wbia-pydarknet

Release latest

Nov 11, 2022

Contents:

CHAPTER 1

pydarknet package

1.1 Submodules

1.2 pydarknet.__main__ module

pydarknet.__main__.**main**()

1.3 pydarknet._pydarknet module

pydarknet._pydarknet.**CONFIG_URL_DICT = {'template': 'https://wildbookiarepository.azureedg**
Bindings for C Variable Types

pydarknet._pydarknet.**C_ARRAY_CHAR**
alias of pydarknet._pydarknet.LP_c_char_p

pydarknet._pydarknet.**C_ARRAY_FLOAT**
alias of pydarknet._pydarknet.LP_c_float

class pydarknet._pydarknet.**Darknet_YOLO_Detector**(*config_filepath=None*, *weights_filepath=None*, *classes_filepath=None*, *verbose=True*, *quiet=False*)

Bases: object

detect(*input_gpath_list*, ***kwargs*)
Run detection with a given loaded forest on a list of images

Parameters

- **input_gpath_list** (*list of str*) – the list of image paths that you want
- **test** (*to*) –
- **config_filepath** (*str, optional*) – the network definition for YOLO to use

- **weights_filepath** (*str, optional*) – the network weights for YOLO to use

Kwargs: sensitivity (float, optional): the sensitivity of the detector, which accepts a value between 0.0 and 1.0; defaults to 0.0 batch_size (int, optional): the number of images to test at a single time in paralell (if None, the number of CPUs is used); defaults to None verbose (bool, optional): verbose flag; defaults to object's verbose or selectively enabled for this function

Yields *(str, (list of dict))* – tuple of the input image path and a list of dictionaries specifying the detected bounding boxes

The dictionaries returned by this function are of the form: xtl (int): the top left x position of the bounding box ytl (int): the top left y position of the bounding box width (int): the width of the bounding box height (int): the hiehgt of the bounding box class (str): the most probably class detected by the network confidence (float): the confidence that this bounding box is of the class specified by the trees used during testing

CommandLine: python -m pydarknet._pydarknet detect –show

Example

```
>>> # DISABLE_DOCTEST
>>> from pydarknet._pydarknet import *  # NOQA
>>> dpath = '/media/raid/work/WS_ALL/localizer_backup/'
>>> weights_filepath = join(dpath, 'detect.yolo.2.39000.weights')
>>> config_filepath = join(dpath, 'detect.yolo.2.cfg')
>>> dark = Darknet_YOLO_Detector(config_filepath=config_filepath,
>>>                              weights_filepath=weights_filepath)
>>> input_gpath_list = [u'/media/raid/work/WS_ALL/_ibsdb/images/0cb41f1e-d746-
↪3052-ded4-555e11eb718b.jpg']
>>> kwargs = {}
>>> (input_gpath, result_list_) = dark.detect(input_gpath_list)
>>> result = ('(input_gpath, result_list_) = %s' % (ut.repr2((input_gpath,␣
↪result_list_)),))
>>> print(result)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> ut.show_if_requested()
```

dump(*file*)
UNIMPLEMENTED

dumps()
UNIMPLEMENTED

load(*file*)
UNIMPLEMENTED

loads(*string*)
UNIMPLEMENTED

train(*voc_path*, *weights_path*, ***kwargs*)
Train a new forest with the given positive chips and negative chips.

Parameters

- **train_pos_chip_path_list** (*list of str*) – list of positive training chips
- **train_neg_chip_path_list** (*list of str*) – list of negative training chips

- **trees_path** (*str*) – string path of where the newly trained trees are to be saved

Kwargs:

chips_norm_width (int, optional): Chip normalization width for resizing; the chip is resized to have a width of chips_norm_width and whatever resulting height in order to best match the original aspect ratio; defaults to 128

If both chips_norm_width and chips_norm_height are specified, the original aspect ratio of the chip is not respected

chips_norm_height (int, optional): Chip normalization height for resizing; the chip is resized to have a height of chips_norm_height and whatever resulting width in order to best match the original aspect ratio; defaults to None

If both chips_norm_width and chips_norm_height are specified, the original aspect ratio of the chip is not respected

verbose (bool, optional): verbose flag; defaults to object's verbose or selectively enabled for this function

Returns None

pydarknet._pydarknet.**RESULTS_ARRAY**
alias of numpy.ctypeslib.ndpointer_<u8_1d_ALIGNED_C_CONTIGUOUS_WRITEABLE

pydarknet._pydarknet.**test_pydarknet**()

CommandLine: python -m pydarknet._pydarknet –exec-test_pydarknet –show

Example

```
>>> # ENABLE_DOCTEST
>>> from pydarknet._pydarknet import *  # NOQA
>>> test_pydarknet()
>>> ut.quit_if_noshow()
>>> ut.show_if_requested()
```

pydarknet._pydarknet.**test_pydarknet2**(*input_gpath_list=None*, *config_filepath=None*, *weights_filepath=None*, *classes_filepath=None*)

CommandLine: python -m pydarknet._pydarknet test_pydarknet2 –show

python -m pydarknet test_pydarknet2 –show –input_gpath_list=["~/work/WS_ALL/_ibsdb/images/0cb41f1e-d746-3052-ded4-555e11eb718b.jpg"] –config_filepath="~/work/WS_ALL/localizer_backup/detect.yolo.2.cfg" –weights_filepath="~/work/WS_ALL/localizer_backup/detect.yolo.2.39000.weights" –classes_filepath="~/work/WS_ALL/localizer_backup/detect.yolo.2.cfg.classes"

Ignore:

```
>>> # Load in the second command line strings for faster testing
>>> from pydarknet._pydarknet import *  # NOQA
>>> cmdstr = ut.get_func_docblocks(test_pydarknet2)['CommandLine:'].split(
↪'\n\n')[1]
>>> ut.aug_sysargv(cmdstr)
```

Example

```
>>> # DISABLE_DOCTEST
>>> from pydarknet._pydarknet import *  # NOQA
>>> funckw = ut.argparse_funckw(test_pydarknet2)
>>> exec(ut.execstr_dict(funckw), globals())
>>> output_fpaths = test_pydarknet2(**funckw)
>>> ut.quit_if_noshow()
>>> import wbia.plottool as pt
>>> inter = pt.MultiImageInteraction(output_fpaths)
>>> inter.start()
>>> ut.show_if_requested()
```

1.4 pydarknet.ctypes_interface module

pydarknet.ctypes_interface.**find_lib_fpath**(*libname*, *root_dir*, *recurse_down=True*, *verbose=False*)

Search for the library

pydarknet.ctypes_interface.**get_lib_dpath_list**(*root_dir*)

returns possible lib locations

Parameters **root_dir** (*str*) – deepest directory to look for a library (dll, so, dylib)

Returns plausible directories to look for libraries

Return type list

pydarknet.ctypes_interface.**get_lib_fname_list**(*libname*)

Parameters **libname** (*str*) – library name (e.g. 'hesaff', not 'libhesaff')

Returns list of plausible library file names

Return type list

pydarknet.ctypes_interface.**load_clib**(*libname*, *root_dir*)

Does the work.

Parameters

- **libname** (*str*) – library name (e.g. 'hesaff', not 'libhesaff')
- **root_dir** (*str*) – the deepest directory searched for the library file (dll, dylib, or so).

Returns clib a ctypes object used to interface with the library

Return type ctypes.cdll

1.5 pydarknet.pydarknet_helpers module

pydarknet.pydarknet_helpers.**ensure_bytes_strings**(*str_list*)

1.6 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index