



TKO_7096-3001 :Computer Vision & Sensor Fusion

Lecture 6-2

Fahimeh Farahnakian

University of Turku, Finland

<https://users.utu.fi/fahfar/>

Email: fahfar@utu.fi



TensorFlow Object Detection API

https://github.com/tensorflow/models/tree/master/research/object_detection

- **TensorFlow Detection API** is an **open source** framework built on top of TensorFlow that helps build, train and deploy object detection models.
- Instead of training your own model from scratch, you can build on existing models and fine-tune them for your own purpose without requiring as much computing power.

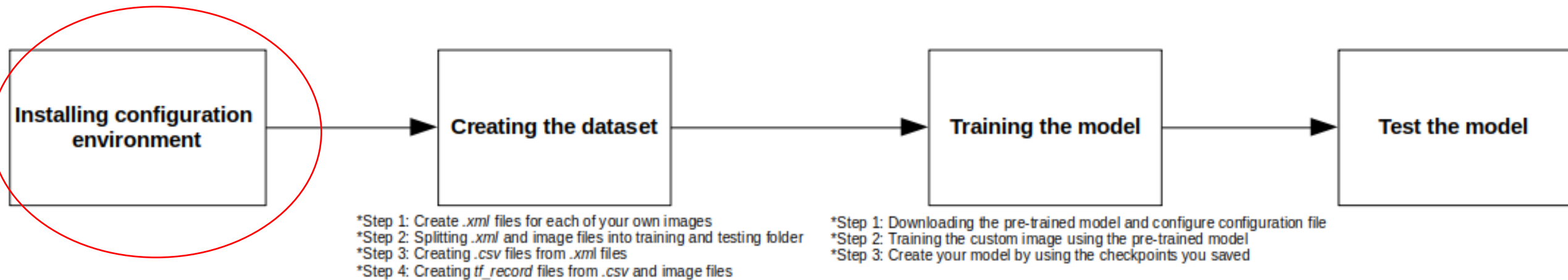


Detecting objects in images and videos

For whom?

Very powerful tool that can quickly enable anyone with less or no knowledge on machine learning to build and deploy powerful object detection model.

Tensorflow Object Detection API flowchart



Step 1: Libraries Required

Tensorflow object detection API depends on the following libraries:

- ✓ Protobuf
- ✓ Python
- ✓ Pillow
- ✓ Lxml
- ✓ Tf Slim
- ✓ Matplotlib
- ✓ Tensorflow
- ✓ Cocoapi
- ✓ Jupyter Notebook
- ✓ Cython

Step 1: Installation steps:

1. First step is to go to the following link: <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html>
2. Go the installation process step by step according to your environment

```
Anaconda Prompt - conda create -n object_detection_test pip python=3.9

(base) C:\Users\your user>conda create -n object_detection_test pip python=3.9
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test

added / updated specs:
- pip
- python=3.9

The following packages will be downloaded:

package                        build                                size  channel
-----
ca-certificates-2024.2.2      h56e8100_0                          152 KB  conda-forge
libsqlite-3.45.1              hcfcfb64_0                          850 KB  conda-forge
openssl-3.2.1                 hcfcfb64_0                          7.8 MB  conda-forge
pip-24.0                       pyhd8ed1ab_0                        1.3 MB  conda-forge
tzdata-2024a                  h0c530f3_0                          117 KB  conda-forge
vc-14.3                        hcf57466_18                         17 KB  conda-forge
vs2015_runtime-14.38.33130    hcb4865c_18                         17 KB  conda-forge
wheel-0.42.0                  pyhd8ed1ab_0                        56 KB  conda-forge

Total: 10.4 MB
```

Step 1

```
Anaconda Prompt - conda create -n object_detection_test pip python=3.9

-----
Total: 10.4 MB

The following NEW packages will be INSTALLED:

bzip2                conda-forge/win-64::bzip2-1.0.8-hcfcfb64_5
ca-certificates      conda-forge/win-64::ca-certificates-2024.2.2-h56e8100_0
libffi               conda-forge/win-64::libffi-3.4.2-h8ffe710_5
libsqlite            conda-forge/win-64::libsqlite-3.45.1-hcfcfb64_0
libzlib              conda-forge/win-64::libzlib-1.2.13-hcfcfb64_5
openssl              conda-forge/win-64::openssl-3.2.1-hcfcfb64_0
pip                  conda-forge/noarch::pip-24.0-pyhd8ed1ab_0
python               conda-forge/win-64::python-3.9.18-h4de0772_1_cpython
setuptools            conda-forge/noarch::setuptools-69.0.3-pyhd8ed1ab_0
tk                   conda-forge/win-64::tk-8.6.13-h5226925_1
tzdata               conda-forge/noarch::tzdata-2024a-h0c530f3_0
ucrt                  conda-forge/win-64::ucrt-10.0.22621.0-h57928b3_0
vc                   conda-forge/win-64::vc-14.3-hcf57466_18
vc14_runtime         conda-forge/win-64::vc14_runtime-14.38.33130-h82b7239_18
vs2015_runtime       conda-forge/win-64::vs2015_runtime-14.38.33130-hcb4865c_18
wheel                conda-forge/noarch::wheel-0.42.0-pyhd8ed1ab_0
xz                   conda-forge/win-64::xz-5.2.6-h8d14728_0

Proceed ([y]/n)? y_
```

Step 2

Step 1: Installation steps:

Step 3

```
Anaconda Prompt

bzip2 conda-forge/win-64::bzip2-1.0.8-hcfcfb64_5
ca-certificates conda-forge/win-64::ca-certificates-2024.2.2-h56e8100_0
libffi conda-forge/win-64::libffi-3.4.2-h8ffe710_5
libsqlite conda-forge/win-64::libsqlite-3.45.1-hcfcfb64_0
libzlib conda-forge/win-64::libzlib-1.2.13-hcfcfb64_5
openssl conda-forge/win-64::openssl-3.2.1-hcfcfb64_0
pip conda-forge/noarch::pip-24.0-pyhd8ed1ab_0
python conda-forge/win-64::python-3.9.18-h4de0772_1_cpython
setuptools conda-forge/noarch::setuptools-69.0.3-pyhd8ed1ab_0
tk conda-forge/win-64::tk-8.6.13-h5226925_1
tzdata conda-forge/noarch::tzdata-2024a-h0c530f3_0
ucrt conda-forge/win-64::ucrt-10.0.22621.0-h57928b3_0
vc conda-forge/win-64::vc-14.3-hcf57466_18
vc14_runtime conda-forge/win-64::vc14_runtime-14.38.33130-h82b7239_18
vs2015_runtime conda-forge/win-64::vs2015_runtime-14.38.33130-hcb4865c_18
wheel conda-forge/noarch::wheel-0.42.0-pyhd8ed1ab_0
xz conda-forge/win-64::xz-5.2.6-h8d14728_0

Proceed ([y]/n)? y

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate object_detection_test
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#

(base) C:\Users\your user >conda activate object_detection_test
(object_detection_test) C:\Users\your user >pip install --ignore-installed --upgrade tensorflow==2.5.0
Collecting tensorflow==2.5.0
  Using cached tensorflow-2.5.0-cp39-cp39-win_amd64.whl (422.6 MB)
Collecting numpy~=1.19.2 (from tensorflow==2.5.0)
  Using cached numpy-1.19.5-cp39-cp39-win_amd64.whl (13.3 MB)
Collecting absl-py~=0.10 (from tensorflow==2.5.0)
  Using cached absl_py-0.15.0-py3-none-any.whl (132 kB)
Collecting astunparse~=1.6.3 (from tensorflow==2.5.0)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
```

Conda activate object_detection_env

Step 4

```
Anaconda Prompt - pip install --ignore-installed --upgrade tensorflow==2.5.0

tk conda-forge/win-64::tk-8.6.13-h5226925_1
tzdata conda-forge/noarch::tzdata-2024a-h0c530f3_0
ucrt conda-forge/win-64::ucrt-10.0.22621.0-h57928b3_0
vc conda-forge/win-64::vc-14.3-hcf57466_18
vc14_runtime conda-forge/win-64::vc14_runtime-14.38.33130-h82b7239_18
vs2015_runtime conda-forge/win-64::vs2015_runtime-14.38.33130-hcb4865c_18
wheel conda-forge/noarch::wheel-0.42.0-pyhd8ed1ab_0
xz conda-forge/win-64::xz-5.2.6-h8d14728_0

Proceed ([y]/n)? y

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate object_detection_test
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#

(base) C:\Users\your user >conda activate object_detection_test
(object_detection_test) C:\Users\your user >pip install --ignore-installed --upgrade tensorflow==2.5.0
Collecting tensorflow==2.5.0
  Using cached tensorflow-2.5.0-cp39-cp39-win_amd64.whl (422.6 MB)
Collecting numpy~=1.19.2 (from tensorflow==2.5.0)
  Using cached numpy-1.19.5-cp39-cp39-win_amd64.whl (13.3 MB)
Collecting absl-py~=0.10 (from tensorflow==2.5.0)
  Using cached absl_py-0.15.0-py3-none-any.whl (132 kB)
Collecting astunparse~=1.6.3 (from tensorflow==2.5.0)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
```

pip install --ignore-installed --upgrade tensorflow==2.5.0

Step 1: Installation steps:

Downloading the TensorFlow Model Garden

Create a new folder under a path of your choice and name it `TensorFlow` (e.g.

```
C:\Users\sglv\Documents\TensorFlow
```

).

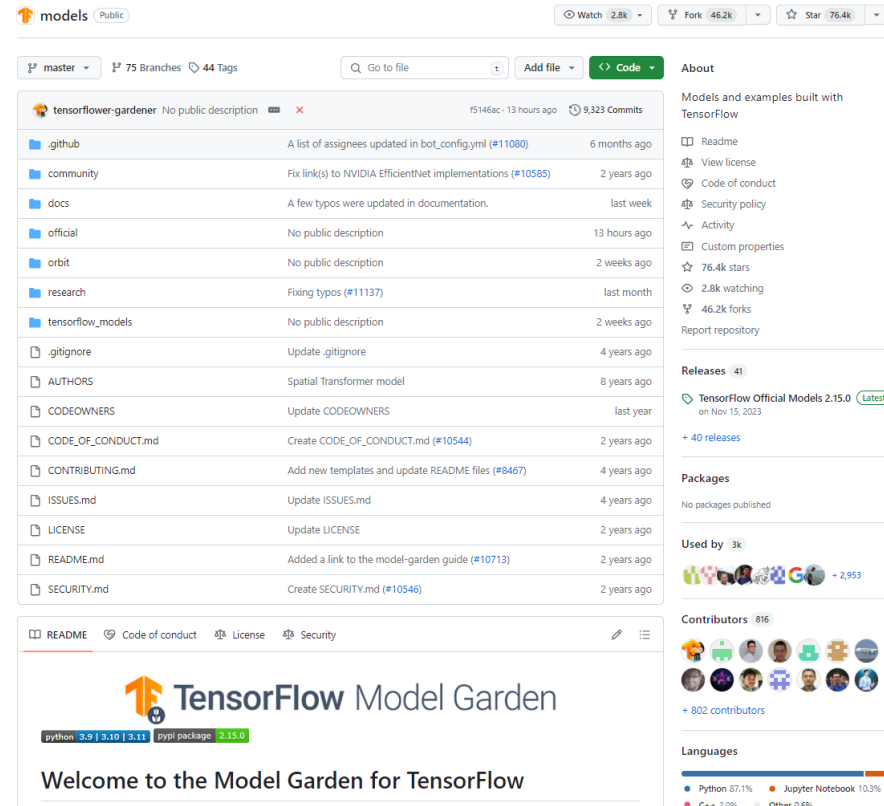
From your *Terminal* `cd` into the `TensorFlow` directory.

To download the models you can either use [Git](#) to clone the [TensorFlow Models repository](#) inside the `TensorFlow` folder, or you can simply download it as a [ZIP](#) and extract its contents inside the `TensorFlow` folder. To keep things consistent, in the latter case you will have to rename the extracted folder `models-master` to `models`.

You should now have a single folder named `models` under your `TensorFlow` folder, which contains another 4 folders as such:

```
TensorFlow/  
└─ models/  
  ├── community/  
  ├── official/  
  ├── orbit/  
  ├── research/  
  └─ ...
```

Step 5



Link:

<https://github.com/tensorflow/models>

Step 1: Installation steps:

Jan 10
sbenzaquen
v25.2
a9b006b
Compare

Protocol Buffers v25.2 Latest

Announcements

- Protobuf News may include additional announcements or pre-announcements for upcoming changes.

C++

- Only substitute prefixes during installation setup. (05ad652)
- Register a shutdown delete for C++ feature defaults (3d5c709)

Assets 14

protobuf-25.2.tar.gz	5.61 MB	Jan 10
protobuf-25.2.zip	7.22 MB	Jan 10
protoc-25.2-linux-aarch_64.zip	2.93 MB	Jan 10
protoc-25.2-linux-ppc64.zip	3.21 MB	Jan 10
protoc-25.2-linux-s390_64.zip	3.79 MB	Jan 10
protoc-25.2-linux-x86_32.zip	3.22 MB	Jan 10
protoc-25.2-linux-x86_64.zip	2.96 MB	Jan 10
protoc-25.2-osx-aarch_64.zip	2.11 MB	Jan 10
protoc-25.2-osx-universal_binary.zip	4.23 MB	Jan 10
protoc-25.2-osx-x86_64.zip	2.13 MB	Jan 10
Source code (zip)		Jan 8
Source code (tar.gz)		Jan 8

Show all 14 assets

5

28

1

2

33 people reacted

Step 6
Link:

<https://github.com/protocolbuffers/protobuf/releases>

Windows PowerShell (x86)

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\ user > d:
PS D:\> cd .\Object_detection_test\
PS D:\Object_detection_test> cd .\models\
PS D:\Object_detection_test\models> cd .\research\
PS D:\Object_detection_test\models\research> protoc object_detection/protos/*.proto --python_out=.
PS D:\Object_detection_test\models\research>

Step 7

Step 1: Installation steps:

Step 8

```
Anaconda Prompt
(object_detection_test) D:\Object_detection_test\models\research>copy object_detection/packages/tf2/setup.py .
(object_detection_test) D:\Object_detection_test\models\research>python -m pip install .
Processing d:\Object_detection_test\models\research
  Preparing metadata (setup.py) ... done
Collecting avro-python3 (from object_detection==0.1)
  Using cached avro_python3-1.10.2-py3-none-any.whl
Collecting apache-beam (from object_detection==0.1)
  Using cached apache_beam-2.53.0-cp39-cp39-win_amd64.whl.metadata (6.7 kB)
Collecting pillow (from object_detection==0.1)
  Using cached pillow-10.2.0-cp39-cp39-win_amd64.whl.metadata (9.9 kB)
Collecting lxml (from object_detection==0.1)
  Using cached lxml-5.1.0-cp39-cp39-win_amd64.whl.metadata (3.6 kB)
Collecting matplotlib (from object_detection==0.1)
  Using cached matplotlib-3.8.2-cp39-cp39-win_amd64.whl.metadata (5.9 kB)
Requirement already satisfied: Cython in c:\users\your user\appdata\local\anaconda3\envs\object_detection_test\lib\site-packages (from object_detection==0.1) (3.0.8)
Collecting contextlib2 (from object_detection==0.1)
  Using cached contextlib2-21.6.0-py2.py3-none-any.whl (13 kB)
Collecting tf-slim (from object_detection==0.1)
  Using cached tf_slim-1.1.0-py2.py3-none-any.whl (352 kB)
Requirement already satisfied: six in c:\users\your user\appdata\local\anaconda3\envs\object_detection_test\lib\site-packages (from object_detection==0.1) (1.15.0)
Requirement already satisfied: pycocotools in c:\users\your user\appdata\local\anaconda3\envs\object_detection_test\lib\site-packages (from object_detection==0.1) (2.0)
Collecting lvis (from object_detection==0.1)
  Using cached lvis-0.5.3-py3-none-any.whl (14 kB)
Collecting scipy (from object_detection==0.1)
```

copy object_detection/packages/tf2/setup.py .
python -m pip install .

Step 9

```
Anaconda Prompt
(object_detection_test) C:\>your user \>javsh>pip install cython
Collecting cython
  Using cached Cython-3.0.8-cp39-cp39-win_amd64.whl.metadata (3.2 kB)
Using cached Cython-3.0.8-cp39-cp39-win_amd64.whl (2.8 MB)
Installing collected packages: cython
Successfully installed cython-3.0.8

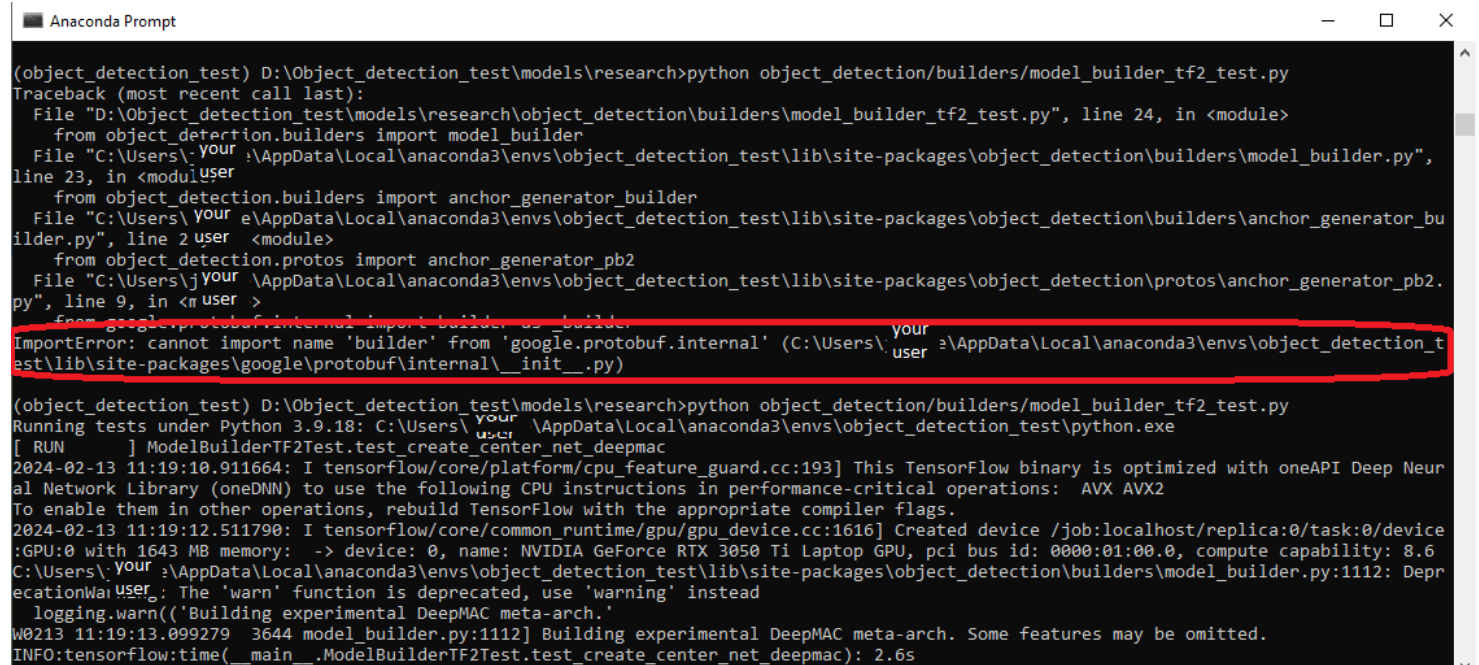
(object_detection_test) C:\Users\your user >pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
Collecting git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
  Cloning https://github.com/philferriere/cocoapi.git to c:\users\your user\appdata\local\temp\pip-req-build-ggql your user
  Running command git clone --filter=blob:none --quiet https://github.com/philferriere/cocoapi.git 'C:\Users\your user\appdata\local\temp\pip-req-build-ggql'
  Resolved https://github.com/philferriere/cocoapi.git to commit 2929bd2ef6b451054755dfd7ceb09278f935f7ad
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: pycocotools
  Building wheel for pycocotools (setup.py) ... done
  Created wheel for pycocotools: filename=pycocotools-2.0-cp39-cp39-win_amd64.whl size=82943 sha256=0e2aa855add9f40573b51074f7921bd4dbbd3e77ae38c6c6c83f5
  Stored in directory: C:\Users\your user\appdata\local\temp\pip-ephem-wheel-cache-3ewbshv5\wheels\8d\25\59\5840f59a3c30bac7223e8d0f1dcf19ac4d3c249d7659d97e
Successfully built pycocotools
Installing collected packages: pycocotools
Successfully installed pycocotools-2.0

(object_detection_test) C:\Users\your user >
```

pip install cython
pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI

Step 1: Installation steps:

Possible complication



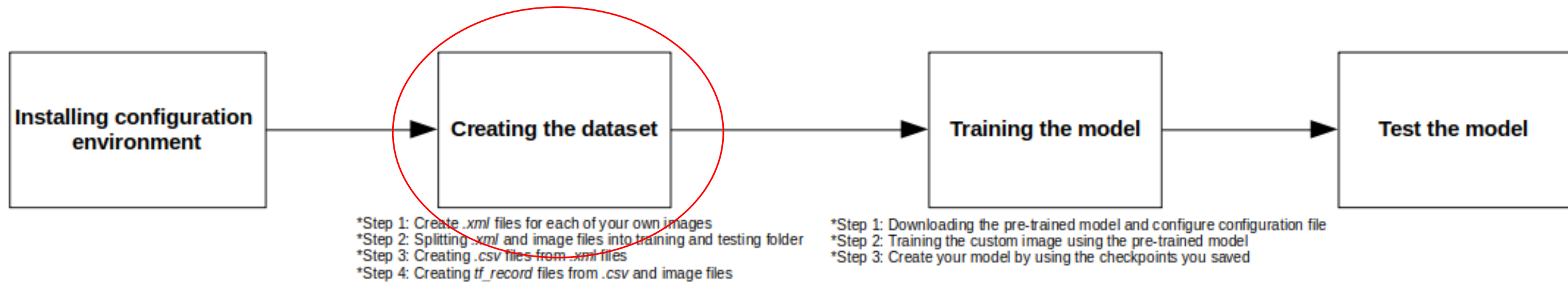
```
(object_detection_test) D:\Object_detection_test\models\research>python object_detection/builders/model_builder_tf2_test.py
Traceback (most recent call last):
  File "D:\Object_detection_test\models\research\object_detection\builders\model_builder_tf2_test.py", line 24, in <module>
    from object_detection.builders import model_builder
  File "C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test\lib\site-packages\object_detection\builders\model_builder.py",
line 23, in <module>
    from object_detection.builders import anchor_generator_builder
  File "C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test\lib\site-packages\object_detection\builders\anchor_generator_bu
ilder.py", line 23, in <module>
    from object_detection.protos import anchor_generator_pb2
  File "C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test\lib\site-packages\object_detection\protos\anchor_generator_pb2.
py", line 9, in <module>
    from google.protobuf.internal import builder as _builder
ImportError: cannot import name 'builder' from 'google.protobuf.internal' (C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_t
est\lib\site-packages\google\protobuf\internal\__init__.py)

(object_detection_test) D:\Object_detection_test\models\research>python object_detection/builders/model_builder_tf2_test.py
Running tests under Python 3.9.18: C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test\python.exe
[ RUN      ] ModelBuilderTF2Test.test_create_center_net_deepmac
2024-02-13 11:19:10.911664: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neur
al Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-02-13 11:19:12.511790: I tensorflow/core/common/runtime/gpu/gpu_device.cc:1616] Created device /job:localhost/replica:0/task:0/device
:GPU:0 with 1643 MB memory: -> device: 0, name: NVIDIA GeForce RTX 3050 Ti Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.6
C:\Users\your user\AppData\Local\anaconda3\envs\object_detection_test\lib\site-packages\object_detection\builders\model_builder.py:1112: Depr
ecationWarning: The 'warn' function is deprecated, use 'warning' instead
  logging.warn(('Building experimental DeepMAC meta-arch.'
W0213 11:19:13.099279 3644 model_builder.py:1112] Building experimental DeepMAC meta-arch. Some features may be omitted.
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_create_center_net_deepmac): 2.6s
```

Solution:

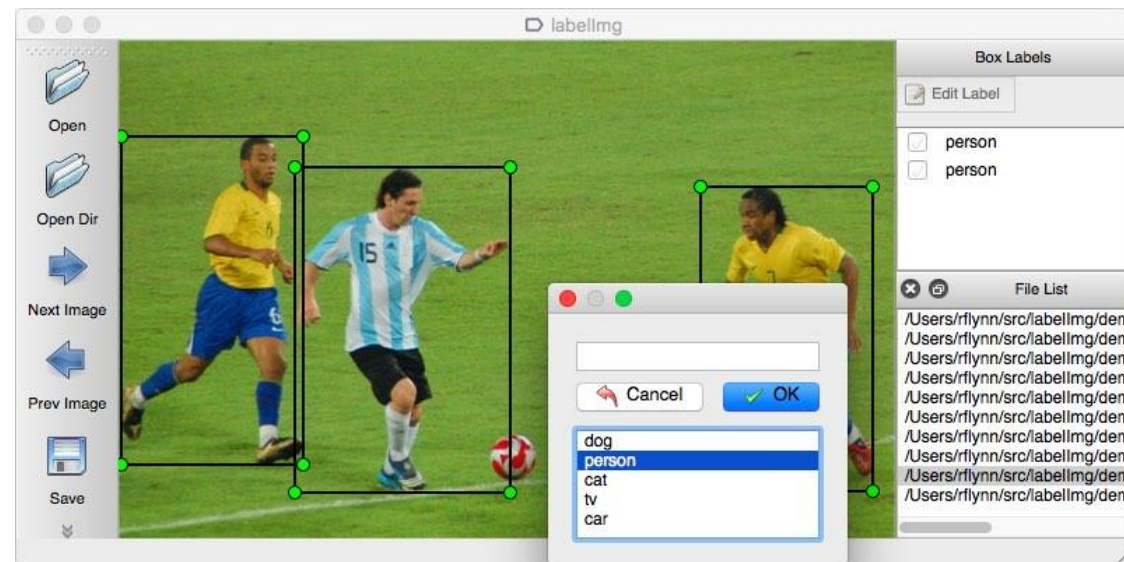
- Copy the builder.py file from moodle to the location inside the red box

Tensorflow Object Detection API flowchart



Step2: Creating the dataset (1/3)

1. **Gather a dataset:** split data into training and test dataset
2. **Label your images:** You need the height, width and class of each object to train your object detection model. This includes the associated xmin, xmax, ymin, and ymax bounding boxes.
 - To help you create these labels, you can use an image labelling software like [LabelImg](#), an open source program that saves an XML label for each image. You can then convert them into a CSV table for training.
 - LabelImg provides a user-friendly GUI.



Step2: Creating the dataset (2/3)

3. Create Label Map (.pbtxt)

Classes need to be listed in the label map. The label map should contain only two items like the following for dog and cat dataset:

```
item {  
  id: 1  
  name: 'Dog'  
  
  id: 2  
  name: 'Cat'  
}
```

➤ Note that id must start from 1, because 0 is a reserved id.

4. Convert XML to CSV file(.csv)

- We have all images and their bounding boxes are in XML format when they are labelled with [LabelImg](#) tool.
- You can use this [link](#) to create XML files to CSV which contains all the XML files and their bounding box co-ordinates to single CSV file which is input for creating TFrecords.

Step2: Creating the dataset (3/3)

5. Create TFRecord (.record)

- TFRecord is an important data format designed for Tensorflow. (Read more about it [here](#)). Before you can train your custom object detector, you must convert your data into the TFRecord format.
- Since we need to train as well as validate our model, the data set will be split into training (train.record) and validation sets (val.record). The purpose of training set is straight forward - it is the set of examples the model learns from. The validation set is a set of examples used DURING TRAINING to iteratively assess model accuracy.
- You can use create_tf_record.py to convert our data set into train.record and val.record. Download [here](#) and save it to models/research/object_detection/dataset_tools/.

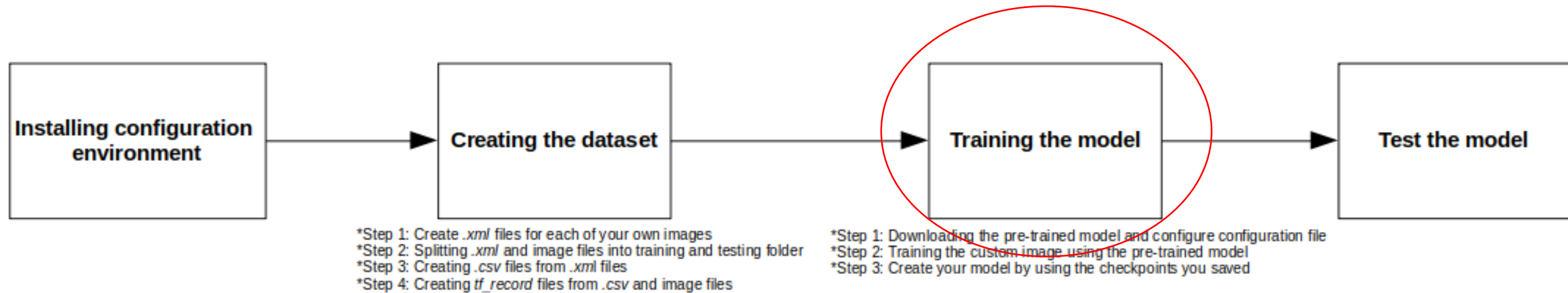
Just change the label name in if row_label == 'Label1': as per your classifications.

From the models directory

\$ python research/object_detection/dataset_tools/create_tf_record.py

- If the script is executed successfully, train.record and val.record

Tensorflow Object Detection API flowchart



Step3: Training the model (1/4)

1. Download a pre-trained model and configure configuration file

- There are many pre-trained object detection models available in the [model zoo](#). In order to train them using our custom data set, the models need to be *restored* in Tensorflow using their checkpoints (.ckpt files), which are records of previous model states. (model.ckpt.meta, model.ckpt.index, model.ckpt.data-000000-of-000001)
- Each of the pretrained models has a config file that contains details about the model. To detect our custom class, the config file needs to be modified accordingly.
- The config files are included in the models directory you cloned in the very beginning. You can find them in:

`models/research/object_detection/samples/configs`

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
CenterNet MobileNetV2 FPN 512x512	6	23.4	Boxes
CenterNet MobileNetV2 FPN Keypoints 512x512	6	41.7	Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
EfficientDet D2 768x768	67	41.8	Boxes
EfficientDet D3 896x896	95	45.4	Boxes
EfficientDet D4 1024x1024	133	48.5	Boxes
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks
ExtremeNet (deprecated)	--	--	Boxes
ExtremeNet	--	--	Boxes

Step3: Training the model (2/4)

Here are the items we need to change:

- Since we're only trying to detect cats and dogs, change **num_classes** to **2** (Dogs & Cats).
- `fine_tune_checkpoint` tells the model which checkpoint file to use.
- The model also needs to know where the TFRecord files and label maps are for both training and validation sets. Since our `train.record` and `val.record` are saved in `tf_record` folder, our config should reflect that:

```
train_input_reader: {  
  tf_record_input_reader {  
    input_path: "tf_record/train.record"  
  }  
  label_map_path: "annotations/label_map.pbtxt"  
}  
eval_input_reader: {  
  tf_record_input_reader {  
    input_path: "tf_record/val.record"  
  }  
  label_map_path: "annotations/label_map.pbtxt"  
  shuffle: false  
  num_readers: 1  
}
```

Step3: Training the model (3/4)

At this point, your models directory should look like this:

```
models
├── annotations
│   ├── label_map.pbtxt
│   ├── trainval.txt
│   └── xmls
│       ├── 1.xml
│       ├── 2.xml
│       └── ...
├── images
│   ├── 1.jpg
│   ├── 2.jpg
│   └── ...
├── checkpoints
│   ├── model.ckpt.data-00000-of-00001
│   ├── model.ckpt.index
│   └── model.ckpt.meta
├── tf_record
│   ├── train.record
│   └── val.record
├── research
└── ...
...
```

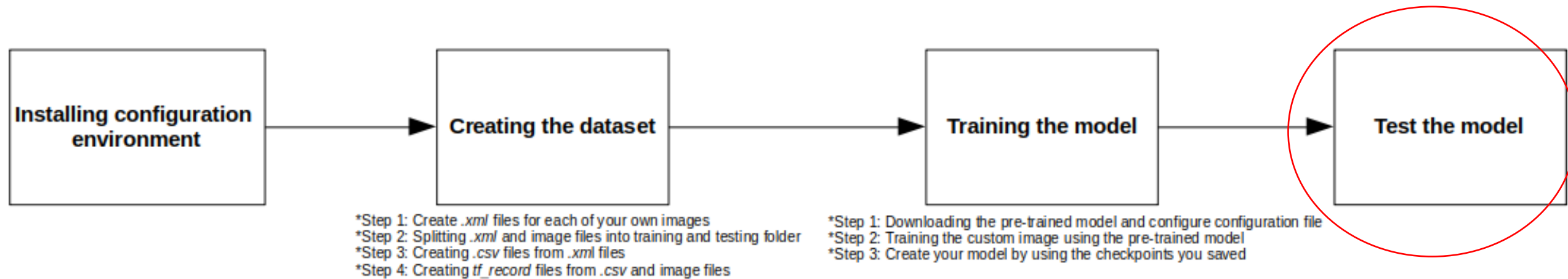
Step3: Training the model (4/4)

If you have successfully completed all previous steps, you're ready to start training!

Follow the steps below:

- **Change into the models directory**
`$ cd tensorflow/models#`
- **Make directory for storing training progress**
`$ mkdir train`
- **Make directory for storing validation results**
`$ mkdir eval`
- **Begin training with TensorFlow 2 Object Detector**
`$ python /content/models/research/object_detection/model_main_tf2.py`
`--pipeline_config_path={pipeline_file} \`
`--model_dir={model_dir} \`
`--alsologtostderr \`
`--num_train_steps={num_steps} \`
`--sample_1_of_n_eval_examples=1 \`
`--num_eval_steps={num_eval_steps}`

Tensorflow Object Detection API flowchart



Step4: Test the model (1/3)

- Evaluation

Evaluation can be run in parallel with training. The *eval.py* script checks the train directory for progress and evaluate the model

```
# From the models directory$ python research/object_detection/legacy/eval.py \  
--logtostderr \  
--pipeline_config_path=ssd_mobilenet_v2_coco.config \  
--checkpoint_dir=train \  
--eval_dir=eval
```

```
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.457  
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.729  
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.502  
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.122  
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.297  
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.659  
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.398  
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.559  
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.590  
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.236  
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.486  
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.746  
INFO:tensorflow:Writing metrics to tf summary.  
INFO:tensorflow:DetectionBoxes_Precision/mAP: 0.456758  
INFO:tensorflow:DetectionBoxes_Precision/mAP (large): 0.659280  
INFO:tensorflow:DetectionBoxes_Precision/mAP (medium): 0.296693  
INFO:tensorflow:DetectionBoxes_Precision/mAP (small): 0.122108  
INFO:tensorflow:DetectionBoxes_Precision/mAP@.50IOU: 0.728587  
INFO:tensorflow:DetectionBoxes_Precision/mAP@.75IOU: 0.502194  
INFO:tensorflow:DetectionBoxes_Recall/AR@1: 0.397509  
INFO:tensorflow:DetectionBoxes_Recall/AR@10: 0.558966  
INFO:tensorflow:DetectionBoxes_Recall/AR@100: 0.590182  
INFO:tensorflow:DetectionBoxes_Recall/AR@100 (large): 0.745691  
INFO:tensorflow:DetectionBoxes_Recall/AR@100 (medium): 0.485964  
INFO:tensorflow:DetectionBoxes_Recall/AR@100 (small): 0.236275  
INFO:tensorflow:Losses/Loss/BoxClassifierLoss/classification_loss: 0.234645  
INFO:tensorflow:Losses/Loss/BoxClassifierLoss/localization_loss: 0.139109  
INFO:tensorflow:Losses/Loss/RPNLoss/localization_loss: 0.603733  
INFO:tensorflow:Losses/Loss/RPNLoss/objectness_loss: 0.206419
```

Step4: Test the model (2/3)

- **Model export**

Once you finish training your model, you can export your model to be used for inference. If you've been following the folder structure, use the following command:

From the models directory

```
$ mkdir fine_tuned_model
```

```
$ python research/object_detection/export_inference_graph \  
--input_type image_tensor \  
--pipeline_config_path ssd_mobilenet_v2_coco.config \  
--trained_checkpoint_prefix train/model.ckpt-<the_highest_checkpoint_number> \  
--output_directory fine_tuned_model
```

Step4: Test the model (3/2)

- **Classify images**

Now that you have a model, you can use it to detect in test pictures and videos!

The models directory came with a notebook file (.ipynb) that we can use to get inference. It is located at `models/research/object_detection/colab_tutorials/object_detection_tutorial.ipynb`. Follow the steps below to tweak the notebook:

1. `MODEL_NAME = 'ssd_mobilenet_v2_coco_2018_03_29'`
2. `PATH_TO_CKPT = 'path/to/your/frozen_inference_graph.pb'`
3. `PATH_TO_LABELS = 'models/annotations/label_map.pbtxt'`
4. `NUM_CLASSES = 2`
5. Comment out cell #5 completely (just below Download Model)
6. Since we're only testing on one image, comment out `PATH_TO_TEST_IMAGES_DIR` and `TEST_IMAGE_PATHS` in cell #9 (just below Detection)
7. In cell #11 (the last cell), remove the for-loop, unindent its content, and add path to your test image:

```
imagepath = 'path/to/image_you_want_to_test.jpg'
```

After following through the steps, run the notebook and you should see your object in your test image highlighted by a bounding box!

How to train your own Object Detector with TensorFlow's Object Detector API

- <https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9>

