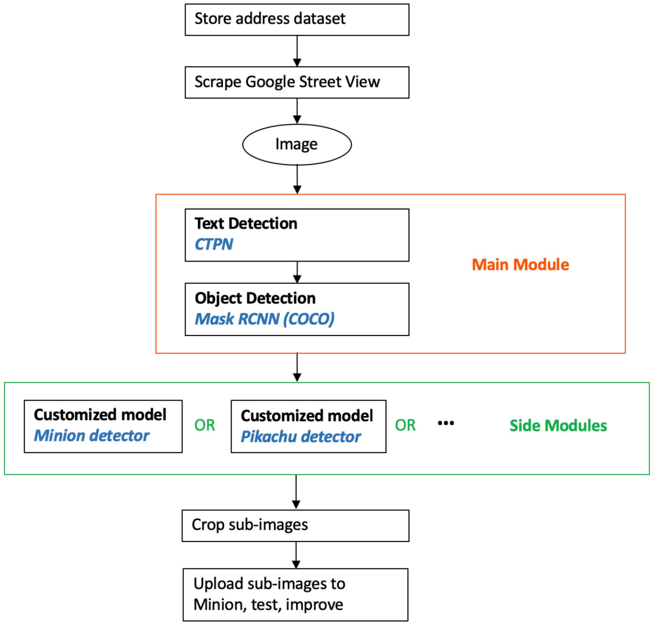
Image Extraction

Github: https://github.com/sibylhe/Project-Minions

1 Workflow



Goal

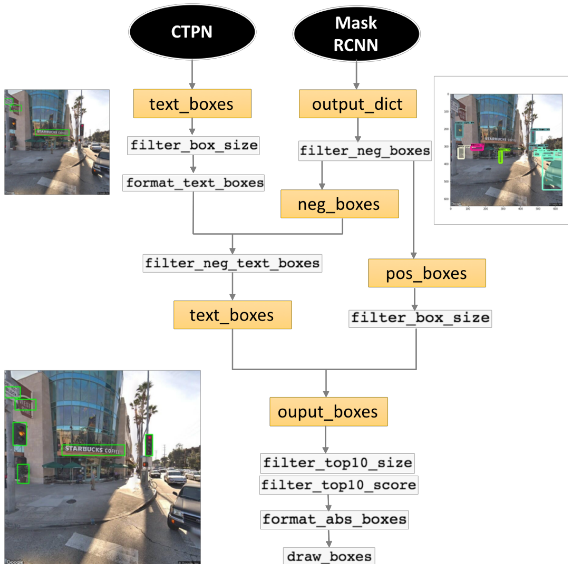
Extract unique and permanent objects from Google Streetview images as AR object anchors.

2 Main module

https://github.com/sibylhe/Project-Minions/blob/master/image-extraction/Image_Extraction_1019.ipynb

The main module does a general search and selection:

- Text detection: extract store name, sign, etc.
- Object detection: extract common permanent objects (building, tree, traffic light, etc.); and exclude non-permanent ones (person, car, etc.)



2.1 CTPN

Forked from: https://github.com/eragonruan/text-detection-ctpn

Detecting text in natural image with connectionist text proposal network

CTPN gets more ideal results than Google Cloud Vision Text Detection API:

- It's more **sensitive**: more objects detected;
- It's more **robust** in combining intensive boxes in a region, i.e., outputs good boxes
- Its output boxes format is **easily accessible**.

Google Cloud Vision performs precise text recognition (reading the content of text, which we are not interested in), but gives poor bounding boxes (which we care). It outputs layers of boxes. The result is complicated to access because to get a box of ideal size, you need to specify which layer it belongs to, otherwise you loop through all the layers and get multiple boxes.

2.2 **Mask RCNN** trained on COCO

Forked from: https://github.com/tensorflow/models

To keep the main module **lean, fast and stable**, I recommend models trained on **COCO** dataset (80 classes, common objects). We extend these classes by training custom side models. Models trained on Open Images datasets are able to detect more classes (500 classes) but may be too complex (20x slower, crashes occasionally).

Other object detection models:

- YOLOv3: fast, precise, and sensitive. Highly recommended, but perhaps not python-friendly.
- Faster RCNN: precise, widely used. Slower than Mask RCNN in my test.

2.3 Filters

- filter_box_size: keep boxes larger than threshold *image size, threshold = 0.004 for CTPN, threshold = 0.002 for Mask RCNN, to be tested
- format_text_boxes: format CTPN boxes to align with Mask RCNN boxes
- filter_neg_boxes: classify positive and negative boxes output by Mask RCNN. White list (COCO): ['stop sign', 'traffic light', 'clock', 'bench', 'potted plant', 'fire hydrant', 'parking meter', 'toilet']
- filter_neg_text_boxes: eliminate text boxes overlapping with negative boxes, threshold = 0.5 (overlapping area >= 0.5 * text_box size, eliminate the text box), to be tested
- filter_top10_boxes_size: filter top10 boxes by size
- filter_top10_boxes_score: filter top10 boxes by score
- format_abs_boxes: boxes in ratio -> absolute coordinates
- draw_boxes: visualize boxes and write output txt file

3 Side modules: training custom Mask RCNN/Faster RCNN (on-going)

https://github.com/sibylhe/Project-Minions/tree/master/mrcnn

Train/switch to **customized modules** per situation – extend the main module flexibly.

3.1 Create Annotations

For each image, Mask RCNN needs

- masks encoded as png
- label information in xml or json, which can be parsed to python dictionary

Annotation tools

Labelme: http://labelme.csail.mit.edu/Release3.0/

I used this. It's easy to create mask by using a brush-like tool to color the area (ROI) you want. Outputs a mask png file for each object in the image respectively and label info in xml.

Labelme for python: https://github.com/wkentaro/labelme

For each image, outputs a png file containing multiple masks and label info in json. But this doesn't have a "brush" tool. To create a mask, you need to draw "polygen", which is much more time-consuming.

PixelAnnotationTool:
https://github.com/abreheret/PixelAnnotationTool/blob/master/README.md
Similar to the first lableme.
VIA: http://www.robots.ox.ac.uk/~vgg/software/via/
Similar to the second labelme.

Steps to create annotation

a) Image extension must be .jpg. Rename other extensions to .jpg

b) Sign up to labelme http://labelme.csail.mit.edu/Release3.0/, create a collection (folder), and upload images to the collection.

c) To annotate an image, click on the image thumbnail, use the red brush to color your object, use the blue brush to subtract area.

Once done with an object, click done and type in:
Name: class name
Attribute: optional, more info about the object

Continue with next object.

d) Click "Download all".

Labelme output (for each image):
We need the masks and xml.

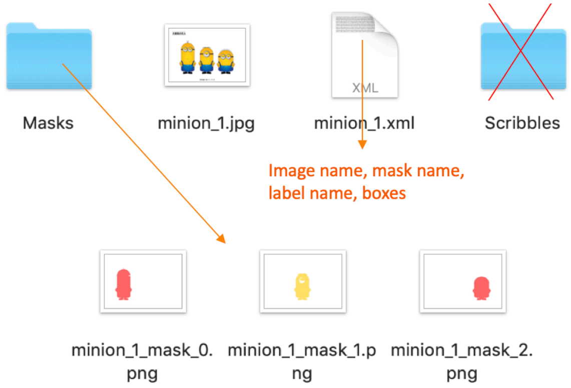


Image name, mask name, label name, boxes

3.2 Train Mask RCNN with Tensorflow Object Detection API

https://towardsdatascience.com/building-a-custom-mask-rcnn-model-with-tensorflow-object-detection-952f5b0c7ab4 https://github.com/priya-dwivedi/Deep-Learning/tree/master/Custom_Mask_RCNN

3.2.1 Generate TF Records

The Tensorflow Object Detection API takes a TFRecord file as input, which is a wrap of the images, boxes, masks, and labels.

For every example in your dataset, you should have the following information:

1. An RGB image for the dataset encoded as jpeg.
2. A list of bounding boxes for the image. Each bounding box should contain:
   i. A bounding box coordinates (with origin in top left corner) defined by 4 floating point numbers [ymin, xmin, ymax, xmax]. Note that we store the *normalized* coordinates (x / width, y / height) in the TFRecord dataset.
   ii. The class of the object in the bounding box.
   iii. The mask of the object encoded as png.

Script to create TFRecord:
https://github.com/liqunfeifei/Deep-Learning/blob/master/Custom_Mask_RCNN/create_pet_tf_record.py

3.2.2 Select model

mask_rcnn_inception_v2_coco
Fastest model, okay-ish accuracy.

3.2.3 Train model

Use Tensorboard to visualize/monitor training progress.

APPENDIX

I. Install and Run CTPN on CPU in Python

git clone git://github.com/eragonruan/text-detection-ctpn.git
or download: https://github.com/eragonruan/text-detection-ctpn

Install dependencies:
Author's environment: python2.7 (also works in python3.6), tensorflow1.3, cython0.24, opencv-python, easydict (recommend to install Anaconda)

```
pip install opencv-python
pip install easydict
```

1. To use cpu only, make the following modifications:
   a) Set "USE_GPU_NMS " in the file ./ctpn/text.yml as "False"
   b) Set the "__C.USE_GPU_NMS" in the file ./lib/fast_rcnn/config.py as "False";
   c) Comment out the line "from lib.utils.gpu_nms import gpu_nms" in the file ./lib/fast_rcnn/nms_wrapper.py;

2. https://github.com/eragonruan/text-detection-ctpn/releases
   Download checkpoints.zip, save in ./text-detection-ctpn and unzip
   Download ctpn.pb, put in data/

3. Write a setup_cpu.py as following and save to ./text-detection-ctpn/lib/utils

```
from Cython.Build import cythonize
import numpy as np
from distutils.core import setup

try:
   numpy_include = np.get_include()
except AttributeError:
   numpy_include = np.get_numpy_include()

setup(
   ext_modules=cythonize(["bbox.pyx","cython_nms.pyx"]),
)
```

4. Execute the following in command line:
   a) export CFLAGS=-I/anaconda3/lib/python3.6/site-packages/numpy/core/include (replace with your numpy path)
   b) cd ./text-detection-ctpn/lib/utils and execute: python setup_cpu.py build (replace with the setup file you write)
   c) copy the .so file from the "build" directory to the ./text-detection-ctpn/lib/utils
   d) put your images in data/demo, the results will be saved in data/results

e) cd ./text-detection-ctpn and execute: python ./ctpn/demo.py

Trouble shooting

KeyError: b'TEST'
./text-detection-ctpn/lib/rpn_msr
line 45: #cfg_key=cfg_key.decode('ascii')
remove the "#", turn this comment back into code
https://github.com/eragonruan/text-detection-ctpn/blob/3abf200be51fb577682681a85814aceb2460f804/lib/rpn_msr/proposal_layer_tf.py#L46

'ValueError: Variable conv1_1/weights already exists, disallowed. Did you mean to set reuse=True or reuse=tf.AUTO_REUSE in VarScope?
demo.py

```
# init session
tf.get_variable_scope().reuse_variables()
config = tf.ConfigProto(allow_soft_placement=True)
sess = tf.Session(config=config)
```

II. Run Mask RCNN in Tensorflow

Install Tensorflow and dependencies
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

Clone https://github.com/tensorflow/models
Unzip and put it under your tensorflow path
e.g., /anaconda3/lib/python3.6/site-packages/tensorflow/models

Follow the tutorial to download and run Mask RCNN (and other object detection models):
./tensorflow/models/research/object_detection/object_detection_tutorial.ipynb
Or run my notebook:
https://github.com/sibylhe/Project-Minions/blob/master/image-extraction/Mask_RCNN_1014.ipynb
(put it under ./tensorflow/models/research/object_detection/)

ARCHIVE
Backup models: YOLOv3, Faster RCNN, Cloud Vision Text Detection

YOLOv3
YOLOv3 (Open Images, 500 classes)
Weights and config: https://github.com/radekosmulski/yolo_open_images
https://www.kaggle.com/c/google-ai-open-images-object-detection-track/discussion/64734
Class labels (2018-class-descriptions-500.csv): https://www.kaggle.com/martial/classes-of-open-images-dataset-v4#challenge-2018-class-descriptions-500.csv

YOLOv3 (COCO) in Python:
80 classes
https://github.com/mystic123/tensorflow-yolo-v3

Implementing YOLOv3 in Python:
https://www.kaggle.com/sajinpgupta/object-detection-using-yolov3
https://itnext.io/implementing-yolo-v3-in-tensorflow-tf-slim-c3c55ff59dbe

Faster RCNN
https://github.com/sibylhe/Project-Minions/blob/master/image-extraction/Tensorflow%20Object%20Detection_0925.ipynb

Faster RCNN (Open Images)
faster_rcnn_inception_resnet_v2_atrous_oid:
This model is slow, 770ms per image as reported by Google, while other models generally use 20-30ms.

Faster RCNN (COCO)
faster_rcnn_inception_resnet_v2_atrous_coco
Gets same results as Mask RCNN (COCO), but is much slower and unstable.

Google Cloud Vision Text Detection
https://github.com/sibylhe/Project-Minions/blob/master/image-extraction/Cloud%20Vision_Text%20Bounding%20Box_0925.ipynb

CTPN on GPU
https://github.com/tianzhi0549/CTPN

Starbucks store address: https://www.kaggle.com/starbucks/store-locations