마스크 착용 형태 판단

김형섭 유다빈 윤한나 이성진

목 차

주제 선정

- 얼굴 인식 기기는 어떠한 원리로 사람의 얼굴을 인식할까?

- 얼굴 인식 기기의 인식률은 실제로 어느 정도일까?

- 얼굴 인식 기기는 사람들의 올바른 마스크 착용 여부까지 판단할 수 있을까?

코드 설명

원본 데이터로 실행한 모델

```
# 필요한 라이브러리 import
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import os
import matplotlib.pyplot as plt
import cv2
import sys
import os
```

Kaggle 원본 데이터 사용

- data : 16269 images
- epochs : 100

Accuracy: 96.02%

```
# 사진이 있는 디렉터리의 경로를 설정
dir = "C:/AI/project/dataset/FacemaskDetector/dataset" #원본 이미지 폴더
```

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_gen = ImageDataGenerator(rescale= 1/255.,           #0~1사이의 값으로 정규화
                               rotation_range=0.2,        #random
                               width_shift_range=0.2,     #좌우 이동
                               height_shift_range=0.2,    #상하 이동
                               zoom_range = 0.2,
                               horizontal_flip=True,
                               validation_split = 0.02)   #유효성 검사를 위한 데이터

test_gen = ImageDataGenerator(rescale= 1/255.,
                              validation_split = 0.2)     #train:test = 8:2
```

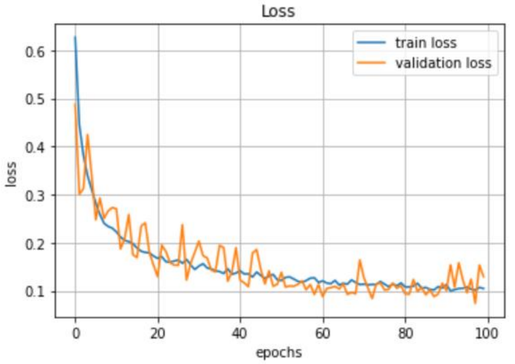
```
train_data = train_gen.flow_from_directory(dir,
                                           target_size = (224,224),
                                           class_mode = "categorical",
                                           seed = 42,
                                           subset = "training"
                                           )

test_data = test_gen.flow_from_directory(dir,
                                         target_size = (224,224),
                                         class_mode = "categorical",
                                         seed = 42,
                                         subset = "validation"
                                         )
```

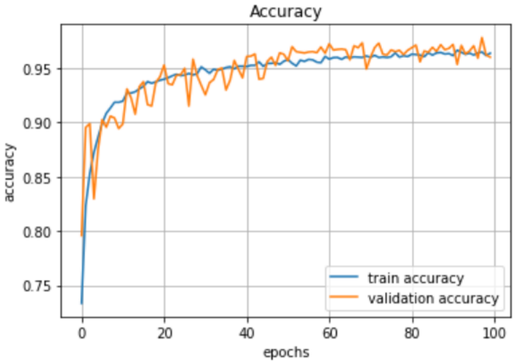
```
Found 16269 images belonging to 3 classes.
Found 3318 images belonging to 3 classes.
```

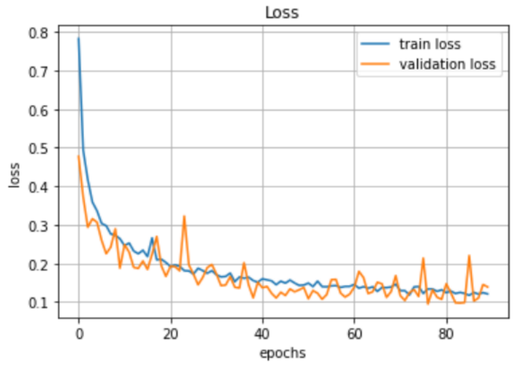
```
labels = list(train_data.class_indices.keys())

labels
```

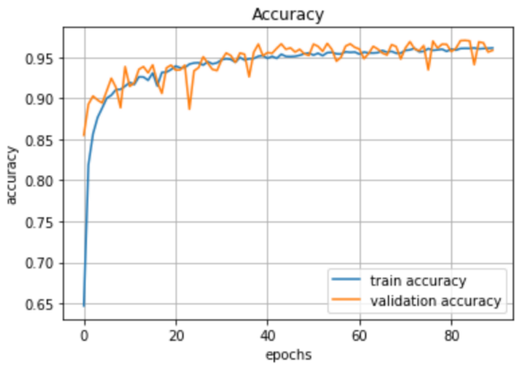
```
['incorrect_mask', 'with_mask', 'without_mask']
```

```
plt.figure(figsize = (16,16))

for i in range(25): #최대 25
    image,label = train_data.next()
```









```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import os
import matplotlib.pyplot as plt
import cv2
import sys
```

```
path_dir = "C:/Users/ICTCOC/Downloads/datasett/incorrect_mask/"
file_list = os.listdir(path_dir)
```

```
file_list[0:10]
```

```
['1.jpg',
 '10.jpg',
 '100.jpg',
 '1000.jpg',
 '1001.jpg',
 '1002.jpg',
 '1003.jpg',
 '1004.jpg',
 '1005.jpg',
 '1006.jpg']
```

```
len(file_list)
```

```
5687
```

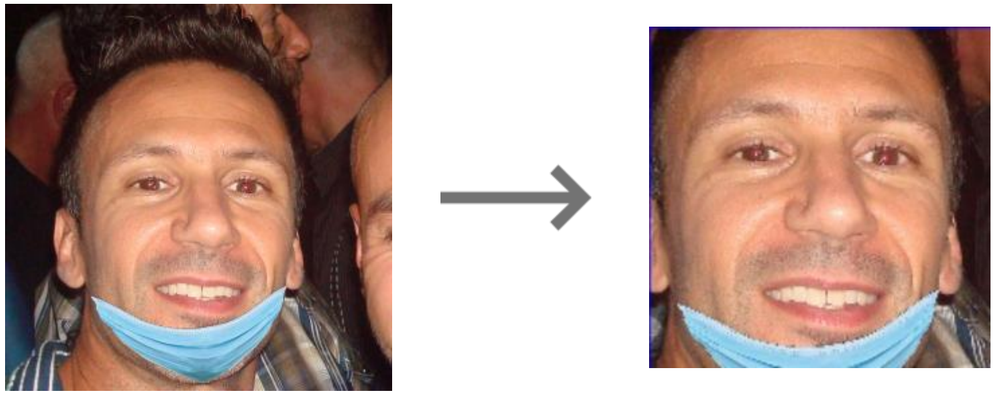
```
def Cutting_face_save(image, name):
    face_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_eye.xml')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 3)
    for (x,y,w,h) in faces:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]
        cropped = image[y: y+h, x: x+w]
        resize = cv2.resize(cropped, (180,180))
        eyes = eye_cascade.detectMultiScale(roi_gray)
        cv2.imwrite(f"C:/Users/ICTCOC/Downloads/datasett_cut/incorrect_mask/+{name}.jpg", resize)
```

```
for name in file_list:
    img = cv2.imread("C:/Users/ICTCOC/Downloads/datasett/incorrect_mask/"+name)
    Cutting_face_save(img, name)
    print(name)
```

```
1.jpg
10.jpg
100.jpg
1000.jpg
1001.jpg
1002.jpg
1003.jpg
1004.jpg
1005.jpg
1006.jpg
1007.jpg
1008.jpg
```

코드 설명

올바르지 않은 마스크 착용 이미지 파일 불러오기 및 크기 자르기



```
path_dir = "C:/Users/ICTCOC/Downloads/datasett/with_mask/"
file_list = os.listdir(path_dir)

def Cutting_face_save(image, name):
    face_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_eye.xml')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 3)
    for (x,y,w,h) in faces:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]
        cropped = image[y: y+h, x: x+w]
        resize = cv2.resize(cropped, (180,180))
        eyes = eye_cascade.detectMultiScale(roi_gray)
        cv2.imwrite(f"C:/Users/ICTCOC/Downloads/datasett_cut/with_mask/+{name}.jpg", resize)

for name in file_list:
    img = cv2.imread("C:/Users/ICTCOC/Downloads/datasett/with_mask/"+name)
    Cutting_face_save(img, name)
    print(name)
```

```
0-with-mask.jpg
1-with-mask.jpg
10-with-mask.jpg
100-with-mask.jpg
101-with-mask.jpg
103-with-mask.jpg
104-with-mask.jpg
105-with-mask.jpg
106-with-mask.jpg
107-with-mask.jpg
108-with-mask.jpg
109-with-mask.jpg
11-with-mask.jpg
110-with-mask.jpg
111-with-mask.jpg
112-with-mask.jpg
113-with-mask.jpg
114-with-mask.jpg
115-with-mask.jpg
```

```
path_dir = "C:/Users/ICTCOC/Downloads/datasett/without_mask/"
file_list = os.listdir(path_dir)

def Cutting_face_save(image, name):
    face_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier('C:/Users/ICTCOC/Downloads/haarcascade_eye.xml')
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 3)
    for (x,y,w,h) in faces:
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]
        cropped = image[y: y+h, x: x+w]
        resize = cv2.resize(cropped, (180,180))
        eyes = eye_cascade.detectMultiScale(roi_gray)
        cv2.imwrite(f"C:/Users/ICTCOC/Downloads/datasett_cut/without_mask/+{name}.jpg", resize)

for name in file_list:
    img = cv2.imread("C:/Users/ICTCOC/Downloads/datasett/without_mask/"+name)
    Cutting_face_save(img, name)
    print(name)
```

```
0.jpg
1.jpg
10.jpg
100.jpg
101.jpg
102.jpg
104.jpg
105.jpg
106.jpg
107.jpg
```

코드 설명

마스크 착용 & 미착용 이미지 파일 불러오기 및 크기 자르기

```
dir = "C:/Users/ICTCOC/Downloads/datasett_cut/"
```

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_gen = ImageDataGenerator(rescale= 1/255.,
                               rotation_range=0.2,
                               #width_shift_range=0.2,
                               #height_shift_range=0.2,
                               #zoom_range = 0.2,
                               horizontal_flip=True,
                               validation_split = 0.02)

test_gen = ImageDataGenerator(rescale= 1/255.,
                              validation_split = 0.2)
```

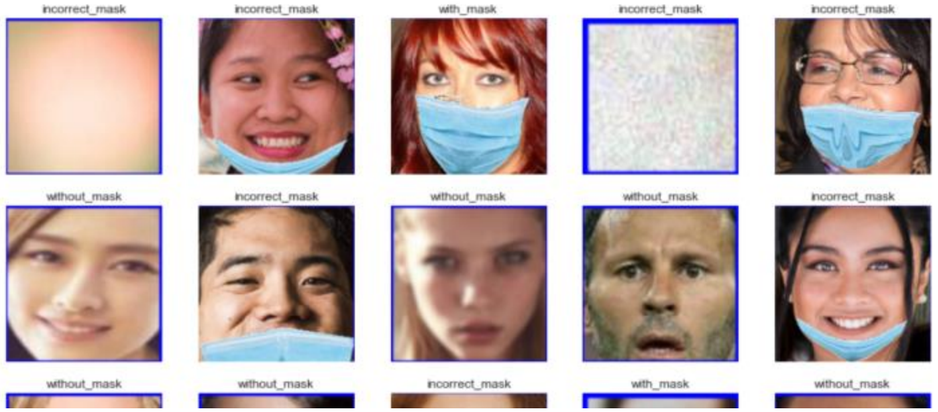
```
train_data = train_gen.flow_from_directory(dir,
                                           target_size = (180,180),
                                           class_mode = "categorical",
                                           seed = 42,
                                           subset = "training"
                                           )

test_data = test_gen.flow_from_directory(dir,
                                         target_size = (180,180),
                                         class_mode = "categorical",
                                         seed = 42,
                                         subset = "validation"
                                         )
```

```
Found 9456 images belonging to 3 classes.
Found 1927 images belonging to 3 classes.
```

```
labels = list(train_data.class_indices.keys())

labels
```

```
['incorrect_mask', 'with_mask', 'without_mask']
```

```
plt.figure(figsize = (16,16))

for i in range(25): #최대 25
    image,label = train_data.next()

    plt.subplot(5,5,i+1)
    plt.imshow(image[i])
    plt.title(labels[tf.argmax(label[i])])
    plt.axis("off")
```



코드 설명

이미지 파일 0~1 사이의 값으로 MinMax 정규화
학습셋 & 테스테셋 준비

```python
# Building a CNN model
import tensorflow as tf
from tensorflow.keras import layers
model  = tf.keras.Sequential([

                              layers.Conv2D(filters= 64, kernel_size= 2, activation="relu", input_shape=(180,180,3)),
                              layers.MaxPooling2D(pool_size= 2),

                              layers.Conv2D(filters = 64, kernel_size= 2, activation= "relu"),
                              layers.MaxPooling2D(pool_size= 2),

                              layers.Conv2D(filters = 64, kernel_size= 2, activation= "relu"),
                              layers.MaxPooling2D(pool_size= 2),

                              layers.Flatten(),

                              layers.Dense(128, activation="relu"),
                              layers.Dropout(0.5),

                              layers.Dense(3, activation= "softmax")
])
```

```python
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 179, 179, 64)      832
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 89, 89, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 88, 88, 64)        16448
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 44, 44, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 43, 43, 64)        16448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 21, 21, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 28224)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               3612800
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 3,646,915
Trainable params: 3,646,915
Non-trainable params: 0
_________________________________________________________________
```

```python
# compiling the model
model.compile(
    loss = tf.keras.losses.categorical_crossentropy,
    optimizer = tf.keras.optimizers.Adam(),
    metrics = ["accuracy"]
)
```

```python
# fitting data to the model
hist = model.fit(train_data,
          epochs = 10,
          steps_per_epoch = len(train_data),
          validation_data = test_data,
          validation_steps = len(test_data)
          )
```

```
Epoch 1/10
296/296 [==============================] - 107s 361ms/step - loss: 0.3298 - accuracy: 0.8813 - val_loss: 0.2418 - val_accuracy: 0.9
139
Epoch 2/10
296/296 [==============================] - 101s 341ms/step - loss: 0.2296 - accuracy: 0.9147 - val_loss: 0.2348 - val_accuracy: 0.9
196
Epoch 3/10
296/296 [==============================] - 101s 342ms/step - loss: 0.2097 - accuracy: 0.9214 - val_loss: 0.2017 - val_accuracy: 0.9
```

코드 설명

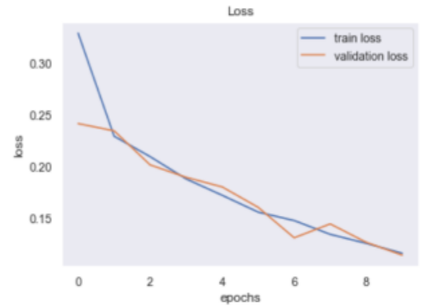
CNN 모델 구현
모델 학습

filter : 64, kernel size : 2, activation : 'relu'

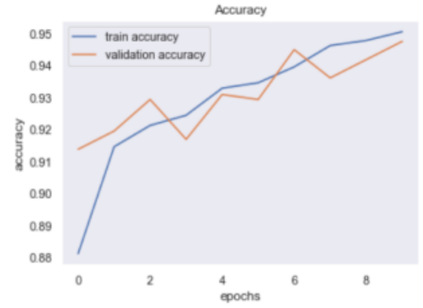
초기 시행

결과

```
import matplotlib.pyplot as plt

plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['loss'], label='train loss')
plt.plot(hist.history['val_loss'], label='validation loss')

plt.legend(loc='best')
plt.show()
```



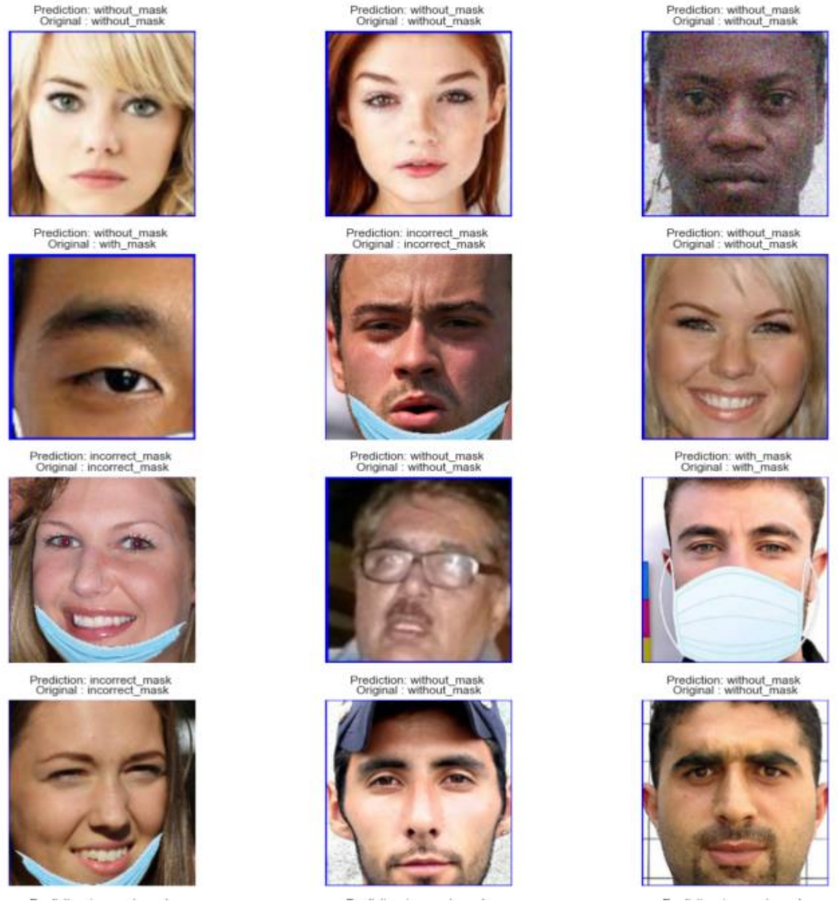

```
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(hist.history['accuracy'], label='train accuracy')
plt.plot(hist.history['val_accuracy'], label='validation accuracy')

plt.legend(loc='best')
plt.show()
```




```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 5s 76ms/step - loss: 0.1140 - accuracy: 0.9476
Model Accuracy:  94.76 %
```

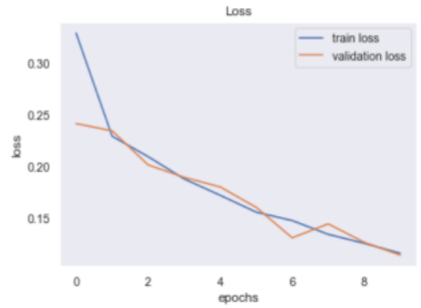
```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} \nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```



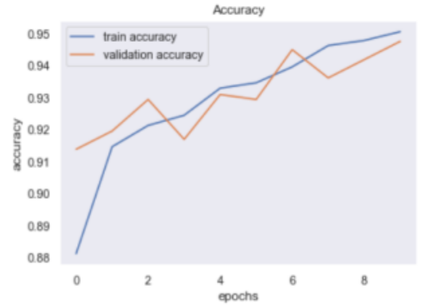
초기 시행

결과

```
import matplotlib.pyplot as plt

plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['loss'], label='train loss')
plt.plot(hist.history['val_loss'], label='validation loss')

plt.legend(loc='best')
plt.show()
```



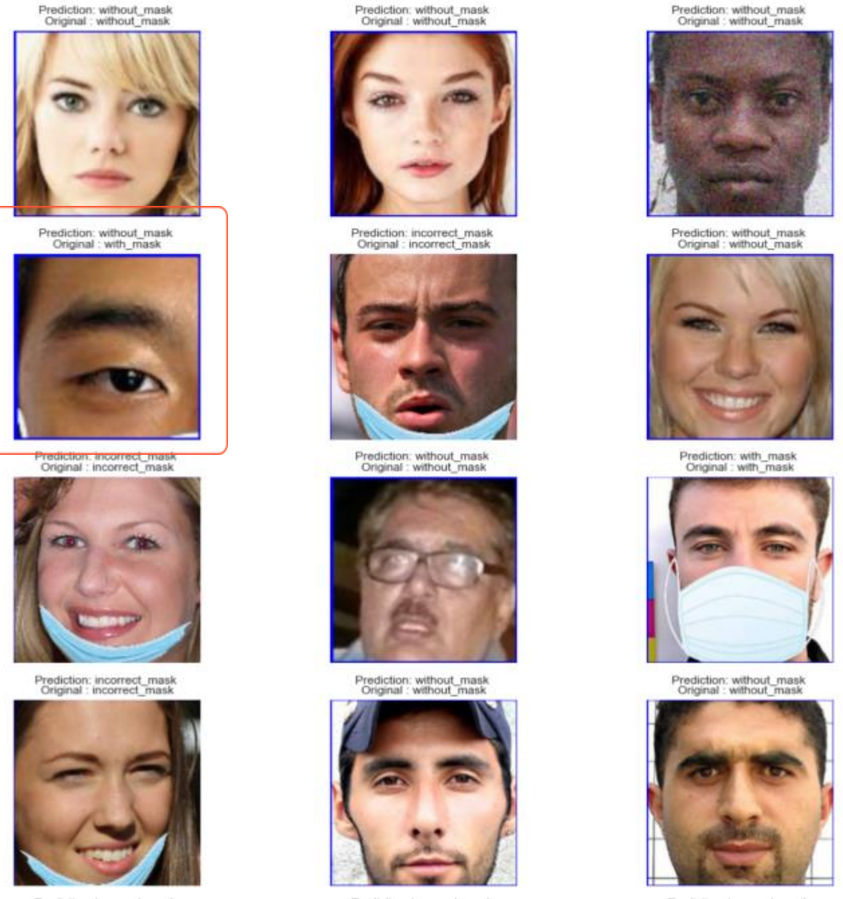

```
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(hist.history['accuracy'], label='train accuracy')
plt.plot(hist.history['val_accuracy'], label='validation accuracy')

plt.legend(loc='best')
plt.show()
```




```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 5s 76ms/step - loss: 0.1140 - accuracy: 0.9476
Model Accuracy:  94.76 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} \nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```



epoch 수 변경
01

hidden layer 변경
02

데이터 수정
03

다른 조건에서의 결과

How to enhance the accuracy of model ?

1 **epoch 10번 → epoch 100번**

초기 코드에서의 epoch 10을 100으로 실행

2 **은닉층 변경**

128, 64, 32, 16, 8

3 **데이터 수정 후 epoch 100번**

얼굴 데이터에서 이상치 제거

각 모델 조건에서의 결과

```
# 모델 학습 및 저장.
fit_history = model.fit(train_data,
                        epochs = 100,
                        steps_per_epoch = len(train_data),
                        validation_data = test_data,
                        validation_steps = len(test_data)
                        )
```

```
0.9964
Epoch 95/100
296/296 [==============================] - 135s 455ms/step - loss: 0.0244 - accuracy: 0.9890 - val_loss: 0.0157 - val_accuracy:
0.9964
Epoch 96/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0287 - accuracy: 0.9897 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 97/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0347 - accuracy: 0.9883 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 98/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0239 - accuracy: 0.9896 - val_loss: 0.0129 - val_accuracy:
0.9979
Epoch 99/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0177 - accuracy: 0.9926 - val_loss: 0.0153 - val_accuracy:
0.9958
Epoch 100/100
296/296 [==============================] - 134s 452ms/step - loss: 0.0312 - accuracy: 0.9903 - val_loss: 0.0141 - val_accuracy:
0.9979
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 7s 113ms/step - loss: 0.0141 - accuracy: 0.9979
Model Accuracy:  99.79 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} \nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```

1

epoch 10번 → epoch 100번

초기 코드에서의 epoch 10을 100으로 실행

각 모델 조건에서의 결과

```
# 모델 학습 및 저장
fit_history = model.fit(train_data,
                        epochs = 100,
                        steps_per_epoch = len(train_data),
                        validation_data = test_data,
                        validation_steps = len(test_data)
                        )
```

```
0.9964
Epoch 95/100
296/296 [==============================] - 135s 455ms/step - loss: 0.0244 - accuracy: 0.9890 - val_loss: 0.0157 - val_accuracy:
0.9964
Epoch 96/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0287 - accuracy: 0.9897 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 97/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0347 - accuracy: 0.9883 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 98/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0239 - accuracy: 0.9896 - val_loss: 0.0129 - val_accuracy:
0.9979
Epoch 99/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0177 - accuracy: 0.9926 - val_loss: 0.0153 - val_accuracy:
0.9958
Epoch 100/100
296/296 [==============================] - 134s 452ms/step - loss: 0.0312 - accuracy: 0.9903 - val_loss: 0.0141 - val_accuracy:
0.9979
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 7s 113ms/step - loss: 0.0141 - accuracy: 0.9979
Model Accuracy:  99.79 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} \nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```


```
# fitting data to the model

hist = model.fit(train_data,
          epochs = 100,
          steps_per_epoch = len(train_data),
          validation_data = test_data,
          validation_steps = len(test_data)
          )
```

```
0.9948
Epoch 95/100
296/296 [==============================] - 102s 346ms/step - loss: 0.1764 - accuracy: 0.9251 - val_loss: 0.0302 - val_accuracy:
0.9912
Epoch 96/100
296/296 [==============================] - 102s 345ms/step - loss: 0.1715 - accuracy: 0.9275 - val_loss: 0.0562 - val_accuracy:
0.9912
Epoch 97/100
296/296 [==============================] - 102s 345ms/step - loss: 0.1788 - accuracy: 0.9242 - val_loss: 0.0272 - val_accuracy:
0.9943
Epoch 98/100
296/296 [==============================] - 101s 341ms/step - loss: 0.1765 - accuracy: 0.9252 - val_loss: 0.0265 - val_accuracy:
0.9959
Epoch 99/100
296/296 [==============================] - 101s 341ms/step - loss: 0.1680 - accuracy: 0.9287 - val_loss: 0.0455 - val_accuracy:
0.9864
Epoch 100/100
296/296 [==============================] - 102s 346ms/step - loss: 0.2593 - accuracy: 0.9036 - val_loss: 0.0629 - val_accuracy:
0.9798
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 5s 75ms/step - loss: 0.0629 - accuracy: 0.9798
Model Accuracy:  97.98 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
  image, label = test_data.next()

  model_pred = model.predict(image)

  plt.subplot(6,3,i+1)
  plt.imshow(image[i])
  plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} \nOriginal : {labels[tf.argmax(label[i])]}")
  plt.subplots_adjust(top= 1.25)
  plt.axis("off")
```

1 **epoch 10번 → epoch 100번**

초기 코드에서의 epoch 10을 100으로 실행

2 **은닉층 변경**

128, 64, 32, 16, 8

각 모델 조건에서의 결과

```
# 모델 학습 및 저장.
fit_history = model.fit(train_data,
                        epochs = 100,
                        steps_per_epoch = len(train_data),
                        validation_data = test_data,
                        validation_steps = len(test_data)
                        )
```

```
0.9964
Epoch 95/100
296/296 [==============================] - 135s 455ms/step - loss: 0.0244 - accuracy: 0.9890 - val_loss: 0.0157 - val_accuracy:
0.9964
Epoch 96/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0287 - accuracy: 0.9897 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 97/100
296/296 [==============================] - 134s 454ms/step - loss: 0.0347 - accuracy: 0.9883 - val_loss: 0.0178 - val_accuracy:
0.9938
Epoch 98/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0239 - accuracy: 0.9896 - val_loss: 0.0129 - val_accuracy:
0.9979
Epoch 99/100
296/296 [==============================] - 134s 453ms/step - loss: 0.0177 - accuracy: 0.9926 - val_loss: 0.0153 - val_accuracy:
0.9958
Epoch 100/100
296/296 [==============================] - 134s 452ms/step - loss: 0.0312 - accuracy: 0.9903 - val_loss: 0.0141 - val_accuracy:
0.9979
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 7s 113ms/step - loss: 0.0141 - accuracy: 0.9979
Model Accuracy:  99.79 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} #nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```

1 **epoch 10번 → epoch 100번**

초기 코드에서의 epoch 10을 100으로 실행

```
# fitting data to the model

hist = model.fit(train_data,
          epochs = 100,
          steps_per_epoch = len(train_data),
          validation_data = test_data,
          validation_steps = len(test_data)
          )
```

```
0.9948
Epoch 95/100
296/296 [==============================] - 102s 346ms/step - loss: 0.1764 - accuracy: 0.9251 - val_loss: 0.0302 - val_accuracy:
0.9912
Epoch 96/100
296/296 [==============================] - 102s 345ms/step - loss: 0.1715 - accuracy: 0.9275 - val_loss: 0.0562 - val_accuracy:
0.9912
Epoch 97/100
296/296 [==============================] - 102s 345ms/step - loss: 0.1788 - accuracy: 0.9242 - val_loss: 0.0272 - val_accuracy:
0.9943
Epoch 98/100
296/296 [==============================] - 101s 341ms/step - loss: 0.1765 - accuracy: 0.9252 - val_loss: 0.0265 - val_accuracy:
0.9959
Epoch 99/100
296/296 [==============================] - 101s 341ms/step - loss: 0.1680 - accuracy: 0.9287 - val_loss: 0.0455 - val_accuracy:
0.9864
Epoch 100/100
296/296 [==============================] - 102s 346ms/step - loss: 0.2593 - accuracy: 0.9036 - val_loss: 0.0629 - val_accuracy:
0.9798
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
61/61 [==============================] - 5s 75ms/step - loss: 0.0629 - accuracy: 0.9798
Model Accuracy:  97.98 %
```

```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
  image, label = test_data.next()

  model_pred = model.predict(image)

  plt.subplot(6,3,i+1)
  plt.imshow(image[i])
  plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} #nOriginal : {labels[tf.argmax(label[i])]}")
  plt.subplots_adjust(top= 1.25)
  plt.axis("off")
```

2 **은닉층 변경**

128, 64, 32, 16, 8

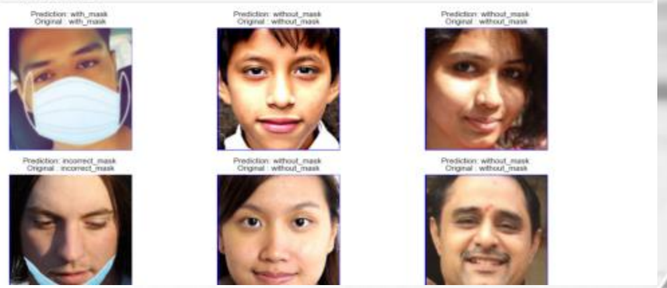
```
# 모델 학습 및 저장.
fit_history = model.fit(train_data,
                        epochs = 100,
                        steps_per_epoch = len(train_data),
                        validation_data = test_data,
                        validation_steps = len(test_data)
                        )
```

```
9972
Epoch 95/100
271/271 [==============================] - 94s 348ms/step - loss: 0.0076 - accuracy: 0.9980 - val_loss: 7.3940e-04 - val_accurac
y: 1.0000
Epoch 96/100
271/271 [==============================] - 93s 345ms/step - loss: 0.0036 - accuracy: 0.9990 - val_loss: 6.2295e-04 - val_accurac
y: 1.0000
Epoch 97/100
271/271 [==============================] - 94s 348ms/step - loss: 0.0038 - accuracy: 0.9987 - val_loss: 3.5209e-05 - val_accurac
y: 1.0000
Epoch 98/100
271/271 [==============================] - 94s 347ms/step - loss: 0.0064 - accuracy: 0.9980 - val_loss: 2.8949e-04 - val_accurac
y: 1.0000
Epoch 99/100
271/271 [==============================] - 94s 348ms/step - loss: 0.0078 - accuracy: 0.9978 - val_loss: 2.6989e-05 - val_accurac
y: 1.0000
Epoch 100/100
271/271 [==============================] - 95s 349ms/step - loss: 0.0110 - accuracy: 0.9963 - val_loss: 2.3399e-04 - val_accurac
y: 1.0000
```

```
model_evaluation = model.evaluate(test_data)
print(f"Model Accuracy: {model_evaluation[1] * 100 : 0.2f} %")
```

```
56/56 [==============================] - 4s 76ms/step - loss: 2.3399e-04 - accuracy: 1.0000
Model Accuracy:  100.00 %
```

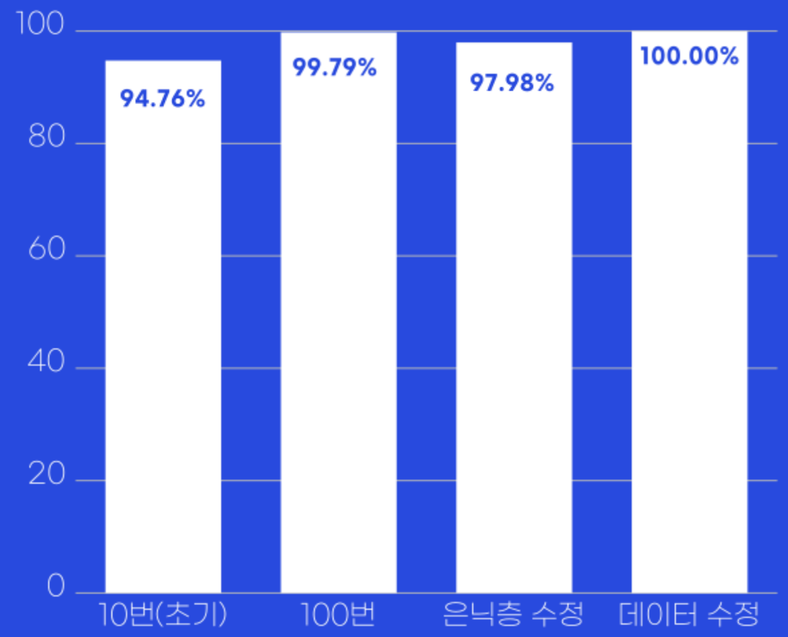
```
# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(16,16))

for i in range(18):
    image, label = test_data.next()

    model_pred = model.predict(image)

    plt.subplot(6,3,i+1)
    plt.imshow(image[i])
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[i])]} #nOriginal : {labels[tf.argmax(label[i])]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```



3 **데이터 수정 후 epoch 100번**

얼굴 데이터에서 이상치 제거

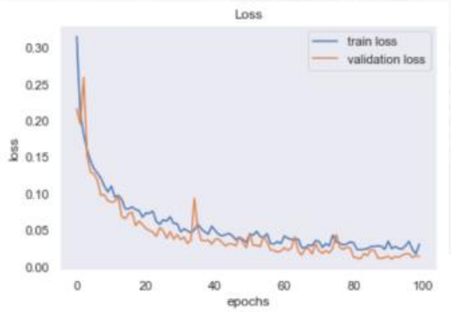
성능 비교

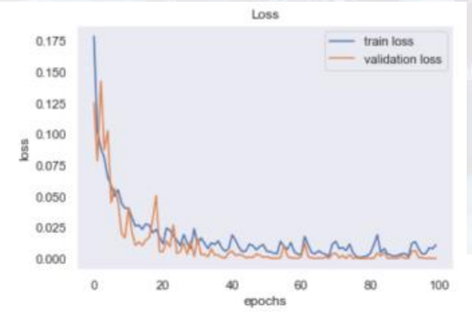


< 각 모델 조건에서의 성능 비교 >

원본 데이터에서의 결과와 수정된 데이터에서의 결과 비교

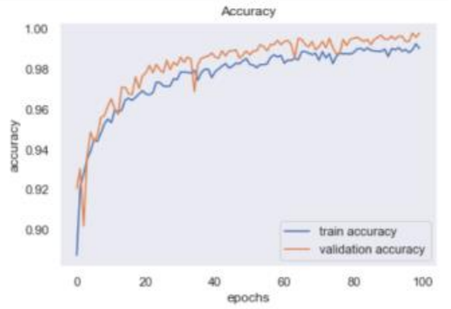
두 모델 모두 epoch 100번 실행

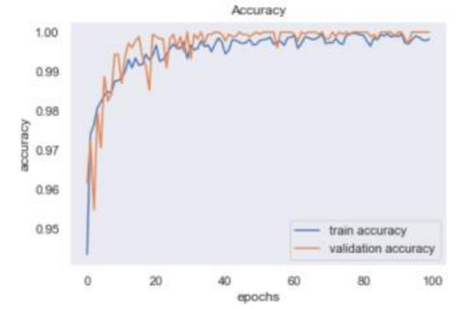




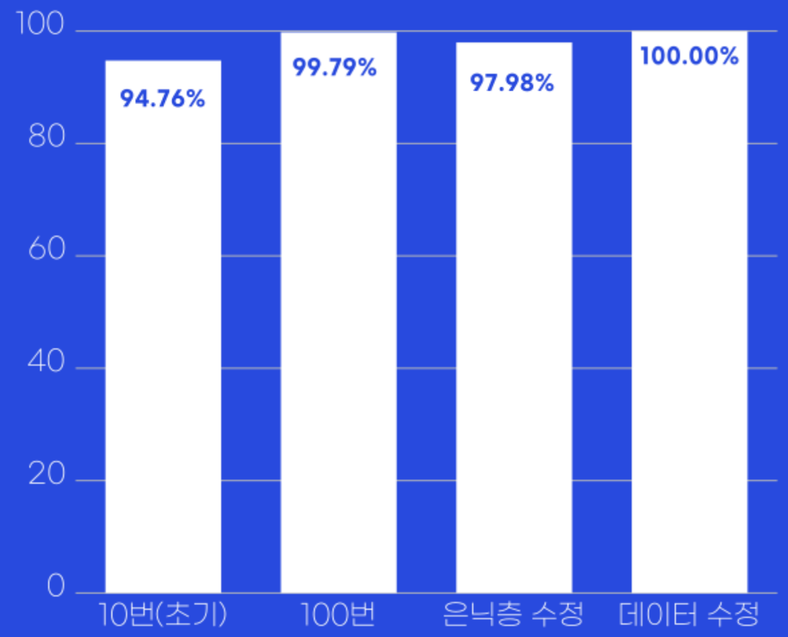

```
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(fit_history.history['accuracy'], label='train
plt.plot(fit_history.history['val_accuracy'], label='v

plt.legend(loc='best')
plt.show()
```

```
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(fit_history.history['accuracy'], label='train
plt.plot(fit_history.history['val_accuracy'], label='va

plt.legend(loc='best')
plt.show()
```





성능 비교



< 각 모델 조건에서의 성능 비교 >

원본 데이터에서의 결과와 수정된 데이터에서의 결과 비교

두 모델 모두 epoch 100번 실행

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 179, 179, 64)      832
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 89, 89, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 88, 88, 64)        16448
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 44, 44, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 43, 43, 64)        16448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 21, 21, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 28224)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               3612800
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 3,646,915
Trainable params: 3,646,915
Non-trainable params: 0
_________________________________________________________________
```

적용

```
path_dir = "C:/Ai/project/picture_cut/"
dile_list = os.listdir(path_dir)
label_list = [2,1,1,1]

# visualizing the test data

import matplotlib.pyplot as plt
import tensorflow as tf

plt.figure(figsize=(12,8))

for i in range(4):
    image, label = plt.imread(path_dir+dile_list[i]), label_list[i]
    image = cv2.resize(image, dsize=(180, 180))
    model_pred = model.predict(np.array([image]))

    plt.subplot(2,2,i+1)
    plt.imshow(image)
    plt.title(f"Prediction: {labels[tf.argmax(model_pred[0])]} \nOriginal : {labels[label]}")
    plt.subplots_adjust(top= 1.25)
    plt.axis("off")
```

executed in 698ms, finished 11:23:08 2022-08-05

Q & A

Thank You ^~^