CHUNGBUK
NATIONAL UNIVERSITY

AI 기반 캔음료 인식 시스템

CNN을 활용한 캔 음료 인식 기술

2024-01 임베디드IoT응용실험 (송영준 교수님)

정보통신공학부
2021039084 유승미

CHUNGBUK
NATIONAL UNIVERSITY

목차

CHUNGBUK
NATIONAL UNIVERSITY

1. 주제 소개

| 시각장애인을 위한 음료 인식 시스템

: AI기반 캔음료 인식(CNN)

CHUNGBUK NATIONAL UNIVERSITY

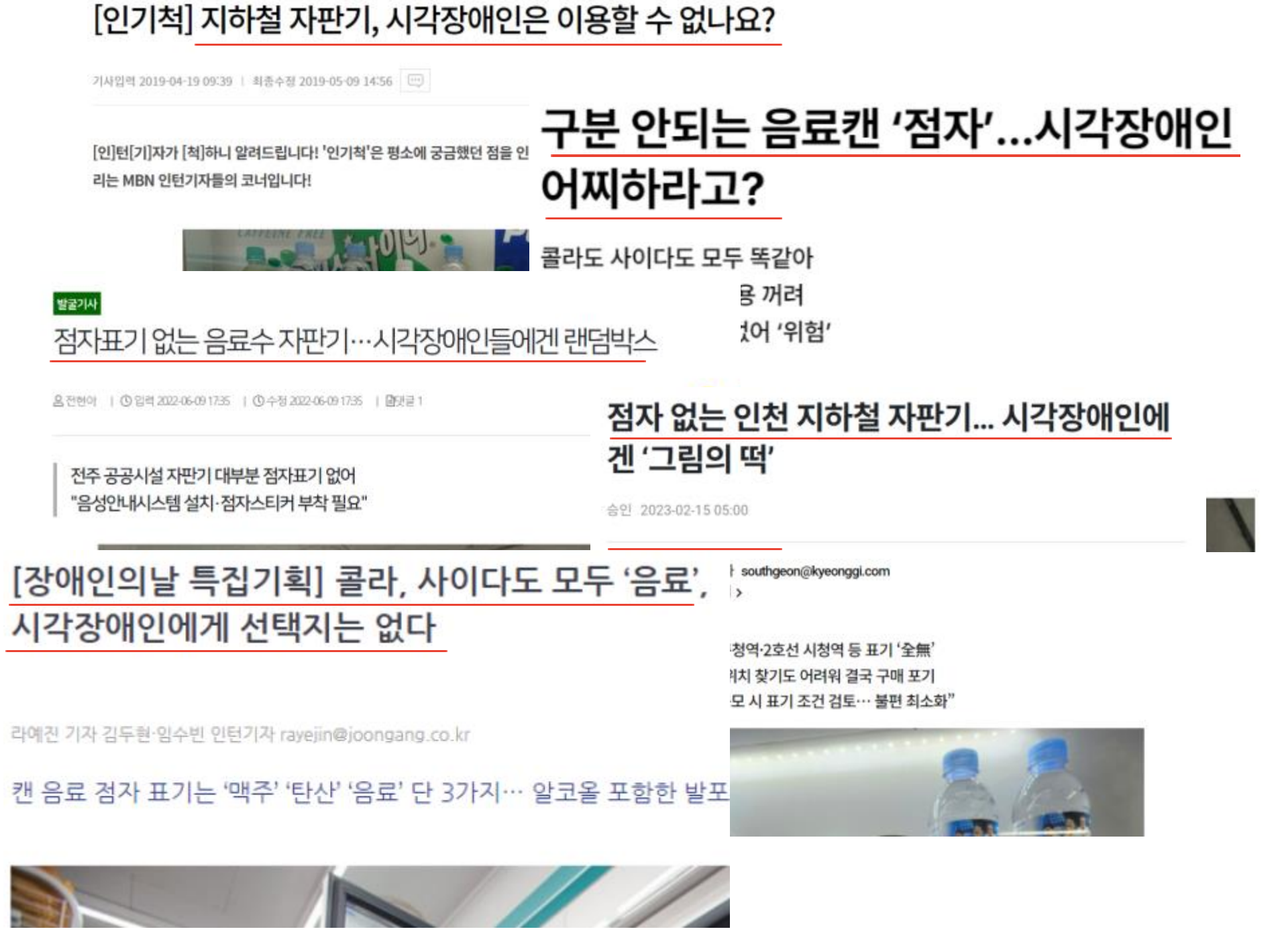
1. 주제 소개

| 시각장애인을 위한 음료 인식 시스템

: AI기반 캔음료 인식(CNN)

시각장애인을 위한

캔음료 접근성 문제 해결

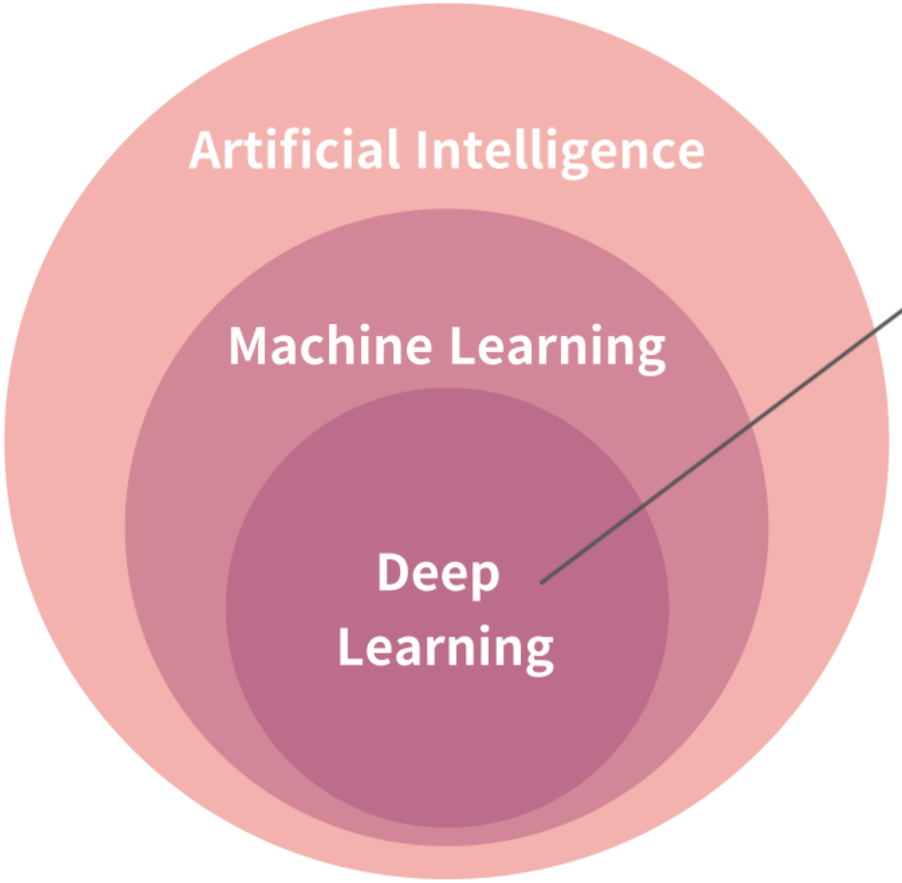


CHUNGBUK NATIONAL UNIVERSITY

개발 환경

CHUNGBUK NATIONAL UNIVERSITY

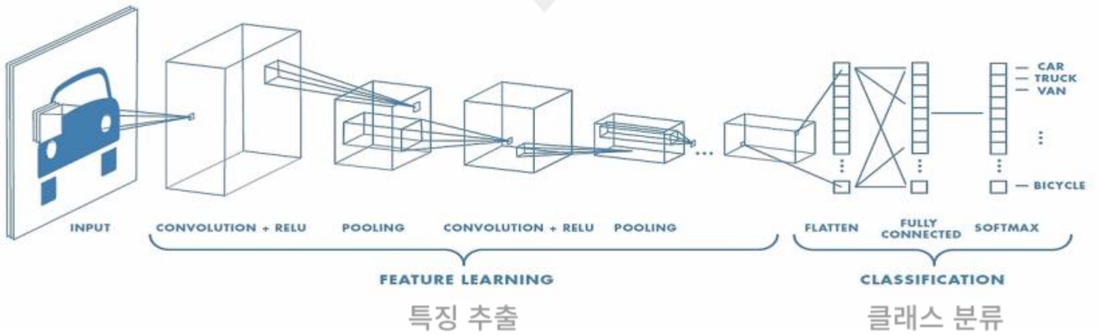
2. 이론적 배경



CNN

Convolutional Neural Network (합성곱 신경망)

"이미지 데이터 처리"



CHUNGBUK NATIONAL UNIVERSITY

3. 데이터 수집 및 전처리

Ⅰ 데이터

코카콜라, 스프라이트, 환타, 펩시, 포카리스웨트 5 종류의 캔 음료수 이미지 (.jpg)

총 670장(134*5)

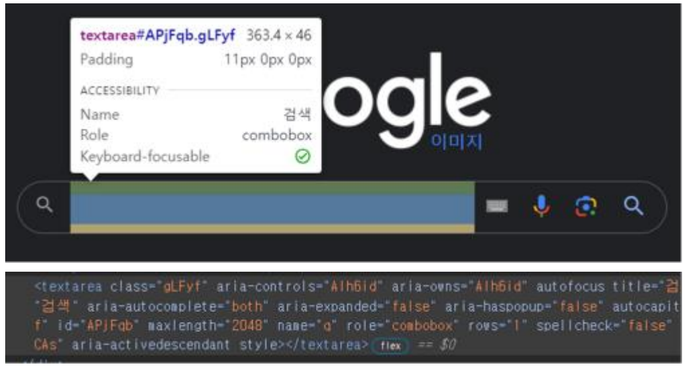
CHUNGBUK NATIONAL UNIVERSITY

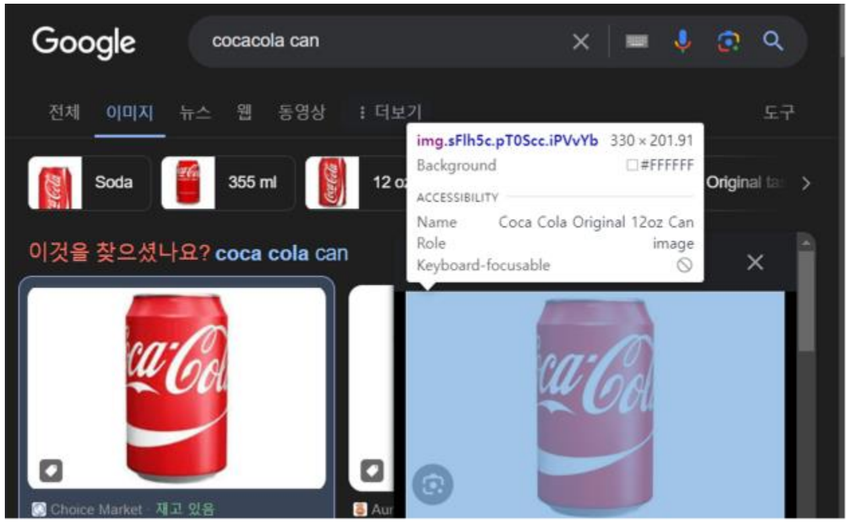
3. 데이터 수집: 구글 이미지 크롤링

구글 웹 드라이버 사용 (selenium)

```python
# 드라이버 초기화
driver = webdriver.Chrome(service=s, options=options)
driver.implicitly_wait(3)

# 검색하고자 하는 음료 이름
beverages = ['cocacola', 'sprite', 'fanta']
# 저장할 기본 경로
base_dir = "D:/Project01/Project02_data/normal_data/"

for beverage in beverages:
    # 해당 음료 이미지 저장 폴더 생성 (폴더가 없다면)
    save_folder = os.path.join(base_dir, beverage)
    if not os.path.exists(save_folder):
        os.makedirs(save_folder)

    # 구글 이미지 검색 페이지 접근
    driver.get("https://www.google.co.kr/imghp?hl=ko&ogbl")
    elem = driver.find_element(By.NAME, "q")
    elem.send_keys(f"{beverage} can")  # 음료 이름으로 검색
    elem.send_keys(Keys.RETURN)
```

```python
# 이미지 다운로드
images = driver.find_elements(By.CSS_SELECTOR, ".rg_i.Q4LuWd")
count = 0
for image in images:
    try:
        image.click()
        time.sleep(2)
        imgUrl = driver.find_element(By.CSS_SELECTOR, '.sFlh5c.pT0Scc.iPVvYb').get_attribute("src")
        opener = urllib.request.build_opener()
        opener.addheaders = [('User-Agent', 'Mozilla/5.0')]
        urllib.request.install_opener(opener)
        time.sleep(2)
        image_path = os.path.join(save_folder, f"{count}.jpg")
        urllib.request.urlretrieve(imgUrl, image_path)
        print(f"{count} {beverage} image download success")
        count += 1
        if count > 400:
            break
    except Exception as e:
        print(e)  # 예외 정보 출력
```





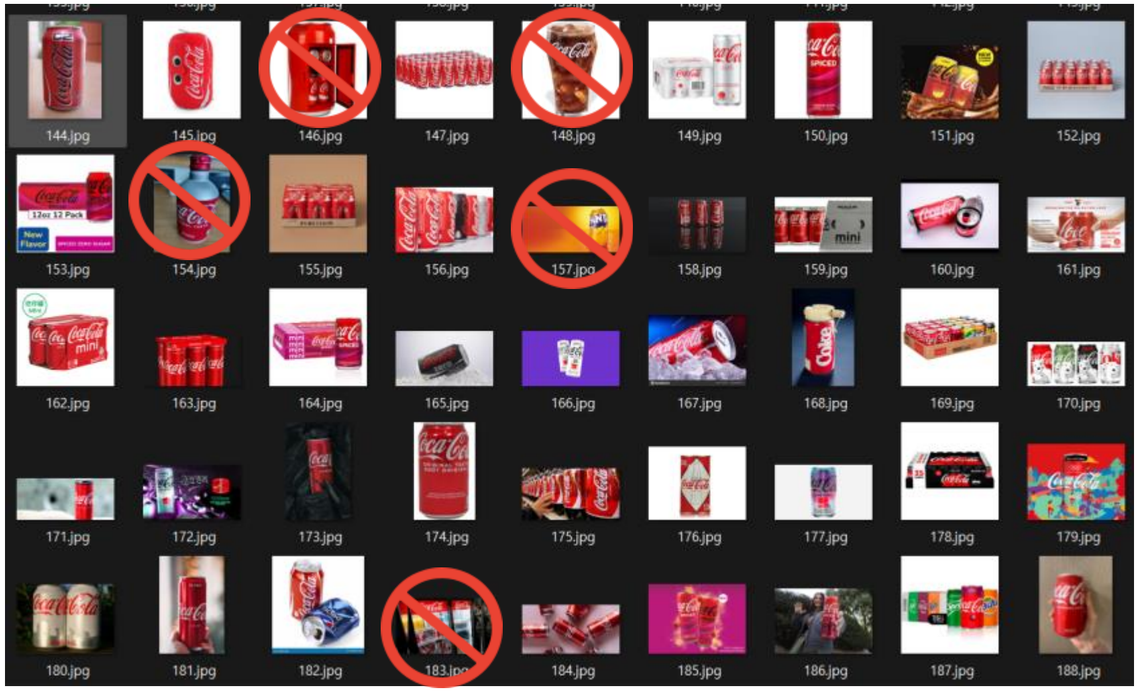
CHUNGBUK NATIONAL UNIVERSITY

3. 데이터 수집: 구글 이미지 크롤링

크롤링 문제점

해당 음료가 아닌 데이터가 존재

"이미지 분류 필요"



CHUNGBUK NATIONAL UNIVERSITY

데이터 전처리

1

이미지 resize (64,64)
&
0~1사이 스케일링

```
name_list = [coke_list, sprite_list, fanta_list, pepsi_list, pocari_list]
name = ['cocacola', 'sprite', 'fanta', 'pepsi', 'pocari']

# 이미지 데이터 처리 및 라벨
def labeling (image_list, label, data_list, label_list) :
    for i in image_list :
        try :
            img = cv.imread(i) # 이미지 파일 읽기
            img = cv.resize(img,(64,64)) # 이미지 크기 조정
            img=np.array(img)/255. # 픽셀 값을 0-1 범위로 정규화
            data_list.append(img) # 데이터 리스트에 추가
            label_list.append(label) # 레이블 리스트에 추가
        except :
            pass
    return data_list, label_list
```

```
index = 0
for std in name_list :
    print("index : ", index)
    print("standard : ", name[index])
```

시행 결과

```
200
200
200
200
200
index :  0
standard :  cocacola
index :  1
standard :  sprite
index :  2
standard :  fanta
index :  3
standard :  pepsi
index :  4
standard :  pocari
```

2

<라벨링>
기준 데이터: 0
기준 아닌 데이터: 1

ex)

기준 음료수: cocacola → 라벨링 값 0
그 외 음료수: sprite, fanta → 라벨링 값 1

```
index = 0
for std in name_list :
    print("index : ", index)
    print("standard : ", name[index])

    # 기준이 되는 음료수 데이터 삽입
    std_data = list()
    std_label = list()

    copy_list = name_list.copy()
    copy_name = name.copy()

    #기준이 되는 음료수: labeling 0
    std_data, std_label = labeling (std, 0, std_data, std_label)

    std_data = np.array(std_data)
    std_label = np.array(std_label)

    # 기준 음료수 외 나머지 데이터 삽입하기 위해 데이터 삭제
    copy_list.remove(std)
    copy_name.remove(copy_name[index])

    dif_data = list()
    dif_label = list()
```

기준 음료수 라벨링 값: 0

```
# 나머지 브랜드에서 무작위로 25%의 데이터 선택하여 처리
for dif in copy_list :
    # 랜덤 추출 : 기준이 아닌 음료수에서는 각각 25% 개수씩 가져오기
    dif = random.sample(dif, len(dif)//4)
    # 1로 라벨링 (different)
    dif_data, dif_label = labeling (dif, 1, dif_data, dif_label)

dif_data = np.array(dif_data)
dif_label = np.array(dif_label)

# 각 브랜드 데이터 관리 (자동 변수 생성)
globals()['std_data_{}'.format(name[index])] = std_data
globals()['std_label_{}'.format(name[index])] = std_label
globals()['dif_data_{}'.format(name[index])] = dif_data
globals()['dif_label_{}'.format(name[index])] = dif_label

index += 1
```

라벨링 값 0, 1 데이터 비율 고려

그 외 음료수 라벨링 값: 1

시행 결과

```
==========자동 변수 생성 확인=========
 std_data|std_label|dif_data|dif_label
200 200 200 200
200 200 200 200
200 200 200 200
200 200 200 200
200 200 200 200
```

CHUNGBUK NATIONAL UNIVERSITY

전처리 분류 모델 생성

1

x,y 생성

x: 이미지 데이터(std,dif)
y: 데이터 라벨링 값(0,1)

2

데이터 분할

훈련 / 테스트 / 검증

데이터 병합

```
std_data = [std_data_cocacola, std_data_sprite, std_data_fanta, std_data_pepsi, std_data_pocari]
std_label = [std_label_cocacola, std_label_sprite, std_label_fanta, std_label_pepsi, std_label_pocari]
dif_data = [dif_data_cocacola, dif_data_sprite, dif_data_fanta, dif_data_pepsi, dif_data_pocari]
dif_label = [dif_label_cocacola, dif_label_sprite, dif_label_fanta, dif_label_pepsi,dif_label_pocari]
```

```
for i in range(0, 5) :  # 각 음료 데이터에서 반복
    print(i)

    # 기준 데이터와 비교 데이터 결합
    x = np.concatenate((std_data[i], dif_data[i]),axis=0)
    y = np.concatenate((std_label[i], dif_label[i]),axis=0)

    # 훈련:테스트=9:1, 훈련데이터 중 검증데이터 비율 10%(전체의 약 9%)
    x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.9, shuffle=True, random_state=42)
    x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, train_size=0.9, shuffle=True, random_state=42)

    print("x : ", x_train.shape, x_test.shape, x_valid.shape)
    print("y : ", y_train.shape, y_test.shape, y_valid.shape)
```

시행 결과

코카콜라 shape	0 x : (324, 64, 64, 3) (40, 64, 64, 3) (36, 64, 64, 3) y : (324,) (40,) (36,)
스프라이트 shape	1 x : (324, 64, 64, 3) (40, 64, 64, 3) (36, 64, 64, 3) y : (324,) (40,) (36,)
환타 shape	2 x : (324, 64, 64, 3) (40, 64, 64, 3) (36, 64, 64, 3) y : (324,) (40,) (36,)
펩시 shape	3 x : (324, 64, 64, 3) (40, 64, 64, 3) (36, 64, 64, 3) y : (324,) (40,) (36,)
포카리 shape	4 x : (324, 64, 64, 3) (40, 64, 64, 3) (36, 64, 64, 3) y : (324,) (40,) (36,)

- 훈련 데이터: 324
- 테스트 데이터: 40
- 검증 데이터: 36
-shape: (64x64)

CHUNGBUK NATIONAL UNIVERSITY

전처리 분류 모델 생성

```python
# 하이퍼파라미터 튜너 설정
tuner = RandomSearch(
    build_model,
    objective='val_accuracy',
    max_trials=50, # 시도할 하이퍼파라미터 세트 수
    executions_per_trial=1, # 각 시도마다의 실행 횟수
    directory='keras_tuning_cocacola', # 튜닝 로그를 저장할 디렉터리
    project_name='image_classification'
)
```

랜덤으로 50개의 조합 생성 및 학습

1. 커널 크기(kernel1~3): 2 or 3
2. 활성화 함수(activation1~4, final):

   relu, elu, prelu
3. 풀링 크기(pool1~3): 2 or 3
4. 드롭아웃 비율(dropout1~3): 0.2~0.4 (step=0.1)
5. 밀집 레이어의 유닛 수:

   dense1_units: 최소값 64, 최대값 128, 스텝 32

   dense2_units: 최소값 32, 최대값 64, 스텝 32
6. 학습률(learning_rate): 1e-4 ~ 1e-2

```python
# 모델 정의를 위한 함수
def build_model(hp):
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=hp.Int('kernel1', min_value=2, max_value=3, step=1), padding='same',
                     input_shape=(x_train.shape[1], x_train.shape[2], x_train.shape[3])))
    activation_choice = hp.Choice('activation1', values=['relu', 'elu', 'prelu'])
    if activation_choice == 'prelu':
        model.add(PReLU())
    else:
        model.add(Activation(activation_choice))
    model.add(BatchNormalization())

    model.add(Conv2D(32, hp.Int('kernel2', min_value=2, max_value=3, step=1), padding='same'))
    activation_choice = hp.Choice('activation2', values=['relu', 'elu', 'prelu'])
    if activation_choice == 'prelu':
        model.add(PReLU())
    else:
        model.add(Activation(activation_choice))
    model.add(BatchNormalization())
    model.add(MaxPool2D(pool_size=hp.Int('pool1', min_value=2, max_value=3, step=1)))
    model.add(Dropout(hp.Float('dropout1', min_value=0.2, max_value=0.3, step=0.1)))

    model.add(Conv2D(64, hp.Int('kernel3', min_value=2, max_value=3, step=1), padding='same'))
    activation_choice = hp.Choice('activation3', values=['relu', 'elu', 'prelu'])
    if activation_choice == 'prelu':
        model.add(PReLU())
    else:
        model.add(Activation(activation_choice))
    model.add(BatchNormalization())
    model.add(MaxPool2D(pool_size=hp.Int('pool2', min_value=2, max_value=3, step=1)))
    model.add(Dropout(hp.Float('dropout2', min_value=0.2, max_value=0.3, step=0.1)))

    model.add(Flatten())
    model.add(Dense(hp.Int('dense1_units', min_value=64, max_value=128, step=32)))
    activation_choice = hp.Choice('activation4', values=['relu', 'elu', 'prelu'])
    if activation_choice == 'prelu':
        model.add(PReLU())
    else:
        model.add(Activation(activation_choice))
    model.add(Dense(1, activation='sigmoid'))

    lr = hp.Float('learning_rate', min_value=1e-4, max_value=1e-2, sampling='LOG')
    model.compile(optimizer=Adam(learning_rate=lr),
                  loss='binary_crossentropy',
                  metrics=['accuracy', tfa.metrics.F1Score(num_classes=1, threshold=0.5)])

    return model
```

CHUNGBUK NATIONAL UNIVERSITY

모델 성능 확인 및 데이터 분류(raw 데이터 전처리)

1 코카콜라 캔 이미지 모델

- val_accuracy: 0.9459459185600281
- loss: 0.09738189727067947
- f1_score: 0.9371980428695679

총 156개의 이미지

2 스프라이트 캔 이미지 모델

- val_accuracy: 0.9189189076423645
- loss: 0.18816091120243073
- f1_score: 0.8629441261291504

총 165개의 이미지

3 환타 캔 이미지 모델

- val_accuracy: 0.9189189076423645
- loss: 0.27995869517326355
- f1_score: 0.8265306353569031

총 156개의 이미지

4 펩시 캔 이미지 모델

- val_accuracy: 0.913948376423645
- loss: 0.21826391322356012
- f1_score: 0.8329441262938471

총 150개의 이미지

5 포카리 캔 이미지 모델

- val_accuracy: 0.9012983746423645
- loss: 0.23995869594583872
- f1_score: 0.80187363535172635

총 134개의 이미지

CHUNGBUK NATIONAL UNIVERSITY

4. 모델 설계 및 구현

1

2 Convolution Layer
(하이퍼파라미터 튜닝)

2

3 Convolution Layer
(하이퍼파라미터 튜닝)

3

2 Convolution Layer
(전이학습)

CHUNGBUK NATIONAL UNIVERSITY

데이터 확인

```python
# 데이터 경로
base_dir = Path(r'C:\Users\ysm\Desktop\dataset')
brands = ['coca', 'fanta', 'pepsi', 'pocari', 'sprite']  # 브랜드 목록

# 각 브랜드별 이미지 파일 경로 수집
filepaths = []
labels = []

for brand in brands:
    brand_dir = base_dir / brand  # 각 브랜드의 폴더 경로
    brand_images = list(brand_dir.glob(r'*.jpg'))  # JPG 이미지만 수집
    filepaths.extend(brand_images)
    labels.extend([brand] * len(brand_images))  # 브랜드 이름으로 라벨 지정

# 데이터프레임 생성
df = pd.DataFrame({
    'Filepath': [str(fp) for fp in filepaths],
    'Label': labels
})

# 음료수 별 이미지 수 최소값 찾기
min_count = df['Label'].value_counts().min()

# 그 수를 기준으로 샘플링
can_df = pd.concat([
    df[df['Label'] == brand].sample(min_count, random_state=0)
    for brand in brands
])
can_df = can_df.sample(frac=1, random_state=0).reset_index(drop=True)

# 라벨링 확인
print(can_df)
```

시행 결과

	Filepath	Label
0	C:\Users\ysm\Desktop\dataset\sprite\spri97.jpg	sprite
1	C:\Users\ysm\Desktop\dataset\pepsi\peps87.jpg	pepsi
2	C:\Users\ysm\Desktop\dataset\coca\coca186.jpg	coca
3	C:\Users\ysm\Desktop\dataset\sprite\spri104.jpg	sprite
4	C:\Users\ysm\Desktop\dataset\coca\coca38.jpg	coca
..	...	...
665	C:\Users\ysm\Desktop\dataset\coca\coca160.jpg	coca
666	C:\Users\ysm\Desktop\dataset\pepsi\peps166.jpg	pepsi
667	C:\Users\ysm\Desktop\dataset\fanta\fant62.jpg	fanta
668	C:\Users\ysm\Desktop\dataset\sprite\spri16.jpg	sprite
669	C:\Users\ysm\Desktop\dataset\sprite\spri81.jpg	sprite

[670 rows x 2 columns]

```
Number of pictures: 670

Number of different labels: 5

Labels: ['sprite' 'pepsi' 'coca' 'fanta' 'pocari']
```

CHUNGBUK NATIONAL UNIVERSITY

| 데이터 준비 과정

1. 데이터 분할 (Train:Test=9:1)

```
# 이미지 데이터 Train, Test 데이터로 분류
train_df,test_df = train_test_split(df, test_size=0.1,random_state=0)
train_df.shape,test_df.shape
```

```
((684, 2), (77, 2))
```

2. 라벨링 및 전처리

```
# 라벨링 및 전처리
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   validation_split=0.2)
train_gen = train_datagen.flow_from_directory(r'C:\Users\ysm\Desktop\dataset',
                                              target_size = (150, 150),
                                              batch_size = 32,
                                              class_mode = 'categorical',subset='training')
val_gen  = train_datagen.flow_from_directory(r'C:\Users\ysm\Desktop\dataset',
                                             target_size = (150, 150),
                                             batch_size = 32,
                                             class_mode = 'categorical',subset='validation')
```

rescale: 0~1사이 정규화

검증데이터: 데이터셋의 20%

모든 이미지 150x150 크기로 조정

한 번에 32개의 이미지 처리

다중 클래스 → 원-핫 인코딩

```
Found 610 images belonging to 5 classes.
Found 151 images belonging to 5 classes.
```

- rescale: 0~1사이 정규화
- validation_split: 데이터셋의 20% (검증 데이터)
- target_size: 모든 이미지 150x150 크기로 조정
- batch_size: 한 번에 32개의 이미지 처리
- class_mode: 다중 클래스 → 원-핫 인코딩

3. 데이터 증강

```
def create_gen():
    # 생성기 및 데이터 증강으로 이미지 로드
    train_generator = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input,
        validation_split=0.1
    )

    test_generator = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=tf.keras.applications.mobilenet_v2.preprocess_input
    )

    train_images = train_generator.flow_from_dataframe(
        dataframe=train_df,
        x_col='Filepath', # 파일위치 열이름
        y_col='Label', # 클래스 열이름
        target_size=(224, 224), # 이미지 사이즈
        color_mode='rgb', # 이미지 채널수
        class_mode='categorical', # Y값(Label값)
        batch_size=32,
        shuffle=True, # 데이터를 섞을지 여부
        seed=0,
        subset='training', # train 인지 val인지 설정
        rotation_range=30, # 회전제한 각도 30도
        zoom_range=0.15, # 확대 축소 15%
        width_shift_range=0.2, # 좌우이동 20%
        height_shift_range=0.2, # 상하이동 20%
        shear_range=0.15, # 반시계방향의 각도
        horizontal_flip=True, # 좌우 반전 True
        fill_mode="nearest"
```

훈련 이미지

```
val_images = train_generator.flow_from_dataframe(
    dataframe=train_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=0,
    subset='validation',
    rotation_range=30,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest"
```

검증 이미지

```
test_images = test_generator.flow_from_dataframe(
    dataframe=test_df,
    x_col='Filepath',
    y_col='Label',
    target_size=(224, 224),
    color_mode='rgb',
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
```

테스트 이미지

CHUNGBUK NATIONAL UNIVERSITY

2 Convolution Layer (하이퍼파라미터 튜닝)

```python
# 튜너 인스턴스 생성
tuner = RandomSearch(
    tuning_model,
    objective='val_accuracy',  # 최적화 목표
    max_trials=30,  # 시도할 최대 트라이얼 수
    executions_per_trial=1,  # 각 트라이얼당 실행 횟수
    directory='my_dir',  # 저장 경로
    project_name='hparam_tuning'  # 프로젝트 이름
)

# 조기 종료 설정
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

# 튜닝 실행
tuner.search(
    train_images,  # 생성된 학습 데이터 생성기
    epochs=100,  # 에폭 수
    validation_data=val_images,  # 생성된 검증 데이터 생성기
    callbacks=[stop_early]  # 조기 종료 콜백 사용
)
```

튜너 결과

```
Best val_accuracy So Far: 0.970588207244873
Total elapsed time: 01h 47m 53s
```

```
첫 번째 컨볼루션 레이어의 최적 필터 수: 48
첫 번째 컨볼루션 레이어의 최적 커널 크기: 3
첫 번째 컨볼루션 레이어의 최적 활성화 함수: elu
두 번째 컨볼루션 레이어의 최적 필터 수: 32
두 번째 컨볼루션 레이어의 최적 커널 크기: 3
두 번째 컨볼루션 레이어의 최적 활성화 함수: elu
완전 연결 레이어의 최적 유닛 수: 288
완전 연결 레이어의 최적 활성화 함수: elu
최적의 학습률: 0.0001
드롭아웃 비율: 0.2
```

```python
# 모델 구성 및 하이퍼파라미터 설정을 위한 함수
def tuning_model(hp):
    model = tf.keras.models.Sequential()

    # 첫 번째 합성곱 레이어
    model.add(tf.keras.layers.Conv2D(
        filters=hp.Int('conv_1_filter', min_value=32, max_value=128, step=16),  # 필터 수 설정
        kernel_size=hp.Choice('conv_1_kernel', values=[3, 5]),  # 커널 크기 선택
        activation=hp.Choice('conv_1_activation', values=['relu', 'elu']),  # 활성화 함수 선택
        input_shape=(224, 224, 3)
    ))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    # 두 번째 합성곱 레이어
    model.add(tf.keras.layers.Conv2D(
        filters=hp.Int('conv_2_filter', min_value=32, max_value=64, step=16),  # 필터 수 설정
        kernel_size=hp.Choice('conv_2_kernel', values=[3, 5]),  # 커널 크기 선택
        activation=hp.Choice('conv_2_activation', values=['relu', 'elu'])  # 활성화 함수 선택
    ))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    # 드롭아웃 레이어 추가
    model.add(tf.keras.layers.Dropout(rate=hp.Float('dropout', min_value=0.0, max_value=0.5, step=0.1)))

    # 평탄화 레이어
    model.add(tf.keras.layers.Flatten())

    # 완전 연결 레이어
    model.add(tf.keras.layers.Dense(
        units=hp.Int('dense_units', min_value=128, max_value=512, step=32),  # 유닛 수 설정
        activation=hp.Choice('dense_activation', values=['relu', 'elu'])  # 활성화 함수 선택
    ))

    # 출력 레이어
    model.add(tf.keras.layers.Dense(5, activation='softmax'))

    # 모델 컴파일
    model.compile(
        optimizer=tf.keras.optimizers.Adam(hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])),  # 학습률 선택
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    return model
```

CHUNGBUK NATIONAL UNIVERSITY

3 Convolution Layer (하이퍼파라미터 튜닝)

```python
# 튜너 인스턴스 생성
tuner = RandomSearch(
    tuning_model,
    objective='val_accuracy',  # 최적화 목표
    max_trials=30,  # 시도할 최대 트라이얼 수
    executions_per_trial=1,  # 각 트라이얼당 실행 횟수
    directory='my_dir',  # 저장 경로
    project_name='hparam_tuning'  # 프로젝트 이름
)

# 조기 종료 설정
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

# 튜닝 실행
tuner.search(
    train_images,  # 생성된 학습 데이터 생성기
    epochs=100,  # 에폭 수
    validation_data=val_images,  # 생성된 검증 데이터 생성기
    callbacks=[stop_early]  # 조기 종료 콜백 사용
)
```

튜너 결과

```
Best val_accuracy So Far: 1.0
Total elapsed time: 20h 39m 53s
```

```
첫 번째 컨볼루션 레이어의 최적 필터 수: 112
첫 번째 컨볼루션 레이어의 최적 커널 크기: 3
첫 번째 컨볼루션 레이어의 최적 활성화 함수: relu
두 번째 컨볼루션 레이어의 최적 필터 수: 32
두 번째 컨볼루션 레이어의 최적 커널 크기: 3
두 번째 컨볼루션 레이어의 최적 활성화 함수: elu
완전 연결 레이어의 최적 유닛 수: 320
완전 연결 레이어의 최적 활성화 함수: elu
최적의 학습률: 0.0001
드롭아웃 비율: 0.0
```

```python
# 모델 구성 및 하이퍼파라미터 설정을 위한 함수
def tuning_model(hp):
    model = tf.keras.models.Sequential()

    # 첫 번째 합성곱 레이어
    model.add(tf.keras.layers.Conv2D(
        filters=hp.Int('conv_1_filter', min_value=32, max_value=128, step=16),  # 필터 수 설정
        kernel_size=hp.Choice('conv_1_kernel', values=[3, 5]),  # 커널 크기 선택
        activation=hp.Choice('conv_1_activation', values=['relu', 'elu']),  # 활성화 함수 선택
        input_shape=(224, 224, 3)
    ))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    # 두 번째 합성곱 레이어
    model.add(tf.keras.layers.Conv2D(
        filters=hp.Int('conv_2_filter', min_value=32, max_value=64, step=16),  # 필터 수 설정
        kernel_size=hp.Choice('conv_2_kernel', values=[3, 5]),  # 커널 크기 선택
        activation=hp.Choice('conv_2_activation', values=['relu', 'elu'])  # 활성화 함수 선택
    ))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    # 세 번째 합성곱 레이어
    model.add(tf.keras.layers.Conv2D(
        filters=hp.Int('conv_2_filter', min_value=32, max_value=64, step=16),  # 필터 수 설정
        kernel_size=hp.Choice('conv_2_kernel', values=[3, 5]),  # 커널 크기 선택
        activation=hp.Choice('conv_2_activation', values=['relu', 'elu'])  # 활성화 함수 선택
    ))
    model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

    # 드롭아웃 레이어 추가
    model.add(tf.keras.layers.Dropout(rate=hp.Float('dropout', min_value=0.0, max_value=0.5, step=0.1)))

    # 평탄화 레이어
    model.add(tf.keras.layers.Flatten())

    # 완전 연결 레이어
    model.add(tf.keras.layers.Dense(
        units=hp.Int('dense_units', min_value=128, max_value=512, step=32),  # 유닛 수 설정
        activation=hp.Choice('dense_activation', values=['relu', 'elu'])  # 활성화 함수 선택
    ))

    # 출력 레이어
    model.add(tf.keras.layers.Dense(5, activation='softmax'))

    # 모델 컴파일
    model.compile(
        optimizer=tf.keras.optimizers.Adam(hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])),  # 학습률 선택
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    return model
```

CHUNGBUK NATIONAL UNIVERSITY

2 Convolution Layer (전이학습)

1 Base model

```
    "DenseNet121": {"model":tf.keras.applications.DenseNet121, "perf":0},
    "MobileNetV2": {"model":tf.keras.applications.MobileNetV2, "perf":0},
```

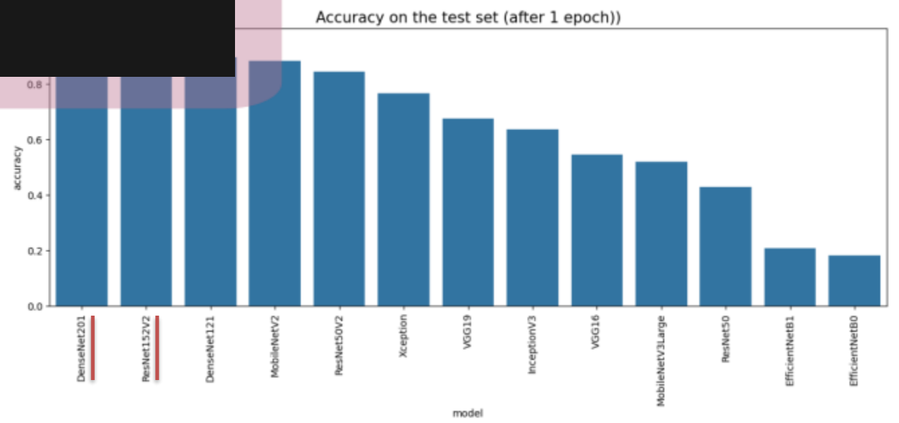
```
cnn = tf.keras.models.Sequential()

# convolutional layer 1
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[150, 150, 3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# convolutional layer 2
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Flattening
cnn.add(tf.keras.layers.Flatten())

# Full Connection
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

# Output Layer
cnn.add(tf.keras.layers.Dense(units=5, activation='softmax'))

# Compiling
cnn.compile(optimizer = 'adam',
            loss = 'categorical_crossentropy',
            metrics = ['accuracy'])
cnn.summary()
```

```
    inputs = pretrained_model.input

    x = tf.keras.layers.Dense(128, activation='relu')(pretrained_model.output)
    x = tf.keras.layers.Dense(128, activation='relu')(x)

    outputs = tf.keras.layers.Dense(5, activation='softmax')(x)
    # 라벨 개수가 5개이므로  Dencs 5로 설정
    model = tf.keras.Model(inputs=inputs, outputs=outputs)

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

    return model
```

2

```
# Train모델 학습
for name, model in models.items():

    # 전이 학습 모델 가져오기
    m = get_model(model['model'])
    el'] = m

    ter()

    rain_images,validation_data=val_images,epochs=1,verbose=0)

    ccuracy 저장
    ounter() - start
    duration,2)
    f'] = duration
     trained in {duration} sec")

    .history['val_accuracy']
    _acc'] = [round(v,4) for v in val_acc]
```

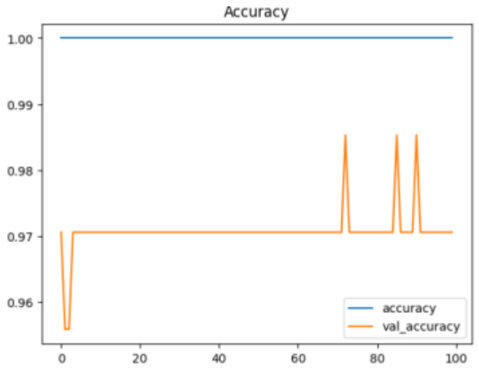
이터로 모델 성능 예측

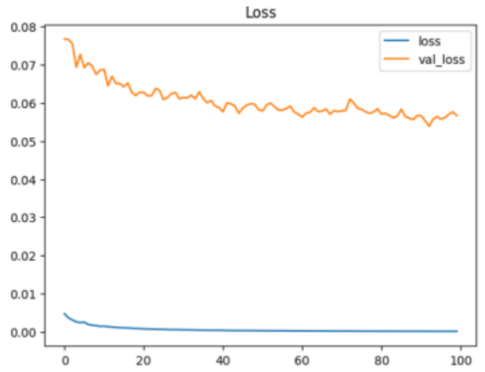


CHUNGBUK NATIONAL UNIVERSITY

5. 결과 분석 (Test data Accuracy & Loss)

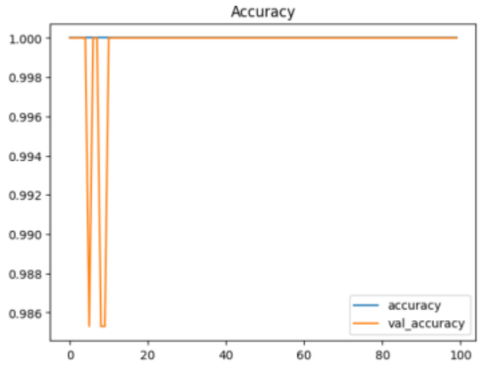
1. 2 Convolution Layer

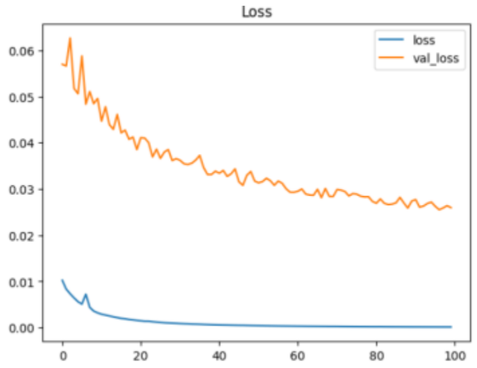




Accuracy on the test set: 97.06%

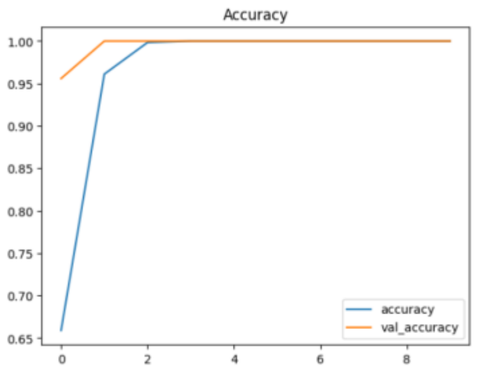
2. 3 Convolution Layer

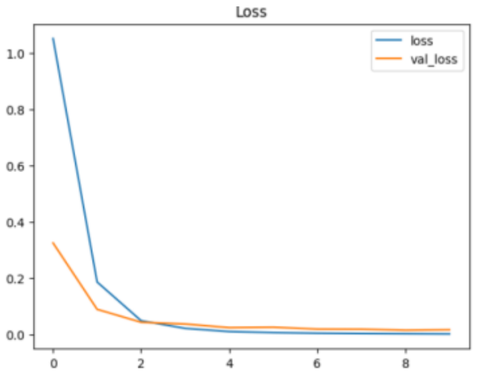




Accuracy on the test set: 100%

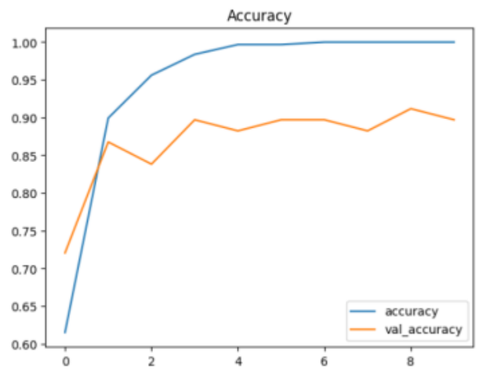
3. DenseNet201 (전이학습)

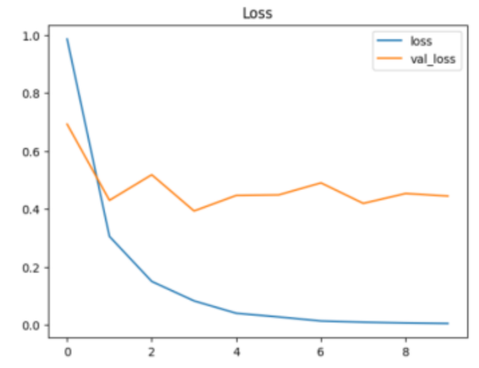




Accuracy on the test set: 93.51%

4. ResNet152V2 (전이학습)





Accuracy on the test set: 90.76%

CHUNGBUK NATIONAL UNIVERSITY

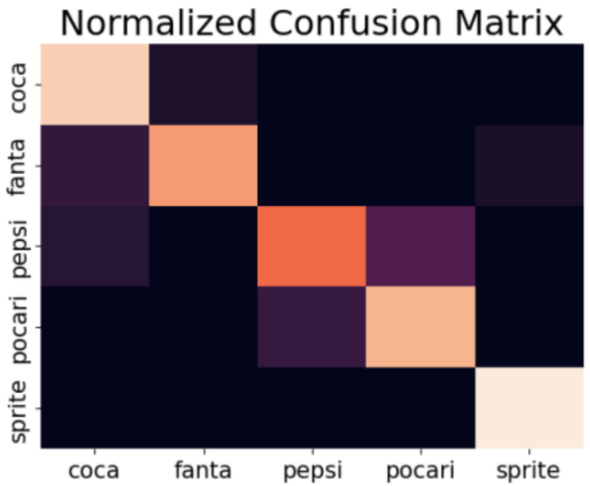
5. 결과 분석

1. 2 Convolution Layer

정밀도 & 재현율

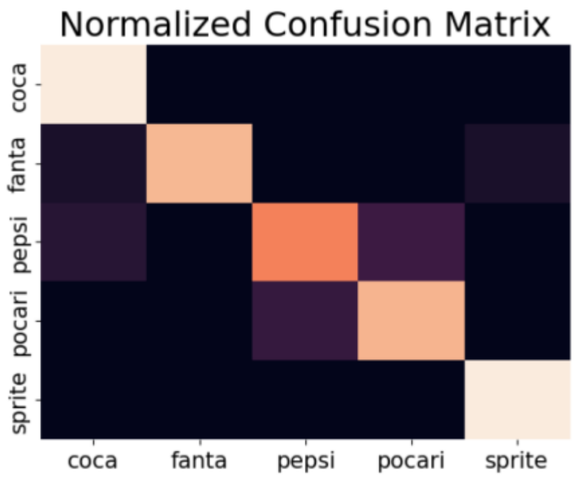
	precision	recall	f1-score	support
coca	0.75	0.92	0.83	13
fanta	0.92	0.80	0.86	15
pepsi	0.87	0.68	0.76	19
pocari	0.75	0.86	0.80	14
sprite	0.94	1.00	0.97	16

Confusion Matrix



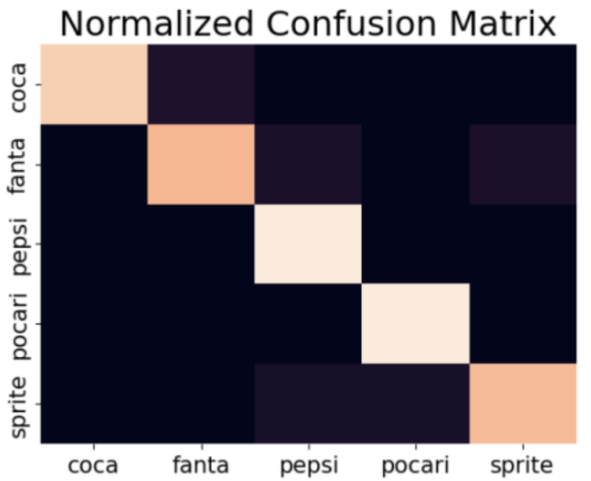
2. 3 Convolution Layer

	precision	recall	f1-score	support
coca	0.81	1.00	0.90	13
fanta	1.00	0.87	0.93	15
pepsi	0.88	0.74	0.80	19
pocari	0.80	0.86	0.83	14
sprite	0.94	1.00	0.97	16



3. ResNet152V2 (전이학습)

	precision	recall	f1-score	support
coca	1.00	0.92	0.96	13
fanta	0.93	0.87	0.90	15
pepsi	0.90	1.00	0.95	19
pocari	0.93	1.00	0.97	14
sprite	0.93	0.88	0.90	16



CHUNGBUK NATIONAL UNIVERSITY

5. 결과 분석

코카콜라 이미지 예측률

1. 2 Convolution Layer

```
1/1 [==============================] - 0s 89ms/step
```

예측률 : 100.00%

Class prediction = coca

2. 3 Convolution Layer

```
1/1 [==============================] - 0s 89ms/step
```

예측률 : 100.00%

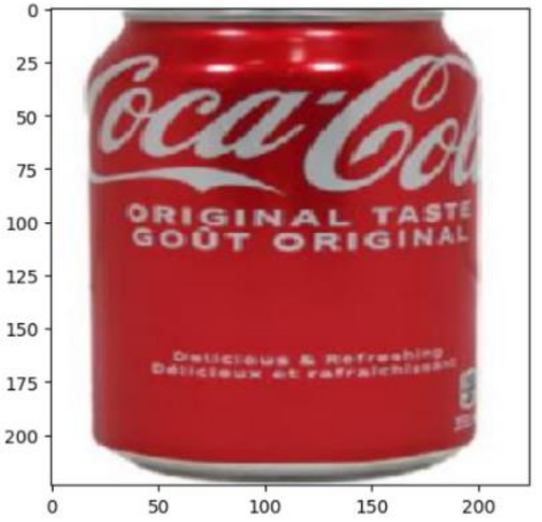
Class prediction = coca

3. ResNet152V2 (전이학습)

```
1/1 [==============================] - 3s 3s/step
```

예측률 : 97.78% ¶

Class prediction = coca



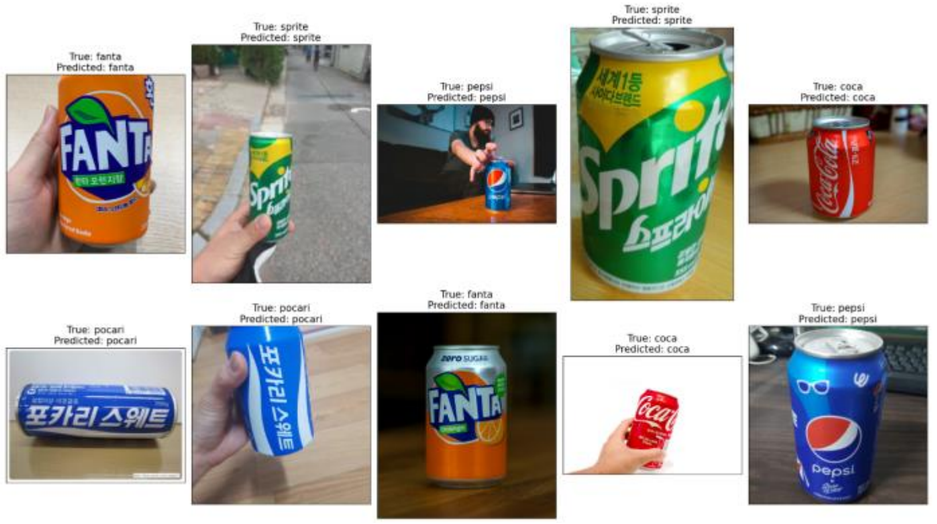
CHUNGBUK NATIONAL UNIVERSITY

5. 결과 분석

새로운 이미지 예측

1. 2 Convolution Layer

2. 3 Convolution Layer



CHUNGBUK NATIONAL UNIVERSITY

5. 결과 분석

새로운 이미지 예측

3. ResNet152V2 (전이학습)

6. 고찰

감사합니다.

정보통신공학부
2021039084 유승미