**DensePose-Android/IOS 포팅 보고서**

박세진

서태영

황혜영

* **DensePose 설치**

아래에서는 DensePose를 설치하는 방법에 대해서 설명하고 있다. 설치 방법은 <https://yongyong-e.tistory.com/69>을 참고해서 기술되었다.

※ DensePose 테스트 환경

- 소스코드: https://github.com/facebookresearch/DensePose

- OS: Ubuntu 16.04 64bit

- GPU: Nvidia GeForce GTX 1080Ti

1. NVIDIA 그래픽 드라이버 설치

$ sudo add-apt-repository ppa:graphics-drivers/ppa

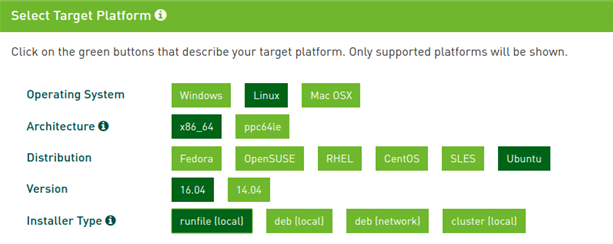
$ sudo apt-get update

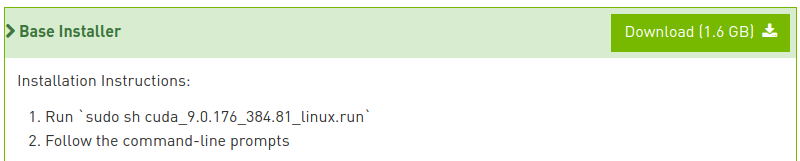
$ sudo apt-get install nvidia-396

$ sudo reboot

1. CUDA 설치

* CUDA는 9.0 버전으로 설치
* [https://developer.nvidia.com/cuda-downloads를](https://developer.nvidia.com/cuda-downloads%EB%A5%BC) 통해 CUDA Toolkit 9.0을 선택하여 아래와 같이 설치하도록 한다.





* 추가적으로 bashrc 파일에 CUDA에 대한 환경변수를 적용하도록 한다.

1. bashrc 파일 편집

$ vim .bashrc

1. 마지막 라인에 아래의 환경변수 추가

export PATH=/usr/local/cuda-9.0/bin${PATH:+:${PATH}}

export LD\_LIBRARY\_PATH=/usr/local/cuda-9.0/lib64${LD\_LIBRARY\_PATH:+:${LD\_LIBRARY\_PATH}}

1. 환경변수 적용

$ source .bashrc

3. cuDNN 설치

* cuDNN은 7.1.4 버전으로 설치
* [https://developer.nvidia.com/cudnn](https://developer.nvidia.com/cudnn%EC%9D%84) 을 통해 다운로드 가능하며 아래와 같은 버전을 받아야 한다.
* cuDNN v7.1.4 (May 16, 2018), for CUDA 9.0
* cuDNN v7.1.4 Library for LInux
* 이후 아래와 같이 /usr/local/cuda-9.2 경로에 복사한다.

$ tar xzvf cudnn-9.0-linux-x64-v7.1.tgz

$ sudo cp cuda/lib64/\* /usr/local/cuda-9.0/lib64/

$ sudo cp cuda/include/\* /usr/local/cuda-9.0/include/

$ sudo chmod a+r /usr/local/cuda-9.0/lib64/libcudnn\*

$ sudo chmod a+r /usr/local/cuda-9.0/include/cudnn.h

$ sudo apt-get install libcupti-dev

4. Anaconda2 설치

* <https://www.anaconda.com/download/#linux> 에서 Python 2.7 버전으로 다운로드하고, 아래 명령어를 통해 Anaconda2를 설치하도록 한다.
* $ bash Anaconda2-{latest}-Linux-x86\_64.sh

5. Caffe2 설치

* 먼저 caffe2가 설치 되어있을 경우 기존의 caffe2를 제거한다.

$ sudo rm -vRf /usr/local/include/caffe2

$ sudo rm -vf /usr/local/lib/libcaffe2\*

$ sudo rm -vRf /usr/local/lib/python2.7/dist-packages/caffe2

* apt 패키지 설치

$ sudo apt-get install -y --no-install-recommends build-essential cmake git libgoogle-glog-dev-libgtest-dev libiomp-dev libleveldb-dev liblmdb-dev libopencv-dev libopenmpi-dev libsnappy-dev libprotobuf-dev openmpi-bin openmpi-doc protobuf-compiler python-dev python-pip

* pip 패키지 설치

$ sudo pip install --upgrade pip

$ sudo pip install setuptools future numpy protobuf

$ sudo apt-get install -y --no-install-recommends libgflags-dev

* pytorch 소스코드 다운로드

$ mkdir caffe2-pytorch && cd caffe2-pytorch

$ git clone --recursive <https://github.com/pytorch/pytorch.git> ./

$ git submodule update --init

* caffe2 빌드

$ mkdir build && cd build

$ cmake ..

$ sudo make -j”$(nproc)” install

* caffe2의 라이브러리와 헤더의 위치 확인

$ sudo ldconfig

* caffe2의 라이브러리와 헤더의 위치 확인

1. locate 데이터베이스를 업데이트

$ sudo updatedb

1. libcaffe2.so가 /usr/local/lib에 있는지 확인

$ locate libcaffe2.so

1. 헤더 파일이 /usr/local/include에 복사되었는지 확인

$ locate caffe2 | grep /usr/local/include/caffe2

* caffe2 라이브러리 환경변수 적용

1. bashrc 파일 편집

$ vim .bashrc

1. 마지막 라인에 아래의 환경변수 추가

export PYTHONPATH=/usr/local

export PYTHONPATH=$PYTHONPATH:/home/username/caffe2-pytorch/build

export PYTHONPATH=$PYTHONPATH:/home/username/caffe2-pytorch/caffe2

export PYTHONPATH=$PYTHONPATH:/home/username/caffe2-pytorch/caffe2/build

export LD\_LIBRARY\_PATH=/usr/local/lib

1. 환경변수 적용

$ source .bashrc

* caffe2 빌드가 성공되었는지 확인

$ python -c ‘from caffe2.python import core’ 2>/dev/null && echo “Success” || echo “Failure”

* caffe2의 GPU 지원 확인 (1이상의 숫자가 출력되어야함)

$ python2 -c ‘from caffe2.python import workspace; print(workspace.NumCudaDevices())’

6. COCO API 설치

* 소스코드 다운로드 및 설치

$ git clone <https://github.com/cocodataset/cocoapi.git>

$ cd cocoapi/PythonAPI

$ make install

$ python2 setup.py install --user

7. DensePose 다운로드

* 소스코드 다운로드

$ git clone <https://github.com/facebookresearch/densepose>

* 파이썬 라이브러리 설치

$ pip install -r densepose/requirements.txt

* 파이썬 모듈 설정

$ cd densepose && make

* Detectron 테스트

$ python2 densepose/detectron/tests/test\_spatial\_narrow\_as\_op.py

* 사용자 정의 연산자 라이브러리 빌드

$ cd densepose && make ops

* 설치 체크

$ python2 densepose/detectron/tests/test\_zero\_even\_op.py

8. DensePose 데이터 다운로드

$ cd densepose/DensePoseData

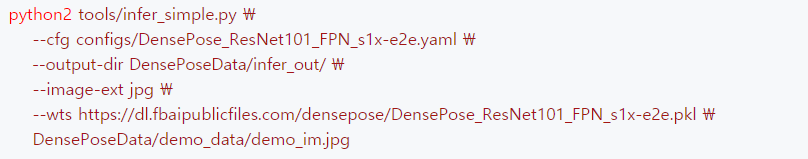
$ bash get\_densepose\_uv.sh

$ bash get\_DensePose\_COCO.sh

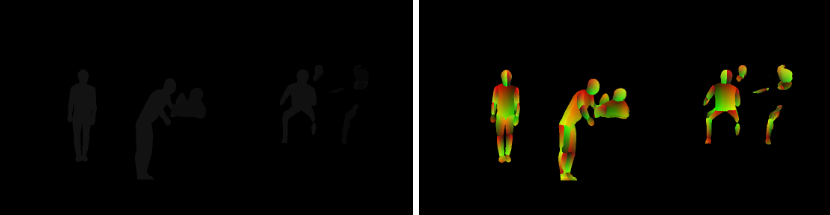
$ bash get\_eval\_data.sh

9. DensePose 테스트

* pre-trained 모델을 사용하여 이미지에 대한 densepose 테스트를 해보자.



* 명령어 실행 결과로 DensePoseData/infer\_out 디렉토리에 \*\_IUV.png 및 \*\_INDS.png 파일이 생성되면 성공이다.



* 또한 jupyter notebook을 실행하여 DensePose-RCNN의 결과를 확인할 수 있다.

$ jupyter notebook

#notebooks/DensePose-RCNN-Visualize-Results.ipynb



1. **TFLite 조사**

TFLITE는 기본적으로 TensorFlow 모델을 TFLITE모델로 변환을 시켜서 사용한다. DENSEPOSE 모델은 CAFFE2 를 사용한 모델로 TF모델이 아니기 때문에 TFLITE로 변환을 하는데 문제가 생긴다. 그래서 TFLITE로 사용하기 힘들다.

1. **Android-Studio에서 python사용하기**

python 코드를 그대로 android에서 사용해보려고 시도하던 도중 APP을 python 코드로만 짤 수 있는 BEEWARE라는 툴을 찾을 수 있었다.

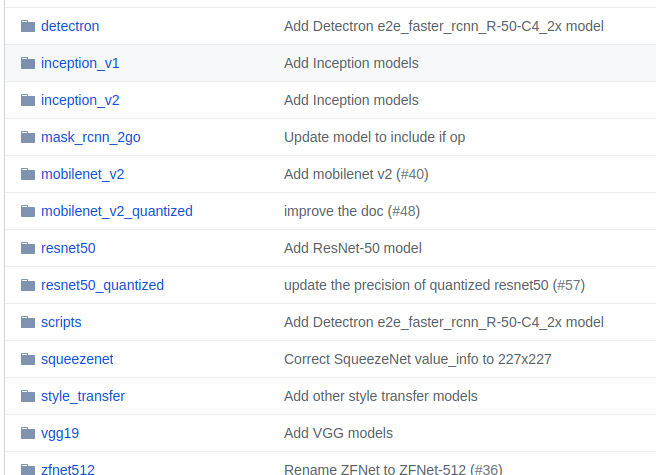
그래서 BEEWARE를 사용하여 파이썬 코드를 그대로 돌리려고 시도를 하던 도중

기본적으로 desepose는 python2 버전으로 짜여져있다는 것을 확인하였고

BEEWARE를 사용하는 방법은 일단 안된다고 판단을 했다.

1. **Caffe2 Porting(AI Camera)**

Caffe2에서 가장 중요한 것은 Model Zoo이다. 이것은 오픈 소스 라이브러리로서 여러 데이터 셋과 training된 모델들을 쉽게 불러와 사용할 수 있다.



(이런식으로 여러 모델들이 저장이 되어있다. <https://github.com/caffe2/models>)

그중에 Densepose는 detectron을 사용하였지만, Caffe2를 Android에 porting하는 기본 예시인 AI Camera부터 찾아보기로 했다.

\*Detectron: 물체인식을 위한 고성능 코드베이스이며, 빠른 구현과 연산이 유연하게 이루어지기 위해 디자인되었다. Detectron은 Facebook AI Reasearch(FAIR)에서 사용된다.

**4. AI Camera**(<https://github.com/caffe2/AICamera>)

* AI Camera 어플리케이션에 대한 이해를 위해 안드로이드 폰 상에서 코드를 실행해보았다. 아래 사진은 AI Camera 어플 실행 화면이다. 폰 카메라에 찍히는 물체를 인식하고 정확도에 따른 우선순위와 퍼센티지를 순서대로 보여준다.

****

AI Camera는 Squeezenet 모델을 사용해서 Android에 porting한다는 것을 확인하였고 그래서 Squeezenet 모델이 어떤식으로 이루어져있는지 먼저 확인을 해보았다.

다음과 같은 방식으로 이루어져있다는 것을 알게되었다.

**SQUEEZENET**

### **Architectural design strategies**

1. Replace 3x3 filters with 1x1 filters
   * 모든 3x3 convolution filter를 1x1 필터로 교체한다. 이는 1x1 필터가 3x3 필터에 비해 9배나 더 적은 파라미터를 가지고 있기 때문이다.
2. Decrease the number of input channels to 3x3 filters
   * 만약 모델의 레이어(layer) 하나가 전부 3x3 필터로 구성되어있다면 파라미터의 총 수는 (input channel) x (number of filters) x (3x3)개와 같다. 따라서 3x3 필터 자체의 수를 줄이고 이에 더해 3x3으로 들어가는 input channel의 수도 줄여야한다.
3. Downsample late in the network so that convolution layers have large activation
   * Downsampling part를 네트워크 후반부에 집중시키는 방법도 사용한다. 보통 downsample은 max(or average) pooling 또는 필터 자체의 stride를 높이는 방식으로 이미지의 spatial resolution을 줄이게된다. 이렇게 줄여서 한 번에 필터가 볼 수 있는 영역을 좁히면서 해당 이미지의 정보를 압축시키는 것이다.

일단 1번과 2번은 CNN 전체 파라미터 수를 줄이면서 정확도를 최대한 보하는 것에 초점을 맞췄습니다. 3번의 경우는 파라미터 수가 제한된 상황에서 정확도를 최대화 시키는 방식이다. 이제 fire module에서 3가지 전략을 적용시키면 된다.

이런 내용을 토대로 Squeezenet은 모델의 크기를 줄이는 방법이라고 생각을 하였고 Detectron 모델을 Squeezenet을 사용하여 바꿀 생각을 하였다.

AI Camera에서 Exporting Squeezenet to mobile.ipynb (<https://github.com/caffe2/AICamera/blob/master/Exporting%20Squeezenet%20to%20mobile.ipynb>)을 사용하여 Squeezenet모델을 최종적으로 pb파일로 변환하여 핸드폰에 포팅을 하였다. 그러나, 전에 생각했던 것과 달리 Dectron모델은 pkl파일로서 Squeezenet모델인 pth파일로 변환이 불가능했다.

여러가지 찾다보니 (<https://github.com/zetyquickly/Detectron2toMobile/blob/master/detectron2_mobile.ipynb>) Caffe2모델을 TF모델로 변환하는 방법이 가능하다는 것을 찾았다. 그러나 Caffe2에서 TF모델로 변환하는 것은 비효율적이고 잘 돌아가지도 않았다. 그래서 바로 detectron2 모델을 바로 pb파일로 바꿔서 모바일에서 사용할 수 있도록 만드는 것이 맞다는 생각을 하였다. 그래서 pkl 파일 자체를 pb파일로 변환하는 방법을 찾아보았다.

그러다 결국

(<https://github.com/facebookresearch/Detectron/blob/master/tools/convert_pkl_to_pb.py>) 이 코드를 찾을 수 있었다.

이 코드를 살펴보면 pkl을 pb파일로 바꿔서 모바일에서도 사용가능한 모델로 만들 수 있다는 것을 알 수 있었고 깃에 있는 여러 글들을 확인한 결과 오류도 해결된 것으로 확인을 하였으나 모델을 변환하여 안드로이드에 직접 적용하기까지는 시간이 턱없이 부족했다.