos/\*.proto --python\_out**=**.

copy object\_detection\\packages\\tf2\\setup.py setup.py

cd models/research/slim && pip install -e .

python models/research/object\_detection/builders/model\_builder\_tf2\_test.py

Create Label MapText

Description automatically generated

labels = [{'name':'person with helmet', 'id':1},{'name':'person without helmet', 'id':2}]

with open(filename 'w') as f:

    for label in labels:

        f.write('item { \n')

        f.write('\tname:\'{}\'\n'.format(label['name']))

        f.write('\tid:{}\n'.format(label['id']))

        f.write('}\n')

## Create TF Record

The above shows using the csv file to create TF record . Here shows another approach to create TF Record by using images and xml files.

[GenerateTFRecord/generate\_tfrecord.py at main · nicknochnack/GenerateTFRecord (github.com)](https://github.com/nicknochnack/GenerateTFRecord/blob/main/generate_tfrecord.py)

## Config File (Same step as above)

## Training the Model

Python models/research/object\_detection/model\_main\_tf2.py --model\_dir=models/my\_ssd\_mobnet --pipeline\_config\_path= models/my\_ssd\_mobnet/pipeline.config --num\_train\_steps=3000

## Evaluating the Model

python models/research/object\_detection/model\_main\_tf2.py --model\_dir= models/my\_ssd\_mobnet --pipeline\_config\_path= models/my\_ssd\_mobnet/pipeline.config --checkpoint\_dir=x` models/my\_ssd\_mobnet

Testing out of your model with selected image

Text

Description automatically generated

Graphical user interface, application

Description automatically generated with medium confidence

img = cv2.imread(IMAGE\_PATH)

image\_np = np.array(img)

input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)

detections = detect\_fn(input\_tensor)

num\_detections = int(detections.pop('num\_detections'))

detections = {key: value[0, :num\_detections].numpy()

              for key, value in detections.items()}

detections['num\_detections'] = num\_detections

# detection\_classes should be ints.

detections['detection\_classes'] = detections['detection\_classes'].astype(np.int64)

label\_id\_offset = 1

image\_np\_with\_detections = image\_np.copy()

viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

            image\_np\_with\_detections,

            detections['detection\_boxes'],

            detections['detection\_classes']+label\_id\_offset,

            detections['detection\_scores'],

            category\_index,

            use\_normalized\_coordinates=True,

            max\_boxes\_to\_draw=5,

            min\_score\_thresh=.8,

            agnostic\_mode=False)

plt.imshow(cv2.cvtColor(image\_np\_with\_detections, cv2.COLOR\_BGR2RGB))

plt.show()

## Real Time Detection with your model

cap = cv2.VideoCapture(0)

width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

while cap.isOpened():

    ret, frame = cap.read()

    image\_np = np.array(frame)

    input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)

    detections = detect\_fn(input\_tensor)

    num\_detections = int(detections.pop('num\_detections'))

    detections = {key: value[0, :num\_detections].numpy()

                  for key, value in detections.items()}

    detections['num\_detections'] = num\_detections

    # detection\_classes should be ints.

    detections['detection\_classes'] = detections['detection\_classes'].astype(np.int64)

    label\_id\_offset = 1

    image\_np\_with\_detections = image\_np.copy()

    viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

                image\_np\_with\_detections,

                detections['detection\_boxes'],

                detections['detection\_classes']+label\_id\_offset,

                detections['detection\_scores'],

                category\_index,

                use\_normalized\_coordinates=True,

                max\_boxes\_to\_draw=5,

                min\_score\_thresh=.8,

                agnostic\_mode=False)

    cv2.imshow('object detection',  cv2.resize(image\_np\_with\_detections, (800, 600)))

    if cv2.waitKey(10) & 0xFF == ord('q'):

        cap.release()

        cv2.destroyAllWindows()

        break

## Performance Tuning

* Add more training images with correct labels and fit boxes
* Increase the training times

## References

[(7) Tensorflow Object Detection in 5 Hours with Python | Full Course with 3 Projects - YouTube](https://www.youtube.com/watch?v=yqkISICHH-U&t=13204s&ab_channel=NicholasRenotte) [(7) Install Tensorflow Object Detection From Scratch in 5 Steps | Python Deep Learning - YouTube](https://www.youtube.com/watch?v=dZh_ps8gKgs&t=1321s&ab_channel=NicholasRenotte)

[TFRecords Explained. Working with TFRecords with an… | by Girija Shankar Behera | Towards Data Science](https://towardsdatascience.com/tfrecords-explained-24b8f2133282)

[nicknochnack/TFODCourse (github.com)](https://github.com/nicknochnack/TFODCourse)