컴퓨터 공학과 2017108258 김유석

마스크 착용 감지

CNN 모델을 활용한 마스크 착용감지

개요 / 필요성

주제 선정이유, 필요성

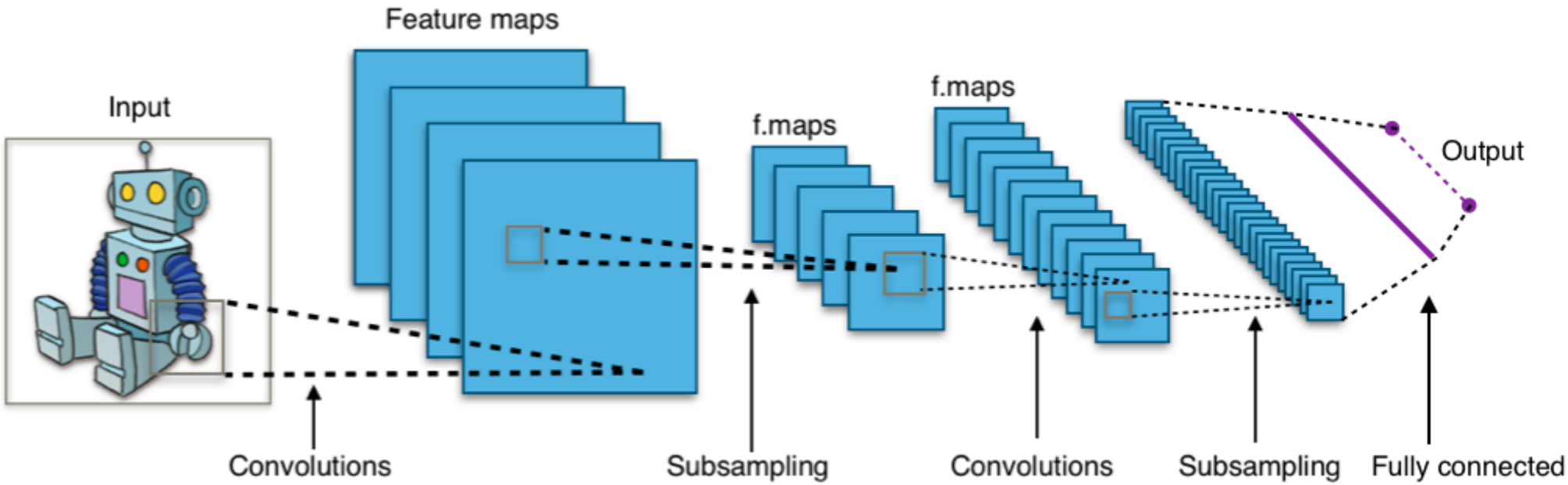
코로나 이전에도, 위생, 공중보건, 전염/감염 예방이 필요한 상황에서 마스크는 언제나 필요해 왔다. 이러한 환경에서 마스크 착용감지 AI를 활용하면 보다 안전하고 실용적인 위생, 보건환경을 유지하는데 도움이 될 수 있다.

관련 연구 / 내용

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 개요

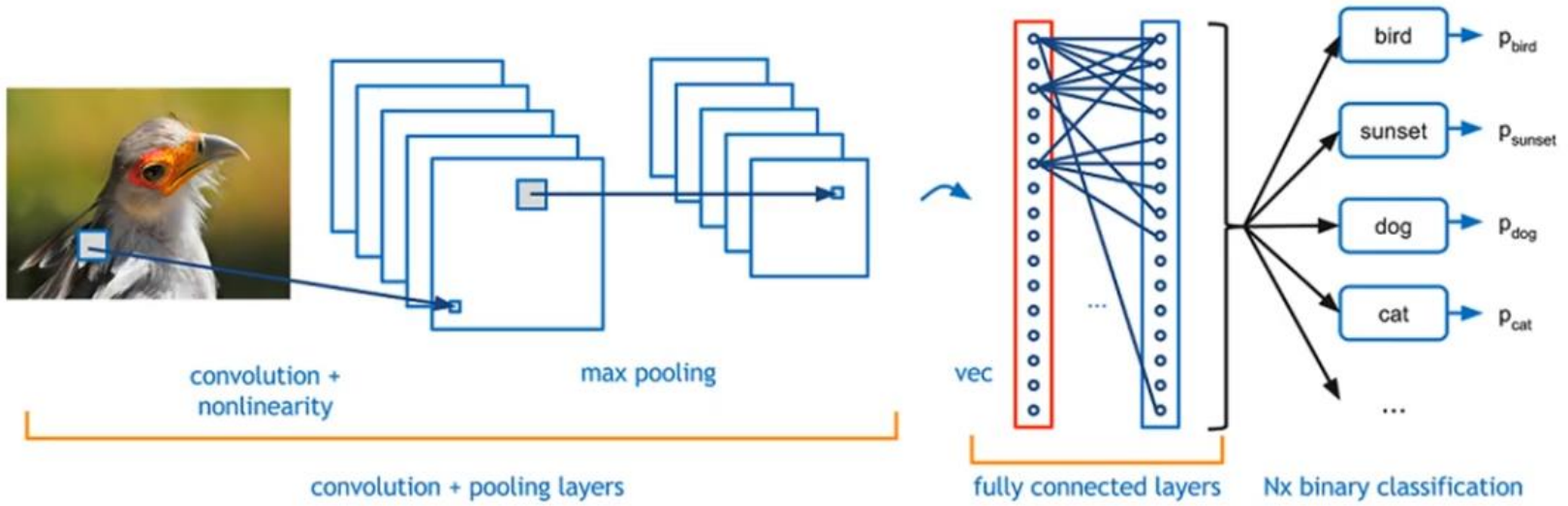


인간의 시신경 구조를 모방해 만들어진 인공신경망 알고리즘, 이를 이용하여 마스크 착용 감지를 구현 할 것이다

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

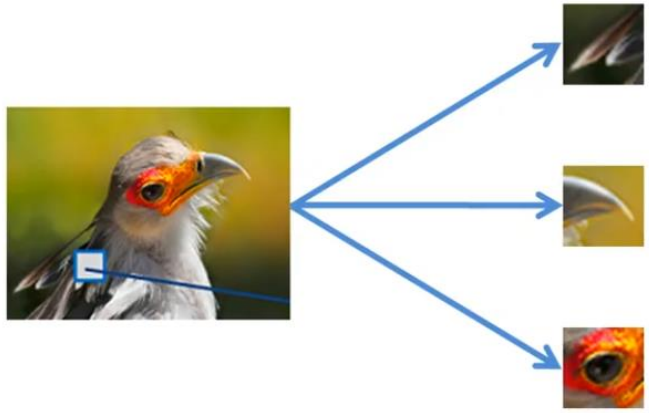


사진이 입력으로 들어오면 특징을 추출

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

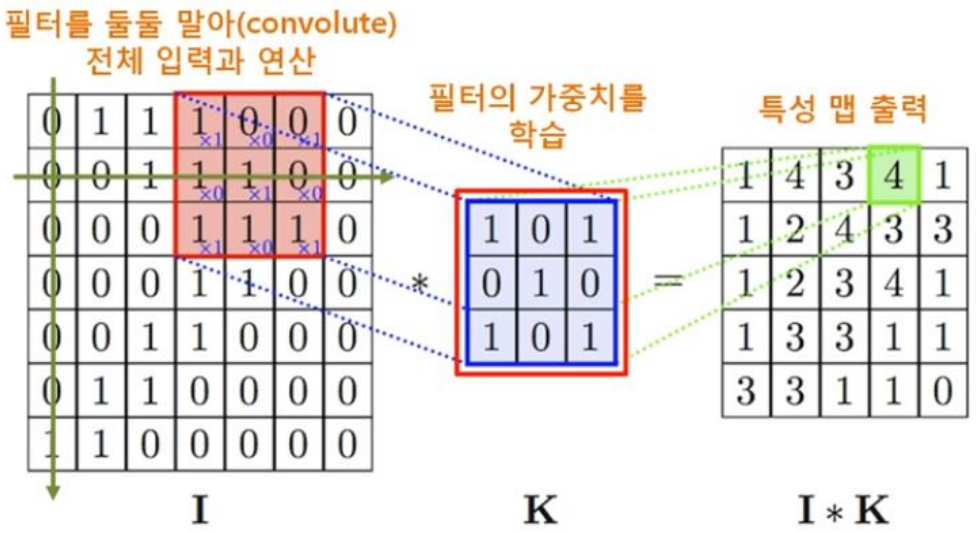


예시의 사진에 경우엔 눈의 모양, 부리, 깃털 등 특징별로 나누어 파악 이 과정을 컨볼루션(convolution)이라고 함

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

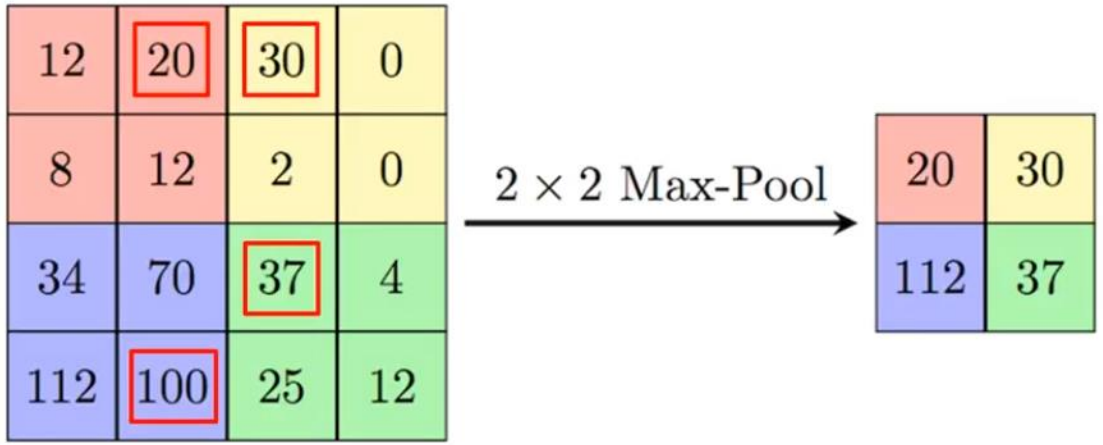


풀링 : 이때 특징은 정사각형 모양의 필터 형식으로 추출함, 마지막으로 나온 특성맵을 통해 이미지를 판별함

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

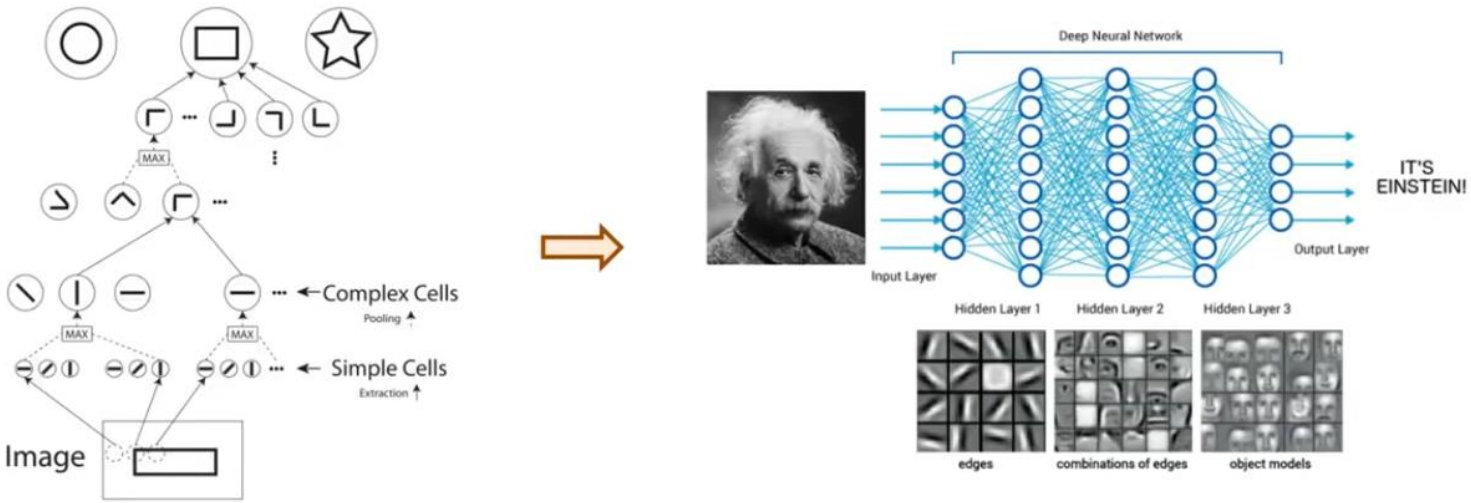


맥스 풀링 : 컨볼루션 이후 폴링을 거쳐 완성된 특성맵에서 값이 높은 가중치만 구역별로 판별해 특성맵 크기를 줄이는 과정

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

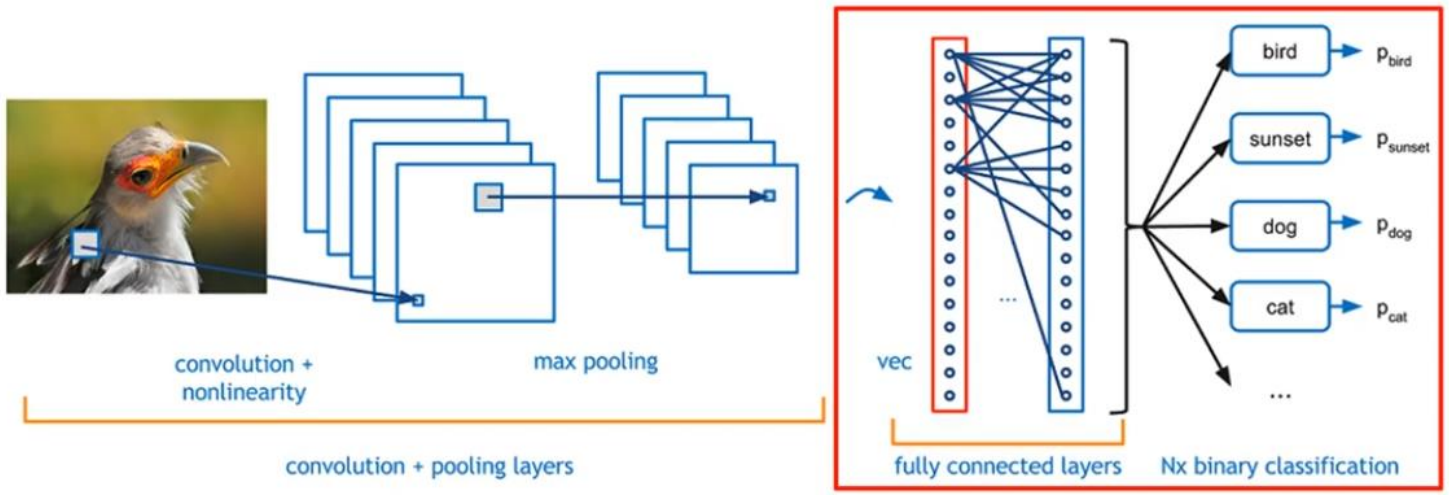


이처럼 CNN모델은 인간의 시각피질처럼 낮은 층에선 간단한 패턴, 높은 층으로 갈수록 복잡한 패턴으로 추상화 함

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식

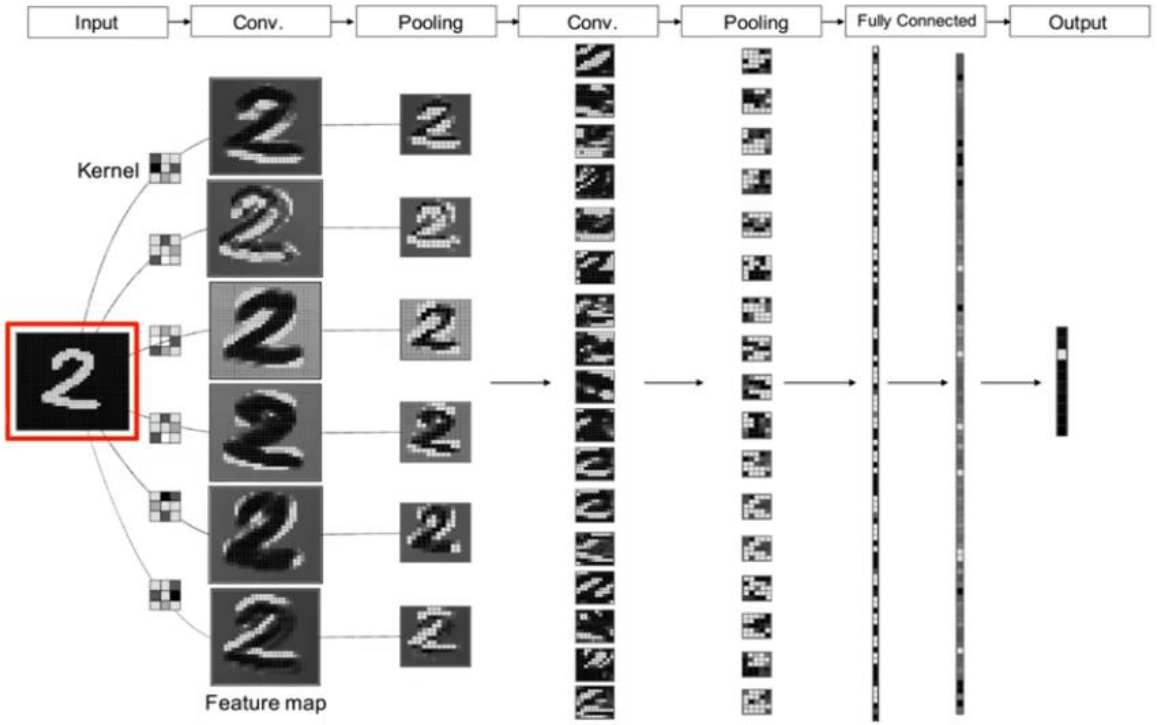


최종적으로 맥스풀링을 마치고 출력된 특성데이터(예시에서 부리, 눈, 깃털 등)를 완전연결 인공신경망과 붙여 클래스를 부여함

관련 연구 / 내용

마스크 착용 감지에 사용된 기술, 내용

CNN 모델 동작 방식



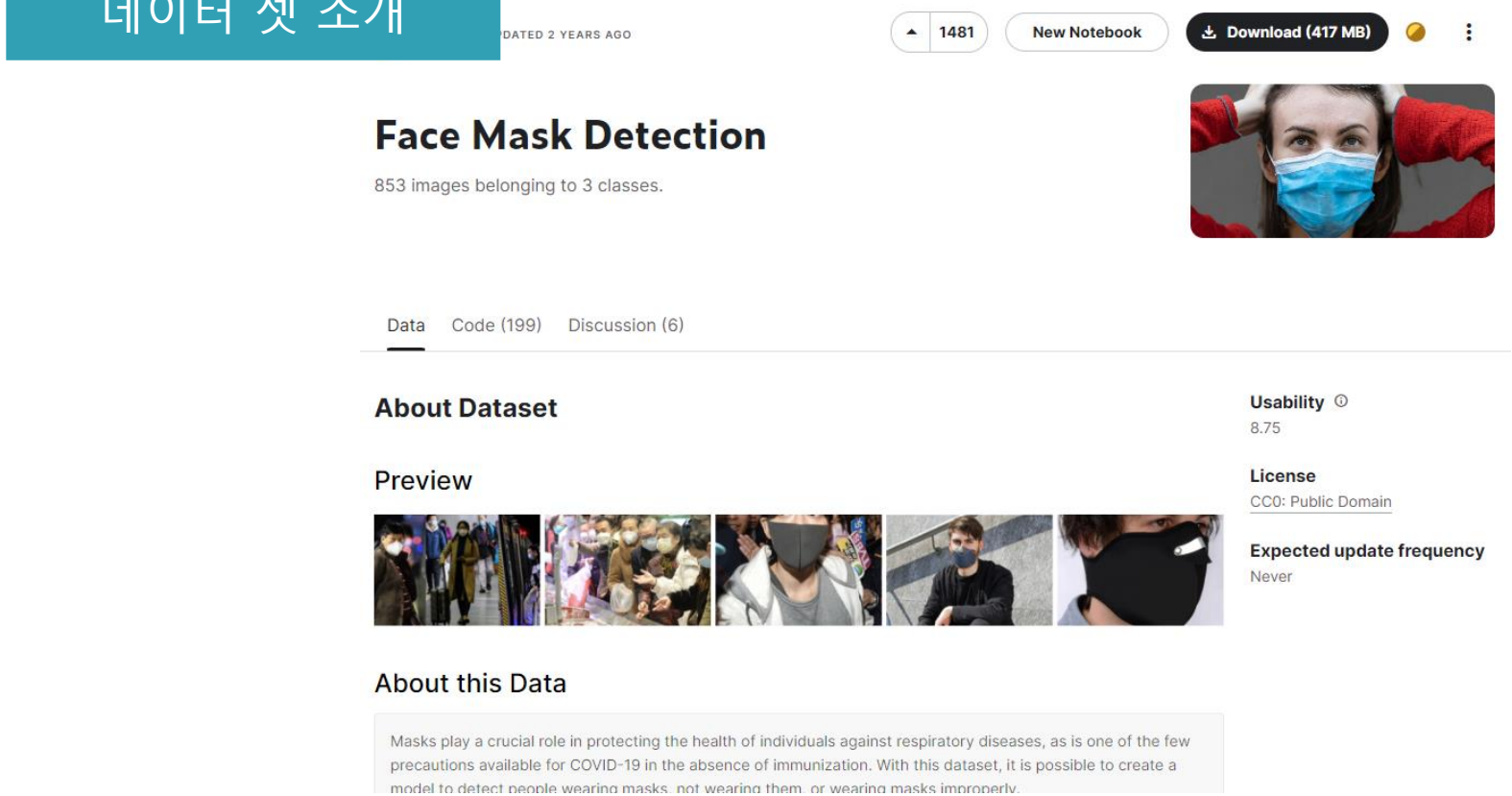
2를 입력받았을때 이를 컨볼루션, 풀링을 통해 패턴을 추출, 이 패턴들을 맥스풀링을 통해 컨볼루션, 풀링과정을 추가로 거쳐서 마지막에서 출력되는 특성맵의 데이터를 일렬로 붙여, 완전연결 신경망을 통해 최종 클래스를 분류함

마스크 착용 감지 내용

데이터 셋 소개

사용한 데이터 셋 소개

데이터 셋 소개



https://www.kaggle.com/datasets/andrewmvd/face-mask-detection

데이터 셋 소개

사용한 데이터 셋 소개

데이터 셋 구조

maksssksksss0.png (329.9 kB)

maksssksksss0.xml (1.25 kB)

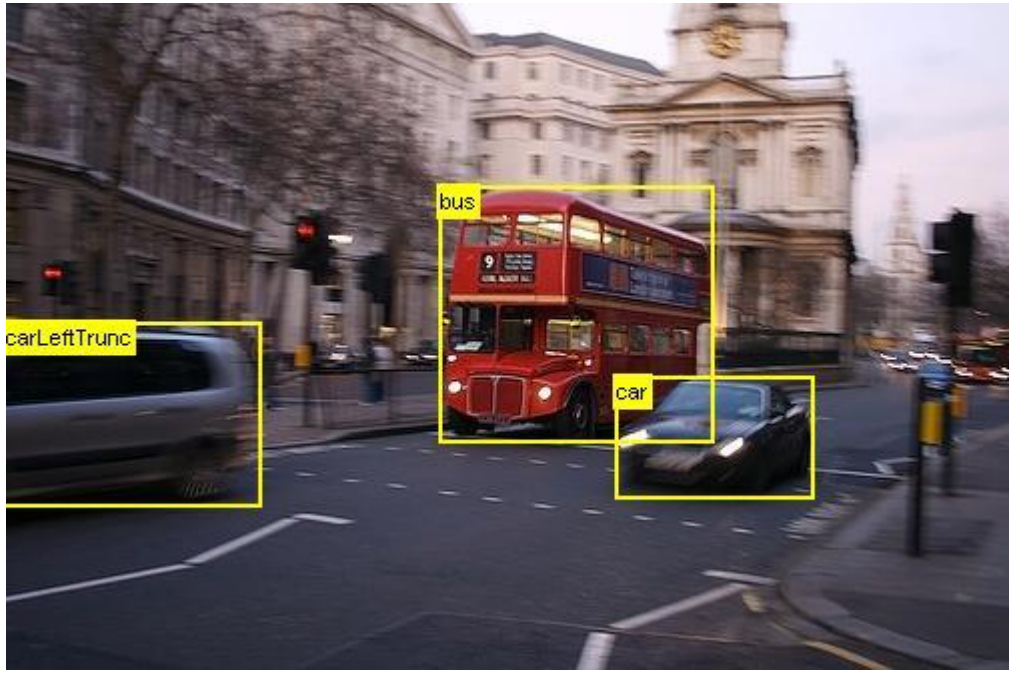
```
<annotation>
    <folder>images</folder>
    <filename>maksssksksss0.png</filename>
    <size>
        <width>512</width>
        <height>366</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>without_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>79</xmin>
            <ymin>105</ymin>
            <xmax>109</xmax>
            <ymax>142</ymax>
        </bndbox>
    </object>
    <object>
        <name>with_mask</name>
```

마스크 착용, 미착용 등의 이미지와 그 이미지에 대한 PASCAL VOC 정보등이 담긴 XML 파일이 동봉 되있는 구조

데이터 셋 소개

사용한 데이터 셋 소개

데이터 셋 구조



PASCAL VOC : 사진의 내용 일부를 클래스를 분류하여 예시와 같이 이미지 좌표에 태그를 붙여둔것

코드 내용

코드 내용

라이브러리 임포트

라이브러리 소개

- 이미지처리
  - CV2
  - PIL
- 연산, 시각화 처리
  - Numpy
  - Metplotlib
- 데이터셋, 환경변수 관리
  - Pandas
  - Xml.etree.ElementTree
  - Glob
  - Os
- 학습데이터 처리
  - Sklearn
  - Keras
  - tensorflow

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import xml.etree.ElementTree as ET
import glob
from PIL import Image
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, accuracy_score,f1_score, classification_report
import cv2
import seaborn as sns


input_data_path = '/kaggle/input/face-mask-detection/images'
annotations_path = "/kaggle/input/face-mask-detection/annotations"
images = [*os.listdir("/kaggle/input/face-mask-detection/images")]
output_data_path =  '.'
```

코드 내용

데이터셋 전처리

전처리 과정

```python
def parse_annotation_object(annotation_object):
    params = {}
    for param in list(annotation_object):
        if param.tag == 'name':
            params['name'] = param.text
        if param.tag == 'bndbox':
            for coord in list(param):
                if coord.tag == 'xmin':
                    params['xmin'] = int(coord.text)
                if coord.tag == 'ymin':
                    params['ymin'] = int(coord.text)
                if coord.tag == 'xmax':
                    params['xmax'] = int(coord.text)
                if coord.tag == 'ymax':
                    params['ymax'] = int(coord.text)
```

```python
def merge(dict1, dict2):
    res = {**dict1, **dict2}
    return res
```

```python
def parse_annotation(path):
    tree = ET.parse(path)
    root = tree.getroot()
    constants = {}
    objects = [child for child in root if child.tag == 'object']
    for element in tree.iter():
        if element.tag == 'filename':
            constants['file'] = element.text[0:-4]
        if element.tag == 'size':
            for dim in list(element):
                if dim.tag == 'width':
                    constants['width'] = int(dim.text)
                if dim.tag == 'height':
                    constants['height'] = int(dim.text)
                if dim.tag == 'depth':
                    constants['depth'] = int(dim.text)
    object_params = [parse_annotation_object(obj) for obj in objects]
    #print(constants)
    full_result = [merge(constants,ob) for ob in object_params]
    return full_result
```

- XML 데이터 전처리

코드 내용

데이터 셋 전처리

전처리 결과

```
dataset = [parse_annotation(anno) for anno in glob.glob(annotations_path+"/*.xml") ]

full_dataset = sum(dataset, [])

df = pd.DataFrame(full_dataset)
df.shape
```

```
df.head()
```

	file	width	height	depth	name	xmin	ymin	xmax	ymax
0	maksssksksss737	400	226	3	with_mask	28	55	46	71
1	maksssksksss737	400	226	3	with_mask	98	62	111	78
2	maksssksksss737	400	226	3	mask_weared_incorrect	159	50	193	90
3	maksssksksss737	400	226	3	with_mask	293	59	313	80
4	maksssksksss737	400	226	3	with_mask	352	51	372	72

○ 전처리 결과

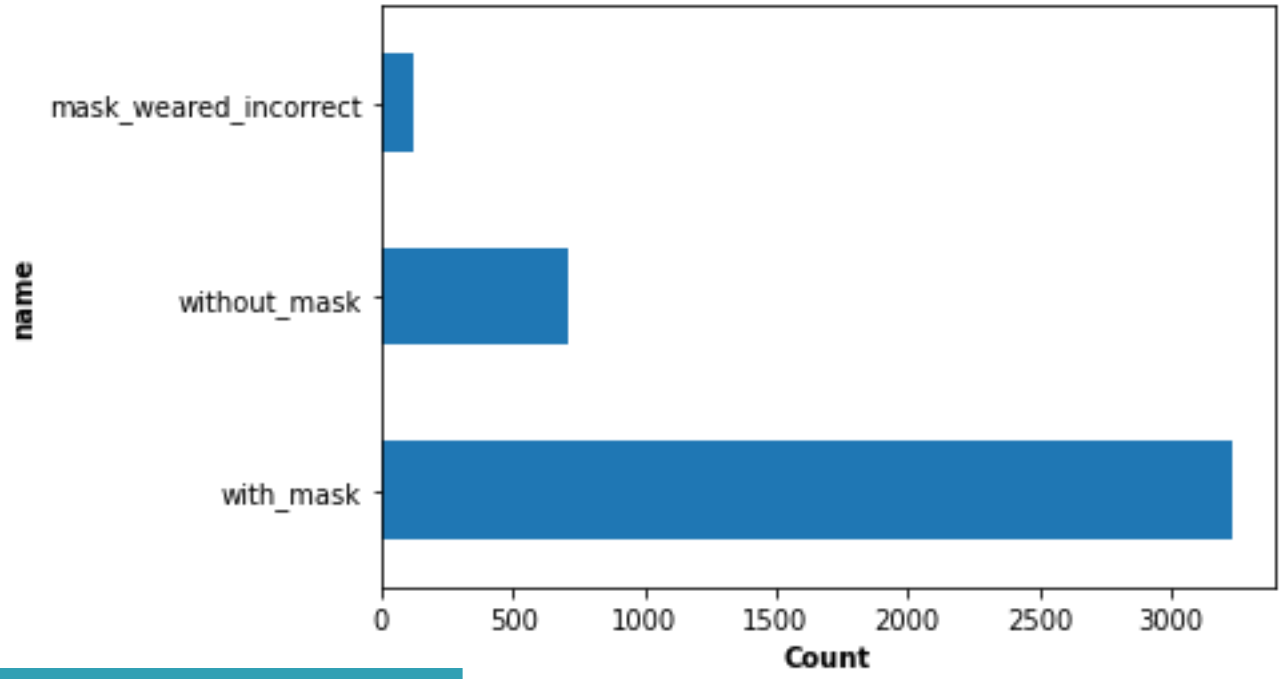
코드 내용

전처리 데이터 시각화

```
[7]: df["name"].value_counts()
```

```
[7]: with_mask                3231
     without_mask              715
     mask_weared_incorrect     123
     Name: name, dtype: int64
```

```
df["name"].value_counts().plot(kind='barh')
plt.xlabel('Count', fontsize = 10, fontweight = 'bold')
plt.ylabel('name', fontsize = 10, fontweight = 'bold')
```



마스크 착용 방식에 따른 전처리된 데이터 갯수를 시각화

코드 내용

학습 시작전 데이터 전처리

추가 데이터 전처리

```
labels = df['name'].unique()
directory = ['train', 'test', 'val']
output_data_path =  '.'

import os
for label in labels:
    for d in directory:
        path = os.path.join(output_data_path, d, label)
        if not os.path.exists(path):
            os.makedirs(path)
```

○ 결과 저장경로 설정

```
def crop_img(image_path, x_min, y_min, x_max, y_max):
    x_shift = (x_max - x_min) * 0.1
    y_shift = (y_max - y_min) * 0.1
    img = Image.open(image_path)
    cropped = img.crop((x_min - x_shift, y_min - y_shift, x_max + x_shift, y_max + y_shift))
    return cropped
```

○ 이미지 자르기

코드 내용

학습 진행전 데이터 전처리

추가 데이터 전처리

[11]:
```python
def extract_faces(image_name, image_info):
    faces = []
    df_one_img = image_info[image_info['file'] == image_name[:-4]][['xmin', 'ymin', 'xmax', 'ymax', 'name']]
    for row_num in range(len(df_one_img)):
        x_min, y_min, x_max, y_max, label = df_one_img.iloc[row_num]
        image_path = os.path.join(input_data_path, image_name)
        faces.append((crop_img(image_path, x_min, y_min, x_max, y_max), label,f'{image_name[:-4]}_{(x_min, y_min)}'))
    return faces
```

[12]:
```python
cropped_faces = [extract_faces(img, df) for img in images]
```

[13]:
```python
flat_cropped_faces = sum(cropped_faces, [])
```

[14]:
```python
with_mask = [(img, image_name) for img, label,image_name in flat_cropped_faces if label == "with_mask"]
mask_weared_incorrect = [(img, image_name) for img, label,image_name in flat_cropped_faces if label == "mask_weared_incorrect"]
without_mask = [(img, image_name) for img, label,image_name in flat_cropped_faces if label == "without_mask"]
```

○ PASCAL VOC 데이터 기준, 얼굴 이미지 추출, 전처리된 데이터 변수에 정의

코드 내용

데이터 교차 검증 진행

학습 시작

```
train_with_mask, test_with_mask = train_test_split(with_mask, test_size=0.20, random_state=42)
test_with_mask, val_with_mask = train_test_split(test_with_mask, test_size=0.7, random_state=42)

train_mask_weared_incorrect, test_mask_weared_incorrect = train_test_split(mask_weared_incorrect, test_size=0.20, random_state=42)
test_mask_weared_incorrect, val_mask_weared_incorrect = train_test_split(test_mask_weared_incorrect, test_size=0.7, random_state=42)

train_without_mask, test_without_mask = train_test_split(without_mask, test_size=0.20, random_state=42)
test_without_mask, val_without_mask = train_test_split(test_without_mask, test_size=0.7, random_state=42)
```

- 마스크 착용, 착용 안함, 옳지 않게 착용함 3가지로 학습을 진행
- Train_test_split() : 교차검증
- test_size : 테스트 데이터셋 비율
- Random_state : 데이터 분할 랜덤 시드

Overfitting 방지를 위해 각 패턴에서 train(0.7), test(0.2) 두가지로 나누어 학습을 진행

추가 개념 내용

과적합(Overfitting) 이란?

Overfitting(과적합)

Overfitting(과적합) : 모델이 학습데이터에 최적화 되어서 약간의 변동이 발생하는 새로운데이터에 대응이 어려운 상태

코드 내용

CNN 컨볼루션

CNN 모델 생성

○ 생성된 패턴 예시 (8*8)

```python
def save_image(image, image_name, output_data_path,  dataset_type, label):
    output_path = os.path.join(output_data_path, dataset_type, label ,f'{image_name}.png')
    image.save(output_path)
```

```python
for image, image_name in train_with_mask:
    save_image(image, image_name, output_data_path, 'train', 'with_mask')

for image, image_name in train_mask_weared_incorrect:
    save_image(image, image_name, output_data_path, 'train', 'mask_weared_incorrect')

for image, image_name in train_without_mask:
    save_image(image, image_name, output_data_path, 'train', 'without_mask')

for image, image_name in test_with_mask:
    save_image(image, image_name, output_data_path, 'test', 'with_mask')

for image, image_name in test_mask_weared_incorrect:
    save_image(image, image_name, output_data_path, 'test', 'mask_weared_incorrect')

for image, image_name in test_without_mask:
    save_image(image, image_name, output_data_path, 'test', 'without_mask')

for image, image_name in val_with_mask:
    save_image(image, image_name, output_data_path, 'val', 'with_mask')

for image, image_name in val_without_mask:
    save_image(image, image_name, output_data_path, 'val', 'without_mask')

for image, image_name in val_mask_weared_incorrect:
    save_image(image, image_name, output_data_path, 'val', 'mask_weared_incorrect')
```

교차 검증으로 학습된 이미지 패턴을 저장

코드 내용

CNN 맥스풀링 진행

CNN 모델 생성

```
model = Sequential()
model.add(Conv2D(filters = 16, kernel_size = 3,  padding='same', activation = 'relu', input_shape = (35,35,3)))
model.add(MaxPooling2D(pool_size = 2))
model.add(Conv2D(filters = 32, kernel_size = 3,  padding='same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = 2))
model.add(Conv2D(filters = 64, kernel_size = 3,  padding='same', activation = 'relu'))
model.add(MaxPooling2D(pool_size = 2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(units = 500, activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 3, activation = 'softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 35, 35, 16)        448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 17, 17, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               512500
_________________________________________________________________
dropout_3 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 1503
=================================================================
Total params: 537,587
Trainable params: 537,587
```

출력된 여러패턴을 기반, 순차모델 학습을 통한 맥스풀링을 진행

코드 내용

완전 연결인공 신경망 생성, 클래스분류

CNN 모델 생성

```
batch_size = 8
epochs = 50

datagen = ImageDataGenerator(
    rescale=1.0 / 255, horizontal_flip=True, zoom_range=0.1, shear_range=0.2, width_shift_range=0.1,
    height_shift_range=0.1, rotation_range=4, vertical_flip=False

)

val_datagen = ImageDataGenerator(
    rescale=1.0 / 255
)


train_generator = datagen.flow_from_directory(
    directory='/kaggle/working/train',
    target_size = (35,35),
    class_mode="categorical", batch_size=batch_size, shuffle=True

)

# Validation data
val_generator = val_datagen.flow_from_directory(
    directory='/kaggle/working/val',
    target_size = (35,35),
    class_mode="categorical", batch_size=batch_size, shuffle=True
)

# Test data
test_generator = val_datagen.flow_from_directory(
    directory='/kaggle/working/test',
    target_size = (35,35),
    class_mode="categorical", batch_size=batch_size, shuffle=False
)
```

완전연결 인공 신경망 생성, 맥스풀링을 마친 패턴 클래스 분류

코드 내용

최종 모델 생성전 전처리

CNN 모델 생성

```
data_size = len(train_generator)

steps_per_epoch = int(data_size / batch_size)
print(f"steps_per_epoch: {steps_per_epoch}")

val_steps = int(len(val_generator) // batch_size)
print(f"val_steps: {val_steps}")
```

```
steps_per_epoch: 50
val_steps: 9
```

```
model.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=['accuracy', 'Recall', 'Precision', 'AUC']
)
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)
```

```
lrr = ReduceLROnPlateau(monitor='val_loss',patience=8,verbose=1,factor=0.5, min_lr=0.00001)
```

최종 모델 생성전 인자값 생성, 처리

코드 내용

최종 모델 생성

CNN 모델 생성

```
model_history = model.fit_generator(
    generator=train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    shuffle=True,
    validation_data=val_generator,
    validation_steps=val_steps,
    callbacks=[early_stopping, lrr]
)
```

```
Epoch 1/50
 4/50 [=>............................] - ETA: 0s - loss: 0.2333 - accuracy: 0.9375 - recall: 0.9375 - precision: 0.9375 - auc: 0.9795
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:1972: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
50/50 [==============================] - 1s 25ms/step - loss: 0.1842 - accuracy: 0.9450 - recall: 0.9425 - precision: 0.9496 - auc: 0.9895 - val_loss: 0.3926 - val_accuracy: 0.8611 - v
al_recall: 0.8611 - val_precision: 0.8611 - val_auc: 0.9694
Epoch 2/50
50/50 [==============================] - 1s 22ms/step - loss: 0.2242 - accuracy: 0.9275 - recall: 0.9175 - precision: 0.9362 - auc: 0.9833 - val_loss: 0.1665 - val_accuracy: 0.9444 - v
al_recall: 0.9444 - val_precision: 0.9577 - val_auc: 0.9930
Epoch 3/50
50/50 [==============================] - 1s 22ms/step - loss: 0.2128 - accuracy: 0.9425 - recall: 0.9375 - precision: 0.9494 - auc: 0.9849 - val_loss: 0.1686 - val_accuracy: 0.9306 - v
al_recall: 0.9306 - val_precision: 0.9571 - val_auc: 0.9939
Epoch 4/50
50/50 [==============================] - 1s 24ms/step - loss: 0.2150 - accuracy: 0.9350 - recall: 0.9350 - precision: 0.9373 - auc: 0.9848 - val_loss: 0.2158 - val_accuracy: 0.9444 - v
al_recall: 0.9306 - val_precision: 0.9571 - val_auc: 0.9838
Epoch 5/50
50/50 [==============================] - 1s 21ms/step - loss: 0.1635 - accuracy: 0.9450 - recall: 0.9425 - precision: 0.9544 - auc: 0.9922 - val_loss: 0.1556 - val_accuracy: 0.9444 - v
al_recall: 0.9444 - val_precision: 0.9577 - val_auc: 0.9944
Epoch 6/50
50/50 [==============================] - 1s 21ms/step - loss: 0.1982 - accuracy: 0.9397 - recall: 0.9397 - precision: 0.9444 - auc: 0.9865 - val_loss: 0.1219 - val_accuracy: 0.9444 - v
al_recall: 0.9444 - val_precision: 0.9577 - val_auc: 0.9982
```

최종 모델 생성

코드 내용

최종 모델 검증

CNN 모델 검증

```
model_loss, model_acc, recall, precision, auc = model.evaluate(test_generator)
print(f'{model_loss} || {model_acc*100} || {recall*100} || {precision*100} || {auc*100}')
```

```
31/31 [==============================] - 0s 9ms/step - loss: 0.1759 - accuracy: 0.9259 - recall: 0.9259 - precision: 0.9336 - auc: 0.9926
0.1758619248867035 || 92.59259104728699 || 92.59259104728699 || 93.36099624633789 || 99.2616355419159
```

```
predictions = model.predict(test_generator)
predictions
```

```
array([[1.87266190e-02, 3.68486345e-02, 9.44424748e-01],
       [6.78586215e-02, 9.31995094e-01, 1.46305480e-04],
       [4.62042749e-01, 5.27890444e-01, 1.00668063e-02],
       [1.24159150e-01, 7.17904925e-01, 1.57935962e-01],
       [1.89739749e-01, 8.09600294e-01, 6.59941696e-04],
       [1.35804102e-01, 8.52146685e-01, 1.20491534e-02],
       [1.05921060e-01, 8.94061685e-01, 1.72328710e-05],
       [3.40306317e-03, 9.96074677e-01, 5.22339426e-04],
       [1.23956138e-02, 9.86249268e-01, 1.35508506e-03],
       [3.86655107e-02, 8.46311450e-01, 1.15023002e-01],
       [4.62259166e-02, 9.44414437e-01, 9.35973041e-03],
       [1.08485809e-02, 9.85631466e-01, 3.51999537e-03],
       [7.16880243e-03, 9.92536187e-01, 2.95082806e-04],
       [3.05408705e-03, 9.96933818e-01, 1.20733430e-05],
       [2.06655245e-02, 9.77108002e-01, 2.22647213e-03],
       [5.80807514e-02, 9.19853926e-01, 2.20653582e-02],
       [2.03959271e-02, 9.67638493e-01, 1.19656678e-02],
       [2.29558581e-03, 9.97681260e-01, 2.30243313e-05],
       [7.54893338e-03, 9.92437661e-01, 1.33534650e-05],
       [2.14830739e-03, 9.97850776e-01, 9.34227387e-07],
       [7.76486238e-03, 9.92079437e-01, 1.55805916e-04],
       [5.12117110e-02, 9.38800633e-01, 9.98764485e-03],
       [2.65497220e-04, 9.99734461e-01, 3.64974042e-08],
       [4.13060375e-02, 9.45459127e-01, 1.32348342e-02],
       [1.49108106e-02, 9.84836996e-01, 2.52194237e-04],
       [4.96018901e-02, 9.43275690e-01, 7.12243235e-03],
       [2.86361639e-04, 9.99713480e-01, 9.64771445e-08],
       [3.05484538e-03, 9.96940374e-01, 4.73620184e-06],
       [3.93911125e-03, 9.96059775e-01, 1.15554076e-06],
       [3.03878915e-03, 9.96911466e-01, 4.98349946e-05],
       [1.36832539e-02, 9.85658109e-01, 6.58593781e-04],
       [2.80138813e-02, 9.63390172e-01, 8.59590154e-03],
       [1.07185806e-04, 9.99892831e-01, 1.23541861e-08],
       [1.08364820e-02, 9.89160240e-01, 3.32803347e-06],
       [8.77574179e-03, 9.90953684e-01, 2.70676770e-04],
       [1.02980935e-03, 9.98969793e-01, 3.03770008e-07],
       [3.88853229e-03, 9.96110976e-01, 5.10527457e-07],
       [2.95076403e-03, 9.97049034e-01, 2.75866569e-07],
       [3.02533321e-02, 7.99793661e-01, 1.69952989e-01],
       [3.17385327e-03, 9.95404005e-01, 1.42212457e-03],
       [4.14324924e-04, 9.99585450e-01, 1.86392043e-07],
       [1.58635564e-02, 9.82921541e-01, 1.21496257e-03],
       [6.77804754e-04, 9.99322176e-01, 2.51175827e-08],
       [7.25577306e-03, 9.92609084e-01, 1.35163689e-04],
       [2.15986241e-02, 9.77311671e-01, 1.08966231e-03],
       [3.45488568e-03, 9.96208191e-01, 3.36926227e-04],
       [9.41107050e-04, 9.99058902e-01, 1.07174758e-08],
       [1.89291164e-01, 7.92743862e-01, 1.79649480e-02],
```

최종 모델 검증

코드 내용

최종 모델 검증 시각화

CNN 모델 검증

```
def plot_loss_and_accuracy(history):
    history_df = pd.DataFrame(history)
    fig, ax = plt.subplots(1,2, figsize=(12, 6))

    history_df.loc[0:, ['loss', 'val_loss']].plot(ax=ax[0])
    ax[0].set(xlabel = 'epoch number', ylabel = 'loss')

    history_df.loc[0:, ['accuracy', 'val_accuracy']].plot(ax=ax[1])
    ax[1].set(xlabel = 'epoch number', ylabel = 'accuracy')
```

+ Code + Markdown

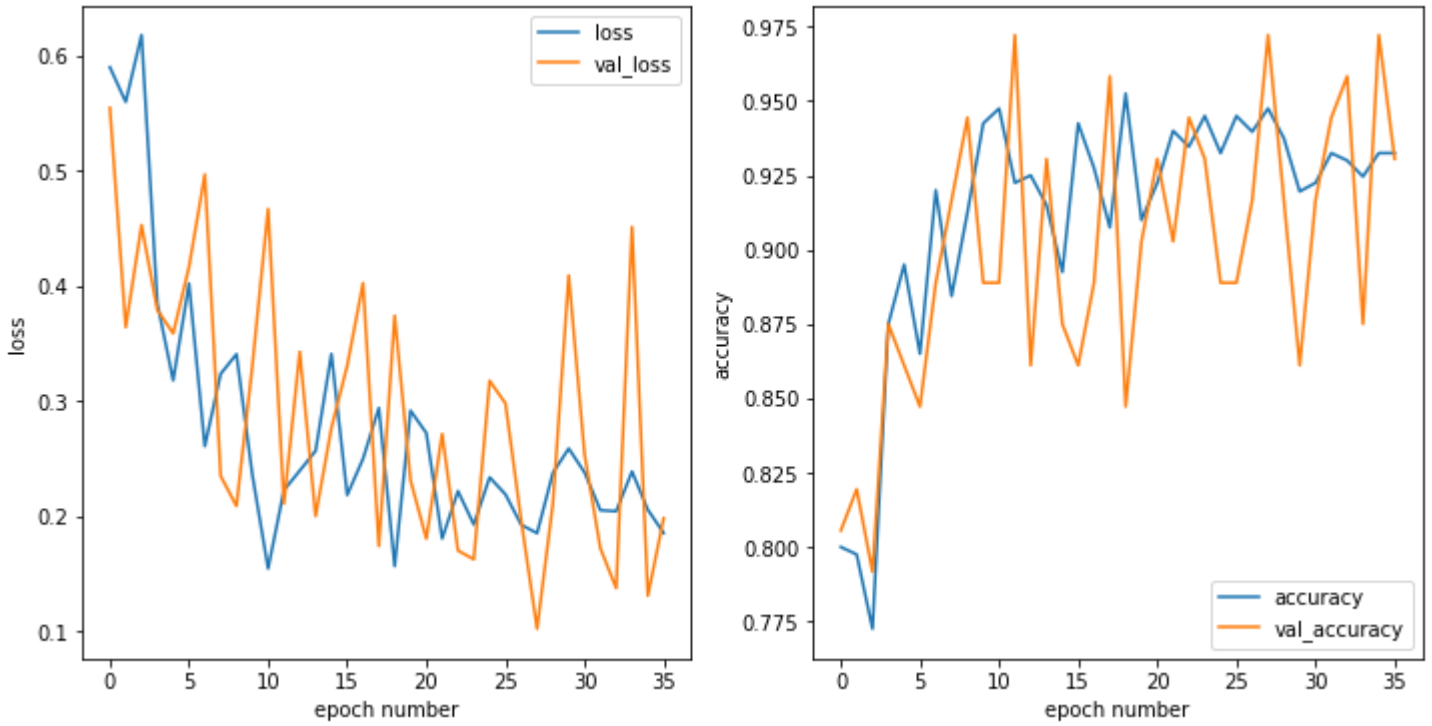
```
plot_loss_and_accuracy(model_history.history)
```

최종 모델 검증 시각화

코드 내용

최종 모델 검증 시각화

CNN 모델 검증



최종 모델 검증 시각화, 에포크가 진행될수록 실패율은 낮아지고 성공율이 올라감을 볼 수 있다

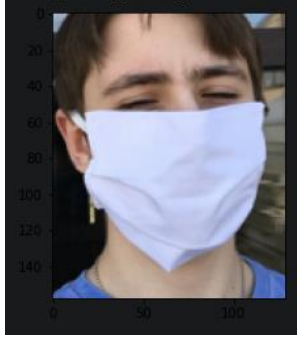
코드 내용

최종 모델 검증 시각화

CNN 모델 검증

```
paths = test_generator.filenames
y_pred = model.predict(test_generator).argmax(axis=1)
classes = test_generator.class_indices

a_img_rand = np.random.randint(0,len(paths))
img = cv2.imread(os.path.join(output_data_path,'test', paths[a_img_rand]))
print(os.path.join(output_data_path,'test', paths[a_img_rand]))
colored_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

plt.imshow(colored_img)
true_label = paths[a_img_rand].split('/')[0]
predicted_label = list(classes)[y_pred[a_img_rand]]
print(f'{predicted_label} || {true_label}')
```

```
./test/with_mask/maksssksksss515_(57, 198).png
with_mask || with_mask
```



테스트 데이터를 직접 삽입하고 모델 작동여부 검증

코드 내용

최종 모델 검증 시각화

CNN 모델 검증

```
def evaluation(y, y_hat, title = 'Confusion Matrix'):
    cm = confusion_matrix(y, y_hat)
    sns.heatmap(cm,  cmap= 'PuBu', annot=True, fmt='g', annot_kws={'size':20})
    plt.xlabel('predicted', fontsize=18)
    plt.ylabel('actual', fontsize=18)
    plt.title(title, fontsize=18)

    plt.show()
```

+ Code + Markdown

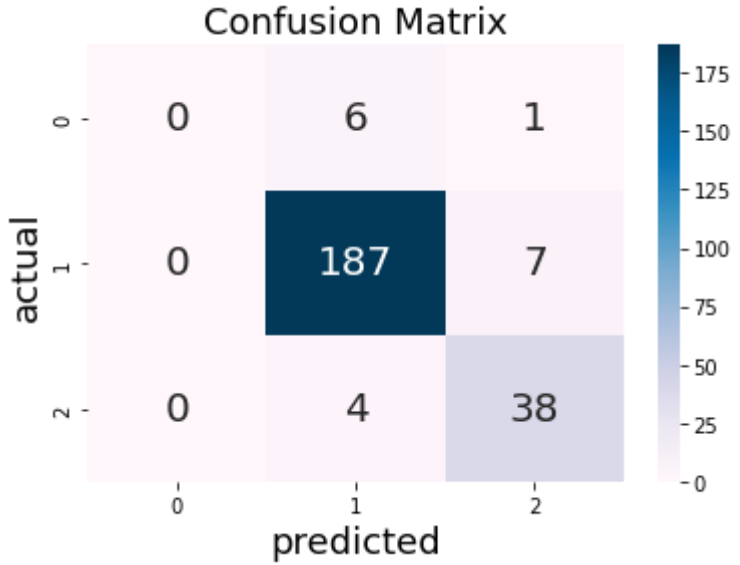
```
y_true = test_generator.labels
y_pred = model.predict(test_generator).argmax(axis=1) # Predict prob and get Class Indices

evaluation(y_true, y_pred)
```

분류모델 성능 평가 지표 결과를 위한 히트맵 생성

코드 내용

최종 모델 검증 시각화

CNN 모델 검증

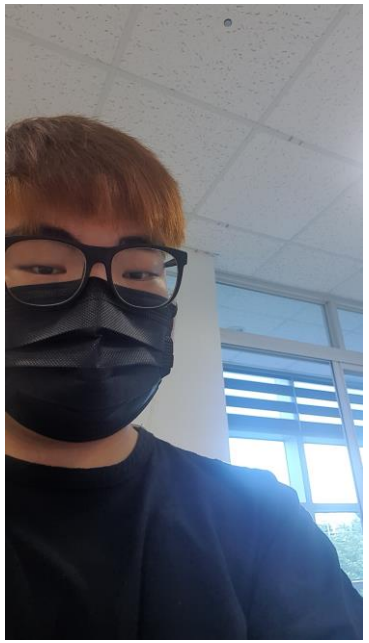


분류모델 성능 평가 지표 결과를 위한 히트맵 생성

결론

결론

결론



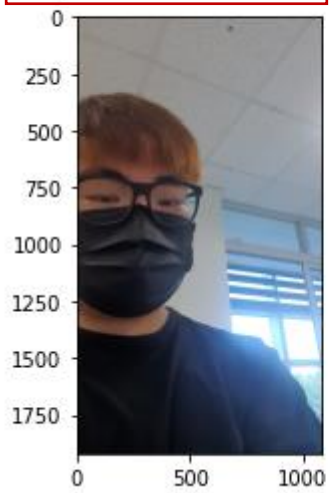
```
paths = test_generator.filenames
y_pred = model.predict(test_generator).argmax(axis=1)
classes = test_generator.class_indices

# 내사진 데이터
input_data_path2 = '/kaggle/input/mypicture/IMG_20220704_085823.jpg'

a_img_rand = np.random.randint(0,len(paths))
img = cv2.imread(os.path.join(input_data_path2))

print(os.path.join(input_data_path2))
colored_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)

plt.imshow(colored_img)
true_label = paths[a_img_rand].split('/')[0]
predicted_label = list(classes)[y_pred[a_img_rand]]
print(f'{predicted_label} || {true_label}')
```

내 사진을 넣어서 결과를 확인해보자

결론

결론

```
/kaggle/input/mypicture/IMG_20220704_085823.jpg
with_mask || with_mask
```



내 사진을 넣어서 결과를 확인해보자

결론

결론

- CNN 모델의 이해
  - CNN 모델은 인간의 시각피질을 모방하여 이미지를 분류하는 모델이다
  - CNN 모델을 생성하는 과정은 인간의 시각피질과 유사하게 계층구조로 되어 있으며, 올라갈수록 복잡하게 이미지를 추상화 한다

- CNN 모델 처리과정의 이해
  - 컨볼루션, 풀링, 맥스풀링, 완전연결인공신경망, 클래스분류

- 중간 검증과정 시각화
  - 모델 검증 시각화 방법(heatmap)
  - Metplotlib 활용한 그래프 활용 검증, overfitting 여부 확인

CNN 모델을 생성, 활용하여 마스크 착용 감지를 구현하는 과정에서 얻은 내용과, 느낀점

참고자료 / 문헌

Referencing & Citation

- 코드, 기반 데이터
  - 코드 : https://www.kaggle.com/code/abdalrahmanshahrour/face-mask-detection-cnn
  - 기반 데이터 : https://www.kaggle.com/datasets/andrewmvd/face-mask-detection
- 문헌, 인터넷 자료
  - https://en.wikipedia.org/wiki/Convolutional_neural_network
- 설명자료
  - CNN 모델 설명 (https://www.youtube.com/watch?v=9Cu2UfNO-gw)
  - AlexNet 논문 (https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf)
- 이미지
  - 보안뉴스 (https://www.boannews.com/media/view.asp?idx=87709)
  - 동아사이언스 (https://dongascience.com/news.php?idx=-5208224)
  - 신세계푸드 (https://www.shinsegaefood.com/company/pr/news_detail.sf?newsId=753)

End