섹션 7. TensorFlow 2.0을 이용한 YOLO 논문 구현 2_datset.py, model.py, evaluate.py

dataset.py

```
# process ground-truth data for YOLO format
def process_each_ground_truth(original_image,
                              bbox,
                              class_labels,
                              input_width,
                              input_height
                              ):
```

- train.py에서 처리하는 데이터를 원하는 형상의 ground-truth로 변경해주는 로직
- `original_image` : resize하기 전 원래 이미지의 크기

```
    image = original_image.numpy()
    image = zero_trim_ndarray(image)
```

- `original_image.numpy()` : tensor type이기 때문에 numpy array형태로 변환
- `zero_trim_ndarray(image)` : 0인 부분을 잘라 주어서 실제 이미지가 있는 부분만 반환

```
    # set original width height
    original_h = image.shape[0]
    original_w = image.shape[1]

    width_rate = input_width * 1.0 / original_w
    height_rate = input_height * 1.0 / original_h

    image = tf.image.resize(image, [input_height, input_width])
```

- `original_w` : original image의 width, `original_h` : original image의 hight
- `width_rate` : yolo input으로 넣기 위해 resizing하는데, resizing된 image의 width 대비 original image의 width
- `tf.image.resize(image, [input_height, input_width])` : 다양한 사이즈의 original image를 yolo에 넣기 위해 고정 사이즈로 변환

```
    object_num = np.count_nonzero(bbox, axis=0)[0]
    labels = [[0, 0, 0, 0, 0]] * object_num
    for i in range(object_num):
      xmin = bbox[i][1] * original_w
      ymin = bbox[i][0] * original_h
      xmax = bbox[i][3] * original_w
      ymax = bbox[i][2] * original_h

      class_num = class_labels[i]

      xcenter = (xmin + xmax) * 1.0 / 2 * width_rate
      ycenter = (ymin + ymax) * 1.0 / 2 * height_rate

      box_w = (xmax - xmin) * width_rate
      box_h = (ymax - ymin) * height_rate

      labels[i] = [xcenter, ycenter, box_w, box_h, class_num]
```

- `np.count_nonzero(bbox, axis=0)[0]` : 전체 bounding box에서 0이 아닌 실제 object가 있는 개수 count
- `[[0, 0, 0, 0, 0]] * object_num` : [[xcenter (Absolute Coordinate), ycenter (Absolute Coordinate), w (Absolute Coordinate), h (Absolute Coordinate), class_num]] * object_num
- `for i in range(object_num)` : object 개수만큼 for loop
- `bbox[i][1] * original_w` : 상대 좌표 값을 절대 좌표로 변경

```
  return [image.numpy(), labels, object_num]
```

- 최종적으로 return하는 값은 이미지 resizing 값, labels의 절대 좌표, object 개수

model.py

```
# Implementation using tf.keras.applications (https://www.tensorflow.org
/api_docs/python/tf/keras/applications)
# & Keras Functional API (https://www.tensorflow.org/guide/keras
/functional)

class YOLOv1(tf.keras.Model):
  def __init__(self, input_height, input_width, cell_size,
boxes_per_cell, num_classes):
    super(YOLOv1, self).__init__()
    base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet', input_shape=(input_height, input_width, 3))
    base_model.trainable = True
    x = base_model.output

    # Global Average Pooling
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    output = tf.keras.layers.Dense(cell_size * cell_size * (num_classes
+ (boxes_per_cell*5)), activation=None)(x)
    model = tf.keras.Model(inputs=base_model.input, outputs=output)
    self.model = model
    # print model structure
    self.model.summary()

  def call(self, x):
    return self.model(x)
```

- tf.keras.applications를 이용해 미리 pre-trained된 InceptionV3 모델을 가져옴
- `InceptionV3(include_top=False` : 전체 layer에서 마지막 feature map 전까지만 가져옴
- `base_model.trainable = True` : 전체 InceptionV3에서 마지막 feature map을 제외하고 앞의 모델을 변경할 수 있음. 이렇게 하면 속도는 조금 느려지지만 성능은 더 좋아짐. False면 앞의 모델을 변경할 수 없음.
- `GlobalAveragePooling2D()` : feature map을 하나의 스칼라 벡터로 모아줌. pooling을 전체의 feature map에 대해 수행

evaluate.py

```
# set voc label dictionary
cat_label_to_class_dict = {
  0:"cat"
}
cat_class_to_label_dict = {v: k for k, v in cat_label_to_class_dict.
items()}
```

- train.py에서와 마찬가지로 'cat' label에 대해서만 값을 추출

```
# checkpoint restore  checkpoint file  path
flags.DEFINE_string('checkpoint_path', default='saved_model',
help='path to a directory to restore checkpoint file')
# test directory path
flags.DEFINE_string('test_dir', default='test_result', help='directory
which test prediction result saved')

FLAGS = flags.FLAGS
```

- `test directory path` : evaluate.py script에서 test data에 대한 결과를 image 형태로 저장할 때, 이 경로를 지정

```
# set configuration value
batch_size = 1
input_width = 224 # original paper : 448
input_height = 224 # original paper : 448
cell_size = 7
num_classes = 1 # original paper : 20
boxes_per_cell = 2
```

- train.py에서와 동일하게 configuration value 지정 해줘야 함

```
if __name__ == '__main__':
  app.run(main)
```

- main function을 호출하면, Flags가 포함된 형태로 main function을 실행

main function

```
def main(_):
  # check if checkpoint path exists
  if not os.path.exists(FLAGS.checkpoint_path):
    print('checkpoint file is not exists!')
    exit()
```

- evaluation은 무조건 check point를 restore해서 진행해야 하기 때문에 checkpoint_path가 없으면 에러 메시지 출력하고 프로그램을 종료 시킴

```
  num_images = len(list(test_data))  # batch_size = 1
  print('total test image :', num_images)
  for image_num, features in enumerate(test_data):
    batch_image = features['image']
    batch_bbox = features['objects']['bbox']
    batch_labels = features['objects']['label']

    batch_image = tf.squeeze(batch_image, axis=1)
    batch_bbox = tf.squeeze(batch_bbox, axis=1)
    batch_labels = tf.squeeze(batch_labels, axis=1)

    image, labels, object_num = process_each_ground_truth(batch_image
[0], batch_bbox[0], batch_labels[0], input_width, input_height)
```

- `for image_num, features in enumerate(test_data)` : 전체 데이터에 대한 image 개수를 셈
- `batch_image = features['image']` : 정답 데이터에 대한 값들을 parsing해서 가져옴
- `batch_image = tf.squeeze(batch_image, axis=1)` : 불필요한 차원을 제거
- `process_each_ground_truth` : function을 이용하여 한 장의 이미지에 대한 값들을 pasing해서 가져옴

```
    predict = YOLOv1_model(image)
    predict = reshape_yolo_preds(predict)

    predict_boxes = predict[0, :, :, num_classes + boxes_per_cell:]
    predict_boxes = tf.reshape(predict_boxes, [cell_size, cell_size,
boxes_per_cell, 4])

    confidence_boxes = predict[0, :, :, num_classes:num_classes +
boxes_per_cell]
    confidence_boxes = tf.reshape(confidence_boxes, [cell_size,
cell_size, boxes_per_cell, 1])

    class_prediction = predict[0, :, :, 0:num_classes]
    class_prediction = tf.argmax(class_prediction, axis=2)
```

- `reshape_yolo_preds(predict)` : 예측한 값을 parsing할 수 있는 형태로 reshape해 줌