# Few-shot detection benchmark and the RepMet detection engine

## Introduction

This manual describes the python code package implementing two modules:

**Few-shot detection benchmark**

* Produce episodic data
* Manage a detection engine

**RepMet detection engine**

* Train a model for episode data (novel categories) based on pretrained model
* Perform detection and perfromance evaluation

The code package is available in the IBM GitHub repository

<https://github.ibm.com/CVAR-DL-DET/RepMet/tree/FSD_bench_br> (please notice the branch FSD\_bench\_br).

The RepMet algorithm is described in the [paper [1]](#_References:)

## Deployment of code and data

The codebase was developed in a conda environment with Python 2.7, MXNet 1.0.0, and CUDA 8.0.

To build the environment, follow the steps:

1. conda create -n env-python2.7-mxnet1.0.0 python=2.7
2. . activate env-python2.7-mxnet1.0.0
3. Put the file requirements.txt under Anaconda\*/.../envs.
4. pip install -r requirements.txt
5. Installing additional packages:
   1. conda install matplotlib
   2. pip install sklearn
   3. pip install PyYAML
   4. pip install opencv-python
6. conda activate env-python2.7-mxnet1.0.0

requirements.txt can be found in the root of the repository (RepMet/), or created as a text file with following lines:

Cython

EasyDict

opencv-python

mxnet-cu80==1.0.0

The dataset, models and the rest of the data are originally located on /dccstor storage, used at the CCC computation cluster. All the relevant files are transferred to the folder ‘data’ in the IBM BOX folder <https://ibm.box.com/s/z5ikclgkvnli84fxl1ag3b57fdpopc0s> . For deployment, this folder should be placed under the RepMet root of the git package.

The dataset information (images, Ground Truth boxes and classes) is given in a roidb\_\*.pkl file, produced during the training of the base model. The structure of roidb is described below. In order to use the package with other datasets, a code for creating one may be based on RepMet/lib/dataset/imagenet.py  
Before the benchmark can be executed, all the image paths in the roidb need to be replaced; all the rest of the paths are in the format ./data/…, they should work once the Box content is copied under the repository root ‘RepMet/’.

## Example execution of a benchmark experiment

From the main folder of the repository, /RepMet, execute

python fpn/run\_FSD\_benchmark\_v1.py --bcfg\_fname=./experiments/bench\_configs/Inloc\_RepMet\_3\_6\_2\_5.yaml

The run script run\_FSD\_benchmark\_v1.py creates an instance of FSD\_bench class with given arguments,

benchmark = FSD\_bench(args)

and runs its following routines:

benchmark.setup()  
benchmark.gen\_episodes() *# prep training data, setup detector*benchmark.run\_episodes()

The setup() routine initiates the experiment.

The gen\_episodes() routine loads the set of episodes for the benchmark, or generates the episodes with the bcfg parameters if the episodes file does not exist.

The run\_episodes() routine creates a FSD\_engine class instance to train a model for each episode and runs a test on the query images for this episode.

### Expected output for this experiment:

2019-03-29 13:53:01,336 - my\_logger - INFO - ========= starting FSD benchmark Inloc\_RepMet\_3\_6\_2\_5

2019-03-29 13:53:11,092 - my\_logger - INFO - Creating new episodes data.

2019-03-29 13:53:23,588 - my\_logger - INFO - Starting episode 0 -------

2019-03-29 13:54:08,492 - my\_logger - INFO - \_ #Dets: 16570, #GT: 15 TP: 15 FP: 16555 = 75 wrong + 16480 bkgnd Recall: 1.000 AP: 0.852

2019-03-29 13:54:08,493 - my\_logger - INFO - Starting episode 1 -------

Box #0 - no overlapping detection boxes found

2019-03-29 13:54:18,727 - my\_logger - INFO - No rois found for training box of eskimo dog, husky

2019-03-29 13:54:46,936 - my\_logger - INFO - \_ #Dets: 30426, #GT: 30 TP: 30 FP: 30396 = 150 wrong + 30246 bkgnd Recall: 1.000 AP: 0.654

2019-03-29 13:54:46,937 - my\_logger - INFO - Starting episode 2 -------

2019-03-29 13:55:24,407 - my\_logger - INFO - \_ #Dets: 45216, #GT: 45 TP: 43 FP: 45173 = 217 wrong + 44956 bkgnd Recall: 0.956 AP: 0.606

2019-03-29 13:55:24,407 - my\_logger - INFO - Starting episode 3 -------

Box #0 - no overlapping detection boxes found

2019-03-29 13:55:32,141 - my\_logger - INFO - No rois found for training box of king snake, kingsnake

2019-03-29 13:56:03,580 - my\_logger - INFO - \_ #Dets: 61764, #GT: 58 TP: 56 FP: 61708 = 282 wrong + 61426 bkgnd Recall: 0.966 AP: 0.631

2019-03-29 13:56:03,580 - my\_logger - INFO - Starting episode 4 -------

2019-03-29 13:56:41,246 - my\_logger - INFO - \_ #Dets: 76367, #GT: 73 TP: 71 FP: 76296 = 350 wrong + 75946 bkgnd Recall: 0.973 AP: 0.565

Files in the data folder (in Box repository):

All the available files are related to the Imagenet-LOC dataset, hence are placed under the /data/Imagenet\_LOC/ folder

|  |  |
| --- | --- |
| **File** | **Information** |
| inloc\_first101\_categories.txt | List of all the classes in Imagenet-LOC dataset |
| in\_domain\_categories\_ord.txt | List of the classes selected for few-shot experiments, along with their original indices in the total list |
| voc\_inloc\_gt\_roidb.pkl | The roidb database containing ground truth information. |
| RepMet\_pascal\_imagenet-0015.params | Model weights for RepMet detection model |
| Episodes/epi\_inloc\_in\_domain\_1\_5\_10\_500.pkl (and similar ones) | An episodic data file for benchmark |

Replacement of image pathes:

The list entries from voc\_inloc\_gt\_roidb.pkl have a field ‘image’ (see section ‘Structure of roidb for rull information). The replacement needs to be performed as follows:

|  |  |
| --- | --- |
| Current value | New value: |
| roidb[i][‘image’] = <old path>/<image\_name.jpg> | roidb[i][‘image’] = <new path>/<image\_name.jpg> |

Where the new value depends on the Imagenet-LOC dataset deployment.

The experiments, described in RepMet [CVPR paper [1]](#_References:) may be repeated using the corresponding bcfg config files and stored episodic data.

Execute python fpn/run\_FSD\_benchmark\_v1.py -- bcfg\_fname=<bench config fname>  
with the following values of <bench config fname>:

|  |  |
| --- | --- |
| Experiment | bcfg\_fname |
| RepMet, 3-shot, 6-way | ./experiments/bench\_configs/Inloc\_RepMet\_3\_6\_2\_5.yaml |
| RepMet, 1-shot, 5-way | ./experiments/bench\_configs/Inloc\_RepMet\_1\_5\_10\_500.yaml |
| RepMet, 5-shot, 5-way | ./experiments/bench\_configs/Inloc\_RepMet\_5\_5\_10\_500.yaml |

## Data structures

### Structure of roidb

The roidb object, loaded from file data/Imagenet\_LOC/voc\_inloc\_gt\_roidb.pkl is a list of entries corresponding to set of images. Each entry is a dictionary with following fields:

**entry[‘gt\_classes’]** – list of class indices present in the image, a subset of [1,2,…1000] with possible repititions (the 1000 classes of imagenet). List of corresponding class names is given in data/Imagenet\_LOC/inloc\_classes\_list.txt

**entry[‘image’]** – full path to the image.

In the provided roidb files, the image pathes need to be replaced with those available at the location of benchmark deployment.

**entry[‘boxes’]** – a numpy array of bounding boxes, where the rows contain four box coordinates (left, top, right, bottom) and the rows are ordered correspondingly to the list of classes entry[‘gt\_classes’]

### Structure of episodic data

According to the concept of meta-learning, the training and evaluation of a few-shot detection engine is performed using subsets of the given large datasets, known as tasks or episodes.

Each episode is an instance of a few-shot task that is comprised from a support set and query set, and contains data from Nway visual categories, each represented by Nshot examples (ROIs in the support set images). The query (evaluation) data consists of Nquery images per category, each containing one or more examples from this category (and possibly instances of other support set categories). The test data for the benchmark consists of Nepisodes such episodes, randomly drawn from the list of visual categories and images not seen during the offline training of the base model.

The episodic data (in particular, that of RepMet paper benchmark) for varying experiments is stored in the folder data\Imagenet\_LOC. Each episode data file contains the list of episode objects, where each episode is a dictionary with the following fields:

episode['epi\_cats'] – set of Nway class indices randomly picked from the dataset list

episode['epi\_cats\_names'] – corresponding set of class names (strings)

episode[' train\_boxes'] – list of train objects. Each object is represented by the list   
 train\_box = [class\_index, class\_name, image\_path, bounding\_box]

episode[' query\_images'] – list of paths to query images of the episode

episode[' query\_gt'] – list of roidb entries corresp. to the query images.

For other datasets or new benchmarks, the set of episodes can also be generated using the routine FSD\_bench.gen\_episodes().

### Structure of input parameters

The benchmark class FSD\_bench has one required and two optional parameters.

#### bcfg\_fname

(necessary parameter) is a path to the benchmark configuration (bcfg) file. A fast working example is RepMet/experiments/bench\_configs/Inloc\_RepMet\_3\_6\_2\_5.yaml. The name format is <dataset>\_<detection engine>\_<Nshot>\_<Nway>\_<Nquery>\_<Nepisodes>.yaml (see the meaning of the variable names in “structure of episodic data” section of this manual).

The bcfg file for a specific experiment contains only a few fields overriding the values in the full bench\_config.py set of parameters. So all the default values of the benchmark configuration are given in RepMet/fpn/config/bench\_config.py

The detection engine RepMet has its own main configuration file RepMet/fpn/config/config.py and a model-specific update for this configuration, which path is provided as a parameter in the bcfg. The detection engine configuration is loaded within the FSD\_engine class.

#### bcfg\_args

(optional) is a command line argument string containing numeric confuguration parameters (from either benchmark or detection engine configuration files) that shall override any values in the default configuration files (e.g., RepMet/fpn/config/bench\_config.py or RepMet/fpn/config/config.py) and in the experiment-specific yml files (e.g., experiments/bench\_configs/Inloc\_RepMet\_3\_6\_2\_5.yaml or ./experiments/fpn/cfgs/oneshot/resnet\_v1\_101\_voc0712\_trainval\_fpn\_dcn\_oneshot\_end2end\_ohem\_8\_dev.yaml).   
An example for bcfg\_args value:  
--bcgf\_args=”TRAIN.lr=0.002, detector.nms\_train=0.4”

The first parameter belongs to detection engine (dcfg.TRAIN.lr = learning rate), and the second happens to be from benchmark configuration (NMS value for train object rois selection).

#### gpu

(optional) – the index of allocated GPU number (default=0). Relevant for multi-GPU servers.

## References:

[1] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, Alex M. Bronstein, RepMet: Representative-based metric learning for classification and one-shot object detection. Accepted to CVPR 2019. <https://arxiv.org/abs/1806.04728>