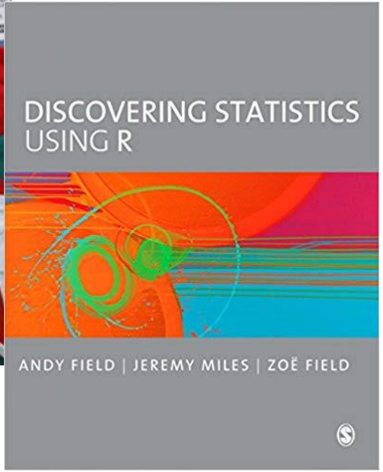
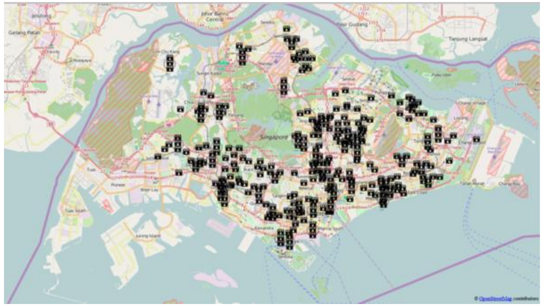
OBJECT DETECTION WITH TENSORFLOW API

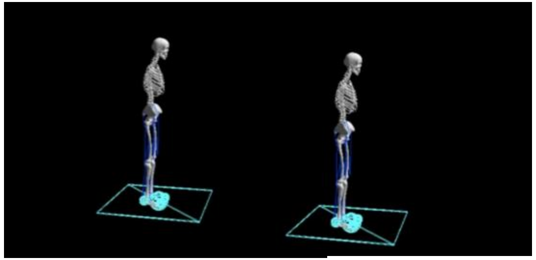
18th MAY, 10AM – 2.30PM
SHERLY CENDANA

Connect with me! ☺
Email: sherlyck2013@gmail.com
LinkedIn: www.linkedin.com/in/sherlyck

“When it’s fun, you enjoy it. When you’re afraid of it, it becomes stress. “







Tag #WWCodeSingapore on Instagram

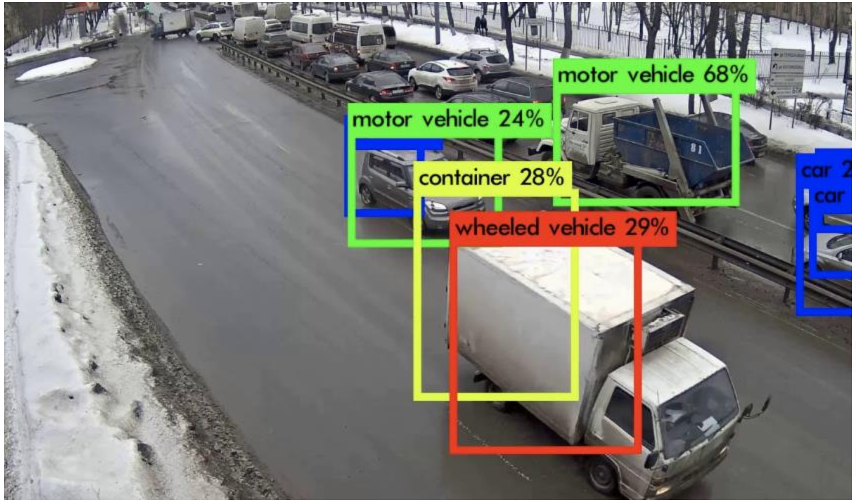
Take a picture or upload pictures of

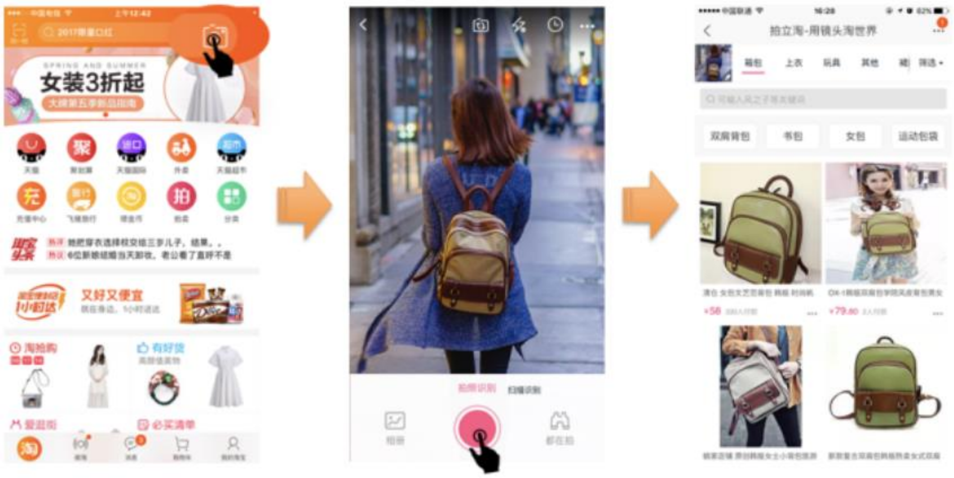
1. Something green
2. Someone cool
3. Your favourite item

AGENDA

1. Introduction
2. What is Object Detection
3. State of Object Detection
4. Tensorflow Object Detection API
   - Preparing Data – Crawling data from Instagram #hashtags
   - Selecting the model
   - Training & Evaluating *(Optional)*
   - Using the model – Visualizing
5. References

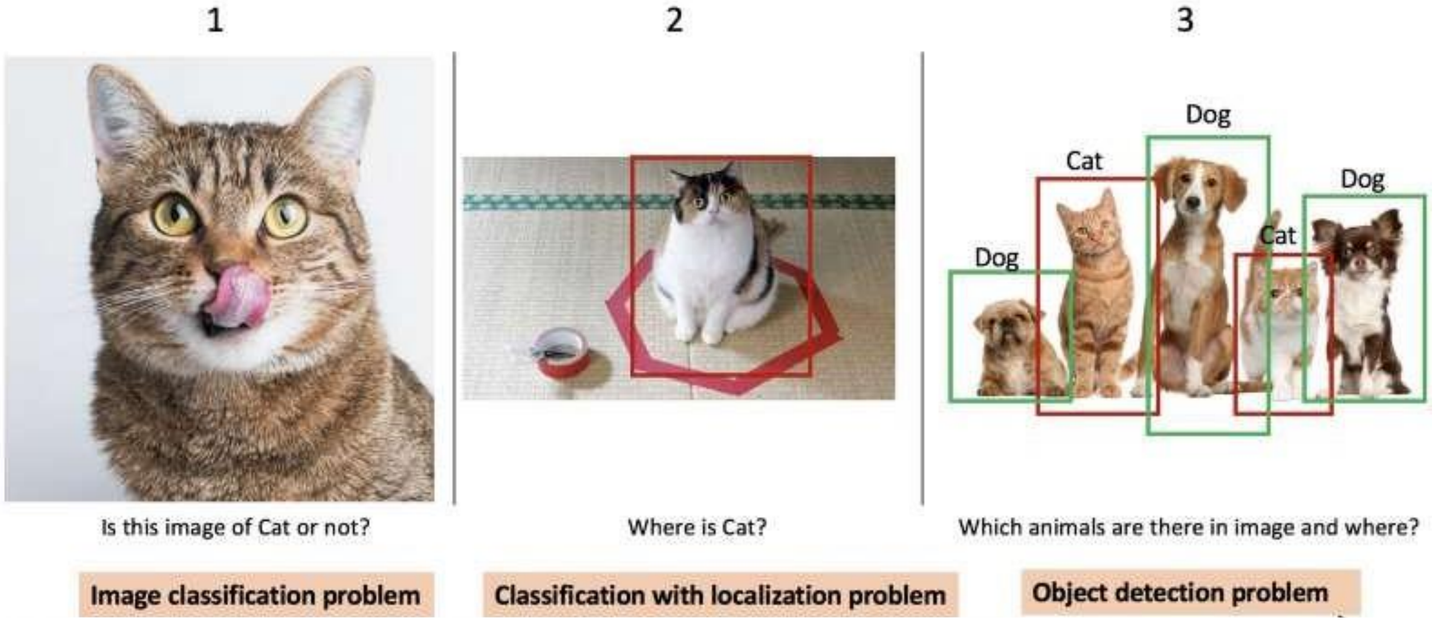
USE CASES


motor vehicle 68%
motor vehicle 24%
container 28%
wheeled vehicle 29%
car
car


女装3折起
又好又便宜
拍立淘-用镜头淘世界

WHAT IS OBJECT DETECTION?

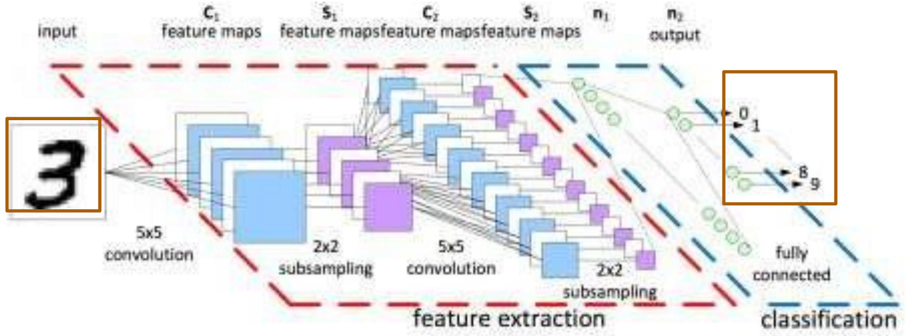
Object detection = Object Classification + Object Localization



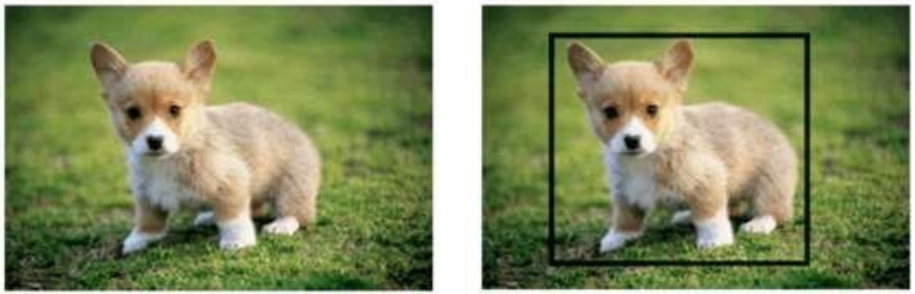
OUTPUT OF OBJECT DETECTION

Object detection = Object Classification + Object Localization

Object classification: Output is the one number (index) of a class



Object Localization: Output is the four numbers - coordinates of bounding box



OBJECT DETECTION FRAMEWORK

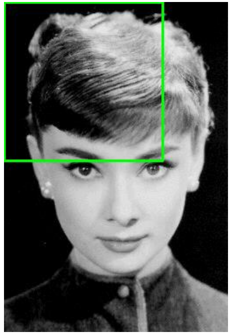
Object detection = Object Classification + Object Localization

1 Region Proposal

Generate regions of interest

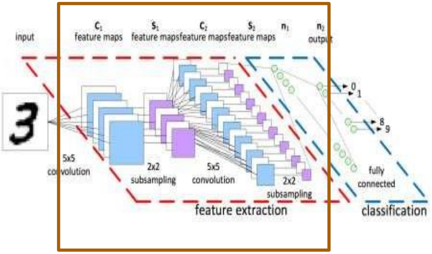
- Selective search
  Clustering approach to group pixels

- Sliding window approach
  Bounding boxes used as ROI

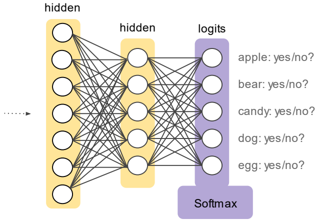


2 Object Classification

1. Feature extraction & learning

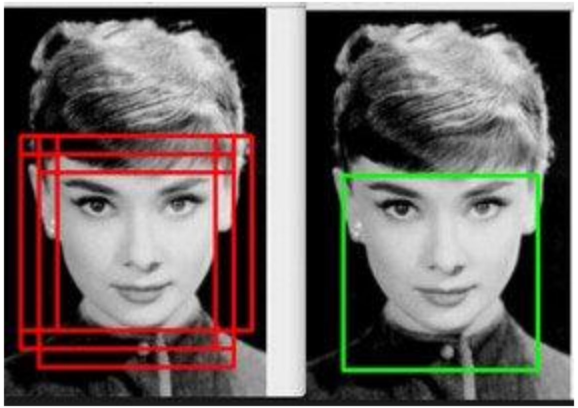


2. Classification



3 Non maximum suppression (NMS)

Post-processing step where overlapping boxes are combined into a single bounding

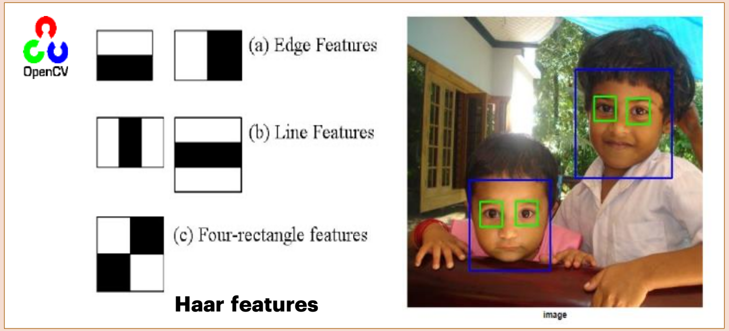


STATE OF OBJECT DETECTION

APPROACH

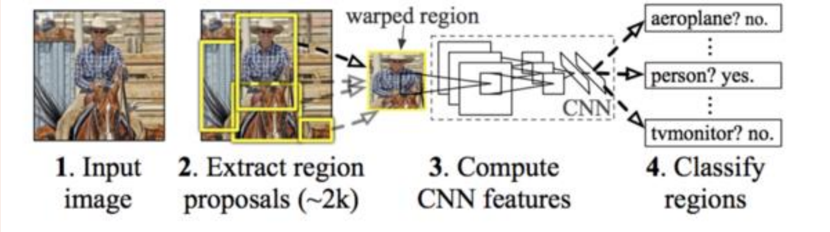
First Object Detection Framework:

- Haar feature-based cascade classifiers

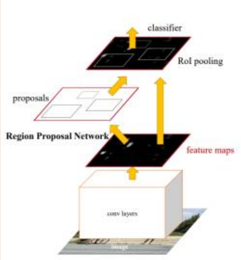


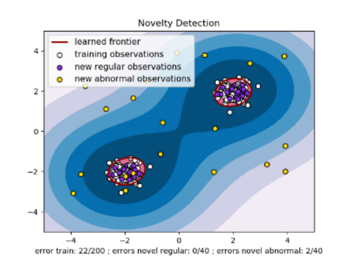
Deep learning approach:

- R-CNN
  Selective Search Region Proposal - Convolutional Neural Network



- Faster R-CNN
  Region Proposal Network (RPN) – Convolutional Neural network

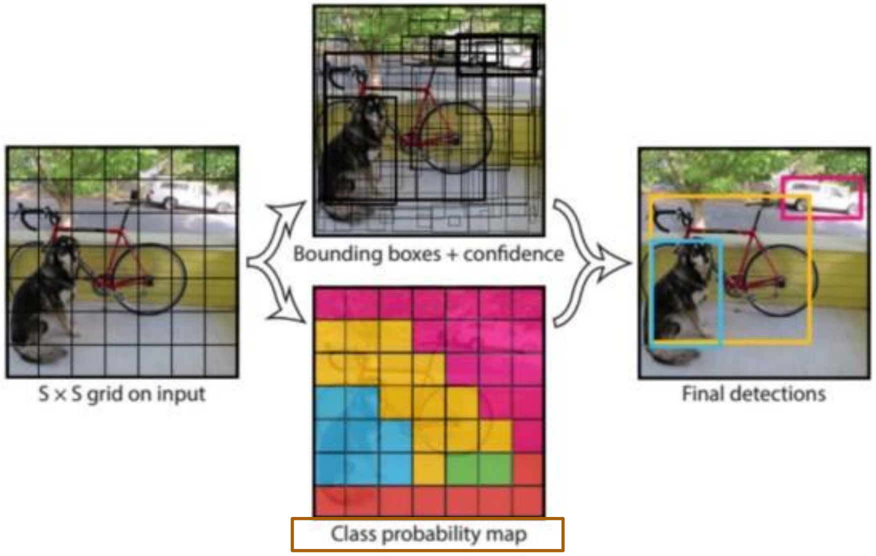


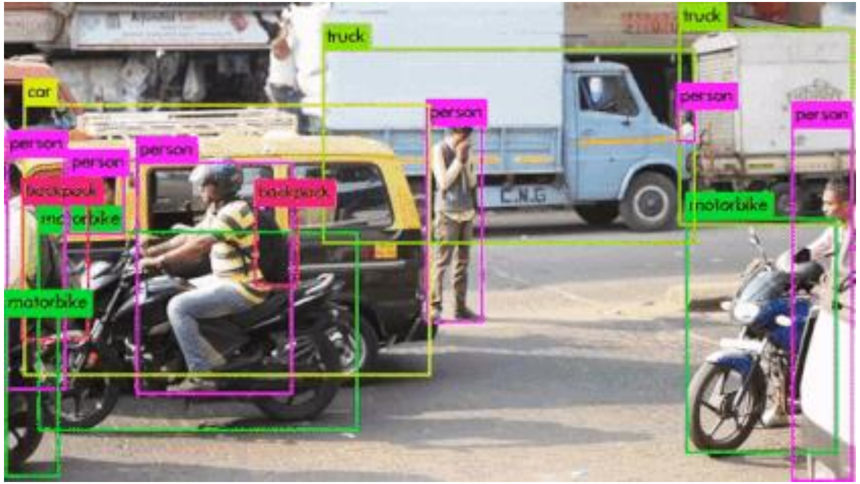


- YOLO (You Look Only Once)
- SSD (Single Shot MultiBox Detector)

DEEP LEARNING APPROACH

YOLO (You Look Only Once)





Class probability map: how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts

TENSORFLOW OBJECT DETECTION API

Open Source trainable models which makes it easy to construct, train and deploy object detection models

INSTALLATION - 1

1. Install Anaconda
   Go to https://www.anaconda.com/download/

2. Create new virtual environment and activate the environment

```
conda create -n tfdev pip python=3.6
conda activate tfdev
```

3. Install tensorflow & pre-requisites

```
pip install --ignore-installed --upgrade tensorflow==1.9
pip install pillow==5.41 lxml==4.3.1 jupyter notebook matplotlib==3.02 opencv-python
pip install –r requirements.txt
```

TENSORFLOW OBJECT DETECTION API

Open Source trainable models which makes it easy to construct, train and deploy object detection models

INSTALLATION -2

4. Download tensorflow models repo
   Go to https://github.com/tensorflow/models, git-clone or download zip file.

   Create a new folder under a path of your choice and name it TensorFlow. (C:\Users\XX\tensorflow).
   From your Anaconda/Command Prompt cd into the TensorFlow directory. Extract content inside the TensorFlow folder. Rename the extracted folder models-master to models

5. Install Protobuf

   Load the Google Protobuf folder in C:\Program Files
   (Windows) add 'C:\Program Files\Google Protobuf\bin' into your environtment variable

```
cd tensorflow/models/research
protoc object_detection/protos/*.proto --python_out=.
for /f %i in ('dir /b object_detection\protos\*.proto') do protoc object_detection\protos\%i --python_out=.
```

TENSORFLOW OBJECT DETECTION API

Open Source trainable models which makes it easy to construct, train and deploy object detection models

INSTALLATION -3

6. Add necessary environment variables

   (Windows) Add into Environment Variables > System Variables > PATH
   ‘C:\Users\XX\tensorflow\models\research\object_detection’ ,
   ‘C:\Users\ XX \tensorflow\models\research’
   ‘C:\Users\ XX \tensorflow\models\research\slim’

   (Linux)
   export PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research/object_detection
   export PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research:<PATH_TO_TF>/TensorFlow/models/research/slim

```
cd tensorflow/models/research
python setup.py build
python setup.py install
```

7. Test installation and run object_detection_tutorial.ipynb

```
cd tensorflow/models/research/object_detection
jupyter notebook
```

CREATE DATASET

Getting Images

1. INTERNET CRAWL

1. Download from Instagram

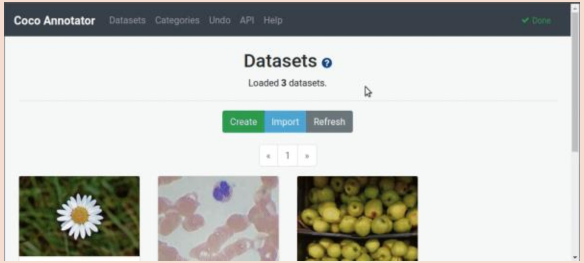
   ^ what we are doing today!

2. Scrap images from Google using Faktun Bulk Image Downloader

2. CREATE IMAGE DATASET

Image Annotation Tools

1. Coco Annotator
2. VGG Annotator (VIA)
3. LabelImg
4. FIAT (Fast Image Data Annotation Tool)



CRAWLING DATA FROM INSTAGRAM #HASHTAGS

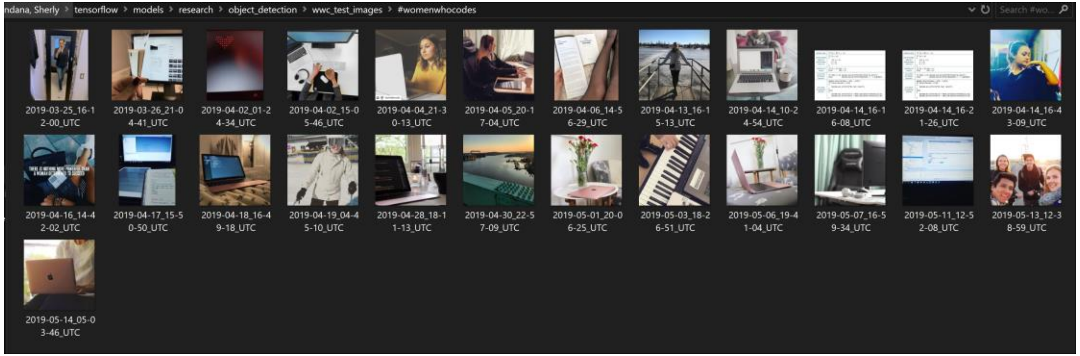
INSTALLATION

1. Install [InstaLoader]

```
pip install instaloader
```

2. Create a folder ‘wwc_test_images’ in /tensorflow/models/research/object_detection

```
instaloader --no-videos --no-metadata-json --no-captions "#womenwhocodes“
instaloader --no-videos --no-metadata-json --no-captions “#womenwhocodesg”
```



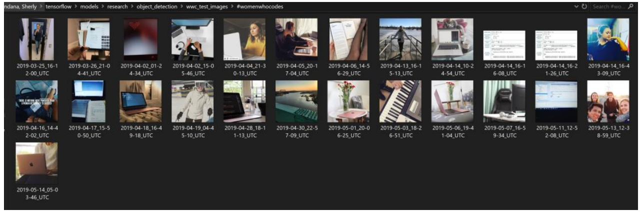
OPTIONAL

CREATE DATASET – TF RECORDS

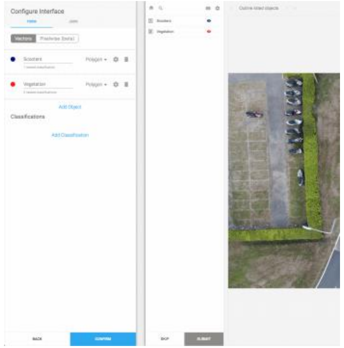
1. Tensorflow Object Detection API uses the TFRecord file format
2. Crawl Images > Annotation Tool > Generate tf records

```
python generate_tfrecord.py --csv_input=data/train_labels.csv  --output_path=train.record
```

Create dataset



Annotate dataset



Generate TF Records

MODEL SELECTION

1. Tensorflow Object Detection API contains detection models pre-trained on the COCO dataset, the Kitti dataset, and the Open Images dataset.

2. Go to: Model Collection

Item	Description
Model Name	Config file that was used to train this model
Speed (ms)	running time in ms per 600x600 image
COCO mAP[^1]	Mean Average Precision of how well the model performed on the COCO dataset (0 to 100, higher the better)
Outputs	Type of output: Boxes, and Masks if applicable

COCO-trained models

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes

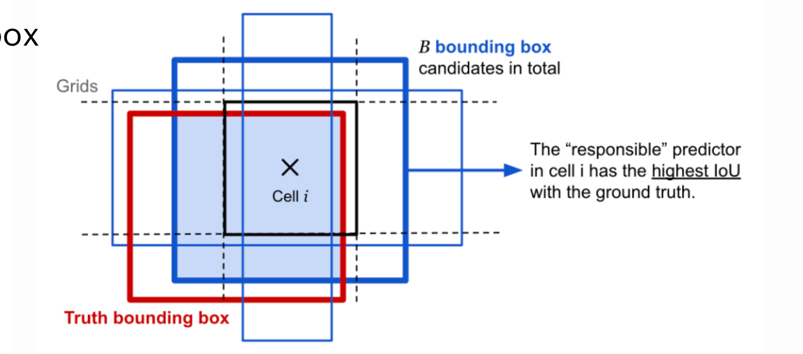
DEEP LEARNING APPROACH

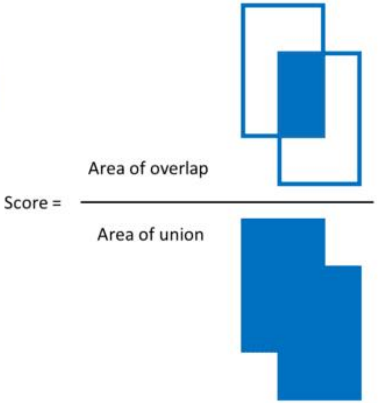
Model & Frozen weights used: ssd_mobilenet_v1_coco_2017_11_17

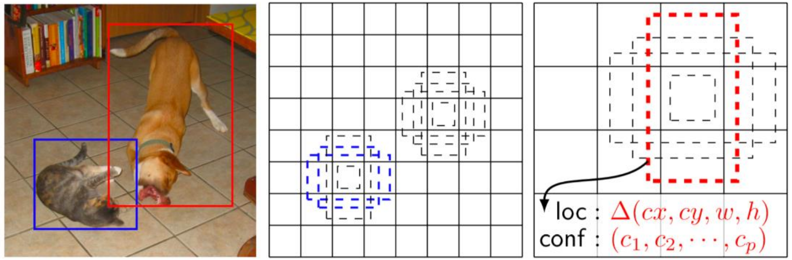
SSD: SINGLE SHOT MULTIBOX DETECTOR

Each bounding box will have a score associated (likelihood of the box containing an object).

Detection is a true positive if it has an 'intersection over union' (IoU or overlap)







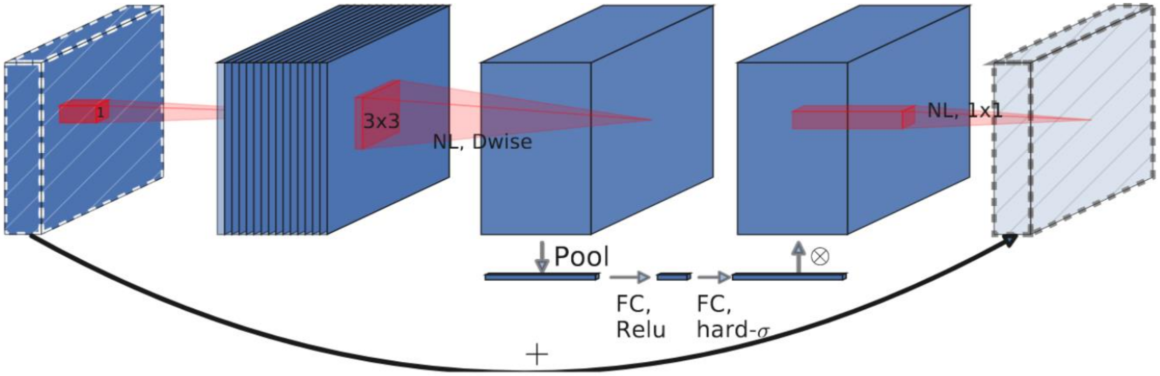
DEEP LEARNING APPROACH

Model & Frozen weights used: ssd_mobilenet_v1_coco_2017_11_17

MOBILENET

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000



OPTIONAL

CONFIGURE PIPELINE

1. Tensorflow Object Detection API uses protobuf files to configure the training and evaluation process
2. Go to: [Configuration Pipeline]
3. Refer to pipeline.config in the folder

Item	Description
model	Type of model
train_config	Model parameters ie. SGD parameters, input preprocessing and feature extractor initialization values
eval_config	Evaluation metrics
train_input_config	Training dataset
eval_input_config	Evaluation dataset

model

```
model {
  faster_rcnn {
    num_classes: 1
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 300
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
```

train_config

```
train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
```

eval_config

```
eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 100
  num_visualizations: 15
  max_num_boxes_to_visualize: 1000
  visualization_export_dir: "/sherly/tfdevtest/output/viz"
  keep_image_id_for_visualization_export: true
  eval_interval_secs: 60
```

train_input_config

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "/sherly.cendana/tfdevtest/output/train_holdout.record"
  }
  label_map_path: "/data/tfdevtest/output/labelmap.pbtxt"
```

eval_input_config

```
eval_input_reader: {
  tf_record_input_reader {
    input_path: "/sherly.cendana/tfdevtest/output/test_holdout.record"
  }
  label_map_path: "/data/tfdevtest/output/labelmap.pbtxt"
  shuffle: false
  num_readers: 1
```

OPTIONAL

TRAINING AND EVALUATING

1. Training the model

```
cd tensorflow/models/research

python object_detection/train.py
--logtostderr
--pipeline_config_path=/tensorflow/models/object_detection/samples/configs/ssd_mobilenet_v1_p  ets.config
--train_dir=${PATH_TO_ROOT_TRAIN_FOLDER}
```

2. Evaluating the model

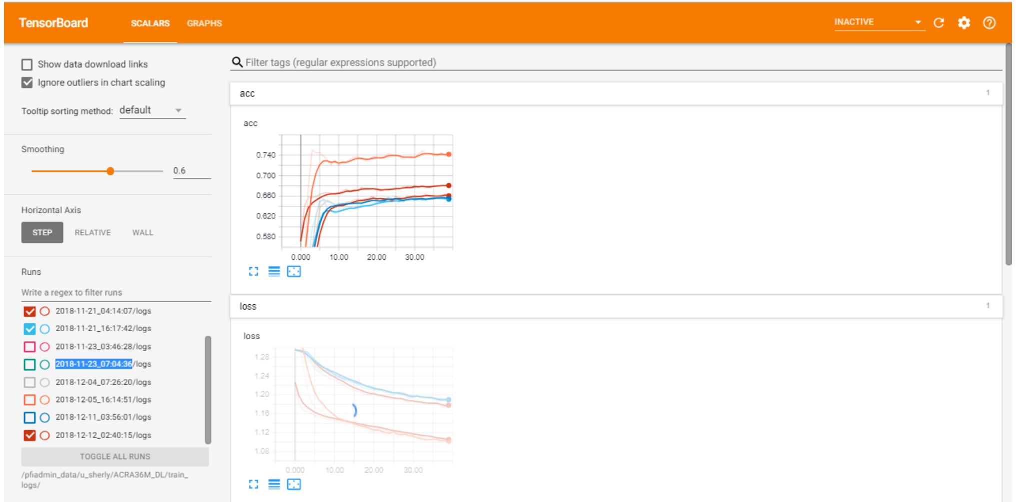
```
cd tensorflow/models/research

python object_detection/eval.py \
--logtostderr \
--pipeline_config_path=${PATH_TO_YOUR_PIPELINE_CONFIG} \
--checkpoint_dir=${PATH_TO_TRAIN_DIR} \
--eval_dir=${PATH_TO_EVAL_DIR}
```

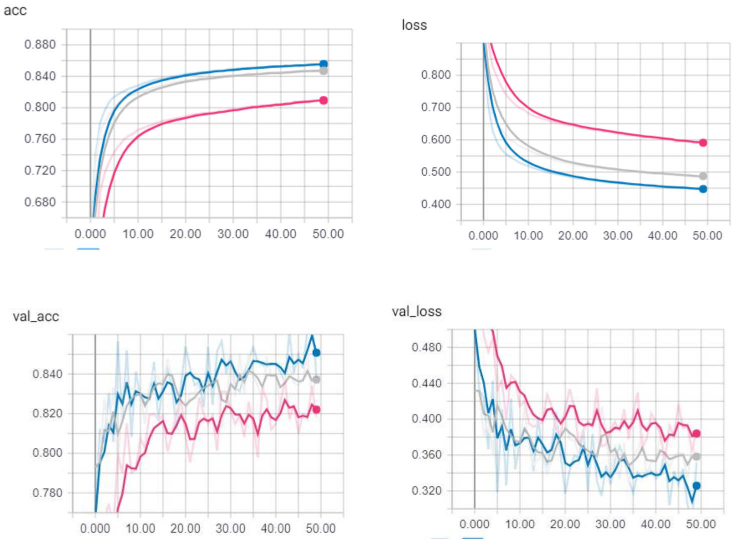
3. Visualise the training and evaluation results using tensorboard

```
cd tensorflow/models/research
tensorboard --logdir <directory>/train_logs --port 6004
```

TENSORBOARD – TENSORFLOW'S VISUALIZATION TOOLKIT

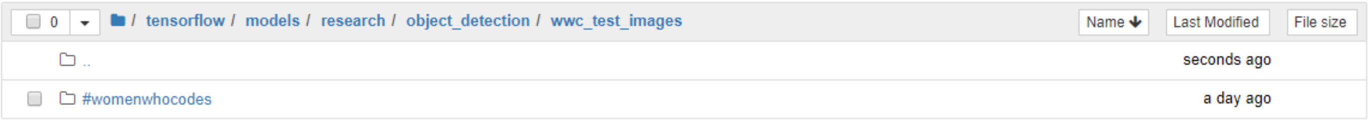
1. Tensorboard aids understanding, help in debug, and optimize TensorFlow programs
2. Go to: [Tensorboard]





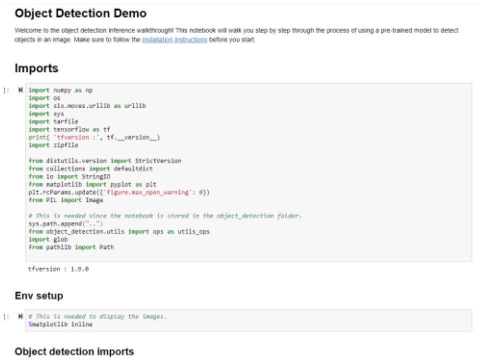
MODEL USAGE (1/2)

1. Using the photos we have crawled from Instagram, apply the object detection model on the images. Ensure that dataset is present in the folder:



2. Load jupyter notebook 'wwc_object_detection_tutorial.ipynb'

```
cd tensorflow/models/research
Jupyter notebook
```



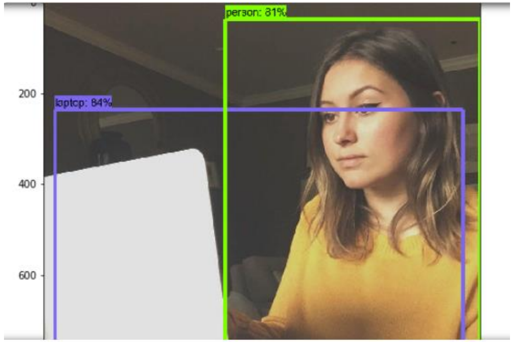
Ensure that your tensorflow version is later than 1.12

MODEL USAGE (2/2)

3. Load the path to your images

```
PATH_TO_TEST_IMAGES_DIR ='wwc_test_images/#womenwhocodes'
TEST_IMAGE_PATHS =[]
for file in os.listdir(PATH_TO_TEST_IMAGES_DIR):
    if file.endswith(".jpg"):
        print(os.path.join(PATH_TO_TEST_IMAGES_DIR, file))
        TEST_IMAGE_PATHS.append(os.path.join(PATH_TO_TEST_IMAGES_DIR, file))
```

4. Run the notebook and you should see the following output:



THANK YOU!

Go to: www.menti.com and use the code 20 40 96

REFERENCES

- https://towardsdatascience.com/how-to-train-your-own-object-detector-with- tensorflows-object-detector-api-bec72ecfe1d9
- https://www.kdnuggets.com/2017/10/deep-learning-object-detection- comprehensive-review.html
- http://www.machinelearninguru.com/deep_learning/tensorflow/basics/tfrecord/tfreco rd.html
- https://www.coursera.org/learn/convolutional-neural-networks
- https://www.tensorflow.org/guide/summaries_and_tensorboard
- https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html
- https://arxiv.org/pdf/1512.02325.pdf