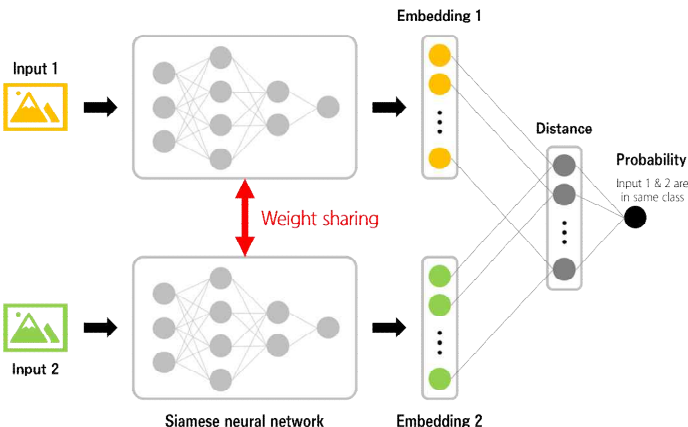


2024-1학기 창의학기제 주간학습보고서 (3주차)

창의과제	세종대학교 집현캠퍼스를 개선시킨 웹서비스 개발				
이름	이지민	학습기간	4월 1일 ~ 4월 12일		
학번	23012127	학습주차	3	학습시간	3
학과(전공)	인공지능	과목명	자기주도 창의전공1	수강학점	3
※ 수강학점에 따른 회차별 학습시간 및 10주차 이상 학습 준수					
금주 학습목표	생체인식을 위한 안면인식 모델을 만들기위해 어떤 방법을 사용할 수 있는지 알아본다.				
학습내용	<p>1주차때 MobileNetV3를 활용한 얼굴 분류를 실행해본 결과, 분류를 사용할 경우 label을 계속 추가해줘야 한다는 점을 알게되어 이를 해결하기 위해 알아본 결과 siamese network에 대해 알게 되었다.</p> <p>siamese network는 CNN을 통과한 두 이미지 데이터의 거리차를 이용하여 학습시키는 모델이다. 오차 함수로는 contrastive loss 혹은 triplet loss를 활용한다.</p>  <p>그림 1 siamese network 구조</p> $contrastiveloss(i, j) = y_{ij}d_{ij}^2 + (1 - y_{ij})\max(\alpha - d_{ij}^2, 0)$ $tripletloss(A, P, N) = \max(\ f(A) - f(P)\ _2 - \ f(A) - f(N)\ _2 + \alpha, 0)$ <p>수식 1 contrastive loss와 triplet loss의 수식</p> <p>먼저 contrastive loss를 분석해보면 label이 1이면 현재 거리만큼 오차값을 줌으로써 더 가 가까워지도록 label이 0이면 α만큼만 가까워지도록 학습시키는 오차함수라는 것을 알 수 있다. triplet loss는 Anchor(기준이 되는 데이터), Positive, Negative 3개의 데이터를 입력으로 받아 anchor와 negative의 거리가 anchor와 positive의 거리보다 α만큼 벌어지도록 학습시키는 오차함수 라는 것을 알 수 있다.</p>				



이전에 mobileNetV3를 활용하기 위해 만든 model클래스를 siamese network로 만들기 위해 input과 output을 두 개로 늘렸다.

<siamese network class>

```
class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()
        self.model=mobilenet_v3_large(pretrained=True)
        self.fc = nn.Sequential(
            nn.Linear(960, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 5))
        self.model.classifier=self.fc
    def forward(self, input1, input2):
        output1 = self.model(input1)
        output2 = self.model(input2)
        return output1, output2
```

또한 이 모델에 데이터를 넣을 수 있도록 dataset도 새로 정의하였다. positive 데이터와 negative데이터를 정하여 positive와 positive 짝과 postive, negative짝으로 dataset을 만들어 주었다.

<수정한 dataset 코드>

```
class CatsDogsSiameseDataset(Dataset):
    def __init__(self, cat_directory, dog_directory, transform=None):
        self.cat_images = [os.path.join(cat_directory, img) for img in
os.listdir(cat_directory)]
        self.dog_images = [os.path.join(dog_directory, img) for img in
os.listdir(dog_directory)]
        self.count=0
        if len(self.cat_images) < len(self.dog_images):
            for i in range(0,len(self.dog_images)-len(self.cat_images)):
                if self.count==len(self.cat_images):
                    self.count=0
                    self.cat_images.append(self.cat_images[self.count])
                    self.count+=1
            elif len(self.cat_images) > len(self.dog_images):
                for i in range(0,len(self.cat_images)-len(self.dog_images)):
                    if self.count==len(self.dog_images):
                        self.count=0
                        self.dog_images.append(self.dog_images[self.count])
                        self.count+=1

        self.all_images = self.cat_images + self.dog_images
        self.transform = transform
        # 고양이는 0, 강아지는 1로 레이블 지정
        self.labels = [0] * len(self.cat_images) + [1] * len(self.dog_images)
```



	<pre> def __getitem__(self, index): # 첫 번째 이미지 선택 img1_path = self.all_images[index] label1 = self.labels[index] # 동일한 클래스 내의 이미지(양성 쌍) 또는 다른 클래스의 이미지(음성 쌍) 선택 should_get_same_class = random.randint(0, 1) if should_get_same_class: while True: # 같은 클래스에서 이미지 선택 index2 = random.choice(range(len(self.all_images))) if self.labels[index2] == label1: break else: while True: # 다른 클래스에서 이미지 선택 index2 = random.choice(range(len(self.all_images))) if self.labels[index2] != label1: break img2_path = self.all_images[index2] img1 = Image.open(img1_path) img2 = Image.open(img2_path) if self.transform is not None: img1 = self.transform(img1) img2 = self.transform(img2) return img1, img2, torch.from_numpy(np.array([int(label1) != self.labels[index2]]), dtype=np.float32)) def __len__(self): return len(self.all_images) </pre> <p>위에서 배운 contrastive loss 개념을 활용하고 nn.module을 상속하여 contrastive loss class를 만들어 criterion으로 설정하였다.</p> <p><contrastive loss></p> <pre> class ContrastiveLoss(torch.nn.Module): def __init__(self, margin=2.0): super(ContrastiveLoss, self).__init__() self.margin = margin def forward(self, output1, output2, label): euclidean_distance = F.pairwise_distance(output1, output2, keepdim = True) loss_contrastive = torch.mean((1-label) * torch.pow(euclidean_distance, 2) + (label) * torch.pow(torch.clamp(self.margin - euclidean_distance, min=0.0), 2)) return loss_contrastive </pre>
학습방법	인터넷 자료를 찾아보며 얼굴 생체 인식에 활용하는 기술을 알아보았다. pytorch document를 살펴보며 contrastive loss를 만들어 주었다.
학습성과	accuracy값은 만족스럽지 않게 나왔다. 하지만 두 사진의 distance값을 출력해본 결과



및 목표달성도	positive positive쌍과 positive negative쌍의 거리값이 차이가 나는 것으로 보이므로 다른 하 이퍼파라미터에 대한 고민을 좀 더 해야할 것 같다.
참고자료 및 문헌	siamese network에 대한 참고자료 https://tyami.github.io/deep%20learning/Siamese-neural-networks/ contrastive loss와 triplet loss에 대한 참고 자료 https://iambeginnerdeveloper.tistory.com/198 https://www.youtube.com/watch?v=d2XB5-tuCWU https://www.v7labs.com/blog/triplet-loss#:~:text=Triplet%20loss%20is%20a%20way,Basic%20idea%20of%20triplet%20loss
내주 계획	contrastive loss의 margin값 수정 및 정확도를 좀 더 높일 방법을 탐구한다.

년 월 일

지도교수

(인)