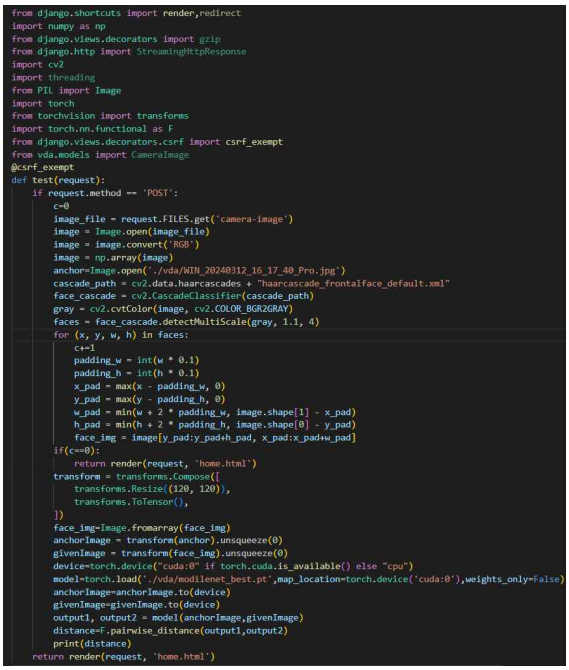
세종대학교

팀번호	팀번호 입력

2024-1학기 창의학기제 주간학습보고서 (10주차)

창의과제	세종대학교 집현캠퍼스를 개선시킨 웹서비스 개발				
이름	신찬영	학습기간	6월 1일 ~ 6월 9일		
학번	23012094	학습주차	10	학습시간	3
학과(전공)	인공지능	과목명	자기주도창의전공1	수강학점	3
※ 수강학점에 따른 회차별 학습시간 및 10주차 이상 학습 준수					
금주 학습목표	사용자의 웹캠을 통해 사진을 받고 그것을 서버에서 샴네트워크를 사용해 사용자가 맞는지 아닌지 확인하고 맞으면 로그인 페이지로 이동하게 하는 코드를 작성한다.				
학습내용	project와 app을 만들고 url을 지정해주는 것은 저번 주차에서 설명했던 것과 똑같다. 저번 주차에서 달라진 것은 html과 view의 코드와 사용한 딥러닝 모델이 달라졌다.  다음은 바뀐 view의 파일이다. 먼저 'POST'형식일 때(HTML에서 fetch으로 데이터를 받을 때) request에서 file을 받는다 image file을 Image 객체를 이용하여 연다. 연 image를 RGB의 형태로 바꿔주고 이것을 다시 넘파이 배열로 바꿔준다. 샴 네트쿼트를 사용하기 위해 이미 서버에 저장된 anchor 이미지를 열어준다. 이후 harcascade을 이용해 받은 사진을 잘라준다. 잘라준 사진과 anchor 사진을 transform을 통래 120 120 크기로 설정하고 텐서 형태로 바꿔준다. 이후 샴네트워크 모델을 로드한 다음 통과시켜 distance를 출력하게 해준다.				

```python
from django.shortcuts import render,redirect
import numpy as np
from django.views.decorators import gzip
from django.http import StreamingHttpResponse
import cv2
import threading
from PIL import Image
import torch
from torchvision import transforms
import torch.nn.functional as F
from django.views.decorators.csrf import csrf_exempt
from vda.models import CameraImage
@csrf_exempt
def test(request):
    if request.method == 'POST':
        c=0
        image_file = request.FILES.get('camera-image')
        image = Image.open(image_file)
        image = image.convert('RGB')
        image = np.array(image)
        anchor=Image.open('./vda/WIN_20240312_16_17_40_Pro.jpg')
        cascade_path = cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
        face_cascade = cv2.CascadeClassifier(cascade_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.1, 4)
        for (x, y, w, h) in faces:
            c+=1
            padding_w = int(w * 0.1)
            padding_h = int(h * 0.1)
            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 = image[y_pad:y_pad+h_pad, x_pad:x_pad+w_pad]
        if(c==0):
            return render(request, 'home.html')
        transform = transforms.Compose([
            transforms.Resize((120, 120)),
            transforms.ToTensor(),
        ])
        face_img=Image.fromarray(face_img)
        anchorImage = transform(anchor).unsqueeze(0)
        givenImage = transform(face_img).unsqueeze(0)
        device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
        model=torch.load('./vda/modilenet_best.pt',map_location=torch.device('cuda:0'),weights_only=False)
        anchorImage=anchorImage.to(device)
        givenImage=givenImage.to(device)
        output1, output2 = model(anchorImage,givenImage)
        distance=F.pairwise_distance(output1,output2)
        print(distance)
    return render(request, 'home.html')
```

```html
<html>
  <head>
    <title>세종 얼굴 인식</title>
  </head>
  <body>
    <h1>얼굴을 인식하는 중입니다.</h1>
    <div>
      <video id="video" width="800" height="600" autoplay></video>
      <br>
      <button onclick="captureImage()">수동인식</button>
    </div>
    <div>
      <br>
    </div>
    <script>
      const video = document.getElementById("video");
      const canvas = document.createElement('canvas');
      const context = canvas.getContext('2d');

      navigator.mediaDevices.getUserMedia({ video: true })
        .then(stream => {
          video.srcObject = stream;
        });

      function captureImage() {
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        context.drawImage(video, 0, 0);
        canvas.toBlob(blob => {
          const formData = new FormData();
          formData.append('camera-image', blob);
          fetch("/vda/", {
            method: 'POST',
            body: formData
          });
        });
      }

      setInterval(captureImage, 500);
    </script>
  </body>
</html>
```

바뀐 html 코드이다 먼저 비디오를 통해 받은 사진을 capture image()라는 자바스크립트 함수로 보낸다. 보낸 이미지는 feteh 형태로 바뀌어서 POST형식으로 해당 url로 보낸다. 그러면 view함수에서 해당 이미지 데이터를 받아 작업을 처리한다.

위의 방식은 hascascade가 디바이스 사진이나 복사한 얼굴 이미지 등을 따로 구분하지 못해 적합한 방식이 아니다. 따라서 보안을 위해 디바이스나 가면 복사된 사진을 감지하는 YOLO를 추가해 줘야한다.

```python
from facenet_pytorch import InceptionResnetV1
class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()
        # self.model=mobilenet_v3_large(pretrained=True)
        self.model=InceptionResnetV1(pretrained='vggface2')

        self.fc = nn.Sequential(
            nn.Linear(960, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 128),
            #nn.ReLU(inplace=True),
            #nn.Linear(256, 128),
        )

        self.model.classifier=self.fc
    def forward(self, input1, input2):
        output1 = self.model(input1)
        output2 = self.model(input2)
        return output1, output2
```

먼저 샴네트워크에 사용된 pre-trained model이 달라졌다. 기존에는 mobileNet을 활용했지만 정확성을 더 높이기 faceNet을 사용했다.

```python
@csrf_exempt
def test(request):
    if request.method == 'POST':
        model_yolo=YOLO('./vda/best.pt')
        c=0
        image_file = request.FILES.get('camera-image')

        image = Image.open(image_file)
        image = image.convert('RGB')
        image = np.array(image)
        yolo_result=model_yolo.predict(image)
        clss=yolo_result[0].boxes.cls.cpu().tolist()[0]
        print(int(clss))
        if(int(clss)==1 or int(clss)==3 or int(clss)==4 or int(clss)==5):
            return render(request, 'vda/home.html')
        anchor=Image.open('./vda/WIN_20240312_16_17_40_Pro.jpg')
        cascade_path = cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
        face_cascade = cv2.CascadeClassifier(cascade_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.1, 4)
        for (x, y, w, h) in faces:
            c+=1
            padding_w = int(w * 0.1)
            padding_h = int(h * 0.1)
            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 = image[y_pad:y_pad+h_pad, x_pad:x_pad+w_pad]
        if(c==0):
            return render(request, 'vda/home.html')
        transform = transforms.Compose([
            transforms.Resize((160, 160)),
            transforms.ToTensor(),
        ])
        face_img=Image.fromarray(face_img)
        anchorImage = transform(anchor).unsqueeze(0)
        givenImage = transform(face_img).unsqueeze(0)
        device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
        model=torch.load('./vda/facenet_best.pt',map_location=torch.device('cuda:0'),weights_only=False).t
        anchorImage=anchorImage.to(device)
        givenImage=givenImage.to(device)
        output1, output2 = model(anchorImage,givenImage)
        distance=F.pairwise_distance(output1,output2)
        print(distance)
        if distance<0.6:
            redirect('done/')
    return render(request, 'vda/home.html')
```

이것은 바꾼 view파일이다. 달라진 점은 디바이스같은 여러 보안 부정행위를 감지할 수 있는 커스텀 학습된 YOLOv8 모델을 추가한 것과 샴네트워크 모델이 facenet을 사용한 모델로 바뀌었다는 것이다. 과정을 설명하자면 받은 이미지 파일을 열고 이미지를 먼저 YOLO를 통과 시켜 부정행위인지 아닌지 감지하고 부정행위가 아니면 harcascade를 통해 사진을 자른 후 샴네트워크 모델을 통해 distance를 구해 distance가 0.6보다 작으면 done/링크로 redirect해주는 코드이다.

이 코드에 치명적인 문제점을 발견했다. html에서 fetch형식으로 받아서 반환할때고 fetch 형식으로 반환해 제대로 redirect되지 않을뿐더러 html도 사용자에게 제대로 반환하지 못한다. 또한 YOLO->harcascade->siamnetwork로 3개의 모델을 통과하기 때문에 리소스의 소모가 심하다.

```html
<html>
  <head>
    <title>세종 얼굴 인식</title>
  </head>
  <body>
    <h1>얼굴을 인식하는 중입니다.</h1>
    <div>
      <video id="video" width="800" height="600" autoplay></video>
      <br>
      <button onclick="captureImage()">수동인식</button>
    </div>
    <div>
      <br>
    </div>
    <script>
      const video = document.getElementById("video");
      const canvas = document.createElement('canvas');
      const context = canvas.getContext('2d');

      navigator.mediaDevices.getUserMedia({ video: true })
        .then(stream => {
          video.srcObject = stream;
        });

        function captureImage() {
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    context.drawImage(video, 0, 0);
    canvas.toBlob(blob => {
        const formData = new FormData();
        formData.append('camera-image', blob);
        fetch("/vda/video/", {
            method: 'POST',
            body: formData
        }).then(response => response.text())
          .then(data => {
            console.log(data); // 서버 응답 로깅
            if (data == 'Done') {
                // 추가적인 사용자 알림 또는 동작
                window.location.href = "http://127.0.0.1:8000/vda/done/done2";
            }
            else if (data=='detect dvice') {
              alert('디바이스 감지 및 부정 행위 감지');
            }
          });
    });
}
    setInterval(captureImage,500)
    </script>
  </body>
</html>
```

바뀐 html 코드이다. 기존 코드에서 fetch형태로 반환된 문자열을 받아 그 문자열이 특정 조건을 충족하면 사용자에게 부정행위가 감지되었다고 알리거나 로그인된 링크로 이동하게 하는 코드가 추가되었다. 따라서 자바스크립트를 이용해 기존에 할 수 없었던 redirect를 할 수 있다.

세종대학교

```python
def test(request):
    if request.method == 'POST':
        model_yolo=YOLO('./vda/best.pt')
        c=0
        image_file = request.FILES.get('camera-image')
        image = Image.open(image_file)
        image = image.convert('RGB')
        image = np.array(image)
        yolo_result=model_yolo.predict(image)
        clss=yolo_result[0].boxes.cls.cpu().tolist()[0]
        if(int(clss)==1 or int(clss)==3 or int(clss)==4 or int(clss)==5):
            return HttpResponse('detect dvice')
        boxes = yolo_result[0].boxes.xyxy.cpu().tolist()
        anchor=Image.open('./vda/WIN_20240312_16_17_40_Pro.jpg')

        for (x, y, x1, y1) in boxes:
            c+=1
            pad=y1-y-x1+x
            face_img = image[int(y):int(y1), int(x-pad/2):int(x1+pad/2)]
        if(c==0):
            return HttpResponse('no')
        transform = transforms.Compose([
            transforms.Resize((160, 160)),
            transforms.ToTensor(),
        ])
        face_img=Image.fromarray(face_img)
        anchorImage = transform(anchor).unsqueeze(0)
        givenImage = transform(face_img).unsqueeze(0)
        device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
        model=torch.load('./vda/facenet_best.pt',map_location=torch.device('cuda:0'),weights_only=False).t
        anchorImage=anchorImage.to(device)
        givenImage=givenImage.to(device)
        output1, output2 = model(anchorImage,givenImage)
        distance=F.pairwise_distance(output1,output2)
        print(distance)
        if distance<0.6:
                return HttpResponse('Done')
    return render(request, 'vda/home.html')
def index(request):
    return render(request,'vda/home_2.html')
def index1(request):
    return HttpResponse('Done')
```

바뀐 view파일이다. hacascade를 쓰는 과정이 없어지고 사진의 얼굴만 자르는 기능을 yolo가 같이 하도록 통합되었다. 따라서 기존의 방식보다 리소스를 아낄 수 있다.

학습방법	인터넷을 검색하거나 stackoverflow등을 참고하여 제대로 redirect되지 않는 문제를 해결했다. harcascade를 없애고 yolo로 통합하는 방식은 기존에 배웠던 지식들을 활용했다.(yolo를 공부했던 주차)
학습성과 및 목표달성도	성공적으로 부정행위가 감지되면 얼굴 인식 시스템이 멈추고 사용자의 얼굴을 인식하여 사용자의 얼굴이 맞다고 판단되면 다른 링크로 넘겨주는 웹앱 시스템이 드디어 완성되었다.
참고자료 및 문헌	https://lbdiaryl.tistory.com/367(작성자가 글을 삭제하여 지금은 볼 수 없다.) https://stackoverflow.com/questions/3024168/how-to-redirect-with-post-data-django
내주 계획	나의 로그인 시스템을 나머지 팀원들이 만든 집현캠퍼스와 통합한다.

2024 년 4 월 2 일

지도교수 (인)