Tensorflow 설치 및 예제

2016.08.24

KAIST SIIT 김영수

1. 작업환경
2. UBUNTU 설치
3. CUDA 설치
4. CUDNN 설치
5. Protobuf, GRPC 설치
6. Tensorflow 설치
7. Tensorflow 예제 실행
8. Inception-v3

1. 작업환경

본 문서는 Ubuntu 14.04와 Tensorflow의 설치과정을 담고 있습니다.
작업환경은 다음과 같습니다.

- Mainboard: x99, H97
- CPU: i7-5930K, i7-4790K
- GPU: TITAN-X, P1080
- Ubuntu 14.04.3, cuda 8.0, cudnn 5.1.5, P1080
- Ubuntu 14.04.3, cuda 7.5, cudnn 5.1.3, TITAN-X
- Kernel: (uname -r) 3.13.0-24-generic
- python 2.7.6
- gcc 4.8.4

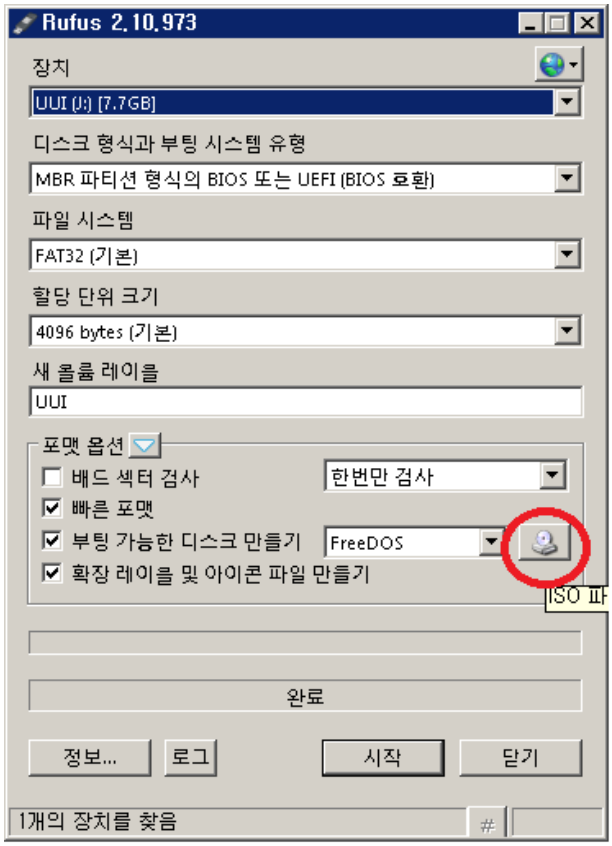
설치할 OS,프로그램 목록

- UBUNTU 14.04
- Tensorflow
- Bazel
- GRPC
- Protobuf

본 문서를 수정하여 배포하시면 안됩니다.

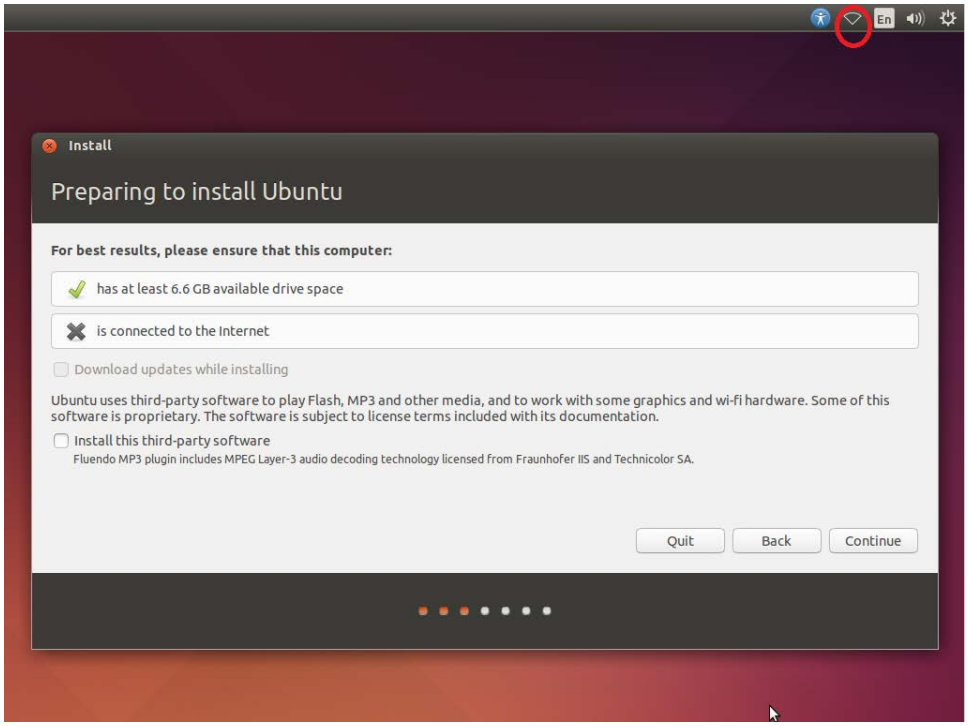
2. Ubuntu 설치

- Ubuntu는 14.04 또는 16.04를 설치합니다.
- 현재 가장 많이 사용하고 있는 버전은 14.04로서 패치도 많이 되어있고 해당OS에서 프로그램을 설치 및 실행 중 발생한 다양한 문제가 이미 해결되어 웹에 공유되어 있으므로 추천합니다.
- Download site: http://releases.ubuntu.com/14.04/
  - ubuntu-14.04.4-desktop-amd64.iso
  - ubuntu-14.04.4-desktop-amd64.iso.torrent
  - ubuntu-14.04.5-desktop-amd64.iso
  - ubuntu-14.04.5-desktop-amd64.iso.torrent
- USB를 이용한 설치
  백신프로그램 실시간 감시 정지
  https://rufus.akeo.ie/downloads/rufus-2.10.exe
  download 후 실행 ➔ ISO파일을 선택 ➔ 시작

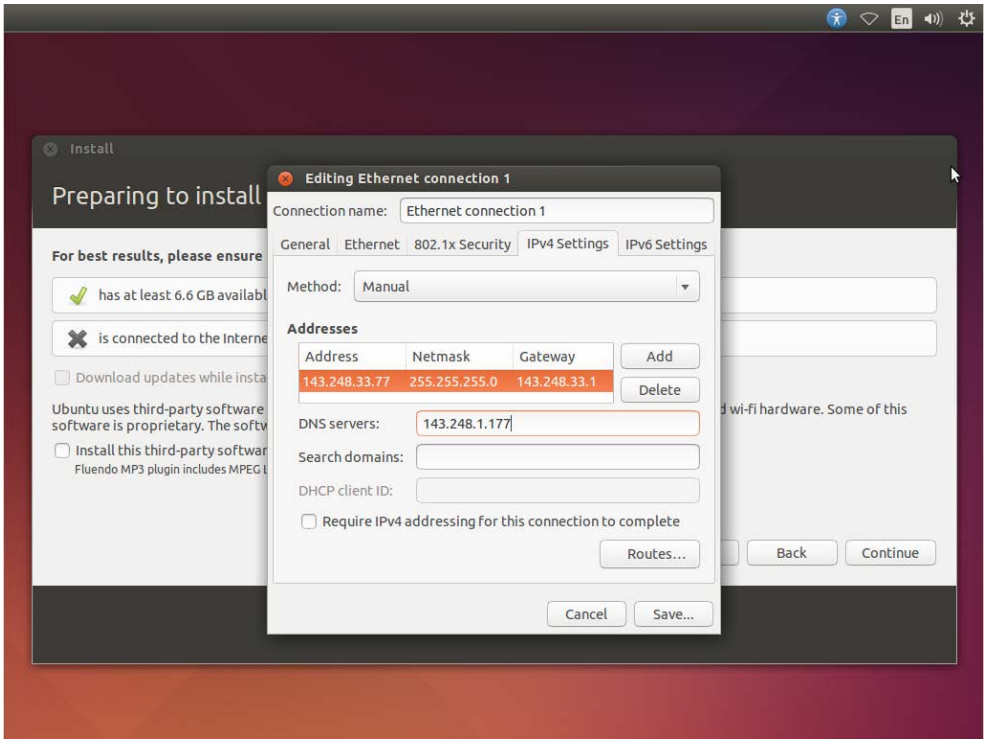


- http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows (참조)
- 독립된 UBUNTU용 HDD를 한 개 사용하는 것을 권장합니다.
- 만든 USB부팅디스크를 이용하여 부팅후 INSTALL UBUNTU를 선택.
- 언어는 English를 선택.

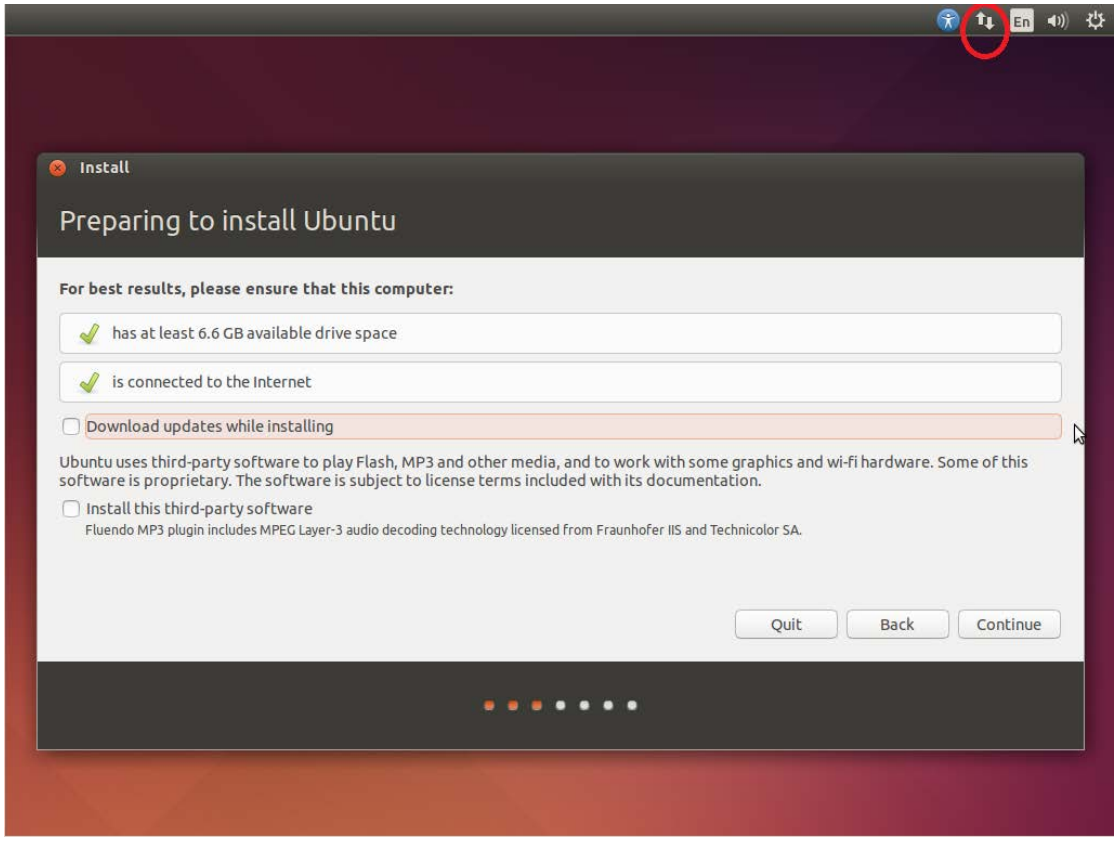
- 사용하는 IP가 있다면 Ethernet을 설정하기 위해 빨간부분을 클릭 후 "edit connections"를 클릭.



- 기존의 설정된 Ethernet설정(wired connection등)은 모두 삭제 후 IP 세팅.

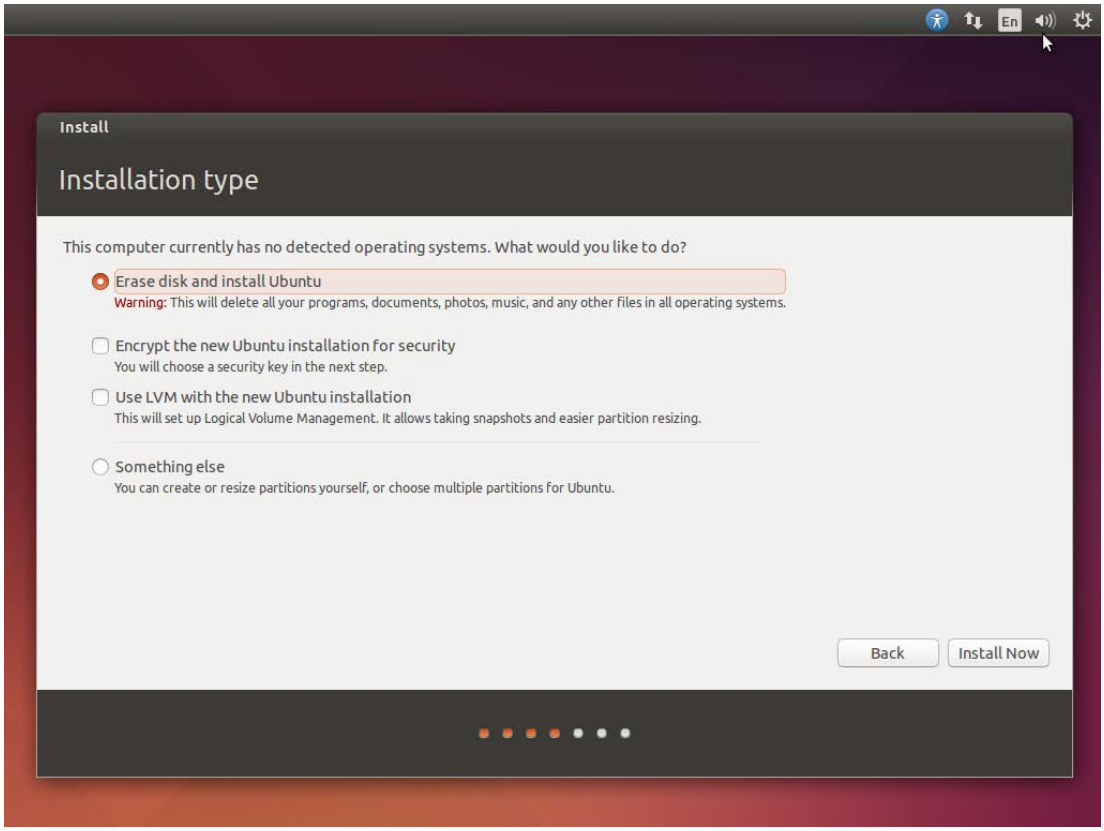


- 빨간 부분처럼 표시되면 연결 성공.



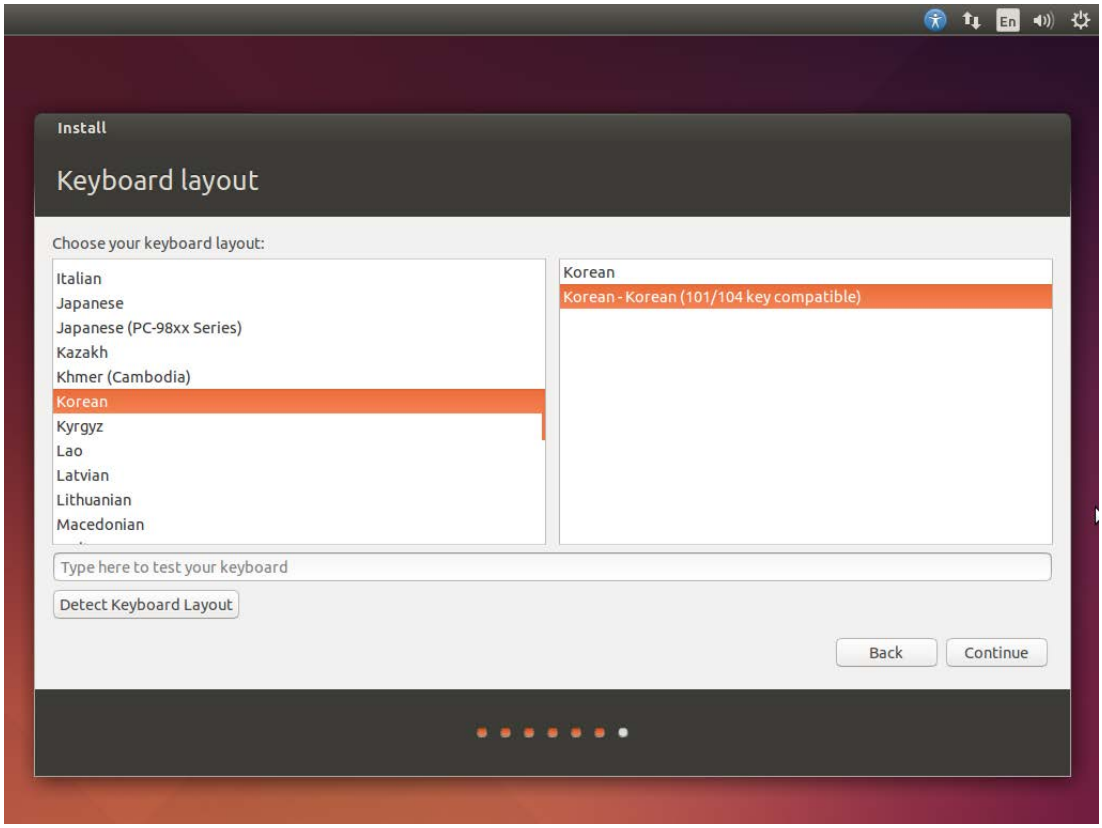
-

- Installation type : 익숙하지 않은 사용자는 간단한 설치를 위해 제일 위 옵션 선택.



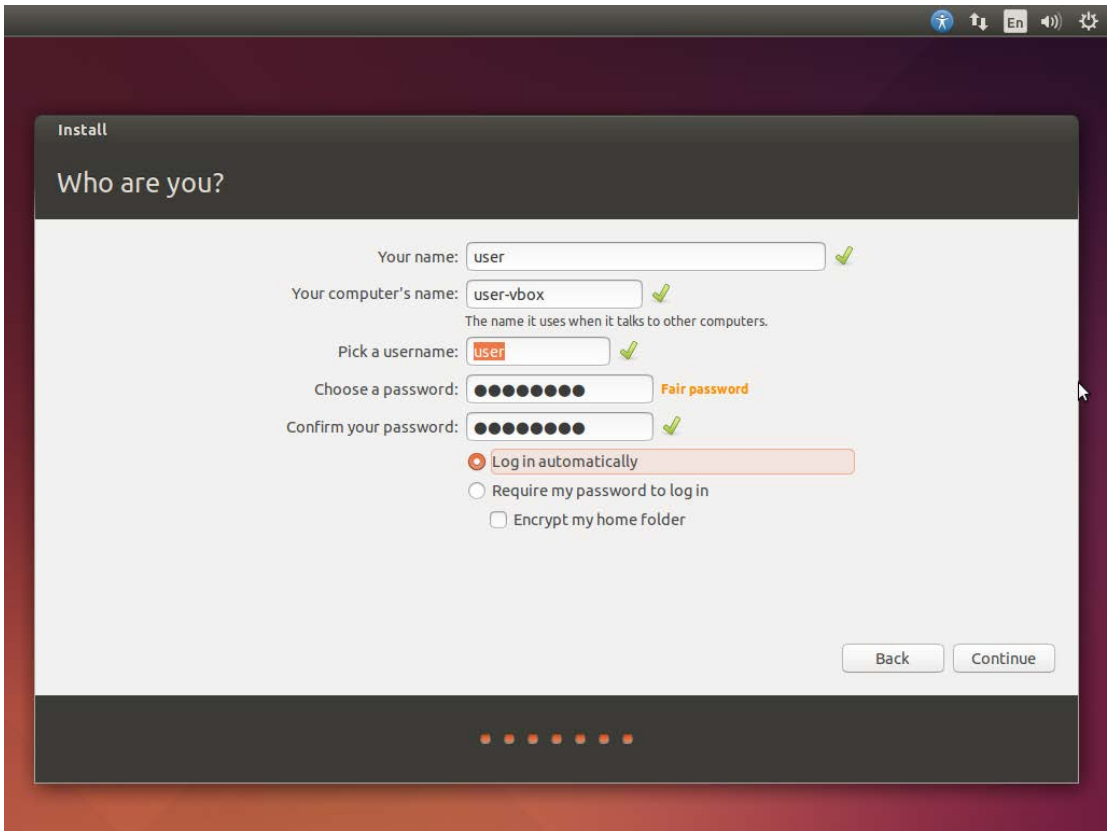
-

- 키보드 선택



-

- 사용자이름 및 비번 설정



-

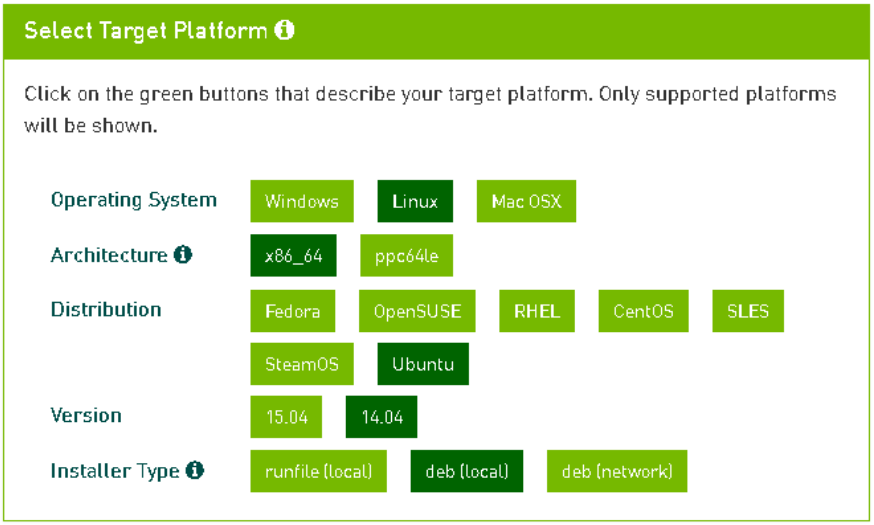
- Continue ➔ install 시작.

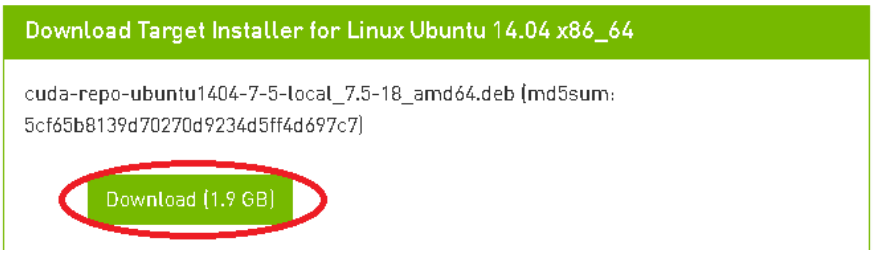
3. CUDA Install

- 7.5 또는 8.0 을 Download 해야 합니다. P1080 을 사용하는 분은 무조건 8.0 을 Download 합니다. 부팅된 UBUNTU 에서 browser(firefox)를 실행하여 다음 사이트로 이동.

- **CUDA7.5 download**

  https://developer.nvidia.com/cuda-downloads

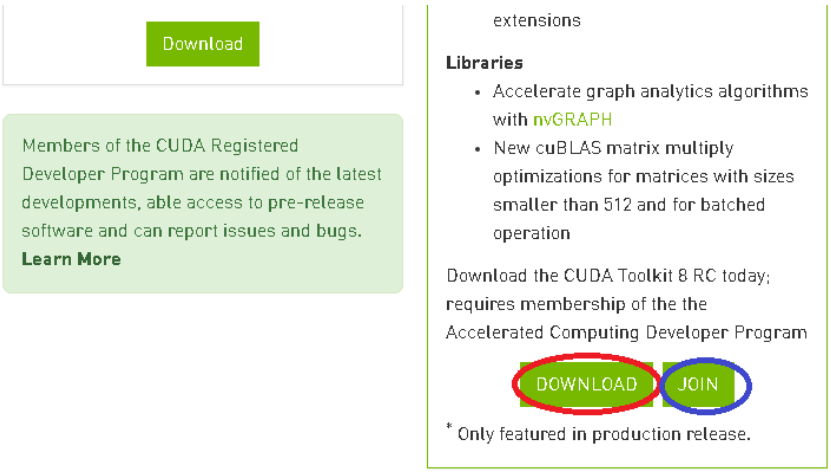




- **CUDA8.0 download**

  https://developer.nvidia.com/cuda-toolkit

  NVIDIA ID 가 없는 분은 **파란색 JOIN**, 있는 분은 DOWNLOAD



- **다운로드 하였으면 다음을 수행.**

'$~~~'와 같은 형태의 명령어는 terminal 을 실행하여 타이핑함.

$cd Downloads

$sudo dpkg –i cuda-repo-ubuntu1404~~~ (해당 폴더에 cuda 로 시작하는 파일이 1 개이면 cuda 를 타이핑한 후 "tab"키를 이용하여 자동으로 타이핑을 완성 시킬 수 있음)

$sudo apt-get update

$sudo apt-get install cuda (약 3GB 를 설치함.)

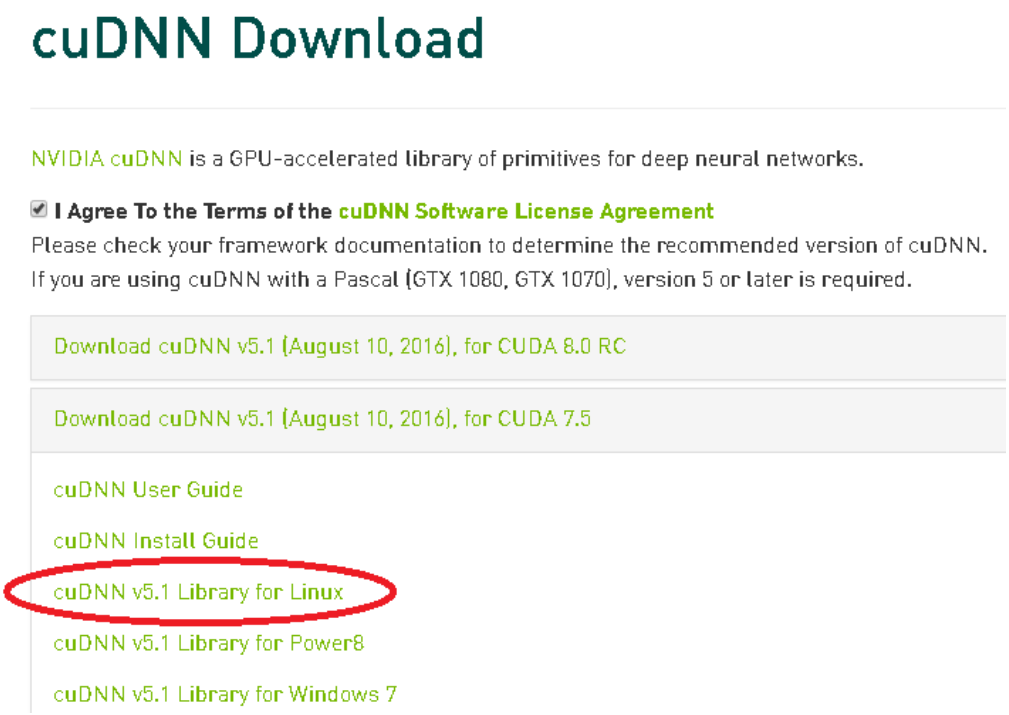
$cd ~

설치완료후 $nvidia-smi 를 수행하여 GPU 가 보이는지 확인

```
user@user-216:~$ nvidia-smi
Mon Aug 22 02:36:03 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.35                 Driver Version: 367.35                    |
|-------------------------------+----------------------+----------------------+
| 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 1080    Off  | 0000:05:00.0      On |                  N/A |
| 30%   45C    P8     7W / 180W |    239MiB /  8110MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    Off  | 0000:06:00.0     Off |                  N/A |
| 29%   44C    P8     6W / 180W |      1MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

4. CUDNN 설치

- https://developer.nvidia.com/cudnn
- Download 선택 (login 필요)
- CUDA 버전에 맞는 v5.1 을 선택하여 Linux 용을 Download



-

$cd Downloads

$tar –xvf cudnn~~~

$cd cuda

$sudo cp lib64/lib* /usr/local/cuda/lib64/

$sudo cp include/cudnn.h /usr/local/cuda/include/

$cd ~

- **SETTING**

$nano ~/.bashrc

파일의 가장 아래에 다음 3 줄을 추가함

PATH=/usr/local/cuda/bin:/usr/local/cuda/include:$PATH

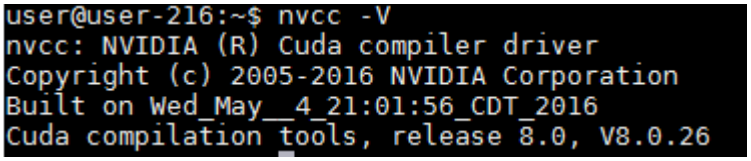
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/lib"

export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/local/cuda/lib64"

ctrl+x 를 눌러서 종료 시키며 'y' , 'enter' 를 눌러서 수정된 것을 저장.

$source ~/.bashrc (변경 내용을 반영)

$nvcc –V (다음과 같은 결과가 나와야 함)



$sudo nano /etc/ld.so.conf.d/cuda.conf

다음 두줄을 추가

/usr/local/cuda/lib64

/usr/local/cuda/lib

$sudo ldconfig (변경 내용 반영)

5. Protobuf/GRPC 설치

- Tensorflow 의 prerequisite 으로 Protobuf, 분산처리를 위해 GRPC 를 설치.

- Protobuf 설치

  $cd ~

  $sudo apt-get install git

  $sudo apt-get install curl

  $sudo apt-get install build-essential autoconf libtool

  $git clone https://github.com/google/protobuf.git

  $cd protobuf

  $./autogen.sh

  $./configure

  $make -j8

  $make check -j8

  $sudo make install -j8

  $sudo ldconfig

- GRPC 설치

  $cd ~

  $git clone https://github.com/grpc/grpc.git

  $cd grpc

  $git submodule update –init

  $make -j8

  $sudo make install -j8

  $sudo ldconfig

6. Tensorflow 설치

- https://github.com/tensorflow/tensorflow/blob/master/tensorflow/g3doc/get_started/os_setup.md#installing-from-sources (필요시 참조)

- Bazel 설치 (Google's build tool)

  $sudo add-apt-repository ppa:webupd8team/java

  $sudo apt-get update

  $sudo apt-get install oracle-java8-installer

  $sudo apt-get install software-properties-common

  $sudo apt-get install pkg-config zip g++ zlib1g-dev unzip

  $echo "deb [arch=amd64] http://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo tee /etc/apt/sources.list.d/bazel.list

  $curl https://storage.googleapis.com/bazel-apt/doc/apt-key.pub.gpg | sudo apt-key add -

  $sudo apt-get update

  $sudo apt-get install bazel

  $sudo apt-get upgrade bazel

- Tensorflow 설치

  $git clone https://github.com/tensorflow/tensorflow ~/tensorflow

  $sudo apt-get install python-numpy swig python-dev python-wheel

  $cd tensorflow

  $./configure

```
user@user-216:~/tensorflow$ ./configure
Please specify the location of python. [Default is /usr/bin/python]:
Do you wish to build TensorFlow with Google Cloud Platform support? [y/N] n
No Google Cloud Platform support will be enabled for TensorFlow
Found possible Python library paths:
  /usr/local/lib/python2.7/dist-packages
  /usr/lib/python2.7/dist-packages
Please input the desired Python library path to use.  Default is [/usr/local/lib/python2.7/dist-packages]

/usr/local/lib/python2.7/dist-packages
Do you wish to build TensorFlow with GPU support? [y/N] y
GPU support will be enabled for TensorFlow
Please specify which gcc should be used by nvcc as the host compiler. [Default is /usr/bin/gcc]:
Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave empty to use system default]: 8.0
Please specify the location where CUDA 8.0 toolkit is installed. Refer to README.md for more details. [Default is /usr/local/cuda]:
Please specify the Cudnn version you want to use. [Leave empty to use system default]: 5.1.5
Please specify the location where cuDNN 5.1.5 library is installed. Refer to README.md for more details. [Default is /usr/local/cuda]:
Please specify a list of comma-separated Cuda compute capabilities you want to build with.
You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.
Please note that each additional compute capability significantly increases your build time and binary size.
[Default is: "3.5,5.2"]: 6.1
Setting up Cuda include
Setting up Cuda lib64
Setting up Cuda bin
Setting up Cuda nvvm
Setting up CUPTI include
Setting up CUPTI lib64
Configuration finished
user@user-216:~/tensorflow$
```

path 를 설정하는 부분은 기본 path 를 그대로 사용해도 문제가 없을 겁니다. (enter = default)
문제가 생긴다면 compiler, cuda, cudnn 이 설치된 폴더를 찾아서 path 를 적어주어야 합니다.
cuda version **7.5** or **8.0** 과 cudnn 버전 **5.1.3**(cuda7.5) or **5.1.5**(cuda8.0)을 적어줍니다.

만약 cudnn version 이 5.1 이 아니라면 다음 명령으로 버전을 확인합니다.

$ cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2

capability 는 보유하고 있는 GPU 에 맞는 값을 적어 주어야 합니다.

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

위 사이트에서 본인이 보유하고 있는 GPU 를 검색. (ex 780, 960M, 980)

python package build

$bazel build -c opt --config=cuda //tensorflow/tools/pip_package:build_pip_package

- 위 명령 수행후 **kernels/BUILD: undeclared inclusion Error** 발생시 cuda7.5 또는 8.0 의 include path 를 CROSSTOOL 에 삽입.

ctrl+w 로 cxx_builtin_include_directory 를 찾아서 삽입.

$nano ~/tensorflow/third_party/gpus/crosstool/CROSSTOOL

cxx_builtin_include_directory: "/usr/local/cuda-7.5/include"

cxx_builtin_include_directory: "/usr/local/cuda-8.0/include"

```
  # TODO(bazel-team): In theory, the path here ought to exactly match the path
  # used by gcc. That works because bazel currently doesn't track files at
  # absolute locations and has no remote execution, yet. However, this will need
  # to be fixed, maybe with auto-detection?
  cxx_builtin_include_directory: "/usr/lib/gcc/"
  cxx_builtin_include_directory: "/usr/local/include"
  cxx_builtin_include_directory: "/usr/include"
  cxx_builtin_include_directory: "/usr/local/cuda-7.5/include"
  tool_path { name: "gcov" path: "/usr/bin/gcov" }
```

저장 후 다시 build $bazel build -c opt --config=cuda //tensorflow/tools/pip_package:build_pip_package

$mkdir _python_build

$cd _python_build

$ln -s ../bazel-bin/tensorflow/tools/pip_package/build_pip_package.runfiles/org_tensorflow/* .

$ln -s ../tensorflow/tools/pip_package/* .

$sudo python setup.py develop

- 설치가 잘되었는지 다음 명령으로 확인

$cd ~

$python -c "import tensorflow; print(tensorflow.__version__)"

```
user@user-216:~$ python -c "import tensorflow; print(tensorflow.__version__)"
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcublas.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcudnn.so.5.1.5 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcufft.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcurand.so.8.0 locally
0.10.0rc0
```

이제 두가지 방식으로 convolutional.py 를 실행 가능합니다.

방법 1) $cd ~

$python tensorflow/tensorflow/models/image/mnist/convolutional.py

방법 2) $cd tensorflow

$bazel build -c opt --config=cuda //tensorflow/models/image/mnist:convolutional

$bazel-bin/tensorflow/models/image/mnist/convolutional --use_gpu

7. 예제 실행

1) mn1_1.py

Mnist 기본 예제

설명은 코드내 주석 참조

예제 파일을 모두 home 의 tensor_ex 폴더로 이동

$mkdir ~/tensor_ex

mn1_1.py 파일이 있는 폴더로 이동

$cp *.py ~/tensor_ex

$cd ~/tensor_ex

실행

$python mn1_1.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating
step 0, training accuracy 0
step 100, training accuracy 0.6875
step 200, training accuracy 0.6875
step 300, training accuracy 0.75
step 400, training accuracy 0.9375
step 500, training accuracy 0.875
step 600, training accuracy 0.875
step 700, training accuracy 0.9375
step 800, training accuracy 0.875
step 900, training accuracy 0.9375
test accuracy 0.9174
```

2) mn1_2.py

mn1_1 에서 conv + relu + pooling layer 를 추가함.

$python mn1_2.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creat
step 0, training accuracy 0.0625
step 100, training accuracy 0.75
step 200, training accuracy 0.5625
step 300, training accuracy 0.75
step 400, training accuracy 0.9375
step 500, training accuracy 0.875
step 600, training accuracy 0.9375
step 700, training accuracy 0.75
step 800, training accuracy 0.9375
step 900, training accuracy 0.9375
test accuracy 0.9353
```

3) mn1_3.py

mn1_2 에서 L2 regularization loss 를 추가함.

Momentum optimizer 로 변경.

$python mn1_3.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creat
step 0, training accuracy 0.125
step 100, training accuracy 0.75
step 200, training accuracy 0.8125
step 300, training accuracy 0.9375
step 400, training accuracy 1
step 500, training accuracy 0.9375
step 600, training accuracy 0.9375
step 700, training accuracy 1
step 800, training accuracy 0.9375
step 900, training accuracy 1
test accuracy 0.9651
```

4) mn2_1.py

mn1_3 에서 variable saver 를 추가함.

log 폴더생성

$mkdir log

$python mn2_1.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creat
step 0, training accuracy 0.0625
step 100, training accuracy 0.75
step 200, training accuracy 0.6875
step 300, training accuracy 0.875
step 400, training accuracy 1
step 500, training accuracy 0.875
step 600, training accuracy 0.9375
step 700, training accuracy 1
step 800, training accuracy 1
step 900, training accuracy 1
Total Iteration is 1000
test accuracy 0.9604
```

5) mn2_2.py

mn2_1 에서 저장한 variable 을 load.

$python mn2_2.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creat
test accuracy 0.9604 at step=0
step 0, training accuracy 0.9375
step 100, training accuracy 1
step 200, training accuracy 0.8125
step 300, training accuracy 1
step 400, training accuracy 1
step 500, training accuracy 0.9375
step 600, training accuracy 1
step 700, training accuracy 1
step 800, training accuracy 1
step 900, training accuracy 1
Total Iteration is 2000
test accuracy 0.9769
```

Iteration0 의 test 결과 mn2_1 의 accuracy(0.9604)와 동일한 것을 확인.

6) mn2_3.py

mn2_2 에서 저장한 variable 의 일부만 load

$python mn2_3.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creat
step 0, training accuracy 0
step 100, training accuracy 0.875
step 200, training accuracy 0.8125
step 300, training accuracy 0.9375
step 400, training accuracy 1
step 500, training accuracy 0.9375
step 600, training accuracy 0.9375
step 700, training accuracy 0.9375
step 800, training accuracy 1
step 900, training accuracy 1
Total Iteration is 3000
test accuracy 0.9744
```

Iteration0 에서의 train 정확도가 0 인 것을 확인.

7) mn3_1.py

tensorboard 사용 예제

terminal1) $python mn3_1.py

tensorboard monitoring 을 위해 iteration 10000 번 설정.

terminal 창을 한 개 더 실행

terminal2) $python ~/tensorflow/tensorflow/tensorboard/tensorboard.py --logdir=./tensorboard

실행중인 PC 의 IP 를 123.123.222.111 라 할 때 http://123.123.222.111:6006 으로 web page 를 open

tensorboard 종료시

terminal1) ctrl+c

terminal2) ctrl+c

terminal2 가 종료되지 않을경우

$ps -al

```
root@user-216:~# ps -al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0   816 18344  0  80   0 - 18174 poll_s pts/17   00:00:00 sudo
4 S     0   817   816  0  80   0 -  7375 wait   pts/17   00:00:00 bash
0 S  1000  3211  3186  1  80   0 - 286424 poll_s pts/6   00:00:00 python
4 R     0  3222   817  0  80   0 -  3857 -      pts/17   00:00:00 ps
```

python 명령의 PID 를 kill 명령으로 종료.

$sudo kill -9 3211

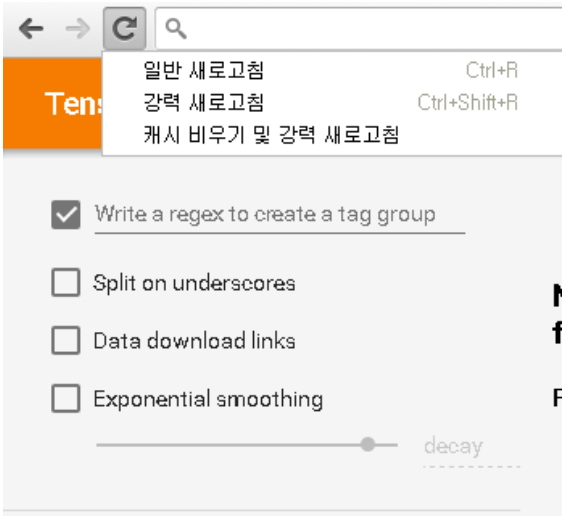
tensorboard graph 가 겹칠때

$rm -rf tensorboard

webpage 의 cache 비우기.

방법 1. ctrl+F5

방법 2. F12 ➔ 새로고침 우클릭 ➔ 캐시 비우기 및 강력 새로고침 선택



8) cf4_1.py

cifar-10 데이터셋을 이용

multi gpu 를 사용하는 코드

get_variable()을 사용하는 부분과 gradient average 하는 부분이 중요.

$python mn4_1.py

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:118] Found device 1 with properties:
name: GeForce GTX 1080
major: 6 minor: 1 memoryClockRate (GHz) 1.7335
pciBusID 0000:05:00.0
Total memory: 7.92GiB
Free memory: 7.58GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0 1
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y Y
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 1:   Y Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating TensorFlow device (/gpu:0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating TensorFlow device (/gpu:1)
step 0, loss = 2.28
step 100, loss = 2.21
step 200, loss = 2.14
step 300, loss = 2.15
step 400, loss = 2.08
step 500, loss = 2.14
step 600, loss = 2.08
step 700, loss = 2.07
step 800, loss = 2.10
```

빨간 부분은 GPU 간의 DMA(Direct Memory Access) 가능 여부를 보여줌.

실행 후 terminal2 에서 $nvidia-smi 명령으로 2 개의 GPU 가 사용되는 것을 볼수있음.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.35                 Driver Version: 367.35                    |
|-------------------------------+----------------------+----------------------+
| 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 1080    Off  | 0000:05:00.0      On |                  N/A |
| 30%   48C    P2    43W / 180W |    722MiB /  8110MiB |     15%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    Off  | 0000:06:00.0     Off |                  N/A |
| 30%   48C    P2    38W / 180W |    484MiB /  8113MiB |     15%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 1080    Off  | 0000:09:00.0     Off |                  N/A |
| 27%   39C    P8    10W / 180W |      1MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      1323    G   /usr/bin/X                                   120MiB |
|    0      2540    G   compiz                                       117MiB |
|    0     14746    C   python                                       481MiB |
|    1     14746    C   python                                       481MiB |
+-----------------------------------------------------------------------------+
user@user-216:~$ nvidia-smi
Tue Aug 23 15:08:54 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.35                 Driver Version: 367.35                    |
|-------------------------------+----------------------+----------------------+
| 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 1080    Off  | 0000:05:00.0      On |                  N/A |
| 30%   48C    P2    42W / 180W |    722MiB /  8110MiB |     12%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 1080    Off  | 0000:06:00.0     Off |                  N/A |
| 30%   48C    P2    41W / 180W |    484MiB /  8113MiB |     12%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 1080    Off  | 0000:09:00.0     Off |                  N/A |
| 27%   39C    P8    10W / 180W |      1MiB /  8113MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      1323    G   /usr/bin/X                                   120MiB |
|    0      2540    G   compiz                                       117MiB |
|    0     14746    C   python                                       481MiB |
|    1     14746    C   python                                       481MiB |
+-----------------------------------------------------------------------------+
```

8. Inception-v3

- 본 예제에서 inception-v3 를 분산처리, Multi-GPU 환경에서 실행시켜봅니다.
- paper: Christian Szegedy ,'Rethinking the Inception Architecture for Computer Vision', 2015
  http://arxiv.org/pdf/1512.00567v3.pdf

- **Dataset download**

  http://image-net.org/challenges/ilsvrc+coco2016

  위 사이트로 이동 후 "To download challenge data for ILSVRC2016, please register at here." 클릭 Sign Up 후 이메일로 온 링크를 클릭.

  "CLS-LOC dataset"를 download (flashget 등을 이용하면 빠르게 download 가능함)

  $cd ~/tensor_ex

  $mkdir data

  다운받은 파일을 data 폴더로 이동

  $cd Downloads

  $mv ILSVRC2016_CLS-LOC.tar ~/tensor_ex/data

  $cd ~/tensor_ex/data

  $tar –xvf ILSVRC2016_CLS-LOC.tar

- **Dataset 생성**

  validation data 정리

  $cd ~/tensor_ex

  $CMD_PY=~/tensor_ex/models/inception/inception/data/preprocess_imagenet_validation_data.py

  $DATA_DIR=~/tensor_ex/data/ILSVRC2016/Data/CLS-LOC

  $VAL_SYN=~/tensor_ex/models/inception/inception/data/imagenet_2012_validation_synset_labels.txt

  $python $CMD_PY $DATA_DIR/val/ $VAL_SYN

  validation 파일을 synset 폴더로 정리함. 5 분정도 소요.

  bounding box "xml ➔ csv" 정리

  $cd ~/tensor_ex

  $CMD_PY=~/tensor_ex/models/inception/inception/data/process_bounding_boxes.py

  $SYNSETS=~/tensor_ex/models/inception/inception/data/imagenet_lsvrc_2015_synsets.txt

  $BOUNDING_BOX_DIR=~/tensor_ex/data/ILSVRC2016/Annotations/CLS-LOC/train/

  $BOUNDING_BOX_FILE=~/tensor_ex/data/ILSVRC2016/Annotations/CLS-LOC/imagenet_2015_bounding_boxes.csv

  $python $CMD_PY $BOUNDING_BOX_DIR $SYNSETS | sort >$BOUNDING_BOX_FILE

```
user@user-216:~/tensor_ex$ python $CMD_PY $BOUNDING_BOX_DIR $SYNSETS | sort >$BOUNDING_BOX_FILE
Identified 544546 XML files in /home/user/tensor_ex/data/ILSVRC2015/Annotations/CLS-LOC/train/
Identified 1000 synset IDs in /home/user/tensor_ex/models/inception/inception/data/imagenet_lsvrc_2015_synsets.txt
--> processed 1 of 544546 XML files.
--> skipped 0 boxes and 0 XML files.
--> processed 5001 of 544546 XML files.
--> skipped 0 boxes and 0 XML files.
--> processed 10001 of 544546 XML files.
--> skipped 0 boxes and 0 XML files.
```

dataset build

$cd ~/tensor_ex

$mkdir tf

$CMD_PY=~/tensor_ex/models/inception/inception/data/build_imagenet_data.py

$METAFILE=~/tensor_ex/models/inception/inception/data/imagenet_metadata.txt

$python $CMD_PY --train_directory=$DATA_DIR/train/ --validation_directory=$DATA_DIR/val/ --output_directory=tf --imagenet_metadata_file=$METAFILE --labels_file=$SYNSETS --bounding_box_file=$BOUNDING_BOX_FILE

```
I tensorflow/core/common_runtime/gpu/gpu_init.cc:138] DMA: 0 1 2
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 0:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 1:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_init.cc:148] 2:   Y Y Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating TensorFlow device (/gpu
 1080, pci bus id: 0000:09:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating TensorFlow device (/gpu
 1080, pci bus id: 0000:06:00.0)
I tensorflow/core/common_runtime/gpu/gpu_device.cc:869] Creating TensorFlow device (/gpu
 1080, pci bus id: 0000:05:00.0)
2016-08-24 15:26:59.554015 [thread 5]: Wrote 390 images to tf/validation-00080-of-00128
2016-08-24 15:27:00.025772 [thread 2]: Wrote 390 images to tf/validation-00032-of-00128
2016-08-24 15:27:00.040060 [thread 0]: Wrote 390 images to tf/validation-00000-of-00128
2016-08-24 15:27:00.448040 [thread 6]: Wrote 390 images to tf/validation-00096-of-00128
2016-08-24 15:27:00.606375 [thread 7]: Wrote 390 images to tf/validation-00112-of-00128
2016-08-24 15:27:01.457401 [thread 1]: Wrote 390 images to tf/validation-00016-of-00128
2016-08-24 15:27:01.774212 [thread 4]: Wrote 390 images to tf/validation-00064-of-00128
2016-08-24 15:27:02.036259 [thread 3]: Wrote 390 images to tf/validation-00048-of-00128
2016-08-24 15:27:23.832335 [thread 0]: Wrote 391 images to tf/validation-00001-of-00128
```

Inception-v3 실행

$cd ~/tensor_ex/models/inception

$bazel build inception/imagenet_train

$bazel-bin/inception/imagenet_train --num_gpus=1 --batch_size=4 --train_dir=/home/user/tensor_ex/log --data_dir=/home/user/tensor_ex/tf

여기서 train_dir 와 data_dir 의 위치는 / (root)기준으로 직접 설정하셔야 합니다.

즉, $cd /home/user/tensor_ex/tf 를 수행하면 위에서 만든 file 들이 보여야 합니다.

```
user@user-216:~/tensor_ex/tf$ cd /home/user/tensor_ex/tf
user@user-216:~/tensor_ex/tf$ ls
train-00000-of-01024  train-00231-of-01024  train-00462-of-01024  train-00693-of-01024  train-00924-of-01024
train-00001-of-01024  train-00232-of-01024  train-00463-of-01024  train-00694-of-01024  train-00925-of-01024
train-00002-of-01024  train-00233-of-01024  train-00464-of-01024  train-00695-of-01024  train-00926-of-01024
train-00003-of-01024  train-00234-of-01024  train-00465-of-01024  train-00696-of-01024  train-00927-of-01024
train-00004-of-01024  train-00235-of-01024  train-00466-of-01024  train-00697-of-01024  train-00928-of-01024
train-00005-of-01024  train-00236-of-01024  train-00467-of-01024  train-00698-of-01024  train-00929-of-01024
train-00006-of-01024  train-00237-of-01024  train-00468-of-01024  train-00699-of-01024  train-00930-of-01024
```

GPU 개수와 GPU memory 크기에 따라 num_gpus 와 batch_size 를 변경하시면 됩니다.

Inception-v3 분산처리실행

Protobuf, GRPC, Tensorflow 가 설치된 PC 가 두대 있어야 합니다.

두대 모두 dataset 을 가지고 있어야 합니다. 두대 모두 바로 전 단계의 명령을 실행시켜보는 것이 좋습니다.

두대 모두 IP 를 확인해 놓습니다.

$ifconfig 이 명령을 수행하면 IP (inet addr)를 확인할 수 있습니다.

xxx.xxx.xxx.xxx : PC1 의 IP, 000.000.000.000 : PC2 의 IP

둘중 한대 또는 두대 모두에 parameter server 를 실행시켜야 하므로 parameter server 를 실행시킬 PC 에는 terminal 을 2 개 열어놓습니다. 다른 PC 에는 1 개 열어놓습니다.

terminal 1,2,3 에서 다음을 수행

$cd ~/tensor_ex/models/inception

$bazel build inception/imagenet_distributed_train

worker 를 모두 실행시켜 놓고 ps 를 실행시킵니다.

terminal3 에서 다음을 수행

$CUDA_VISIBLE_DEVICES='0' bazel-bin/inception/imagenet_distributed_train --batch_size=4 --data_dir=/home/siit/tensor_ex/tf --job_name='worker' --task_id=1 --ps_hosts='xxx.xxx.xxx.xxx:2222' --worker_hosts=' xxx.xxx.xxx.xxx:2223, 000.000.000.000:2223'

terminal2 에서 다음을 수행

$CUDA_VISIBLE_DEVICES='0' bazel-bin/inception/imagenet_distributed_train --batch_size=4 --data_dir=/home/user/tensor_ex/tf --job_name='worker' --task_id=0 --ps_hosts='xxx.xxx.xxx.xxx:2222' --worker_hosts=' xxx.xxx.xxx.xxx:2223, 000.000.000.000:2223'

terminal1 에서 다음을 수행

$CUDA_VISIBLE_DEVICES='' bazel-bin/inception/imagenet_distributed_train --job_name='ps' --task_id=0 --ps_hosts='xxx.xxx.xxx.xxx:2222' --worker_hosts=' xxx.xxx.xxx.xxx:2223, 000.000.000.000:2223'

warning 은 code 가 작성될때의 tensorflow 와 실행중인 tensorflow 의 버전이 다르며 몇몇 class 가 변경되어서 생기는 것입니다. 일단 무시하셔도 괜찮습니다.

terminal2 의 실행 화면

```
INFO:tensorflow:SyncReplicas enabled: replicas_to_aggregate=2; total_num_replicas=2
INFO:tensorflow:2016-08-24 17:41:34.506281 Supervisor
INFO:tensorflow:Started 3 queues for processing input data.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Worker 0: 2016-08-24 17:42:57.969784: step 0, loss = 13.06(0.1 examples/sec; 45.402  sec/batch)
INFO:tensorflow:Worker 0: 2016-08-24 17:43:45.914237: step 30, loss = 17.38(2.1 examples/sec; 1.931  sec/batch)
INFO:tensorflow:global_step/sec: 0.369983
INFO:tensorflow:Worker 0: 2016-08-24 17:44:44.044067: step 60, loss = 14.11(2.1 examples/sec; 1.936  sec/batch)
INFO:tensorflow:Running Summary operation on the chief.
INFO:tensorflow:Finished running Summary operation.
INFO:tensorflow:Worker 0: 2016-08-24 17:45:40.943386: step 90, loss = 14.08(2.1 examples/sec; 1.922  sec/batch)
INFO:tensorflow:global_step/sec: 0.517156
INFO:tensorflow:Worker 0: 2016-08-24 17:46:38.754841: step 120, loss = 13.56(2.1 examples/sec; 1.919  sec/batch)
```

terminal3 의 실행 화면

```
atchNorm/moving_mean/ExponentialMovingAverage, mixed_8x8x2048b/branch_pool/Conv/BatchNorm/moving_variance/Exponer
INFO:tensorflow:Started 3 queues for processing input data.
INFO:tensorflow:Worker 1: 2016-08-24 17:42:43.834732: step 0, loss = 13.08(0.1 examples/sec; 27.513  sec/batch)
INFO:tensorflow:Worker 1: 2016-08-24 17:42:45.760475: step 0, loss = 12.99(2.1 examples/sec; 1.926  sec/batch)
INFO:tensorflow:Worker 1: 2016-08-24 17:43:53.361862: step 30, loss = 18.38(2.1 examples/sec; 1.940  sec/batch)
INFO:tensorflow:Worker 1: 2016-08-24 17:44:51.470710: step 60, loss = 13.18(2.1 examples/sec; 1.927  sec/batch)
INFO:tensorflow:Worker 1: 2016-08-24 17:45:48.193253: step 90, loss = 13.71(2.1 examples/sec; 1.926  sec/batch)
INFO:tensorflow:Worker 1: 2016-08-24 17:46:46.005617: step 120, loss = 13.39(2.1 examples/sec; 1.919  sec/batch)
```