Computer Vision

Deep Learning Basics
(#18: Google Colab-based deep learning environment setup)

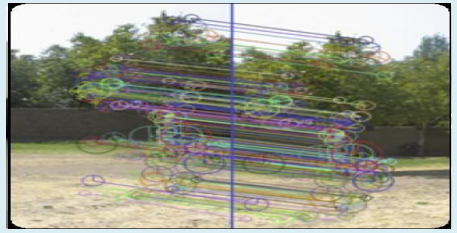
2023 Autumn

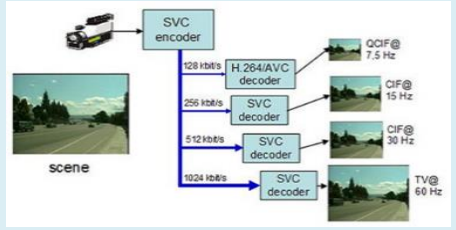
Prof. Byung-Gyu Kim
Intelligent Vision Processing Lab. (IVPL)
http://ivpl.sookmyung.ac.kr
Dept. of IT Engineering, Sookmyung Women's University
E-mail: bg.kim@sookmyung.ac.kr

Gaol of this lecture

❖ How to set-up the deep learning development using **Google Colab**?

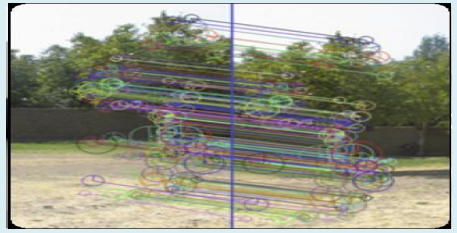
- What is **Google Colab**?
- How to use it and develop the deep learning system?
- Basic configuration of **Google Colab**

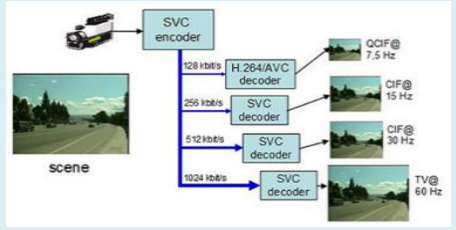




Contents

- What is the **Google Colab**?
- Basic configuration of **Google Colab**





Contents

- What is the **Google Colab**?
- Basic configuration of **Google Colab**

Google Colab: What is it? (1)

❖ Deep learning requirements

- **Big data (images)**
- **Huge time to compute and train the CNN**
- **Parallel processing → GPU is necessary because of time**

GPU system is always needed to compute the desired algorithms efficiently.

Google Colab: What is it? (2)

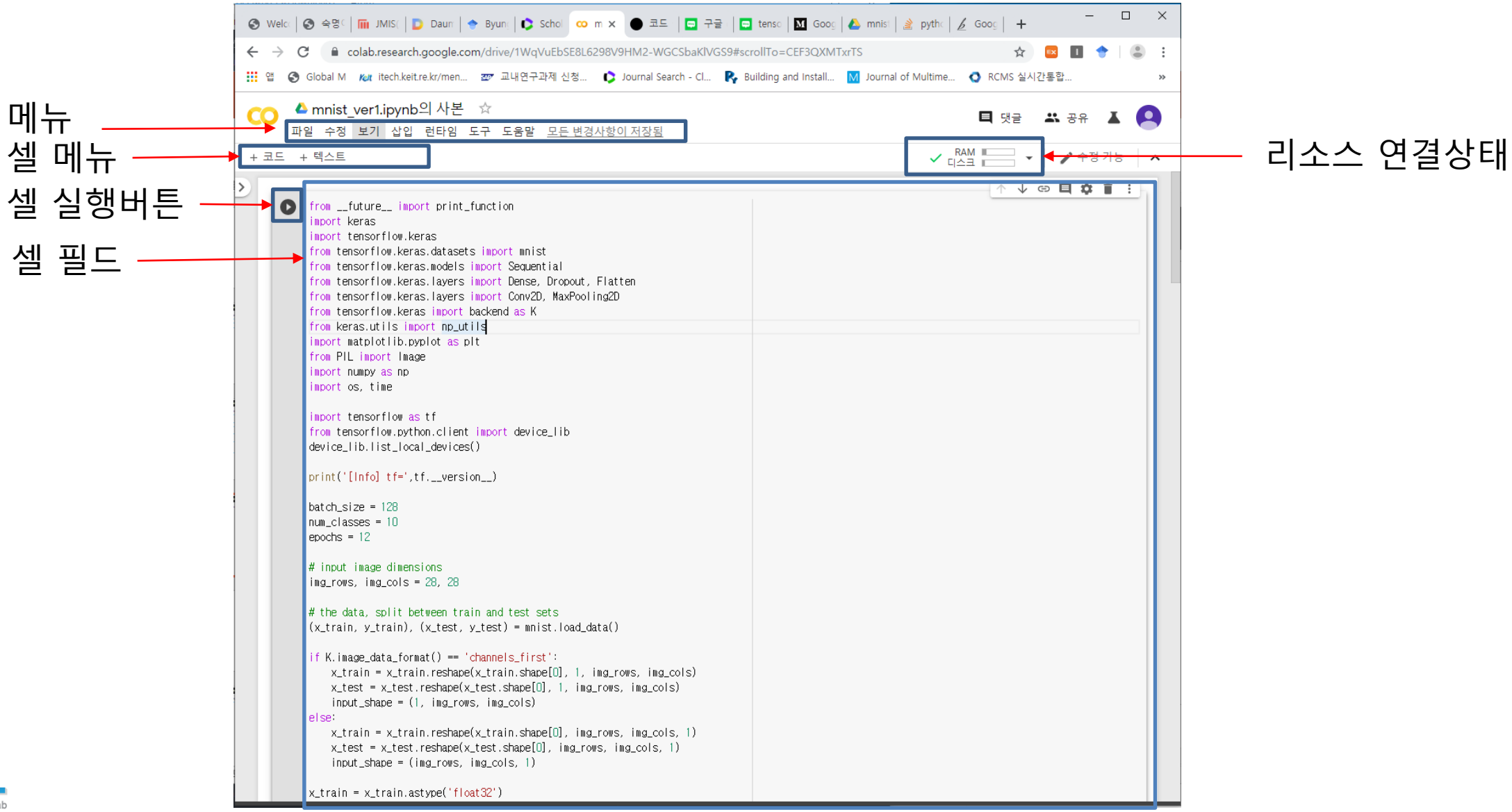
❖ Google Colaboratory

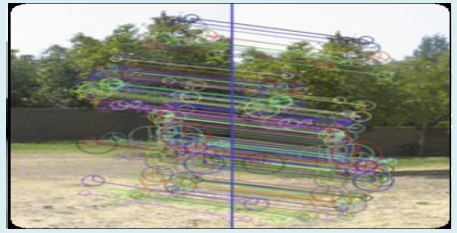
- **Google 내부에서 사용하던 jupyter Notebook을 교육과 연구 목적으로 customize한 데이터 분석 도구**
- **특히 machine learning 교육 및 연구용 도구로 open된 클라우드 기반의 서비스**
- **이미 Python2.x와 3.x 버전이 설치되어 있고 GPU 클라우드 기반의 GPU 병렬 처리를 제공하여 Google 계정만 있으면 기본 GPU 서비스 기반 병렬 처리를 지원함**
- **웹브라우저 기반으로 docker 환경에서 google GPU 서버에 접속하여 서비스가 지원되므로 기본적으로 Google chrom 브라우저를 권장함**
- **Colaboratory 실행 코드는 google 계정 전용의 가상 머신에서 동작하므로 세션이 끊으지거나 유휴 상태가 오래 지속되면 리소스가 자동 재할당 되므로 본인의 데이터가 메모리에서 제거됨**
- **따라서 Google Drive (내 드라이브)를 연결하여 주로 본인의 개발 소스를 관리하는 체계를 지원함**

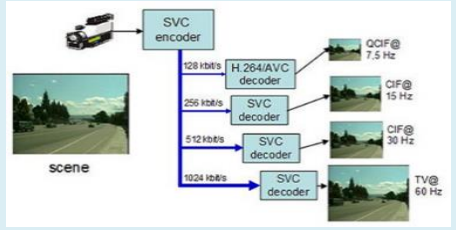
Google Colaboratory = Google Drive + Juputer Notebook

Google Colab: What is it? (3)

❖ Google Colaboratory: **Basic UI**







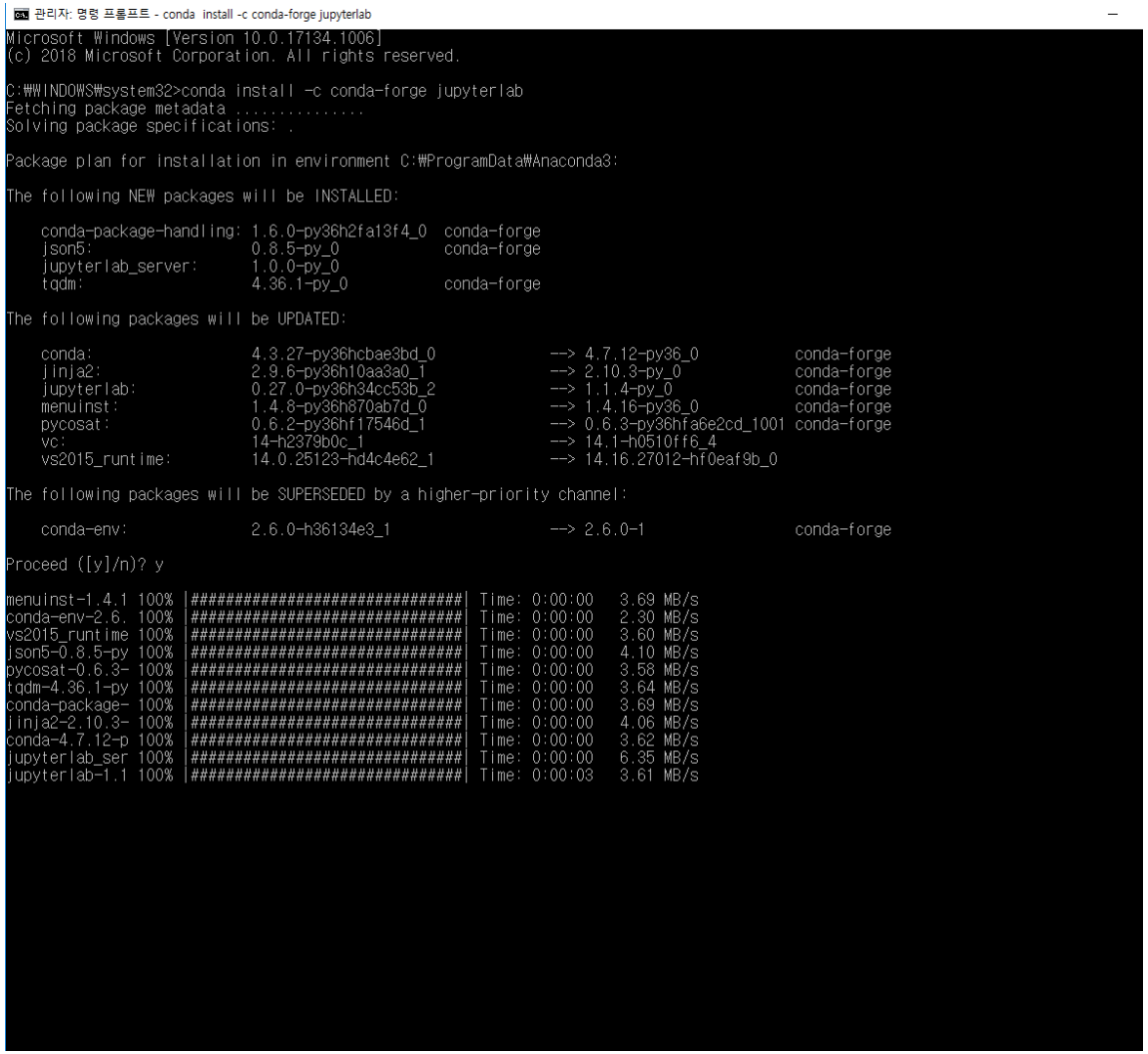
Contents

- What is the **Google Colab**?
- Basic configuration of **Google Colab**

Google Colab 활용하기: 기본 설정 및 동작성 확인

❖ Jupyter 노트 또는 Jupyterab 설치 (anaconda 설치된 상태)

- 1) conda install -c conda-forge jupyterlab (CMD 창에서)

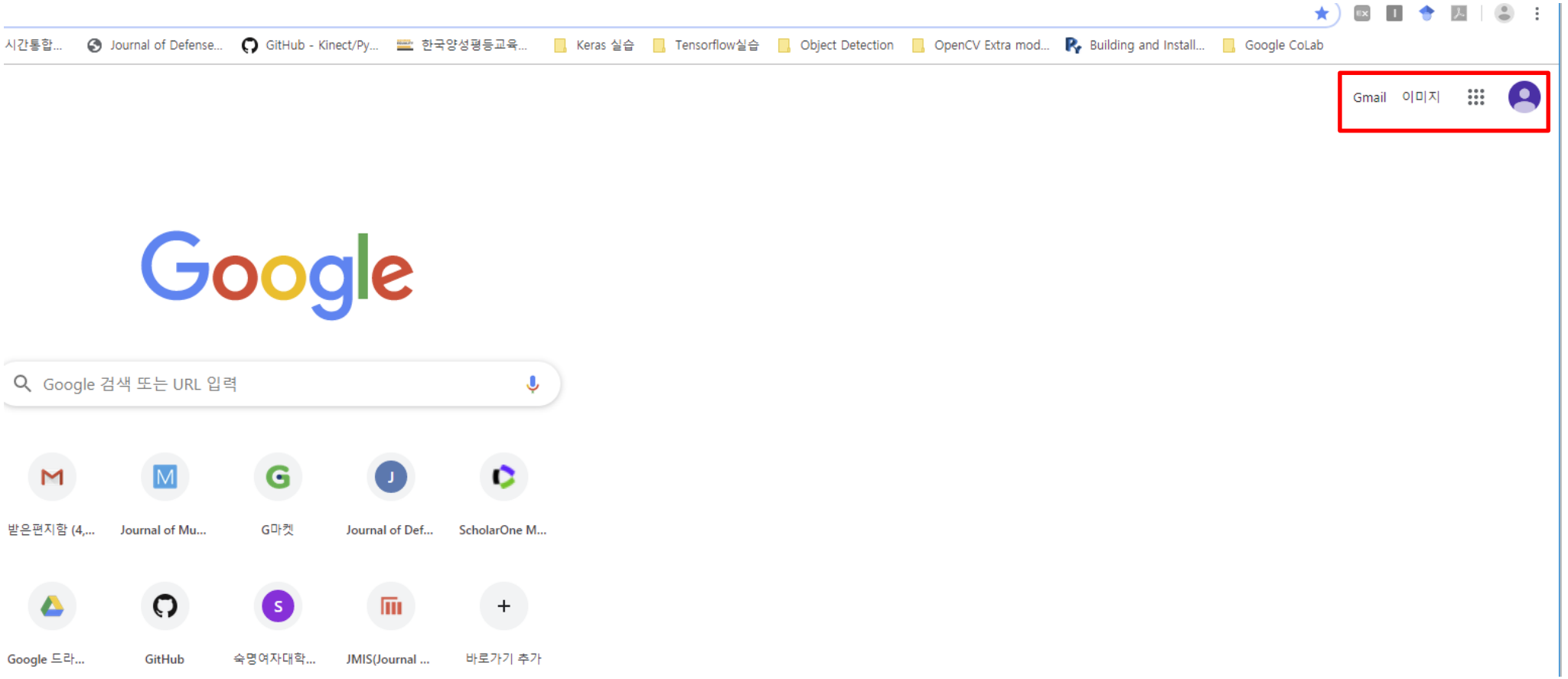


Google Colab 활용하기 : 기본 설정 및 동작성 확인

- 2) Anaconda (최신 버전 설치): jupyter notebook 자동으로 설치됨

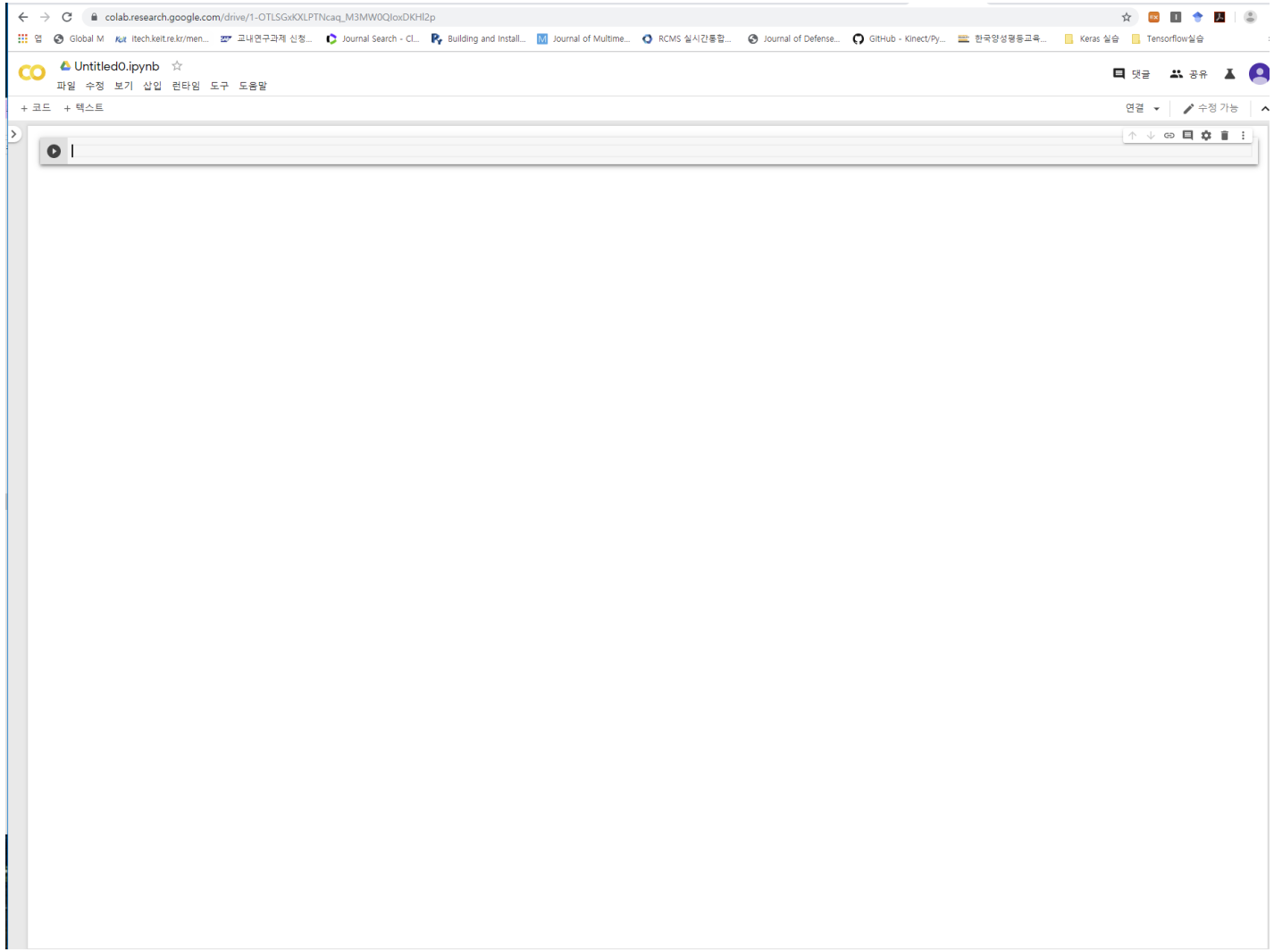
Google Colab 활용하기 : 기본 설정 및 동작성 확인

- Google에 로그인 하기 (상태 확인)



Google Colab 활용하기 : 기본 설정 및 동작성 확인

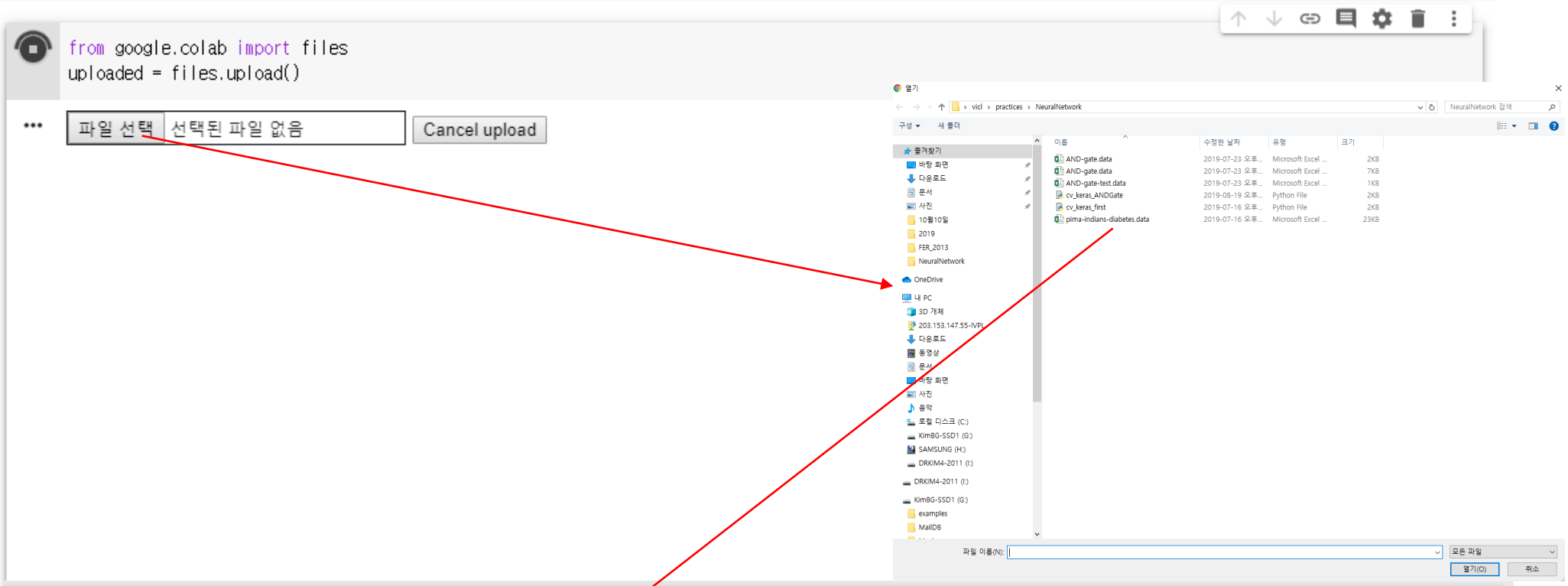
❖ https://colab.research.google.com/ (실제 colab 을 시작하기)

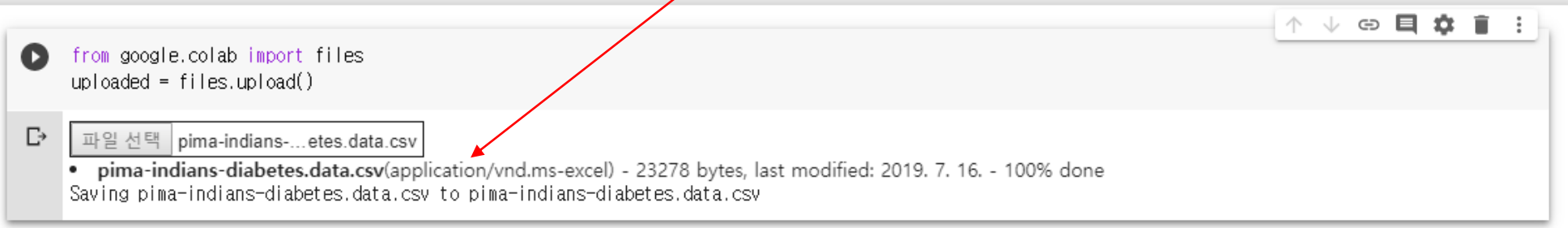


Google Colab 활용하기 : 기본 설정 및 동작성 확인

❖ Jupyter book에서 간단한 예제 실행 (python 기반)

- 오류 발생은 “pop-up” 창 disable되어 있는 것 “허용”으로 해 주면 아래와 같이 동작함

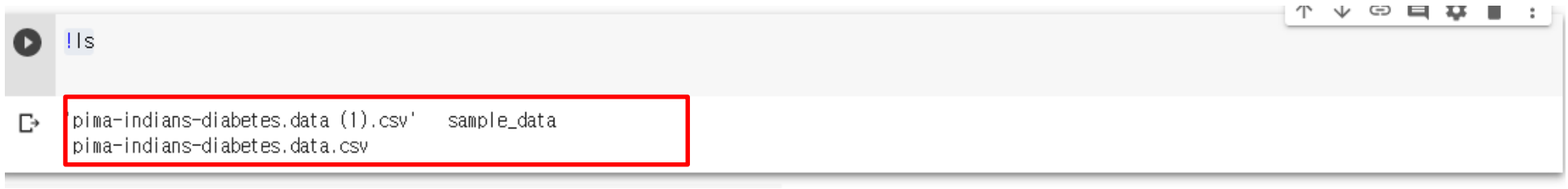




Google Colab 활용하기 : 기본 설정 및 동작성 확인

❖ Upload된 데이터 로딩하여 실행확인해 보기

- 명령어 필드에서 "!ls" 명령어 실행: 업로드한 파일 보이죠???



❖ 실제 저장된 데이터 python 프로그램으로 출력해 보기

```
import numpy as np
dataset = np.loadtxt("pima-indians-diabetes.data.csv", delimiter=",")
print(dataset)
```

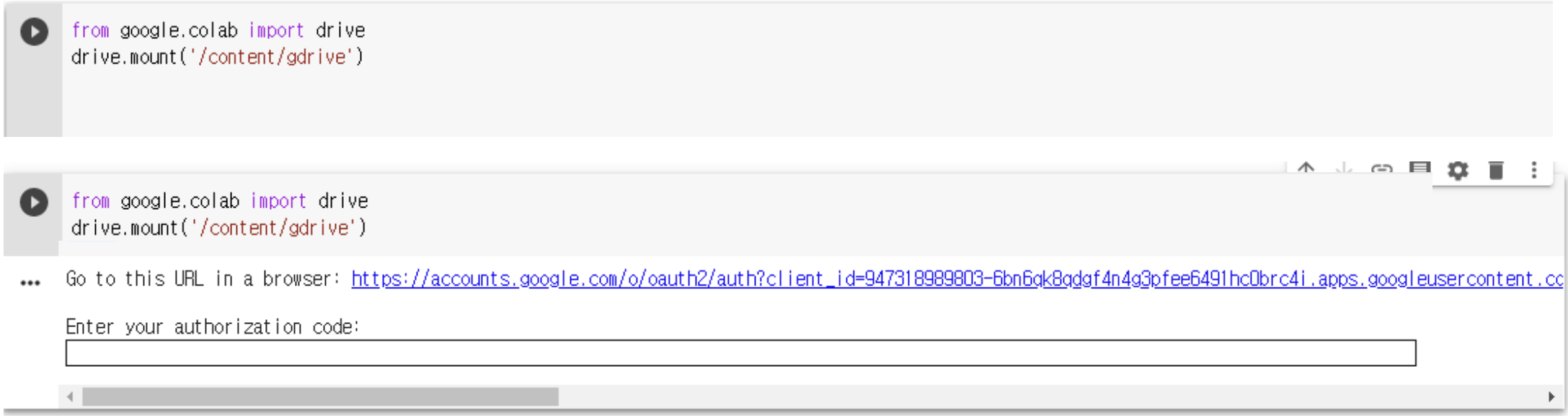
Google Colab 활용하기 : GoogleDrive 연결하기

❖ 프로젝트 코드 및 데이터 저장소 만들기

- Colab 클라우드에 파일을 업로드하는 방식: 파일이 삭제되면 다시 업로드를 해야 함
- GoogleDrive(개인): 특정 파일을 계속 사용할 경우 구글 드라이브에 파일을 업로드 한 후에 계속 사용하는 방식이 바람직함

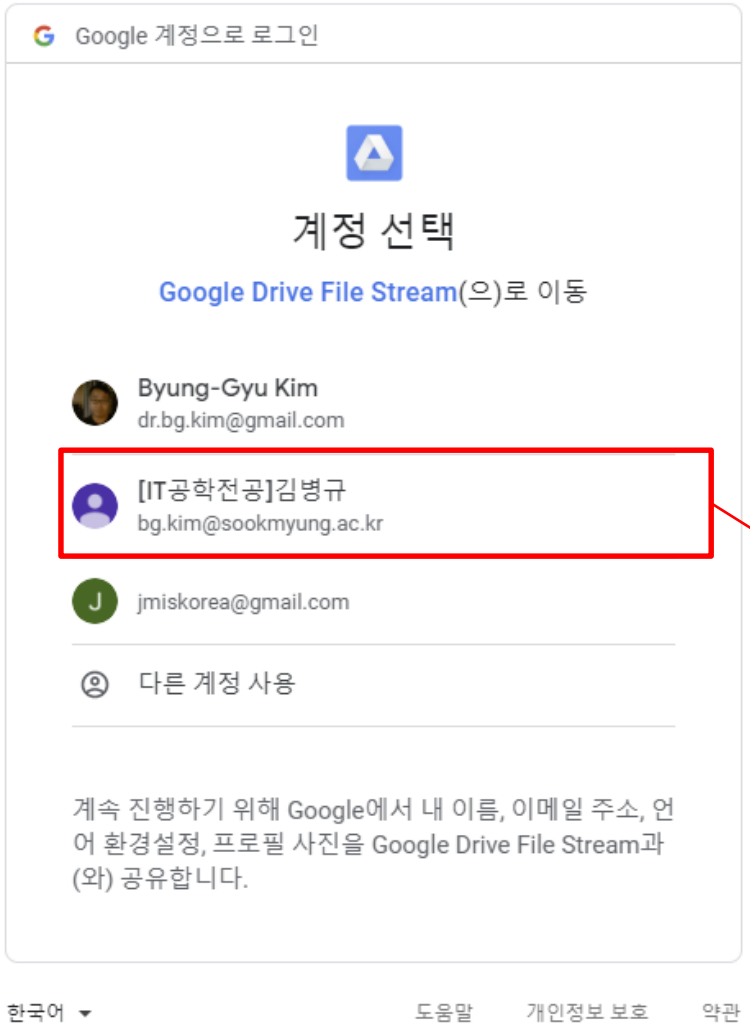
- 1) 아래 명령어 수행

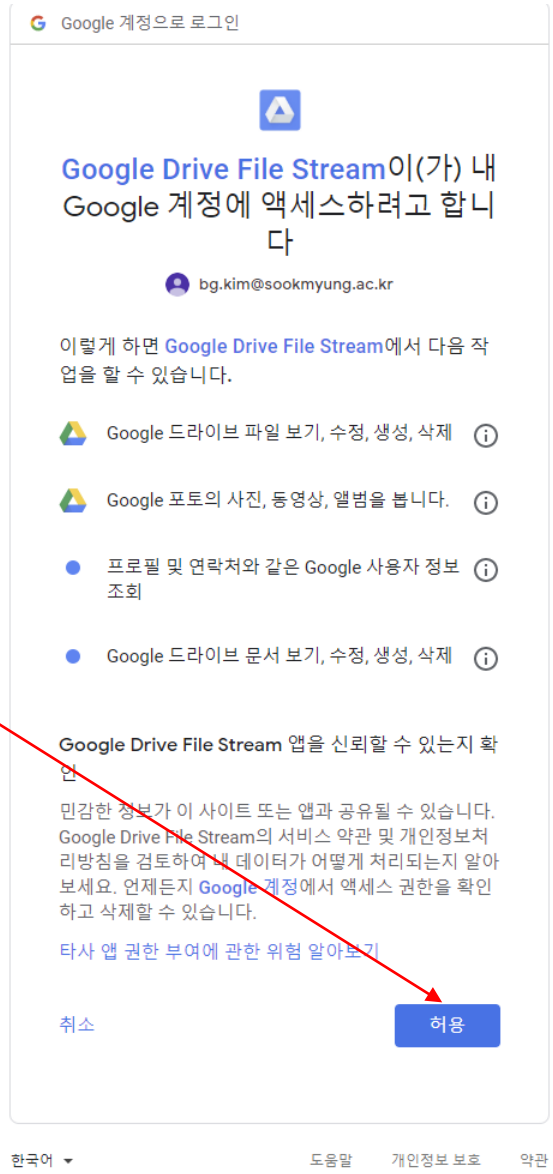
```
from google.colab import drive
drive.mount('/content/gdrive')
```

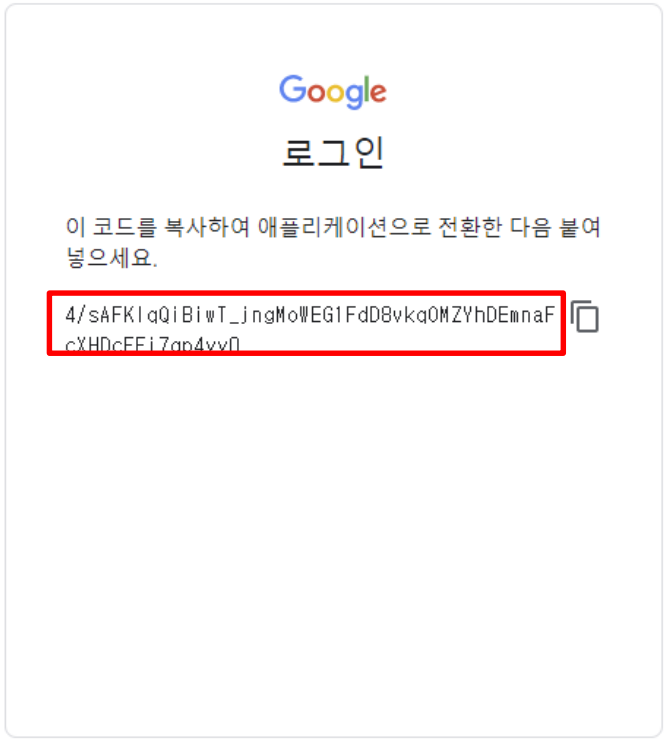


Google Colab 활용하기 : GoogleDrive 연결하기

- 2) 계정 선택화면으로

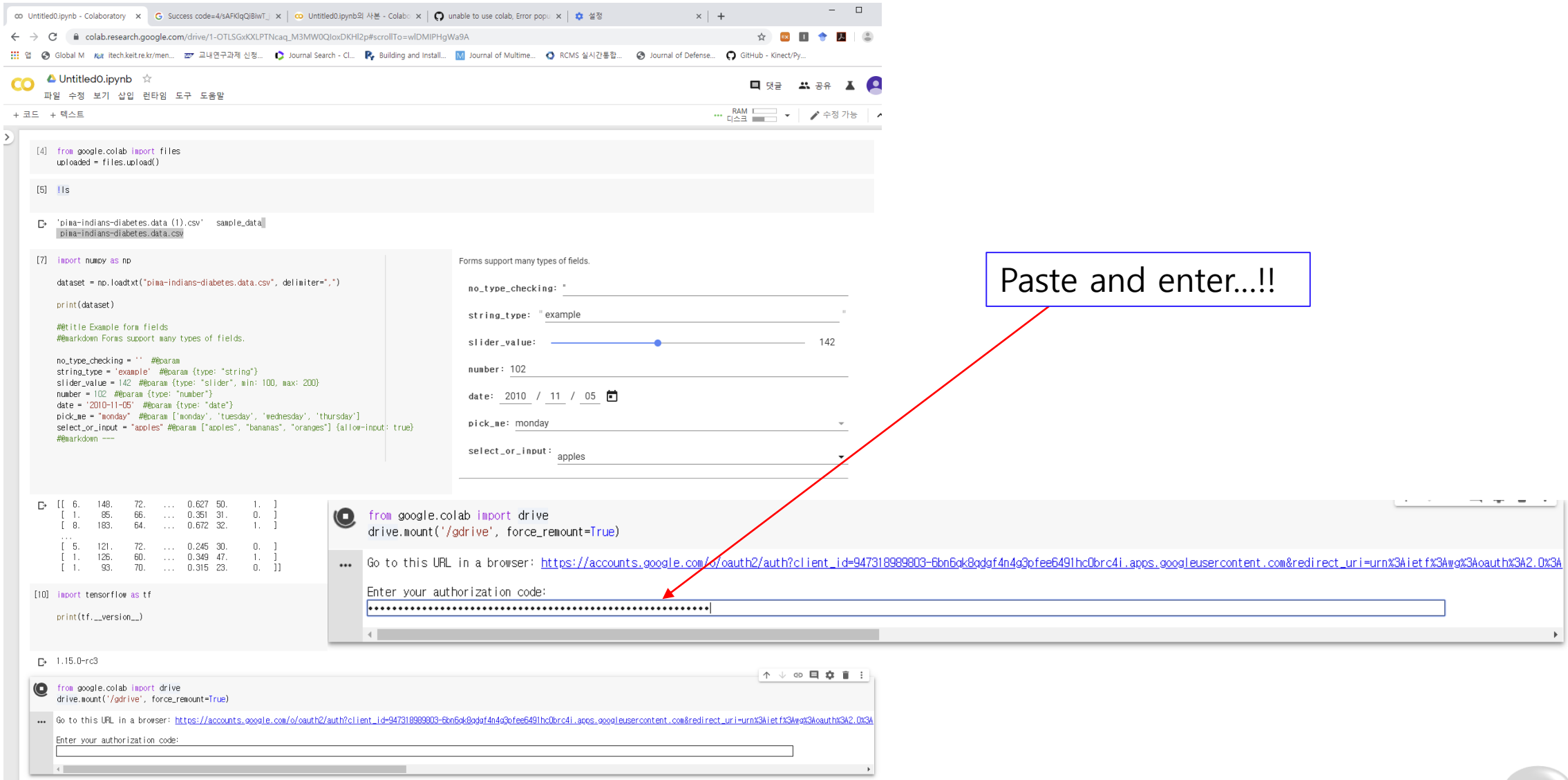






Google Colab 활용하기 : GoogleDrive 연결하기

- 3) 원래 jupyter notebook으로 돌아 온후 복사해 준다.



Google Colab 활용하기 : GoogleDrive 연결하기

- 4) "gdrive"라는 폴더가 생성되어 mount 된 것을 볼 수 있다.

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf

Enter your authorization code:
Mounted at /content/gdrive
```

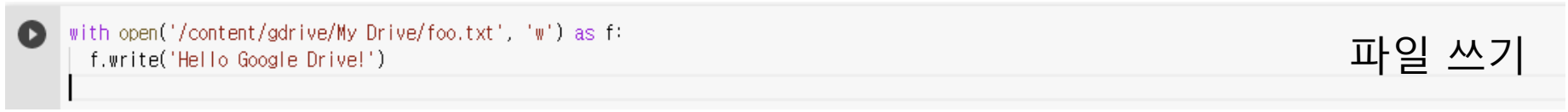
```
!cd /gdrive/
!ls -al
```

```
total 16
drwxr-xr-x 1 root root 4096 Aug 27 16:17 .
drwxr-xr-x 1 root root 4096 Oct 16 10:12 ..
drwxr-xr-x 1 root root 4096 Oct  8 20:06 .config
drwxr-xr-x 1 root root 4096 Aug 27 16:17 sample_data
```

Google Colab 활용하기 : GoogleDrive 연결하기

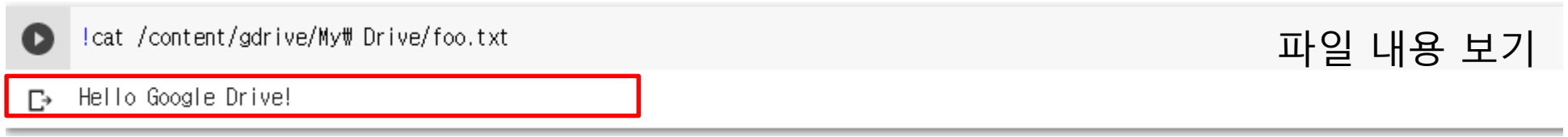
- 5) "gdrive"→"My drive"에 있는 파일 쓰기/접근하기

```
with open('/content/gdrive/My Drive/foo.txt', 'w') as f:
    f.write('Hello Google Drive!')
```



파일 쓰기

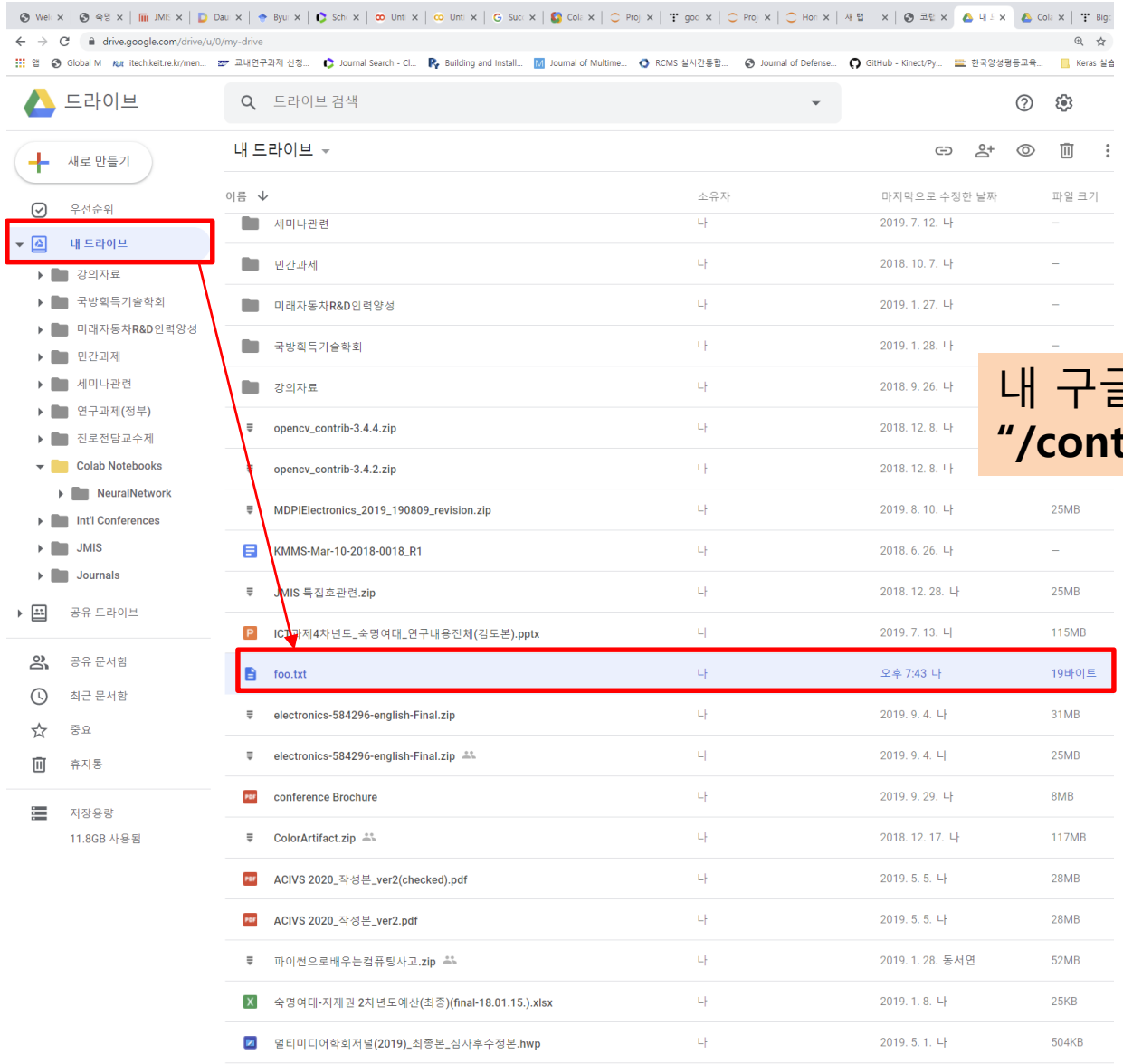
```
!cat /content/gdrive/My\ Drive/foo.txt
```



파일 내용 보기

Google Colab 활용하기 : GoogleDrive 연결하기

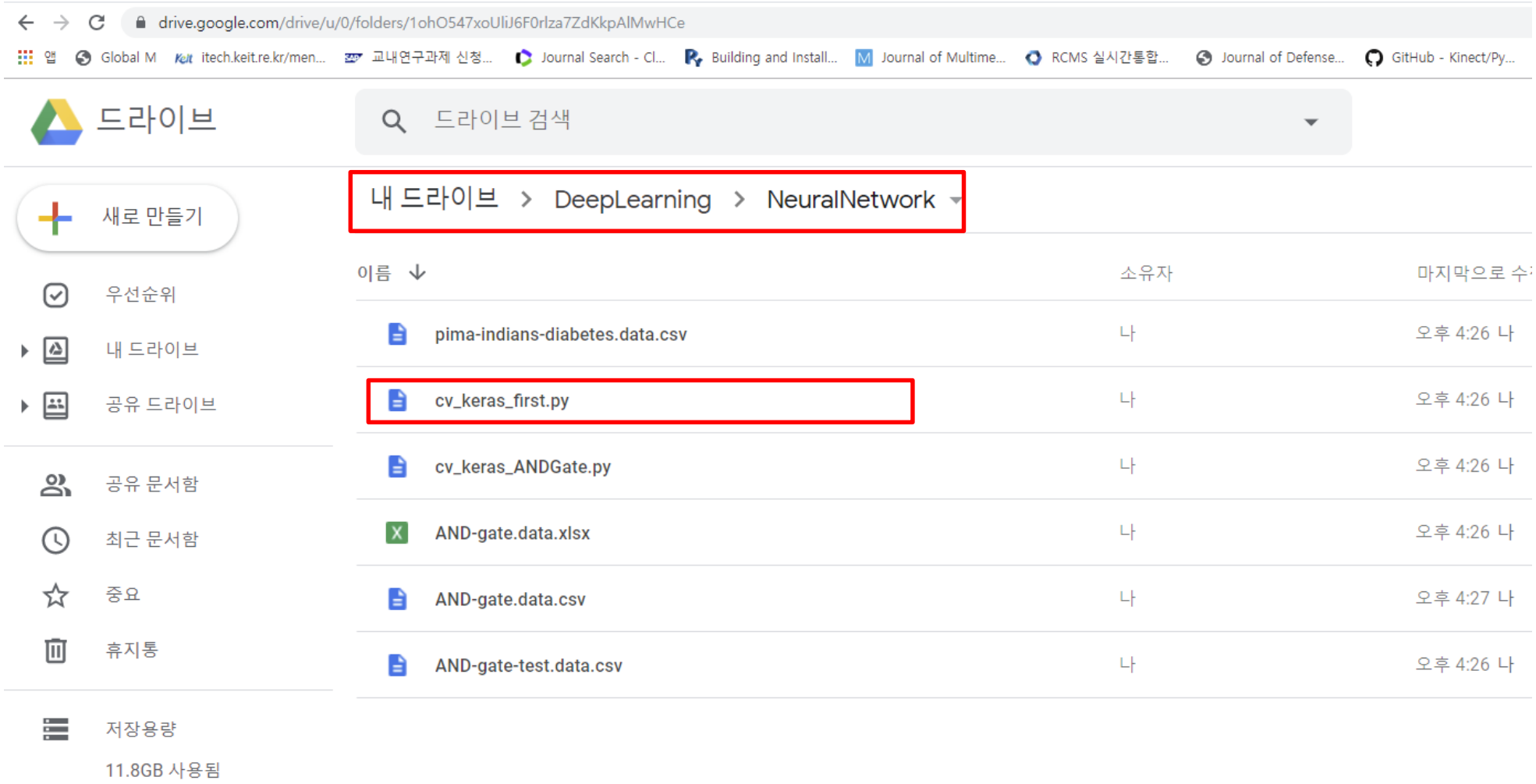
❖ 실제 google drive 어디에 foo.txt가 만들어졌는지 가볼까요??



내 구글 드라이브까지 표준 경로:
"/content/gdrive/My Drive/"

Google Colab 활용하기 : GoogleDrive 연결하기

❖ Google drive→"내드라이브" → 임의의 폴더 내 파일 읽어 보기



Google Colab 활용하기 : GoogleDrive 연결하기

- Jupyter Notebook 에서

```
!cat /content/gdrive/My\ Drive/DeepLearning/NeuralNetwork/cv_keras_first.py
```

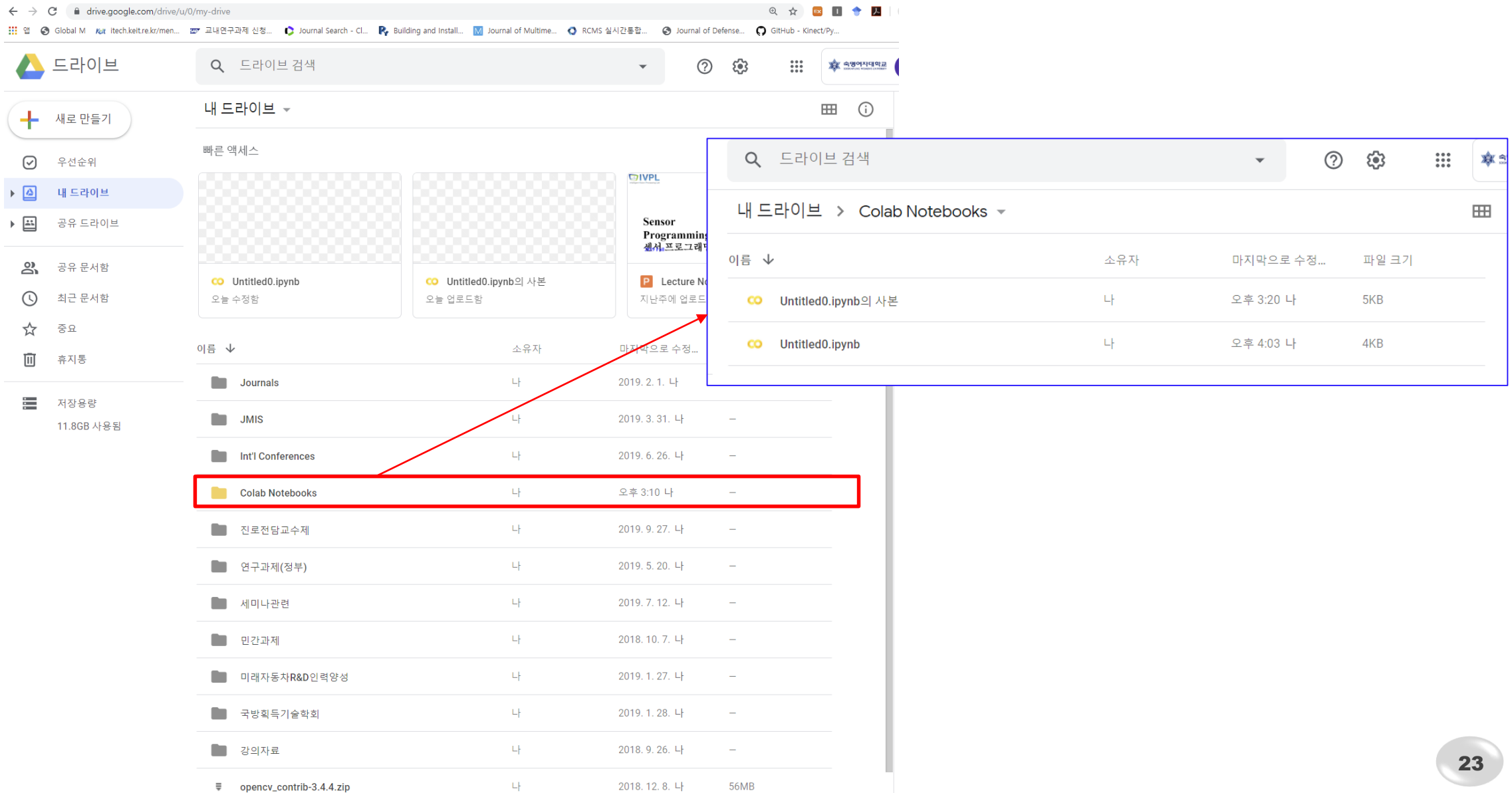
```
!cat /content/gdrive/My\ Drive/DeepLearning/NeuralNetwork/cv_keras_first.py

## Visualize training history
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy
## fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
## load pima indians dataset
dataset = numpy.loadtxt("pima-indians-diabetes.data.csv", delimiter=",")
## split into input (X) and output (Y) variables
X = dataset[:,0:8]
Y = dataset[:,8]
#print(X)
#print(Y)
## create model
# 선형적으로 차원을 쌓아 모델을 만듦
model = Sequential()
# input layer
model.add(Dense(12, input_dim=8, kernel_initializer='uniform', activation='relu'))
# hidden layer
model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
# output layer
model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
## Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
## Fit the model
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10, verbose=0)
## list all data in history
#print(history.history['acc'])
#print(history.history['loss'])
#print(history.history['val_acc'])
#print(history.history['val_loss'])
## summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

실제 파일을 접근하여 잘 읽어올 수 있음을 알 수 있다.

즉 여러분들이 소스나 실습 코드 구글 드라이브에 올리고 파일이나 데이터에 대한 접근이 가능하다는 것을 확인함

Google Colab 활용하기 : GoogleDrive 연결하기

- Google drive 확인: Colab Notebooks 폴더 확인(본인이 작업하는 작업의 임시 저장소)



Google Colab 활용하기 : Keras 기반 python code testing (1)

❖ Test code 수행(keras 기반)

```
## Visualize training history
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy
## fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
## load pima indians dataset
dataset = numpy.loadtxt("pima-indians-
diabetes.data.csv", delimiter=",")
## split into input (X) and output (Y)
variables
X = dataset[:,0:8]
Y = dataset[:,8]
#print(X)
#print(Y)
## create model
```

(계 속)

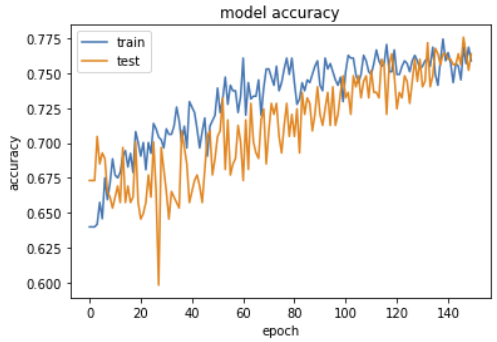
```
## create model
# 선형적으로 차원을 쌓아 모델을 만듦
model = Sequential()
# input layer
model.add(Dense(12, input_dim=8,
kernel_initializer='uniform',
activation='relu'))
# hidden layer
model.add(Dense(8,
kernel_initializer='uniform',
activation='relu'))
# output layer
model.add(Dense(1,
kernel_initializer='uniform',
activation='sigmoid'))
## Compile model
model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
## Fit the model
history = model.fit(X, Y,
validation_split=0.33, epochs=150,
batch_size=10, verbose=0)
```

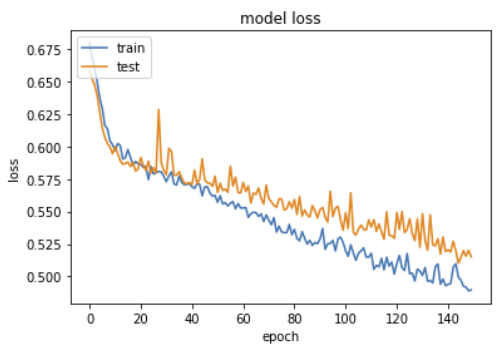
Google Colab 활용하기 : Keras 기반 python code testing (2)

(계 속)

```
## list all data in history
## summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper
left')
plt.show()
## summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper
left')
plt.show()
```

```
W1016 07:28:59.075867 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.093713 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.100692 139888131606400 deprecation.py:323] From /usr/local/lib/python2.7/dist-packages
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W1016 07:28:59.296449 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.368737 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.428307 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.437418 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.438689 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.559381 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.561050 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa

W1016 07:28:59.747308 139888131606400 module_wrapper.py:139] From /usr/local/lib/python2.7/dist-packa
```

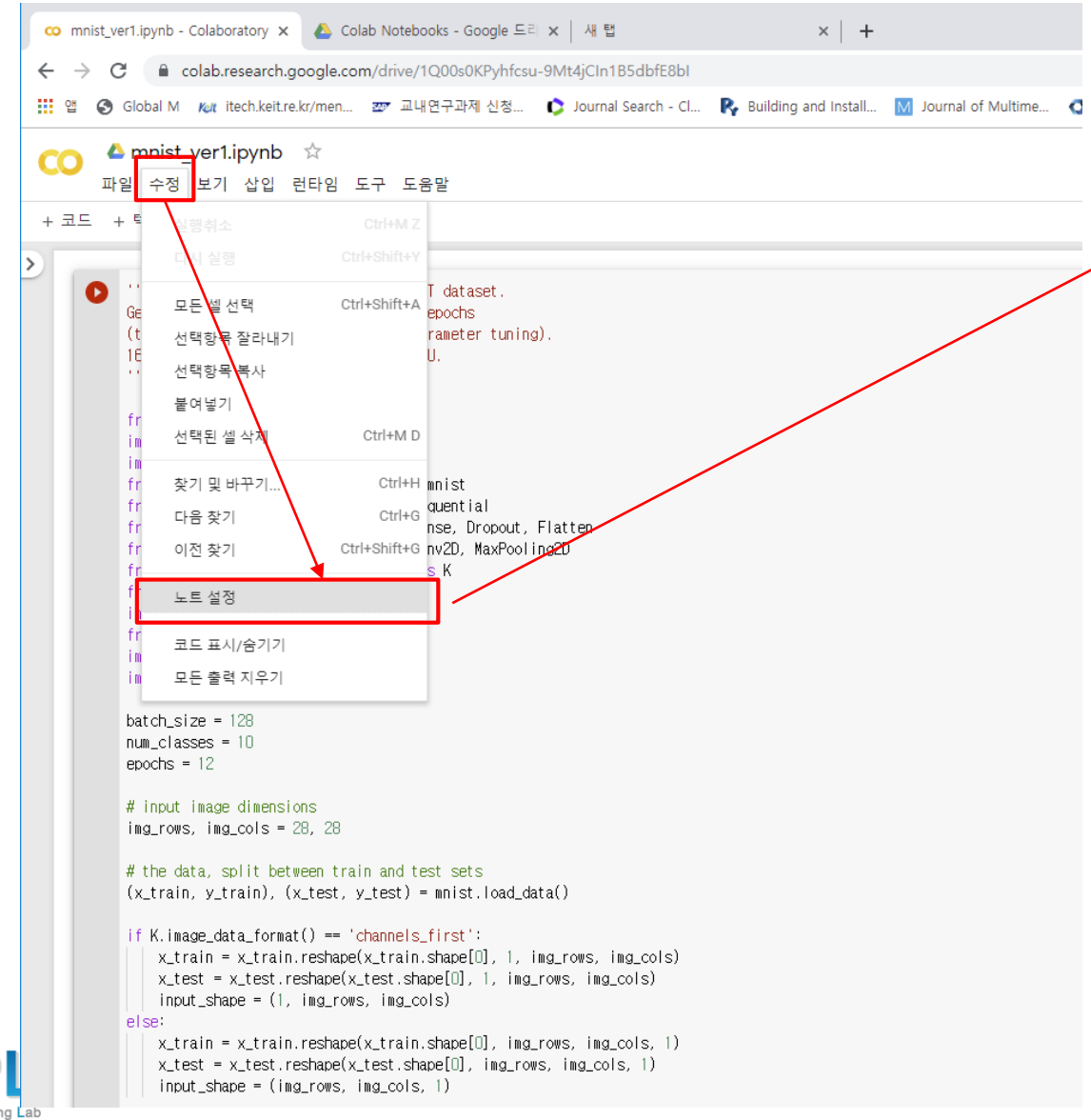


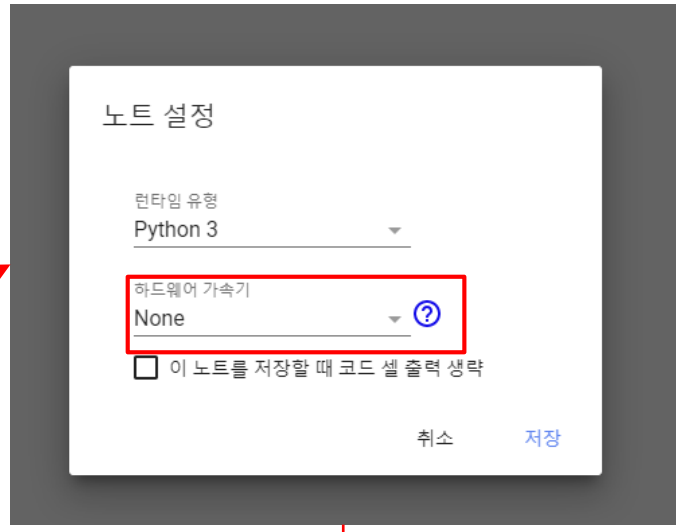


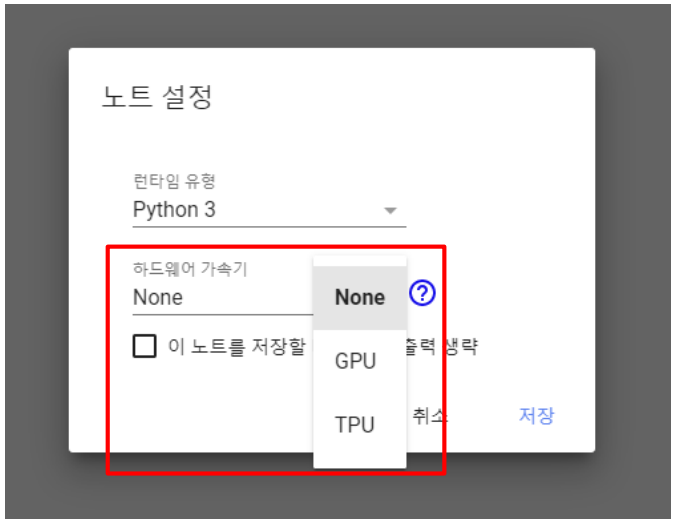
Jupyter notebook에서 수행 결과 확인

Google Colab 활용하기 : GPU 사용 설정하기 (1)

❖ Notebook 메뉴: "수정"→ "노트설정 "

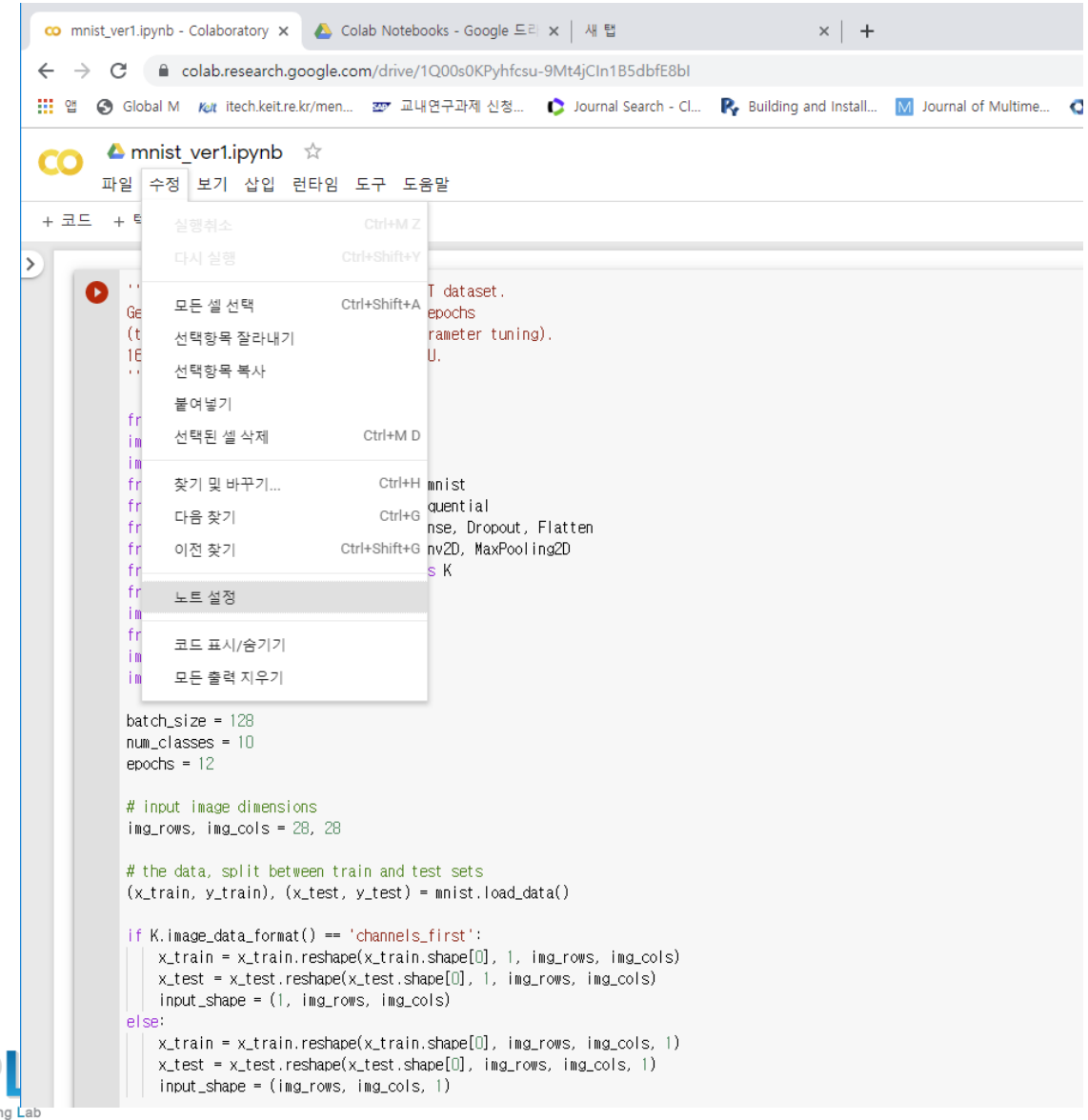


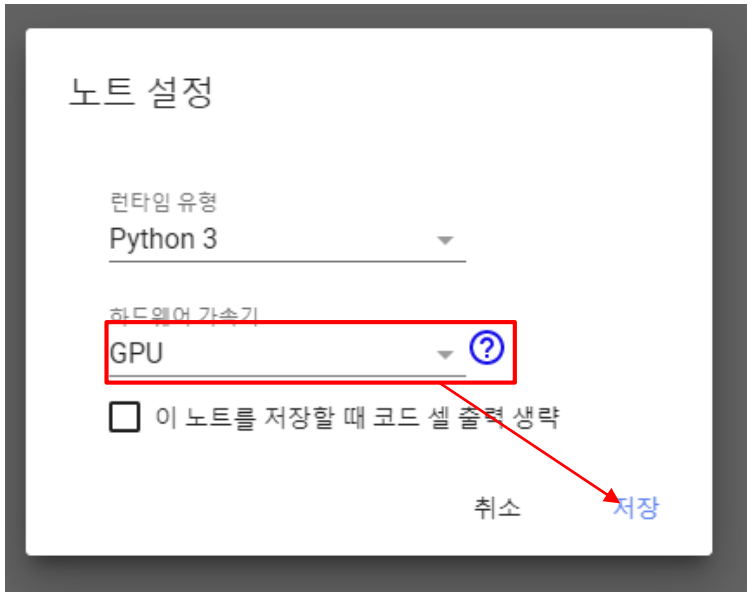




Google Colab 활용하기 : GPU 사용 설정하기 (2)

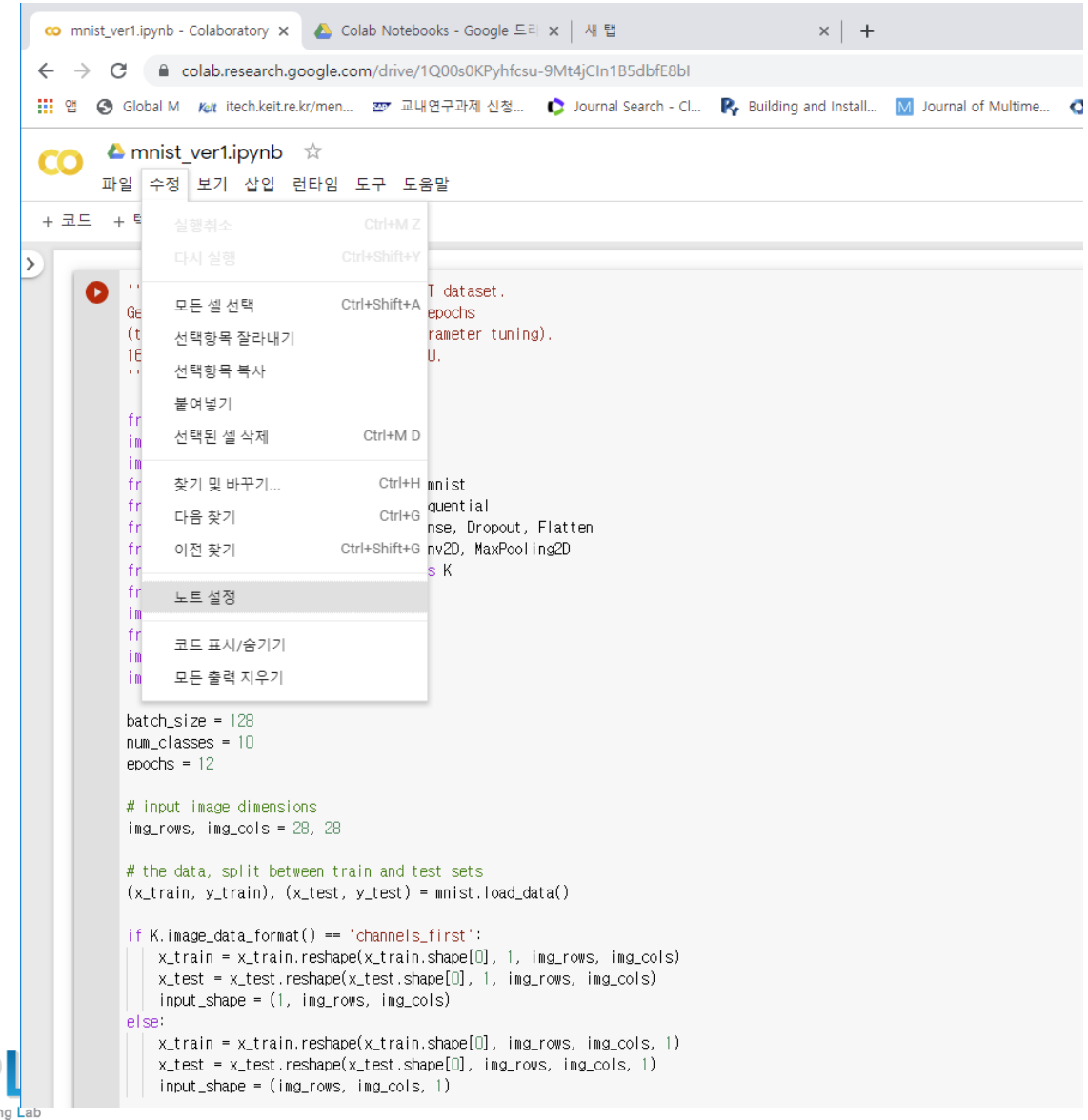
❖ Notebook 메뉴: "수정"→ "노트설정"→ 하드웨어 가속기: GPU 선택 후 저장

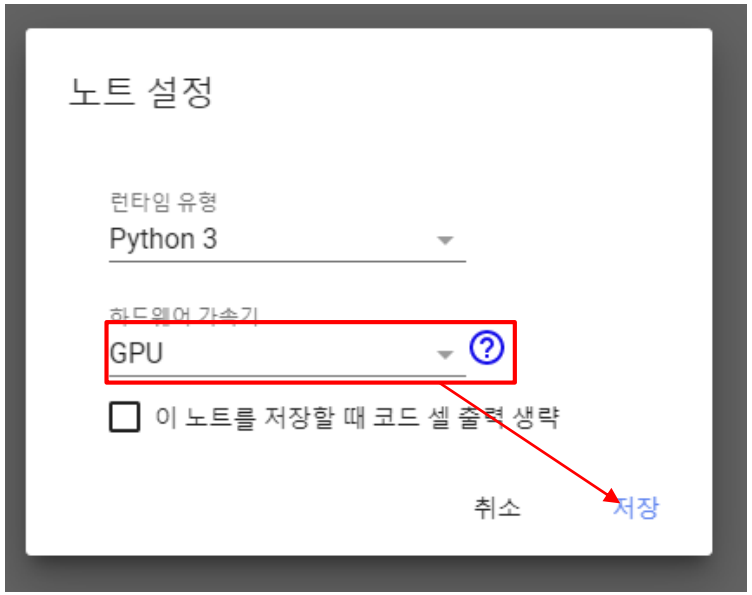




Google Colab 활용하기 : GPU 사용 설정하기 (3)

❖ Notebook 메뉴: “수정”→ “노트설정”→ 하드웨어 가속기: GPU 선택 후 저장

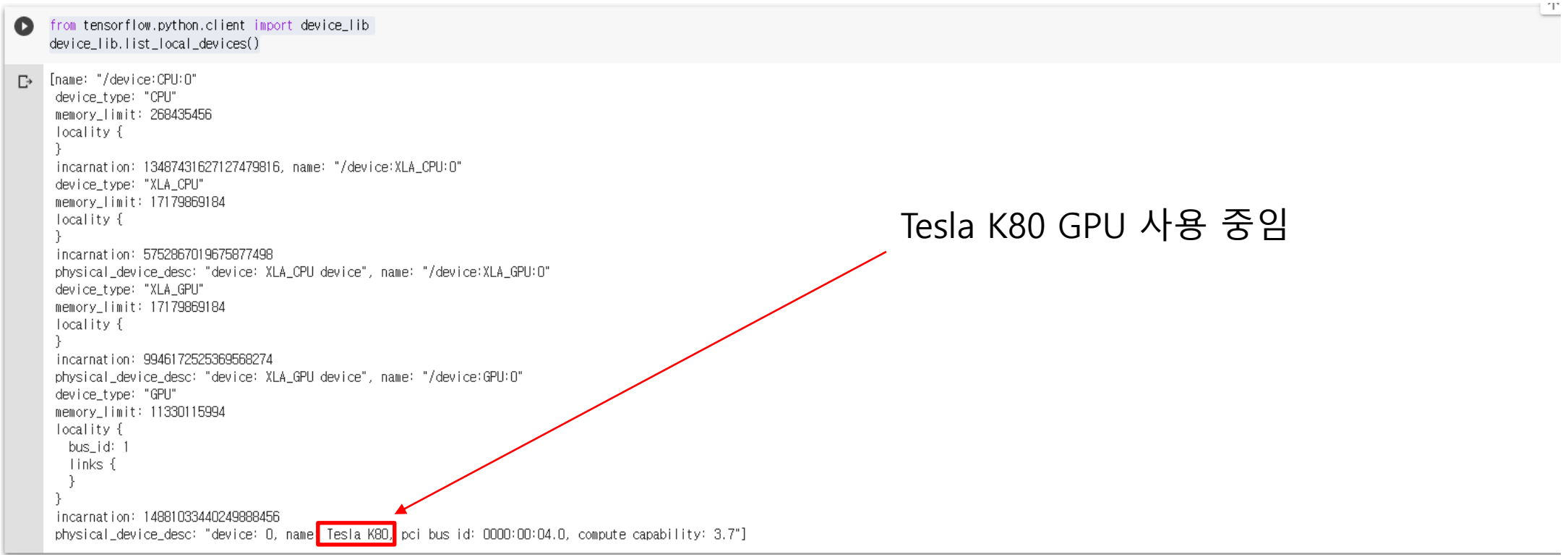




Google Colab 활용하기 : GPU 사용 설정하기 (4)

- GPU 종류 확인하기

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



Mnist : Digit recognition project (1)

❖ 1) google drive를 먼저 연결한다 (강의 자료 10페이지 이후 참고).

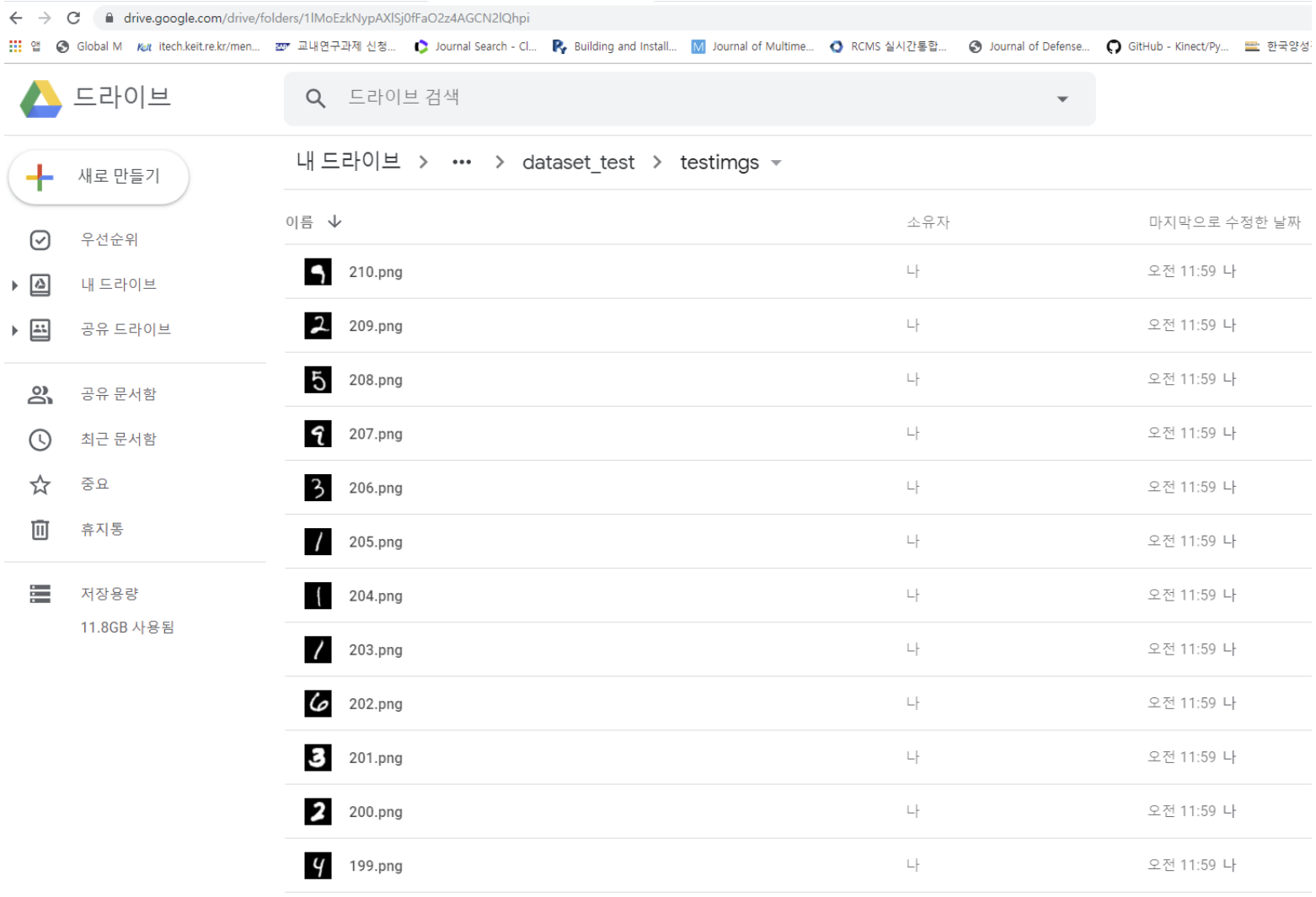
```
from google.colab import drive
drive.mount('/content/gdrive')
```

- "Mounted at /content/gdrive" : Mount is successful....!!!!

Mnist : Digit recognition project (2)

❖ 2) mnist 프로젝트에 필요한 데이트를 원하는 google drive (내 드라이브)에 미리 복사해 놓는다.

"My Drive/DeepLearning/cnn/mnist/dataset_test/testimgs/"



Mnist : Digit recognition project (3)

❖ 3) mnist deep learning 코드를 준비한다.

```
'''Trains a simple convnet on the MNIST dataset.
Gets to 99.25% test accuracy after 12 epochs
(there is still a lot of margin for parameter tuning).
16 seconds per epoch on a GRID K520 GPU.
'''

from __future__ import print_function
import keras
import tensorflow.keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import backend as K
from keras.utils import np_utils
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
import os
from tensorflow.python.client import device_lib

device_lib.list_local_devices()


batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, 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=tensorflow.keras.losses.categorical_crossentropy,
              optimizer="adam",
              metrics=['accuracy'])
```

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

##-- summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()


##-- Model Test using Test datasets
print()
print("----Actual test for digits----")
img = Image.open('/content/gdrive/My Drive/DeepLearning/cnn/mnist/dataset_test/testimgs/1.png').convert("L")
img = np.resize(img, (28,28,1))
im2arr = np.array(img)
im2arr = im2arr.reshape(1,28,28,1)
y_pred = model.predict_classes(im2arr)
print(y_pred)

img = Image.open('/content/gdrive/My Drive/DeepLearning/cnn/mnist/dataset_test/testimgs/5.png').convert("L")
img = np.resize(img, (28,28,1))
im2arr = np.array(img)
im2arr = im2arr.reshape(1,28,28,1)
y_pred = model.predict_classes(im2arr)
print(y_pred)
```

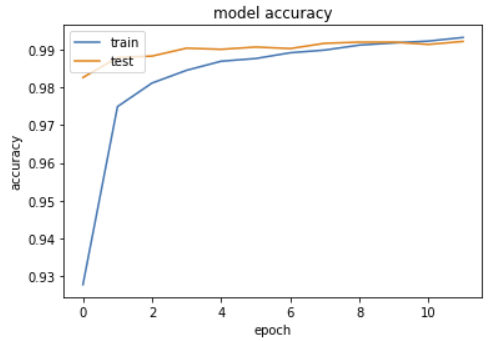
코드 내에 내가 필요한 데이터 폴더를 정확히 명시하여 준다.

IVPL Intelligent Vision Processing Lab

mnist : Digit recognition project (4)

❖ 4) mnist deep learning 코드 실행해 본다 (실행 결과 아래와 같음).

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 8s 128us/sample - loss: 0.2381 - acc: 0.9276 - val_loss: 0.0517 - val_acc: 0.9827
Epoch 2/12
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0821 - acc: 0.9750 - val_loss: 0.0345 - val_acc: 0.9880
Epoch 3/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0624 - acc: 0.9812 - val_loss: 0.0355 - val_acc: 0.9884
Epoch 4/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0504 - acc: 0.9846 - val_loss: 0.0295 - val_acc: 0.9905
Epoch 5/12
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0440 - acc: 0.9870 - val_loss: 0.0284 - val_acc: 0.9902
Epoch 6/12
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0377 - acc: 0.9877 - val_loss: 0.0269 - val_acc: 0.9908
Epoch 7/12
60000/60000 [==============================] - 7s 123us/sample - loss: 0.0342 - acc: 0.9893 - val_loss: 0.0301 - val_acc: 0.9904
Epoch 8/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0315 - acc: 0.9900 - val_loss: 0.0256 - val_acc: 0.9918
Epoch 9/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0277 - acc: 0.9913 - val_loss: 0.0272 - val_acc: 0.9921
Epoch 10/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0251 - acc: 0.9919 - val_loss: 0.0248 - val_acc: 0.9921
Epoch 11/12
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0235 - acc: 0.9924 - val_loss: 0.0282 - val_acc: 0.9915
Epoch 12/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0202 - acc: 0.9934 - val_loss: 0.0259 - val_acc: 0.9923
Test loss: 0.025931414561194334
Test accuracy: 0.9923
```

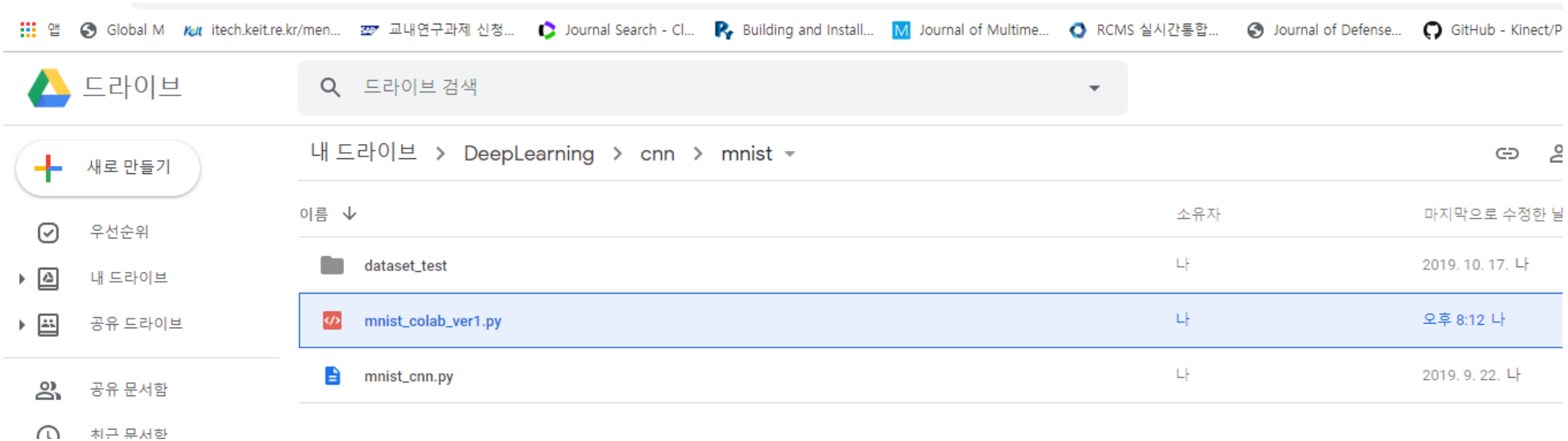



```
----Actual test for digits----
[7]
[4]
```

How to run the developed CNN code? (1)

❖ If already you have your own CNN code, how to run that python code?

```
!python3 "/content/gdrive/My Drive/DeepLearning/cnn/mnist/mnist_colab_ver1.py"
```



How to run the developed CNN code? (2)

- In your Jupyter NoteBook, the following, "**!python3 (your python code)**" and run.

```
!python3 "/content/gdrive/My Drive/DeepLearning/cnn/mnist/mnist_colab_ver1.py"
```

```
Using TensorFlow backend.
2019-10-20 11:12:47.702370: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2300000000 Hz
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
2019-10-20 11:12:
name: Tesla K80
```

```
Epoch 1/12
2019-10-20 11:12:49.336796: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcublas.so.10.0
2019-10-20 11:12:49.513900: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
60000/60000 [==============================] - 9s 147us/sample - loss: 0.2463 - acc: 0.9246 - val_loss: 0.0532 - val_acc: 0.9824
Epoch 2/12
60000/60000 [==============================] - 7s 122us/sample - loss: 0.0884 - acc: 0.9734 - val_loss: 0.0400 - val_acc: 0.9869
Epoch 3/12
60000/60000 [==============================] - 7s 120us/sample - loss: 0.0672 - acc: 0.9802 - val_loss: 0.0362 - val_acc: 0.9882
Epoch 4/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0536 - acc: 0.9837 - val_loss: 0.0365 - val_acc: 0.9877
Epoch 5/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0469 - acc: 0.9854 - val_loss: 0.0322 - val_acc: 0.9897
Epoch 6/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0408 - acc: 0.9879 - val_loss: 0.0275 - val_acc: 0.9910
Epoch 7/12
60000/60000 [==============================] - 7s 120us/sample - loss: 0.0352 - acc: 0.9887 - val_loss: 0.0279 - val_acc: 0.9910
Epoch 8/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0311 - acc: 0.9899 - val_loss: 0.0283 - val_acc: 0.9915
Epoch 9/12
60000/60000 [==============================] - 7s 120us/sample - loss: 0.0308 - acc: 0.9900 - val_loss: 0.0267 - val_acc: 0.9918
Epoch 10/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0271 - acc: 0.9916 - val_loss: 0.0326 - val_acc: 0.9901
Epoch 11/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0248 - acc: 0.9920 - val_loss: 0.0258 - val_acc: 0.9920
Epoch 12/12
60000/60000 [==============================] - 7s 121us/sample - loss: 0.0231 - acc: 0.9919 - val_loss: 0.0289 - val_acc: 0.9914
Test loss: 0.028908748149796337
Test accuracy: 0.9914
Consumed time:  89.42521524429321 (sec)
<Figure size 640x480 with 1 Axes>

----Actual test for digits----
[9]
[4]
```

How to run the developed CNN code? (3)

- If you got the following syntax error when "**!python3 (your python code)**":
  - **"from __future__ import print_function" should be in the first import line.** That is, all comments and some sentences should be removed in your python source file.

```
'''Trains a simple convnet on the MNIST dataset.
Gets to 99.25% test accuracy after 12 epochs
(there is still a lot of margin for parameter tu
ning).
16 seconds per epoch on a GRID K520 GPU.
'''
##-- google drive mounting to this project
#from google.colab import drive
#drive.mount('/content/gdrive')


from __future__ import print_function
import keras
import tensorflow.keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
     ( ~~~~~ )
```

```
from __future__ import print_function
import keras
import tensorflow.keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, F
latten
from tensorflow.keras.layers import Conv2D, MaxPoolin
g2D
from tensorflow.keras import backend as K
from keras.utils import np_utils
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
import os, time

              ( ~~~~~~  )
```

Thank you for your attention.!!!
QnA

http://ivpl.sookmyung.ac.kr