딥러닝과 머신러닝

#2 Tensorflow API 를 이용한 object detection 실습

이근영(메디웨일)

Medi Whale

Github

https://github.com/SoulDuck/learning_tensorflow

Tensorflow Object Detection

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

Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.

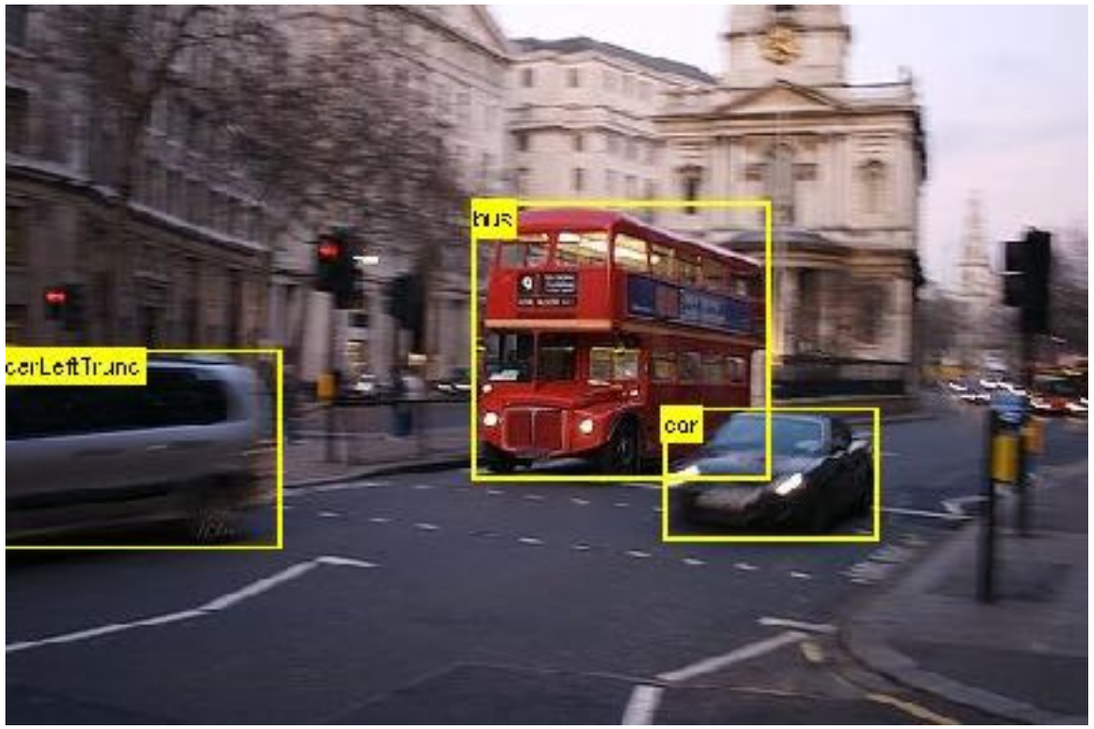
Tensorflow Object Detection Pretrained Model

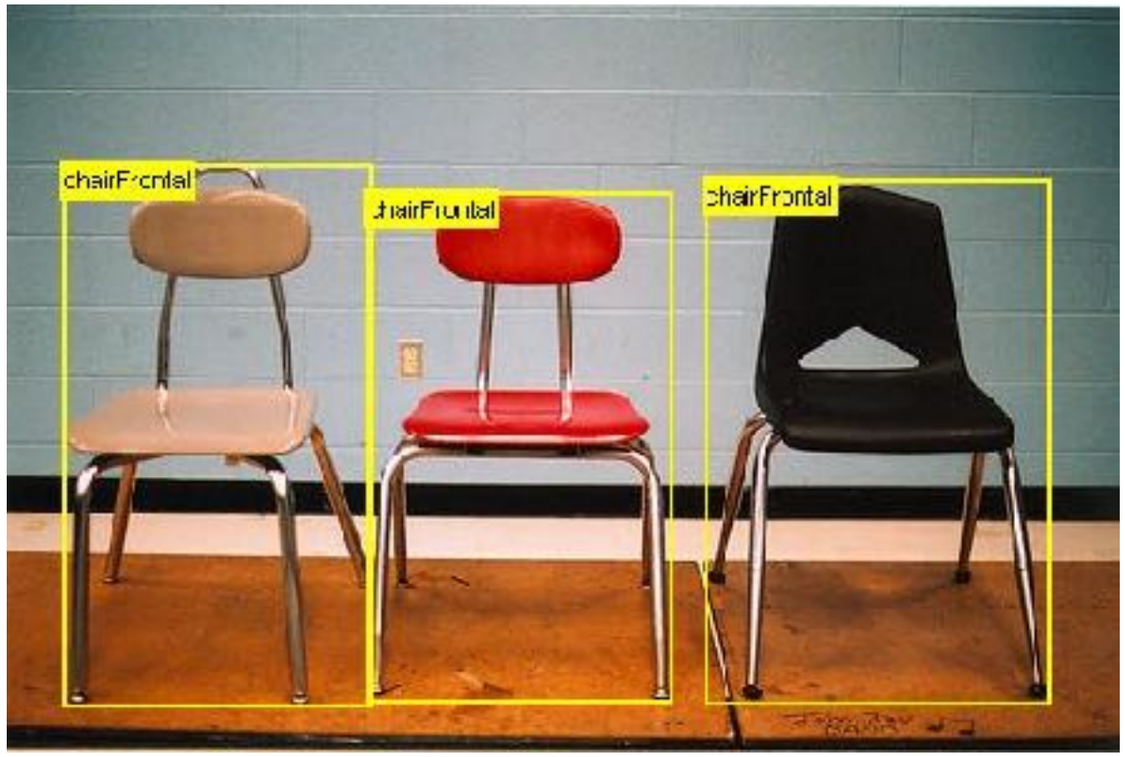
MS coco

90 class objects

Tensorflow Object Detection Pretrained Model

PASCAL VOC





20 class objects

Tensorflow Object Detection Pretrained Model

Oxford Pet image






37 class objects

Tensorflow Object Detection Pretrained Network

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes

- Single Shot Multibox Detector (SSD) with MobileNets
- SSD with InceptionV2
- Region-based Fully Convolutional Networks (R-FCN) with Resnet 101
- Faster RCNN with Resnet 101
- Faster RCNN with Inception Resnet v2

실습환경

http://13.84.153.73:8888/ 로 접속

tensorflow_api 폴더 안에 실습 파일 및 예제 폴더

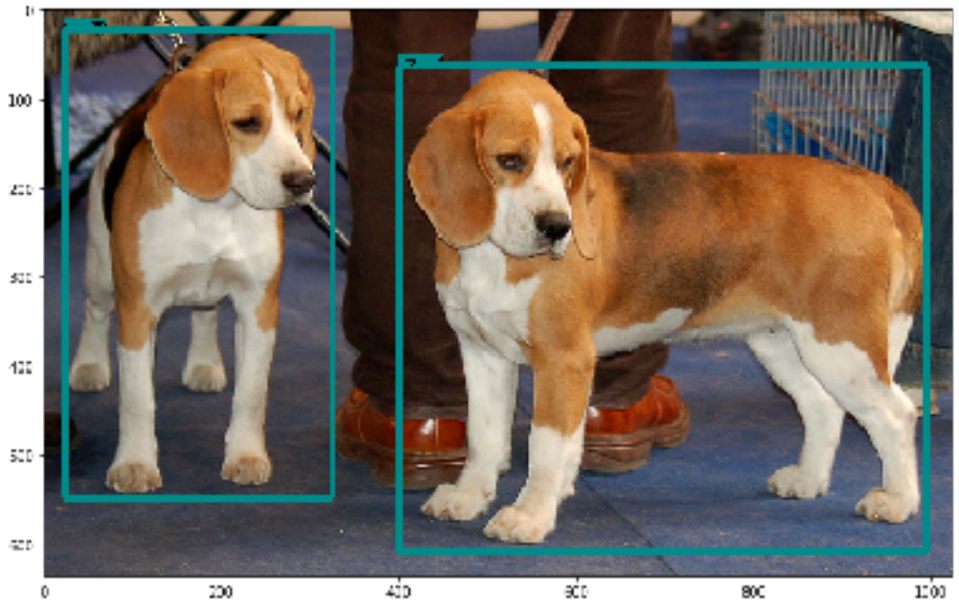
tensorflow object detection 설치 및 설정

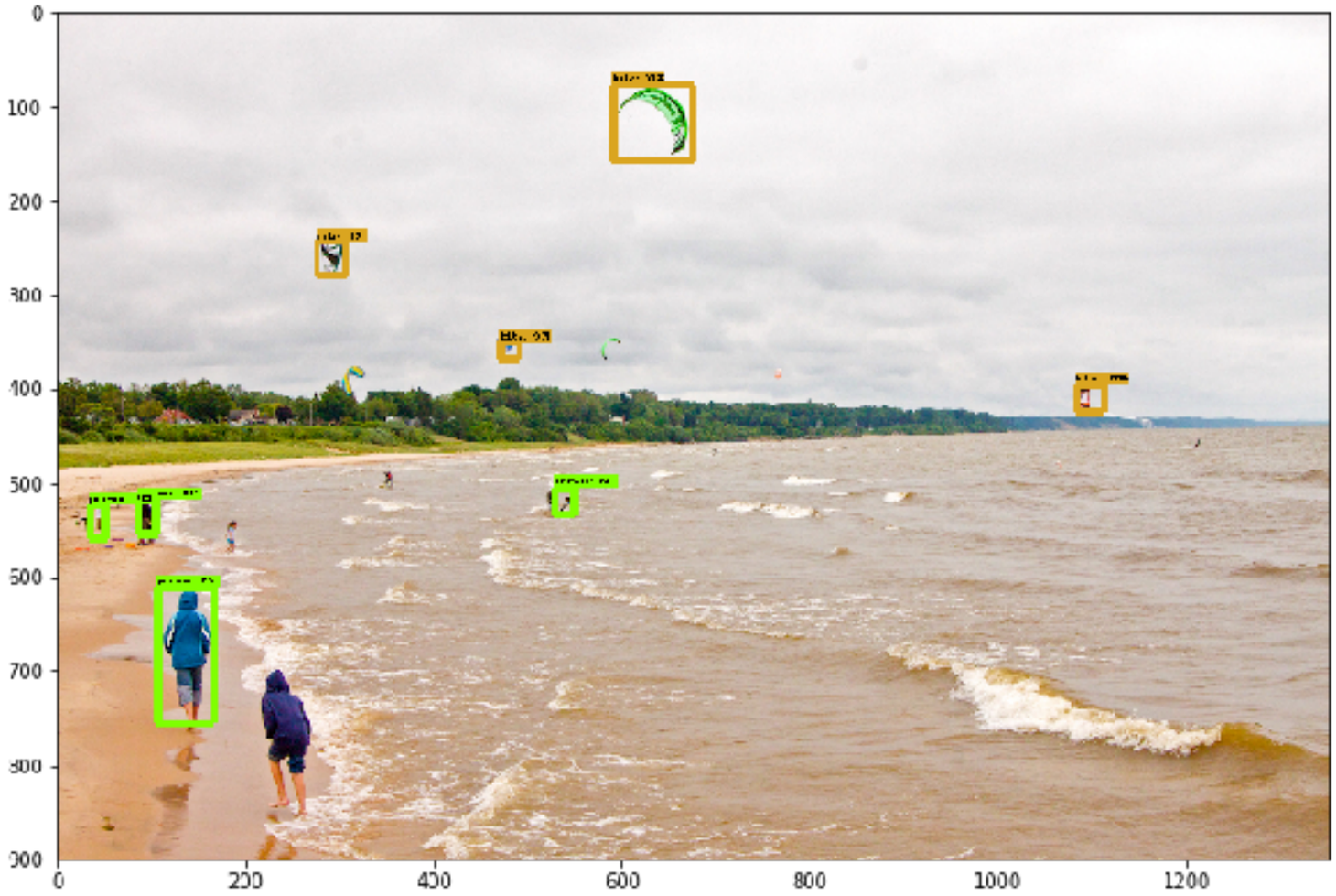
1. git clone https://github.com/tensorflow/models.git

2.
```
# From tensorflow/models/
protoc object_detection/protos/*.proto --python_out=.
```

3.
```
# From tensorflow/models/
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

tensorflow object detection pretrained model 활용





object_detection/object_detection_tutorial.ipynb -> 실습파일

Custom 데이터를 이용한 학습

1. 데이터 준비

Standard Tensorflow Format : TFRecord

In this post we will cover how to convert a dataset into *.tfrecord* file. Binary files are sometimes easier to use, because you don’t have to specify different directories for images and groundtruth annotations. While storing your data in binary file, you have your data in one block of memory, compared to storing each image and annotation separately. Openning a file is a considerably time-consuming operation especially if you use *hdd* and not *ssd*, because it involves moving the disk reader head and that takes quite some time. Overall, by using binary files you make it easier to distribute and make the data better aligned for efficient reading.

Custom 데이터를 이용한 학습

1. 데이터 준비

Dataset Requirements

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

1. An RGB image for the dataset encoded as jpeg or png.
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.

Custom 데이터를 이용한 학습

1. 데이터 준비

class labeling data

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


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

Custom 데이터를 이용한 학습

1. 데이터 준비

```
def create_cat_tf_example(encoded_cat_image_data):
  """Creates a tf.Example proto from sample cat image.

  Args:
    encoded_cat_image_data: The jpg encoded data of the cat image.

  Returns:
    example: The created tf.Example.
  """

  height = 1032.0
  width = 1200.0
  filename = 'example_cat.jpg'
  image_format = b'jpg'

  xmins = [322.0 / 1200.0]
  xmaxs = [1062.0 / 1200.0]
  ymins = [174.0 / 1032.0]
  ymaxs = [761.0 / 1032.0]
  classes_text = ['Cat']
  classes = [1]

  tf_example = tf.train.Example(features=tf.train.Features(feature={
      'image/height': dataset_util.int64_feature(height),
      'image/width': dataset_util.int64_feature(width),
      'image/filename': dataset_util.bytes_feature(filename),
      'image/source_id': dataset_util.bytes_feature(filename),
      'image/encoded': dataset_util.bytes_feature(encoded_image_data),
      'image/format': dataset_util.bytes_feature(image_format),
      'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
      'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
      'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
      'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
      'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
      'image/object/class/label': dataset_util.int64_list_feature(classes),
  }))
  return tf_example
```

tfrecord file 생성

Custom 데이터를 이용한 학습

1. 데이터 준비

pascal voc 2012 data download

- wget http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar

데이터 구조

```
+VOCdevkit
  +VOC2012
     +Annotations
     +ImageSets
        +Action
        +Layout
        +Main
        +Segmentation
     +JPEGImages
     +SegmentationClass
     +SegmentationObject
```

Custom 데이터를 이용한 학습

1. 데이터 준비

PASCAL VOC data - > tfrecord 파일로 변환

train data 생성

```
python object_detection/create_pascal_tf_record.py \
    --label_map_path=object_detection/data/pascal_label_map.pbtxt \
    —data_dir=/데이터 경로/VOCdevkit --year=VOC2012 --set=train \
    —output_path=pascal_train.record
```

test data 생성

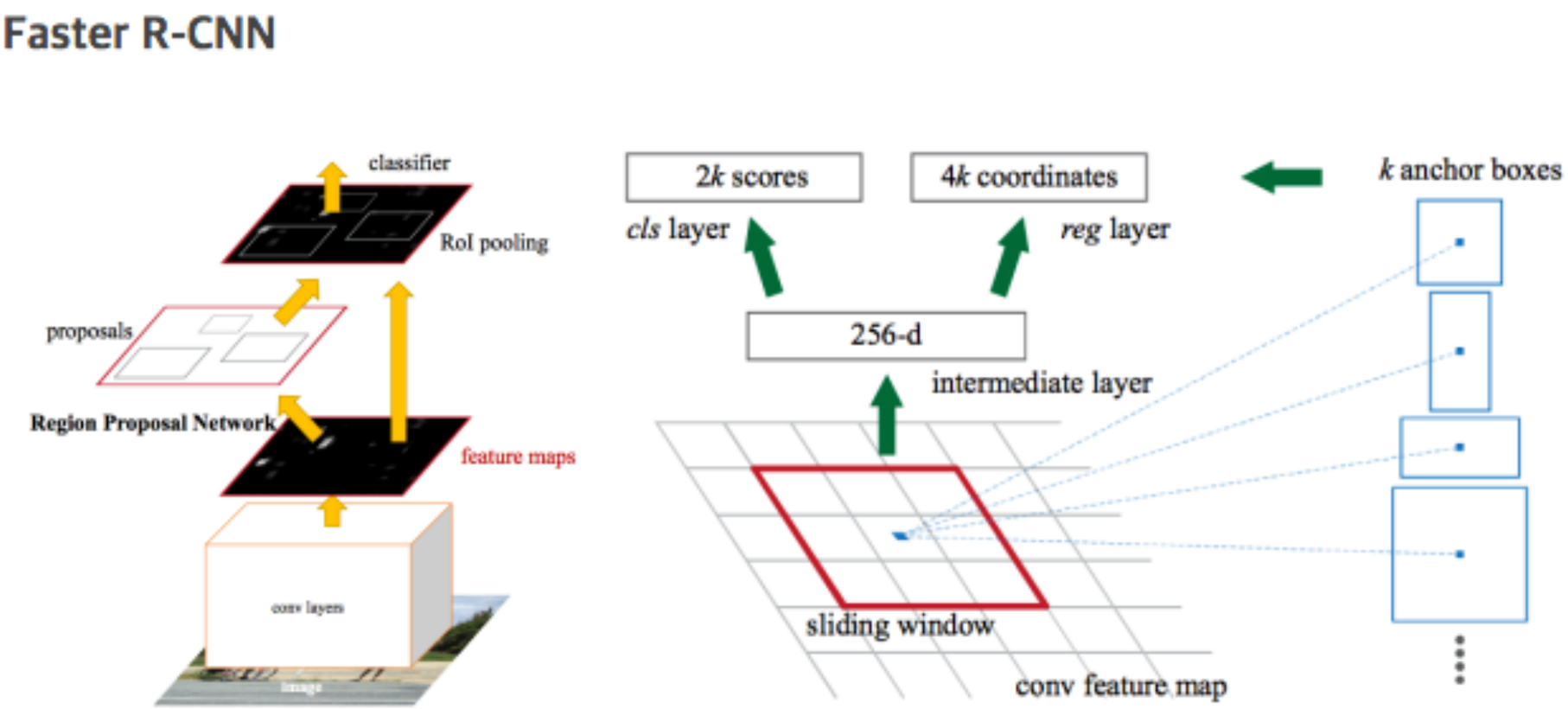
```
python object_detection/create_pascal_tf_record.py \
    --label_map_path=object_detection/data/pascal_label_map.pbtxt \
    —data_dir=/데이터 경로/VOCdevkit --year=VOC2012 --set=val \
    —output_path=pascal_val.record
```

object_detection/create_pascal_tf_record.py 파일 참고

Custom 데이터를 이용한 학습

2. 네트워크 모델링

config 파일 설정을 통해 빠른 모델링 가능
object_detection/samples/configs 폴더 안에 config 파일 예시



Custom 데이터를 이용한 학습

2. 네트워크 모델링

```
model {
  faster_rcnn {
    num_classes: 20
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet50'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
```

- 모델 설정
- input data 설정
- hyper parameter 설정
- regularizer 설정
- initializer 설정

Custom 데이터를 이용한 학습

2. 네트워크 모델링

```
train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "object_detection/train/model.ckpt-20000"
  from_detection_checkpoint: true
  # Note: The below line limits the training process to 200K steps, which we
  # empirically found to be sufficient enough to train the pets dataset. This
  # effectively bypasses the learning rate schedule (the learning rate will
  # never decay). Remove the below line to train indefinitely.
  num_steps: 100000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}
```

- learning rate 설정
- optimizer 설정
- iteration 설정
- data augmentation 설정

Custom 데이터를 이용한 학습

2. 네트워크 모델링

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "pascal_train.record"
  }
  label_map_path: "object_detection/data/pet_label_map.pbtxt"
}

eval_config: {
  num_examples: 2000
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "pascal_val.record"
  }
  label_map_path: "object_detection/data/pet_label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```

- 데이터 경로 설정

Custom 데이터를 이용한 학습

3. 학습

```
python object_detection/train.py \
    --logtostderr \
    --pipeline_config_path=object_detection/samples/configs/faster_rcnn_resnet50_pascal.config \
    --train_dir=object_detection/train
```

```
INFO:tensorflow:global step 301: loss = 3.8821 (0.332 sec/step)
INFO:tensorflow:global step 302: loss = 3.2983 (0.288 sec/step)
INFO:tensorflow:global step 303: loss = 1.3917 (0.323 sec/step)
INFO:tensorflow:global step 304: loss = 0.8608 (0.374 sec/step)
INFO:tensorflow:global step 305: loss = 2.9666 (0.648 sec/step)
INFO:tensorflow:global step 306: loss = 1.0907 (0.311 sec/step)
INFO:tensorflow:global step 307: loss = 1.5099 (0.292 sec/step)
INFO:tensorflow:global step 308: loss = 0.3442 (0.314 sec/step)
INFO:tensorflow:global step 309: loss = 1.8612 (0.288 sec/step)
INFO:tensorflow:global step 310: loss = 2.2088 (0.309 sec/step)
INFO:tensorflow:global step 311: loss = 1.1319 (0.325 sec/step)
INFO:tensorflow:global step 312: loss = 0.1494 (0.324 sec/step)
INFO:tensorflow:global step 313: loss = 4.6393 (0.301 sec/step)
INFO:tensorflow:global step 314: loss = 0.2084 (0.317 sec/step)
INFO:tensorflow:global step 315: loss = 1.1099 (0.306 sec/step)
INFO:tensorflow:global step 316: loss = 1.3451 (0.308 sec/step)
INFO:tensorflow:global step 317: loss = 1.4141 (0.300 sec/step)
INFO:tensorflow:global step 318: loss = 2.4797 (0.319 sec/step)
INFO:tensorflow:global step 319: loss = 1.4936 (0.499 sec/step)
INFO:tensorflow:global_step/sec: 2.66987
INFO:tensorflow:global step 320: loss = 3.0718 (0.381 sec/step)
INFO:tensorflow:Recording summary at step 320.
INFO:tensorflow:global step 321: loss = 0.5927 (0.315 sec/step)
INFO:tensorflow:global step 322: loss = 3.0717 (0.302 sec/step)
INFO:tensorflow:global step 323: loss = 1.1572 (0.305 sec/step)
INFO:tensorflow:global step 324: loss = 1.0379 (0.279 sec/step)
INFO:tensorflow:global step 325: loss = 0.8246 (0.271 sec/step)
INFO:tensorflow:global step 326: loss = 0.5936 (0.305 sec/step)
INFO:tensorflow:global step 327: loss = 2.0112 (0.317 sec/step)
INFO:tensorflow:global step 328: loss = 3.6632 (0.341 sec/step)
```

Custom 데이터를 이용한 학습

3. 학습

checkpoint file -> protobuf file

```
python object_detection/export_inference_graph \
    --input_type image_tensor \
    --pipeline_config_path ${PIPELINE_CONFIG_PATH} \
    --checkpoint_path model.ckpt-${CHECKPOINT_NUMBER} \
    --inference_graph_path output_inference_graph.pb
```

설정한 경로에 pb 파일 생성

Custom 데이터를 이용한 학습

4. 학습결과 확인

object_detection/object_detection_pascal.ipynb 파일 참고