

---

# **YOLOv3 Documentation**

***Release 0.3***

**Anthony DeGennaro**

**Jan 09, 2019**



**CONTENTS:**

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Requirements</b>	<b>3</b>
<b>3</b>	<b>Installation</b>	<b>5</b>
3.1	Anaconda . . . . .	5
3.2	PyTorch . . . . .	6
3.3	GPU Support . . . . .	6
3.4	YOLOv3 . . . . .	7
<b>4</b>	<b>Code Docs</b>	<b>9</b>
4.1	Src/ . . . . .	9
4.2	Utils/ . . . . .	14
4.3	Tests/ . . . . .	14
<b>5</b>	<b>Contact</b>	<b>17</b>
<b>6</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



## **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 (<https://www.bnl.gov/compsci/people/staff.php?q=168> / [adegennaro@bnl.gov](mailto:adegennaro@bnl.gov)).

You may access the main project repository at <https://github.com/adegenna/xview-yolov3>.



## REQUIREMENTS

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

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





## 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 privileges. 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., nouveau), 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
```



## 4.1 Src/

`src.train2.main()`

Main driver script for training the YOLOv3 network.

**Inputs:**

*args*: command line arguments used in shell call for this main driver script. *args* must have a *inputfilename* member that specifies the desired inputfile name.

**Outputs:**

*inputs.outdir/results.txt*: output metrics for each training epoch

*inputs.loadaddr/latest.pt*: checkpoint file for latest network configuration

*inputs.loadaddr/best.pt*: checkpoint file for best current network configuration

*inputs.loadaddr/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.InputFile.InputFile(args=[])`

Class for packaging all input/config file options together.

**Inputs:**

*args*: (passed to constructor at runtime) command line arguments used in shell call for main driver script. *args* must have a *inputfilename* member that specifies the desired inputfile name.

**Options:**

*inputtype*: Options are *train* or *detect*  
*projdir*: Absolute path to project directory  
*datadir*: Absolute path to data directory  
*loaddir*: Absolute path to load directory  
*outdir*: Absolute path to output directory  
*targetspath*: Absolute path to target file  
*targetfiletype*: Type of target file  
*traindir*: Type of target file  
*targetfiletype*: Type of target file

**Options (Train-Specific):**

*traindir*: Type of target file  
*epochs*: Number of training epochs  
*epochstart*: Starting epoch  
*batchsize*: Training batch size  
*networkcfg*: Network architecture file  
*imgsize*: Base image crop size  
*resume*: Boolean value specifying whether training is resuming from previous iteration  
*invalid\_class\_list*: Comma-separated list of classes to be ignored from training data  
*boundingboxclusters*: Desired number of bounding-box clusters for the YOLO architecture  
*computeboundingboxclusters*: Boolean value specifying whether to compute bounding box clusters

**Options (Detect-Specific):**

*imagepath*: Image path  
*plotflag*: Flag for plotting  
*secondary\_classifier*: Boolean value specifying whether to use a secondary classifier  
*networkcfg*: Network architecture file  
*class\_path*: Absolute path to class  
*conf\_thres*: Confidence threshold for detection  
*nms\_thres*: NMS threshold  
*batch\_size*: Desired batchsize  
*img\_size*: Desired cropped image size

**printInputs ()**

Method to print all config options.

**readDetectInputfile (inputfilestream)**

Method to read config options from a detection inputfile

**Inputs:**

*inputfilestream*: specified inputfilestream.

**readTrainingInputfile (inputfilestream)**

Method to read config options from a training inputfile.

**Inputs:**

*inputfilestream*: specified inputfilestream.

**class** `src.models.Darknet` (*inputs*)  
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_architecture` (*inputs*, *targets*)  
Creates a yolo-v3 layer configuration file from desired options

`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.

**detect\_nonexistent\_chip(chip\_i)**

Method to detect all instances in database of a chip that does not exist

**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.



**remove\_nonexistent\_chips\_from\_database** (*idx\_nonexistent*)

Method to remove all nonexistent chips from database

**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.

`src.targets.fcn_sigma_rejection.fcn_sigma_rejection` (*x*, *srl*=3, *ni*=3)

Function to perform sigma rejection on a dataset.

**Inputs:**

*x*: dataset

*srl*: desired cutoff number of standard deviations for rejection

*ni*: desired number of iterations

**Outputs:**

*x*: dataset with outliers removed

*inliers*: indices of inliers w.r.t. original dataset

`src.targets.per_class_stats.per_class_stats` (*classes*, *w*, *h*)

Function to calculate statistics of target data.

**Inputs:**

*classes*: target data processed/produced with the Target class

*w*: image width

*h*: image height

**Outputs:**

*class\_mu*: mean of target classes

*class\_sigma*: standard deviation of target classes

*class\_cov*: covariance of target classes

## 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.convert_tif2bmp (p)`

Function to convert .tif -> .bmp

**Inputs:**

*p*: Absolute path to the dataset directory

`utils.utils.load_classes (path)`

Loads class labels at 'path'

`utils.utils.readBmpDataset (path)`

Function to read a .bmp dataset. If the provided directory does not contain .bmp files, a conversion is attempted.

**Inputs:**

*path*: Absolute path to the dataset directory

## 4.3 Tests/

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

Class for all data processing unit tests.

`setUp ()`

Basic setup method. Note that ResourceWarnings and DeprecationWarnings are ignored.

`test_get_dataset_filenames ()`

Test loading of dataset filenames.

`test_get_dataset_height_width_channels ()`

Test loading sizes of dataset images.

`test_get_labels_geojson ()`

Test loading of geojson formatted data.

`test_strip_image_number_from_filename ()`

Test functionality to strip image number from image filename.

```

class tests.unittests.DatasetTests (methodName='runTest')
    Class for all dataset-involved unit tests.

    setUp ()
        Basic setup method. Note that ResourceWarnings and DeprecationWarnings are ignored.

    test_load_targets ()
        Test functionality to load training data.

    test_show_targets ()
        Test functionality to label training data.

class tests.unittests.GPUtests (methodName='runTest')
    Class for all GPU/cuda unit tests.

    setUp ()
        Basic setup method. Note that ResourceWarnings and DeprecationWarnings are ignored.

    test_cuda_available ()
        Test whether cuda is available.

    test_cuda_version ()
        Test that cuda version is >= 9.

    test_gpu_avail ()
        Test that GPU hardware is available.

class tests.unittests.ModelsTests (methodName='runTest')
    Class for all models-involved unit tests.

    setUp ()
        Basic setup method. Note that ResourceWarnings and DeprecationWarnings are ignored.

    test_create_yolo_config_file ()
        Test functionality to create custom YOLOv3 config file from a template.

class tests.unittests.TargetTests (methodName='runTest')
    Class for all target-involved unit tests.

    setUp ()
        Basic setup method. Note that ResourceWarnings and DeprecationWarnings are ignored.

    test_apply_mask_to_filtered_data ()
        Test mask application to filtered data method.

    test_area_requirements ()
        Test area requirements method.

    test_compute_bounding_box_clusters_using_kmeans ()
        Test bounding box cluster computation method.

    test_compute_cropped_data ()
        Test functionality for cropping targets.

    test_compute_image_weights_with_filtered_data ()
        Test class weight computation method.

    test_compute_width_height_area ()
        Test functionality for computing target coordinate area.

    test_edge_requirements ()
        Test functionality for computing edge requirements on target data.

```

**test\_fcn\_sigma\_rejection()**  
Test functionality for computing sigma rejection.

**test\_invalid\_class\_requirement()**  
Test invalid class requirement method.

**test\_load\_target\_file()**  
Test functionality for loading target data (.json file).

**test\_manual\_dimension\_requirements()**  
Test functionality for imposing manual dimensions requirements on target data.

**test\_nan\_inf\_size\_requirements()**  
Test nan/inf/size requirements method.

**test\_per\_class\_stats()**  
Test per\_class\_stats function.

**test\_sigma\_rejection\_indices()**  
Test functionality for computing sigma rejection indices.

**test\_xy\_coords()**  
Test functionality for target coordinate parsing.

## CONTACT

Any questions/comments may be directed to the main BNL project developer, Anthony DeGennaro (<https://www.bnl.gov/compsci/people/staff.php?q=168> / [adegennaro@bnl.gov](mailto:adegennaro@bnl.gov)).



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### S

- `src.detect`, 9
- `src.InputFile`, 9
- `src.models`, 11
- `src.NetworkTrainer`, 11
- `src.targets.fcn_sigma_rejection`, 13
- `src.targets.per_class_stats`, 13
- `src.targets.Target`, 11
- `src.train2`, 9

### t

- `tests.unittests`, 14

### u

- `utils.datasetProcessing`, 14
- `utils.datasets`, 14
- `utils.utils`, 14
- `utils.utils_xview`, 14



## A

`apply_mask_to_filtered_data()`  
(*src.targets.Target.Target method*), 11

## B

`bbox_iou()` (*in module utils.utils*), 14  
`build_targets()` (*in module utils.utils*), 14

## C

`compute_ap()` (*in module utils.utils*), 14  
`compute_bounding_box_clusters_using_kmeans()`  
(*src.targets.Target.Target method*), 11  
`compute_cropped_data()`  
(*src.targets.Target.Target method*), 12  
`compute_filtered_data_mask()`  
(*src.targets.Target.Target method*), 12  
`compute_filtered_variables_from_filtered_coords()`  
(*src.targets.Target.Target method*), 12  
`compute_filtered_variables_from_filtered_data()`  
(*src.targets.Target.Target method*), 12  
`compute_image_weights_with_filtered_data()`  
(*src.targets.Target.Target method*), 12  
`convert_tif2bmp()` (*in module utils.utils*), 14  
`ConvNetb` (*class in src.detect*), 9  
`create_modules()` (*in module src.models*), 11  
`create_yolo_architecture()` (*in module src.models*), 11  
`create_yolo_config_file()` (*in module src.models*), 11

## D

`Darknet` (*class in src.models*), 11  
`DataProcessingTests` (*class in tests.unittests*), 14  
`DatasetTests` (*class in tests.unittests*), 14  
`detect_nonexistent_chip()`  
(*src.targets.Target.Target method*), 12

## E

`edge_requirements()` (*src.targets.Target.Target method*), 12  
`EmptyLayer` (*class in src.models*), 11

## F

`fcn_sigma_rejection()` (*in module src.targets.fcn\_sigma\_rejection*), 13  
`forward()` (*src.detect.ConvNetb method*), 9  
`forward()` (*src.models.Darknet method*), 11  
`forward()` (*src.models.YOLOLayer method*), 11

## G

`GPUtests` (*class in tests.unittests*), 15

## I

`InputFile` (*class in src.InputFile*), 9

## L

`load_classes()` (*in module utils.utils*), 14  
`load_target_file()` (*src.targets.Target.Target method*), 12

## M

`main()` (*in module src.train2*), 9  
`manual_dimension_requirements()`  
(*src.targets.Target.Target method*), 12  
`ModelsTests` (*class in tests.unittests*), 15

## P

`parse_model_config()` (*in module src.models*), 11  
`per_class_stats()` (*in module src.targets.per\_class\_stats*), 13  
`printInputs()` (*src.InputFile.InputFile method*), 10  
`process_target_data()` (*src.targets.Target.Target method*), 12

## R

`readBmpDataset()` (*in module utils.utils*), 14  
`readDetectInputfile()` (*src.InputFile.InputFile method*), 10  
`readTrainingInputfile()`  
(*src.InputFile.InputFile method*), 10  
`remove_nonexistent_chips_from_database()`  
(*src.targets.Target.Target method*), 12

## S

`set_image_w_and_h()` (*src.targets.Target.Target method*), 13

`setUp()` (*tests.unittests.DataProcessingTests method*), 14

`setUp()` (*tests.unittests.DatasetTests method*), 15

`setUp()` (*tests.unittests.GPUtests method*), 15

`setUp()` (*tests.unittests.ModelsTests method*), 15

`setUp()` (*tests.unittests.TargetTests method*), 15

`sigma_rejection_indices()` (*src.targets.Target.Target method*), 13

`src.detect (module)`, 9

`src.InputFile (module)`, 9

`src.models (module)`, 11

`src.NetworkTrainer (module)`, 11

`src.targets.fcn_sigma_rejection (module)`, 13

`src.targets.per_class_stats (module)`, 13

`src.targets.Target (module)`, 11

`src.train2 (module)`, 9

`strip_image_number_from_chips_and_files()` (*src.targets.Target.Target method*), 13

## T

`Target (class in src.targets.Target)`, 11

`TargetTests (class in tests.unittests)`, 15

`test_apply_mask_to_filtered_data()` (*tests.unittests.TargetTests method*), 15

`test_area_requirements()` (*tests.unittests.TargetTests method*), 15

`test_compute_bounding_box_clusters_using_kmeans()` (*tests.unittests.TargetTests method*), 15

`test_compute_cropped_data()` (*tests.unittests.TargetTests method*), 15

`test_compute_image_weights_with_filtered_data()` (*tests.unittests.TargetTests method*), 15

`test_compute_width_height_area()` (*tests.unittests.TargetTests method*), 15

`test_create_yolo_config_file()` (*tests.unittests.ModelsTests method*), 15

`test_cuda_available()` (*tests.unittests.GPUtests method*), 15

`test_cuda_version()` (*tests.unittests.GPUtests method*), 15

`test_edge_requirements()` (*tests.unittests.TargetTests method*), 15

`test_fcn_sigma_rejection()` (*tests.unittests.TargetTests method*), 15

`test_get_dataset_filenames()` (*tests.unittests.DataProcessingTests method*), 14

`test_get_dataset_height_width_channels()` (*tests.unittests.DataProcessingTests method*), 14

`test_get_labels_geojson()` (*tests.unittests.DataProcessingTests method*), 14

`test_gpu_avail()` (*tests.unittests.GPUtests method*), 15

`test_invalid_class_requirement()` (*tests.unittests.TargetTests method*), 16

`test_load_target_file()` (*tests.unittests.TargetTests method*), 16

`test_load_targets()` (*tests.unittests.DatasetTests method*), 15

`test_manual_dimension_requirements()` (*tests.unittests.TargetTests method*), 16

`test_nan_inf_size_requirements()` (*tests.unittests.TargetTests method*), 16

`test_per_class_stats()` (*tests.unittests.TargetTests method*), 16

`test_show_targets()` (*tests.unittests.DatasetTests method*), 15

`test_sigma_rejection_indices()` (*tests.unittests.TargetTests method*), 16

`test_strip_image_number_from_filename()` (*tests.unittests.DataProcessingTests method*), 14

`test_xy_coords()` (*tests.unittests.TargetTests method*), 16

`tests.unittests (module)`, 14

## U

`utils.datasetProcessing (module)`, 14

`utils.datasets (module)`, 14

`utils.utils (module)`, 14

`utils.utils_xview (module)`, 14

## Y

`YOLOLayer (class in src.models)`, 11