그린컴퓨터아카데미

음식 이미지 분류 모델

TEAM 5조

목차 A table of contents

1 프로젝트 개요

2 프로젝트 팀 구성 및 역할

3 데이터 수집 & EDA

4 모델링 & 테스트

5 자체 평가의견

Part 1

프로젝트 개요

01 프로젝트 개요

프로젝트 주제 선정 배경

[그림] 코로나19 이후 일반국민의 체중 증감 인식

*자료 출처 : SM C&C 틸리언프로, '다이어트는 왜 항상 내일부터 일까?', 2020.08.24. (전국 20-59세 남녀, 928명, 온라인/모바일 조사, 2020.8.19.)

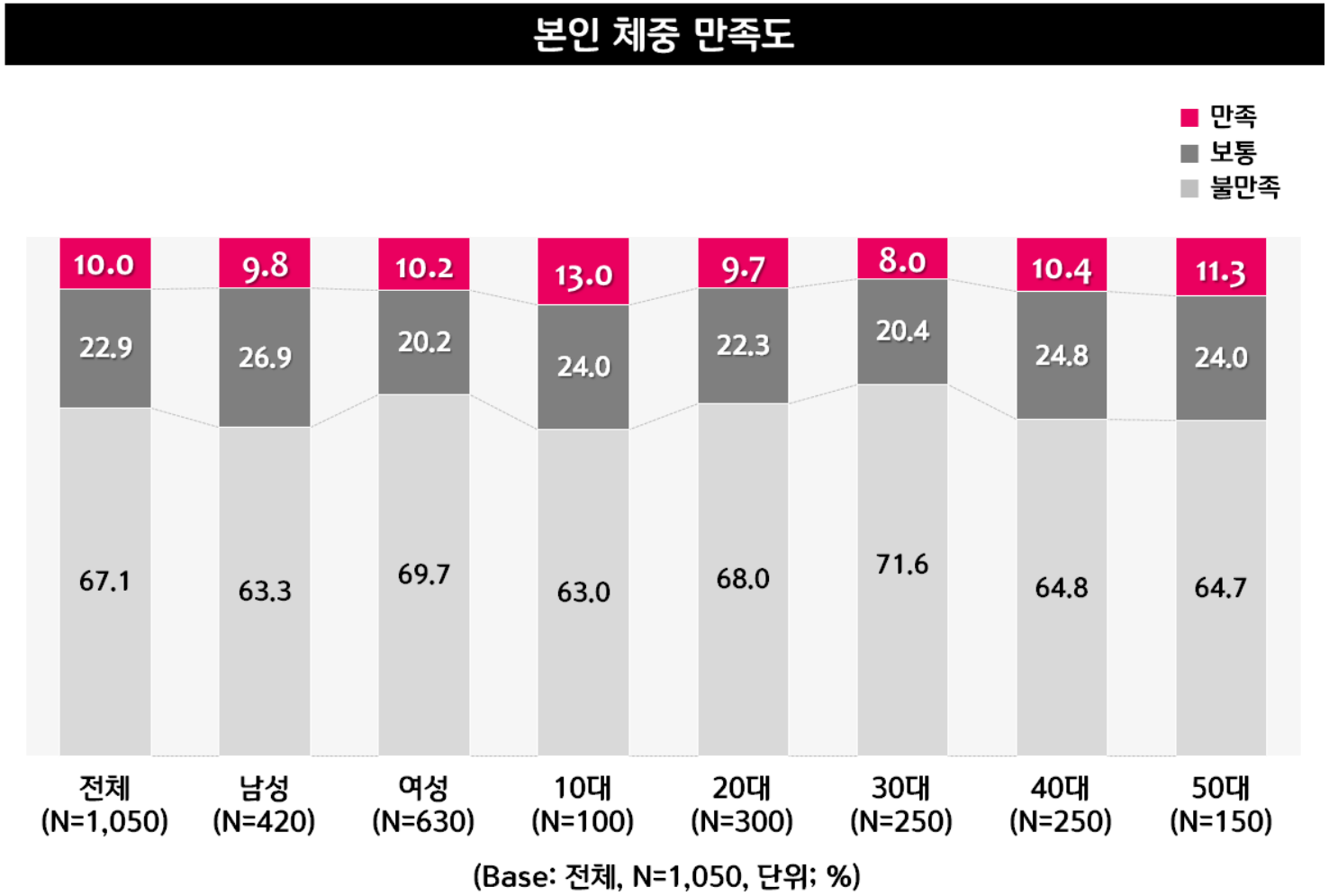
[그림] 주관적 비만 인식도**

*자료 출처 : 국민건강보험공단, '비만, 올바른 문제 인식이 중요', 2018.11.22. (전국, 만 19세 이상 2017년 기준 건강보험 가입자 2,040명, 전화설문조사, 2018.6.01.-15)
**'살쪘다' = '매우 살쪘다'와 '살찐 편이다' 합한 수치임

- 현대인이 가장 많이 하는 고민 하고 도전하는 것 중 하나는 바로 “다이어트”
- 통계로 보면 코로나-19 이후 체중이 증가되었다고 체감하는 사람들이 늘어났음

01 프로젝트 개요

프로젝트 주제 선정 배경



- 전체 10명 중 8명(81.1%)이 현재 자신은 체중 조절이 필요한 상태라고 생각하고 있음
- 상대적으로 남성(77.1%)보다는 여성(83.8%), 그리고 20~30대가 체중 조절의 필요성을 보다 많이 느끼고 있음

01 프로젝트 개요

프로젝트 주제 선정 배경

- 코로나-19로 인해 실외 활동 비중 감소로 인한 체중증가에 대한 걱정거리
- 지속적인 배달음식 섭취로 인한 체중증가 해결방안이 필요함
- 국민의 최대 관심사인 건강을 위한 다이어트
- 우선적인 체중감량 방안인 식단 관리를 위해 음식이미지를 활용한 분류 모델 구축

01 프로젝트 개요

프로젝트 요구 사항

- 다양한 음식 이미지 분류를 위해 최대한 많은 종류의 음식 이미지를 크롤링이 필요함
- 수집된 정보는 분석 및 활용에 적용하기 위해 전처리하고 데이터베이스 또는 파일로 적재해야함
- 사용자가 원하는 음식 이미지 분류를 위해 학습정확도와 검증정확도가 높은 모델을 채택함
- 클래스별 이미지 갯수의 불균형 문제를 해결할 수 있는 방안이 필요함
- 데이트 증강 및 전이 학습을 통한 모델 성능 개선 작업이 필요함

01 프로젝트 개요

구현 기능

음식이미지 입력

현재 사용자가 섭취하는
이미지 입력

음식이미지 분류

이미지를 인식
학습된 모델을 활용하여
음식 이미지 분류

음식이미지 출력

현재 섭취하는 음식에 대한
정보 출력

01 프로젝트 개요

- AI허브 (www.aihub.or.kr)

데이터 수집

- Colab
- VisualStudioCode

개발환경

- Github

형상관리도구

- Python

활용언어

- Numpy

데이터 전처리

- Tensorflow
- keras

ML/DL

- matpolotlib

시각화

Part 2

프로젝트 팀 구성 및 역할

02 프로젝트 팀 구성 및 역할

팀 구성원별 역할

훈련생	역할
이용수	• 프로젝트 주제 탐색 • 데이터 전처리 • 데이터 모델링 • 발표문서작성 • 소스코드정리
임준혁	• 프로젝트 주제 탐색 • 데이터 시각화 • 기술통계분석 • 발표시연영상준비

02 프로젝트 팀 구성 및 역할

프로젝트 수행 일정

구 분	기 간	활 동	비 고
프로젝트 주제 선정 및 기획	2023.05.19~2023.05.25	• 프로젝트 기획 및 주제 선정 • 기획안 작성	• 관심 분야에 따른 주제 선정
데이터 수집 및 환경구축	2023.05.25~2023.05.26	• 필요 데이터 및 수집 절차 정의 • 외부 데이터 수집	• AI허브 건강관리음식이미지데이터 활용
탐색적 자료분석 및 데이터 전처리	2023.05.27~2023.06.02	• 데이터 정제 및 정규화	
예측모델링(지도학습) 및 개선	2023.06.03~2023.06.18	• 모형 구현	• 모델 중 정확도 높은 모델 채택
모델 최종 테스트 및 서비스 구축	2023.06.19~2023.06.20	• 모델 최종 테스트	• 최적화, 오류수정
총 개발기간	2023.05.19~2023.06.20		

Part 3

데이터 수집 & EDA

03 데이터 수집 & EDA

활용 데이터 정의

#건강관리 # 영양분석 # 식이관리 # 음식이미지 # 음식정보 # 접근성 # 사용성

건강관리를 위한 음식 이미지

분야 헬스케어 유형 이미지

갱신년월 : 2023-02 구축년도 : 2020 조회수 : 7,457 다운로드 : 3,235 용량 : 844.79 GB

다운로드 샘플 데이터

코드	음식/식품명	에너지	탄수화물	단백질	지방
A020102	고구마케이크	326.4	39.7	4.5	16.6
A020103	고로케	245.6	23.8	4.8	14.6
A020104	과배기	242.4	28.3	4.7	12.2
A020105	녹차카스테라	239.2	47.6	5.5	3.0
A020106	도넛	340.8	32.5	5.8	20.8
A020107	롤케이크	296.8	38.5	6.0	13.2
A020109	머핀	296.0	41.4	6.9	11.4
A020110	모닝빵	331.0	52.1	9.9	9.3
A020111	모카빵	292.8	46.3	7.0	8.9
A020112	바게트빵	223.2	46.0	7.5	1.0
A020113	밤식빵	223.2	39.7	7.2	3.9
A020114	베이글	208.5	39.8	8.0	1.6
A020115	보리빵	152.4	26.5	2.9	3.9
A020116	붕어빵	203.2	35.3	3.8	5.2
A020119	생크림빵	181.8	26.3	3.9	6.8
A020120	생크림케이크	222.5	22.1	3.1	13.5
A020121	소보로빵	323.0	48.0	7.5	11.2
A020122	소시지빵	258.4	26.4	8.5	13.2
A020123	슈크림빵	220.0	36.4	4.8	6.2
A020124	식빵	213.0	38.3	6.2	3.9
A020127	와플	232.8	26.3	6.3	11.3
A020131	찐빵	158.6	33.5	4.7	0.6
A020132	찹쌀도넛츠	236.6	45.7	4.0	4.2
A020133	초코머핀	355.2	49.7	8.3	13.7
A020134	초코케이크	362.4	32.2	3.8	24.3

코드	음식/식품명	에너지	탄수화물	단백질	지방
A020526	다식	120.1	26.6	1.5	0.8
A020534	다쿠아즈	293.4	36.2	3.4	15.0
A020536	건빵	124.5	22.1	2.6	2.8
A030117	소면	358.0	76.1	8.9	0.3
A030128	당면	87.3	21.5	0.0	0.1
A030129	쇼파게티면	363.0	74.6	11.2	0.2
A170105	단무지	7.2	1.5	0.2	0.0
A180101	간장	3.5	0.5	0.4	0.0
A180103	고추장	10.9	2.2	0.3	0.1
A180104	된장	17.1	1.2	1.4	0.8
A180107	쌈장	10.0	1.5	0.5	0.2
A180115	청국장	21.6	3.0	2.0	0.2
A180118	춘장	7.9	0.6	0.7	0.3
A180303	고춧가루	3.2	0.6	0.1	0.1
A180304	꽃소금	0.2	0.1	0.0	0.0
A180310	산초가루	7.5	1.4	0.2	0.1
A180311	설탕	7.7	2.0	0.0	0.0
A180312	소금	0.0	0.0	0.0	0.0
A180314	식초	0.4	0.1	0.0	0.0
A180315	와사비	5.2	0.9	0.1	0.2
A180316	참기름	45.8	0.0	0.0	5.0
A180318	후추	1.5	0.3	0.1	0.0
A190111	생크림	433.0	3.1	2.0	45.0
A190113	슬라이스치즈	47.2	4.0	3.5	1.8
A190121	치즈	47.2	4.0	3.5	1.8
A190122	커스터드크림	105.0	10.7	5.0	4.6

- 음식 이미지 수집을 위해 AI 허브의 건강관리를 위한 음식 이미지 데이터를 활용
- 카테고리 500여개의 음식 데이터 확보 (한식, 중식, 일식, 수산물, 분식, 정통 양식, 패스트푸드, 제과제빵케익, 커피 등)

(출처 : AI허브 - 건강관리를 위한 음식 이미지
https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=242)

03 데이터 수집 & EDA

활용 데이터 정의


감귤주스


감자그라탕


갈비구이


가츠동

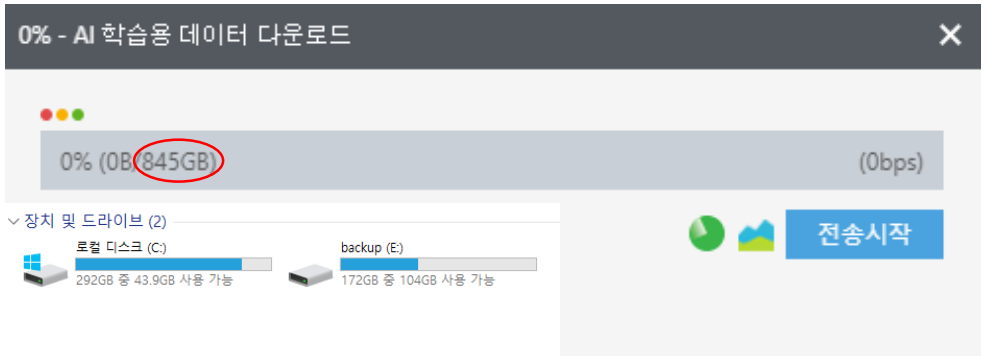
- **데이터 포맷**

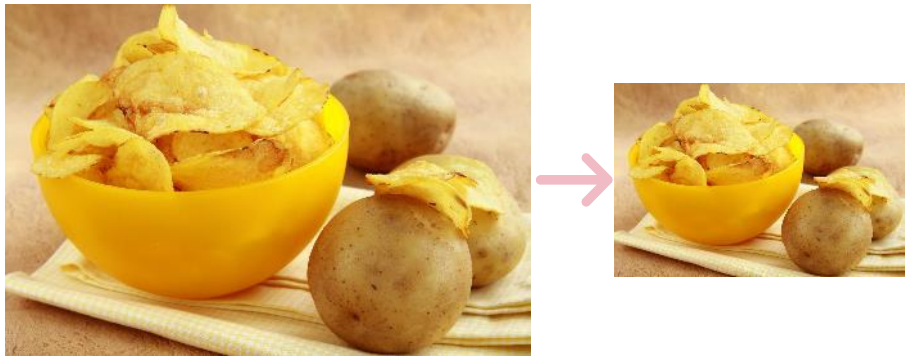
– 이미지
a) 포맷 : jpeg, png
b) 최저해상도 : 720dpi이상, HD 급
이상(90만화소 이상),
해상도: 1,280 x 720 이상

– 데이터 셋
a) 이미지 수량 : 300만장
b) JSON 수량 : 300만개
c) 작업폴더 수량 : 500여개
d) 폴더당 구성 : 카테고리
₩image(.jpg or .png) 500~6000장 + (json or xml) 1개

03 데이터 수집 & EDA

데이터 선정





데이터 수집

수집데이터 총용량이 845GB로 데이터 수집 환경상 문제가 발생함

전처리(resizing)

용량의 문제를 해결하기 위해 파일 해상도를 (224x224) size로 전처리 하여 저장하였음

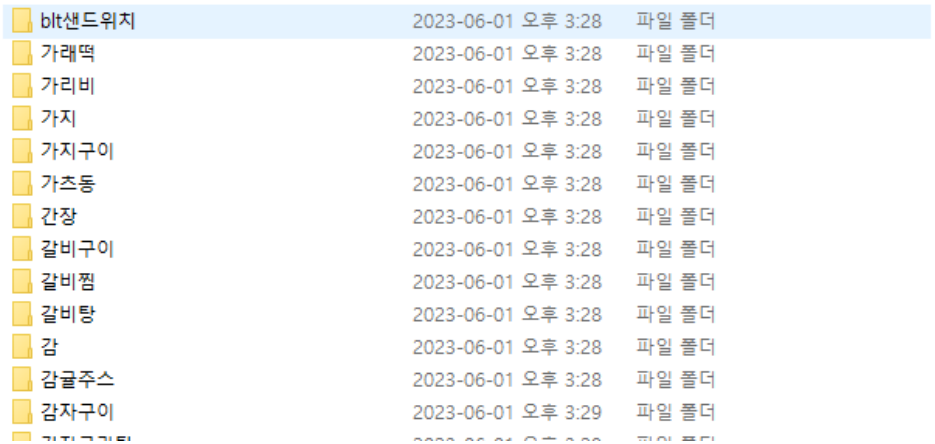
03 데이터 수집 & EDA

데이터 선정

```
Epoch 1/10
57784/57784 [==============================] - 28101s 486ms/step
Epoch 2/10
57784/57784 [==============================] - 27659s 479ms/step
Epoch 3/10
57784/57784 [==============================] - 27633s 478ms/step
Epoch 4/10
57784/57784 [==============================] - 27587s 477ms/step
Epoch 5/10
57784/57784 [==============================] - 27637s 478ms/step
Epoch 6/10
57784/57784 [==============================] - 27608s 478ms/step
Epoch 7/10
57784/57784 [==============================] - 27602s 478ms/step
```

데이터 학습 문제 발생

방대한 이미지 데이터로 인해 학습에 매우 많은 시간이 소모됨



데이터 최종 선정

학습의 환경에 대한 제약때문에 비교적 간소한 총 데이터셋의 15%를 추출한 검증용 데이터셋을 총 데이터셋으로 채택

03 데이터 수집 & EDA

데이터 전처리

```python
from PIL import Image

def magic(base_dir):
    base_dir_list = os.listdir(base_dir)
    filtered_list = list(filter(lambda x: '.' not in x and 'transformed' not in x, base_dir_list))
    max_size = (224, 224)

    for i in filtered_list:
        current_dir = os.path.join(base_dir / i)
        if not os.path.exists(os.path.join(pathlib.Path(base_dir) / 'transformed' / i)):
            os.mkdir(os.path.join(pathlib.Path(base_dir) / 'transformed' / i))

        for idx, image in enumerate(os.listdir(current_dir)):
            input_path = os.path.join(pathlib.Path(current_dir) / image)
            output_path = os.path.join(pathlib.Path(base_dir) / 'transformed' / i / image)

            image = Image.open(input_path)
            image.thumbnail(max_size, Image.ANTIALIAS)
            image.save(output_path, "JPEG", optimize=True)
            print(idx+1, '/', len(os.listdir(current_dir)),'\n',percentage(idx+1,len(os.listdir(current_dir))),i, '까지 완료..')
```

```
0.04 %
1 / 2242
 None 호두파이 까지 완료..
0.09 %
2 / 2242
 None 호두파이 까지 완료..
0.13 %
3 / 2242
 None 호두파이 까지 완료..
0.18 %
4 / 2242
 None 호두파이 까지 완료..
0.22 %
5 / 2242
 None 호두파이 까지 완료..
```

- 해당 파일 경로에서 Image 라이브러리를 이용해 해상도를 224x224 size로 조정하고 경로에 수정된 이미지 파일을 저장하는 magic 함수를 만들어 전처리 수행함

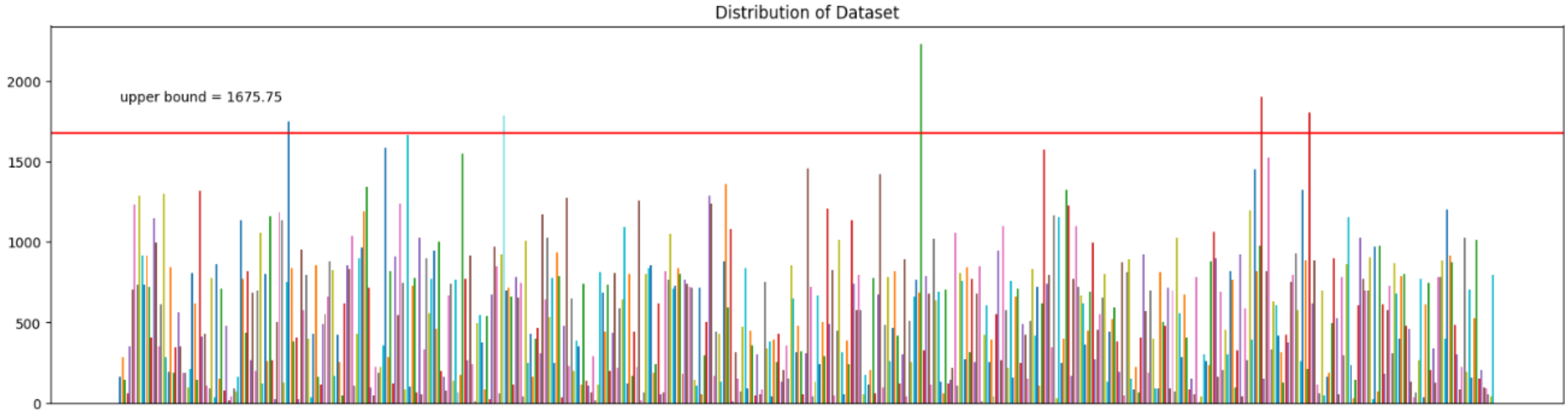
03 데이터 수집 & EDA

EDA

- 기초통계량
- 심한 불균형 데이터

```
299348 files and 570 classes
Mean=525.1719298245614, Median=475.5, StdDev=383.8629642321577
Max=2231, Min=4
percentile = Q1 : 184.5, Q2 : 475.5, Q3 : 781.0
lower_bound = -710.25, upper_bound = 1675.75
outlier = {'춘권': 1750, '스프': 1785, '다크초콜릿': 2231, '청포도': 1904, '소시지구이': 1804}
```

```python
def stats_data_from_dictionary(dictionary):
    mean = statistics.mean(dictionary.values())
    median = statistics.median(dictionary.values())
    stdev = statistics.stdev(dictionary.values())
    sorted_dictionary = np.sort(list(dictionary.values()))
    Q1 = np.percentile(sorted_dictionary, 25)
    Q2 = np.percentile(sorted_dictionary, 50)
    Q3 = np.percentile(sorted_dictionary, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5*IQR
    upper_bound = Q3 + 1.5*IQR
    outlier = {}
    plt.figure(figsize=(20, 5))

    for names, counts in dictionary.items():
        if counts < lower_bound or counts > upper_bound:
            outlier[names] = counts
        plt.bar(names, counts)
        plt.gca().axes.xaxis.set_visible(False)
    plt.title('Distribution of Dataset')
    plt.axhline(upper_bound, color='r')
    plt.text(0, upper_bound+200, f'upper bound = {upper_bound}')

    print(f"{sum(dictionary.values())} files and {len(dictionary)} classes")
    print(f"Mean={mean}, Median={median}, StdDev={stdev}\nMax={max(dictionary.values())}, Min={min(dictionary.values())}")
    print(f"percentile = Q1 : {Q1}, Q2 : {Q2}, Q3 : {Q3}")
    print(f"lower_bound = {lower_bound}, upper_bound = {upper_bound}\noutlier = {outlier}")
```



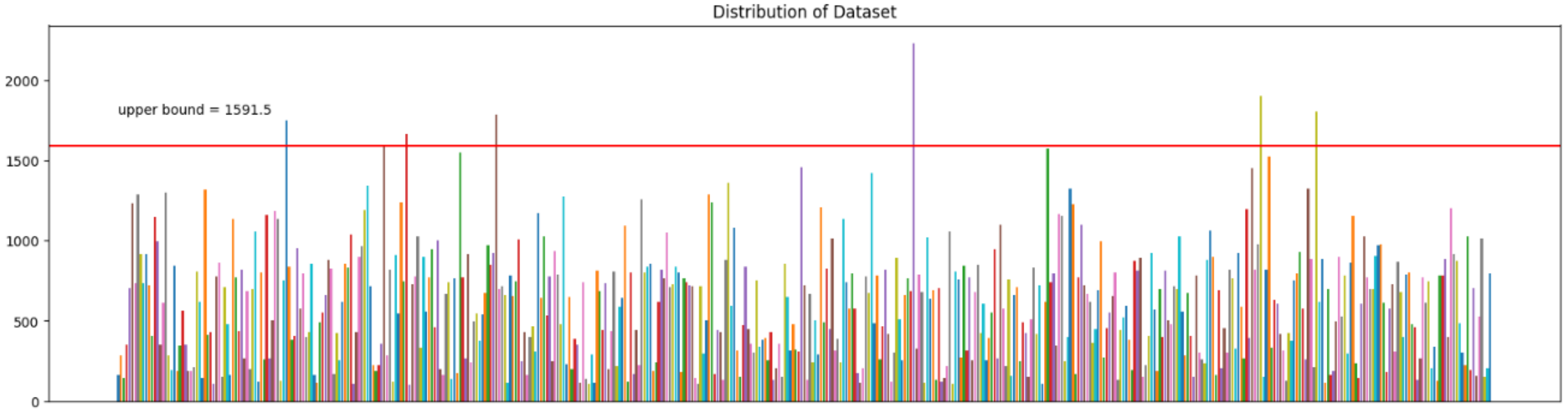
03 데이터 수집 & EDA

EDA

- 100개 이하 데이터 클래스 삭제
- 570 -> 491 classes

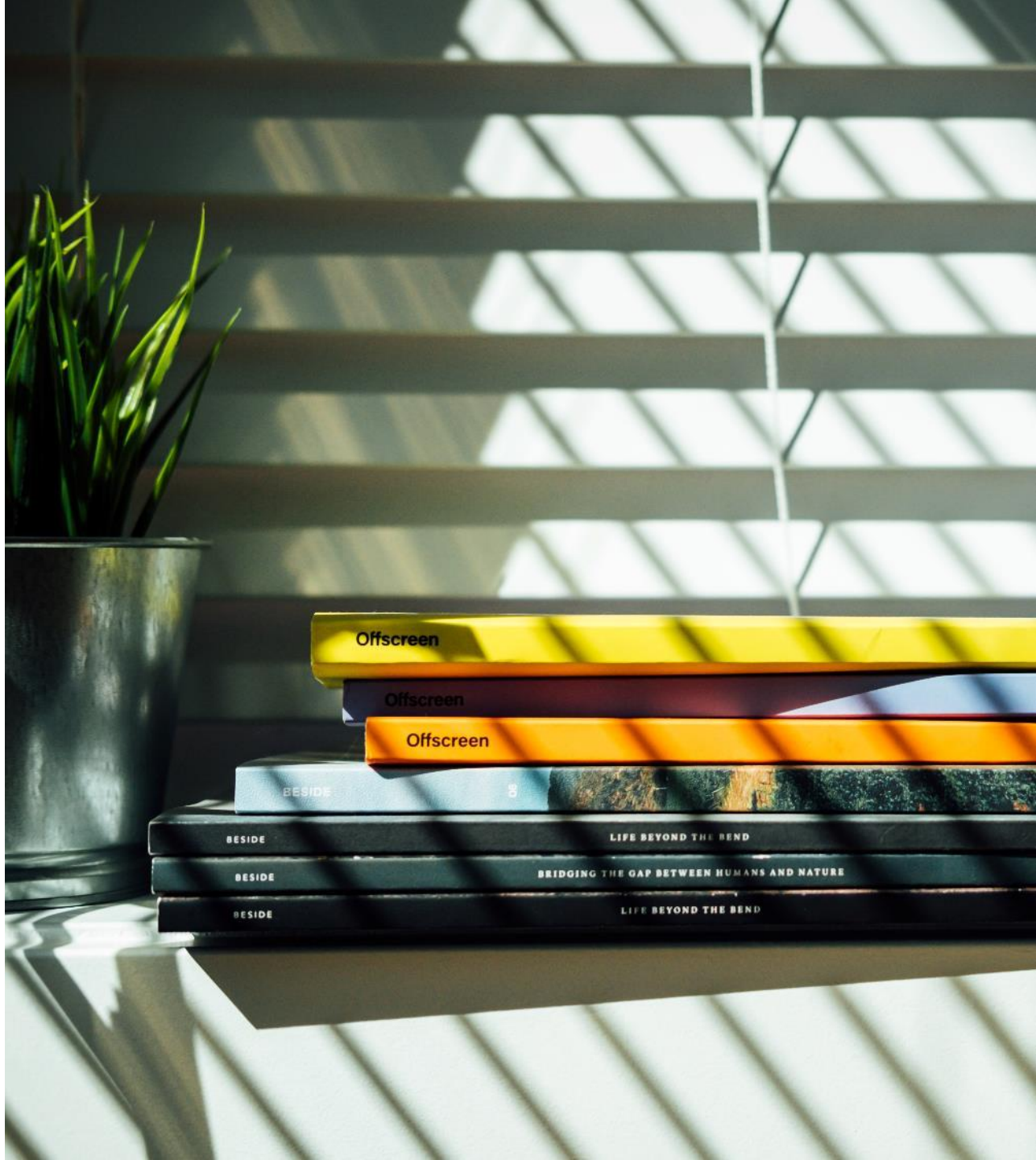
```
min100 = filter_counts(dir, 100)
to_dir = '/content/moved'
moving(dir, to_dir,min100)
```

```
295061 files and 491 classes
Mean=600.938900203666, Median=591, StdDev=359.86946660405914
Max=2231, Min=101
percentile = Q1 : 291.5, Q2 : 591.0, Q3 : 811.5
lower_bound = -488.5, upper_bound = 1591.5
outlier = {'춘권': 1750, '코코아': 1662, '스프': 1785, '다크초콜릿': 2231, '청포도': 1904, '소시지구이': 1804}
```



Part 4

모델링 & 테스트



04 모델링 & 테스트

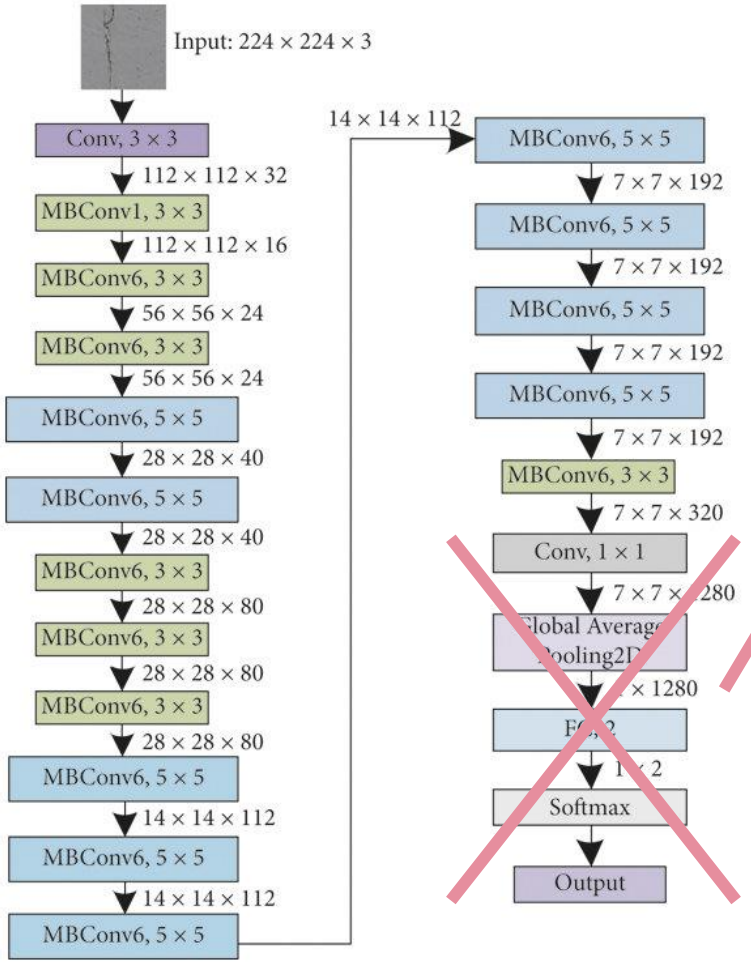
데이터 증강

```
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.efficientnet import preprocess_input
dir = '/content/food/images'
batch_size = 256
image_size = (224, 224)

data_generator = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.15,
    rotation_range=10,
    shear_range=0.2,
    width_shift_range= 0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = data_generator.flow_from_directory(
    dir,
    subset='training',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True,
    seed=42
)

validation_generator = data_generator.flow_from_directory(
    dir,
    subset='validation',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False,
    seed=42
)
```

- Keras ImageDataGenerator 사용하여 데이터 증강
- 85%, 15% 비율로 훈련 검증 데이터 분할
- Efficientnet 모델을 위한 전처리 함수(preprocess_input)

04 모델링 & 테스트

전이학습

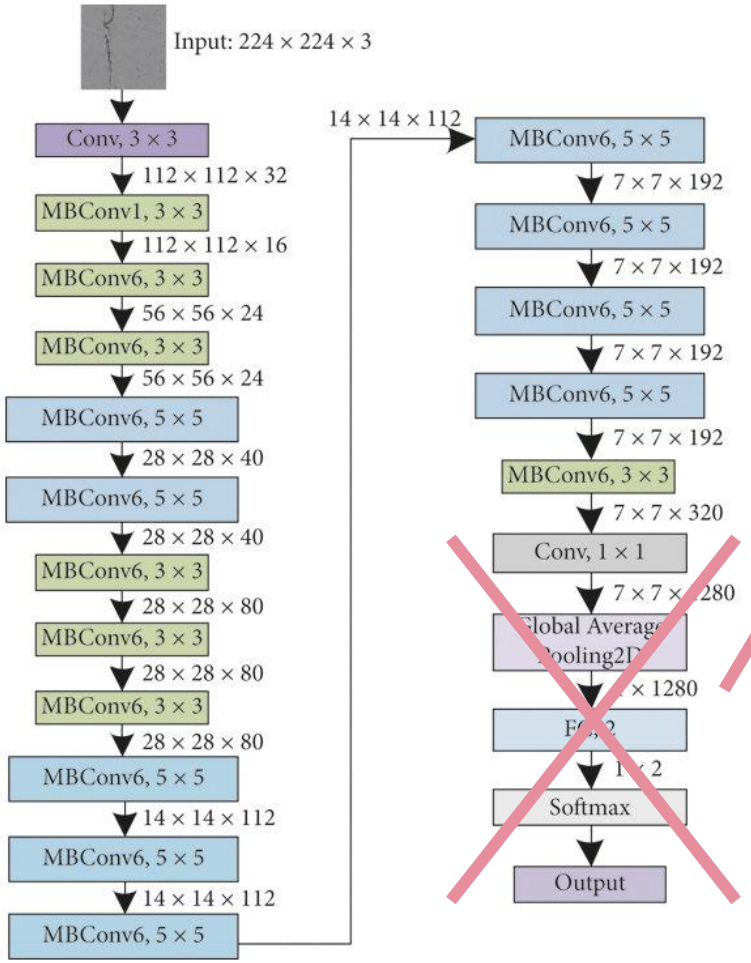


- EfficientNetB0 모델
- 'Imagenet' 사전학습 모델 가중치 초기설정
- Top layer 제외한 레이어 freeze
- 과적합 방지를 위해 dropout 0.5

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(train_generator.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# 모델의 일부 레이어는 학습되지 않도록 설정
for layer in base_model.layers:
    layer.trainable = False

# 모델 컴파일
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

04 모델링 & 테스트

전이학습



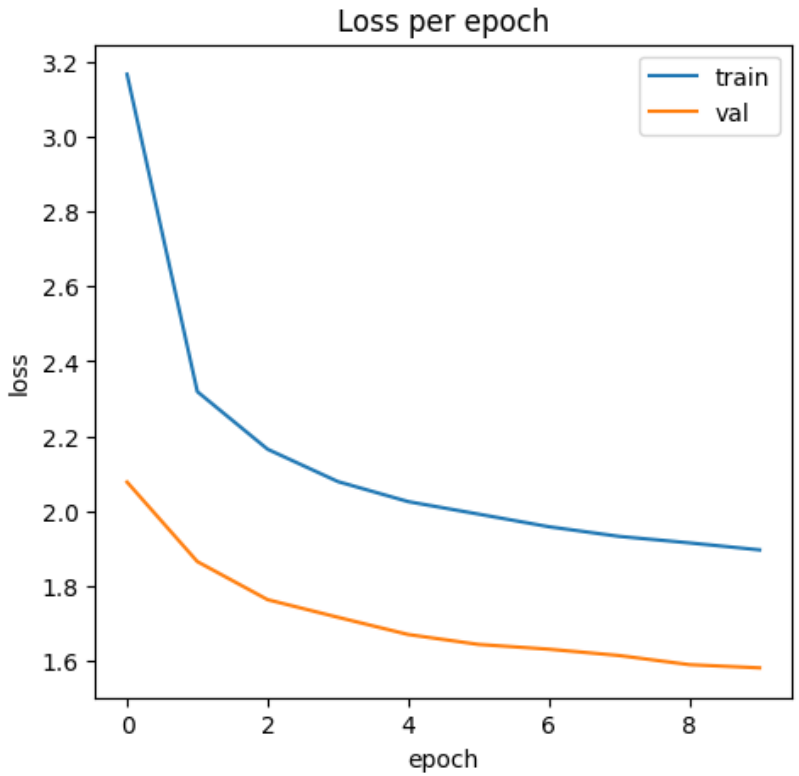
- EfficientNetB0 모델
- 'Imagenet' 사전학습 모델 가중치 초기설정
- Top layer 제외한 레이어 freeze
- 과적합 방지를 위해 dropout 0.5

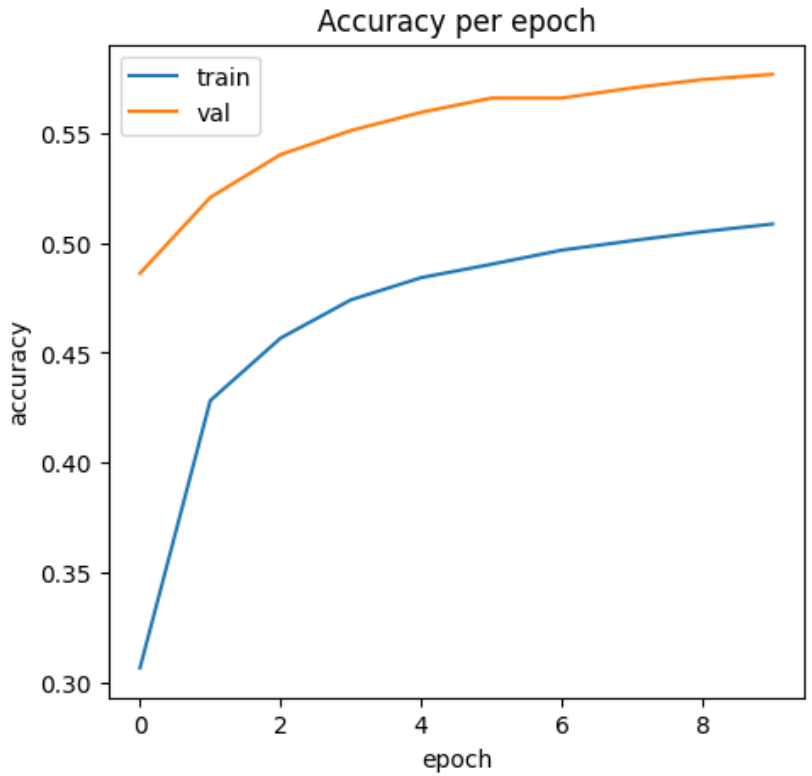
```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(train_generator.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# 모델의 일부 레이어는 학습되지 않도록 설정
for layer in base_model.layers:
    layer.trainable = False

# 모델 컴파일
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

04 모델링 & 테스트

결과

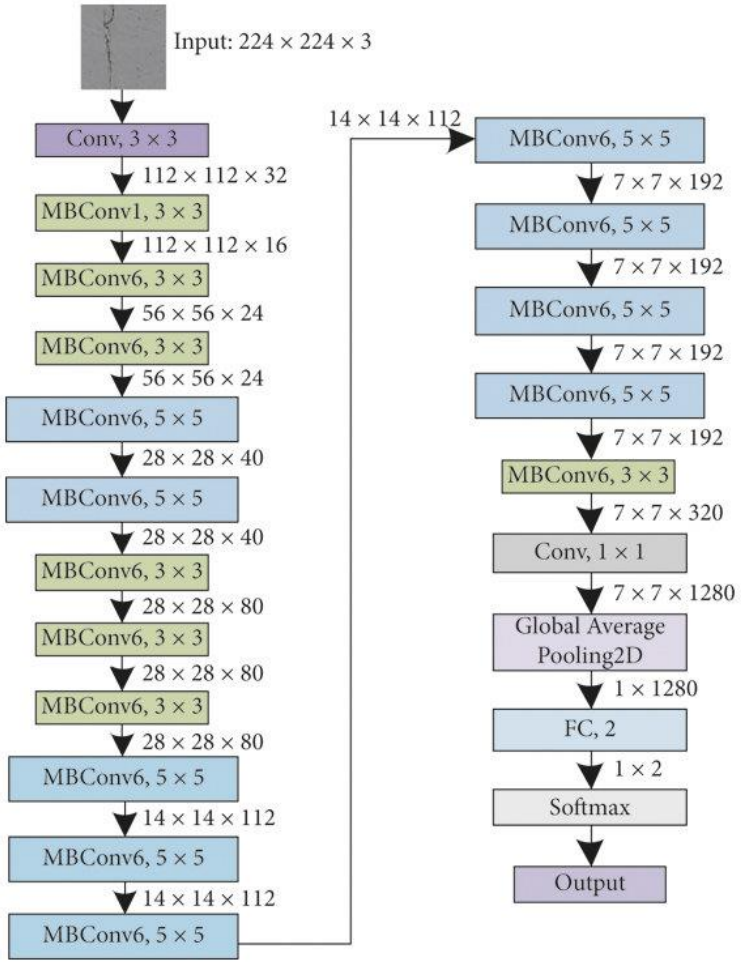
- loss: 1.8963 - accuracy: 0.5086 - val_loss: 1.5813 - val_accuracy: 0.5767





04 모델링 & 테스트

개선점 파악

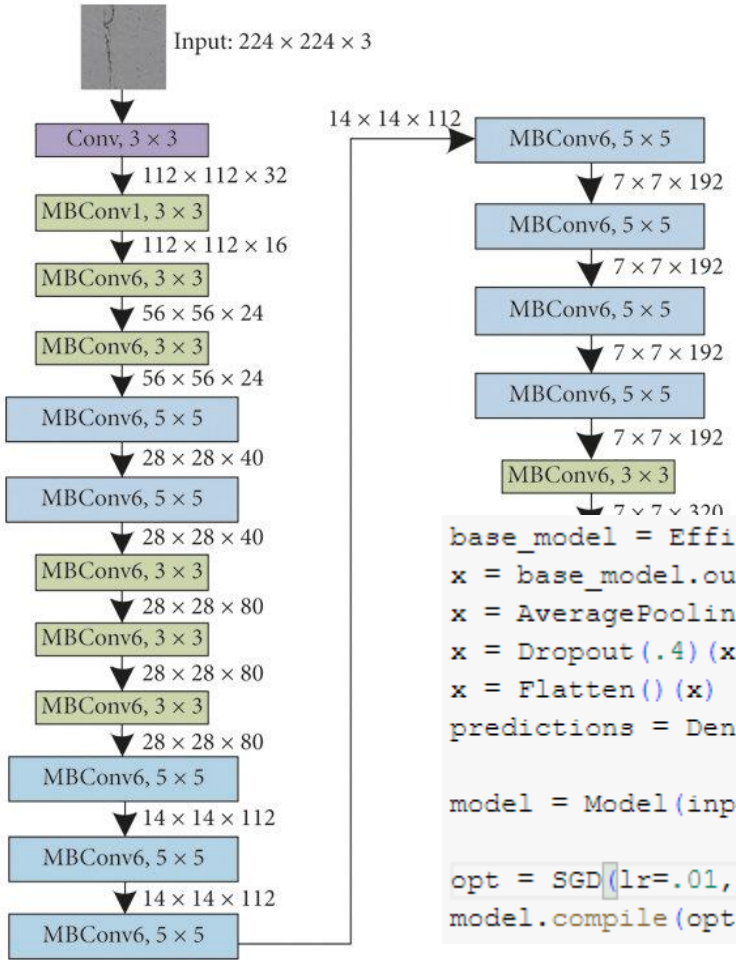



```
for layer in base_model.layers:
    layer.trainable = False
```

- 학습률을 낮춰서 기존 레이어 가중치 업데이트
- 기존 accuracy가 매우 낮아 dropout 비중 하향
- L2규제 추가
- Adam -> SGD + Momentum(0.9)
- Learning Rate Scheduler사용

04 모델링 & 테스트

2차 모델링

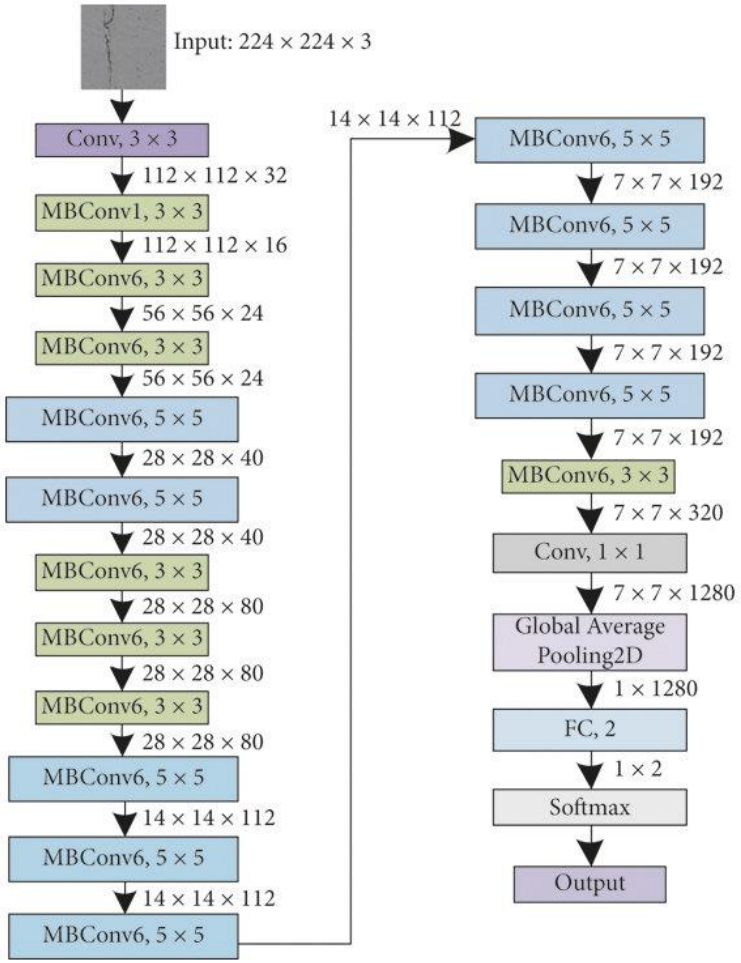



```
for layer in base_model.layers:
    layer.trainable = False
```

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_tensor=Input(shape=(224, 224, 3)))
x = base_model.output
x = AveragePooling2D(pool_size=(7, 7))(x)
x = Dropout(.4)(x)
x = Flatten()(x)
predictions = Dense(train_generator.num_classes, kernel_regularizer=l2(.0005), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

opt = SGD(lr=.01, momentum=.9)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

04 모델링 & 테스트

2차 모델링




```
for layer in base_model.layers:
    layer.trainable = False
```

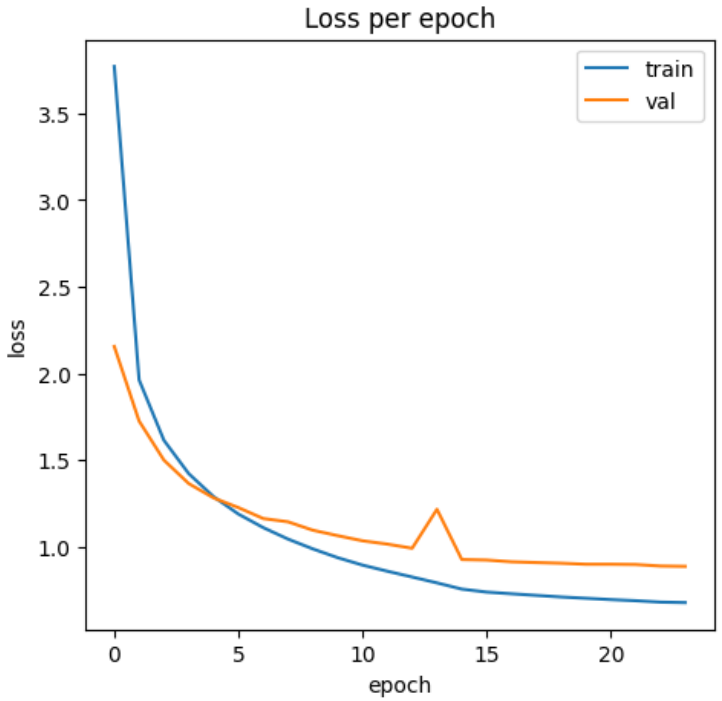
```
def schedule(epoch):
    if epoch < 15:
        return .01
    else:
        return .002
lr_scheduler = LearningRateScheduler(schedule)
```

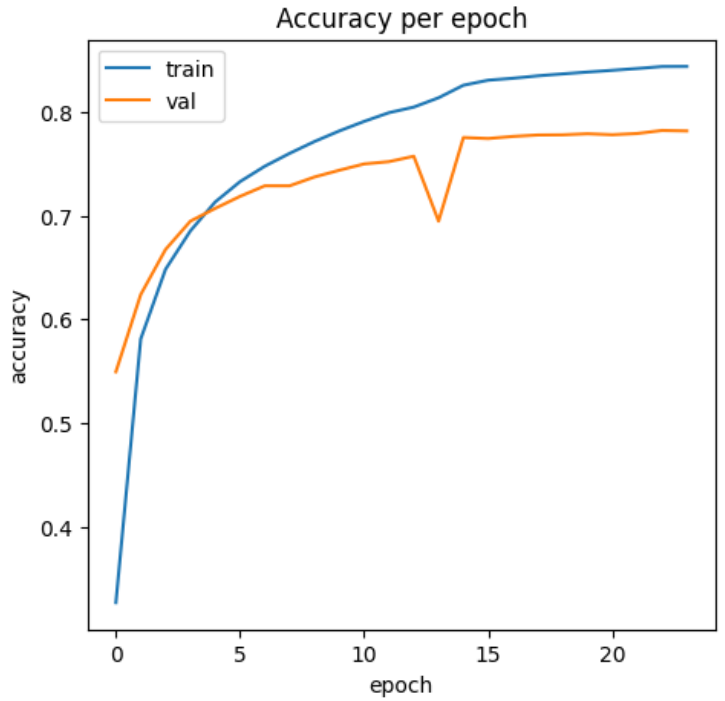
04 모델링 & 테스트

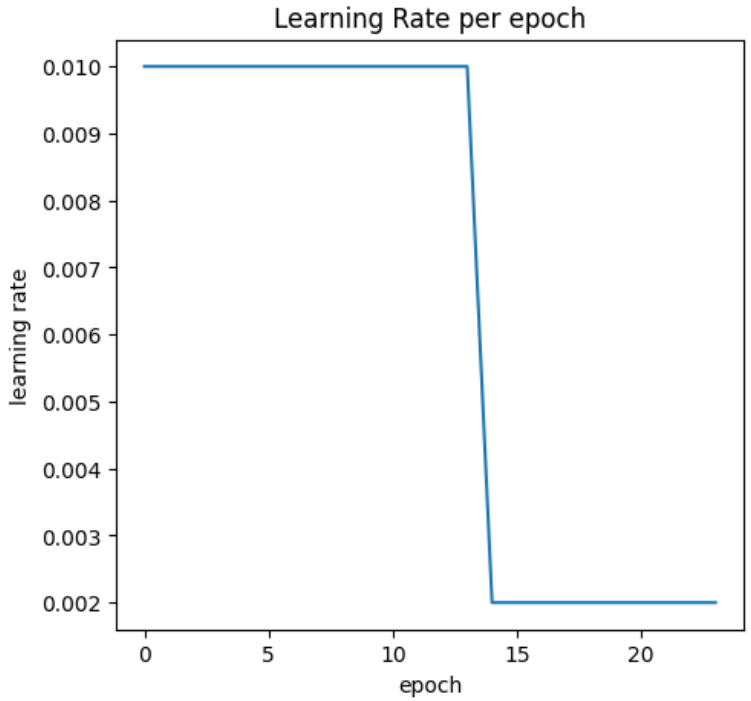
2차 결과

- Accuracy = 0.84 Validation Accuracy = 0.78

```
loss: 0.6820 - accuracy: 0.8438 - val_loss: 0.8903 - val_accuracy: 0.7822 - lr: 0.0020

loss: 0.6794 - accuracy: 0.8439 - val_loss: 0.8882 - val_accuracy: 0.7817 - lr: 0.0020
```







04 모델링 & 테스트

테스트

- sample
  - A020136XX_01136.jpg
  - A190208XX_02819.jpg
  - A260123_50117.jpg
  - A260172XX_00131.jpg
  - A270334XX_10160.jpg
  - B010316XX_13586.jpg
  - B031109XX_00727.jpg
  - B031113XX_11519.jpg
  - B080221XX_31356.jpg
  - B080265XX_14379.jpg
  - B080267XX_10390.jpg
  - B080317XX_10156.jpg
  - B110242XX_14402.jpg
  - B120227XX_00942.jpg
  - B120249XX_14190.jpg
  - B270105XX_03573.jpg
  - label.pkl

```
def show_pred(sample_dir, labels, model, class_names):
  with open(class_names, 'rb')as f:
    class_names = pickle.load(f)

  with open(labels, 'rb')as f:
    labels = pickle.load(f)

  samples = []
  samplelist = os.listdir(sample_dir)

  for f in samplelist:
    if f.endswith('.jpg'):
      file_dir = os.path.join(sample_dir, f)
      samples.append(file_dir)

  fig, ax = plt.subplots(4, 4, figsize=(15,10))
  ax = ax.ravel()
  for i, image in enumerate(samples):
      img = Image.open(image)
      img = img.resize((224, 224))
      ax[i].imshow(img)
      ax[i].axis('off')
      img = np.array(img)
      img = np.expand_dims(img, axis=0)
      pred = model.predict(img)
      pred_class = np.argmax(pred)
      filename = image.split(os.sep)[-1]

      for key, value in class_names.items():
        if value == pred_class:
          ax[i].set_title(f'예측 : {key}   정답 : {labels[filename]}')

  return plt.show()
sample_dir = '/content/drive/MyDrive/Colab Notebooks/sample'
class_names = '/content/drive/MyDrive/foodmodel/class_names.pkl'
labels = '/content/drive/MyDrive/Colab Notebooks/sample/label.pkl'
model = load_model('/content/drive/MyDrive/foodmodel/efficientnet_foodsval2_model.h5')
show_pred(sample_dir, labels, model, class_names)
```

- 임의추출방식 16개 샘플링 및 테스트 시각화

04 모델링 & 테스트

테스트

예측 : 애호박 정답 : 배추

예측 : 훈제치킨 정답 : 닭훈제구이

예측 : 족발 정답 : 족발

예측 : 치킨 정답 : 후라이드치킨

예측 : 치즈빵 정답 : 치즈빵

예측 : 통닭 정답 : 양념치킨

예측 : 만두국 정답 : 만두국

예측 : 스테이크 정답 : 안심스테이크

예측 : 요구르트 정답 : 바닐라아이스크림

예측 : 떡국 정답 : 떡만두국,고기만두

예측 : 애호박 정답 : 애호박

예측 : 육회 정답 : 육회

예측 : 타코야키 정답 : 마늘구이

예측 : 달걀비빔밥 정답 : 비빔밥

예측 : 크로와상 정답 : 양꼬치

예측 : 생선구이 정답 : 고등어

04 모델링 & 테스트

CAM시각화

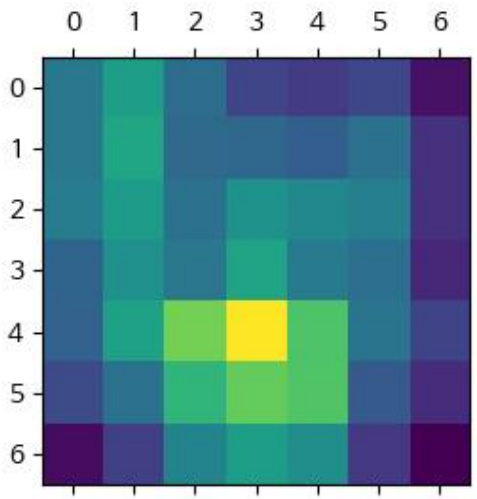
```
last_conv_layer = model.get_layer('top_conv')
heatmap_model = Model([model.input] , [last_conv_layer.output , model.output])
for index, image in enumerate(batch):
    org_Image = image
    image = np.expand_dims(image, axis=0)
    with tf.GradientTape() as gtape:

        conv_output, predictions = heatmap_model(image)

        # 가장 가능성이 높은 클래스에 대한 정보에서 loss 파악
        loss = predictions[:, np.argmax(predictions[0])]

        # top_conv의 특성 맵 출력에 대한 해당 클래스의 그래디언트
        grads = gtape.gradient(loss, conv_output)

        # 특성 맵 채널별 그래디언트 평균값이 담긴 벡터
        pooled_grads = K.mean(grads, axis=(0, 1, 2))

    # 특성 맵의 출력
    heatmap = tf.reduce_mean(tf.multiply(pooled_grads, conv_output), axis=-1)

    # 0 ~ 1 사이의 값으로 정규화
    heatmap = np.maximum(heatmap, 0)

    max_heat = np.max(heatmap)

    if max_heat == 0:
        max_heat = 1e-10

    heatmap /= max_heat

    print('Heamap Shape : ', heatmap.shape)
```

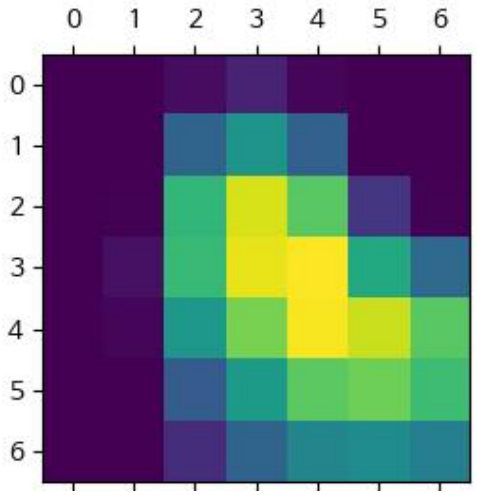
클래스 활성화 맵(Class Activation Map, CAM) 시각화

- 모델의 시점에서 어느 부분을 기준으로 분류
- 마지막 합성곱 층 이전의 특성 맵 가져옴
- 채널별로 가중치를 곱한 특성 맵 시각화

04 모델링 & 테스트

CAM시각화





클래스 활성화 맵(Class Activation Map, CAM) 시각화

- 마지막 합성곱 층 이전의 특성 맵 가져옴
- 채널별로 가중치를 곱한 특성 맵 시각화
- 히트맵을 반투명화 후 두 사진 겹치기

04 모델링 & 테스트

CAM시각화

클래스 활성화 맵(Class Activation Map, CAM) 시각화

```
# heatmap을 원본 이미지 사이즈에 맞춘다
heatmap_Resized = cv2.resize(heatmap[0]  , (width , height))

# 값을 0 ~ 255 사이 int 형으로 변경 // RGB 형식
heatmap_Resized = np.uint8(255 * heatmap_Resized)

# heatmap으로 변환
heatmap_Resized = cv2.applyColorMap(heatmap_Resized , cv2.COLORMAP_JET)

# 기존 이미지와 히트맵 이미지를 겹쳐서 그리기 위해
superimposed_img = np.zeros((height , width , channel))

for c in range(channel) :

    for i in range(height) :

        for j in range(width) :

            h = heatmap_Resized[i][j][c]
            v = org_Image[i][j][c]

            # 히트맵의 강도를 0.4로 설정
            superimposed_img[i][j][c] = h * 0.4 + v
```

- 마지막 합성곱 층 이전의 특성 맵 가져옴
- 채널별로 가중치를 곱한 특성 맵 시각화
- 히트맵을 반투명화 후 두 사진 겹치기

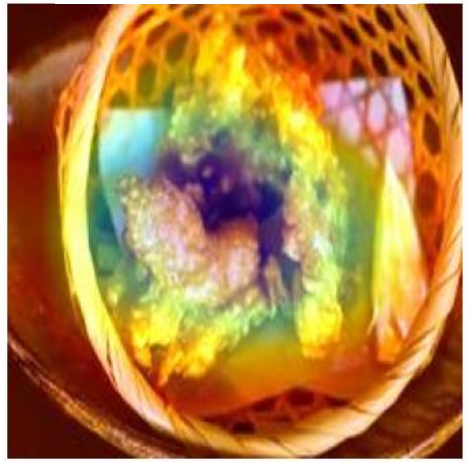
04 모델링 & 테스트

테스트

예측 : 요구르트 정답 : 바닐라아이스크림

예측 : 족발 정답 : 족발

예측 : 치킨 정답 : 후라이드치킨



예측 : 타코야키 정답 : 마늘구이

예측 : 애호박 정답 : 배추

예측 : 크로와상 정답 : 양꼬치

예측 : 육회 정답 : 육회

예측 : 애호박 정답 : 애호박

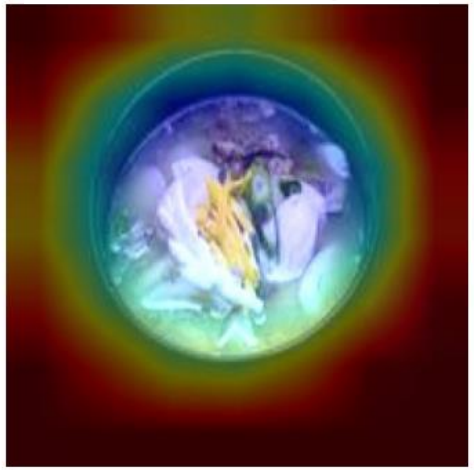
04 모델링 & 테스트

테스트

예측 : 통닭 정답 : 양념치킨

예측 : 만두국 정답 : 만두국

예측 : 떡국 정답 : 떡만두국,고기만두



예측 : 치즈빵 정답 : 치즈빵

예측 : 달걀비빔밥 정답 : 비빔밥

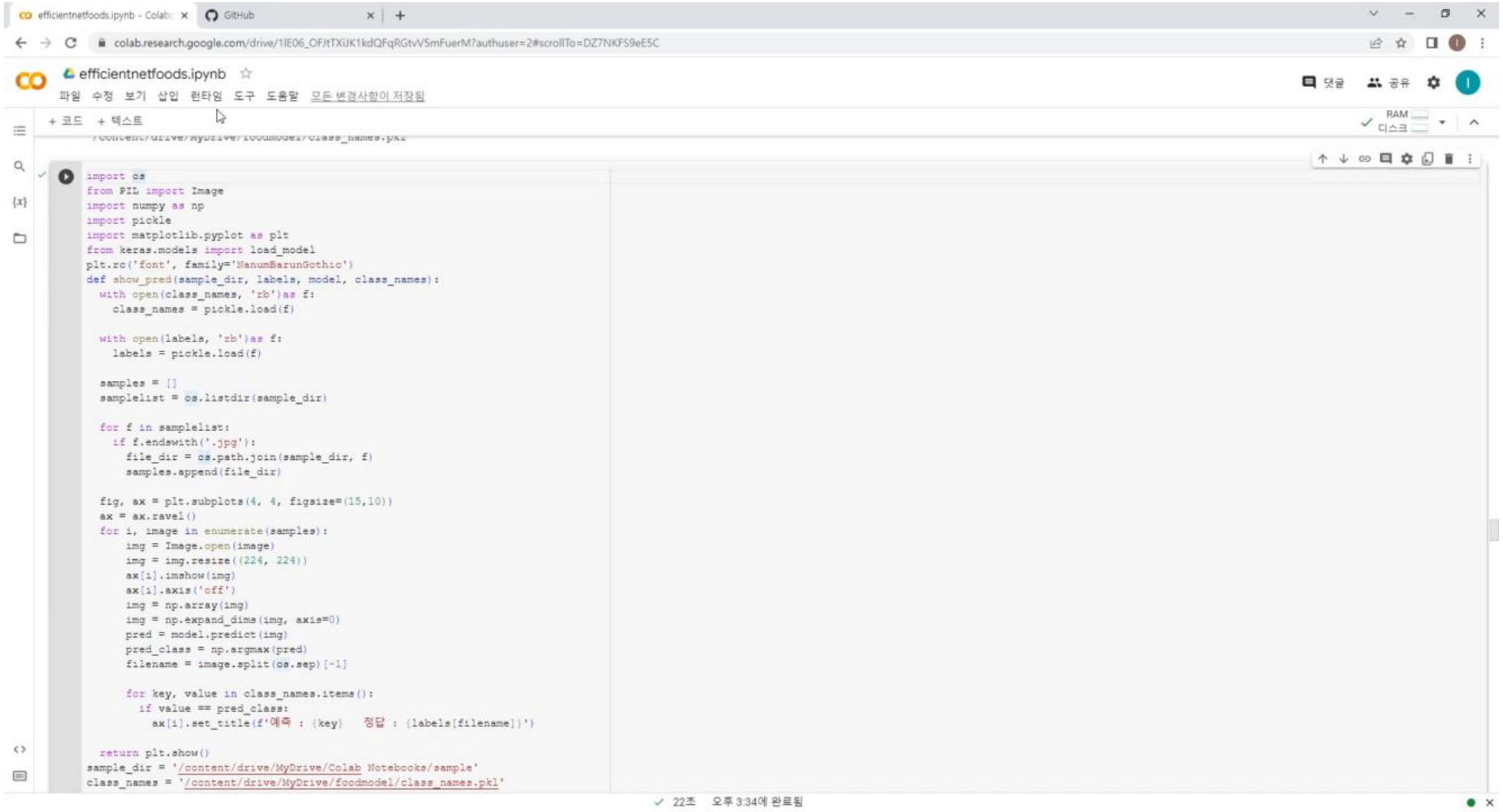
예측 : 스테이크 정답 : 안심스테이크

예측 : 생선구이 정답 : 고등어

예측 : 훈제치킨 정답 : 닭훈제구이

04 프로젝트 수행 결과

시연영상



Part 5

자체 평가 의견

05 자체 평가 의견

개선 사항 및 주요 고려 사항

- 모델링도 중요하지만 데이터의 전처리가 매우 중요한 데이터 분석 과정임을 확인함
- 대용량 이미지 데이터를 다루는데 학습의 시간에 대한 애로사항으로 시스템 환경이 중요한 것을 확인함
- 반영되지 않은 음식의 종류가 많아 향후 추가적인 음식 카테고리 학습이 필요함
- 비슷한 계열의 음식(ex. 된장찌개, 청국장) 같은 음식을 구분할 수 있을 정도의 정확도의 개선이 필요함
- 음식 이미지를 구분하고 출력된 결과에 추가적으로 칼로리, 탄수화물, 나트륨 함량, 단백질 함량 등 메타 정보 제공이 필요함