Title: README Setup YOLACT GPU

Document Number: nnn-n

CTI One Corporation

Table 1a. Document History

2022-09-12	Establish this document, document archive: /media/harry/easystore1/backup-2020-2-15/CTI/3proejcts/3-8-smart-tech/3-8-4-CTI/3-8-4-6-products/CV100/102-algorithm-source-code-documents/102n-yolact/102n-1-readme-yolact$	YY, HL
2022-11-22	Update the downloading COCO dataset part	YY

Table 1b. Testing and Release Approval Form

2022-09-12	Tested by YY and approved for release by HL	Pending for testing and approval

Table 2. References

Number	Name and URL	Note
1.	YOLACT https://github.com/dbolya/yolact	Original, Pytorch Implementation
2.	YOLACT TensorFlow https://github.com/anshkumar/yolact	

3.	YOLACT TensorFlow https://github.com/leohsuofnthu/Tensorflow-YOLACT	
4.	Mask R-CNN for Object Detection and Segmentation using TensorFlow 2.0 https://github.com/ahmedfgad/Mask-RCNN-TF2	
5.	READEME Setup YOLACT GPU nnn-n-README-YOLACT-GPU-v1-YY-2022-9-12.odt	To create the YOLACT environment
6.	README Create COCO format annotation nnn-n-README-Create-COCO-Format-Annotation-ZW-YY-2022-09-12.odt	

Table 3. Prerequisite

Software Prerequisite No.	Description and Version	Note
1.	Ubuntu 18.04	
2.	Anaconda version 4.7.12 or later	On Ubuntu
3.	git version 2.17.1 or later	
4.	Anaconda environment for YOLACT	
Hardware Prerequisite No.	Description and Version	
1.	NVIDIA GPU	

There are some ways to prepare the YOLACT training dataset which contains image files and annotation data. One way is to generate the annotation file from own image files, another is using the existing dataset such as COCO dataset or ImageNet dataset. This document contains using the generated own dataset and COCO dataset.

If you do not have the Anaconda environment for YOLACT and the YOLACT Github code, see 102n-1a-README-YOLACT-GPU-v1-YY-2022-9-12.odt.

1. Compile Protobuf library

1.1. Install Protobuf compiler;

$ sudo apt-get install protobuf-compiler

1.2. Open terminal in yolact folder and activete the YOLACT environment;

$ conda activate yolact

1.3. Compile Protobuf library;

$ protoc protos/*.proto --python_out=.

string_int_label_map_pb2.py files will be in ./protos folder

1.4. Install Python packages

$ pip install scikit-image==0.17.2

$ pip install keras==2.3.0

2. Training YOLACT model using own dataset

2.1. Generate COCO format annotation file (.json) from your own image files for training and validation;

See 102n-1b-README-Create-COCO-Format-Annotation-ZW-YY-2022-09-12.odt

2.2. Download coco_tfrecord_creator.py and coco_tfrecord_utils.py from the below URL link and place it to “yolact/data” folder;

https://github.com/leohsuofnthu/Tensorflow-YOLACT/tree/master/data

2.3. Open coco_tfrecord_creator.py and modify the following lines;

133 'image/object/class/label':

209 FLAGS.include_masks,
210 num_shards=1)

216 FLAGS.include_masks,
217 num_shards=1)

2.4. Create folders as “train2014”, “val2014”, “annotations”, “coco” in yolact/data folder;

2.5. Put training images to train2014 folder, put validation images to val2014 folder, put training annotation file to annotation folder as “cti_train2014.json”, put validating annotation file to annotation as “cti_val2014.json”;

After putting the files, the data folder looks like below

```
data
├── annotations
│   ├── cti_train2014.json
│   └── cti_val2014.json
├── coco
├── coco_tfrecord_creator.py
├── coco_tfrecord_utils.py
├── train2014
│   └── test.jpg
└── val2014
    └── test.jpg
```

2.6. Convert the COCO format annotation files and image files to TF Record format file. Open a terminal in the YOLACT root folder and activate the YOLACT Anaconda environment;

$ conda activate yolact

$ python -m data.coco_tfrecord_creator -train_image_dir './data/train2014' -val_image_dir './data/val2014' -train_annotations_file './data/annotations/cti_train2014.json' -val_annotations_file './data/annotations/cti_val2014.json' -output_dir './data/coco'

After converting is finished, the ouput files are in data/coco folder

2.7. Create “train” and “val” folders in ./data/coco folder. Copy train.record-00000-of-00001 file to the train folder, and val.record-00000-of-00001 file to val folder;

2.8. Create cti_label_map.pbtxt in data folder, which contains the following code;

```
item {
  name: "person"
  id: 1
}
item {
  name: "dummy"
  id: 2
}
```

2.9. Execute the YOLACT training program;

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '10000' -lr_total_steps '10000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 300 -valid_iter 500
```

If -train_iter or -lr_total_steps is very small like 100 or 1000, it will cause an error.

The model will be created in saved_models folder and checkpoints files will be created in checkpoints folder.

To train the human detection, the below command was used with 1,479 training images and 444 validataion images.

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '50000' -lr_total_steps '50000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 5000 -valid_iter 500
```

3. Training YOLACT model using COCO 2014 dataset

Using COCO dataset senario is 1. Extract only human(person) images from COCO dataset; 2. Choose indoor images manually; 3. Extract annotation relating the indoor image which are chosen in step 2.; 4. Convert COCO data format annotation and images to TF record annotation;

COCO 2014 dataset requires 42 GB storage space to download.

https://cocodataset.org/#download

3.1. Download COCO 2014 dataset;

$ wget http://images.cocodataset.org/zips/train2014.zip

$ wget http://images.cocodataset.org/zips/val2014.zip

$ wget http://images.cocodataset.org/annotations/annotations_trainval2014.zip

$ unzip train2014.zip

$ unzip val2014.zip

$ unzip annotations_trainval2014.zip

The relation of image folders and annotation files

train2014: annotatinos/instances_train2014.json

val2014: annotatinos/instances_val2014.json

3.2. Copy cocodata folder to yolact/data folder;

3.3. Create extract_coco_images.py in yolact/data/cocodata. Copy and paste the follow lines to extract_coco_images.py;

```
from pycocotools.coco import COCO
import os

INPUT_FILE_PATH = "train2014"      # train2014 or  val2014
OUTPUT_FILE_PATH = "./cti_train2014"  # "cti_train2014" or "cti_val2014"

# INPUT_FILE_PATH = "val2014"      # train2014 or  val2014
# OUTPUT_FILE_PATH = "./cti_val2014"  # "cti_train2014" or "cti_val2014"
```

```
coco = COCO('./annotations/instances_' + INPUT_FILE_PATH + '.json')
# Specify a list of category names of interest
catIds = coco.getCatIds(catNms=['person'])
# Get the corresponding image ids and images using loadImgs
imgIds = coco.getImgIds(catIds=catIds)
images = coco.loadImgs(imgIds)

# Create output folder
if not os.path.exists(OUTPUT_FILE_PATH):
    os.mkdir(OUTPUT_FILE_PATH)

print("imageCount:", len(images))
print("Type(images)", type(images))

for index, image in enumerate(images):
    print("file_name", image["file_name"])

    # if index <= 350:  # if skip images
    #     continue

    inputFile = open(INPUT_FILE_PATH + "/" + image["file_name"], "rb")
    outFile = open(OUTPUT_FILE_PATH + "/" + image["file_name"], "wb")
    outFile.write(inputFile.read())

    inputFile.close()
    outFile.close()

    if index >= 1000:    # index starts 0
        break
```

3.4. Execute extract_coco_images.py in cocodata folder on yolact Anaconda environment;

$ python extract_coco_images.py

The extracted images are in cti_train2014 folder.

3.5. Modify extract_coco_images.py like the below lines to extract validation image files;

```
21 # INPUT_FILE_PATH = "train2014"       # train2014 or  val2014
22 # OUTPUT_FILE_PATH = "./cti_train2014"  # "cti_train2014" or "cti_val2014"
23
24 INPUT_FILE_PATH = "val2014"       # train2014 or  val2014
25 OUTPUT_FILE_PATH = "./cti_val2014"  # "cti_train2014" or "cti_val2014"
```

3.6. Execute extract_coco_images.py again;

$ python extract_coco_images.py

The extracted images are in cti_val2014 folder.

3.7. Check and copy image files from cti_train2014 and cti_val2014 folder to cti_train2014_indoor and cti_val2014_indoor respectively;

3.8. Create extract_coco_annotation.py in yolact/data/cocodata. Copy and paste the follow lines to extract_coco_annotation.py;

```
'''-------------------------------------------------------------------------------------------------
* Company Name : CTI One Corporation                                                    *
* Program name : extract_coco_annotation.py (Testing)                                       *
* Coded By     : YY                                                                 *
* Date         : 2022-08-04                                                           *
* Updated By   :                                                                    *
* Date         :                                                                  *
* Version      : v1.0.0                                                             *
* Copyright    : Copyright (c) 2022 CTI One Corporation                                     *
* Purpose      : Extract annotations from COCO dataset using image file names                *
*              :                                                                  *
*              : v1.0.0 2022-08-04 YY Create                                          *
-------------------------------------------------------------------------------------------------'''
# https://stackoverflow.com/questions/51100191/how-can-i-download-a-specific-part-of-coco-dataset
# https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch#:~:text=The%20COCO
%20dataset%20is%20formatted,%E2%80%9D%20(in%20one%20case).
# https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocotools/coco.py
# To extract annotation by image file names
# Category ID is "1", person

from pycocotools.coco import COCO
import json
import glob

COCO_FILE_NAME = './annotations/instances_train2014.json'   # instances_train2014.json or
instances_val2014.json
# COCO_FILE_NAME = './annotations/instances_val2014.json'   # instances_train2014.json or
instances_val2014.json

TARGET_IMAGE_FOLDER = './cti_train2014_indoor/*'      # image file names have to be
'COCO_xxx_00000123.jpg'
# TARGET_IMAGE_FOLDER = './cti_val2014_indoor/*'      # './cti_train2014_indoor/*' or './cti_val2014_indoor/*'

OUTPUT_FILE_NAME = './cti_coco_train2014.json'          # When Mask RCNN traing program run, need to be
changed to instances_train2014.json or instances_minival2014.json
# OUTPUT_FILE_NAME = './cti_coco_val2014.json'                                          # cti_train2014.json or
cti_val2014.json

# Load COCO data file
coco = COCO(COCO_FILE_NAME)

category_list = []      # "categories" information in COCO dataset annotation file
```

```
image_info_list = []    # "images" information in COCO dataset annotation file
annotation_list = []    # "annotations" information in COCO dataset annotation file

# Get "categories" information
catIds = coco.getCatIds(catNms=['person'])
category_list = coco. loadCats(catIds)

# Get "images" information
# Get the image file names at the target image folder
img_file_names = glob.glob(TARGET_IMAGE_FOLDER)

for image_file_name in img_file_names:
    # Get file name from './cti_train2014/COCO_train2014_000000000735.jpg', to
'COCO_train2014_000000000735.jpg'
    file_name = image_file_name.split("/")[-1]

    # Get image file ID from filename 'COCO_train2014_000000000735.jpg' to '000000000735'
    split_file_names = file_name.split("_")
    split_file_name = split_file_names[-1].split(".")

    img_Ids = coco.getImgIds(imgIds=[int(split_file_name[0]), ])
    image_info_list.append(coco.imgs[img_Ids[0]])

    # Get "annotations" information
    annotation_ids = coco.getAnnIds(imgIds=img_Ids)
    annotations = coco.loadAnns(annotation_ids)
    for annotation in annotations:

        # print("type(annotation):", type(annotation))
        # print("annotation[category_id]:", annotation['category_id'])
        # print("coco.loadCats(annotation[\'category_id\'])[\"name\"]:",
coco.loadCats(annotation['category_id']))

        # Add segmentation data if category id is 1, person
        if annotation['category_id'] == 1:
            annotation_list.append(annotation)

# Create a dictionary of the dataset JSON file
coco_dict = {"categories": category_list,
             "images": image_info_list,
             "annotations": annotation_list}

# Write the JSON file
with open(OUTPUT_FILE_NAME, "w") as outfile:
    json.dump(coco_dict, outfile)
```

3.9. Execute extract_coco_annotation.py in cocodata folder on yolact Anaconda environment;

$ python extract_coco_annotation.py

cti_coco_train2014.json are generated in cocodata folder.

3.10. Modify extract_coco_annotation.py like the below lines to extract validation annotations;

```
24 # COCO_FILE_NAME = './annotations/instances_train2014.json'   # instances_train2014.json or instances_val2014.json
25 COCO_FILE_NAME = './annotations/instances_val2014.json'   # instances_train2014.json or instances_val2014.json
26
27 # TARGET_IMAGE_FOLDER = './cti_train2014_indoor/*'      # image file names have to be 'COCO_xxx_00000123.jpg'
28 TARGET_IMAGE_FOLDER = './cti_val2014_indoor/*'      # './cti_train2014_indoor/*' or './cti_val2014_indoor/*'
29
30 # OUTPUT_FILE_NAME = './cti_coco_train2014.json'          # When Mask RCNN traing program run, need to be changed to instances_train2014.json or instances_minival2014.json
31 OUTPUT_FILE_NAME = './cti_coco_val2014.json'
```

3.11. Execute extract_coco_annotation.py again;

$ python extract_coco_annotation.py

cti_coco_val2014.json are generated in cocodata folder.

3.12. Copy cti_train2014_indoor and cti_val2014_indoor folders to yolact/data, and change the folder name as train2014 and val2014 respectively;

3.13. Copy cti_coco_train2014.json and cti_coco_val2014.json to yolact/data/annotations, and change the file name as cti_train2014.json and cti_val2014.json respectively;

3.14. Convert the image files and COCO format annotation files to TF record format file;

In a terminal, navigate to yolact folder, and excute data/coco_tfrecord.py

$ python -m data.coco_tfrecord_creator -train_image_dir './data/train2014' -val_image_dir './data/val2014' -train_annotations_file './data/annotations/cti_train2014.json' -val_annotations_file './data/annotations/cti_val2014.json' -output_dir './data/coco'

3.15. Create “train” and “val” folders in ./data/coco folder. Copy train.record-00000-of-00001 file to the train folder, and val.record-00000-of-00001 file to val folder;

3.16. Create cti_label_map.pbtxt in data folder, which contains the following code;

```
item {
  name: "person"
  id: 1
```

}
item {
  name: "dummy"
  id: 2
}

3.17. Execute the YOLACT training program

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '10000' -lr_total_steps '10000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 300 -valid_iter 500
```

If -train_iter or -lr_total_steps is very small like 100 or 1000, it will cause an error.

The model will be created in saved_models folder and checkpoints files will be created in checkpoints folder.

To train the human detection, the below command was used with 1,479 training images and 444 validataion images.

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '50000' -lr_total_steps '50000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 5000 -valid_iter 500
```

4. Training YOLACT model using COCO dataset and own dataset

You have image files and COCO format annotation files from COCO dataset, and your own dataset;

cti_coco_train2014.json: Training annotation file from COCO dataset

cti_coco_val2014.json: Validataion annotation file from COCO dataset

cti_train2014_indoor: Folder which contains trainging image files from COCO dataset

cti_val2014_indoor: Folder which contains validation image files from COCO dataset

cti_train2014_own.json: Training annotation file from own dataset

cti_val2014_own.json: Validataion annotation file from own dataset

train2014_own: Folder which contains trainging image files from own dataset

val2014_own: Folder which contains validation image files from own dataset

4.1. Put all annotation files into yolact/data/cocodata;

4.2. Create combine_annotation_files.py in yolact/data/cocodata. Copy and paste the follow lines to combine_annotation_files.py;

```
'''-------------------------------------------------------------------------------------------------
* Company Name : CTI One Corporation                                                              *
* Program name : combine_annotation_files.py (Testing)                                                *
* Coded By     : YY                                                                           *
* Date         : 2022-08-04                                                                     *
* Updated By   :                                                                              *
* Date         :                                                                              *
* Version      : v1.0.0                                                                       *
* Copyright    : Copyright (c) 2022 CTI One Corporation                                                *
* Purpose      : Combine COCO format annotation files                                                  *
*            :                                                                                *
*            : v1.0.0 2022-08-04 YY Create                                                          *
-------------------------------------------------------------------------------------------------'''
import json



INPUT_ANNOTATION_FILES = ['./cti_coco_train2014.json',
                          './cti_train2014_own.json',
                          ]
OUTPUT_ANNOTATION_FILE = "./cti_train2014.json"

'''
INPUT_ANNOTATION_FILES = ['./cti_coco_val2014.json',
```

```
                    './cti_val2014_own.json',
                    ]
OUTPUT_ANNOTATION_FILE = "./cti_val2014.json"
'''
category_list = []      # "categories" information in COCO dataset annotation file
image_info_list = []    # "images" information in COCO dataset annotation file
annotation_list = []    # "annotations" information in COCO dataset annotation file



# Get Category_list
with open(INPUT_ANNOTATION_FILES[0]) as firstFile:
    firstFileJSON = json.load(firstFile)
    category_list = firstFileJSON['categories']
    print("firstFileJSON:", firstFileJSON['categories'])

# Get image info list and annotation list
for fileName in INPUT_ANNOTATION_FILES:
    with open(fileName) as annotationFile:
        annotationFileJSON = json.load(annotationFile)
        imageInfoJSONList = annotationFileJSON['images']
        annotationJSONList = annotationFileJSON['annotations']


        for imageInfo in imageInfoJSONList:
            image_info_list.append(imageInfo)


        for annotation in annotationJSONList:
            annotation_list.append(annotation)


# for fileName

# Create a dictionary of the dataset JSON file
coco_dict = {"categories": category_list,
             "images": image_info_list,
             "annotations": annotation_list}


# Write the JSON file
with open(OUTPUT_ANNOTATION_FILE, "w") as outfile:
    json.dump(coco_dict, outfile)
```

4.3. Open a terminal in yolact/data/cocodata folder and activate YOLACT Anaconda environment;

$ conda activate yolact

4.4. Execute combine_annotation_files.py;

$ python combine_annotation_files.py

cti_train2014.json is generated.

4.5. Modify combine_annotation_files.py as the below lines to combine validation annotation files;

```
16 '''
17 INPUT_ANNOTATION_FILES = ['./cti_coco_train2014.json',
18                          './cti_train2014_own.json',
19                          ]
20 OUTPUT_ANNOTATION_FILE = "./cti_train2014.json"
21 '''
22
23 INPUT_ANNOTATION_FILES = ['./cti_coco_val2014.json',
24                          './cti_val2014_own.json',
25                          ]
26 OUTPUT_ANNOTATION_FILE = "./cti_val2014.json"
```

4.4. Execute combine_annotation_files.py again;

$ python combine_annotation_files.py

cti_val2014.json will be generated.

4.5. Copy all training image files to yolact/data/train2014, and all validation image files to yolact/data/val2014;

4.6. Copy cti_train2014.json and cti_val2014.json to yolact/data/annotations;

4.7. Convert the image files and COCO format annotation files to TF record format file. In a terminal, navigate to yolact folder, and excute data/coco_tfrecord.py;

$ python -m data.coco_tfrecord_creator -train_image_dir './data/train2014' -val_image_dir './data/val2014' -train_annotations_file './data/annotations/cti_train2014.json' -val_annotations_file './data/annotations/cti_val2014.json' -output_dir './data/coco'

4.8. Create “train” and “val” folders in yolact/data/coco folder. Copy yolact/data/coco/train.record-00000-of-00001 file to the yolact/data/coco/train folder, and yolact/data/coco/val.record-00000-of-00001 file to yolact/data/coco/val folder

4.9. Execute the YOLACT training program

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '10000' -lr_total_steps '10000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 300 -valid_iter 500
```

If -train_iter or -lr_total_steps is very small like 100 or 1000, it will cause an error.

The model will be created in saved_models folder and checkpoints files will be created in checkpoints folder.

To train the human detection, the below command was used with 1,479 training images and 444 validataion images.

```
$ python train.py -tfrecord_train_dir './data/coco/train' \
        -tfrecord_val_dir './data/coco/val' \
        -logs_dir './logs' -saved_models_dir './saved_models' \
        -label_map './data/cti_label_map.pbtxt' -train_iter '50000' -lr_total_steps '50000' \
        -img_h '550' -img_w '550' -batch_size '1' -num_class '2' -base_model_trainable True \
        -print_interval 100 -save_interval 5000 -valid_iter 500
```

5. TensorBoard

5.1. Open a terminal in yolact root folder and activate the YOLACT Anaconda envirnment;

$ conda activate yolact

5.2. Start TensorBoard

$ tensorboard --logdir=./logs

5.3. Access to http://localhost:6006/ on Firefox, not Google Chrome.

Appendix A. GPU Memory Error

When GPU memory shortage error, CUBLAS_STATUS_NOT_INITIALIZED error, or fail to initialize cuDNN, add the following line after import section in the Python code;

The follow code make the limit of 70% GPU memory usage.

```
from tensorflow.python.keras import backend as K
physical_devices = tf.config.list_physical_devices('GPU')
config = tf.compat.v1.ConfigProto(allow_soft_placement=True,
                                  intra_op_parallelism_threads=0,     # 2022-08-08 YY set 0 means, system decides
                                  inter_op_parallelism_threads=0)     # 2022-08-08 YY set 0 means system decides
config.gpu_options.per_process_gpu_memory_fraction = 0.7
config.gpu_options.allow_growth = False
sess = tf.compat.v1.Session(config=config)
K.set_session(sess)
```

(END)