

## Chapter 44. Mask man을 찾아라

---



# 그럼 LeNET으로 뭘 얼마나 해볼 수 있을까?

# 마스크 쓴 사람 안쓴사람 데이터~



Dataset 204

**Face Mask Detection ~12K Images Dataset**  
12K Images divided in training testing and validation directories.

Ashish Jangra • updated 2 years ago (Version 1)

Data    Code (119)    Discussion (2)    Activity    Metadata    Download (345 MB)    New Notebook    :

Usability 7.5

License CC0: Public Domain

Tags earth and nature

<https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset>

난 코드와 같은 위치에 data라는 폴더를 두고 그 안에 다운 받은 파일을 위치시킴

이름	수정일	크기	옵션
06_a_new_hope.txt	2018년 4월 2일 오전 7:42	326KB	열기
06_alice_mask.png	2018년 4월 2일 오전 7:42	7KB	P
06_alice.txt	2018년 4월 2일 오전 7:42	149KB	열기
06_stormtrooper_mask.png	2018년 4월 2일 오전 7:42	13KB	P
06. Naver API.ipynb	2021년 9월 29일 오후 1:35	893KB	V
32. recommendations.ipynb	2021년 12월 3일 오후 11:35	85KB	V
33. good books.ipynb	2021년 12월 2일 오전 8:47	93KB	V
44. mask man.ipynb	오늘 오전 10:41	1MB	V
clf_quiz.ipynb	2021년 12월 9일 오후 1:47	4MB	V
▼ data	오늘 오후 4:09	--	폴더
archive.zip	오늘 오전 10:28	346.3MB	Z
▶ goodbooks-10k	2021년 12월 2일 오전 12:48	--	폴더
▶ OpenCV	오늘 오전 10:11	--	폴더
Untitled.ipynb	2021년 12월 2일 오전 10:16	1KB	V
x09.txt	2019년 2월 8일 오후 11:19	1KB	열기

## 그러니까 경로를 확인하고~

```
| ls
```

06. Naver API.ipynb*	44. mask man.ipynb
06_a_new_hope.txt*	OpenCV/
06_alice.txt*	Untitled.ipynb
06_alice_mask.png*	clf_quiz.ipynb
06_stormtrooper_mask.png*	<b>data/</b> ←
32. recommendations.ipynb	<b>goodbooks-10k/</b>
33. good books.ipynb	x09.txt*

```
| ls './data/'
```

archive.zip

## 파이썬도 압축파일을 관리하는 툴이 있다

```
import zipfile

content_zip = zipfile.ZipFile("./data/archive.zip")
content_zip.extractall("./data")

content_zip.close()
```

결과~

```
| ls './data/'
```

Face Mask Dataset/ archive.zip

```
| ls './data/Face Mask Dataset/'
```

Test/

Train/

Validation/

## 사용할 모듈이 좀 많다...

```
import numpy as np  
import pandas as pd  
import os  
import glob  
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
import tensorflow as tf  
from tensorflow.keras import Sequential, models  
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPool2D  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import classification_report, confusion_matrix
```

```
ls './data/Face Mask Dataset/'
```

**Test/**      **Train/**      **Validation/**

```
ls './data/Face Mask Dataset/Train'
```

**WithMask/**      **WithoutMask/**

## 생소할 수 있지만 파일을 이렇게 정리하기도 한다

```
path = "./data/Face Mask Dataset/"  
dataset = {"image_path": [], "mask_status": [], "where": []}  
  
for where in os.listdir(path):  
    for status in os.listdir(path + "/" + where):  
        for image in glob.glob(path + where + "/" + status + "/" + "*.png"):  
            dataset["image_path"].append(image)  
            dataset["mask_status"].append(status)  
            dataset["where"].append(where)
```

## 경로와 목록의 정리~

```
dataset = pd.DataFrame(dataset)  
dataset.head()
```

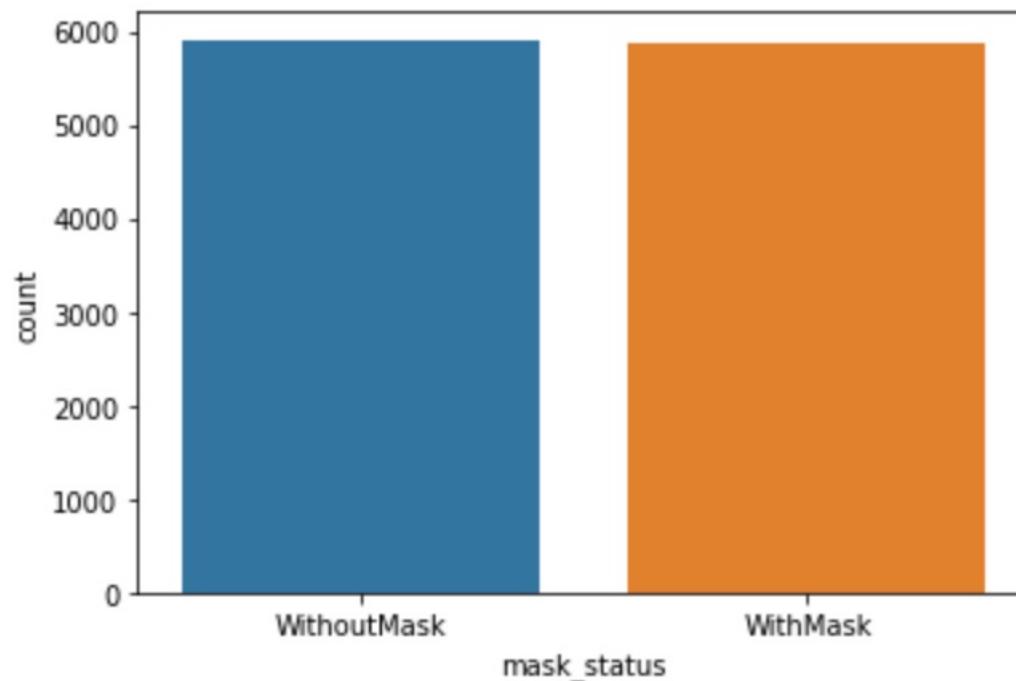
	image_path	mask_status	where
0	./data/Face Mask Dataset/Test/WithoutMask/2734...	WithoutMask	Test
1	./data/Face Mask Dataset/Test/WithoutMask/4345...	WithoutMask	Test
2	./data/Face Mask Dataset/Test/WithoutMask/4423...	WithoutMask	Test
3	./data/Face Mask Dataset/Test/WithoutMask/2052...	WithoutMask	Test
4	./data/Face Mask Dataset/Test/WithoutMask/3364...	WithoutMask	Test

```
print("With Mask:", dataset.value_counts("mask_status")[0])
print("Without Mask:", dataset.value_counts("mask_status")[1])

sns.countplot(x=dataset["mask_status"]);
```

With Mask: 5909

Without Mask: 5883



## 랜덤하게 어떤 그림들이 있는지 보자

```
import cv2

plt.figure(figsize=(15, 10))
for i in range(9):
    random = np.random.randint(1, len(dataset))
    plt.subplot(3, 3, i + 1)
    plt.imshow(cv2.imread(dataset.loc[random, "image_path"]))
    plt.title(dataset.loc[random, "mask_status"], size=15)
    plt.xticks([])
    plt.yticks([])
plt.show()
```

WithMask



WithMask



WithoutMask



WithMask



WithMask



WithoutMask



WithMask



WithoutMask



WithoutMask



## Train / test / validation으로 나눌 필요는 없다. (되어있으니까)

```
train_df = dataset[dataset["where"] == "Train"]
test_df = dataset[dataset["where"] == "Test"]
valid_df = dataset[dataset["where"] == "Validation"]

train_df.head(5)
```

	image_path	mask_status	where
992	./data/Face Mask Dataset/Train/WithoutMask/397...	WithoutMask	Train
993	./data/Face Mask Dataset/Train/WithoutMask/348...	WithoutMask	Train
994	./data/Face Mask Dataset/Train/WithoutMask/180...	WithoutMask	Train
995	./data/Face Mask Dataset/Train/WithoutMask/496...	WithoutMask	Train
996	./data/Face Mask Dataset/Train/WithoutMask/181...	WithoutMask	Train

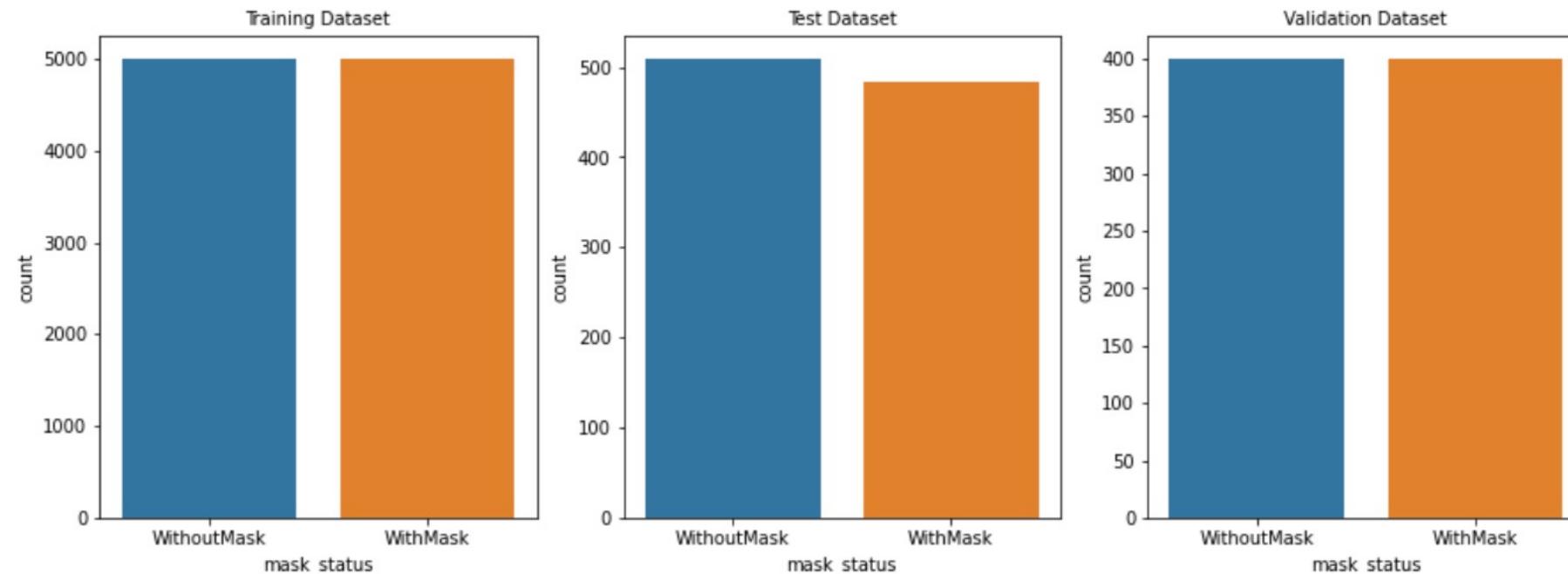
## Train / test / validation 데이터의 분포 확인

```
| plt.figure(figsize=(15, 5))
| plt.subplot(1, 3, 1)
| sns.countplot(x=train_df[ "mask_status" ])
| plt.title("Training Dataset", size=10)

plt.subplot(1, 3, 2)
sns.countplot(x=test_df[ "mask_status" ])
plt.title("Test Dataset", size=10)

plt.subplot(1, 3, 3)
sns.countplot(x=valid_df[ "mask_status" ])
plt.title("Validation Dataset", size=10)

plt.show()
```



## Index를 다시 정리

```
train_df = train_df.reset_index().drop("index", axis=1)  
train_df.head()
```

	image_path	mask_status	where
0	./data/Face Mask Dataset/Train/WithoutMask/397...	WithoutMask	Train
1	./data/Face Mask Dataset/Train/WithoutMask/348...	WithoutMask	Train
2	./data/Face Mask Dataset/Train/WithoutMask/180...	WithoutMask	Train
3	./data/Face Mask Dataset/Train/WithoutMask/496...	WithoutMask	Train
4	./data/Face Mask Dataset/Train/WithoutMask/181...	WithoutMask	Train

## 드디어 데이터 전처리

```
data = []
image_size = 150

for i in range(len(train_df)):
    # Converting the image into grayscale
    img_array = cv2.imread(train_df["image_path"][i], cv2.IMREAD_GRAYSCALE)

    # Resizing the array
    new_image_array = cv2.resize(img_array, (image_size, image_size))

    # Encoding the image with the label
    if train_df["mask_status"][i] == "WithMask":
        data.append([new_image_array, 1])
    else:
        data.append([new_image_array, 0])
```

## Data 하나는 이렇게 생겼다

data[0]

```
[array([[19, 20, 21, ..., 28, 29, 28],  
       [19, 20, 21, ..., 25, 24, 21],  
       [20, 20, 21, ..., 22, 21, 19],  
       ...,  
       [18, 18, 18, ..., 24, 26, 24],  
       [16, 17, 18, ..., 25, 29, 33],  
       [16, 18, 20, ..., 24, 33, 46]], dtype=uint8),  
  0]
```

## Data를 좀 흔들고~

```
| np.random.shuffle(data)
```

```
| data[0]
```

```
[array([[24, 27, 28, ..., 67, 67, 67],  
       [24, 27, 28, ..., 68, 68, 68],  
       [24, 27, 28, ..., 68, 68, 68],  
       ...,  
       [80, 81, 82, ..., 90, 90, 90],  
       [80, 81, 82, ..., 90, 90, 90],  
       [80, 81, 82, ..., 90, 90, 90]], dtype=uint8),  
 1]
```

```
| fig, ax = plt.subplots(2, 3, figsize=(10, 6))

for row in range(2):
    for col in range(3):
        image_index = row * 100 + col

        ax[row, col].axis("off")
        ax[row, col].imshow(data[image_index][0], cmap="gray")

        if data[image_index][1] == 0:
            ax[row, col].set_title("Without Mask")
        else:
            ax[row, col].set_title("With Mask")
```

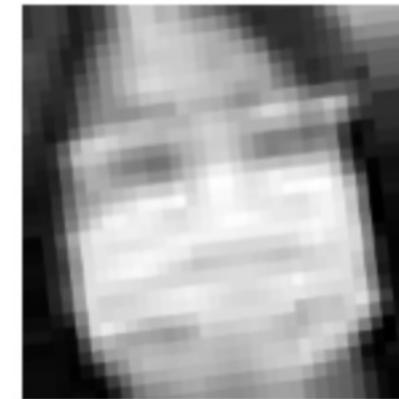
With Mask



With Mask



With Mask



Without Mask



With Mask



Without Mask



## X, y 데이터로 저장하고

```
| X = []
| y = []

for image in data:
    X.append(image[0])
    y.append(image[1])

X = np.array(X)
y = np.array(y)
```

## 트레인 데이터를 다시 나누자~ (이유는 딱히 없음)

```
| X_train, X_val, y_train, y_val = train_test_split(X, y,  
|                                                 test_size=0.2,  
|                                                 random_state=13)
```

```
| from tensorflow.keras import layers, models  
  
model = models.Sequential(  
    [  
        layers.Conv2D(  
            32, kernel_size=(5, 5), strides=(1, 1), padding="same",  
            activation="relu", input_shape=(150, 150, 1) ),  
        layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),  
        layers.Conv2D(64, (2, 2), activation="relu", padding="same"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Dropout(0.25),  
        layers.Flatten(),  
        layers.Dense(1000, activation="relu"),  
        layers.Dense(1, activation="sigmoid"),  
    ]  
)
```

## Compile~

```
model.compile(  
    optimizer="adam", loss=tf.keras.losses.BinaryCrossentropy(),  
    metrics=["accuracy"]  
)
```

휴~ 느리다.ㅠㅠ.

```
X_train = X_train.reshape(len(X_train), X_train.shape[1], X_train.shape[2], 1)
X_val = X_val.reshape(len(X_val), X_val.shape[1], X_val.shape[2], 1)
history = model.fit(X_train, y_train, epochs=4, batch_size=32)
```

```
Epoch 1/4
250/250 [=====] - 109s 436ms/step - loss: 0.12
85 - accuracy: 0.9519
Epoch 2/4
34/250 [==>.....] - ETA: 1:48 - loss: 0.0742 - a
ccuracy: 0.9733
```

## Validation accuracy가 나쁘지 않다

```
| model.evaluate(X_val, y_val)
```

```
63/63 [=====] - 4s 68ms/step - loss: 0.1026 -
accuracy: 0.9700
```

```
[0.10262694209814072, 0.9700000286102295]
```

상대적으로 0에 대한 recall이 조금 떨어지지만 ..

```
prediction = (model.predict(X_val) > 0.5).astype("int32")

print(classification_report(y_val, prediction))
print(confusion_matrix(y_val, prediction))
```

	precision	recall	f1-score	support
0	0.99	0.95	0.97	1046
1	0.95	0.99	0.97	954
accuracy			0.97	2000
macro avg	0.97	0.97	0.97	2000
weighted avg	0.97	0.97	0.97	2000

```
[[991  55]
 [  5 949]]
```

틀린것만 추리고~

```
wrong_result = []

for n in range(0, len(y_val)):
    if prediction[n] != y_val[n]:
        wrong_result.append(n)

len(wrong_result)
```

```
import random

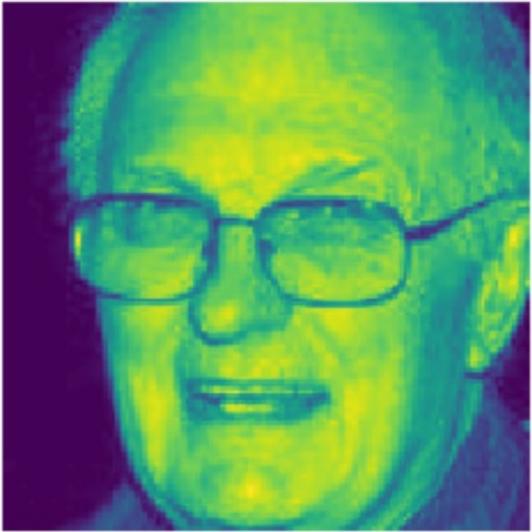
samples = random.choices(population=wrong_result, k=6)

plt.figure(figsize=(14, 12))

for idx, n in enumerate(samples):
    plt.subplot(3, 2, idx + 1)
    plt.imshow(X_val[n].reshape(150, 150), interpolation="nearest")
    plt.title(prediction[n])
    plt.axis("off")

plt.show()
```

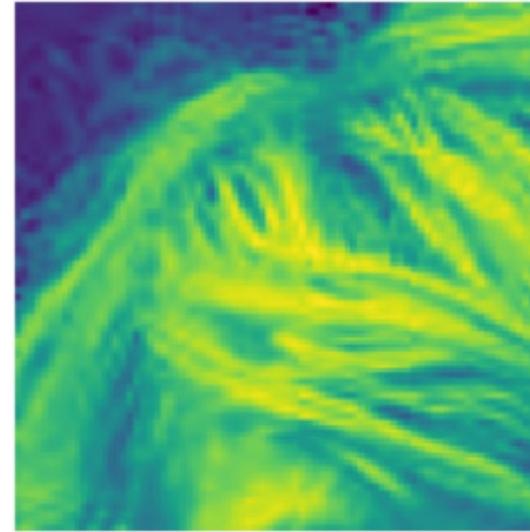
[1]



[1]



[1]



ero-base /

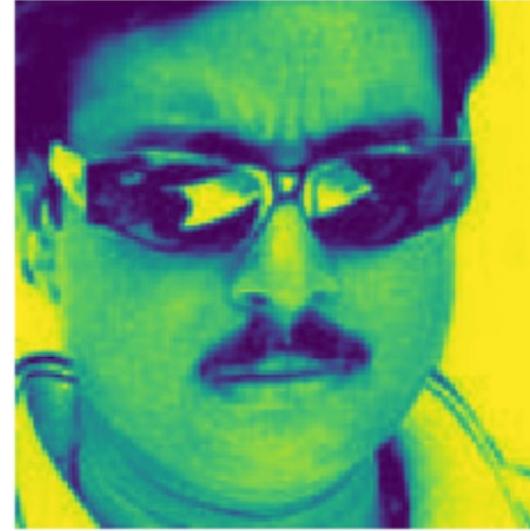
[1]



[1]



[1]



Reserved

너무 느리다면 colab으로 해보자

이름 ↑	소유자
.ipynb_checkpoints	나
archive.zip 	나
cats_dogs_sounds.zip	나
quiz1.ipynb	나



드라이브



드라이브에서 검색

브 &gt; 임시공유 &gt; Python &gt; DeepLearning ▾



폴더



파일 업로드



폴더 업로드



Google 문서



Google 스프레드시트



Google 프레젠테이션



Google 설문지



더보기



휴지통



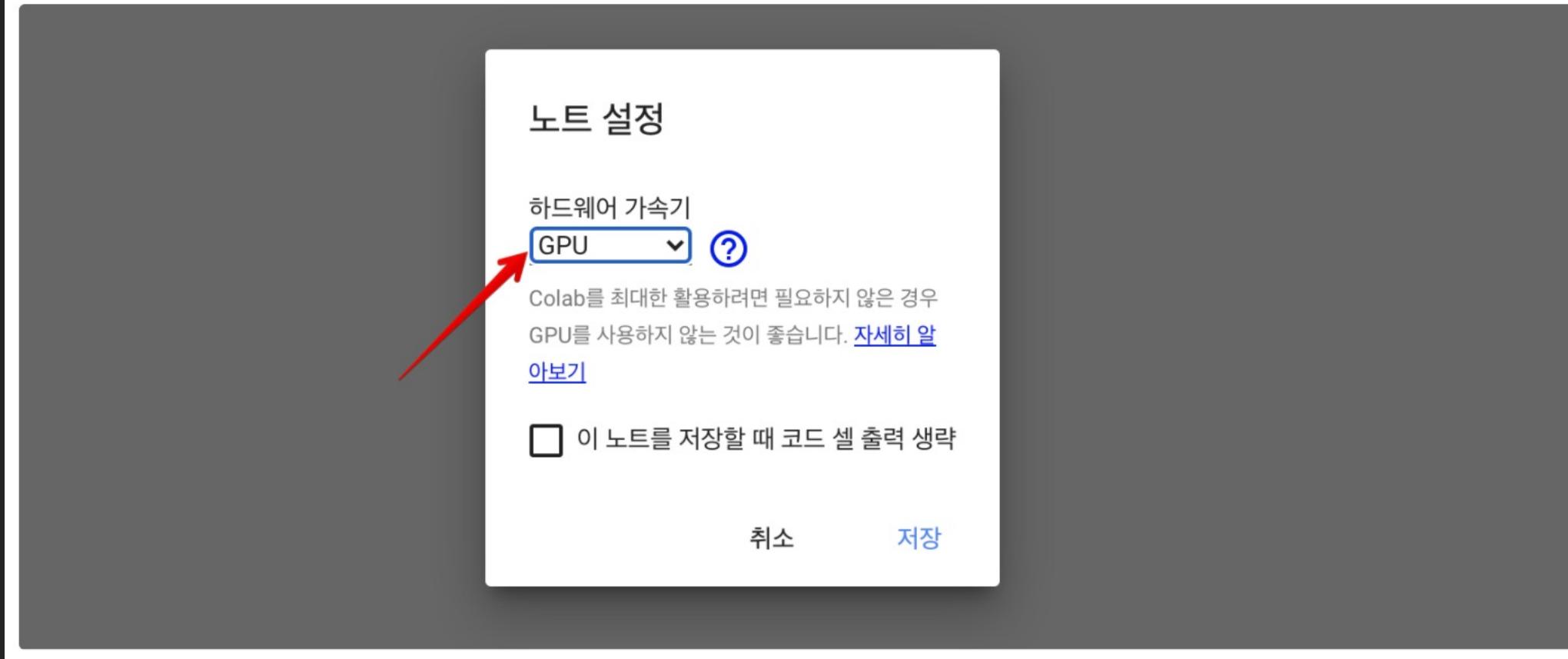
저장용량

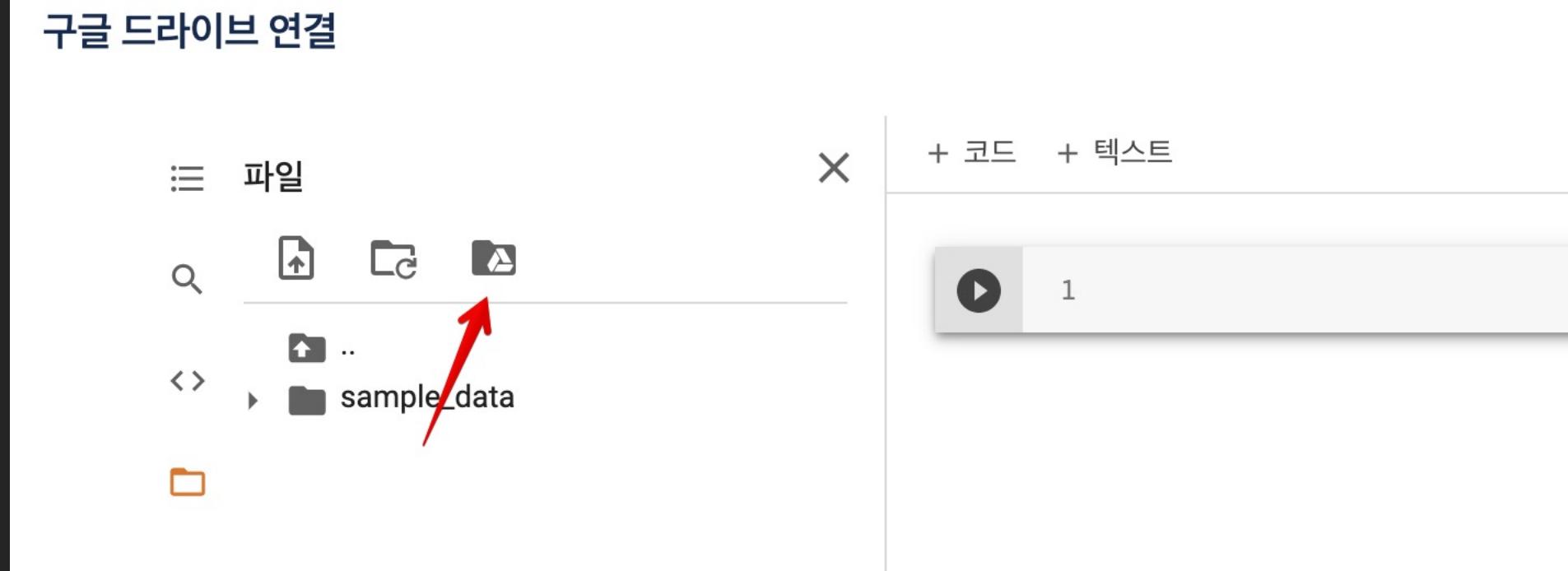
100GB 중 57.28GB 사용

저장용량 구매

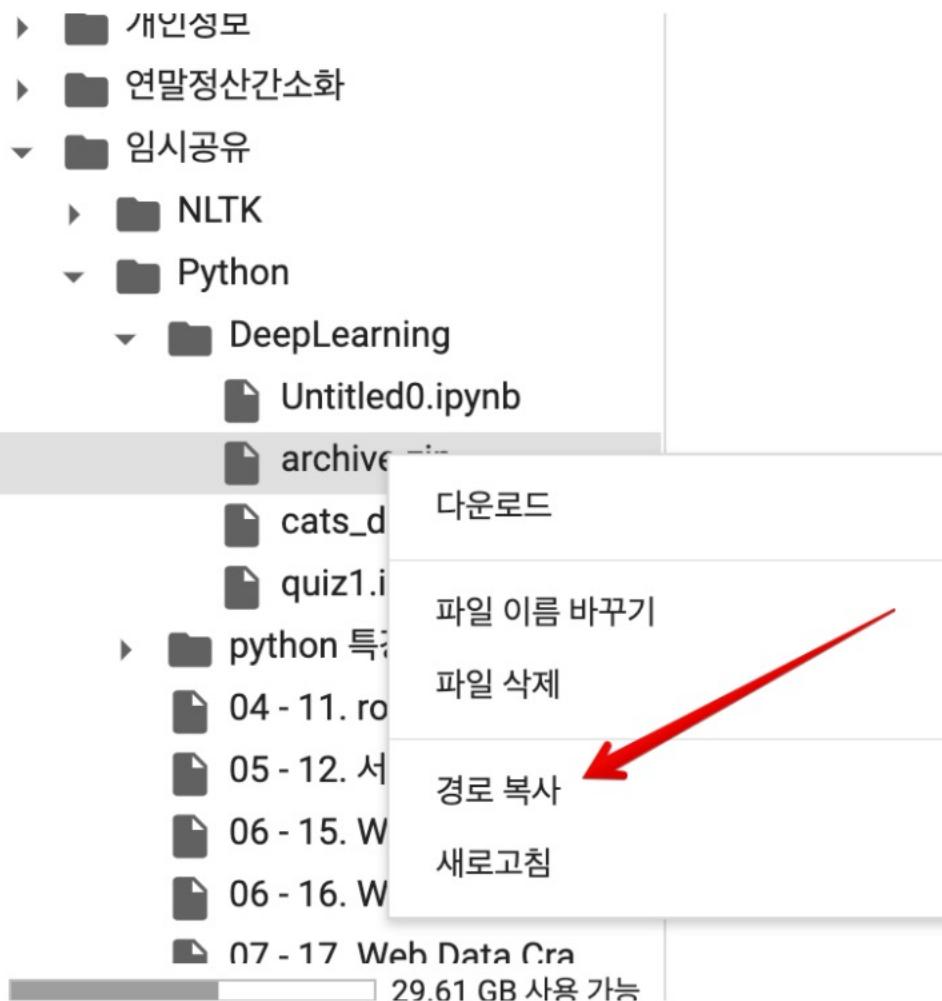
- Google 드로잉
  - Google 내 지도
  - Google 사이트 도구
  - Google Apps Script
  - Google Colaboratory ←
  - Google Jamboard
  - Text Editor
- + 연결할 앱 더보기

## 런타임 유형변경





## 압축파일 경로를 찾아서 복사



현재 경로가 Content인게 중요

```
[1] 1 %pwd
```

```
'/content' ←
```



```
1 # /content/drive/MyDrive/임시공유/Python/DeepLearning/archive.zip
```

+ 코드

+ 텍스트

## 압축을 풀고 새로 고침을 하면 나타남

The screenshot shows a Jupyter Notebook environment. On the left, there is a sidebar titled "파일" (File) with icons for search, upload, download, and folder operations. A red arrow points to the folder icon. Below the sidebar, a tree view shows the directory structure: .., Face Mask Dataset (highlighted with a red arrow), drive, and sample\_data. The main area contains a code editor with the following content:

```
[1] 1 %pwd  
'/content'  
  
[2] 1 # /content/drive/MyDrive/임시공유/Python/DeepLearning/archive.zip  
  
1 !unzip -qq "/content/drive/MyDrive/임시공유/Python/DeepLearning/archive.zip"
```

At the bottom of the code editor, there are buttons for "+ 코드" (Add Code) and "+ 텍스트" (Add Text).

이후 코드 동일~

# 단 학습 시간만 놓고 보면 좋음~~~



```
1 X_train = X_train.reshape(len(X_train), X_train.shape[1], X_train.shape[2], 1)
2 X_val = X_val.reshape(len(X_val), X_val.shape[1], X_val.shape[2], 1)
3
4 history = model.fit(X_train, y_train, epochs=5, batch_size = 32)
```

↪ Epoch 1/5

250/250 [=====] - 40s 35ms/step - loss: 41.0547 - accuracy: 0.9001

Epoch 2/5

250/250 [=====] - 9s 35ms/step - loss: 0.0519 - accuracy: 0.9816

Epoch 3/5

250/250 [=====] - 9s 35ms/step - loss: 0.0353 - accuracy: 0.9860

Epoch 4/5

250/250 [=====] - 9s 35ms/step - loss: 0.0191 - accuracy: 0.9936

Epoch 5/5

250/250 [=====] - 9s 35ms/step - loss: 0.0160 - accuracy: 0.9941