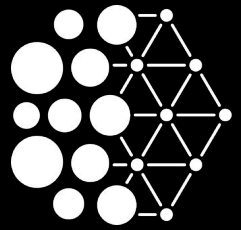
Institut
québécois
d'intelligence
artificielle



Mila

Implementation details (Block 3)
Door number detection project

Jeremy Pinto

Code

We will be using facebook research's MaskRCNN benchmark repo:

https://github.com/facebookresearch/maskrcnn-benchmark

Pros

- Code is already well implemented and optimized
- Few modifications needed for our task
- Quick iteration

Cons

- The code is coco-dataset centric
- Not as much documentation around training from scratch
- Still actively being developed (daily commits)
- HELIOS

Data preparation model 1 (done by TAs)

- We've preprocess the data so that it's in the format of the COCO dataset.
- In the case of this synthetic dataset, there is only one class, i.e. "House Number".
- Each annotation contains a bbox, as well as a segmentation (which is a polygon made from the bbox; you do not have to use it).
- More info on COCO format: http://cocodataset.org/#format-data

```
annotation{
   "id"                : int,
   "image_id"          : int,
   "category_id"       : int,
   "segmentation"      : RLE or [polygon],
   "area"              : float,
   "bbox"              : [x,y,width,height],
   "iscrowd"           : 0 or 1,
}

categories[{
   "id"                : int,
   "name"              : str,
   "supercategory"     : str,
}]
```

Setup Required

There are a few steps, considerations and limitations to keep in mind for accomplishing this task on Helios.

1. You need setup helios to use CUDA >= 9.
2. You need to compile the code on a GPU and set it up so you can edit it directly. You will need to install the library locally.
3. GPU nodes have no internet - we will provide you with the downloaded pre-trained models

Setup Required

1. You need setup helios to use CUDA >= 9.

The following will ensure you are using the right CUDA version (default is 8 on Helios):

```
echo source /rap/jvb-000-aa/COURS2019/etudiants/common_digit_detection.env >> ~/.bashrc
```

This will take care of using the right CUDA version.

Setup Required

2. You need to compile the code on a GPU and set it up so you can edit it directly. You will need to this yourselves. If we do it, you will not be able to modify the source code.

To do so, create a new conda environment **INSIDE YOUR SINGULARITY CONTAINER.**

```
$ s_shell
$ conda create --name digit-detection
```

This will create a new container on your $HOME that you will be able to write to.

Setup Required

2. Part a) Install pycocotools:

```
$ cd # Go to your home
$ git clone https://github.com/cocodataset/cocoapi # Clone pycocotools repo
$ s_shell # Enter the container
$ source activate digit-detection # Activate your newly created environment
$ cd cocoapi/PythonAPI
$ python setup.py build_ext --inplace install --user

# Test your setup
$ cd && ipython
$ >>> import pycocotools
```

Setup Required

2. Part b) clone maskrcnn-benchmark and checkout the appropriate commit. The latest version of maskrcnn-benchmark requires pytorch-nightly - we will not use the latest version but a commit from ~2 weeks ago (before this constraint)

```
$ cd # Go to your home
$ git clone https://github.com/cocodataset/cocoapi # Clone maskrcnn-benchmark repo
$ cd maskrcnn-benchmark
$ git checkout 90080e6 # Commit on master before the requirement became pytorch-nightly
```

Setup Required

2. Part c) Compile maskrcnn on GPU

```
$ mdebug # Get a gpu in interactive mode, you may want to reserve more than 15 minutes
$ s_shell # Enter container

$ export
LD_LIBRARY_PATH=/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx/Compiler/intel2016.4/cuda/9.0.176/lib64/:$LD_LIBRARY_PATH

# This needs to be done every time when using s_shell, not s_exec. This is a “feature” of singularity.

$ source activate digit-detection #  activate your local conda environment
$ python setup.py build develop --user # Compile maskrcnn on GPU

# You should see some warnings, they are safe to ignore. This should be your final output:

Installed /home/jerpint/maskrcnn-benchmark
Processing dependencies for maskrcnn-benchmark==0.1
Finished processing dependencies for maskrcnn-benchmark==0.1
(humanware) Singularity ift6759.simg:~/maskrcnn-benchmark>
```

Setup Required

2. Additional Notes

- Everyone will have to do this individually on their session to be able to run the code. You should start tracking this folder in your git. You should be able to modify it and see the changes reflected in the code immediately.

Setup Required

3. GPU nodes have no internet - we will provide you with the downloaded pre-trained models. You should symlink these in case we update them.

```
$ ln -sfn /rap/jvb-000-aa/COURS2019/etudiants/data/digit-detection/torch/ ~/.torch # Symlink the downloaded pretrained models, only needs to be done once
```

Data

The data is located at

```
/rap/jvb-000-aa/COURS2019/etudiants/data/digit-detection/

├── train
│   ├── image_0.jpg
│   ├── image_1.jpg
│   ├── ...
│   └── instances_train.json
└── valid
    ├── image_5000.jpg
    ├── image_5001.jpg
    ├── ...
    └── instances_valid.json
```

Next Steps

- Dive into the maskrcnn-benchmark code
- Read their instructions carefully, tune the models and parameters to the ways you see fit (We are NOT training on the coco dataset!)
- Coming soon : labels for the new dataset in the same format as SVHN for fine-tuning - stay tuned!
- Pray to Helios to spare us from stack smashing errors

https://en.wikipedia.org/wiki/Helios