DEEPLESION: AUTOMATED MINING OF LARGE-SCALE LESION ANNOTATIONS AND UNIVERSAL LESION DETECTION WITH DEEP LEARNING 실습

http://yanke23.com/articles/research/2018/06/13/DeepLesion-dataset-CVPR-2018.html

소스 : https://github.com/rsummers11/CADLab/tree/master/lesion_detector_3DCE

GPU 서버 세팅

- 테스트 서버 사양
  - *OS : centos 7.4*
  - *GPU : Geforce 980Ti*
  - *Mem : 32GB*
- 세팅 순서
  1) *Docker 설치*
  2) *GPU에 맞는 NVIDIA Driver 설치*
  3) *nvidia-docker 설치*
  4) *NVIDIA GPU cloud에서 mxnet py2용 Docker 이미지 다운로드*
  5) *Mxnet 이미지에 jupyter 추가*

Docker 설치

- https://docs.docker.com/install/linux/docker-ce/centos/
- Uninstall old versions

```
$ sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-selinux \
                  docker-engine-selinux \
                  docker-engine
```

- Install using the repository

```
$ sudo yum install -y yum-utils \
  device-mapper-persistent-data \
  lvm2
```

```
$ sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

```
$ sudo yum install docker-ce
```

- Test

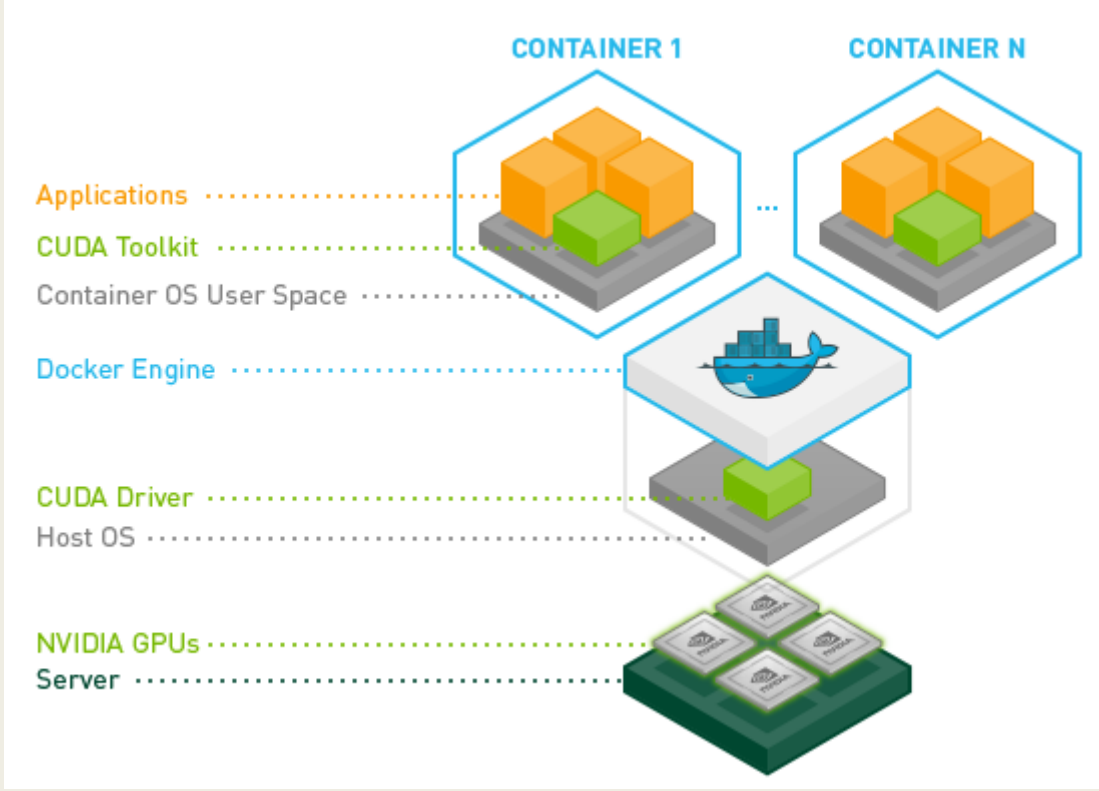
```
$ sudo systemctl start docker
```

$ sudo systemctl enable docker

```
$ sudo docker run hello-world
```

GPU에 맞는 NVIDIA Driver 설치

- https://devblogs.nvidia.com/nvidia-docker-gpu-server-application-deployment-made-easy/
- https://www.nvidia.com/Download/index.aspx?lang=en-us 에서 GPU 종류와 OS에 종류에 따른 드라이버 다운로드



GPU에 맞는 NVIDIA Driver 설치

```
# Disable Nouveau Driver
$ cat <<EOT >> /etc/modprobe.d/blacklist.conf
blacklist nouveau
EOT

$ mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak
$ dracut -v /boot/initramfs-$(uname -r).img $(uname -r)
$ yum update -y
$ reboot

# Set Up the Operating System and Kernel
$ yum install -y flex gcc gcc-c++  redhat-rpm-config strace \
  rpm-build make pkgconfig gettext automake \
  gdb bison libtool autoconf gcc-c++ gcc-gfortran \
  binutils rcs patchutils wget

$ yum install -y kernel-devel-`uname -r`
```

```
[root@cdsw tools]# /usr/bin/nvidia-smi
Sun Nov 25 17:54:16 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.77                 Driver Version: 390.77                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 980 Ti  Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   35C    P0    51W / 300W |      0MiB /  6083MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
# Install the NVIDIA Driver on GPU Nodes
$ export NVIDIA_DRIVER_VERSION=390.67
$ wget http://us.download.nvidia.com/XFree86/Linux-x86_64/390.67/NVIDIA-Linux-x86_64-${NVIDIA_DRIVER_VERSION}.run
$ chmod 755 ./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run
$./NVIDIA-Linux-x86_64-$NVIDIA_DRIVER_VERSION.run -asq

$ /usr/bin/nvidia-smi
```

nvidia-docker 설치

- https://github.com/NVIDIA/nvidia-docker
- CUDA 와 CUDNN 을 버전에 맞추어서 설치가 필요없음.
- CUDA와 CUDNN을 버전업할려면 OS부터 다시 설치필요.

# Enable Docker NVIDIA Volumes on GPU Nodes
$ wget https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.1/nvidia-docker-1.0.1-1.x86_64.rpm
$ yum install -y nvidia-docker-1.0.1-1.x86_64.rpm

$ systemctl start nvidia-docker
$ systemctl enable nvidia-docker

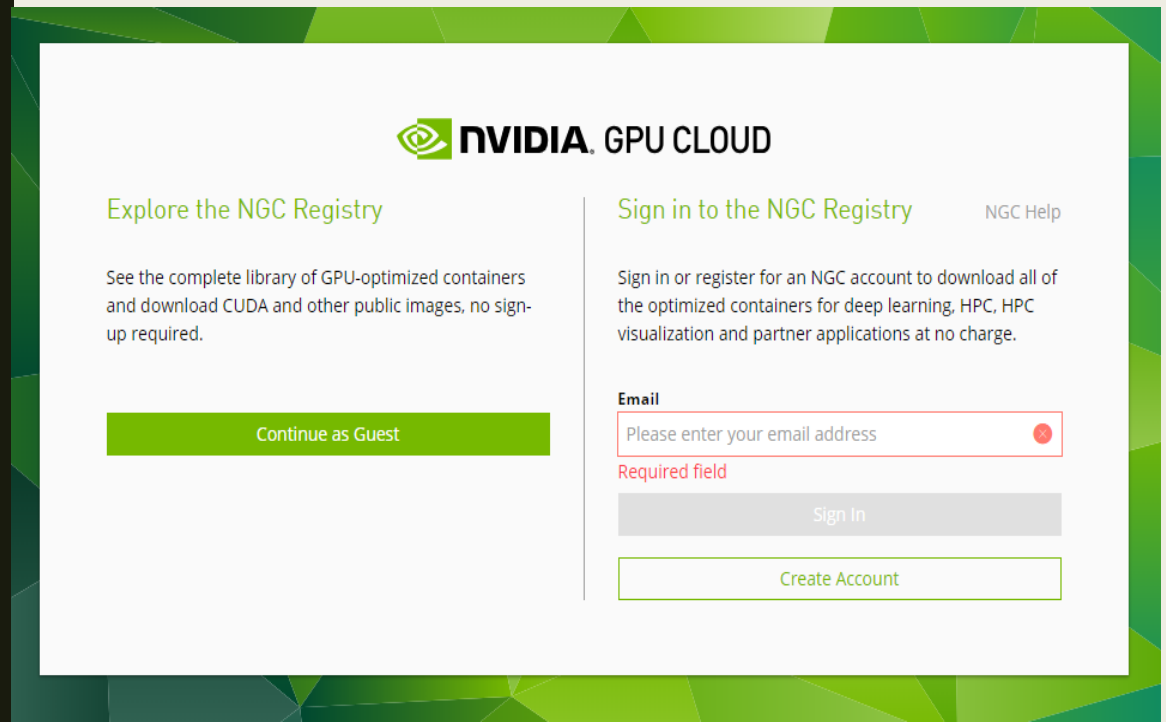
$ nvidia-docker run --rm nvidia/cuda:9.1-cudnn7-runtime nvidia-smi

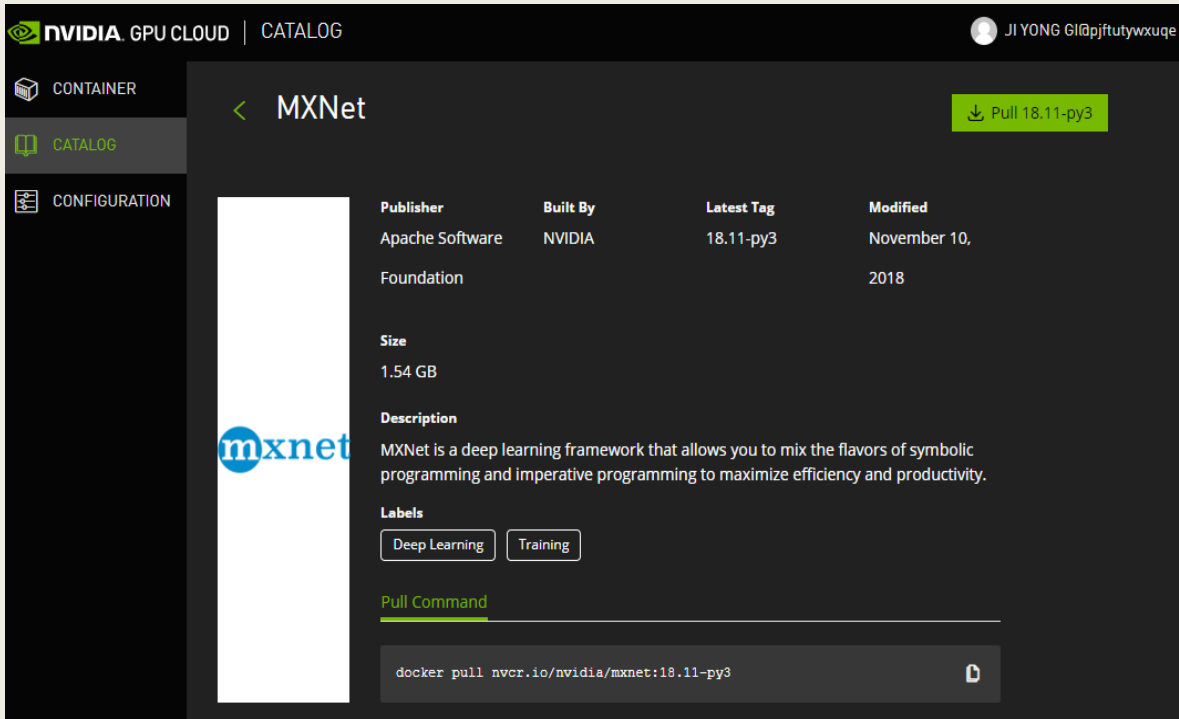
```
[root@cdsw tools]# nvidia-docker run --rm nvidia/cuda:9.1-cudnn7-runtime  nvidia-smi
Unable to find image 'nvidia/cuda:9.1-cudnn7-runtime' locally
9.1-cudnn7-runtime: Pulling from nvidia/cuda
18d680d61657: Pull complete
0addb6fece63: Pull complete
78e58219b215: Pull complete
eb6959a66df2: Pull complete
ec55825c3d11: Pull complete
15afc0d212f9: Pull complete
5ead56729c9f: Pull complete
cfc9b2783b7c: Pull complete
86d09c468e38: Pull complete
Digest: sha256:e062a7376503692325393fb5c0ee73a74f1eabc3298fbbc29c61bb14b442cc66
Status: Downloaded newer image for nvidia/cuda:9.1-cudnn7-runtime
Sun Nov 25 09:03:24 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.77                 Driver Version: 390.77                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 980 Ti  Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   35C    P0    51W / 300W |      0MiB /  6083MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

NVIDIA GPU cloud에서 mxnet py2용 Docker 이미지 다운로드

- https://ngc.nvidia.com/
- docker login nvcr.io
- Username: $oauthtoken
- Password: dGl0ajBh
- docker pull nvcr.io/nvidia/mxnet:18.11-py3





```
[root@cdsw ~]# docker login nvcr.io
Username ($oauthtoken): $oauthtoken
Password:
Login Succeeded
[root@cdsw ~]# docker pull nvcr.io/nvidia/mxnet:18.04-py2
18.04-py2: Pulling from nvidia/mxnet
f2233041f557: Pull complete
f321bcc6a76c: Pull complete
2f25d8d1d058: Pull complete
87bfe0d2f0e8: Pull complete
145c1bf7947a: Pull complete
b146afd09c39: Pull complete
daadc0376ab4: Downloading  43.17MB/441.7MB
b35ad3405786: Download complete
a33366ff728e: Download complete
9384272f4e0f: Download complete
7537bf7c96ee: Downloading  41.55MB/225MB
91e40537a771: Download complete
fc54fc28c6ac: Download complete
8916482f2511: Download complete
a839dd4b292e: Download complete
cbe311e8f219: Downloading  1.078MB/207.6MB
d6c58d429a77: Waiting
1e8cbca05768: Waiting
8ed7f16fc955: Waiting
06694f8026e7: Waiting
287952c822bb: Waiting
1a3b7522879b: Waiting
0df0cfe43860: Pulling fs layer
```

Mxnet 이미지에 jupyter 추가

```
# mxnet.18.07-py2.Dockerfile 내용
FROM nvcr.io/nvidia/mxnet:18.07-py2
RUN pip --no-cache-dir install Cython  ipykernel  jupyter  path.py Pillow pygments  six  sphinx  wheel  zmq  && \
    python -m ipykernel.kernelspec
# Set up notebook config
COPY jupyter_notebook_config.py /root/.jupyter/
# Jupyter has issues with being run directly: https://github.com/ipython/ipython/issues/7062
COPY run_jupyter.sh /root/
# Expose Ports for TensorBoard (6006), Ipython (8888)
EXPOSE 6006 8888
RUN mkdir work
WORKDIR "/work"
CMD ["/root/run_jupyter.run --allow-root"]
```

Mxnet 이미지에 jupyter 추가

- 이미지 빌드

$ docker build -t mycompany.com/nvidia/mxnet:18.07-py2 -f mxnet.18.07-py2.Dockerfile .

- 실행 방법

$ nvidia-docker run -it --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 -e PASSWORD=mypasswd \

--net=host --pid=host -e TINI_SUBREAPER=true \

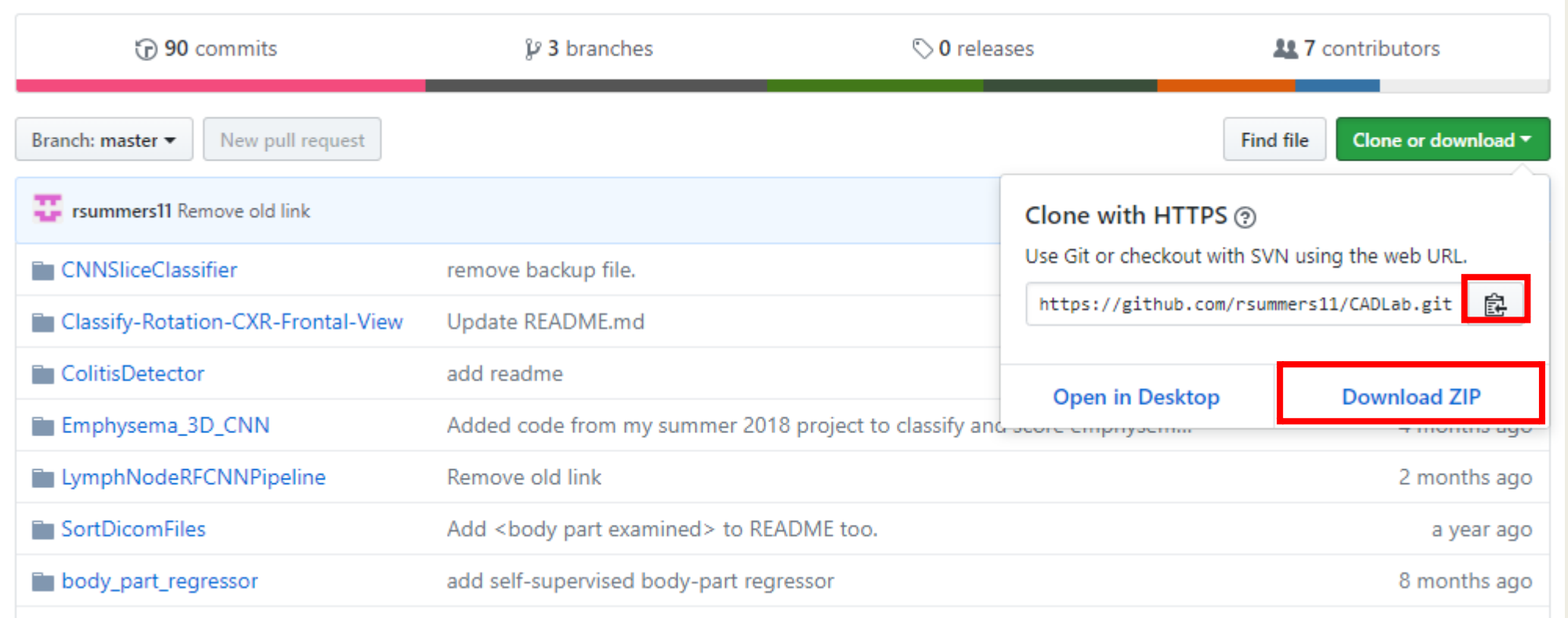
-v /root/CADLab:/root/CADLab \

mycompany.com/nvidia/mxnet:18.07-py2

Mxnet 이미지에 jupyter 추가

소스 다운로드

- https://github.com/rsummers11/CADLab
- $ git clone https://github.com/rsummers11/CADLab.git



소스 소개

- 구현한 frameworks: Faster RCNN, R-FCN, Improved R-FCN [1], 3DCE R-FCN
  - *rcnn/symbol/symbol_vgg.py*
  - *tools/train.py*
- **DeepLesion** dataset 설명
  - *Load data split and annotations from DL_info.csv (see dataset/DeepLesion.py)*
  - *Load images from 16-bit png files (see fio/load_ct_img.py)*
- CT image들 전처리
  - *슬라이스 간격에 따라 중간 슬라이스를 선형으로 보정*
  - *intensity windowing 작업 수행*
  - *픽셀 간격 표준화*
  - *검은 색 테두리를 자르기*
- 다른 유용한 특징
  - *각각의 epoch 후에 validation set으로 평가. 몇번의 epoch 후에 best model을 사용해서 test set을 평가.* (tools/train.py, validate.py, test.py, core/tester.py
  - batch size와 iter_size 을 조절 가능
  - 이전 snapshots (중간결과)를 default.yml 으로 exp_name과 begin_epoch을 설정해서 재계산 가능
  - exp_name 이름으로 로그를 남김

소스 소개

- 준비물
  - MXNet 1.0.0
  - Python 2.7
  - Before running, run "make" to compile binary files
  - To train the universal lesion detector, download the DeepLesion dataset.
- 디렉토리 구조
  - experiment_logs: log files for the results
  - images: images used in this readme.
  - rcnn: the core codes. The main function is in core/tools/train.py.
  - config.yml and default.yml: configuration files to run the code.
  - train.sh and test.sh: run these files to train or test.

소스 수정할 부분

- CADLab/lesion_detector_3DCE/default.yml
  - *dataset_path: '/home/yk/research/data/DeepLesion/‘*
    => dataset_path: '/root/data/DeepLesion/‘
  - *image_path: '/home/yk/research/data/DeepLesion/Images_png/‘*
    => image_path: '/root/data/DeepLesion/Images_png/‘