거북목 탐지 웹사이트를 개설하다.

임소영

✳ 2024.04.30.

✳ 24.04.30.

Table of contents

✳ 24.04.30.

주제 선정 배경

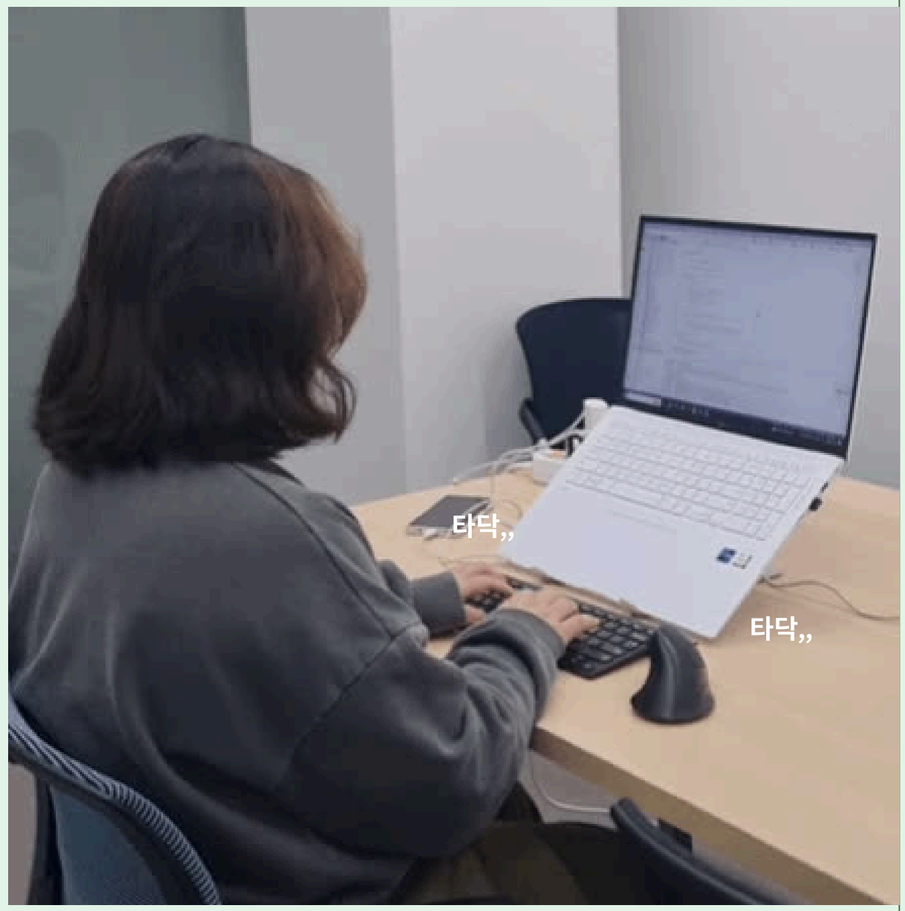
나는 왜 그 주제를 선택했을까?

KDT 5개월차로 들어갈 무렵 나는 **확신**했다.

이미 있었던 거북목이 점점 **악화**되는 것을 느꼈다.
나쁜 자세를 하고 있다면 누가 옆에서 알려줬으면
하는 생각이 들었다.

그래서 만들고자 한다.

노트북 사용시 경각심을 가지게 할 **자세 프로그램** !



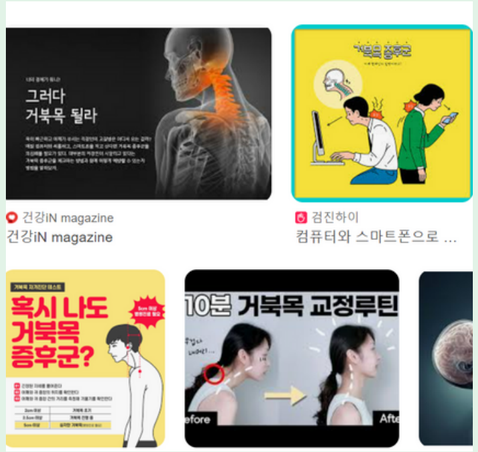
구현 기능

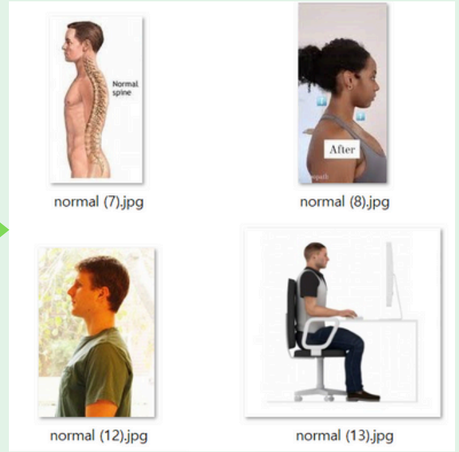
기능 이름	기능 설명	사용 기술
Turtle or Not (Quick_ver)	• **간이 검사** 기능으로 이용자가 사진을 제출해서 본인이 **거북목 유무**를 알 수 있다. • 가장 정확한 검사는 병원에서 X-ray를 찍는 것이기에 간이 검사로 이용할 법하다.	***CNN (ResNet18)***
CCTV for my neck	• 기능이 켜지면 노트북을 이용하면서 **거북목이 되는 순간에 알림창**이 뜬다. • **거리 측정**을 위해 카메라 촬영이 선행된다. • 그 후 거리가 확정되면 사용자가 **종료하기 전까지 계속**해서 자세 교정 프로그램이 가동된다.	***Mediapipe*** (제스처 인식) (객체 감지)
Statistics for my neck	• **추후 업데이트 예정** • CCTV for my neck (자세 감지) 기능 사용하는 동안 자세 교정 알림 **기록을 측정**한다. • 측정된 기록을 **DB에 저장**된다. • 이용자는 본인의 올바르지 못한 자세의 정도를 수치적으로 알 수 있다.	***SQL*** (Database 구축)

✳ 24.04.30.

모델 개설

Data 준비





[크롬 이용]

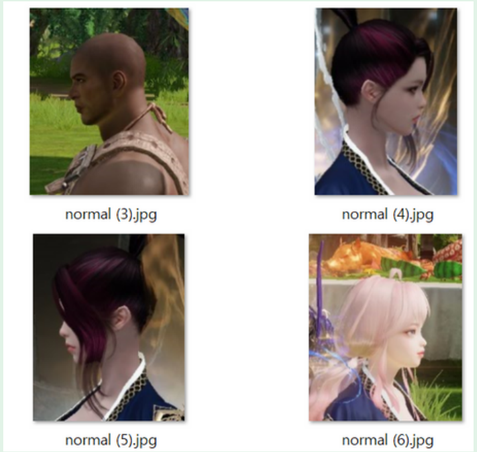
관련 키워드를 검색하여 사진 다운

▼

목 주위에 머리카락이나 옷과 같은 노이즈 있는 사진은 삭제

▼

상반신을 위주로 crop



[게임 캐릭터 이용] 로XX아X

실제 사람 모양을 한 게임 캐릭터 옆모습 캡처

▼

목 주위에 아이템 같은 노이즈 있는 사진은 삭제

▼

상반신을 위주로 crop

전처리

```python
preprocessing = transforms.Compose([
    transforms.Resize((150,150)),
    transforms.RandomCrop((10,10)),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))
])
```

Dataset

```python
file_dir = './data/neck'
imgDS = ImageFolder(root = file_dir, transform = preprocessing)
imgDS.class_to_idx
```

```
{'negative': 0, 'positive': 1}
```

```python
# dataset에서 train, valid, test를 나누어보자
seed_gen = torch.Generator().manual_seed(42)
tr, val, ts = 0.7,0.1,0.2
trainDS, validDS, testDS = random_split(imgDS, [tr, val, ts], generator=seed_gen)
print(len(trainDS), len(validDS), len(testDS))
```

```
115 16 32
```

Dataset 생성 후

7 : 1 : 2 비율로 구분

train : valid : test = 7 : 1 : 2

Dataloader

```
# 각 분류별로 dataloader를 생성해보자
batch_size = 5
train_dl = DataLoader(trainDS, batch_size=batch_size, shuffle=True, drop_last = True)
valid_dl = DataLoader(validDS, batch_size=batch_size, shuffle=True, drop_last = True)
test_dl = DataLoader(testDS, batch_size = batch_size, shuffle=True, drop_last = True)
print(len(train_dl), len(valid_dl), len(test_dl))
```

배치 사이즈 5로 설정

drop_last = True

전이학습

```
# 모델 인스턴스 생성 : 사전 학습된 모델 로딩 => 가중치를 조절
res_model = models.resnet18(weights = "ResNet18_Weights.DEFAULT")

# 전결합층 변경
res_model.fc = nn.Linear(in_features = 512, out_features = 2)
```

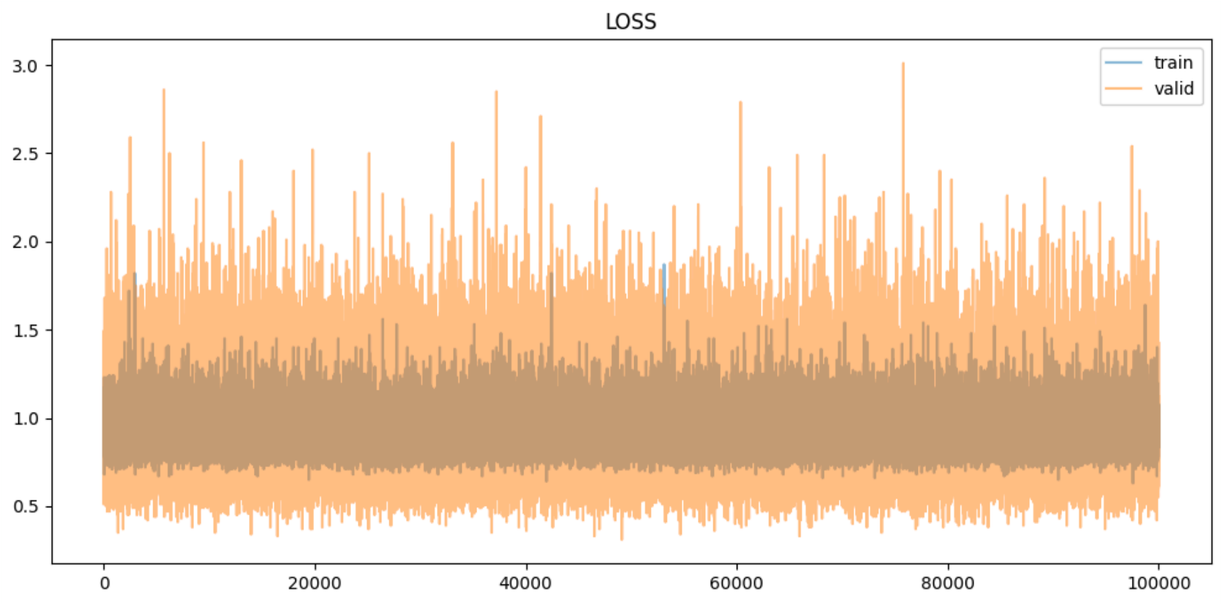
```
# 모델의 합성곱층 가중치 고정
for param in res_model.parameters():
    param.requires_grad = False
for param in res_model.fc.parameters():   # 완전 연결층은 학습
    param.requires_grad = True
```

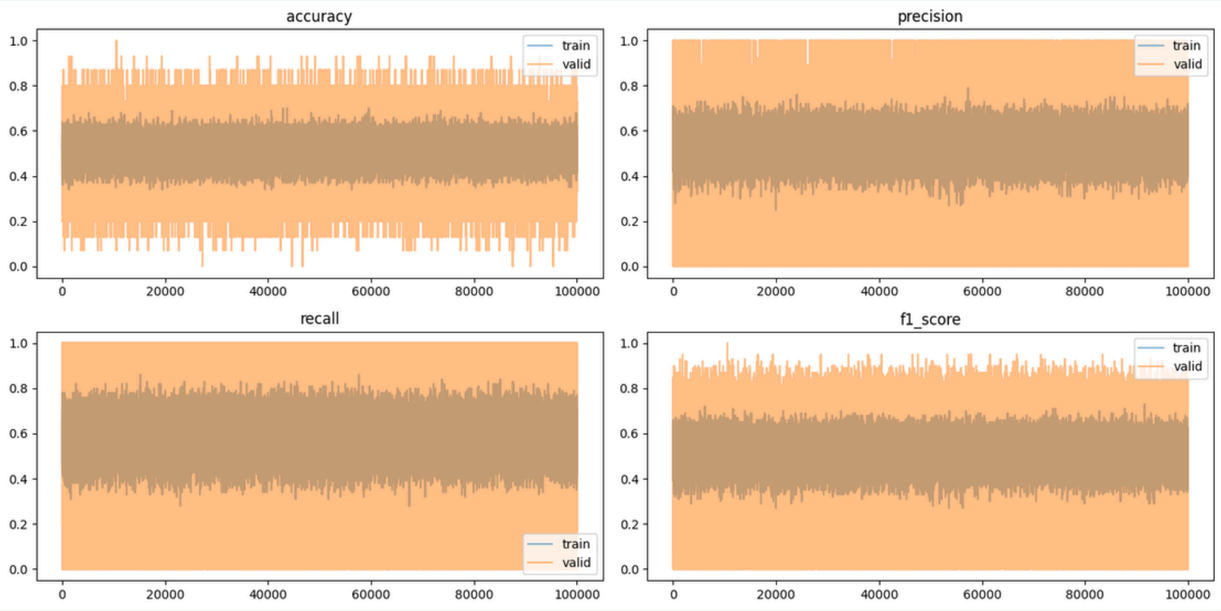
resnet18

- 전결합층 변경 : out_features = 2
- 합성곱층 가중치 고정시킴

학습

하이퍼파라미터	설정값
device	cuda
loss function	BCELoss
optimizer	Adam
epoch	100,000
scheduler	MultiStepLR





성능 - loss & metrics 확인

평균치	train	valid
loss	0.93	0.95
accuracy	0.51	0.51
precision	0.55	0.58
recall	0.59	0.57
f1 - score	0.52	0.54

predict

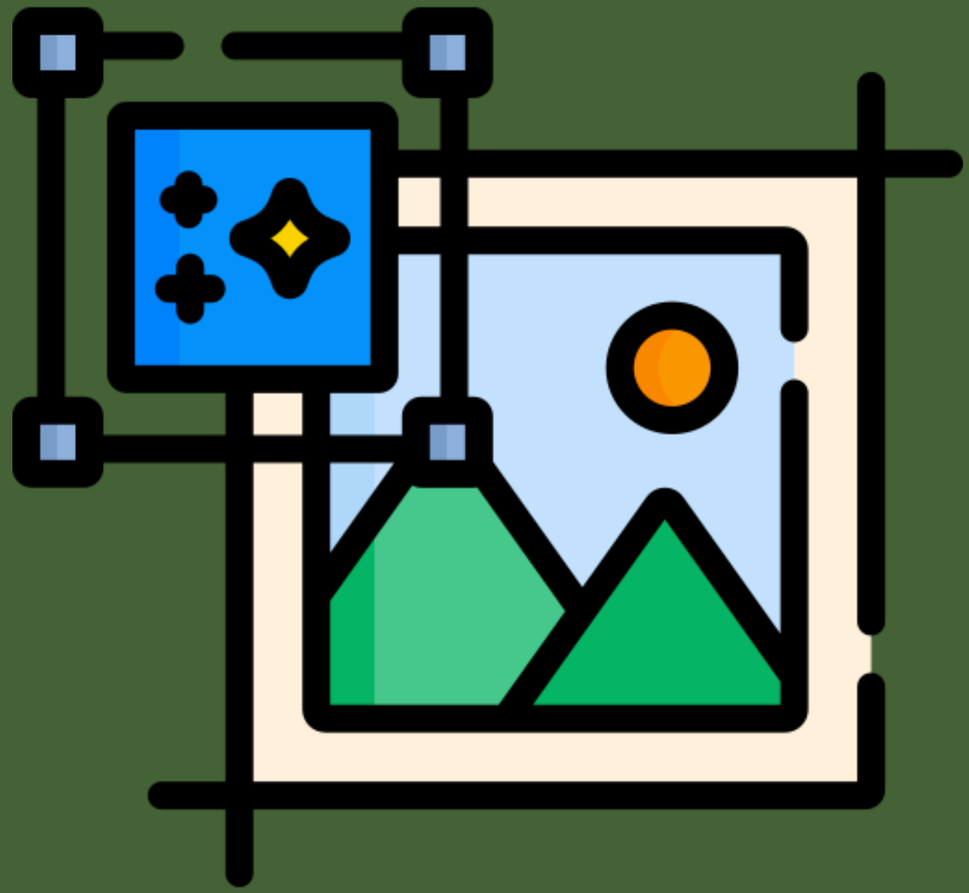
평균치	train	valid	test
loss	0.93	0.95	0.84
accuracy	0.51	0.51	0.47
precision	0.55	0.58	0.58
recall	0.59	0.57	0.45
f1 - score	0.52	0.54	0.46

loss 값에서 더 좋은 성능이 나왔다.

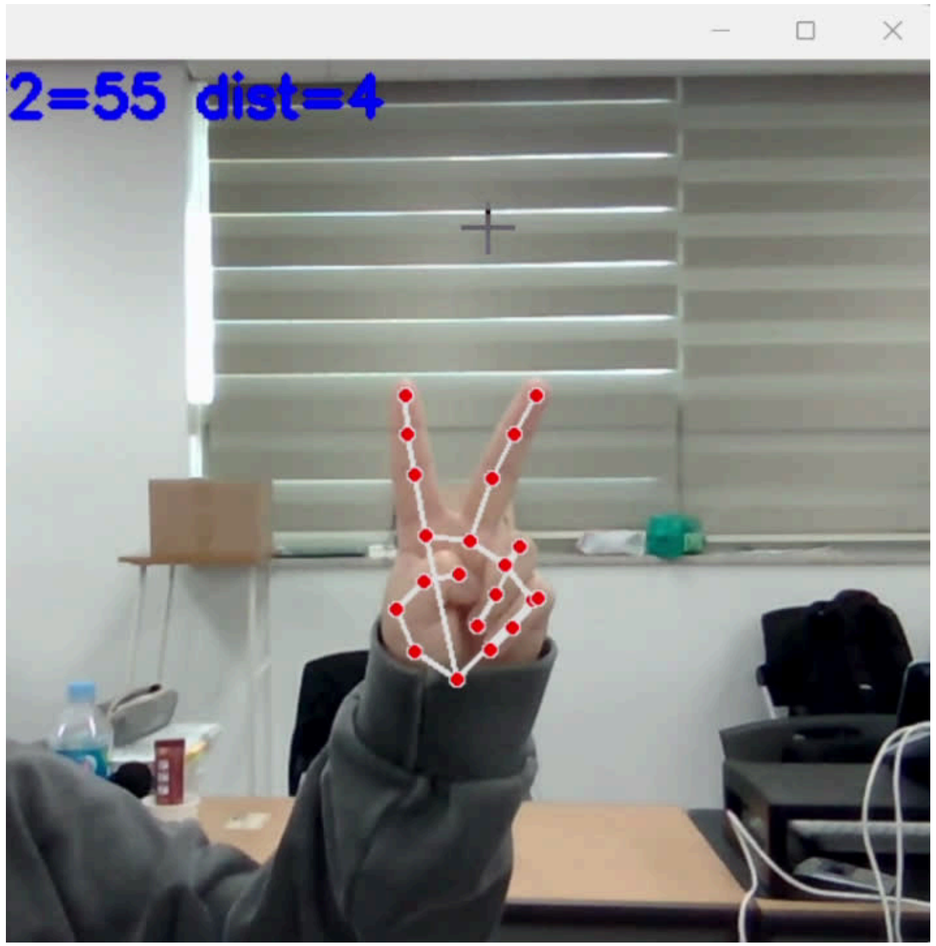
하지만 이 외의 값들은 오히려 더 나쁜 성능이 나왔다.

✳ 24.04.30.

객체 인식



mediapipe 란?



▲ 손을 인식한 샘플 영상

- Google에서 개발한 오픈 소스 프레임워크
- 실시간으로 멀티미디어 데이터를 처리하고 분석하는 기능을 제공
- 얼굴 인식, 제스처 인식, 객체 감지, 모션 트래킹 등 다양하게 사용됨

관련 모듈 및 클래스를 변수로 지정

```
import cv2
import mediapipe as mp
import pyautogui
import math

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose
```

MediaPipe에서 감지된 랜드마크, 연결선 등을 시각화하는 기능을 제공.

시각화 시 사용되는 다양한 스타일(두께, 색상 등)을 정의하는 상수 제공

MediaPipe의 포즈 추정(Pose Estimation) 기능을 제공하는 주요 클래스와 함수들을 포함

거리 구하기

카메라
켜기

이미지 변환
BGR -> Gray)

색을 2진화
(200 이상 => 255)

외곽선
찾기

START

실제 가로 길이와 픽셀
값 사이의 비율 구함

일정 조건에 해당하면
가로 길이의 픽셀값 반환

가로 - 세로
비율 계산

외곽선마다 좌표
와 크기 얻음

자세 탐지

정규화된 좌표값을
원래 값으로 반환

좌표 값을 이용하여
거리값을 구함

픽셀 단위로 구해진 것을
cm단위로 변환

END

거리가 2.5cm 이상이 될
때마다 알림창 발생

pyautogui.alert 사용

최종 거리값을 이미지와
함께 출력

정규화된 좌표값을 원래 값으로 반환

```
# x, y 는 [0.0, 1.0]으로 정규화됨
# z 는 엉덩이 중간 지점의 깊이를 원점으로 하는 랜드마크의 깊이 나타냄
img_h, img_w, _ = image.shape
focal_length = 1 * img_w

# 픽셀 대 cm 비율 계산
pixel_to_cm_ratio = ref_real_width / ref_pixel_width
e_left_x = int(x=results.pose_landmarks.landmark[7].x * img_w)
e_left_y = int(x=results.pose_landmarks.landmark[7].y * img_h)
e_left_z = results.pose_landmarks.landmark[7].z * focal_length

e_right_x = int(x=results.pose_landmarks.landmark[8].x * img_w)
e_right_y = int(x=results.pose_landmarks.landmark[8].y * img_h)
e_right_z = results.pose_landmarks.landmark[8].z * focal_length

s_left_x = int(x=results.pose_landmarks.landmark[11].x * img_w)
s_left_y = int(x=results.pose_landmarks.landmark[11].y * img_h)
s_left_z = results.pose_landmarks.landmark[11].z * focal_length

s_right_x = int(x=results.pose_landmarks.landmark[12].x * img_w)
s_right_y = int(x=results.pose_landmarks.landmark[12].y * img_h)
s_right_z = results.pose_landmarks.landmark[12].z * focal_length
```

좌표 값을 이용하여 거리값을 구함

```
# 귀와 어깨 사이 거리값 구하기 (픽셀 단위)
left_dist_pixel = pointDist(x1=e_left_x, y1=e_left_y, z1=e_left_z, x2=s_left_x, y2=s_left_y, z2=s_left_z)
right_dist_pixel = pointDist(x1=e_right_x, y1=e_right_y, z1=e_right_z, x2=s_right_x, y2=s_right_y, z2=s_right_z)
```

픽셀 단위로 구해진 것을 cm 단위로 변환

```
# 거리값을 cm 단위로 변환
left_dist_cm = round(number=left_dist_pixel * pixel_to_cm_ratio, ndigits=3)
right_dist_cm = round(number=right_dist_pixel * pixel_to_cm_ratio, ndigits=3)
```

개선점 및 수정사항

(1) 다른 전이학습 모델 사용해보기

기존에 사용했던 Resnet 모델의 종류를 바꿔보거나 VGG와 같은 다른 종류를 활용해보기

(2) 알림 간격 조정

너무 잦은 알람 간격으로 인해 오히려 방해가 되는 것이 있음 좀 더 간격 조정이 필요함

(3) 사용자가 이용 후 종료 방식 재지정

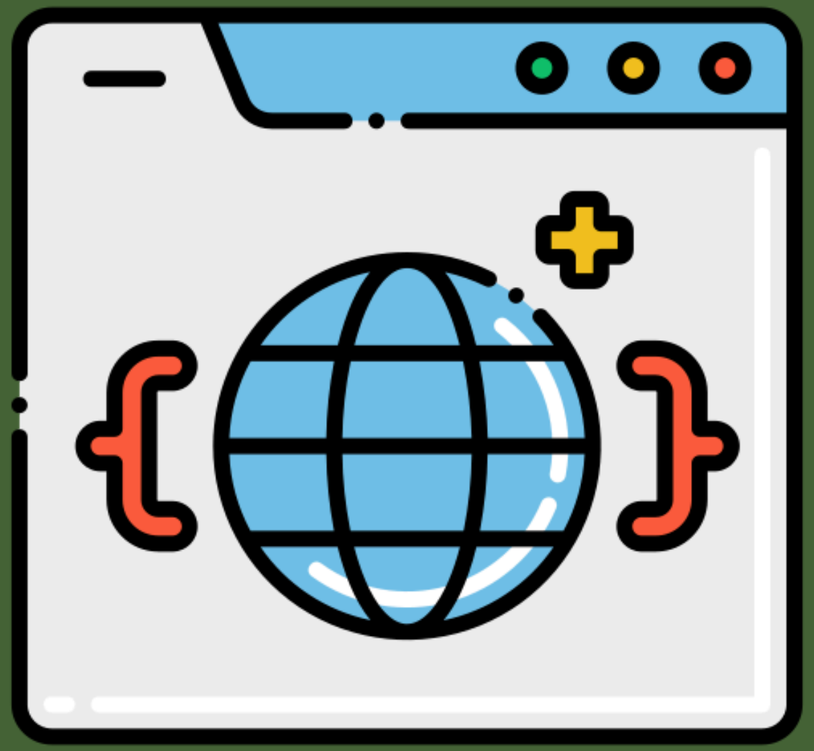
현재는 잦은 알람 간격으로 인해 사용자가 종료하기 힘든 구성이다. 이에 대한 다른 방안이 필요함

(4) 사용자와 노트북 거리 측정 방식 재고려

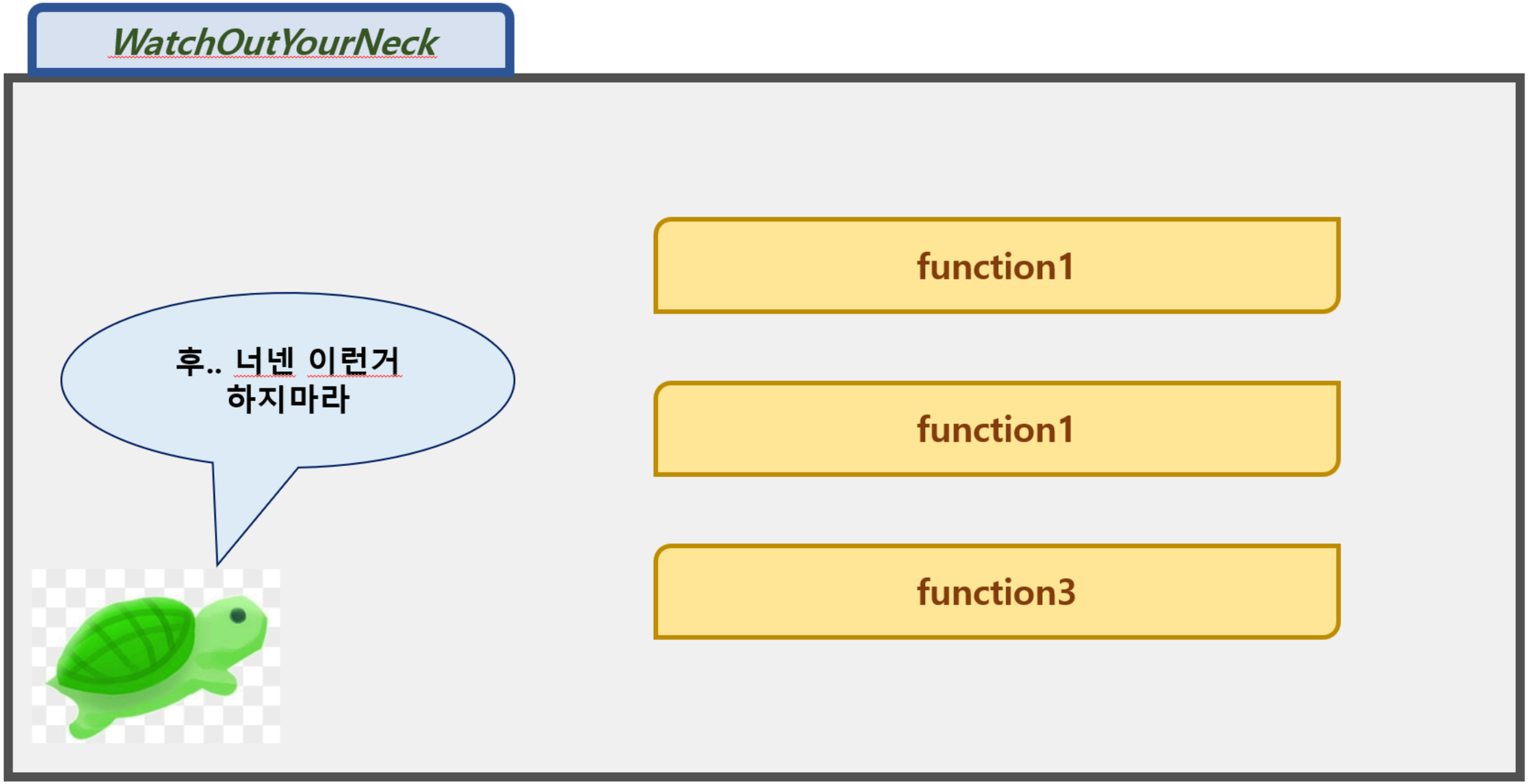
지금의 A4용지를 활용하는 방안은 정확도가 낮아서 거북목 탐지에 도움이 되지 않는 편이다.
이에 대해 좀 더 고민해보고 더 나은 거리 측정 및 거북목 탐지가 필요한 바이다.

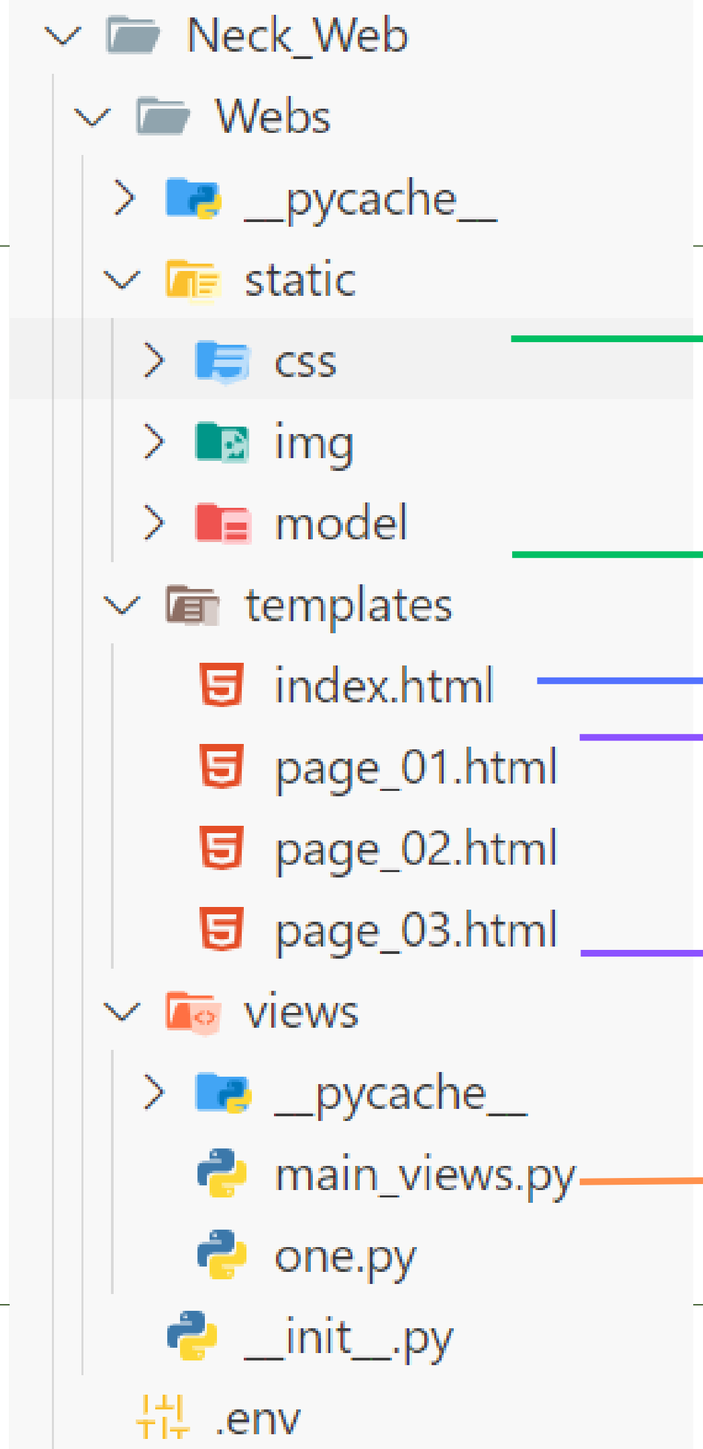
✳ 24.04.30.

웹 설계



web 구성


WatchOutYourNeck
후.. 너넨 이런거
하지마라
function1
function1
function3



flask에 사용될 source들 정리

- css파일
- 업로드 되는 이미지 저장
- 사용될 모델

main page 구성

기능별 sub page 구성

만들어진 기능들 함수로 구현하여 blueprint로 생성

✳ 24.04.30.

DEMONSTRASION