

Google Colab Free GPU Tutorial

Keras, Tensorflow, 및 PyTorch를 사용하여 무료 **Tesla K80 GPU** 에서 Google Colaboratory를 통해 딥러닝 응용프로그램을 개발할 수 있습니다.

Google Colab 이란?

Google Colab은 AI개발자들을 위해 구글에서 제공하는 무료 클라우드 서비스입니다.

Colab으로 무료 GPU에서 딥러닝 응용 프로그램을 개발할 수 있습니다.

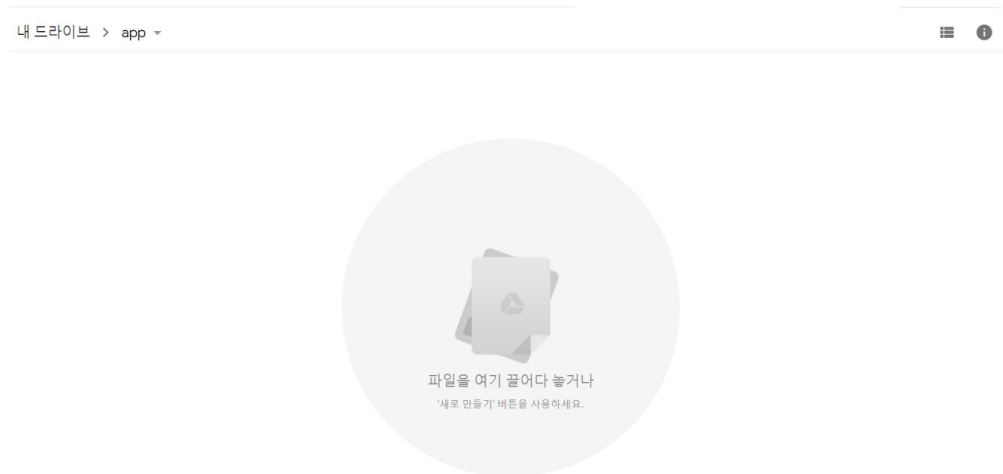
Keras, TensorFlow, PyTorch, OpenCV와 같은 라이브러리를 사용하여 딥러닝 응용프로그램을 개발해보세요!!!

<구글 Colab을 사용하기 위한 준비>


1.구글 드라이브에서 폴더 생성

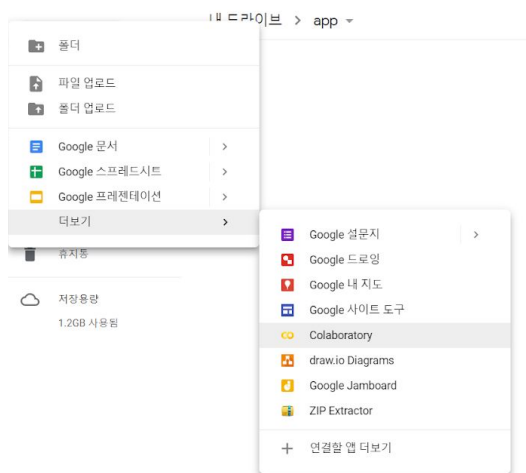


Colab은 자신의 구글 드라이브에서 작동하기 때문에, 작업할 폴더를 먼저 만들어야 합니다. 예시를 위해 드라이브에 “app”이라는 이름의 폴더를 생성했습니다. (폴더를 만들지 않아도 상관 없습니다)

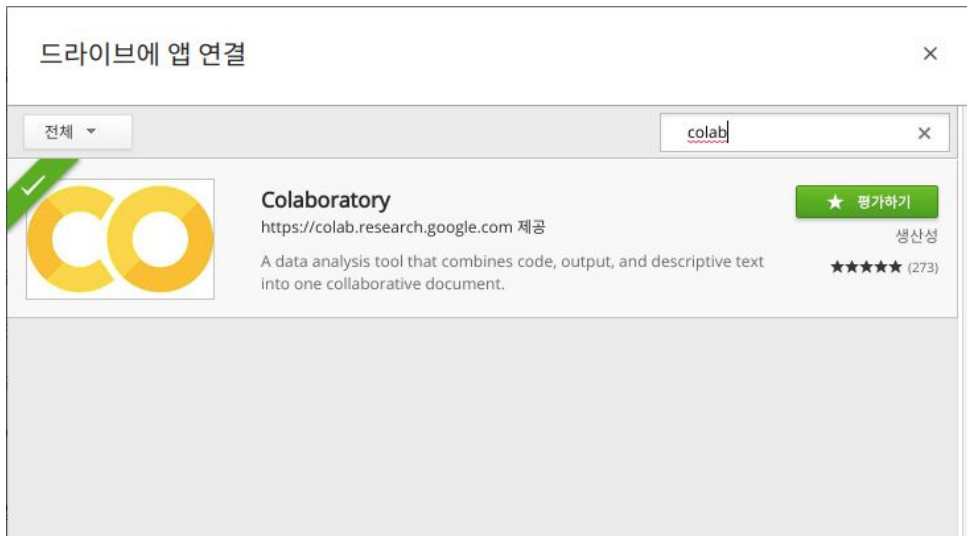


2.Colab Notebook 파일 생성

좌측의  새로 만들기 버튼 누르기 > 더보기 > Colaboratory



만약 Colaboratory가 보이지 않는다면,
연결할 앱 더보기 > colab 검색 > 연결



3. 무료 GPU 설정

수정 > 노트 설정 or 런타임 > 런타임 유형 변경 선택을 통해 하드웨어 가속기로써 GPU를 쉽게 변경할 수 있다.



4. 구글 Colab에서 .py파일 실행 및 가져오기

이제 구글 Colab을 실행할 수 있습니다.

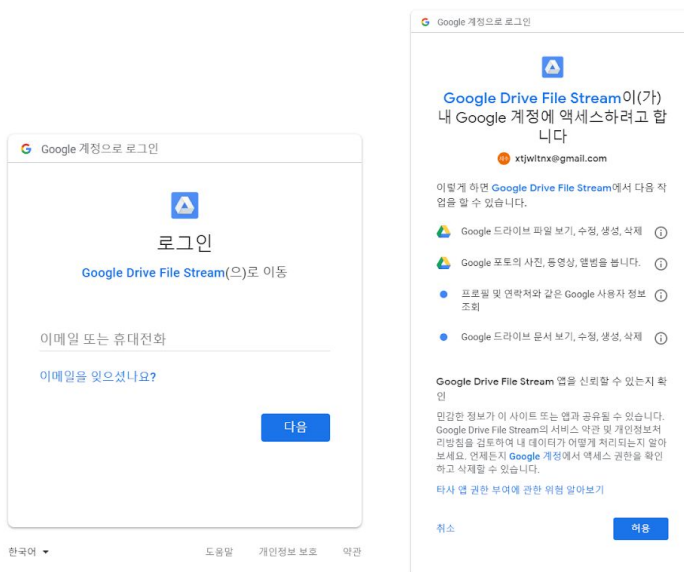
필수적인 라이브러리 설치와 권한 부여를 위해 아래 코드를 실행하세요

```
from google.colab import drive
drive.mount('/content/drive/')
```

위의 코드를 입력하시고 실행하면 결과로 아래와 같이 링크를 얻을 수 있습니다.



링크 클릭 -> 구글 로그인



로그인

이 코드를 복사하여 애플리케이션으로 전환한 다음 붙여넣으세요.
4/qwDvaVMxvV2EbnV6saNyJMMWamx8GclRe95KOzH.RrhSASdsXUUL_8

인증코드를 복사하여 텍스트 박스에 붙여 넣으세요

권한 인증 절차가 완료되면 아래와 같은 화면을 보실 수 있습니다.



이제 아래와 같은 입력으로 구글 드라이브에 성공적으로 들어가실 수 있습니다.

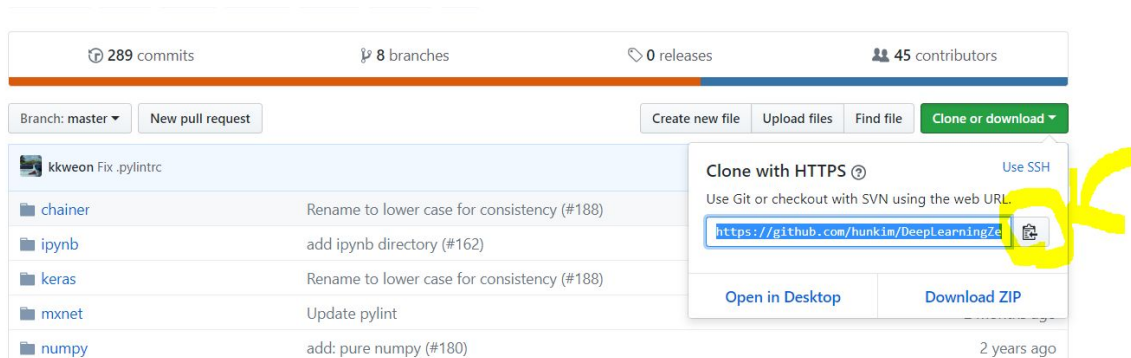


Github 저장소 구글 Colab에 클론하는 방법

Step 1: 원하는 Git 링크 복사

예시) <https://github.com/hunkim/DeepLearningZeroToAll>

클론 또는 다운로드 > 링크 복사



Step 2 : Git Clone하기




실행 명령어: !git clone <https://github.com/hunkim/DeepLearningZeroToAll>

```
!git clone https://github.com/hunkim/DeepLearningZeroToAll.git

Cloning into 'DeepLearningZeroToAll'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 1505 (delta 7), reused 12 (delta 2), pack-reused 1481
Receiving objects: 100% (1505/1505), 672.71 KiB | 3.13 MiB/s, done.
Resolving deltas: 100% (998/998), done.
```

Step 3: 구글 드라이브에서 폴더 열기

폴더는 github repository와 똑같이 드라이브에 올라갑니다

이름 ↓	최종 수정 날짜	크기
 DeepLearningZeroToAll	2018. 12. 7. 나	—
 Untitled0.ipynb	2018. 12. 8. 나	27KB
 OpenSource	오후 9:00 나	—

4.노트북 파일 열기

이름 ↓	최종 수정 날짜	크기
tests	2018. 12. 7. 나	—
pytorch	2018. 12. 7. 나	—
numpy	2018. 12. 7. 나	—
mxnet	2018. 12. 7. 나	—
keras	2018. 12. 7. 나	—
ipynb	2018. 12. 7. 나	—
chainer	2018. 12. 7. 나	—
.git	2018. 12. 7. 나	—
requirements.txt	2018. 12. 7. 나	109바이트
README.md	2018. 12. 7. 나	2KB
lab-13-3-mnist_save_restore.py	2018. 12. 7. 나	6KB
lab-13-2-mnist_tensorboard.py	2018. 12. 7. 나	5KB
lab-13-1-mnist_using_scope.py	2018. 12. 7. 나	4KB
lab-12-5-rnn_stock_prediction.py	2018. 12. 7. 나	3KB

이름 ↓	최종 수정 날짜	크기
lab-12-0-rnn_basics.ipynb	오후 9:08 나	28KB
lab-11-0-cnn_basics.ipynb	2018. 12. 7. 나	39KB
lab-10-6-mnist_nn_basics.ipynb	2018. 12. 7. 나	146KB
lab-08-tensor_manipulation.ipynb	2018. 12. 7. 나	22KB
lab-03-X-minimizing.ipynb	2018. 12. 7. 나	10KB
lab-03-3-minimizing.ipynb	2018. 12. 7. 나	5KB

원하는 파일에서 우클릭>연결 앱>Colaboratory

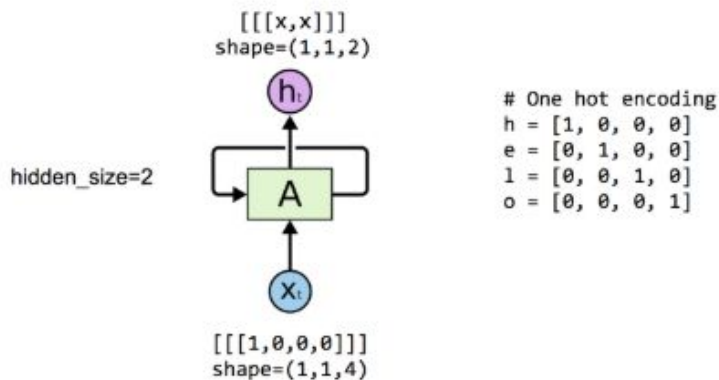
5. 실행

이제 구글Colab에서 Github repository를 이용하실 수 있습니다.


```
# http://www.wildml.com/2016/08/rnns-in-tensorflow-a-practical-guide-and-undocumented-features/
# http://learningtensorflow.com/index.html
# http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/
```

```
import tensorflow as tf
import numpy as np
from tensorflow.contrib.rnn import rnn
import pprint
pp = pprint.PrettyPrinter(indent=4)
```

```
[ ] # One hot encoding for each char in 'hello'
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
```



```
[ ] with tf.variable_scope('one_cell') as scope:
    # One cell RNN input_dim (4) -> output_dim (2)
    hidden_size = 2
    cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)
    print(cell.output_size, cell.state_size)

    x_data = np.array([h], dtype=np.float32) # x_data = [[[1,0,0,0]]]
    pp.pprint(x_data)
    outputs, _states = tf.nn.dynamic_rnn(cell, x_data, dtype=tf.float32)

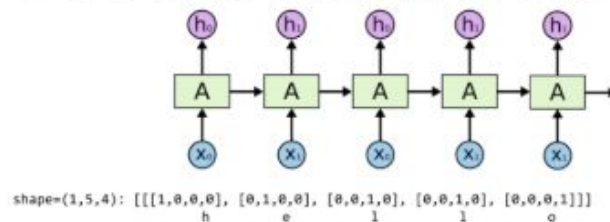
    sess.run(tf.global_variables_initializer())
```

```
2 2
array([[[[ 1.,  0.,  0.,  0.]])], dtype=float32)
```

```
hidden_size=2
sequence_length=5
```

Using word vector

```
shape=(1,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]]]
```



```
[ ] with tf.variable_scope('two_sequences') as scope:
    # One cell RNN input_dim (4) -> output_dim (2). sequence: 5
    hidden_size = 2
    cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)
    x_data = np.array([h, e, l, l, o], dtype=np.float32)
```

사용팁

1. 라이브러리 설치

Keras

```
!pip install -q keras
import keras
```

PyTorch

```
from os import path
from wheel.pep425tags import get_abbr_impl, get_impl_ver,
                             get_abi_tag
platform = '{{}}-{}'.format(get_abbr_impl(), get_impl_ver(),
                             get_abi_tag())
accelerator = 'cu80' if path.exists('/opt/bin/nvidia-smi') else
               'cpu'
!pip install -q
http://download.pytorch.org/whl/{accelerator}/torch-0.3.0.post4-{platform}-linux\_x86\_64.whl
_ torchvision
import torch
```

또는:

```
!pip3 install torch torchvision
```

MxNet

```
!apt install libnVRTC8.0
!pip install mxnet-cu80
import mxnet as mx
```

OpenCV

```
!apt-get -qq install -y libsm6 libxext6 && pip install -q -U  
                                opencv-python  
import cv2
```

XGBoost

```
!pip install -q xgboost==0.4a30  
import xgboost
```

GraphViz

```
!apt-get -qq install -y graphviz && pip install -q pydot  
import pydot
```

7zip Reader

```
!apt-get -qq install -y libarchive-dev && pip install -q -U  
                                libarchive  
import libarchive
```

Other Libraries

!pip install 또는 !apt-get install 으로 다른 라이브러리를 사용할 수 있습니다.

2. GPU를 사용중인지 확인하는 방법

CPU만 사용하고 있을 시

```
import tensorflow as tf
tf.test.gpu_device_name()
```

GPU를 사용하고 있을 시

```
import tensorflow as tf
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

3. 사용중인 GPU확인하기

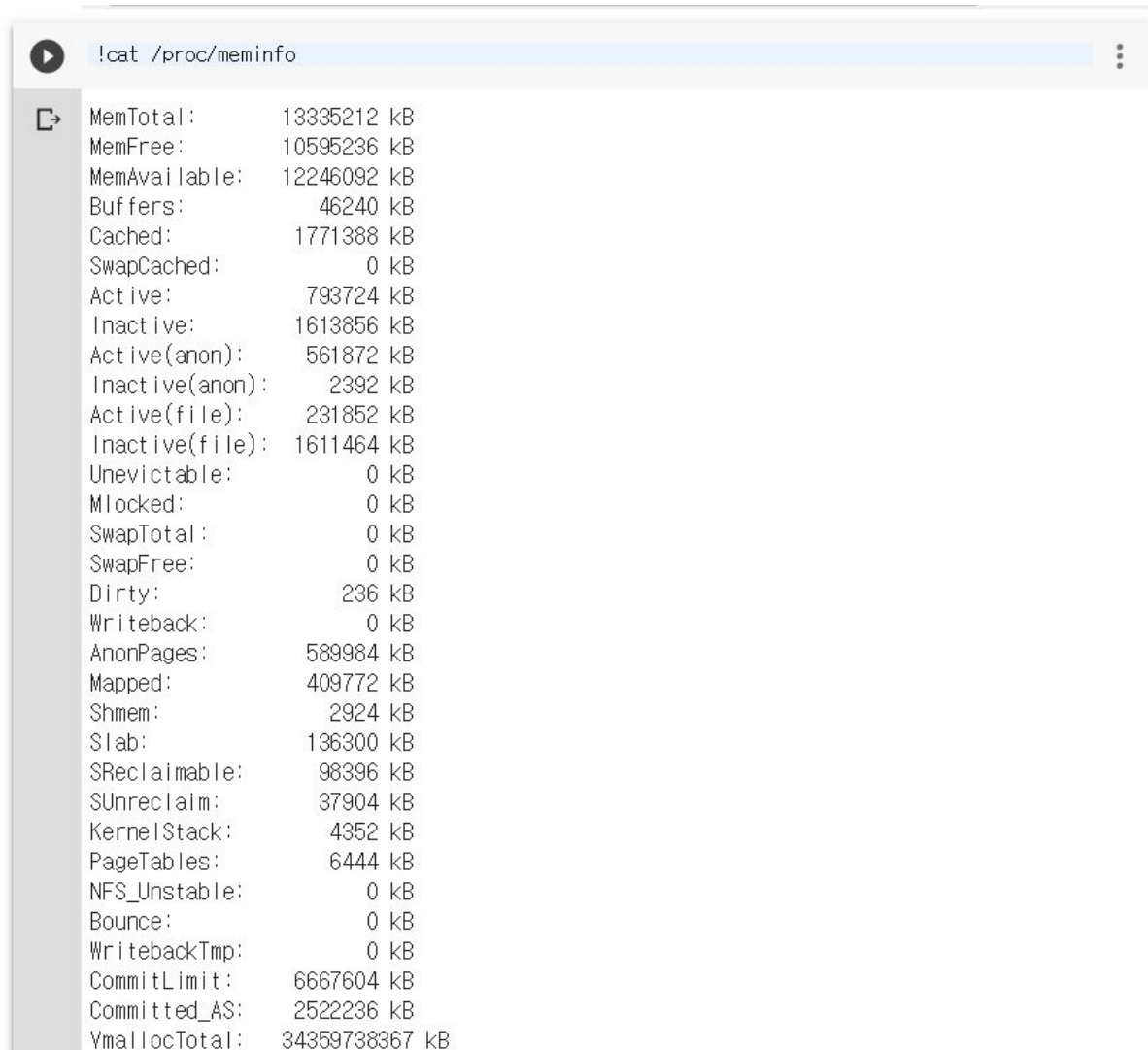
```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 9200129613761559880, name: "/device:XLA_CPU:0"
 device_type: "XLA_CPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 3383000857782022939
 physical_device_desc: "device: XLA_CPU device", name: "/device:XLA_GPU:0"
 device_type: "XLA_GPU"
 memory_limit: 17179869184
 locality {
 }
 incarnation: 16833228284555660350
 physical_device_desc: "device: XLA_GPU device", name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 11281553818
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 14588367858406507542
 physical_device_desc: "device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute ca
```

4. RAM확인

```
!cat /proc/meminfo
```



```
!cat /proc/meminfo
```

MemTotal:	13335212 kB
MemFree:	10595236 kB
MemAvailable:	12246092 kB
Buffers:	46240 kB
Cached:	1771388 kB
SwapCached:	0 kB
Active:	793724 kB
Inactive:	1613856 kB
Active(anon):	561872 kB
Inactive(anon):	2392 kB
Active(file):	231852 kB
Inactive(file):	1611464 kB
Unevictable:	0 kB
Mlocked:	0 kB
SwapTotal:	0 kB
SwapFree:	0 kB
Dirty:	236 kB
Writeback:	0 kB
AnonPages:	589984 kB
Mapped:	409772 kB
Shmem:	2924 kB
Slab:	136300 kB
SReclaimable:	98396 kB
SUnreclaim:	37904 kB
KernelStack:	4352 kB
PageTables:	6444 kB
NFS_Unstable:	0 kB
Bounce:	0 kB
WritebackTmp:	0 kB
CommitLimit:	6667604 kB
Committed_AS:	2522236 kB
VmallocTotal:	34359738367 kB

5. CPU확인

```
!cat /proc/cpuinfo
```

```
!cat /proc/cpuinfo

processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 79
model name     : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping       : 0
microcode      : 0x1
cpu MHz        : 2200.000
cache size     : 56320 KB
physical id    : 0
siblings       : 2
core id        : 0
cpu cores      : 1
apicid         : 0
initial apicid : 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cm
bugs           : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf
bogomips       : 4400.00
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id      : GenuineIntel
cpu family     : 6
model          : 79
model name     : Intel(R) Xeon(R) CPU @ 2.20GHz
```

6. Changing Working Directory

일반적으로 아래 코드를 실행시 :

```
!ls
```

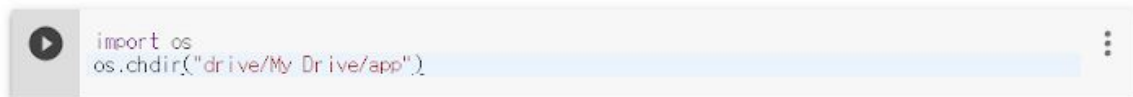
datalab과 drive 폴더들을 볼 수 있습니다.

그러므로 파일이름을 정의하기전에 drive/app을 추가하세요

이런 문제를 없애기 위해, 아래의 코드를 사용하여 쉽게 디렉토리를 바꿀 수 있습니다. (여기선 app 폴더로 이동합니다.):

```
import os
```

```
os.chdir("drive/My Drive/app")
```



위의 코드를 실행시킨 후, 다시 `!ls` 코드를 실행시켜보세요



app폴더안에 있는 파일과 디렉토리의 목록을 보실 수 있습니다. 매번 drive/app을 추가하지 않으셔도 됩니다.

7. “No backend with GPU available“ Error 해결법

```
Failed to assign a backend
```

```
No backend with GPU available. Would you like to use a runtime with no accelerator?
```

다음과 같은 error메세지를 만났을 때, 잠시 후에 다시 사용해 보세요 현재, 많은 사람들이 GPU들을 사용중에 있기 때문에 이런 메세지가 납니다.

Reference

8. 모든 셀들의 출력값을 지우기

도구>>명령 팔레트 >> 모든 출력 지우기

9. “apt-key output should not be parsed (stdout is not a terminal)” 경고문

```
Warning: apt-key output should not be parsed (stdout is not a terminal)
```

다음과 같은 경고 메세지를 만났을 때, 권한부여는 완료된겁니다. 이제 구글드라이브를 마운트하세요

```
!mkdir -p drive
```

```
!google-drive-ocamlfuse drive
```

10. 구글 Colab을 Tensorboard를 사용하는 방법

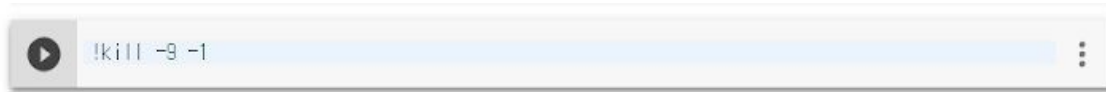
다음 repository를 참고하세요:

https://github.com/mixuala/colab_utils

11. Google Colab을 다시 시작하는 방법

가상머신에서 재시작(또는 초기화) 하는 방법:

!kill -9 -1

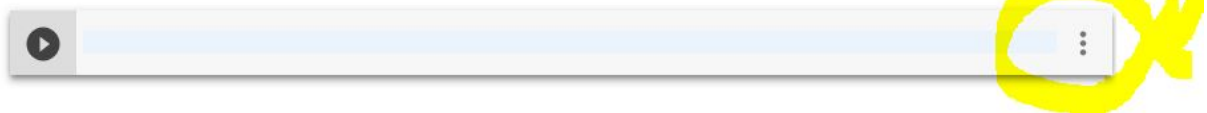


12. Form을 추가하여 사용하기

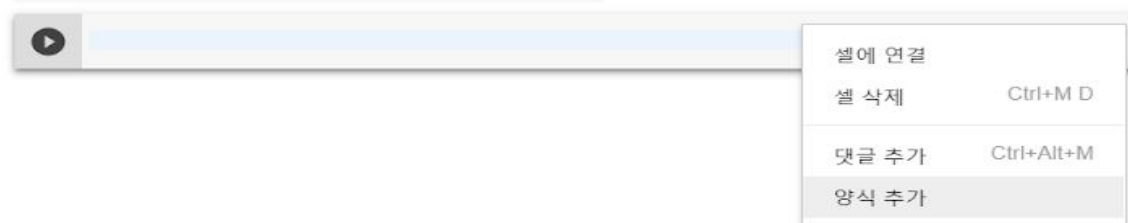
코드에서 매번 hyperparameters을 변경하지 않으려, Google Colab에 간단하게 form을 추가하여 사용하세요

예를 들어, learning_rate와 optimizer 변수를 포함하는 form을 추가하겠습니다.

1. 셀에서 더보기 버튼 클릭



2. 양식 추가



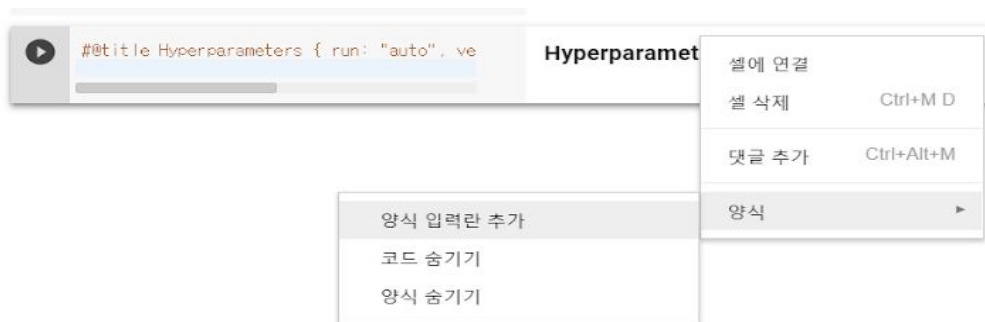
3. 수정 버튼 클릭



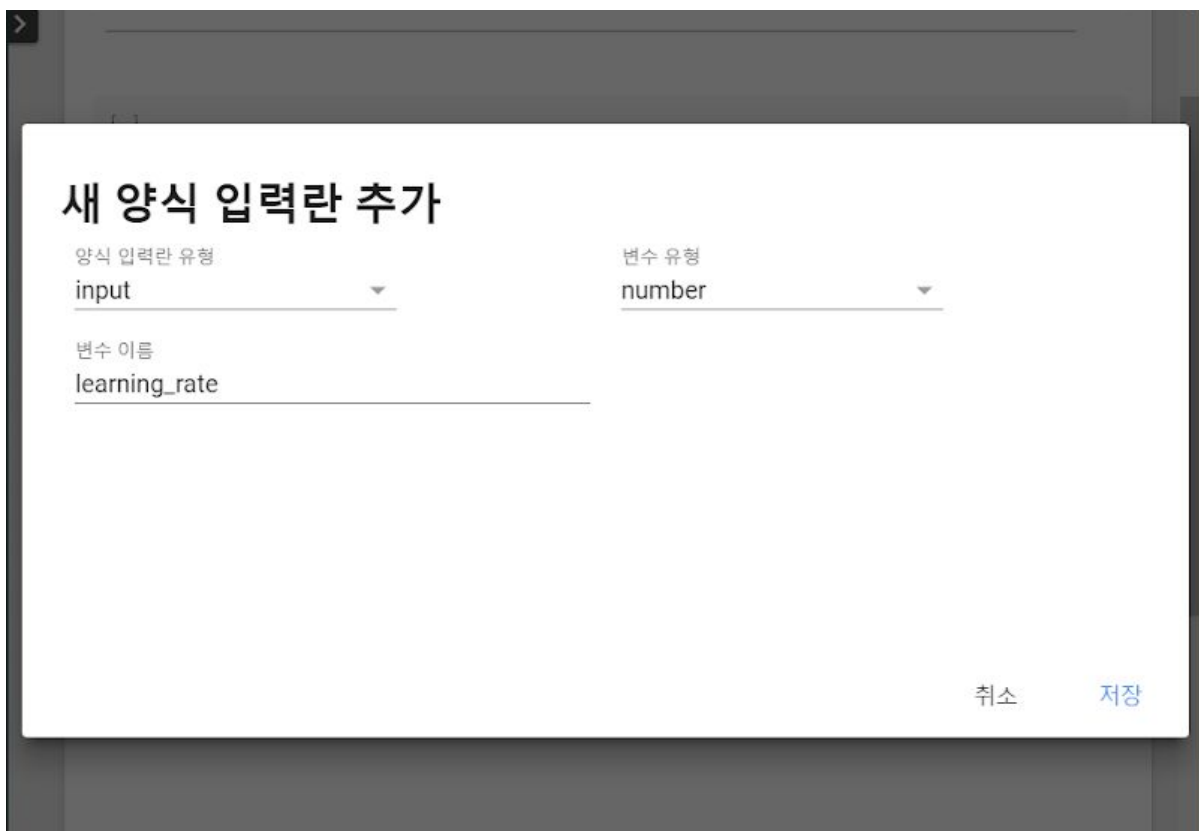
4. 제목 지정 후 저장버튼 클릭



5. 더보기 버튼>양식> 양식 입력란 추가



6. 양식 유형과, 변수 유형, 변수이름 설정 후 저장



결과:



The image shows a configuration interface for hyperparameters. On the left, a code editor contains the following text: `#@title Hyperparameters { run: "auto", ve`, `learning_rate = 0.0001 #@param {type: "num`, and `optimizer = "Adam" #@param {type: "string"`. On the right, a panel titled "Hyperparameters" displays the configured values: "learning_rate: 0.0001" and "optimizer: Adam". Each value has an edit icon (pencil) to its right.

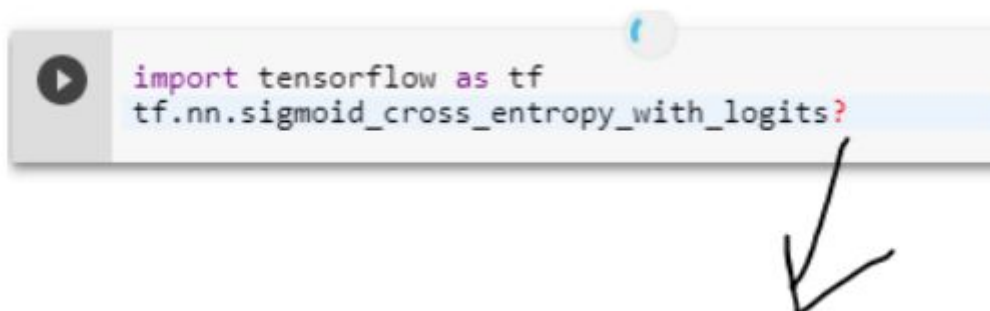
변수 설정 확인하기:



The image shows a variable setting confirmation interface. It features a search bar with the text "optimizer" and a dropdown menu below it showing the selected value, "'Adam'".

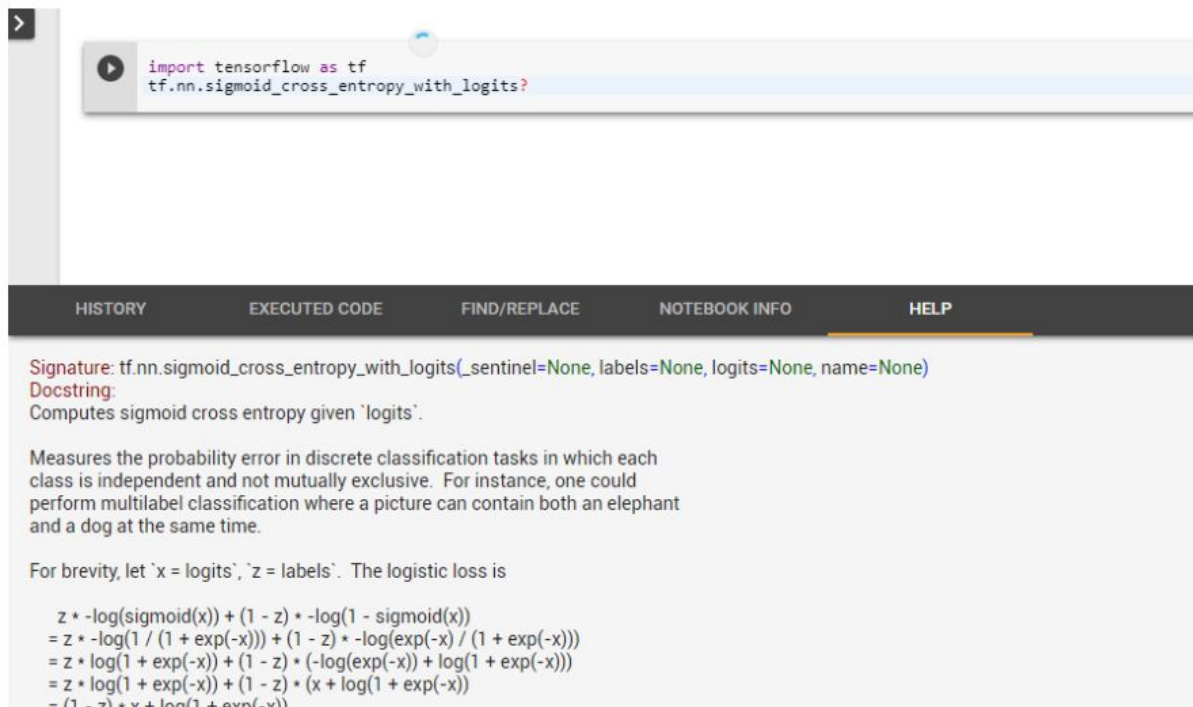
13. 함수 인수를 보는 방법

TensorFlow, Keras 등의 함수 인수를 보려면 함수 이름 뒤에 물음표 (?)를 추가하면됩니다.



The image shows a code editor with the following code: `import tensorflow as tf` and `tf.nn.sigmoid_cross_entropy_with_logits?`. A hand-drawn arrow points from the question mark at the end of the second line to the text below.

이제 TensorFlow 웹 사이트를 클릭하지 않고도 원본 문서를 볼 수 있습니다.



14. 구글Colab에서 Tensorboard를 사용하는 방법

You can change the directory name

LOG_DIR = 'tb_logs'

!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip

!unzip ngrok-stable-linux-amd64.zip

import os

if not os.path.exists(LOG_DIR):
 os.makedirs(LOG_DIR)

get_ipython().system_raw(
 'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &
 .format(LOG_DIR))

get_ipython().system_raw('./ngrok http 6006 &')

!curl -s http://localhost:4040/api/tunnels | python3 -c \
 "import sys, json; print(json.load(sys.stdin))['tunnels'][0]['public_url']"

위 코드를 실행시키고 URL을 여세요

```
# You can change the directory name
LOG_DIR = 'tb_logs'

!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
!unzip ngrok-stable-linux-amd64.zip

import os
if not os.path.exists(LOG_DIR):
    os.makedirs(LOG_DIR)

get_ipython().system_raw(
    'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
    .format(LOG_DIR))

get_ipython().system_raw('./ngrok http 6006 &')

!curl -s http://localhost:4040/api/tunnels | python3 -c '#
import sys, json; print(json.load(sys.stdin)[0][0][\"public_url\"])"
```

—2018-12-09 14:07:40— <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip>
Resolving bin.equinox.io (bin.equinox.io)... 34.206.9.96, 34.206.253.53, 34.226.180.131, ...
Connecting to bin.equinox.io (bin.equinox.io)[34.206.9.96]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5363700 (5.1M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.zip.1'

ngrok-stable-linux- 100%[=====>] 5.11M 3.54MB/s in 1.4s

2018-12-09 14:07:42 (3.54 MB/s) - 'ngrok-stable-linux-amd64.zip.1' saved [5363700/5363700]

Archive: ngrok-stable-linux-amd64.zip
replace ngrok? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: ngrok
<http://28a44891.ngrok.io>

텐서보드 실행을 위해 실행 예제코드를 실행시켜보세요

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.callbacks import TensorBoard
```

```
batch_size = 128
num_classes = 10
epochs = 12
```

```
# input image dimensions
img_rows, img_cols = 28, 28
```

```

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

```

```
model.add(Dense(num_classes, activation='softmax'))
```

```
model.compile(loss=keras.losses.categorical_crossentropy,  
              optimizer=keras.optimizers.Adadelta(),  
              metrics=['accuracy'])
```

```
tbCallBack = TensorBoard(log_dir=LOG_DIR,  
                          histogram_freq=1,  
                          write_graph=True,  
                          write_grads=True,  
                          batch_size=batch_size,  
                          write_images=True)
```

```
model.fit(x_train, y_train,  
         batch_size=batch_size,  
         epochs=epochs,  
         verbose=1,  
         validation_data=(x_test, y_test),  
         callbacks=[tbCallBack])  
score = model.evaluate(x_test, y_test, verbose=0)  
print('Test loss:', score[0])  
print('Test accuracy:', score[1])
```

```

img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

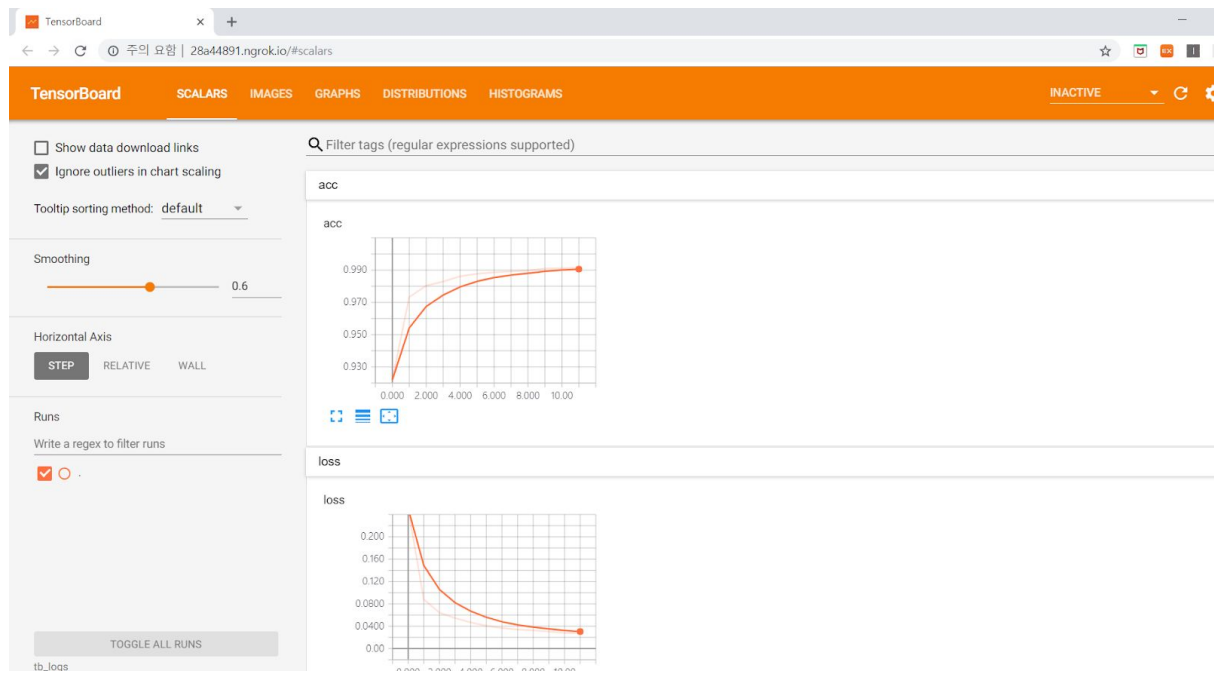
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

tbCallback = TensorBoard(log_dir=LOG_DIR,
                          histogram_freq=1,
                          write_graph=True,
                          write_grads=True,
                          batch_size=batch_size,
                          write_images=True)

model.fit(x_train, y_train,
          batch_size=batch_size,

```

위에서 말한 URL에 들어가면 TensorBoard를 확인하실 수 있습니다!



참고문헌

<https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d>