1-1. 인공지능과 가위바위보 하기
30분

1-2. 데이터를 준비하자!
30분

1-3. 딥러닝 네트워크 설계하기
30분

1-4. 딥러닝 네트워크 학습시키기
30분

1-5. 얼마나 잘 만들었는지 확인하기
30분

1-6. 더 좋은 네트워크 만들어 보기
30분

1-7. 프로젝트: 가위바위보 분류기 만들기
180분

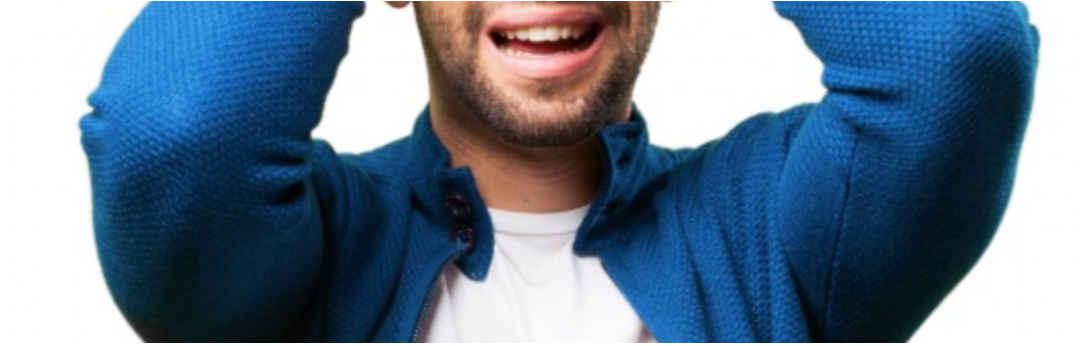
1-7. 프로젝트: 가위바위보 분류기 만들기

오늘 배운 내용을 바탕으로 가위바위보 분류기를 만들도록 하겠습니다. 가장 먼저 해야 할 일은 뭘까요? 네, 첫 번째!!!! 데이터를 준비해야 합니다. 가위바위보 이미지를 모아 놓은 곳은 없으므로, 우리가 직접 사진을 찍어서 모아봅시다.

데이터를 준비하자

데이터 만들기

(1) 우리는 노트북 전면 카메라를 활용하여 가위, 바위, 보 이미지 각 100장을 만들어 볼거예요. 그런데 300장을 어느 세월에 만들까요?

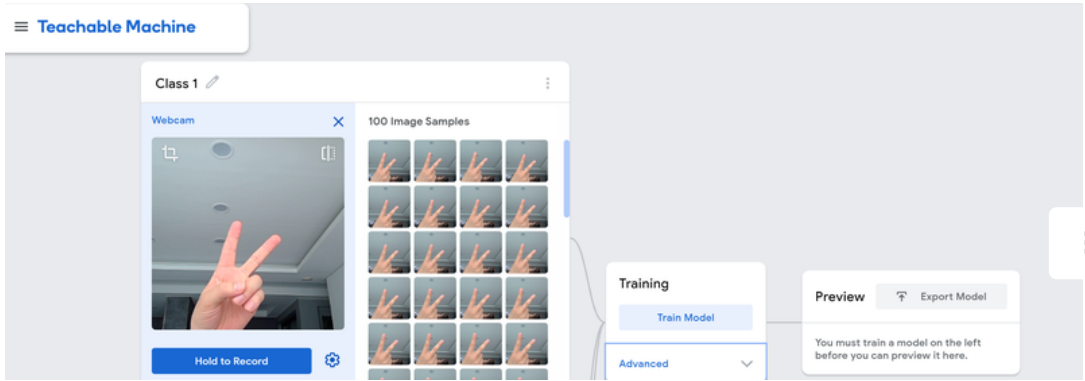


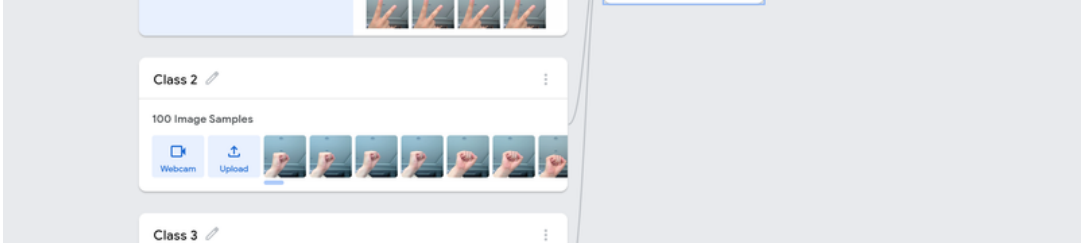
걱정하지 마세요. 구글의 teachable machine 사이트에서 쉽게 데이터를 만들어볼 수 있습니다. 아래 사이트에서 Get Started 버튼을 눌러보세요. 그 다음, Image Project를 선택하면 Webcam을 구동해 클래스별 이미지 데이터를 직접 촬영해서 만들 수 있는 멋진 화면이 나타납니다.

https://teachablemachine.withgoogle.com/

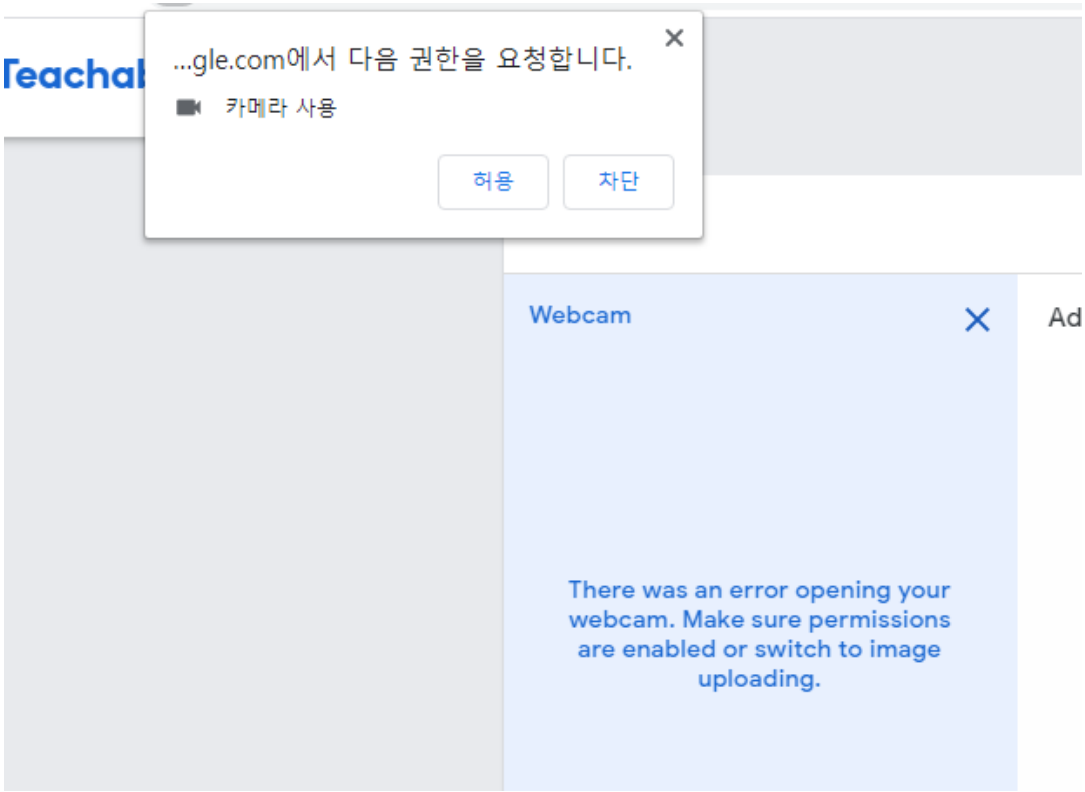
(2) 먼저 가위 이미지 데이터를 만들어 봅시다. 웹캠 앞에 가위 포즈를 취하면서 버튼을 누르면 이미지가 캡쳐됩니다. 딥러닝 모델이 인식하기 좋게끔 여러분들 손이 잘 보이게 찍어주세요.

- 여러 각도에서 찍어보세요.
- 여러 크기로 찍어보세요.
- 혼자하면 다양한 각도와 크기를 저장할 수 없으니, 옆 동료와 함께 하세요.
- 좋은 데이터가 좋은 결과를 낳는다는 것을 꼭 기억하세요.

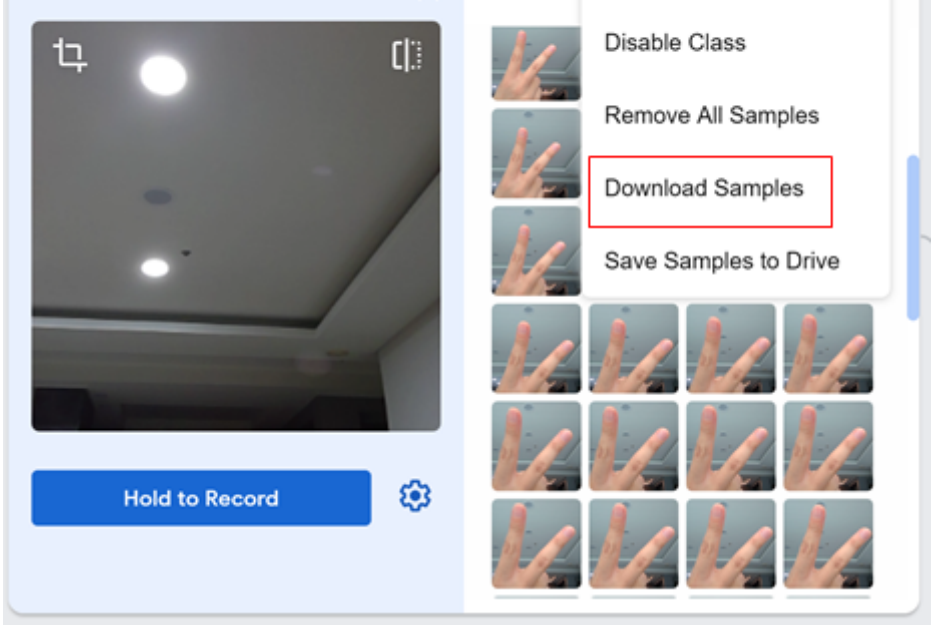




주의 만약 웹캠 사용 버튼을 눌렀을 때 아래 화면처럼 에러가 난다면, 브라우저에서 웹캠을 사용할 수 있는 권한을 허용해 주어야 합니다.



(3) 100장의 가위 이미지를 캡쳐했다면, 우상단의 메뉴 아이콘을 눌러 다운로드 합니다.



(4) 가위 이미지 100장을 모두 저장 했다면, 바위 및 보 이미지에 대해서도 위 과정을 진행하세요. 가위는 scissor 폴더에, 바위는 rock 폴더에, 보는 paper 폴더에 각각 압축을 풀어 봅시다. 각 폴더안에 100개의 이미지가 들어있다면 성공!!

추후 프로그램 작성의 통일성을 위해, rock_scissor_paper 라는 폴더 아래에 scissor, rock, paper 폴더를 만들어서 이미지를 저장합시다. 각 이미지는 아래 폴더 안에 들어가야 합니다.

디렉토리 만들기

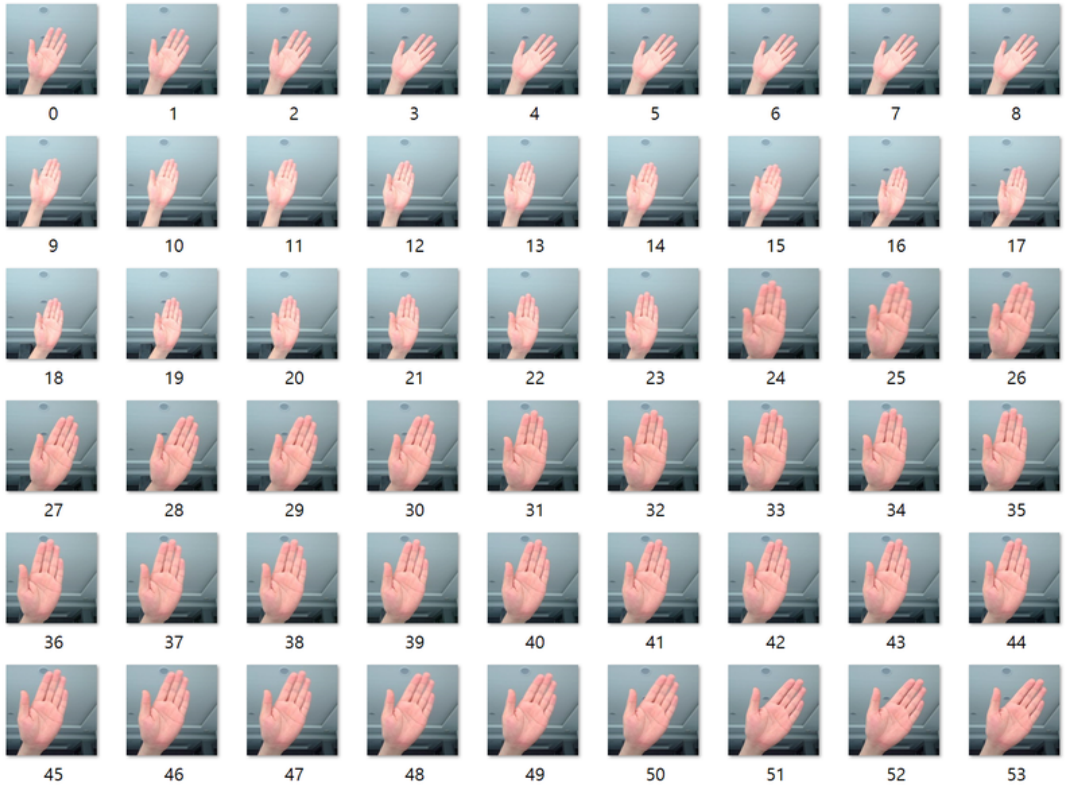
본인의 환경에 따라 실습용 디렉토리 `rock_scissor_paper` 및 하위 디렉토리를 만들어 주세요.

```
$ mkdir -p ~/aiffel/rock_scissor_paper/scissor
$ mkdir -p ~/aiffel/rock_scissor_paper/rock
$ mkdir -p ~/aiffel/rock_scissor_paper/paper

$ ls -l ~/aiffel/rock_scissor_paper
```

(토막 리눅스 사용법)

`mkdir -p` : mkdir를 사용하여 하위 디렉토리를 생성할때 차례대로 만들지 않고 중간 디렉토리 없이 바로 그 다음 하위 디렉토리를 만들게되면 "디렉토리를 생성할 수 없습니다." 라는 메시지가 나오는데, **-p 옵션**을 주어 생성하게 되면 자동으로 중간 단계의 디렉토리를 생성하면서 그 하위 디렉토리를 생성하게 됩니다.



[rock_scissor_paper/paper 폴더 내 이미지들의 예]

Q8. 다운로드 받은 이미지는 크기는 무엇일까요? "nxn"(n은 정수)의 형태로 나타내 보세요.

정답을 입력하신 뒤 예시답안 버튼을 눌러주세요.

제출

데이터 불러오기 + Resize 하기

(5) 숫자 손글씨의 경우 이미지 크기가 28x28 이었기 때문에, 우리의 가위, 바위, 보 이미지도 28x28로 만들어야 합니다. 이를 위해서는 PIL 라이브러리를 사용해볼 거예요. 그러려면 먼저 라이브러리를 불러와야 겠죠?
혹시 PIL 라이브러리가 없는 경우 필요한 패키지를 설치해 주세요.

[Input]

```
# PIL 라이브러리가 설치되어 있지 않다면 설치
!pip install pillow

from PIL import Image
import os, glob

print("PIL 라이브러리 import 완료!")
```

실행

[Output]

이제 가위 이미지를 불러와서 28x28 사이즈로 변경할 겁니다. 아래 코드를 실행해보세요. 이미지의 크기가 28x28 로 바뀌었나요?

[Input]

```
import os
```

```
# 가위 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서
image_dir_path = os.getenv("HOME") + "/aiffel/rock_scissor_paper/scissor"
print("이미지 디렉토리 경로: ", image_dir_path)

images=glob.glob(image_dir_path + "/*.jpg")

# 파일마다 모두 28x28 사이즈로 바꾸어 저장합니다.
target_size=(28,28)
for img in images:
    old_img=Image.open(img)
    new_img=old_img.resize(target_size,Image.ANTIALIAS)
    new_img.save(img,"JPEG")

print("가위 이미지 resize 완료!")
```

실행

[Output]

자 그러면, 바위 이미지도 28x28 로 만들어 볼까요? 아래 빈 칸에 코드를 작성하고, 실행해보세요. 바위 이미지가 모두 28x28로 바뀌어야 합니다.

[Input]

```
# 바위 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서
# [[YOUR CODE]]

# 파일마다 모두 28x28 사이즈로 바꾸어 저장합니다.
# [[YOUR CODE]]
```

실행

[Output]

마지막으로 보 이미지도 28x28로 만들어 봅시다.

[Input]

```
# 보 이미지가 저장된 디렉토리 아래의 모든 jpg 파일을 읽어들여서
# [[YOUR CODE]]

# 파일마다 모두 28x28 사이즈로 바꾸어 저장합니다.
# [[YOUR CODE]]
```

실행

[Output]

(6) 숫자 손글씨 인식기는 `mnist.load_data()` 라는 함수로 데이터를 읽었던 것 기억하시죠? 여러분들이 아직 코딩에 익숙하지 않을 수 있으므로, 가위, 바위, 보 데이터를 읽을 수 있는 `load_data()` 함수를 만들어 드릴 거예요. 이 코드를 활용하면 임의의 사진 데이터(ex. 귤이 잘 익었나, 안 익었나? 웃는 얼굴인가, 우는 얼굴인가, 평범한 표정의 얼굴인가? 등)에 적용하실 수 있을 겁니다.

`load_data()` 함수는 입력으로 이미지가 있는 폴더 위치를 받습니다. 여기서는 rock_scissor_paper 폴더 위치를 적어주면 됩니다. 그리고, 숫자 손글씨는 0~9 까지의 클래스가 있었던 것 기억하시죠? 가위바위보의 경우 3개의 클래스 즉, **가위: 0, 바위: 1, 보: 2** 로 라벨링이 될 것입니다.

[Input]

```
def load_data(img_path):
    # 가위 : 0, 바위 : 1, 보 : 2
    number_of_data=300  # 가위바위보 이미지 개수 총합에 주의하세요.
    img_size=28
    color=3
    #이미지 데이터와 라벨(가위 : 0, 바위 : 1, 보 : 2) 데이터를 담을 행렬(matrix) 영역을 생
    imgs=np.zeros(number_of_data*img_size*img_size*color,dtype=np.int32).re
```

```
    labels=np.zeros(number_of_data,dtype=np.int32)

    idx=0
    for file in glob.iglob(img_path+'/scissor/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=0   # 가위 : 0
        idx=idx+1

    for file in glob.iglob(img_path+'/rock/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=1   # 바위 : 1
        idx=idx+1

    for file in glob.iglob(img_path+'/paper/*.jpg'):
        img = np.array(Image.open(file),dtype=np.int32)
        imgs[idx,:,:,:]=img    # 데이터 영역에 이미지 행렬을 복사
        labels[idx]=2   # 보 : 2
        idx=idx+1

    print("학습데이터(x_train)의 이미지 개수는",idx,"입니다.")
    return imgs, labels

image_dir_path = os.getenv("HOME") + "/aiffel/rock_scissor_paper"
(x_train, y_train)=load_data(image_dir_path)
x_train_norm = x_train/255.0   # 입력은 0~1 사이의 값으로 정규화

print("x_train shape: {}".format(x_train.shape))
print("y_train shape: {}".format(y_train.shape))
```

실행

[Output]

한 번 이미지를 불러 볼까요?

[Input]

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
print('라벨: ', y_train[0])
```

실행

[Output]

딥러닝 네트워크 설계하기

자 이제 데이터의 준비가 끝났습니다. 이제 여러분들이 가위바위보를 인식하는 딥러닝 네트워크를 설계해 볼까요?

[Input]

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

# model을 직접 만들어 보세요.
# Hint! model의 입력/출력부에 특히 유의해 주세요. 가위바위보 데이터셋은 MNIST 데
# [[YOUR CODE]]

model.summary()
```

실행

[Output]

딥러닝 네트워크 학습시키기

잘 설계가 되었다면, 이제 학습을 시켜봅시다. 아마도 여러분들의 데이터는 거의 비슷비슷할 것이기 때문에 accuracy가 꽤 높게 나올 것입니다.

> **[Input]**
>
> ```
> # model을 학습시키는 코드를 직접 작성해 보세요.
> # Hint! model.compile()과 model.fit()을 사용해 봅시다.
> # [[YOUR CODE]]
> ```
>
> 실행
>
> **[Output]**

얼마나 잘 만들었는지 확인하기(테스트)

여러분들은 300장의 가위바위보 이미지를 만들어 모두 학습에 사용했습니다. 그러므로 테스트 데이터가 없죠. 옆 친구의 이미지 데이터 300장을 받아오세요. 그리고 그것을 테스트 데이터로 하여 test accuracy를 측정해보세요.

우선 테스트용 데이터인 `x_test` , `y_test` 를 만들어 봅시다.

> **[Input]**
>
> ```
> # x_test, y_test를 만드는 방법은 x_train, y_train을 만드는 방법과 아주 유
> # [[YOUR CODE]]
> ```
>
> 실행
>
> **[Output]**

테스트용 데이터가 준비되었으니, 위에서 훈련시킨 model을 사용하여 test_accuracy를 측정해 봅시다.

[Input]

```
# model을 학습시키는 코드를 직접 작성해 보세요.
# Hint! model.evaluate()을 사용해 봅시다.
# [[YOUR CODE]]
```

실행

[Output]

더 좋은 네트워크 만들어보기

시험용 데이터(x_test)에 대한 인식률(test accuracy)이 train accuracy보다 많이 낮게 나오지는 않았나요?
만약 그렇다면 그 이유는 무엇일까요? MNIST 손글씨 데이터 때처럼 test accuracy가 train accuracy에 근접하도록 개선 방법을 찾아 봅시다.

노드를 마치며...

여러분 미니 프로젝트는 잘 마치셨나요? 여러분은 이번 노드를 통해 다음의 내용을 배웠습니다.

- 이미 잘 정제된 10개 클래스의 숫자 손글씨 데이터를 분류하는 classifier 만들기
- 정제되지 않은 웹캠 사진으로 부터 데이터 만들어보기
- 흑백 사진이 아닌 컬러 사진을 학습하는 classifier 만들기
- 분류하고자 하는 클래스의 개수를 마음대로 조절하기 (10개에서 3개로)

그러면 오늘 배운 내용을 바탕으로 마스크 쓴 사람과 안 쓴 사람을 구분하는 프로젝트도 금방 만드실 수 있겠죠? AIFFEL 입구에서 마스크 안 쓴 사람을 자동으로 감지하고 알람을 주는 시스템을 만들어 주실 용자분 계실까요?!!!

AIFFEL | **대표** 김승일 | **주소** 서울특별시 강남구 역삼로 156(역삼동) 4층, 모두의연구소

전화 070-7743-5882 | **이메일** support@aiffel.io | **개인정보보호책임자** 이지석