YOLOv3 Documentation

Release 0.0.1

Anthony DeGennaro

Jan 02, 2019

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The following project is a Python implementation of the YOLOv3 object detection algorithm. This specific software began as a project intended for use with the Xview dataset specifically by Glenn Jocher at Ultralytics (https://github.com/ultralytics/xview-yolov3.git). It has since been modified extensively for the purposes of generality, maintainability, and usability by Anthony DeGennaro at Brookhaven National Laboratory (adegennaro@bnl.gov).

CHAPTER

TWO

REQUIREMENTS

This software requires Python 3.6, along with the following packages:

- numpy
- scipy
- sklearn
- matplotlib
- torch
- opencv-python
- h5py
- tqdm

CHAPTER

THREE

INSTALLATION

The purpose of this document is to provide detailed, step-by-step instructions on how to install Pytorch, YOLOv3, and all associated dependencies.

3.1 Anaconda

We first need to install Anaconda for Python virtual environments.

1. Download the Anaconda installer (shell script) from the Anaconda website:

   ```
   wget https://repo.anaconda.com/archive/Anaconda2-5.3.0-Linux-x86_64.sh
   ```

   Note: this assumes you have a 64-bit Linux architecture. If you have something else, then visit https://www.anaconda.com/download/ and select your preferred version.

2. Launch the Anaconda installer:

   ```
   bash Anaconda2-5.3.0-Linux-x86_64.sh
   ```

   Accept the user terms and accept the default filepath for installation, which should be `/home/[user]/anaconda2/`.

3. Open your `/.bashrc` file in a file editor (e.g., `emacs /.bashrc`) and paste the following line to the end:

   ```
   source /home/[user]/anaconda2/etc/profile.d/conda.sh
   ```

4. Save the `/.bashrc` file, exit, and reload it in your terminal with:

   ```
   source /.bashrc
   ```

5. Confirm conda was installed:

   ```
   conda --version
   ```

   This should output the version of the Anaconda install, if successful

6. Create a custom Anaconda virtual environment for this project:

   ```
   conda create -n [envname] python=3.6 anaconda
   ```

   In the above, replace [envname] with your desired environment name (do not include the brackets)

7. To verify that this was successful, run:

```
conda info --envs
```

If successful, [envname] should appear as one of the choices.

3.2 PyTorch

We will now install PyTorch, a Python deep-learning framework

1. Install PyTorch/Torchvision to your Anaconda environment:

```
conda install -n [envname] pytorch torchvision -c pytorch
```

2. To verify that this was successful, activate your conda environment:

```
conda activate [envname]
```

Then, check the PyTorch version with:

```
python -c "import torch; print(torch.__version__)"
```

Also check the Torchvision version with:

```
python -c "import torchvision; print(torchvision.__version__)"
```

If successful, both commands should output the installed versions.

3.3 GPU Support

If you have Nvidia GPU hardware but do not have the drivers installed, you may do so as follows. If you already have Nvidia drivers installed, skip this. Note: this may require sudo priveleges. Also, the following instructions assume a Redhat OS. The equivalent process for another Linux OS (e.g., Ubuntu) is very similar.

1. Prepare your machine by installing necessary prerequisite packages:

```
yum -y update

yum -y groupinstall "Development Tools"

yum -y install kernel-devel epel-release

yum install dkms
```

2. Download desired Nvidia driver version from their archive at https://www.nvidia.com/object/unix.html (e.g., using wget from the terminal)
3. If your machine is currently using open-source drivers (e.g., noveau), you will need to change the configuration `/etc/default/grub` file. Open this file, find the line beginning with `GRUB_CMDLINE_LINUX` and add the following text to it:

```
nouveau.modeset=0
```

4. Reboot your machine
5. Stop all Xorg servers:

```
systemctl isolate multi-user.target
```

6. Run the bash script installer:

```
bash NVIDIA-Linux-x86_64-*
```

7. Reboot your system
8. Confirm that the installation was successful by inspecting the output of this command:

```
nvidia-smi
```

If successful, this should display all Nvidia GPUs currently installed in your machine

3.4 YOLOv3

Note: For now, we are simply using a version of YOLOv3 freely available on Github. We plan to fork this and modify it as needed. For now, we only describe the installation directions for the community-available version of YOLOv3.

1. Activate your anaconda environment:

```
conda activate [envname]
```

2. Clone the YOLOv3 git repo:

```
git clone https://github.com/adegenna/xview-yolov3
```

3. Navigate to the project directory (xview-yolov3) and open the file requirements.txt. All of Python packages listed there must be installed to your local conda environment. Check whether the listed packages are installed with:

```
conda list | grep [package]
```

4. If one of the required packages is missing, then install it; for example, install opencv-python with:

```
conda install -n [envname] -c menpo opencv
```

CHAPTER

FOUR

CODE DOCS

4.1 Src/

src.train2.**main**(*inputs*)

Main driver script for training the YOLOv3 network.

Inputs:

inputs: an input file formatted according to the InputFile class

Outputs:

inputs.outdir/results.txt: output metrics for each training epoch
inputs.loaddir/latest.pt: checkpoint file for latest network configuration
inputs.loaddir/best.pt: checkpoint file for best current network configuration
inputs.loaddir/backup.pt: checkpoint file for backup purposes

class src.detect.**ConvNetb**(*num_classes=60*)

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class src.models.**Darknet**(*config_path*, *img_size=416*)

YOLOv3 object detection model

forward(*x*, *targets=None*, *requestPrecision=False*, *weight=None*, *epoch=None*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks

while the latter silently ignores them.

---

class src.models.**EmptyLayer**
Placeholder for 'route' and 'shortcut' layers

class src.models.**YOLOLayer**(*anchors*, *nC*, *img_dim*, *anchor_idxs*)

forward(*p*, *targets=None*, *requestPrecision=False*, *weight=None*, *epoch=None*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

src.models.**create_modules**(*module_defs*)
Constructs module list of layer blocks from module configuration in module_defs

src.models.**create_yolo_config_file**(*template_file_path*, *output_config_file_path*, *n_anchors*, *n_classes*, *anchor_coordinates*)
Creates a yolo-v3 layer configuration file from desired options

src.models.**parse_model_config**(*path*)
Parses the yolo-v3 layer configuration file and returns module definitions

class src.targets.Target.**Target**(*inputs*)
Class for handling target pre-processing tasks.

apply_mask_to_filtered_data()
Method to apply mask to filtered data variables.

compute_bounding_box_clusters_using_kmeans(*n_clusters*)

Method to compute bounding box clusters using kmeans.

Inputs:

n_clusters: number of desired kmeans clusters

compute_cropped_data()
Method to crop image data based on the width and height. Filtered variables are then computed based on the updated image coordinates.

compute_filtered_data_mask()
Method to compute filtered data by applying several filtering operations.

compute_filtered_variables_from_filtered_coords()
Method to compute filtered variables from filtered coordinates.

compute_filtered_variables_from_filtered_xy()
Method to compute filtered variables from filtered xy.

compute_image_weights_with_filtered_data()
Method to compute image weights from filtered data. Weight is simply inverse of class frequency.

edge_requirements(*w_lim*, *h_lim*, *x2_lim*, *y2_lim*)
Method to compute filtering based on edge specifications.

Inputs:

w_lim: limit for image width
h_lim: limit for image height
x2_lim: limit for image x2
y2_lim: limit for image y2

Outputs:

indices where filtered variables satisfy the dimension requirements.

load_target_file()
Method to load a targetfile of type specified in the input file. Supported types: .json.

manual_dimension_requirements(*area_lim*, *w_lim*, *h_lim*, *AR_lim*)
Method to compute filtering based on specified dimension requirements.

Inputs:

area_lim: limit for image area
w_lim: limit for image width
h_lim: limit for image height
AR_lim: limit for image aspect ratio

Outputs:

indices where filtered variables satisfy the dimension requirements.

process_target_data()
Method to perform all target processing.

set_image_w_and_h()
Method to set width and height of images associated with targets.

sigma_rejection_indices(*filtered_data*)
Method to compute a mask based on a sigma rejection criterion.

Inputs:

filtered_data: data to which sigma rejection is applied and from which mask is computed

Outputs:

mask_reject: binary mask computed from sigma rejection

strip_image_number_from_chips_and_files()
Method to strip numbers from image filenames from both chips and files.

4.2 Utils/

utils.utils.**bbox_iou**(*box1*, *box2*, *x1y1x2y2=True*)
Returns the IoU of two bounding boxes

utils.utils.**build_targets**(*pred_boxes*, *pred_conf*, *pred_cls*, *target*, *anchor_wh*, *nA*, *nC*, *nG*, *requestPrecision*)
returns nGT, nCorrect, tx, ty, tw, th, tconf, tcls

utils.utils.**compute_ap**(*recall*, *precision*)
Compute the average precision, given the recall and precision curves. Code originally from https://github.com/rbgirshick/py-faster-rcnn. # Arguments

recall: The recall curve (list). precision: The precision curve (list).

 # Returns The average precision as computed in py-faster-rcnn.

utils.utils.**load_classes**(*path*)
Loads class labels at 'path'

4.3 Tests/

class tests.unittests.**DataProcessingTests**(*methodName='runTest'*)

setUp()
Hook method for setting up the test fixture before exercising it.

class tests.unittests.**DatasetTests**(*methodName='runTest'*)

setUp()
Hook method for setting up the test fixture before exercising it.

class tests.unittests.**GPUtests**(*methodName='runTest'*)

setUp()
Hook method for setting up the test fixture before exercising it.

class tests.unittests.**ModelsTests**(*methodName='runTest'*)

setUp()
Hook method for setting up the test fixture before exercising it.

class tests.unittests.**TargetTests**(*methodName='runTest'*)

setUp()
Hook method for setting up the test fixture before exercising it.

CHAPTER

FIVE

CONTACT

Any questions/comments may be directed to the main BNL project developer, Anthony DeGennaro, at adegennaro@bnl.gov.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

s

t

u

INDEX

T

U

Y