전동 킥보드 헬멧 착용여부 판단 모델

슈퍼보드

2016270431 박정수 2016270484 오민섭 2016270464 박형준 2016270447 장종환

팀 소개

박정수
2016270431

- Data Collection
- CNN Modeling
- Report Writing

장종환
2016270447

- Data Collection
- CNN Modeling
- Running Models

오민섭
2016270484

- Data Collection
- CNN Modeling
- Running Models

박형준
2016270464

- Data Collection
- CNN Modeling
- Data Extraction

Contents

1. 목적
2. Data Set 설명
3. Data Collecting 설명
4. 모델 설명
5. 결과 모델
6. 활용 방안

1. 목적

기사 출처 : https://news.kbs.co.kr/news/view.do?ncd=5206254&ref=A,
https://news.mt.co.kr/mtview.php?no=2021051110371737303,

5월 13일부터 범칙금 시행

법 개정 후에도 안전불감증으로 인해 헬멧을 착용하지 않는 현실

"

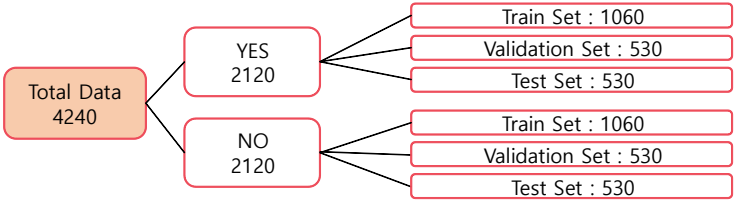
전동 킥보드 이용 시 헬멧 착용여부를 판단할 수 있는 **알고리즘**을 전동 킥보드 대여 어플리케이션에 도입한다면
기존의 헬멧을 착용하지 않았을 경우 발생하는 **안전사고를 사전에 예방**하는 동시에 **안전한 모빌리티 이용 문화가 정착**되는 기대 효과가 있다

"

2. Data Set

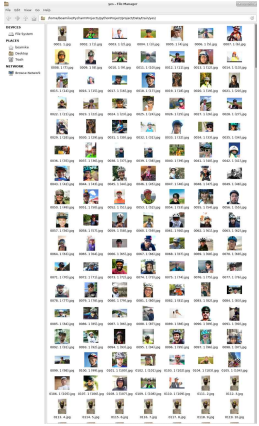
헬멧 착용 사진 (YES)
2120장

헬멧 미착용 사진 (NO)
2120장



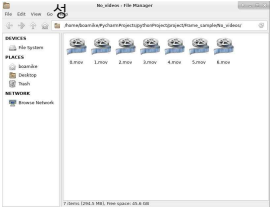
2. Data Set

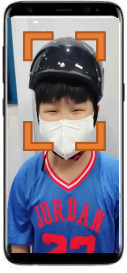
Data Set의 다양성 및 부족한 데이터 보충을 위해
109개의 Google Crawling 데이터를 수집하였음



3. Data Collect

1. 어플리케이션 접목 상황 고려
   - 어플 사용시에 헬멧을 인증할 수 있는 각도를 고려
   - 지인 및 친구들에게 셀프 동영상을 수집
   - 동영상을 프레임 단위로 읽어와 Data [illegible] 성





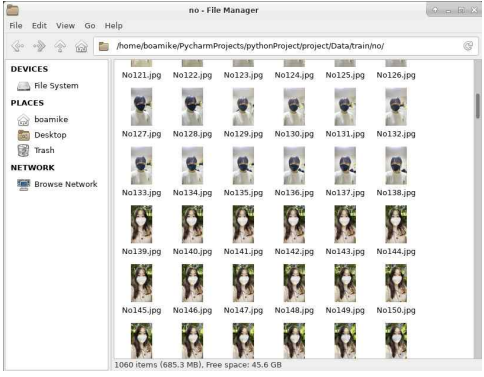

```
# Video Load
Video = cv2.VideoCapture('/home/boamike/PycharmProjects/pythonProject/project/Frame_sampling/Yes_videos/6.mov')
fps = Video.get(cv2.CAP_PROP_FPS)
# Check Frame Per Second
print(fps)

if not Video.isOpened():
    print('CD Check')
    exit()

sample_num = 0
# Continue Numbering
capture_num = 2857

# Videos Capture
while Video.isOpened():

    # Read Frame from Video
    status, frame = Video.read()
    sample_num = sample_num + 1

    if not status:
        break

    # frame = cv2.resize(frame, dsize=(1080, 720), interpolation=cv2.INTER_LINEAR)
    cv2.imshow('Shapture Video Frame', frame)

    if sample_num == 5:
        capture_num = capture_num + 1
        # Image Save
        cv2.imwrite('/home/boamike/PycharmProjects/pythonProject/project/Frame_sampling/Yes_images/'
                    + str(capture_num) + '.jpg', frame)


        sample_num = 0

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

Video.release()
cv2.destroyWindow()
```

3. Data Collect

2. Train Set과 Test Set 객체 분리

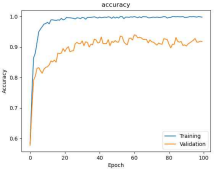
- Train Set에서 훈련한 인물사진은 Test Set에서 사용되지 않게 Numbering을 통해 분리

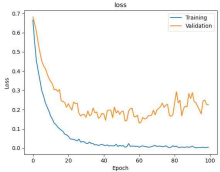
-> 이미지가 겹쳐 판독모델의 정확도가 하락하는 것을 방지



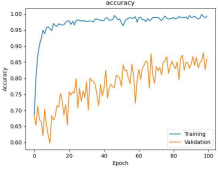
4. 모델 설명 Pre-trained Model VGG16 vs Inception V3

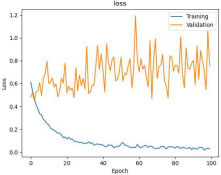
VGG16



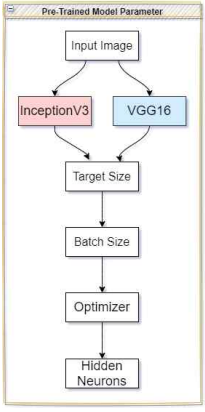


Inception V3





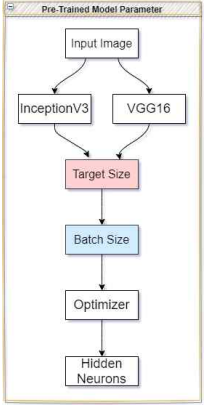
Pre-trained Model		
	VGG16	Inception V3
Train_acc	1	0.94386
Test_acc:	0.8075	0.7688
Val_acc	0.9180	0.860
Val_loss:	0.2254	0.7559



4. 모델 설명 Block Parameters

Target Size											
	128x128	150x150	180x180	210x210	256x256	300x300	**400x400**	420x420	450x450	470x470	512x512
Train acc	0.998	0.998	0.998	0.998	0.998	0.999	0.999	0.999	0.999	0.999	0.998
Test acc	0.721	0.756	0.761	0.797	0.828	0.846	0.905	0.861	0.887	0.858	0.889
Val acc	0.794	0.908	0.928	0.873	0.920	0.954	0.998	0.996	0.979	0.944	0.98
Val loss	0.413	0.223	0.178	0.245	0.208	0.145	0.0905	0.0897	0.132	0.171	0.148

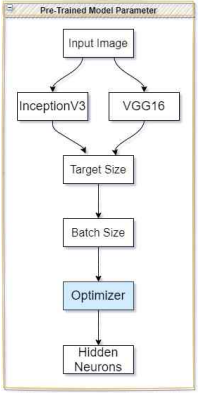
Batch Size					
	Batch10	Batch20	**Batch30**	Batch50	Batch100
Train_acc	0.99386793	0.9971698	0.9985849	0.9990566	0.9995283
Test_acc:	0.7773585	0.7528302	0.81226414	0.72075474	0.76509434
Val_acc	0.8940	0.9250	0.9750	0.8694	0.9025
Val_loss:	0.0907	0.1821	0.1206	0.2822	0.1672



4. 모델 설명 Block Parameters

Optimizer		
	RMSProp	Adam
Train_acc	0.954717	0.946222643
Test_acc:	0.80943395	0.80943395
Val_acc	0.9877	0.8640
Val_loss:	0.7680	0.9689

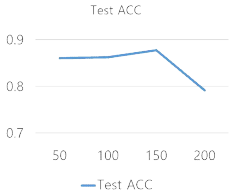
Pre-train Learning Rate				
	PT_lr_1e-3	PT_lr_1e-4	PT_lr_1e-5	**PT_lr_default**
Train acc	0.999	0.992	0.825	1
Test acc	0.808	0.723	0.571	0.835
Val acc	0.928	0.946	0.776	0.960
Val loss	0.219	0.261	0.558	0.098

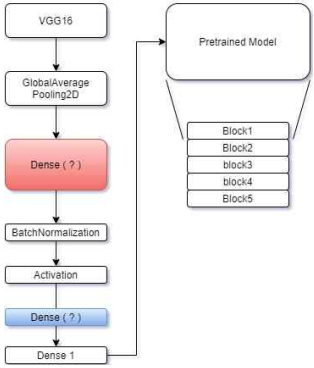


4. 모델 설명 Block Parameters

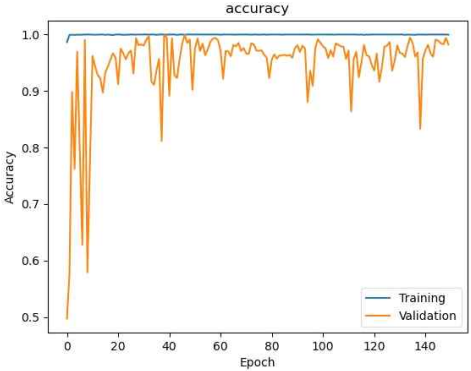
Number of Hidden Neurons								
	16,16	32,16	64,16	256,16	512,16	64,32	128, 32	256,32
train_acc	0.91745285	0.854717	0.91037736	0.9542453	0.9070755	0.9985849	0.9985849	0.9995283
test_acc:	0.7066038	0.70943395	0.7009434	0.7556604	0.7471698	0.75377357	0.73679245	0.76132077
val_acc	0.6800	0.8877	0.7290	0.8320	0.7820	0.8890	0.9060	0.9290
val_loss:	0.8579	0.6680	1.1977	0.6393	1.0196	0.2030	0.2131	0.1445

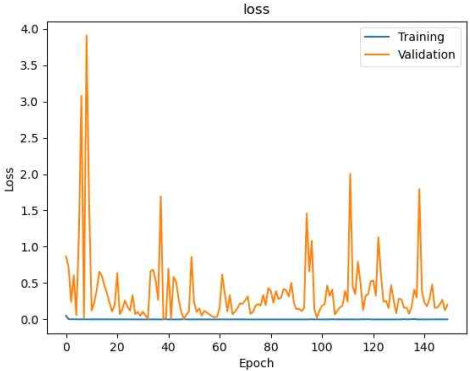
Pre-training Epochs				
	Ep_50	Ep_100	**Ep_150**	Ep_200
Train_acc	0.999	1	1	1
Test_acc	0.860	0.862	0.877	0.791
Val_acc	0.937	0.945	0.982	0.940
Val_loss	0.511	0.498	0.203	0.899





4. 모델 설명 Pre-trained Model





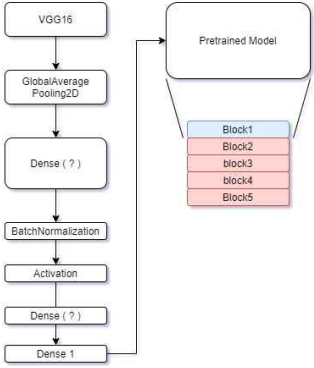
4. 모델 설명 Fine-Tuned Model

Fine Tuning Learning Rate		
	FT_lr_1e-4	**FT_lr_1e-5**
Train_acc	1	1
Test_acc	0.885	0.956
Val_acc	0.960	0.966
Val_loss	0.142	0.173

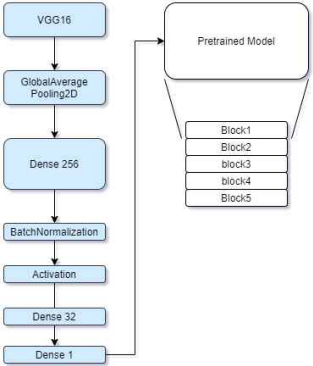
Freezing Layer (.start with)					
	Block1	**Block2**	Block3	Block4	Block5
Train_acc	1	1	1	1	1
Test_acc	0.930	0.935	0.884	0.904	0.895
Val_acc	0.987	0.980	0.980	0.164	0.962
Val_loss	0.140	0.166	0.143	0.980	0.408

Fine Tuning Epochs						
	Ep_50	Ep_100	Ep_150	**Ep_200**	Ep_250	Ep_300
Train_acc	1	1	1	1	1	1
Test_acc	0.928	0.925	0.925	0.936	0.929	0.933
Val_acc	0.985	0.976	0.992	0.980	0.9764	0.9899
Val_loss	0.127	0.149	0.115	0.117	0.1432	0.0858

Freezing Layer (.start with)

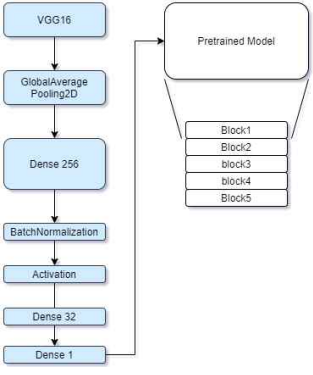


4. 모델 설명



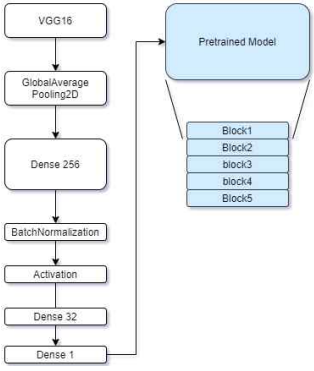

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20, shear_range=0.1,
                                   width_shift_range=0.1, height_shift_range=0.1,
                                   zoom_range=0.1, horizontal_flip=True, fill_mode='nearest')
validation_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(400, 400),
        batch_size=30,
        class_mode='binary')
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(400, 400),
        batch_size=30,
        class_mode='binary')
validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=(400, 400),
        batch_size=30,
        class_mode='binary')
```

4. 모델 설명



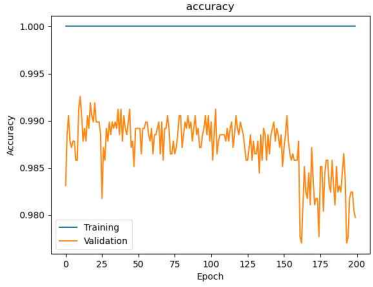

```
input_shape = [400, 400, 3] # as a shape of image
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import BatchNormalization, Dropout, Activation
def build_model():
    model = models.Sequential()
    conv_base = VGG16(weights='imagenet', include_top = False, input_shape = input_shape)
    conv_base.trainable=False

    model.add(conv_base)

    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(256))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(layers.Dense(32, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    # compile
    model.compile(optimizer='RMSprop',
                  loss='binary_crossentropy', metrics=['accuracy'])
    return model
import time
starttime=time.time();
num_epochs = 150
model = build_model()
history = model.fit_generator(train_generator,
                              epochs=num_epochs, steps_per_epoch=100,
                              validation_data=validation_generator, validation_steps=50)
```

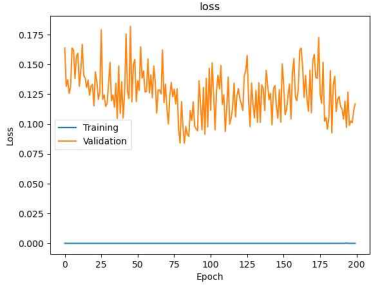
4. 모델 설명




```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(480, 480),
    batch_size=30,
    class_mode='binary')
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(480, 480),
    batch_size=30,
    class_mode='binary')
validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(480, 480),
    batch_size=30,
    class_mode='binary')
# main loop without cross-validation
starttime = time.time()
num_epochs = 200
history = model.fit_generator(train_generator,
                              epochs=num_epochs, steps_per_epoch=100,
                              validation_data=validation_generator, validation_steps=50)

for layer in conv_base.layers:
    if layer.name.startswith('block2'):
        layer.trainable = True
model.compile(optimizer=optimizers.RMSprop(lr=1e-5), loss='binary_crossentropy', metrics=['accuracy'])
```

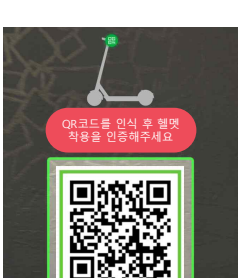
5. 결과 모델

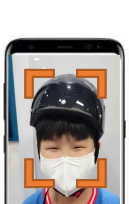




PT_Model	Target size	Batch size	Optimizer	Number of Hidden Neuron
VGG16	**(400,400)**	**30**	**RMSprop**	**(256,32)**
PT_Learning rate	PT_Epochs	FT_learning rate	FT_Frozen blocks	FT_Epochs
Default(1e-3)	**150**	**1e-5**	**Block2**	**200**

6. 활용 방안





“ 전동 킥보드 이용 시 헬멧 착용여부를 판단할 수 있는 **알고리즘**을 전동 킥보드 대여 어플리케이션에 도입한다면
기존의 헬멧을 착용하지 않았을 경우 발생하는 **안전사고를 사전에 예방**하는 동시에 **안전한 모빌리티 이용 문화가 정착**되는 기대 효과가 있다 ”


THANK YOU!