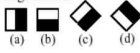
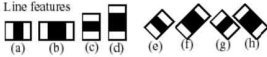

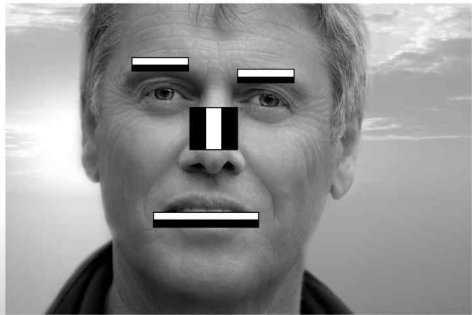
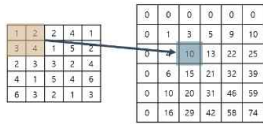
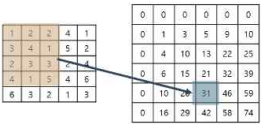
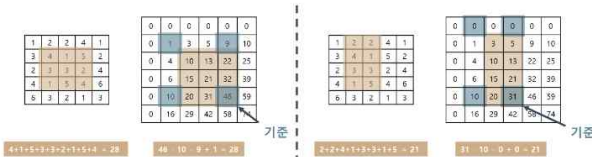


## 2024-1학기 창의학기제 주간학습보고서 (2주차)

창의과제	세종대학교 집현캠퍼스를 개선시킨 웹서비스 개발				
이름	신찬영	학습기간	3월 4일 ~ 3월 15일		
학번	23012094	학습주차	2	학습시간	3
학과(전공)	인공지능	과목명	자기주도창의전공1	수강학점	3
※ 수강학점에 따른 회차별 학습시간 및 10주차 이상 학습 준수					
금주 학습목표	사진에서 얼굴만을 따주는 객체인식 모델을 찾고 학습시키기				
학습내용	<p>얼굴이나 사물을 인식하는 알고리즘을 물체 검출 알고리즘이라한다. 우리는 물체 특징 기반 검출 알고리즘인 Haar cascade이용하여 얼굴을 따낼 것이다.</p> <p>1. Edge features  </p> <p>2. Line features  </p> <p>3. Center-surround features  </p> <p>Haar cascade는 위의 사진의 feature를 이용해서 특징을 추출해낸다. 이 feature들이 이미지를 스캔하면서 인접한 feature 내의 Pixel값의 합끼리 비교하는 방식을 이용한다.</p>  <p>이 사진을 보면 feature들이 어두운 곳은 검은색으로 밝은 곳은 하얀색으로 표시한 것을 볼 수 있다. 즉 명도의 차이를 통해 부위나 사물을 검출해 낸다. feature들을 작게 하면 보다 세세하게 분석할 수 있지만, 연산량이 어마어마해진다. 또한 이미지의 픽셀값이 많아도(즉 해상도가 높아도) 연산량이 어마어마해진다. 이런 연산량이 많다는 단점을 해결하기 위해 Haar cascade는 “integral image”를 사용한다.</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p><math>1+2+3+4 = 10</math></p> </div> <div style="text-align: center;">  <p><math>1+2+2+3+4+1+2+3+3+4+1+5 = 31</math></p> </div> </div> <p>만약 위 사진의 1이 (0,0)이라한다면 “integral image”는 (x,y)의 값에다 (0,0)의 값부터 (x,y)</p>				

의 값까지 모두 더한 값을 집어 넣은 이미지에다 맨 왼쪽과 위쪽에 0을 추가한 이미지라 보면 된다.



이 이미지가 연산량을 줄여주는 이유는 위 사진처럼 기준을 중심으로 위와 왼쪽값은 빼주고 왼쪽 대각선의 값을 더해주면 그 영역의 값이 나온다. 이러한 방식으로 연산량을 줄였다.

```
# import cv2
# import os
# from pathlib import Path

# def save_detect_image(folder_path, padding_factor=0.1):
#     if folder_path[-1] == '/':
#         save_folder_path = folder_path[:-1] + '_face'
#     else:
#         save_folder_path = folder_path + '_face'
#     Path(save_folder_path).mkdir(parents=True, exist_ok=True)

#     cascade_path = cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
#     face_cascade = cv2.CascadeClassifier(cascade_path)

#     for filename in os.listdir(folder_path):
#         image_path = os.path.join(folder_path, filename)
#         image = cv2.imread(image_path)
#         if image is None:
#             continue
#         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#         faces = face_cascade.detectMultiScale(gray, 1.1, 4)

#         for (x, y, w, h) in faces:
#             padding_w = int(w * padding_factor)
#             padding_h = int(h * padding_factor)
#             x_pad = max(x - padding_w, 0)
#             y_pad = max(y - padding_h, 0)
#             w_pad = min(w + 2 * padding_w, image.shape[1] - x_pad)
#             h_pad = min(h + 2 * padding_h, image.shape[0] - y_pad)

#             face_img_with_padding = image[y_pad:y_pad+h_pad, x_pad:x_pad+w_pad]

#             save_path = os.path.join(save_folder_path, filename)
#             cv2.imwrite(save_path, face_img_with_padding)

#     return save_folder_path
```

실습 코드는 이러하다.

학습방법

haar cascade모형을 아래 링크에서 다운 받았다.  
[https://github.com/BaekKyunShin/Computer-Vision-Basic/blob/main/Project1-Face\\_Detection/Haar\\_Cascade\\_Face\\_Detection.ipynb](https://github.com/BaekKyunShin/Computer-Vision-Basic/blob/main/Project1-Face_Detection/Haar_Cascade_Face_Detection.ipynb)  
 다운받은 haarcascade모형을 이용해 사진에서 얼굴을 따준다.  
 이렇게 따낸 사진들을 MovableNet\_V3에 학습시켰다. 배경이나 옷의 종류의 특징을 배제시켜서 학습시키니 정확도가 상승하였다. 즉 다른 사람이어도 옷의 종류가 같으면 같은 사람으로 인식하는 문제를 해결하였다.

학습성과  
및  
목표달성도

얼굴 분류 모델의 정확도가 대략 0.7에서 0.9 정도로 올라갔다. 그러나 haar cascade모델이 오래된 모델이라 기존 사진에서 얼굴을 따내는 과정에서 얼굴이 아닌 눈을 따내는등의 오류가 있었다. 그리고 이렇게 잘린 사진을 MovableNet에 학습 시켰다. 분류기로서의 성능을 올라갔지만 MovableNet 자체가 얼굴의 생체인식에 사용하기에 부적합하였다.

참고자료  
및 문헌

[https://github.com/BaekKyunShin/Computer-Vision-Basic/blob/main/Project1-Face\\_Detection/Haar\\_Cascade\\_Face\\_Detection.ipynb](https://github.com/BaekKyunShin/Computer-Vision-Basic/blob/main/Project1-Face_Detection/Haar_Cascade_Face_Detection.ipynb)  
<https://kay-dev.tistory.com/entry/Open-CV-Haar-Cascade%EC%99%80-%EC%96%BC%EA%B5%B4-%EC%9D%B8%EC%8B%9D-Ft-Viola-Jones>

내주 계획

MovableNet 말고 얼굴 보안 인식에 적합한 신경망을 찾아볼 것이다. 또한 haecascade모델



세종대학교

의 오류 등을 보완한 모델을 찾아 우리가 만들거나 찾은 데이터로 학습시켜 위 모델을 대체할 예정이다.

년 월 일

지도교수

(인)