CoachMyLife Report

August 5, 2019

Chapter 1

Available datasets

1.1 ImageNet

Main dataset

- `http://image-net.org/index`
- categorise images according to the WordNet hierarchy;
- 14M images, 22K synsets (synonims groups);
- 1M images with bounding boxes around dominant object.

Challenges

- ≈1.2M images, 1K categories;
- ≈600K with boxes.

1.2 OpenImage

Statistics:

- `https://storage.googleapis.com/openimages/web/index.html`
- ≈9M images, 19943 image-level classes;
- ≈2M images with bounding boxes (largest existing dataset with object location annotations);
- 545 boxable classes (relevant and with clearly defined spatial extent);
- boxable classes follow hierarchy (see Fig.)

Separated into a number of components

- image index file -> image URL and ID for every image in the dataset (even images that donâĂŹt contain bbox annotations!);
- class descriptions -> correspondence between class code and textual expression (/m/011k07,Tortoise)

Table 1.1: Available Datasets, statistics for **object detection**

Name	Nr. images	Classes
ImageNet	1M	?
OpenImage	2M	545
COCO	200K	90

- annotation file -> ImageID, Source, LabelName, Confidence, XMin, XMax, YMin, YMax, IsOccluded, IsTruncated, IsGroupOf, IsDepiction, IsInside
- trainable classes files. In new release, unified to class_descriptions (boxable class descriptions)

1.3 COCO

- `http://cocodataset.org/#home`
- ≈200K images;
- 90 classes;
- object segmentation, 5 textual captions per image.

Chapter 2

Models

- framework: https://github.com/tensorflow/models/tree/master/research/object_detection;
- protobuf compiling broken **on Windows**, this works:

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

- adapted for object detection in both images and stream from webcam (TODO: with OpenCV, get list of available devices to be able to switch video source, as it is now it only recognizes device #0);
- stream capture from webcam must be run from WITHIN the TF session, to avoid framerate drop;
- available models trained on OI and COCO, main trade-off between accuracy and framerate (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md);
- TODO: extract labels and compare results with test dataset to obtain detection accuracy manually, compare different models and hardware.

2.1 Transfer Learning

2.1.1 Intro stuff

Further training of model by adding new set of images for new objects to detect.

Ways to get dataset for transfer learning:

- Manually
  Manual search and download of images, using LabelImg for boxing and labeling provides single XML files for each image in PASCAL VOC standard format.

- OIDv4_ToolKit script to download single classes images from OI. It's better to use test and validation sets, too many images in training ($\approx 5K$). Images in SCENE!
- Macncheese/racoon datasets, mostly ICONIC VIEWS.

In any case, single .csv table with all bounding boxes information is needed to generate a TFrecord object (data for transfer learning).
Modified code from datitran to do both things (TODO: check better the object recognition API default routine for TFrecord generation to see if it is any better).
Folder organized this way:

```
tf_custom_obj_detector
        -data/
                --test_labels.csv ()
                --train_labels.csv
                --TFrecord objects generated here at
                   the end!!!
        -images/
                --test/
                        ---test_images_00.jpg
                        ---test_images_00.xml
                        ...
                        ---test_images_nn.jpg
                        ---test_images_nn.xml
                --train/
                        ---train_images_00.jpg
                        ---train_images_00.xml
                        ...
                        ---train_images_nn.jpg
                        ---train_images_nn.xml
        -training/
        -generate_tfrecord.py
        -xml_to_csv.py
```

2.1.2 Procedure

Summary of transfer learning procedure:

1. PYTHONPATH="path_to_tf_models"; "path_to_tf_models/slim";
2. from tf_custom_obj_detector, use **xml_to_csv.py** (or similar), obtain:
   - data/train_labels.csv
   - data/test_labels.csv
3. to generate TFrecords for training and set data:

   ```
   TRAIN
   ```

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

TEST
python generate_tfrecord.py --csv_input=data/
   test_labels.csv --output_path=data/custom_test.
   record      --image_dir=images/test
```

4. model training configuration: in the training directory, need:

   - **custom_label_map.pbtxt**, in the format:

     ```
     item {
     id: 1  -> start from 1, 0 is placeholder!!!
     name: 'name_to_display_for_entry_1'
     }

     (.....)

     item {
     id: N
     name: 'name_to_display_for_entry_N'
     }
     ```

   - **configuration file** of the model (examples in object_detection\samples\config), edit:
     - num_classes
     - batch_size = 24 (default)
     - search for all of the ”PATH_TO_BE_CONFIGURED”
     - fine_tune_checkpoint: ”folder_of_model/model.ckpt”
     - for:

       ```
       train_input_reader: {
       tf_record_input_reader {
       input_path: "training_TFrecord_path" ("data/
          custom_train.record")
       }
       label_map_path: "label_map_path" ("data/
          custom_label_map.pbtxt")
       }

       (.....)

       eval_input_reader: {
       tf_record_input_reader {
       input_path: "data/test.record"
       }
       label_map_path: "training/custom_label_map.
          pbtxt"
       ```

```
        shuffle: false
        num_readers: 1
        }
```

5. **RUNNING THE TRAINING**: from within models/object_detection, run:

   ```
   python train.py (or model_train.py***)  --
      logtostderr --train_dir=training/ --
      pipeline_config_path=training/
      ssd_mobilenet_v1_coco.config
   ```

   ***** model_train.py not working on Windows, errors on cocoapi library, need to build it somehow!!!**

6. exporting new inference graph:

   ```
   python export_inference_graph.py --input_type
      image_tensor --pipeline_config_path training/
      ssd_mobilenet_v1_coco.config --
      trained_checkpoint_prefix training/model.ckpt
      -1147 --output_directory
      mac_n_cheese_inference_graph
   ```

Bibliography

University", P. (2010). Wordnet. a lexical database for english.