AlexNet 논문 구현 및
운전 중 졸음 방지 서비스 제작

1조 척척석사

김동협, 강소희

01

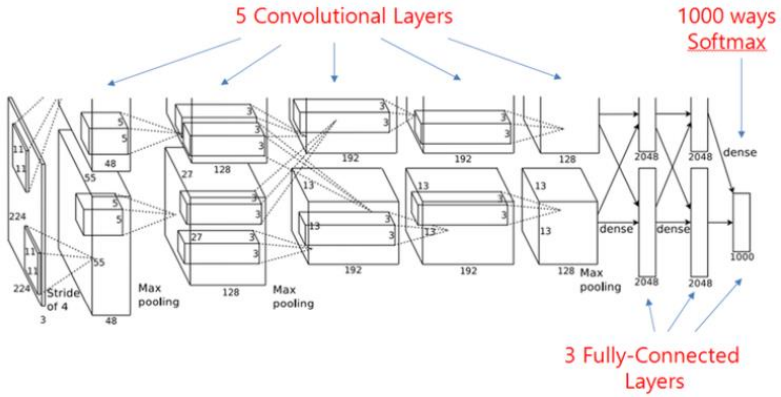
Index

01

프로젝트 개요

1. AlexNet 논문 분석 및 구현

1) AlexNet이란?



- 2012년 CNN이 주목 받게 된 계기 AlexNet
- 1989년 CNN의 기반이 된 LeNet 구조 계승
- 단, 사이즈의 확장 (32*32 >> 224*224)
- Activation Function으로 ReLU 사용 (Vanishing Gredient 극복)
- Overfitting 문제 해결을 위해 Dropout layer 개발
- 현재까지도 이어지고 있는 CNN의 발전에 초석이 되는 CNN 구조

2. '운전 중 졸음 방지 서비스' 제작

1) 1번에서 사용한 구조를 입력 데이터, 출력 데이터만 바꾸고 그대로 학습에 사용

02

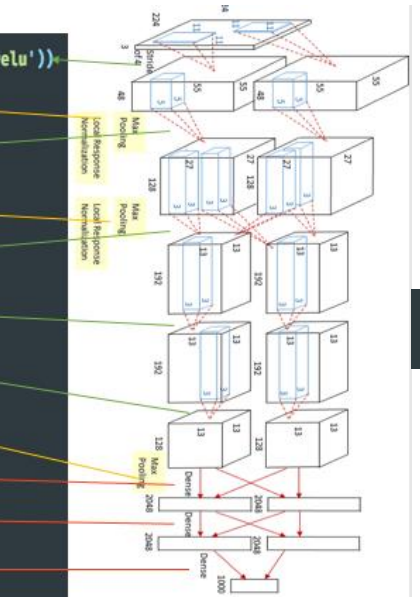
팀 구성 및 역할

이름	역할
김동협	데이터 전처리 및 모델링 PyQT 프로그램 구축 PPT 작성 및 발표
강소희	데이터 전처리 및 모델링 프로그램 최적화 PPT 작성

프로젝트 한 눈에 보기

1. 원형 데이터(32x32x3) 전체 사용, 다양한 파라미터 변경하며 학습 -> 정확도 저조(약 60%)
2. 원형 데이터(32x32x3) 을 (224x224x3) 으로 변형하여 데이터 일부 학습, 파라미터는 논문 동일 -> 정확도 저조(63%), 학습 시간 증가
3. 원형 데이터(32x32x3) 을 (224x224x3) 으로 변형하여 데이터 전체 학습, 파라미터는 논문 동일 -> 정확도 상승(80%), Epoch를 조절해서 학습 시간 조정

```
model.add(Conv2D(filters=96,kernel_size=(11,11),strides=(4,4),input_shape=(224,224,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(256,(5,5),padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())

model.add(Conv2D(384,(3,3),padding='same',activation='relu'))
model.add(Conv2D(384,(3,3),padding='same',activation='relu'))
model.add(Conv2D(256,(3,3),padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(10,activation='softmax'))
```

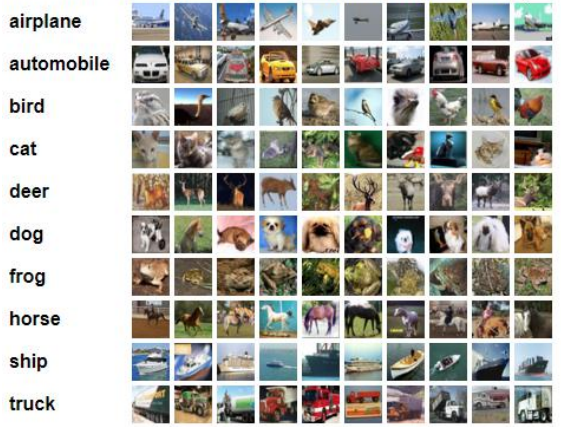


The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:




```
sgd=keras.optimizers.SGD(lr=0.01, momentum=0.9, decay=0.0005 , nesterov=True)
```

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w}\Big|_{w_i} \right\rangle_{D_i}$$

$$w_{i+1} := w_i + v_{i+1}$$

The learning rate was initialized at 0.01

프로젝트 한 눈에 보기

1. 검증 데이터의 정확도가 80% 정도
2. Batch size:128, epochs 20, patience 5로 변경. 6시간 40분 소요, 12epoch 에서 멈춤


```
Epoch 00012: val_loss did not improve from 0.67274
313/313 [==============================] - 85s 273ms/step - loss: 0.7298 - accuracy: 0.8093

 Test Accuracy: 0.8093
```

프로젝트 한 눈에 보기

• 모델 검증

```
wrong_list=[]
for i in range (len(test_images)):
    print(i+1,':the answer is',column[int(test_labels[i])] ,'and prediction is',column[test_predict[0][i].argmax()])
    print()
    if column[int(test_labels[i])] != column[test_predict[0][i].argmax()]:
        wrong_list.append(i)
print('###################################    Accuracy of sample images is:',(number_of_samples-len(wrong_list))/number_of_samples,'%
 print('####################################Below is informations about wrong prediction#############################################')

 plt.figure(figsize=(4,4))

 for idx,wrong_i in enumerate(wrong_list):

     plt.subplot(int(len(wrong_list)/6)+1,6,idx+1)
     plt.subplots_adjust(left=0.0,bottom=0.0,right=5,top=2)
     plt.imshow(test_images[wrong_i],cmap='Greys',interpolation='nearest')
     plt.title('Label:'+column[int(test_labels[wrong_i])]+'    predict:'+column[test_predict[0][wrong_i].argmax()])
     plt.axis('off')
 plt.show()
```

정확도 계산 및 틀린 사진 plotting

```
91 :the answer is airplane and prediction is airplane

92 :the answer is cat and prediction is frog

93 :the answer is ship and prediction is ship

94 :the answer is frog and prediction is frog

95 :the answer is deer and prediction is deer

96 :the answer is frog and prediction is frog

97 :the answer is frog and prediction is frog

98 :the answer is airplane and prediction is airplane

99 :the answer is airplane and prediction is airplane

100 :the answer is horse and prediction is horse

###################################    Accuracy of sample images is: 0.82 %    ##########################################
####################################Below is informations about wrong prediction#############################################
```

03

프로젝트 한 눈에 보기

1. 처음 계획(중간 발표 전까지):

1) AlexNet 논문 분석 및 구현
2) AlexNet 구조를 이용한 간단한 서비스 제작

2. 변경된 계획:

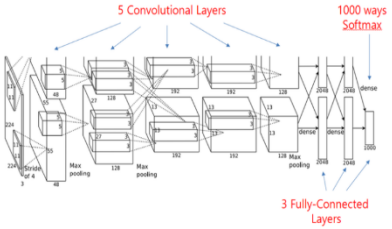
1) AlexNet 구현만으론 임팩트가 부족하다고 판단 : 분석은 개인적인 공부에 의의가 있었지만 구현은 수업시간에 했던 CNN 실습과 사이즈가 큰 것 빼고는 크게 차이가 나지 않았음
2) 서비스 제작에 무게를 두고 진행

03

프로젝트 한 눈에 보기

프로젝트 한 눈에 보기

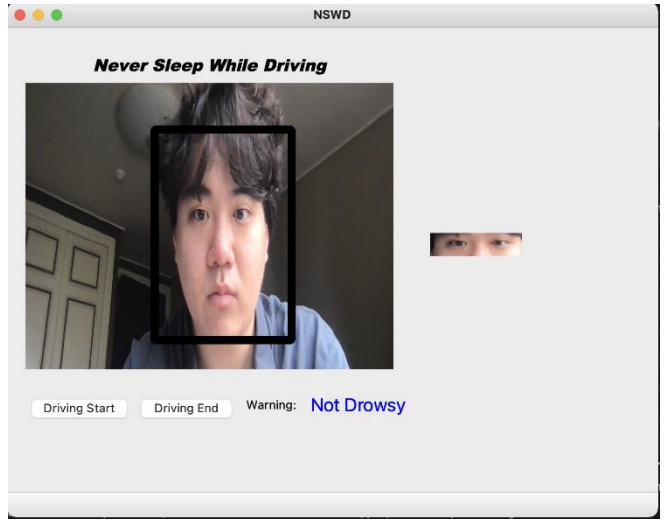
CEW Dataset, My picture



AlexNet

주피터 노트북

model



최종 결과물

vscode

04

프로젝트 수행 절차 및 방법

1. 이진 분류 머신러닝 모델링 (Open eyes:0, Closed Eyes:1)
2. PyQT, OpenCV 실시간 영상 모듈 설계
3. 주요 문제 및 해결방법

(1) 눈 깜빡임 감지 모델

1. 사용 데이터 : Closed Eyes In The Wild (CEW) + 자신의 사진

	Closed Eyes	Open Eyes
Train Data	1100	1100
Test Data	92	92

모두 100 x 100 x 3

25 Open, 25 Closed,
막 자른 사진 총 50개,
모두 Train Data 로 이용

(1) 눈 깜빡임 감지 모델

2. 모델 구현 특징

차후 설명 예정

```
closed_img = [cv2.imread(file)[25:50] for file in glob.glob("/ClosedFace/*.jpg")]
closed_img=np.array(closed_img)
open_img = [cv2.imread(file)[25:50] for file in glob.glob("/OpenFace/*.jpg")]
open_img=np.array(open_img)
my_img=[cv2.resize(np.array(cv2.imread(file)),(100,100))[25:50] for file in glob.glob("/myface/*.jpg")]
my_img=np.array(my_img)

for i in range(len(open_img)):
    open_img[i]= cv2.cvtColor(open_img[i], cv2.COLOR_BGR2RGB)
for i in range(len(closed_img)):
    closed_img[i]= cv2.cvtColor(closed_img[i], cv2.COLOR_BGR2RGB)
for i in range(len(my_img)):
    my_img[i]= cv2.cvtColor(my_img[i], cv2.COLOR_BGR2RGB)
```

cv2,plt 차이 때문

막 자른 사진 => 사진의 크기가 모두 다르다 => 데이터의 shape가 각각 모두 달라서 array 형성 불가=> 파일 단위로 읽어오면서 resize 진행

```
(50, 25, 100, 3)
[[[[ 22  19  20]
   [ 21  19  19]
   [ 19  16  18]
   ...
   [ 66  67  77]
   [ 46  48  56]
   [ 51  53  61]]

  [[ 20  17  21]
   [ 23  19  24]
   [ 21  17  22]
```

vs

```
(50,)
[array([[[ 45,  49,  5
          [ 44,  48,  53
          [ 44,  48,  53
          ...,
          [ 42,  50,  57
          [ 42,  50,  57
          [ 46,  51,  60

         [[ 44,  48,  53
          [ 44,  48,  53
          [ 44,  48,  53
```

(1) 눈 깜빡임 감지 모델

2. 모델 구현 특징

```
my_label=np.array([1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
                   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])
open_img4train=open_img[:1100]
closed_img4train=closed_img[:1100]
open_img4test=open_img[1100:-39]
closed_img4test=closed_img[1100:]

open_label4train=np.zeros(shape=(len(open_img4train),), dtype=np.int8)
closed_label4train=np.ones(shape=(len(closed_img4train),), dtype=np.int8)
open_label4test=np.zeros(shape=(len(open_img4test),), dtype=np.int8)
closed_label4test=np.ones(shape=(len(closed_img4test),), dtype=np.int8)

train_images=np.concatenate((open_img4train,closed_img4train),axis=0)
train_labels=np.concatenate((open_label4train,closed_label4train),axis=0)
train_images=np.concatenate((train_images,my_img),axis=0)
train_labels=np.concatenate((train_labels,my_label),axis=0)
test_images=np.concatenate((open_img4test,closed_img4test),axis=0)
test_labels=np.concatenate((open_label4test,closed_label4test),axis=0)
```

My_img 라벨 형성

데이터 분류

라벨 형성

배열 결합

(1) 눈 깜빡임 감지 모델

2. 모델 구현 특징

```
#데이터 섞어 주기

train_idx=np.arange(len(train_images))
test_idx=np.arange(len(test_images))

np.random.shuffle(train_idx)
np.random.shuffle(test_idx)

train_images=train_images[train_idx]
train_labels=train_labels[train_idx]
test_images=test_images[test_idx]
test_labels=test_labels[test_idx]
```

(1) 눈 깜빡임 감지 모델

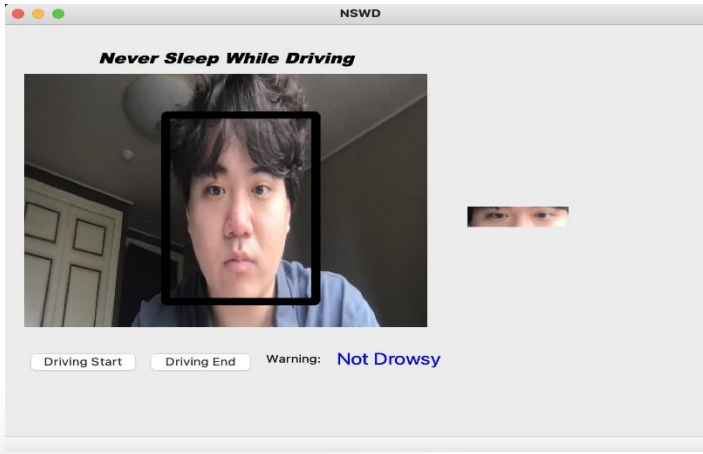
2. 학습 결과

```
Epoch 00015: val_loss improved from 0.40582 to 0.35432, saving model to ./model₩NSWD15-0.3543.hdf5

Epoch 00016: val_loss improved from 0.35432 to 0.31929, saving model to ./model₩NSWD16-0.3193.hdf5

Epoch 00017: val_loss improved from 0.31929 to 0.27982, saving model to ./model₩NSWD17-0.2798.hdf5

Epoch 00018: val_loss improved from 0.27982 to 0.25146, saving model to ./model₩NSWD18-0.2515.hdf5

Epoch 00019: val_loss did not improve from 0.25146

Epoch 00020: val_loss did not improve from 0.25146
6/6 [==============================] - 0s 16ms/step - loss: 0.2851 - accuracy: 0.8859

 Test Accuracy: 0.8859
```

(2) PyQt 졸음 방지 서비스

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
import cv2
import threading
from PyQt5 import QtGui
from tensorflow.keras.models import load_model
import numpy as np
import time
##############################모델코드##############################
model=load_model('NSWD18-0.2515.hdf5')
column=['Open Eyes','Closed Eyes']
####################################################################
```

(2) PyQt 졸음 방지 서비스



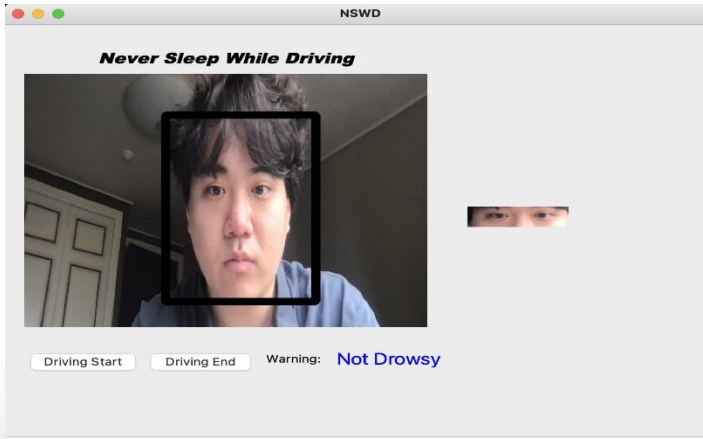

```
form_class = uic.loadUiType("NSWD_GUI.ui")[0]

class NSWD(QMainWindow,form_class):
  warning_count=0
  def __init__(self):
      super().__init__()
      self.setupUi(self)

  running = False
  def run(self):
      warning_count=0
      global running
      global roi_scaled

      cap = cv2.VideoCapture(0)
      self.trans_imgsize=(400,400,3)
      face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
```

카메라 →

얼굴 인식 →

눈을 인식하는 Classifier도 있으나, 프로젝트의 취지에 맞지 않아 배제

(2) PyQt 졸음 방지 서비스

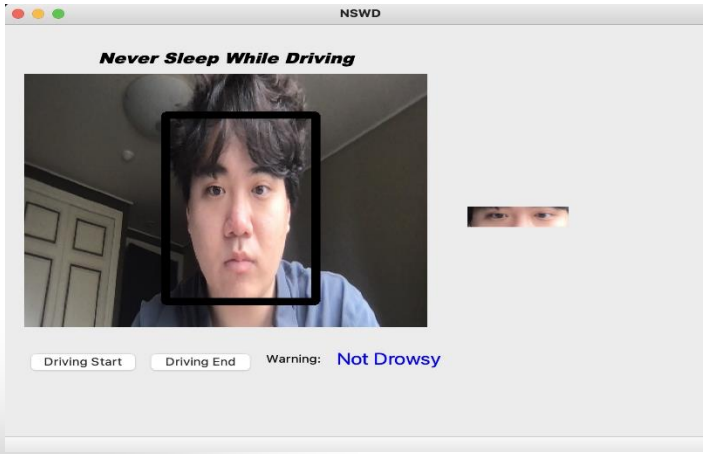



```
while running:
        ret, frame = cap.read()
        frame=cv2.resize(frame,(400,400))
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        faces=face_cascade.detectMultiScale(gray,1.3,4)

        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x-10,y-50),(x+w+10,y+h+50),(0,0,0),7)
            roi_color=frame[x-30:x+w+30,y+15:y+h-15]

            if x-50<0 or x+w+50>400 or y-50<0 or y+h+50>400:
                self.stop()

            roi_scaled=cv2.resize(roi_color,(100,100))
            roi_scaled=roi_scaled[25:50]
            roi_scaled_4predict=np.expand_dims(roi_scaled,axis=0)
```

차원이 맞지 않아 모델에 fit이 안된다. 따라서 한 차원 늘려줌

Roi : region of interest

(2) PyQt 졸음 방지 서비스




```
if warning_count>50:
                    self.statuslabel.setText('WARNING: WAKE UP')
                    self.statuslabel.setStyleSheet('Color:red')
                    self.statuslabel.setFont(QtGui.QFont('MS Gothic',40))
                else:
                    self.statuslabel.setText('Not Drowsy')
                    self.statuslabel.setStyleSheet('Color:blue')
                    self.statuslabel.setFont(QtGui.QFont('MS Gothic',20))

                print(model.predict(roi_scaled_4predict))

                if model.predict(roi_scaled_4predict)[0][0]>0.3:
                    warning_count=0
                else:
                    warning_count+=1
```

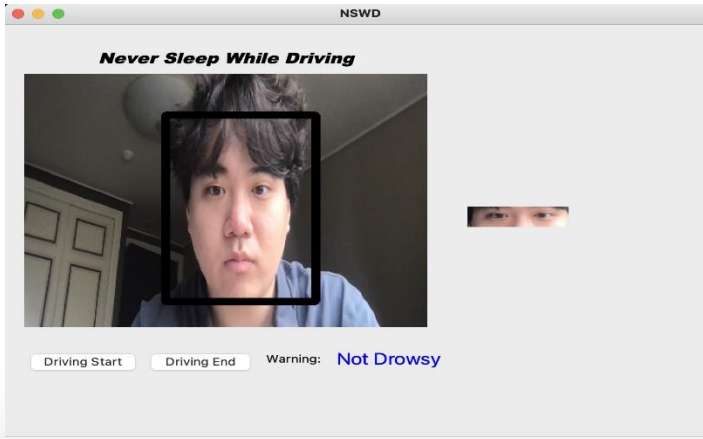
Warning

민감도

[0.26401758 0.73598236]]

0: open 1:closed

(2) PyQt 졸음 방지 서비스

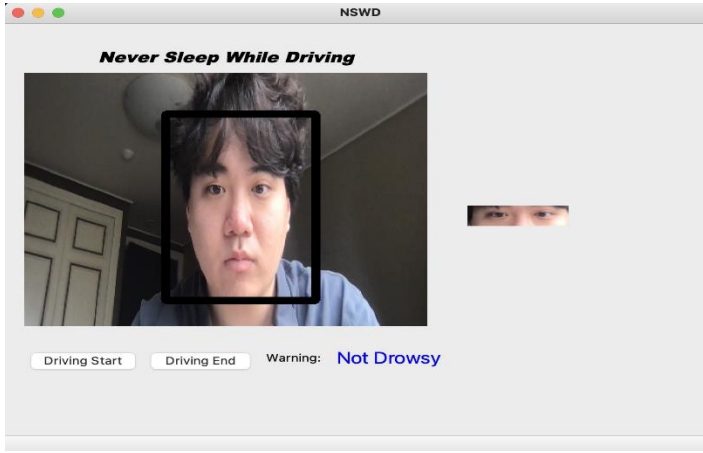



```
                                                w    h   w*c
            fImg = QtGui.QImage(roi_scaled.data, 100,25,300, QtGui.QImage.Format_RGB888)

            fpixmap = QtGui.QPixmap.fromImage(fImg)
            self.facelabel.setPixmap(fpixmap)
            qImg = QtGui.QImage(frame.data, self.trans_imgsize[0], self.trans_imgsize[1],
                                self.trans_imgsize[0]*self.trans_imgsize[2],
                                QtGui.QImage.Format_RGB888)
            pixmap = QtGui.QPixmap.fromImage(qImg)

            self.camlabel.setPixmap(pixmap)

        cap.release()
```

이미지를 PyQT에 붙이는 부분

전체 이미지, roi 이미지 2가지 표시한다.

(2) PyQt 졸음 방지 서비스




```
  def stop(self):
      global running
      running = False
  def start(self):
      global running
      running = True
      th = threading.Thread(target=self.run)
      th.start()
  def statuschange(self):
      self.statuslabel.setText('warning')

if __name__=="__main__":
  app=QApplication(sys.argv)
  object1=NSWD()
  object1.setWindowTitle('NSWD')
  object1.show()
  sys.exit(app.exec_())
```

이 함수는 미사용

(3) 주요 문제 및 해결 방법

- 주요 문제 및 해결 방법

1. 제작 과정에서 많은 문제가 있었지만 가장 큰 난관이었던
   ‘데이터셋에서 성능(90%)을 보인 모델이 실시간 영상 데이터를 인식 못함‘
   위주로 서술 예정

(3) 주요 문제 및 해결 방법

- 문제를 해결하기까지

1. 100x100x3 CEW 데이터셋만을 이용해서 학습 후 실시간 영상 데이터 테스트 -> 거의 인식 못함

2. 샘플 사진을 찍어서 (Open 5개 Closed 5개) model.predict(img) 검증

→ 똑같이 인식 못함을 확인

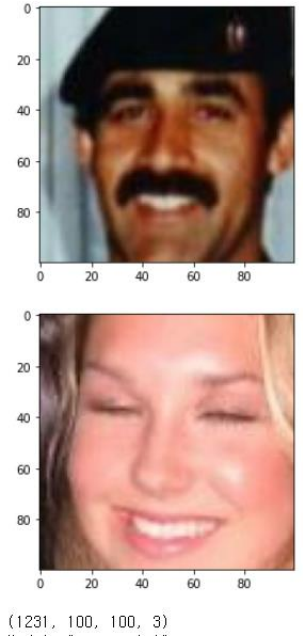
3. 문제에 대해 다음과 같은 가정을 내림. 순서는 생각해낸 순서와 같다

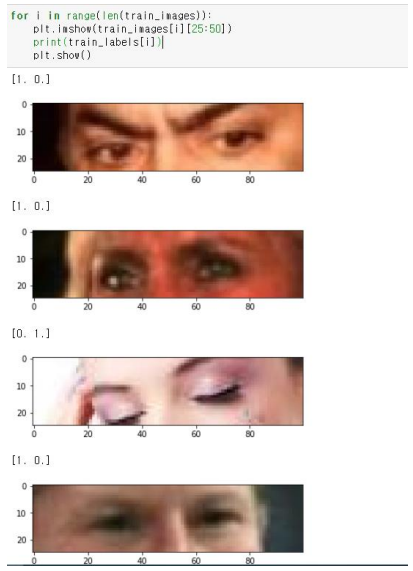
1) **과적합 문제** -> Early Stopping Patience를 10에서 5로 변경.
-> 효과X, 게다가 Test Data는 정확도가 높으므로 과적합은 아니라고 판단

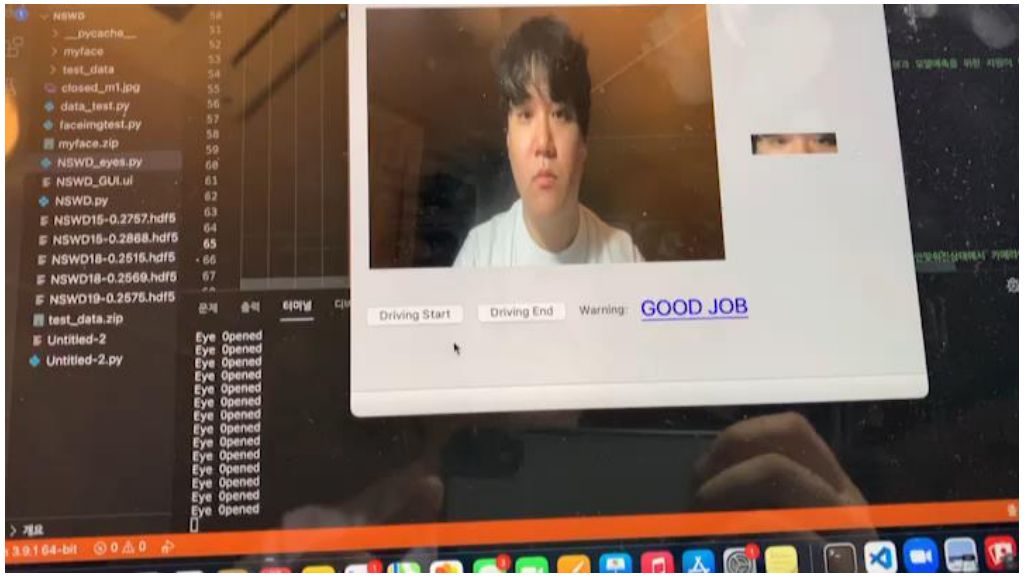
2) **데이터가 적거나 자신의 이미지에 맞지 않음**
-> 위에서 찍은 총 10개의 샘플 이미지 데이터를 추가한 후 다시 학습
-> 10개의 샘플 이미지 데이터에서 성능 향상을 보임

(3) 주요 문제 및 해결 방법

- **문제를 해결하기까지**
- **문제에 대해 다음과 같은 가정을 내림. 순서는 생각해낸 순서와 같다**
  1) 하지만 여전히 실시간 영상에 대해서는 인식 불가.
  2) 전체 100x100x3 이미지에서 눈이 차지하는 비율은 10%도 안되므로 눈 가까운 부분에 집중하기로 전략 변경
     → y축 [25:50] 슬라이싱 → 전체 이미지에서 눈이 차지하는 비율이 높아짐 → 실시간 데이터에서도 인식하기 시작

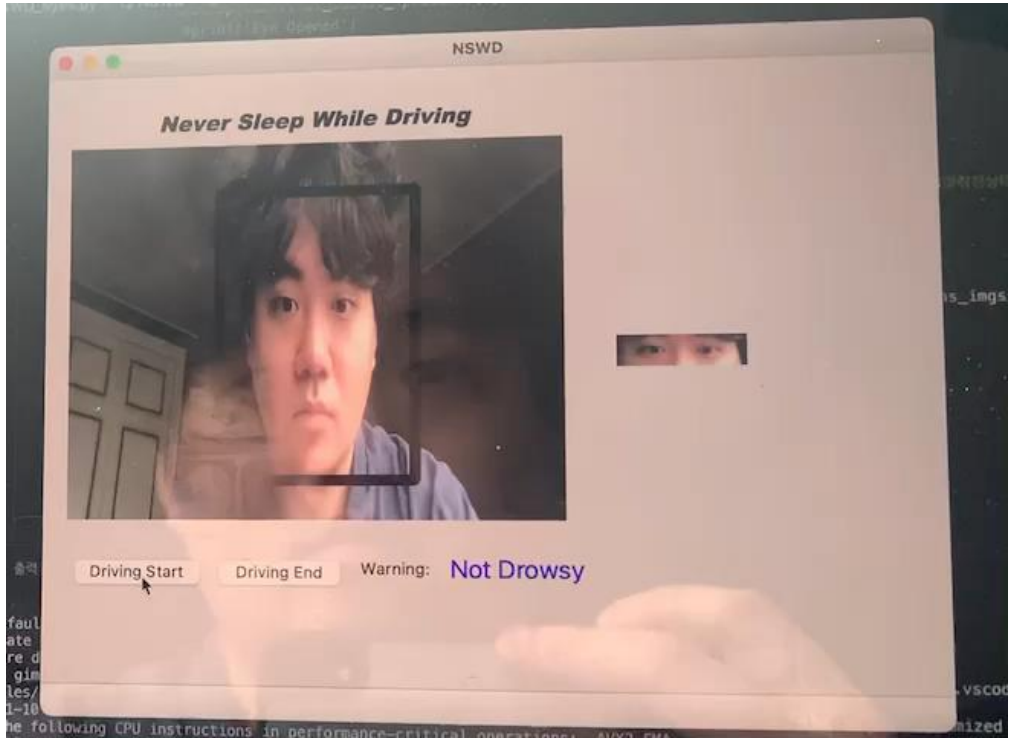






프로젝트 결과

- 시간이 충분히 남을 시 시연으로 대체



06

자체 평가 의견

- **프로젝트 기대 효과 및 배운 점**

1. AlexNet 논문을 분석, 구현함으로써 차후 현업에서 마주할 새로운 논문에 대한 분석 및 구현 능력을 키울 수 있었음
2. 기본적인 이미지 전처리(resize, cvtcolor, slicing 등)능력 및 머신러닝 이미지 처리 관련 문제 해결 능력을 키울 수 있었음
3. 다양한 라이브러리와 메서드에 이미지 데이터를 적용함으로써 데이터의 차원에 대한 이해도 증가
4. 프로그램 최적화에 대한 고민을 통해 기본적인 프로그래밍 역량 증가

06

자체 평가 의견

- **프로젝트 개선점**

1. 한정된 상황에서만 이용 가능 (빛, 얼굴의 위치 등)
2. 프로그램 최적화 부족 : 카메라가 잘 끊긴다.

감사합니다!