SPORTS IMAGE

START

스포츠 이미지 분류 모델 개발

김정명 / 김하영 / 유영상 / 한수정

00 INDEX

01
프로젝트 배경

02
데이터 소개

03
CNN 모델

04
전이학습 모델

001 EfficientNet

002 MobileNet

05
모델별 비교 및 시각화

06
결론 및 한계점

01 프로젝트 배경

[비바100] AI **딥러닝**, 인류 난치병 해결할 '해결사' 급부상

콘볼루션 신경망은 2차원 데이터 분석에 적합한 구조를 가져 **이미지 분류** 학습에 쓰이는 **딥러닝** 기술의 한 종류

딥러닝 모델 제작, 클릭 몇 번에 되네

엔비디아의 류현곤 시니어 솔루션 아키텍쳐 부장은 "이번에 소개한 **이미지 분류**는 **딥러닝**에서 기본이 되는 작업"이라며 "**이미지 분류**는 CCTV 영상을 실시간으로 ...

내시경 **이미지** 통해 귀 질환 진단하는 AI 개발

콘볼루션 신경망은 2차원 데이터 분석에 적합한 구조를 가져 **이미지 분류** 학습에 쓰이는 **딥러닝** 기술의 한 종류다. 학습에는 2013년 ~2019년 세브란스병원 이비...

"데이터가 부족해요"...**딥러닝**(DL) 통한 **이미지 분류**할 때 고려할 ...

이미지 분류를 **딥러닝**(DL)을 이용해 시도하려는데 데이터가 부족하면 제 성능을 발휘할 수 없다고 알고있다. 머신러닝(ML)과 달리 **딥러닝**은 **이미지**의 특성을 자...

소규모 데이터로 딥러닝을 시도할 때 고려할 점

수십개의 이미지 데이터 샘플로 복잡한 합성곱 신경망 훈련을 시도하는 것은 불가능할 것이지만, 만약 작업이 간단하고 모델이 작으며 일반화가 잘 되었다면 수백장의 이미지 데이터로도 가능할 수 있다.

가령 2,000장 정도의 개와 고양이 사진 데이터를 갖고 합성곱 신경망 훈련을 맨 처음부터 일반화(regularization) 없이 시도한다고 하면 약 70% 정도의 분류 정확도를 보여준다. 이 경우, 일반화 고려가 없었기 때문에 과적합(overfitting)이 발생한다.

데이터증식(data augmentation)을 통하여 정확도를 80~85% 정도까지 끌어 올릴 수 있다. 그 다음에 생각할 수 있는 방법으로는 대용량의 데이터를 기반으로 이미지 분류에 사전학습된 모델을 활용하는 전이학습(transfer learning)이다.

전이학습(transfer learning)

딥러닝에서 가장 강력한 아이디어 중에 하나는, 한 작업에 대해 학습을 이미 한 경우에, 그 지식을 다른 종류의 작업에 적용할 수 있는 것이다.

- **실생활에서 다양하게 쓰이는 이미지 데이터 분류**

  딥러닝을 통한 이미지 데이터 분류 학습은
  사회 여러 분야에서 효율적인 작업이 가능하도록 해준다.

- **전이학습을 통한 이미지 분류 성능 향상**

  CNN모델과 전이학습 모델 두 종류를 개발하여
  예측 결과를 가지고, 모델의 성능을 비교해본다.

02

데이터 소개

100 Sports Image Classification

- 구성

100가지의 스포츠 종목

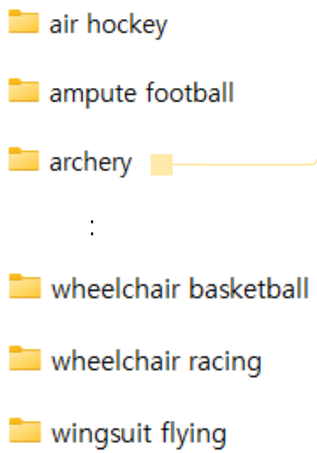
train images : 13,572장

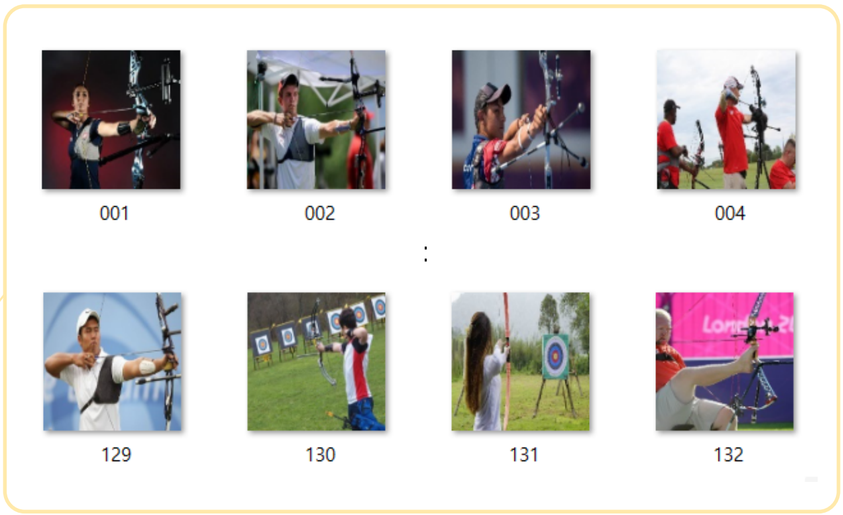
valid images : 500장

test images : 500장

- 출처

https://www.kaggle.com/datasets/gpiosenka/sports-classification



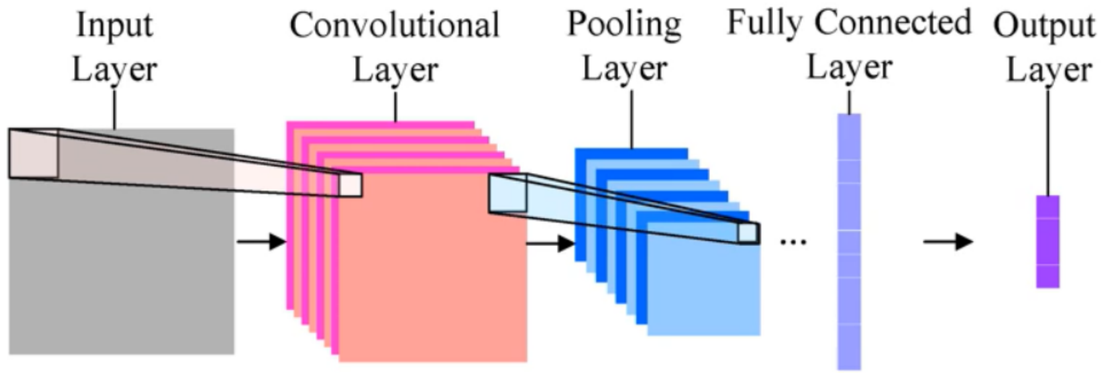


03

CNN 모델

CNN

데이터의 공간적 정보를 유지하면서
배열 데이터 정보를 다음 레이어로 보낼 수
있어 이미지(RGB 채널의 3차원 배열) 분야에서
적극 활용되고 있는 모델



- Input Layer : 입력층
- Convolutional Layer : 필터(커널)을 통해 가중치와 입력값을 곱한 후 활성화 함수를 취하여 이미지 특징 추출
- Pooling Layer : Convolutional Layer의 출력데이터의 크기를 줄이거나, 특정데이터 강조
- Fully Connected Layer : 1차원 배열의 형태로 평탄화된 행렬을 통해 이미지를 분류
- Output Layer : 출력층

03 CNN 모델

파라미터 설정

· 1차

```
# 파라미터 설정
image_size = (150, 150)
image_shape = (150, 150, 3)

num_classes = len(os.listdir(test_path))

batch = 100
epoch = 20
learning_rate = 0.001
```

· 2차

```
# 파라미터 설정
image_size = (224, 224)
image_shape = (224, 224, 3)

num_classes = len(os.listdir(test_path))

batch = 100
epoch = 20
learning_rate = 0.001
```

image_size, image_shape 150 → 224 수정
layer4 추가(filter:16)

· 3차

```
# 파라미터 설정
image_size = (224, 224)
image_shape = (224, 224, 3)

num_classes = len(os.listdir(test_path))

batch = 100
epoch = 30
learning_rate = 0.001
```

epoch = 20 → 30 수정

03 CNN 모델

이미지 생성기 객체 생성 & 이미지 생성

directory : 데이터 경로

target_size : 이미지 세로/가로 규격화 크기

color_mode : 이미지 구성 색상 → 컬러

class_mode : 종속변수 범주 분류 방법 → 다항분류

batch_size : 1회 학습 시 공급되는 데이터 개수

```
# 이미지 생성기 객체 생성
data_generator = image.ImageDataGenerator(rescale = 1./255)

# 훈련셋이미지 생성
train_generator = data_generator.flow_from_directory(directory= train_path,
                                                     target_size= image_size,
                                                     color_mode= 'rgb',
                                                     class_mode= 'categorical',
                                                     batch_size= batch)

# 검증셋이미지 생성
val_generator = data_generator.flow_from_directory(directory= val_path,
                                                   target_size= image_size,
                                                   color_mode= 'rgb',
                                                   class_mode= 'categorical',
                                                   batch_size= batch)

# 평가셋이미지 생성
test_generator = data_generator.flow_from_directory(directory= test_path,
                                                    target_size= image_size,
                                                    color_mode= 'rgb',
                                                    class_mode= 'categorical',
                                                    batch_size= batch,
                                                    shuffle= False)
```

03

CNN 모델

모델 생성

1) CNN model layer

· 1차

```
# 모델 생성
def cnn_model(image_shape):

    input_layer = Input(image_shape)
    x = layers.Conv2D(filters=32, kernel_size=(3, 3), activation="relu")(input_layer)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Conv2D(filters=64, kernel_size=(3, 3), activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Conv2D(filters=32, kernel_size=(3, 3), activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Flatten()(x)
    outputs = layers.Dense(num_classes, name="final_dense", activation='softmax')(x)
    return Model(input_layer, outputs)

model = cnn_model(image_shape)
```

· 2, 3차

(layer4 추가)

```
# 모델 생성
def cnn_model(image_shape):

    input_layer = Input(image_shape)
    x = layers.Conv2D(filters=32, kernel_size=(3, 3), activation="relu")(input_layer)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Conv2D(filters=64, kernel_size=(3, 3), activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Conv2D(filters=32, kernel_size=(3, 3), activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=(2, 2),strides=(2,2))(x)
    x = layers.Conv2D(filters=16, kernel_size=(3, 3), activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.Flatten()(x)
    outputs = layers.Dense(num_classes, name="final_dense", activation='softmax')(x)
    return Model(input_layer, outputs)

model = cnn_model(image_shape)
```

03

CNN 모델

모델 생성

2) model compilation

```
model.compile(optimizer = optimizers.Adam(learning_rate),
              loss = losses.categorical_crossentropy,
              metrics = ['accuracy'])
```

optimizer : 최적화 알고리즘 → Adam 알고리즘

loss : 손실함수 → 다항분류 모델 : categorical_crossentropy

metrics : 평가 방법 → accuracy(분류 정확도)

3) model training

```
# model training
history = model.fit(train_generator,
                    validation_data= val_generator,
                    epochs = epoch)
```

03 CNN 모델

모델 생성

4) model evaluation

```
# model evaluation
score = model.evaluate(test_generator)
print('acc = ', score[1], 'loss = ', score[0])
```

1차 : epoch = 20, batch = 100, lr = 0.001, image_size =(150, 150)

[학습 모델] loss: 0.0270 - accuracy: 0.9949
val_loss: 6.7876 - val_accuracy: 0.3620
[실제 평가] acc = 0.3860, loss = 6.5801

2차 : epoch = 20, batch = 100, lr = 0.001 , image_size =(150, 150), layer4 추가(filter:16)

[학습 모델] loss: 0.1030 - accuracy: 0.9709
val_loss: 7.1330 - val_accuracy: 0.3580
[실제 평가] acc =0.4080, loss = 6.6239

3차(최종) : epoch = 30, batch = 100, lr = 0.001, image_size =(224, 224)

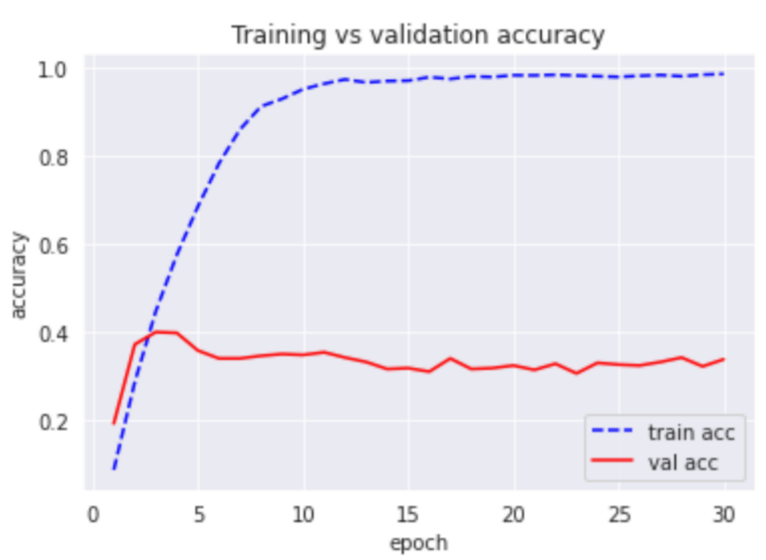
[학습 모델] loss: 0.0389 - accuracy: 0.9894
val_loss: 9.3787 - val_accuracy: 0.3240
[실제 평가] acc =0.3560, loss = 8.4496

03 CNN 모델

모델 생성

5) model history graph

```
# model history graph
plt.figure(figsize = (18,12))
plt.subplot(2,2,1)
plt.title('Training Loss')
plt.plot(history.history['loss'], color = 'tab:red')
plt.subplot(2,2,2)
plt.title('Training Accuracy')
plt.plot(history.history['accuracy'], color = 'tab:blue')
plt.subplot(2,2,3)
plt.title('Validation Loss')
plt.plot(history.history['val_loss'], color = 'tab:red')
plt.subplot(2,2,4)
plt.title('Validation Accuracy')
plt.plot(history.history['val_accuracy'], color = 'tab:blue')
plt.show()
```



03

CNN 모델

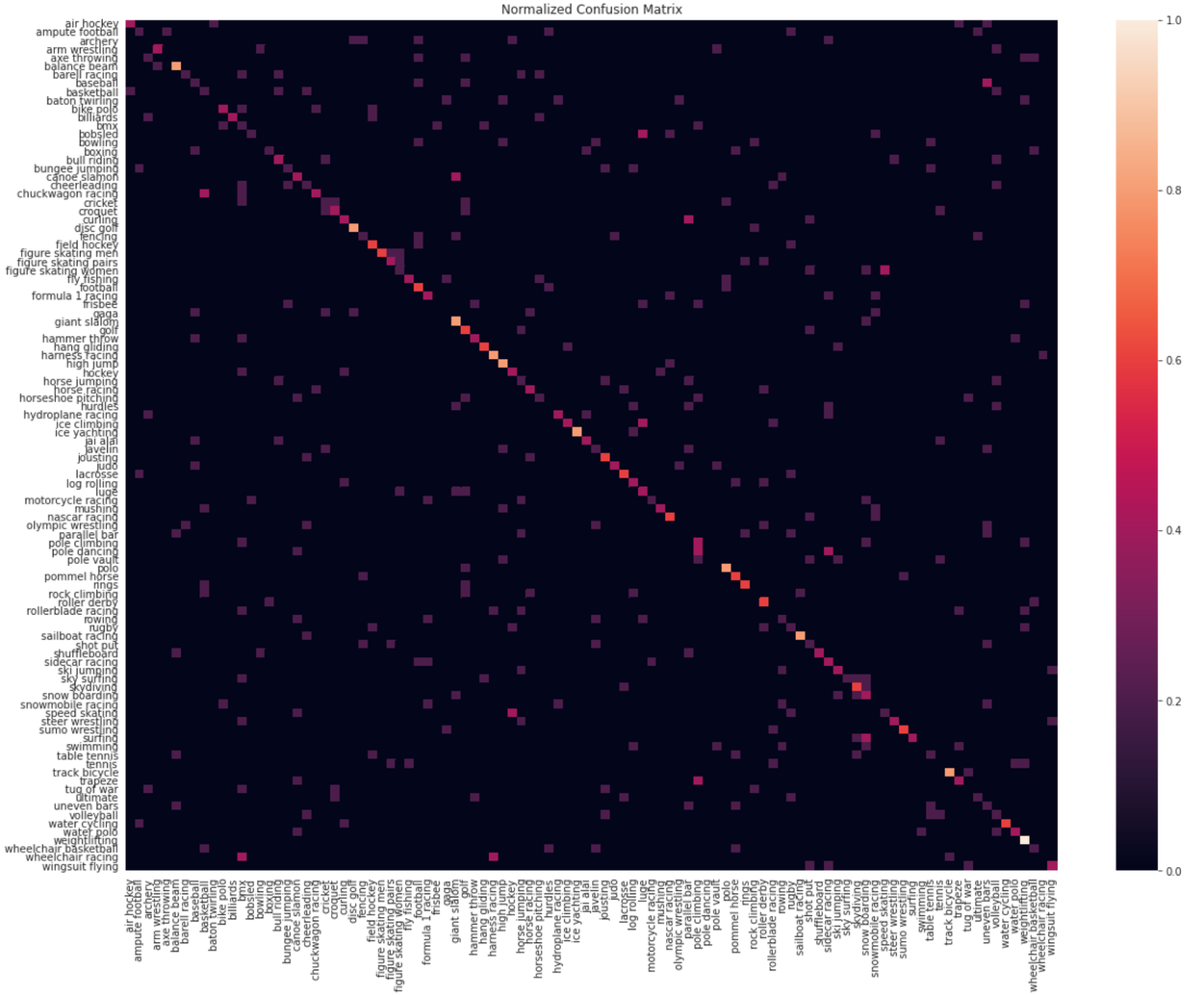
Classification Report

	precision	recall	f1-score	support
air hockey	0.33	0.40	0.36	5
ampute football	1.00	0.20	0.33	5
archery	0.00	0.00	0.00	5
arm wrestling	1.00	0.20	0.33	5
axe throwing	0.33	0.20	0.25	5
balance beam	0.75	0.60	0.67	5
barell racing	0.67	0.40	0.50	5
baseball	0.12	0.20	0.15	5
basketball	0.40	0.40	0.40	5
baton twirling	0.00	0.00	0.00	5
bike polo	0.67	0.40	0.50	5
billiards	0.62	1.00	0.77	5
bmx	0.00	0.00	0.00	5
bobsled	0.10	0.20	0.13	5
bowling	0.00	0.00	0.00	5
boxing	0.83	1.00	0.91	5
bull riding	0.43	0.60	0.50	5
bungee jumping	0.50	0.40	0.44	5
canoe slamon	0.33	0.40	0.36	5
cheerleading	0.00	0.00	0.00	5
chuckwagon racing	0.60	0.60	0.60	5
cricket	0.57	0.80	0.67	5
croquet	0.67	0.40	0.50	5
curling	0.50	0.40	0.44	5
disc golf	0.17	0.20	0.18	5
fencing	0.50	0.40	0.44	5
field hockey	0.29	0.40	0.33	5
figure skating men	0.56	1.00	0.71	5
figure skating pairs	0.33	0.20	0.25	5
figure skating women	0.75	0.60	0.67	5
fly fishing	0.67	0.40	0.50	5
football	0.12	0.20	0.15	5
formula 1 racing	0.00	0.00	0.00	5
frisbee	0.00	0.00	0.00	5
gaga	0.17	0.20	0.18	5
giant slalom	0.50	0.60	0.55	5
golf	0.29	0.40	0.33	5
hammer throw	1.00	0.20	0.33	5
hang gliding	0.33	0.40	0.36	5
harness racing	0.75	0.60	0.67	5
high jump	0.67	0.80	0.73	5
hockey	0.50	0.60	0.55	5
horse jumping	0.40	0.40	0.40	5
horse racing	0.50	0.80	0.62	5
horseshoe pitching	0.00	0.00	0.00	5
hurdles	0.00	0.00	0.00	5
hydroplane racing	0.50	0.40	0.44	5
ice climbing	0.50	0.40	0.44	5
ice yachting	1.00	0.80	0.89	5
jai alai	0.40	0.40	0.40	5
javelin	0.00	0.00	0.00	5
jousting	0.50	0.20	0.29	5
judo	0.50	0.60	0.55	5
lacrosse	0.25	0.20	0.22	5
log rolling	1.00	0.20	0.33	5
luge	0.25	0.20	0.22	5
motorcycle racing	0.67	0.40	0.50	5
mushing	0.75	0.60	0.67	5
nascar racing	0.50	0.80	0.62	5
olympic wrestling	0.40	0.40	0.40	5
parallel bar	0.33	0.20	0.25	5
pole climbing	0.29	0.40	0.33	5
pole dancing	0.00	0.00	0.00	5
pole vault	0.25	0.20	0.22	5
polo	0.57	0.80	0.67	5
pommel horse	0.57	0.80	0.67	5
rings	0.50	0.60	0.55	5
rock climbing	0.25	0.40	0.31	5
roller derby	0.40	0.40	0.40	5
rollerblade racing	0.17	0.20	0.18	5
rowing	1.00	0.20	0.33	5
rugby	0.25	0.40	0.31	5
sailboat racing	0.62	1.00	0.77	5
shot put	0.33	0.20	0.25	5
shuffleboard	1.00	0.60	0.75	5
sidecar racing	0.38	0.60	0.46	5
ski jumping	0.40	0.40	0.40	5
sky surfing	0.50	0.20	0.29	5
skydiving	0.22	0.40	0.29	5
snow boarding	0.00	0.00	0.00	5
snowmobile racing	0.50	0.60	0.55	5
speed skating	0.00	0.00	0.00	5
steer wrestling	0.50	0.40	0.44	5
sumo wrestling	0.25	0.40	0.31	5
surfing	0.50	0.40	0.44	5
swimming	0.67	0.40	0.50	5
table tennis	0.25	0.40	0.31	5
tennis	0.00	0.00	0.00	5
track bicycle	0.50	1.00	0.67	5
trapeze	0.25	0.40	0.31	5
tug of war	0.10	0.20	0.13	5
ultimate	0.33	0.20	0.25	5
uneven bars	0.40	0.40	0.40	5
volleyball	0.10	0.20	0.13	5
water cycling	1.00	0.60	0.75	5
water polo	0.50	0.40	0.44	5
weightlifting	0.50	1.00	0.67	5
wheelchair basketball	1.00	0.40	0.57	5
wheelchair racing	0.00	0.00	0.00	5
wingsuit flying	0.33	0.20	0.25	5
accuracy			0.38	500
macro avg	0.42	0.38	0.37	500
weighted avg	0.42	0.38	0.37	500

03

CNN 모델

Confusion Matrix 시각화



스포츠 이미지 분류 모델 개발

03

CNN 모델

오분류 예측 예시

오분류된 테스트 이미지 개수

: 308 개 이미지

04

EfficientNet

전이학습모델 1) EfficientNet

모델의 규모를 키워 정확도를 올리는 방법

1) 필터 수 늘리기 (b)

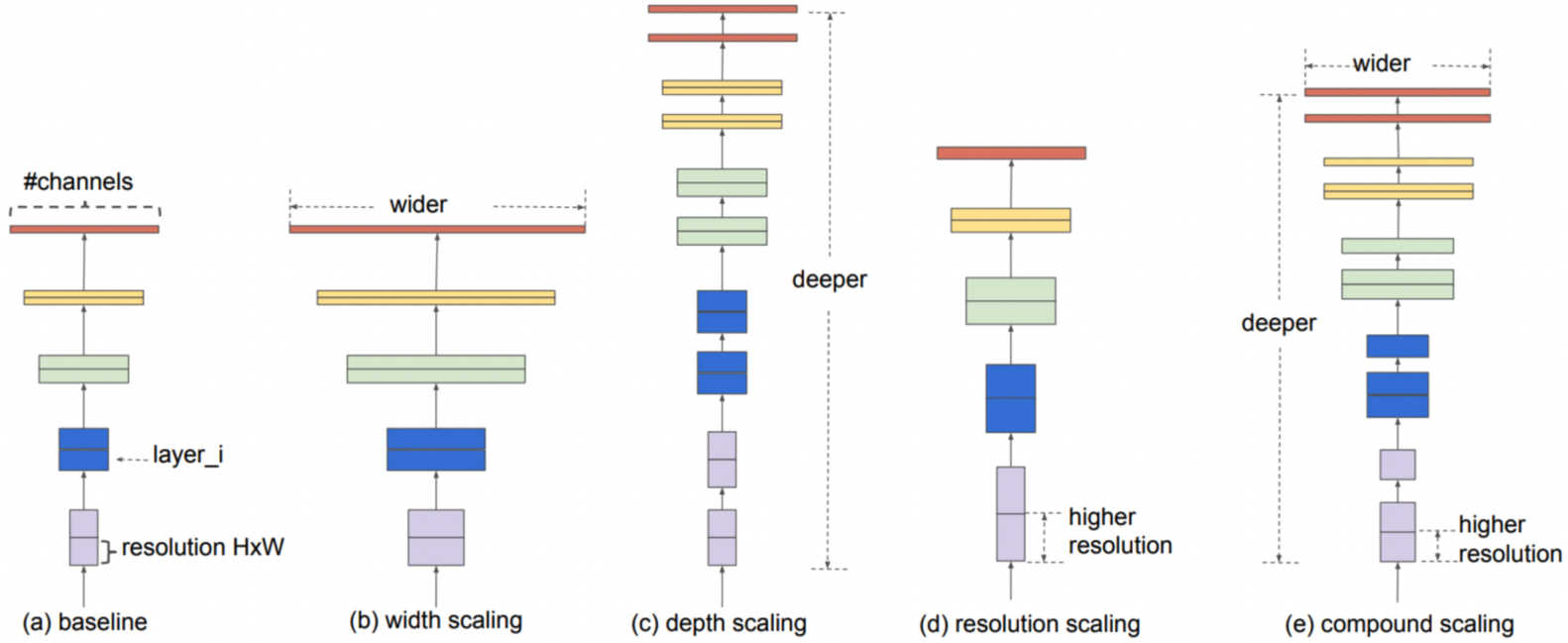
2) 레이어 수 늘리기 (c)

3) 입력 이미지 크기 늘리기 (d)

EfficientNet

위의 width, depth, resolution을

모두 고려한 Compound Scaling (e)



04

EfficientNet

전이학습모델 1) EfficientNet

파라미터 설정

```
# 파라미터 설정
img_height = 224
img_width = 224
input_shape = (img_height, img_width, 3)

num_classes = 100

epochs = 30
batch_size = 64
callbacks = EarlyStopping(monitor='val_loss', patience=10)
```

04

EfficientNet

전이학습모델 1) EfficientNet

이미지 증식

rotation_range : 회전 각도 범위 → 음수: 반시계 방향, 양수: 시계 방향

width_shift_range : 수평 이동 범위 → 음수: 왼쪽 방향, 양수: 오른쪽 방향

height_shift_range : 수직 이동 범위 → 음수: 아래쪽 방향, 양수: 위쪽 방향

shear_range : 전단 (이미지 자르기) 각도 범위

zoom_range : 줌인, 줌아웃 최대 범위

fill_mode : 배경색 채우는 방법

* (neartest : 이미지 가장자리 색상 사용)

horizontal_flip : 좌우 반전

vertical_flip : 상하 반전

brightness_range : 밝기 조절

```
train_data = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.3,
    fill_mode='nearest',
    horizontal_flip=True,
    vertical_flip=False,
    brightness_range=(0.75,1.25)
    )
```

04

EfficientNet

전이학습모델 1) EfficientNet

모델링

1) Base Model 생성

```
# 전이학습 : EfficientNetB0 사용

base_model = EfficientNetB0(weights='imagenet',
                            input_shape= input_shape,
                            include_top = False)
base_model.trainable = True

for layer in base_model.layers[:200]:
  layer.trainable = False
```

```
=================================
Total params: 4,049,564
Trainable params: 1,496,160
Non-trainable params: 2,553,404
_________________________________
```

include_top = False

: 모델의 끝단인 classifier 부분을

학습하고자 하는 데이터에 맞춰서 수정해주기 위해 False로 설정

base_model.trainable = True

: 베이스 모델도 학습할 수 있게 설정

base_model.layers[:200]

: 베이스 모델에서 200층까지는 가중치 값을 조정 하지 않고,

그 다음부터 조정할 수 있게 설정 (전체 레이어 수 : 230)

layer.trainable = False

: 학습해야 할 가중치를 '학습가능 상태'로 설정

* True로 설정할 경우 : 가중치를 freeze 한 상태

(학습이 진행되지 않게 설정해주는 것)

04

EfficientNet

전이학습모델 1) EfficientNet

모델링

2) Fine Tuning

```
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation = 'softmax'))
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b0 (Model)      (None, 7, 7, 1280)        4049564
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1280)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_3 (Dense)              (None, 100)               128100
=================================================================
Total params: 4,177,664
Trainable params: 1,624,260
Non-trainable params: 2,553,404
_________________________________________________________________
```

GlobalAveragePooling2D

: 각 Feature Map 상의 노드값들의 평균을 뽑아내는 방식 사용

* AveragePooling2D : pool size 크기의 윈도우가 영상 안을 stride만큼 이동하면서 윈도우 값의 평균을 출력
* GlobalAveragePooling2D : n개의 채널을 평균하여 하나의 값으로 표현

Dropout(0.2)

: 학습 과정에서 신경망의 일부를 사용하지 않는 방법

학습 과정마다 랜덤으로 뉴런의 20%는 사용하지 않고, 80%만 학습하게 설정 (과적합 방지)

Dense(num_class, activation = 'softmax')

: 모델의 끝단에 classifier 추가

1) num_class : 전체 클래스의 갯수인 100개로 설정
2) activation : 다항분류이므로 softmax 사용

04

EfficientNet

전이학습모델 1) EfficientNet

모델링

3) Model Training

```
callbacks = EarlyStopping(monitor='val_loss', patience=10)

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

history = model.fit(train_generator,
                    validation_data = validation_generator,
                    callbacks = [callbacks],
                    epochs = epochs, verbose = 1)
```

```
Epoch 16/30
213/213 [==============================] - 179s 840ms/step - loss: 0.0880 - accuracy: 0.9730 - val_loss: 0.9671 - val_accuracy: 0.8540
Epoch 17/30
213/213 [==============================] - 180s 844ms/step - loss: 0.1080 - accuracy: 0.9712 - val_loss: 0.4405 - val_accuracy: 0.8440
Epoch 18/30
213/213 [==============================] - 179s 840ms/step - loss: 0.0966 - accuracy: 0.9694 - val_loss: 0.5852 - val_accuracy: 0.8340
Epoch 19/30
213/213 [==============================] - 180s 843ms/step - loss: 0.0886 - accuracy: 0.9744 - val_loss: 0.8782 - val_accuracy: 0.8160
Epoch 20/30
213/213 [==============================] - 178s 837ms/step - loss: 0.1085 - accuracy: 0.9700 - val_loss: 0.6309 - val_accuracy: 0.8360
```

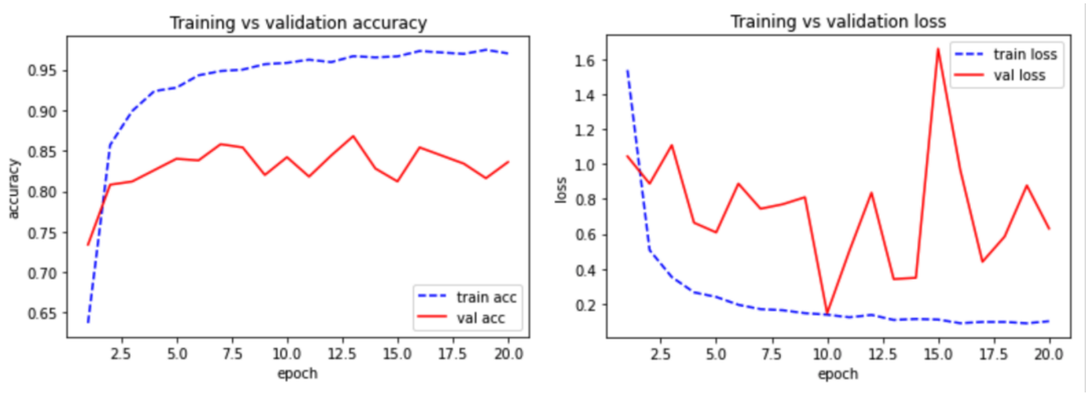
20에폭에서 학습 종료

04

EfficientNet

전이학습모델 1) EfficientNet

History 시각화



모델 평가

```
# test data로 확인

score = model.evaluate(test_generator)
print('acc = ', score[1], 'loss = ', score[0])
'''
acc =  0.871999979019165 loss =  0.47893810272216797
'''
```

04

EfficientNet

전이학습모델 1) EfficientNet

Classification Report

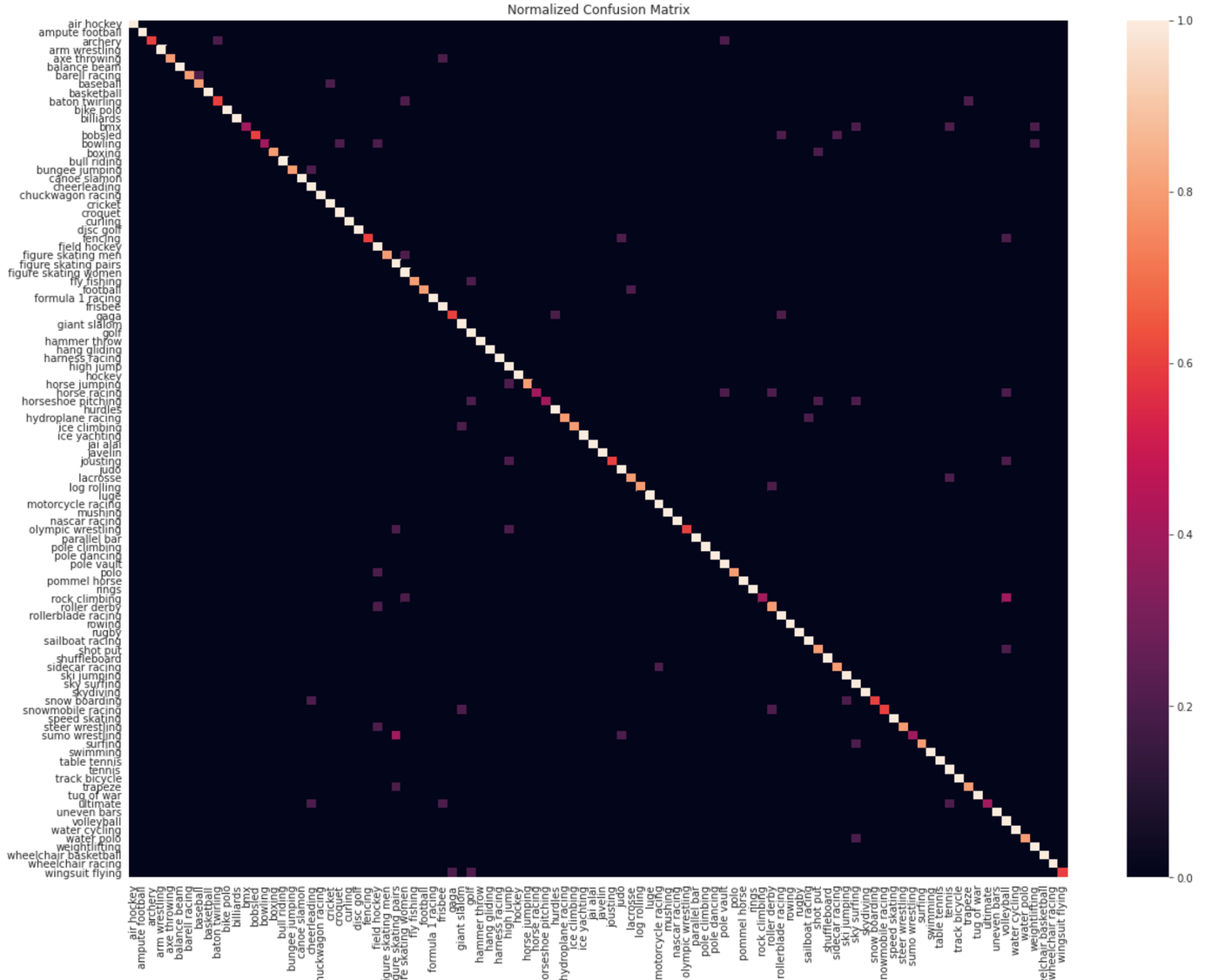
	precision	recall	f1-score	support
air hockey	1.00	1.00	1.00	5
ampute football	1.00	1.00	1.00	5
archery	1.00	0.60	0.75	5
arm wrestling	1.00	1.00	1.00	5
axe throwing	1.00	0.80	0.89	5
balance beam	1.00	1.00	1.00	5
barell racing	1.00	0.80	0.89	5
baseball	0.80	0.80	0.80	5
basketball	1.00	1.00	1.00	5
baton twirling	0.75	0.60	0.67	5
bike polo	1.00	1.00	1.00	5
billiards	1.00	1.00	1.00	5
bmx	1.00	0.40	0.57	5
bobsled	1.00	0.60	0.75	5
bowling	1.00	0.40	0.57	5
boxing	1.00	0.80	0.89	5
bull riding	1.00	1.00	1.00	5
bungee jumping	1.00	0.80	0.89	5
canoe slamon	1.00	1.00	1.00	5
cheerleading	0.62	1.00	0.77	5
chuckwagon racing	1.00	1.00	1.00	5
cricket	0.83	1.00	0.91	5
croquet	0.83	1.00	0.91	5
curling	1.00	1.00	1.00	5
disc golf	1.00	1.00	1.00	5
fencing	1.00	0.60	0.75	5
field hockey	0.56	1.00	0.71	5
figure skating men	1.00	0.80	0.89	5
figure skating pairs	0.56	1.00	0.71	5
figure skating women	0.62	1.00	0.77	5
fly fishing	1.00	0.80	0.89	5
football	1.00	0.80	0.89	5
formula 1 racing	1.00	1.00	1.00	5
frisbee	0.71	1.00	0.83	5
gaga	0.75	0.60	0.67	5
giant slalom	0.71	1.00	0.83	5
golf	0.62	1.00	0.77	5
hammer throw	1.00	1.00	1.00	5
hang gliding	1.00	1.00	1.00	5
harness racing	1.00	1.00	1.00	5
high jump	0.62	1.00	0.77	5
hockey	1.00	1.00	1.00	5
horse jumping	1.00	0.80	0.89	5
horse racing	1.00	0.40	0.57	5
horseshoe pitching	1.00	0.40	0.57	5
hurdles	0.83	1.00	0.91	5
hydroplane racing	1.00	0.80	0.89	5
ice climbing	1.00	0.80	0.89	5
ice yachting	1.00	1.00	1.00	5
jai alai	1.00	1.00	1.00	5
javelin	1.00	1.00	1.00	5
jousting	1.00	0.60	0.75	5
judo	0.71	1.00	0.83	5
lacrosse	0.80	0.80	0.80	5
log rolling	1.00	0.80	0.89	5
luge	1.00	1.00	1.00	5
motorcycle racing	0.83	1.00	0.91	5
mushing	1.00	1.00	1.00	5
nascar racing	1.00	1.00	1.00	5
olympic wrestling	1.00	0.60	0.75	5
parallel bar	1.00	1.00	1.00	5
pole climbing	1.00	1.00	1.00	5
pole dancing	1.00	1.00	1.00	5
pole vault	0.71	1.00	0.83	5
polo	1.00	0.80	0.89	5
pommel horse	1.00	1.00	1.00	5
rings	1.00	1.00	1.00	5
rock climbing	1.00	0.40	0.57	5
roller derby	0.57	0.80	0.67	5
rollerblade racing	0.71	1.00	0.83	5
rowing	1.00	1.00	1.00	5
rugby	1.00	1.00	1.00	5
sailboat racing	0.83	1.00	0.91	5
shot put	0.67	0.80	0.73	5
shuffleboard	1.00	1.00	1.00	5
sidecar racing	0.80	0.80	0.80	5
ski jumping	0.83	1.00	0.91	5
sky surfing	0.56	1.00	0.71	5
skydiving	1.00	1.00	1.00	5
snow boarding	1.00	0.60	0.75	5
snowmobile racing	1.00	0.60	0.75	5
speed skating	1.00	1.00	1.00	5
steer wrestling	1.00	0.80	0.89	5
sumo wrestling	1.00	0.40	0.57	5
surfing	1.00	0.80	0.89	5
swimming	1.00	1.00	1.00	5
table tennis	1.00	1.00	1.00	5
tennis	0.62	1.00	0.77	5
track bicycle	1.00	1.00	1.00	5
trapeze	0.80	0.80	0.80	5
tug of war	1.00	1.00	1.00	5
ultimate	1.00	0.40	0.57	5
uneven bars	1.00	1.00	1.00	5
volleyball	0.45	1.00	0.62	5
water cycling	1.00	1.00	1.00	5
water polo	1.00	0.80	0.89	5
weightlifting	0.71	1.00	0.83	5
wheelchair basketball	1.00	1.00	1.00	5
wheelchair racing	1.00	1.00	1.00	5
wingsuit flying	1.00	0.60	0.75	5
accuracy			0.88	500
macro avg	0.91	0.88	0.87	500
weighted avg	0.91	0.88	0.87	500

04

EfficientNet

전이학습모델 1) EfficientNet

Confusion Matrix 시각화



04

EfficientNet

전이학습모델 1) EfficientNet

오분류 예측 예시

오분류된 테스트 이미지 개수

: 62 개 이미지

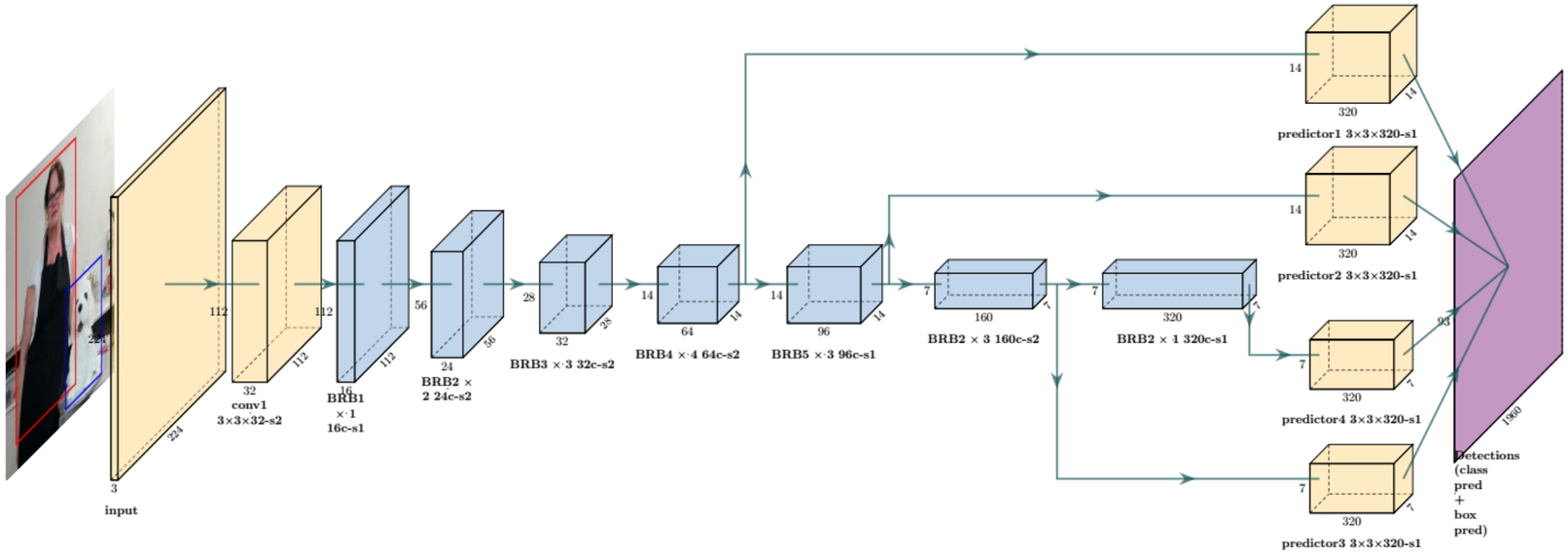
04

MobileNet

전이학습모델 2) MobileNet

MobileNet

연산량 및 모델 크기 축소하여 모바일, 디바이스 등 제한된 환경 내에서도 이미지 예측 가능한 모델



04

MobileNet

전이학습모델 2) MobileNet

파라미터 설정

```
# 파라미터 설정
image_size = (224, 224)
image_shape = (224, 224, 3)

num_classes = len(os.listdir(test_path))

epoch = 30
batch = 100
learning_rate = 0.001
```

→ len(os.listdir(test_path) : 이미지 파일 개수 = 이미지 종류 개수

04

MobileNet

전이학습모델 2) MobileNet

이미지 증식

rotation_range : 회전 각도 범위 → 음수: 반시계 방향, 양수: 시계 방향

zoom_range: 축소/확대 범위 → 1 미만인 경우 축소, 1 초과인 경우 확대

width_shift_range : 수평 이동 범위 → 음수: 왼쪽 방향, 양수: 오른쪽 방향

height_shift_range : 수직 이동 범위 → 음수: 아래쪽 방향, 양수: 위쪽 방향

shear_range : 전단 (이미지 자르기) 각도 범위

horizontal_flip : True면 수평 뒤집기 (좌우반전), False면 뒤집지 않음

fill_mode : 배경 색상 채우는 방법 → "nearest" : 이미지 가장자리 색상 사용

preprocessing_function : 모델에 데이터 입력 시, 사용되는 함수

```
train_data_generator = image.ImageDataGenerator(
        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",
        preprocessing_function=preprocess_input
        )
```

04

MobileNet

전이학습모델 2) MobileNet

데이터 전처리

```
train_generator = train_data_generator.flow_from_directory(directory= train_path,
                                                           target_size= image_size,
                                                           color_mode= 'rgb',
                                                           class_mode= 'categorical',
                                                           batch_size= batch)
```

directory : 데이터 경로

target_size : 이미지 세로/가로 규격화 크기

color_mode : 이미지 구성 색상 → 흑백 : 'grayscale', 컬러 : 'rgb'

class_mode : 종속변수(y) 범주 분류 방법 →'binary' : 이항분류, 'categorical' : 다항분류

batch_size : 1회 학습 시, 사용되는 데이터 개수

04

MobileNet

전이학습모델 2) MobileNet

모델 생성

```
class ModifiedMobileNet():
    '''
    This class creates the mobilenet model.
    '''
    def __init__(self, input_shape, nb_classes):
        self.input_shape = input_shape
        self.nb_classes = nb_classes

    def get_model(self, unfreeze_layers = None, lr_rate = 0.001):
        # load base model
        base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=self.input_shape)

        # freezing the layers
        for layer in (base_model.layers) if not unfreeze_layers else (base_model.layers[:-int(unfreeze_layers)]):
            layer.trainable = False

        inputs = Input(shape=self.input_shape)
        x = base_model(inputs, training=False)
        x = layers.GlobalAveragePooling2D()(x)
        x = layers.Dense(128, activation='relu')(x)
        x = layers.Dropout(0.1)(x)
        x = layers.Dense(128, activation='relu')(x)
        x = layers.Dropout(0.1)(x)
        outputs = layers.Dense(self.nb_classes, activation='softmax')(x)
        model = Model(inputs, outputs)

        # model compilation
        optimizer = optimizers.Adam(learning_rate=lr_rate)
        model.compile(loss='categorical_crossentropy', optimizer= optimizer, metrics=['accuracy'])
        return model
```

04

MobileNet

전이학습모델 2) MobileNet

모델 생성

1) Base Model 생성

```
# load base model
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=self.input_shape)

# freezing the layers
for layer in (base_model.layers) if not unfreeze_layers else (base_model.layers[:-int(unfreeze_layers)]):
    layer.trainable = False
```

① Load Base Model

weight : 지정된 훈련 데이터로 학습된 가중치

include_top : True면 최상위층(전결합층~출력층) 사용, False면 사용하지 않음

input_shape : 현재 이미지 모양

② Freezing the Layers : 가중치 고정

trainable : True인 경우 새로운 모델에 적용 시 가중치 고정, False인 경우 새로운 모델 훈련 중 가중치 수정

04

MobileNet

전이학습모델 2) MobileNet

모델 생성

2) Fine Tuning

```
# Fine Tuning
inputs = Input(shape=self.input_shape)
x = base_model(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.1)(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.1)(x)
outputs = layers.Dense(self.nb_classes, activation='softmax')(x)
model = Model(inputs, outputs)
```

base_model

inputs : 이미지 모양

training : True면 가중치 수정 가능, False면 가중치 수정 불가능

GlobalAveragePooling2D : 특징맵 노드값의 평균 이용한 이미지 축소

Dropout : 뉴런 개수 무작위 제거비율 (과적합 방지)

→ 전체 중 '(1-제거비율)*100'%의 네트워크만 사용

Dense : 은닉층 혹은 출력층

units : 은닉층 혹은 출력층 뉴런 개수 → 출력층의 경우, 이미지 종류 지정

activation : 활성함수

→ 은닉층 : 'relu', 출력층 다항분류 : 'softmax' 또는 이항분류 : 'sigmoid'

04

MobileNet

전이학습모델 2) MobileNet

모델 생성

3) Model Training

```
mobilenet = ModifiedMobileNet(input_shape= image_shape, nb_classes= num_classes)
model = mobilenet.get_model(unfreeze_layers= 0, lr_rate= learning_rate)

model.compile(optimizer = optimizers.Adam(learning_rate),
              loss = losses.categorical_crossentropy,
              metrics = ['accuracy'])
history = model.fit(train_generator, validation_data= val_generator, epochs = epoch)
```

① Model Compilation (학습환경 설정)

loss : 손실 함수 (모델 최적화 시, 가중치 결정에 사용)

→ 분류모델: 'binary_crossentropy'

혹은 'categorical_crossentropy'(다항)

optimizer : 최적화 알고리즘 → Adam

learning_rate : 직접 학습률 지정

metrics : 평가 방법 (최적화된 모델 성능 확인)

→ 'accuracy' : 분류 정확도

② Model Fitting (모델 생성)

epochs: 학습 횟수

04

MobileNet

전이학습모델 2) MobileNet

훈련 및 검증 결과

```
Epoch 26/30
136/136 [==============================] - 159s 1s/step - loss: 0.4925 - accuracy: 0.8452 - val_loss: 0.4176 - val_accuracy: 0.8720
Epoch 27/30
136/136 [==============================] - 160s 1s/step - loss: 0.4902 - accuracy: 0.8470 - val_loss: 0.4638 - val_accuracy: 0.8600
Epoch 28/30
136/136 [==============================] - 161s 1s/step - loss: 0.4830 - accuracy: 0.8479 - val_loss: 0.4142 - val_accuracy: 0.8640
Epoch 29/30
136/136 [==============================] - 163s 1s/step - loss: 0.4903 - accuracy: 0.8493 - val_loss: 0.4420 - val_accuracy: 0.8680
Epoch 30/30
136/136 [==============================] - 158s 1s/step - loss: 0.4742 - accuracy: 0.8505 - val_loss: 0.4589 - val_accuracy: 0.8720
```

평가 결과

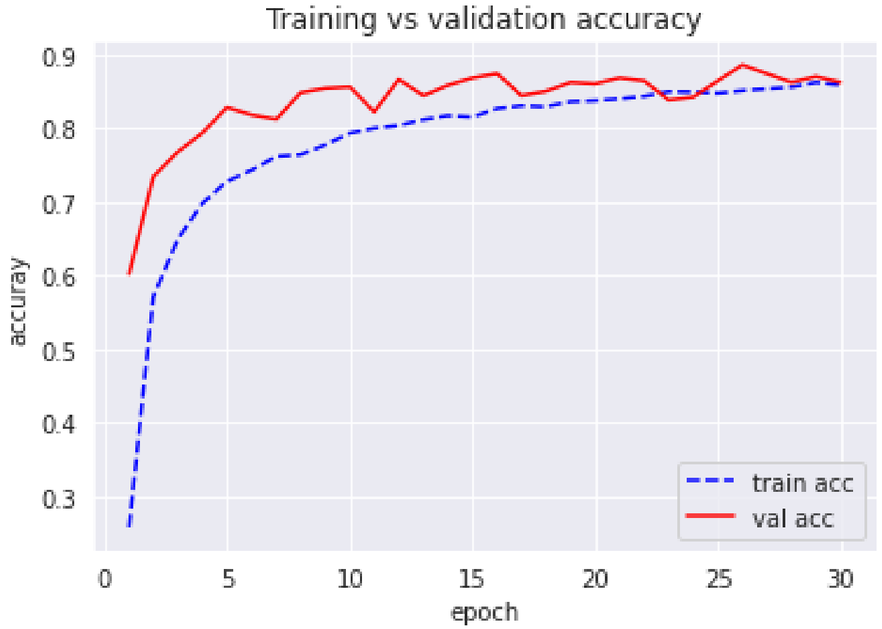
```
# model evaluation
score2 = model.evaluate(test_generator)
print('acc = ', score2[1], 'loss = ', score2[0])
# acc =  0.9160000085830688 loss =  0.25562530755996704
```

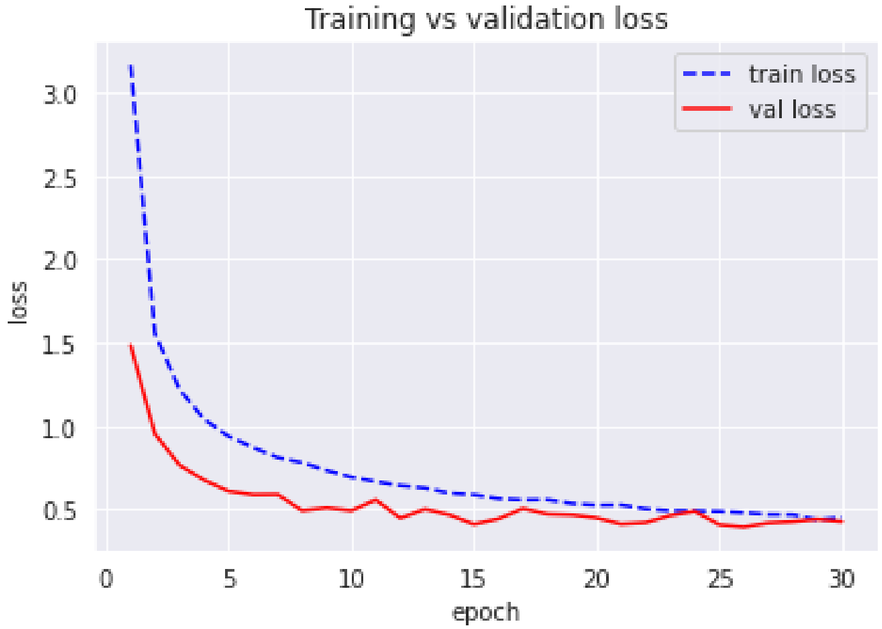
04

MobileNet

전이학습모델 2) MobileNet

History 시각화





04

MobileNet

전이학습모델 2) MobileNet

ClassificationReport

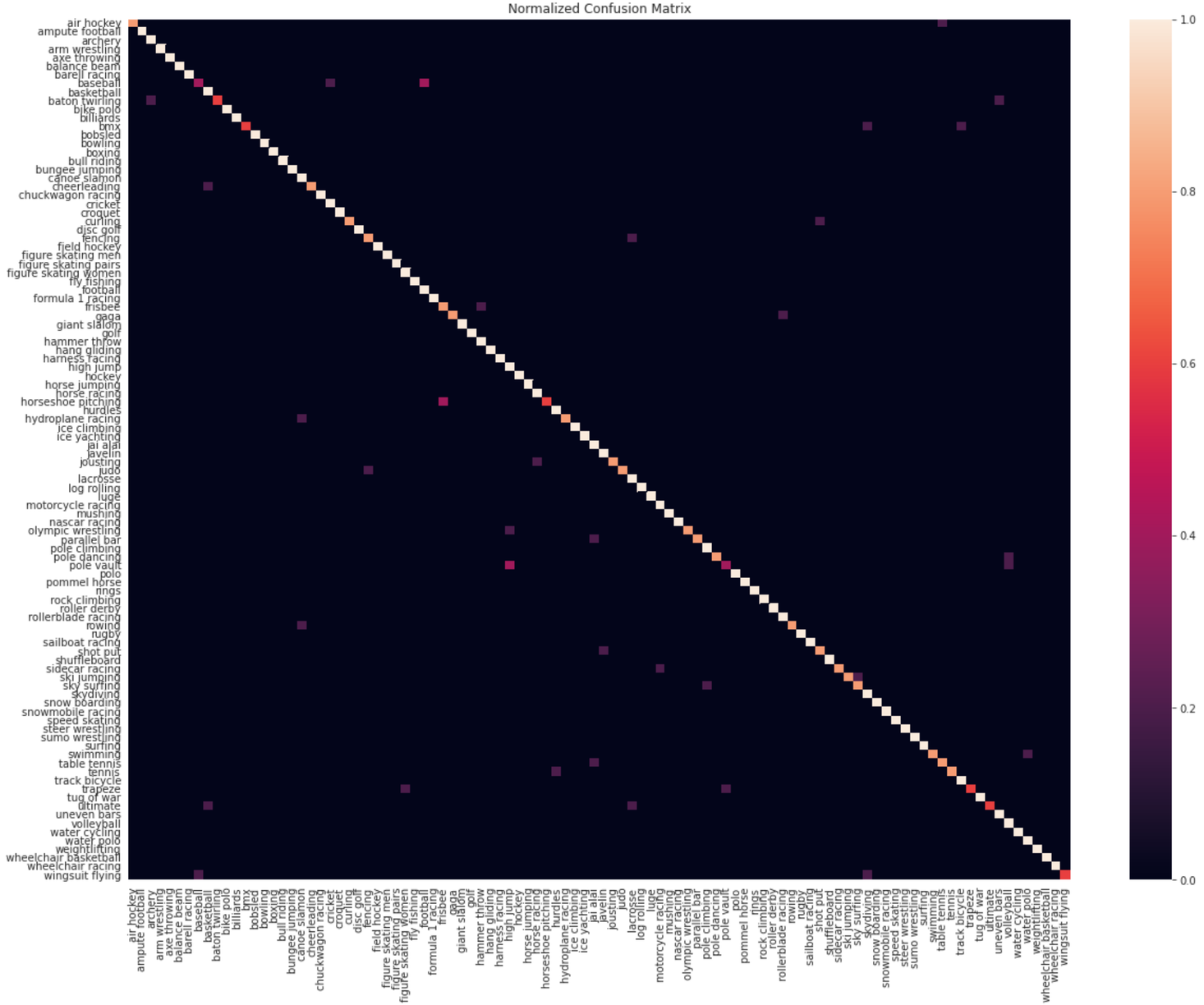
	precision	recall	f1-score	support
air hockey	1.00	0.80	0.89	5
ampute football	1.00	1.00	1.00	5
archery	0.83	1.00	0.91	5
arm wrestling	1.00	1.00	1.00	5
axe throwing	1.00	1.00	1.00	5
balance beam	1.00	1.00	1.00	5
barell racing	1.00	1.00	1.00	5
baseball	0.67	0.40	0.50	5
basketball	0.71	1.00	0.83	5
baton twirling	1.00	0.60	0.75	5
bike polo	1.00	1.00	1.00	5
billiards	1.00	1.00	1.00	5
bmx	1.00	0.60	0.75	5
bobsled	1.00	1.00	1.00	5
bowling	1.00	1.00	1.00	5
boxing	1.00	1.00	1.00	5
bull riding	1.00	1.00	1.00	5
bungee jumping	1.00	1.00	1.00	5
canoe slamon	0.71	1.00	0.83	5
cheerleading	1.00	0.80	0.89	5
chuckwagon racing	1.00	1.00	1.00	5
cricket	0.83	1.00	0.91	5
croquet	1.00	1.00	1.00	5
curling	1.00	0.80	0.89	5
disc golf	1.00	1.00	1.00	5
fencing	0.80	0.80	0.80	5
field hockey	1.00	1.00	1.00	5
figure skating men	1.00	1.00	1.00	5
figure skating pairs	1.00	1.00	1.00	5
figure skating women	0.83	1.00	0.91	5
fly fishing	1.00	1.00	1.00	5
football	0.71	1.00	0.83	5
formula 1 racing	1.00	1.00	1.00	5
frisbee	0.67	0.80	0.73	5
gaga	1.00	0.80	0.89	5
giant slalom	1.00	1.00	1.00	5
golf	1.00	1.00	1.00	5
hammer throw	0.83	1.00	0.91	5
hang gliding	1.00	1.00	1.00	5
harness racing	1.00	1.00	1.00	5
high jump	0.62	1.00	0.77	5
hockey	1.00	1.00	1.00	5
horse jumping	1.00	1.00	1.00	5
horse racing	0.83	1.00	0.91	5
horseshoe pitching	1.00	0.60	0.75	5
hurdles	0.83	1.00	0.91	5
hydroplane racing	1.00	0.80	0.89	5
ice climbing	1.00	1.00	1.00	5
ice yachting	1.00	1.00	1.00	5
jai alai	0.71	1.00	0.83	5
javelin	0.83	1.00	0.91	5
jousting	1.00	0.80	0.89	5
judo	1.00	0.80	0.89	5
lacrosse	0.71	1.00	0.83	5
log rolling	1.00	1.00	1.00	5
luge	1.00	1.00	1.00	5
motorcycle racing	0.83	1.00	0.91	5
mushing	1.00	1.00	1.00	5
nascar racing	1.00	1.00	1.00	5
olympic wrestling	1.00	0.80	0.89	5

04

MobileNet

전이학습모델 2) MobileNet

ConfusionMatrix 시각화



스포츠 이미지 분류 모델 개발

04

MobileNet

전이학습모델 2) MobileNet

오분류 예측 예시

오분류된 테스트 이미지 개수

: 38 개 이미지

True: air hockey
Predicted: table tennis

True: baseball
Predicted: football

True: baseball
Predicted: cricket

True: baseball
Predicted: football

True: baton twirling
Predicted: uneven bars

True: baton twirling
Predicted: archery

True: bmx
Predicted: track bicycle

True: bmx
Predicted: skydiving

True: cheerleading
Predicted: basketball

True: curling
Predicted: shot put

True: fencing
Predicted: lacrosse

True: frisbee
Predicted: hammer throw

True: gaga
Predicted: rollerblade racing

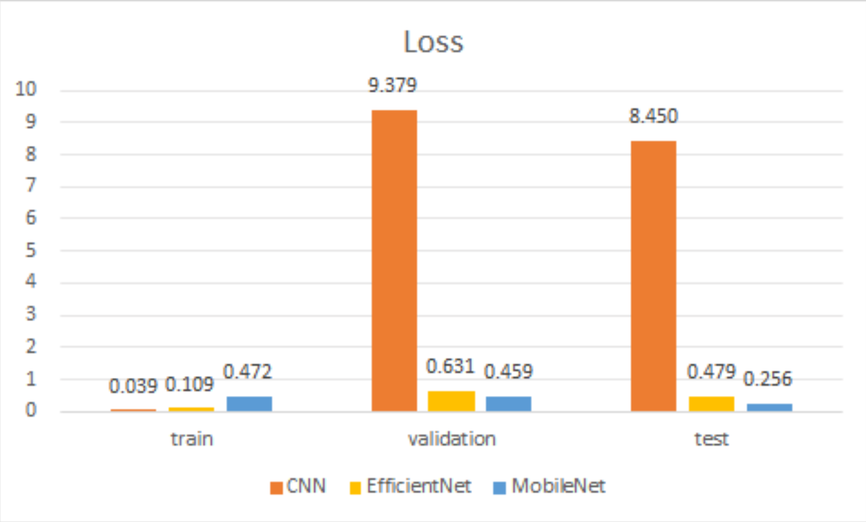
True: horseshoe pitching
Predicted: frisbee

True: horseshoe pitching
Predicted: frisbee

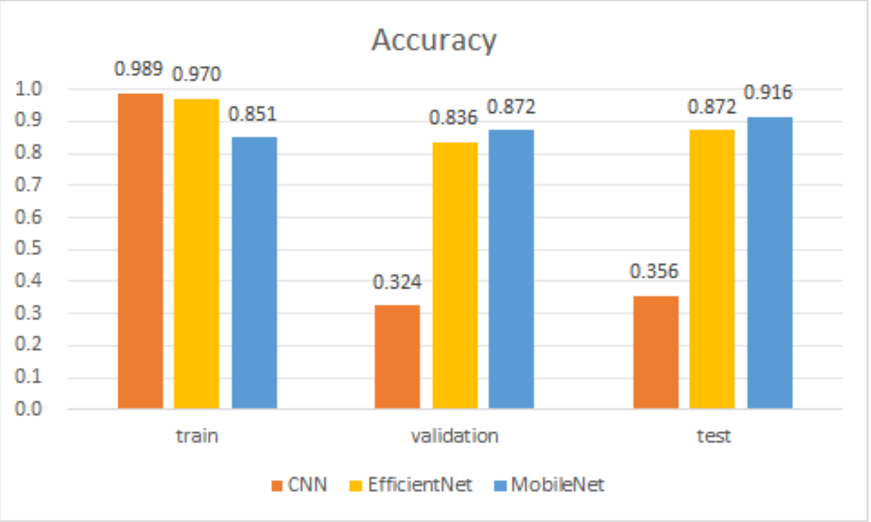
05

모델별 비교 및 시각화

Loss



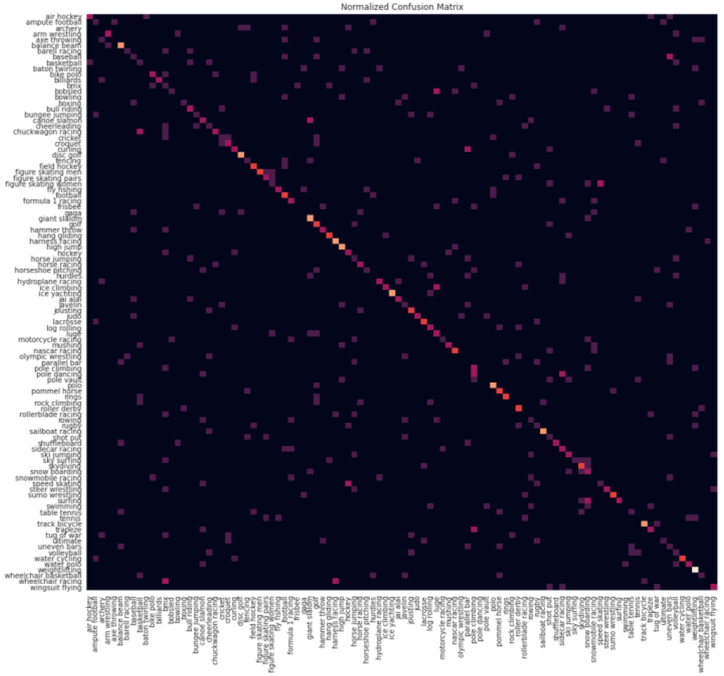
Accuracy



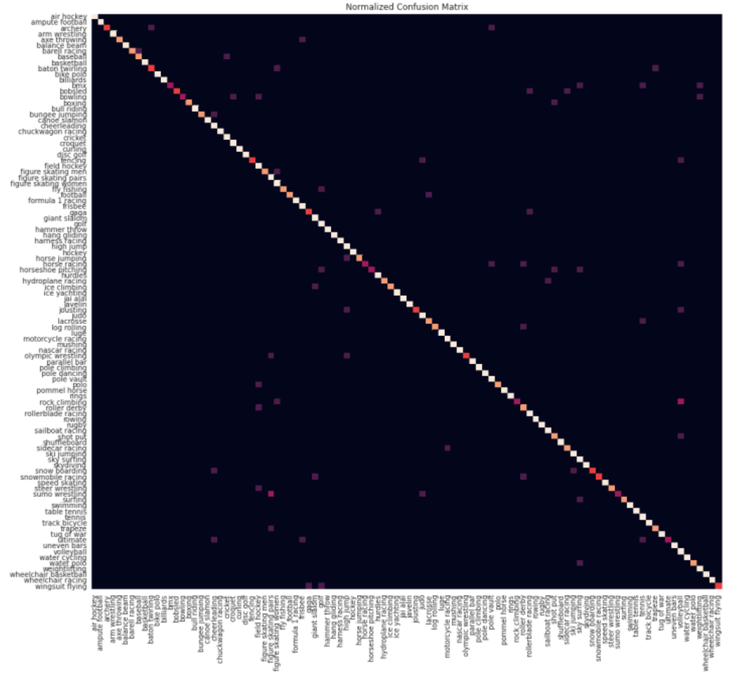
05

모델별 비교 및 시각화

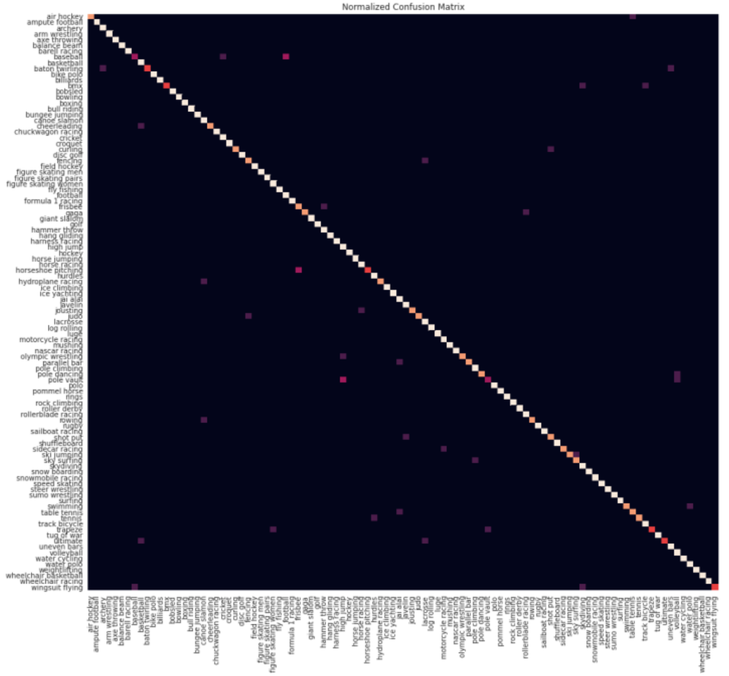
CNN



EfficientNet



MobileNet



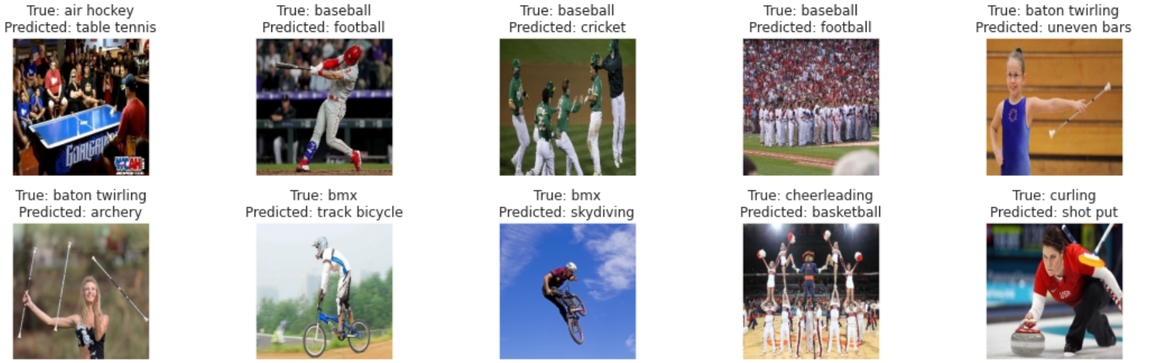
05

모델별 비교 및 시각화

CNN

EfficientNet

MobileNet



06

결론 및 한계점

- 이미지 측면

1) 이미지에 노이즈가 많아서 정확도 높이기에 한계가 있었음

2) ImageNet 범주에 해당되지 않은 특별한 스포츠들이 있어서 학습이 잘 이루어지지 않았음

3) 남 -여 피겨 스케이트나 골프 - 크로케 같이 비슷한 이미지가 많아 분류에 어려움이 있음




- 파라미터 측면

파라미터 최적화에 대한 점이 미흡하여 모델 성능을 좀 더 개선하지 못함

THANK YOU

스포츠 이미지 분류 모델 개발

김정명/김하영/유영상/한수정