[2023 영상처리와 딥러닝 기말프로젝트]
few-shot learning
- Humpback Whale Identification Challenge-

2023.12.05.(화)

한림대학교 컴퓨터공학과
M23048 이가현

1. 대회 평가지표 분석
Mean Average Precision @ 5 (MAP@5)

1.1. Evaluation

1.1.1. Mean Average Pprecision(MAP)

: 여러 쿼리에 대한 평균 정확도를 측정하는 지표. 각 쿼리의 정확한 결과 위치를 기준으로 정확도를 계산해 평균을 낸다.

1.1.2. @5

: 상위 5개의 결과만을 평가 대상으로 삼는다. 이는 각 이미지별로 상위 5개 예측만을 고려하여 평가함을 의미한다.

1.1.3. 계산 방법

$$MAP@5 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,5)} P(k)$$

U: 이미지의 총 수

P(k): k번째에서의 정밀도(Precision)

min(n,5): 각 이미지에 대한 예측의 수 n과 5중 더 작은 값

수식을 통해서 모든 이미지에 대한 평균 정밀도를 계산한다. 각 이미지에 대해, 상위 5개의 예측 중 실제로 관련 있는 결과에 대해 정밀도를 계산하고, 그 정밀도들의 평균값을 취한다. 이 평균값이 MAP@5의 최종 값이 된다. 즉, MAP@5는 여러 데이터 포인트에 대해 모델이 상위 5개의 예측 중 얼마나 잘 맞추는지를 평균적으로 나타내는 지표이다.

1.1.4. 계산 예시

A의 실제 레이블: X/ A에 대한 모델의 예측: {W, X, Y, Z, N}
→ 정확한 예측이 두 번째에 있으므로 정밀도는 1/2
B의 실제 레이블: T/ B에 대한 모델의 예측: {T, S, R, Q, P}
→ 정확한 예측이 첫 번째에 있으므로 정밀도는 1/1
C의 실제 레이블: M/ C에 대한 모델의 예측: {A, B, C, D, M}
→ 정확한 예측이 다섯 번째에 있으므로 정밀도는 1/5

따라서, MAP@5 = (1/2 + 1/1 + 1/5) / 3 이다.

1.2. Submission file

: 각 이미지에 대해 최대 5개의 label을 예측할 수 있다. 훈련 데이터에 없는 고래는 'new_whale'로 label을 붙여야 한다. 'new_whale' label은 훈련 데이터에 없는 새로운 고래를 얼마나 잘 감지하는지 평가하는 지표이다. 즉, 모델이 보지 못한 데이터에 대해 얼마나 잘 예측하는지를 보여준다.

2. Metric learning에 대한 조사

2.1. Metric learning의 개념

딥러닝 모델의 성능을 위해서는 많은 양의 데이터가 필요하다. 그래서 충분한 양의 데이터를 사용할 수 없는 경우 좋은 성능을 보장할 수 없다. 이처럼 딥러닝 모델은 데이터 의존도가 높다는 특성을 가진다.

Meta learning는 학습을 위한 학습(Learning to learn)이라고 할 수 있다. 이를 통해 적은 데이터로도 일반화(Few-shot learning)가 가능하게 하도록 학습(Meta learning)시키는 것이 목표이다. Metric learning의 개념을 아래 그림으로 설명하겠다.

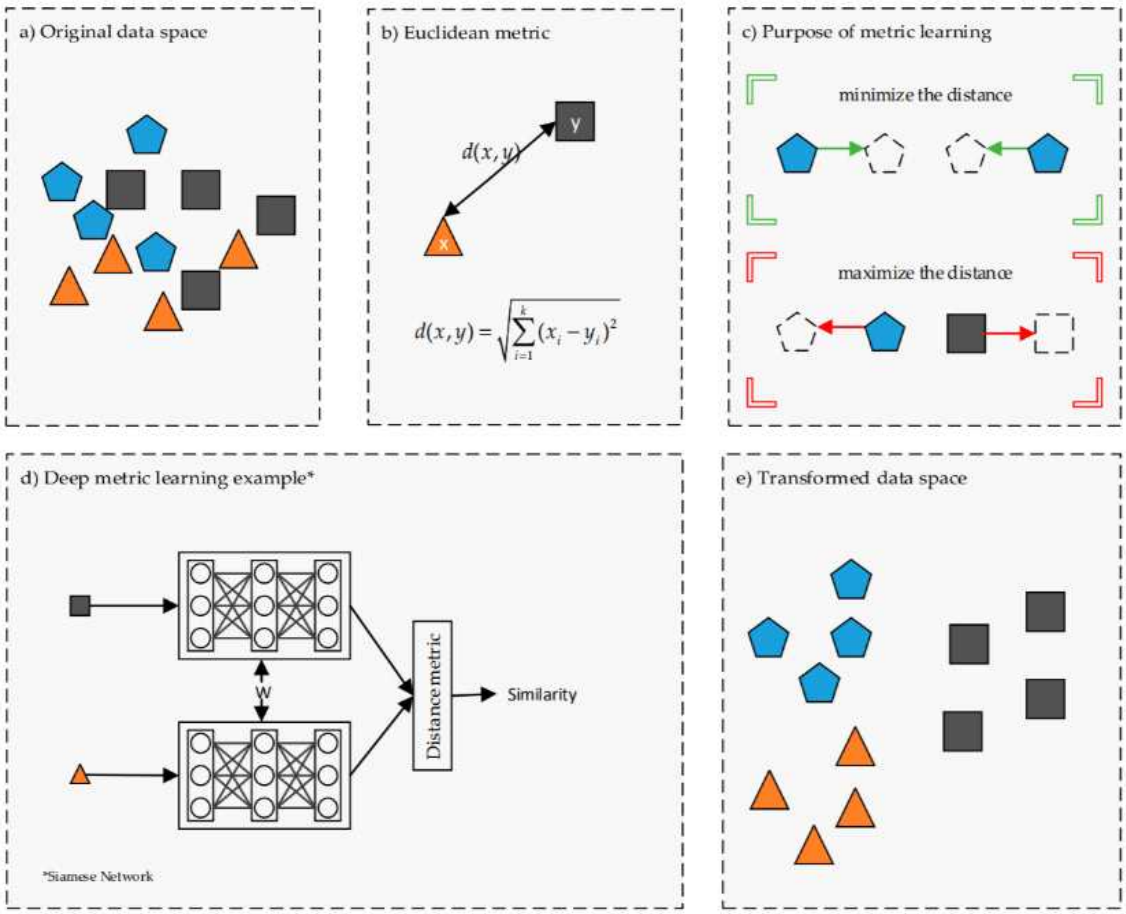


그림1. metric learning overview

(a) Original data space

- data space에 표시된 데이터들은 색깔과 모양에 따라 서로 다른 클래스를 나타낸다.

(b) Euclidean metric

- 두 데이터 포인트 x, y 사이의 거리를 유클리디안 거리 공식에 의해 계산한다.

(c) Purpose of metric learning

- 같은 클래스에 속한 데이터 포인트들의 거리를 최소화한다. (즉, 유사한 객체들이 더 가까이 위치하도록 한다.)

- 다른 클래스에 속한 데이터 포인트들의 거리를 최대화한다. (즉, 다른 객체들이 더 멀리 위치하도록 한다.)

(d) Deep metric learning example

- Siamese network가 사용되어 두 입력 데이터가 동일한 네트워크를 통과하여 특징을 추출하고, 이를 바탕으로 유사성을 학습한다.
- 두 이미지를 통과시킨 후, 각 특징벡터를 생성하고, 이 벡터들 사이의 거리를 측정하여 유사도를 평가한다.

(e) Transformed data space

- Metric learning 후, 데이터 포인트들이 새로운 특징 공간으로 변환되어 표시된다.
- 같은 클래스의 데이터 포인트들이 서로 가까워졌고(유사성 증가), 다른 클래스의 데이터 포인트들은 서로 멀어졌다(유사성 감소).
- 결과적으로, 학습 후 transformed data space에서는 original data space보다 클래스 간의 경계가 더 명확해진다.

이처럼, Metric learning은 데이터의 특징을 학습하는 기존의 방식보다는 weight sharing을 통해 학습하는 방법을 학습한다고 볼 수 있다. 즉, 데이터 포인트들 사이의 거리를 조정하여, 비슷한 데이터들은 더 가까이, 다른 것들은 더 멀리 배치함으로써 few shot learning에서 더 나은 성능을 달성하기 위한 기법 중 하나이다.

2.2. Loss function

Metric learning에서 다양한 손실 함수가 중요한 역할을 한다. Contrastive loss, Triplet loss, Center loss, Arcface 등 다양한 손실 함수가 연구되었다. 이들은 유사한 데이터 포인트들을 가까이, 다른 데이터 포인트들을 멀리 배치하는 방식에 초점을 맞춘다. 예를 들어, Contrastive loss는 같은 클래스의 데이터 포인트들을 가깝게, 다른 클래스의 데이터 포인트들을 멀게 두도록 한다. Triplet loss는 Anchor, Positive, Negative라는 세 가지 데이터 포인트를 사용해 유사한 데이터 포인트들이 더 가까이, 불일치하는 데이터 포인트들이 더 멀게 배치되도록 한다. 학습할 때, 분류를 하는 것이 아니라, 같다/다르다를 나타내는 distance를 계산하는 metric을 사용하는 방식을 따르는 것이다.

2.3. 적용 사례/방법

Metric learning은 주로 얼굴 인식(Face Recognition)과 이미지 검색(Image Retrieval) 같은 분야에서 사용된다. 얼굴 인식은 일상생활에서 흔히 볼 수 있으며, 스마트폰의 Face ID 기능이 대표적인 예이다. 이 과정에서는 먼저 영상에서 얼굴을 탐지하고, 해당 얼굴을 벡터로 임베딩한 후, 이를 통해 타인과 사용자를 비교함으로써 사용자를 식별한다. 이미지 검색도 마찬가지로 쇼핑몰이나 사용자의 앨범에서 원하는 사진을 검색해주는 등 일상생활에서 널리 사용되고 있는 기능이다. 사용자가 제공한 이미지를 기반으로 유사한 상품을 찾아내는 과정을 거친다. 이를 위해 이미지의 중요한 특징을 추출하고, 이를 미리 구축된 DB와 비교하여 가장 유사한 상품을 제안한다. 이처럼 Metric learning은 다양한 실용적인 응용 분야에서 중요한 역할을 하고 있다.

3. 본인의 문제 해결 과정에 대한 핵심 내용 작성

3.1. 문제인식

: 주요 목표는 주어진 고래 꼬리 사진이 알려진 4591개의 고래 꼬리 중 하나에 속하는지, 아니면 이전에 관찰된 적이 없는 new_whale 인지 식별하는 것.

3.2. Data processing

3.2.1. Custom dataset으로 YOLOv5 fine-tunning

: bounding box 좌표값 생성을 위한 데이터셋은 kaggle 1위 참가자가 공개한 파일을 사용했다.[1] 해당 파일(Cropping.txt)에는 그림2.와 같이 고래 꼬리 지느러미의 가장자리에 있는 픽셀들을 나타내는 여러 가지 지점들로 구성되어 있다.

그림2. 고래 꼬리의 가장자리에 있는 점의 좌표

Cropping.txt에 존재하는 1200개의 이미지만을 추려내어 YOLOv5의 학습 및 검증 데이터셋으로 사용하였다. 원본 label 형식은 고래 꼬리의 가장자리에 있는 여러 지점들의 좌표값들로 나타나있다. YOLOv5에 적용하기 위해서는 이 모델에서 요구하는 데이터셋 구성에 맞게 변환해주어야한다. 먼저 이 지점들을 사용하여 bounding box를 생성해야했다. 이를 위해 각 객체의 minX, maxX, minY, maxY 값을 구해서 label 값을 그림 3과 같이 구한다.

그림3. YOLOv5 label format

학습을 완료 한 후, Inference 과정을 거친다. 고래 꼬리를 detect하고 bbox를 예측하는 모델이 저장되었고, 이 .pt파일을 이용해 실제로 detection을 진행하였다. inference 결과는 그

림4와 같다. 고래 꼬리에 대한 bbox를 대체적으로 잘 탐지하는 모습을 보인다.

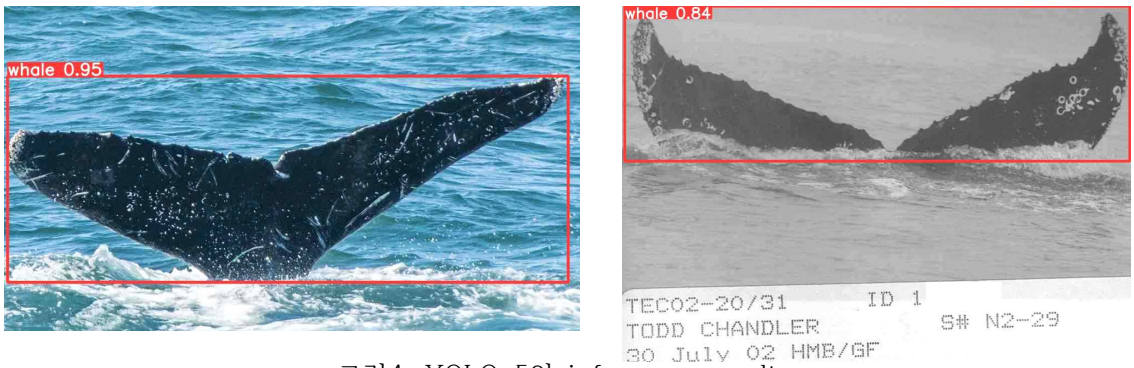


그림4. YOLOv5의 inference results

bbox를 crop하여 학습 및 추론 데이터셋으로 사용하기 위해서 추가 과정을 거쳤다. YOLOv5 모델의 inference code를 수정하여, 각 이미지에서 탐지된 객체의 바운딩 박스 부분만 크롭하여 저장하는 기능을 추가하였다. 따라서 그림 5처럼 고래꼬리에 해당하는 부분만큼을 저장하여 데이터셋을 재구성하였다.

그림5. 새롭게 재구성한 데이터셋의 예시

3.2.2. Data post-processing

YOLOv5의 inference 이미지들을 눈으로 검수하였고, 잘못 inference하여 고래 꼬리가 아닌 다른 부분을 detection한 경우가 여러개 있음을 확인하였다. 그림 6과 같은 경우에는, 원본 이미지로 대체하였다.

그림6. (좌) 원본 이미지 샘플, (우) 모델이 오검출한 inference 이미지 샘플

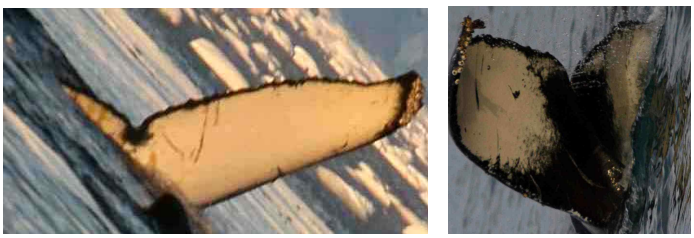
또한, 하나의 이미지에 대해 여러개를 detection한 경우에는 수작업으로 검수하여 고래 꼬리에 해당하는 이미지만 남기고 나머지 이미지는 제거하였다. 약 60개의 이미지에 대해 검수를 진행하였다.

그림7. 모델이 여러개의 물체를 detection한 이미지 샘플

3.2.3. Data Augmentation

하나의 클래스 당 데이터가 한 개만 있는 경우가 대부분이다. 그래서 데이터 증강을 보다 강하게 주기 위해서 다양한 증강 방법을 사용하였다. 실험한 모델 또는 모델의 backbone은 resnet50과 densenet121이므로, input data는 224로 resize했다. 다양한 데이터 증강 방법을 적용하기 전에, 이미 공개된 best solution에서 효과가 입증된 증강 방식을 사용하였고 또한 테스트 데이터셋을 직접 열어서 눈으로 확인한 후에 필요해보이는 방법들을 선택하여 적용하였다.

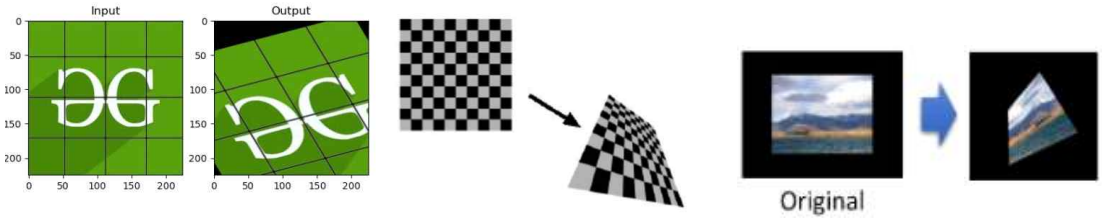
- ShiftScaleRotate: shift, scale, rotate를 한꺼번에 적용시키는 방법으로, 고래 꼬리 이미지가 약간씩 회전 되거나 확대되어 있고, 늘 같은 위치에 있지 않은 것을 확인하였기 때문에 적용하였다.



- ToGray, GuassianBlur, HueSaturationValue: 대체적으로 회색조의 이미지가 많이 눈에 띄었고, 색조 변환이 다양하게 있는 이미지를 확인하였기 때문에 적용하였다.

- Affine, PiecewiseAffine, Perspective: 공개된 best solution에서 공통적으로 사용된 기법이다. Affine 변환은 이미지를 회전, 크기 조정, 이동 및 기울임 등을 통해 변형하는데 이 변환을 통해 모델이 다양한 각도와 위치에서 고래 꼬리를 인식하는 능력을 향상시킬 것이

다. PiecewiseAffine 변환은 이미지의 일부분을 지역적으로 변형시키는데, 이 방법은 고래 꼬리의 특정 부분이 왜곡되거나 다른 방향으로 이동할 때도 해당 부위를 인식할 수 있도록 할 것이다. Perspective 변환은 이미지에 관점 변화(예: 3D 효과)를 추가하게 되는데 이 변환은 고래 꼬리가 다른 관점에서 보았을 때의 모습을 학습하는 데 도움을 준다.




```
data_transforms = A.Compose([
    A.Resize(224, 224),
    A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),

    A.ShiftScaleRotate(shift_limit=0.0625, scale_limit=0.1, rotate_limit=18, p=0.5),
    A.Affine(shear=0.4, p=0.5),

    A.ToGray(p=0.4),

    A.OneOf([
        A.GaussianBlur(blur_limit=3, p=1.0),
        A.HueSaturationValue(hue_shift_limit=5, sat_shift_limit=5, p=1.0)
    ], p=0.5),

    A.OneOf([
        A.PiecewiseAffine(scale=(0.01, 0.03), p=1.0),
        A.Perspective(p=1.0),
    ], p=0.5),

    A.HorizontalFlip(p=0.5),

    ToTensorV2(),
])
```

```
data_transforms_test = A.Compose([
    A.Resize(224, 224),
    A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ToTensorV2(),
])
```

그림8. (상) 학습 데이터셋에 적용한 데이터 증강 (하) 테스트 데이터셋에 적용한 데이터 증강

3.3. Network

실험1. resnet50+BCEWithLogitsLoss → score: 50.154

```
model_conv = pretrainedmodels.resnet50()
model_conv.last_linear = nn.Linear(model_conv.last_linear.in_features, 4251)
print(model_conv)
criterion = nn.BCEWithLogitsLoss()

optimizer = optim.Adam(model_conv.parameters(), lr=0.01)
exp_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)
```

실험2. resnet101+BCEWithLogitsLoss → score: 48.888

```
model_conv = pretrainedmodels.resnet101()
model_conv.last_linear = nn.Linear(model_conv.last_linear.in_features, 4251)
print(model_conv)
criterion = nn.BCEWithLogitsLoss()

optimizer = optim.Adam(model_conv.parameters(), lr=0.001)
exp_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=3, gamma=0.1)
```

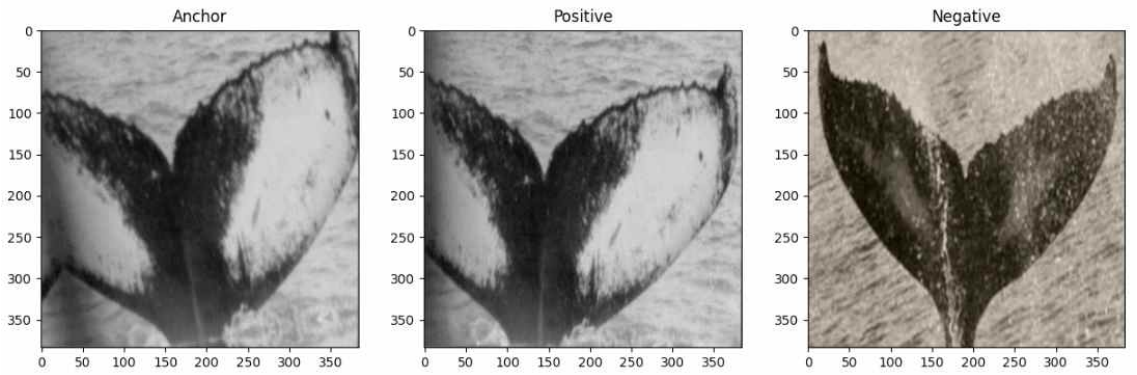
성능 향상에 도움은 되지 않았지만 시도한 방법

: 본 실험에서는 실험 1에서 사전 훈련된 ResNet-50을 backbone으로 하는 Siamese Network를 사용하여 Metric learning을 진행하였다.

실험3. Resnet50+Siamese network+TripletLoss

```
class SiameseNetworkWithPretrained(nn.Module):
    def __init__(self):
        super(SiameseNetworkWithPretrained, self).__init__()
        self.cnn = models.resnet50(pretrained=True)
        self.cnn = nn.Sequential(*list(self.cnn.children())[:-1])

        self.fc = nn.Sequential(
            nn.Linear(2048, 512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 1),
            nn.Sigmoid()
        )

    def forward(self, input1, input2):
        output1 = self.cnn(input1).flatten(1)
        output2 = self.cnn(input2).flatten(1)

        distance = torch.abs(output1 - output2)

        output = self.fc(distance)
        return output

    def get_embedding(self, x):
        embedding = self.cnn(x).flatten(1)
        return embedding
```

모델 학습 과정에서는 Triplet Loss 방식을 채택하였다. 이는 anchor, positive, negative 샘플 간의 거리를 계산하여 모델의 가중치를 업데이트한다. 이를 통해 모델이 동일한 클래스 내에서는 유사성을, 다른 클래스 간에서는 차별성을 더 잘 인식할 수 있도록 학습되었다.




```python
class TripletLoss(nn.Module):
    def __init__(self, margin=1.0):
        super(TripletLoss, self).__init__()
        self.margin = margin

    def forward(self, anchor, positive, negative):
        distance_positive = F.pairwise_distance(anchor, positive, p=2)
        distance_negative = F.pairwise_distance(anchor, negative, p=2)
        losses = torch.relu(distance_positive - distance_negative + self.margin)
        return losses.mean()
```

3.4. Training

실험1,2. CNN+BCEWithLogitsLoss

```
import torch
import tqdm
import numpy as np
import time

model_conv = model_conv.cuda()
n_epochs = 100
best_train_loss = float('inf')

for epoch in range(1, n_epochs + 1):
    print(time.ctime(), 'Epoch:', epoch)

    model_conv.train()
    train_loss = []
    for data, target in tqdm.tqdm(train_loader):
        data, target = data.cuda(), target.cuda()
        optimizer.zero_grad()
        output = model_conv(data)
        loss = criterion(output, target.float())
        train_loss.append(loss.item())
        loss.backward()
        optimizer.step()

    avg_train_loss = np.mean(train_loss)
    print(f'Epoch {epoch}, Train Loss: {avg_train_loss:.4f}')

    torch.save(model_conv.state_dict(), 'densenet121.pt')

    if avg_train_loss <= best_train_loss:
        best_train_loss = avg_train_loss
        torch.save(model_conv.state_dict(), 'densenet121_best.pt')
        print("Best model saved with train loss: {:.4f}".format(avg_train_loss))
```

실험3. SiameseNnetwork+TripletLoss

```
n_epochs = 100
for epoch in range(n_epochs):
    model.train()
    train_loss = 0.0
    progress_bar = tqdm(enumerate(train_loader), total=len(train_loader), desc=f"Epoch {epoch+1}/{n_epochs}")
    for i, batch in progress_bar:
        anchor, positive, negative = batch
        anchor, positive, negative = anchor.cuda(), positive.cuda(), negative.cuda()

        optimizer.zero_grad()

        # 모델에서 임베딩 추출
        anchor_embedding = model.get_embedding(anchor)
        positive_embedding = model.get_embedding(positive)
        negative_embedding = model.get_embedding(negative)

        # Triplet Loss 계산
        loss = criterion(anchor_embedding, positive_embedding, negative_embedding)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()

        # 진행 상태 업데이트
        progress_bar.set_postfix(loss=loss.item())

    scheduler.step()

    avg_loss = train_loss / len(train_loader)
    print(f"Epoch {epoch+1}/{n_epochs}, Average Loss: {avg_loss}")

    torch.save(model.state_dict(), 'base_siamese_triplet_4251.pt')

    if avg_loss < best_loss:
        best_loss = avg_loss
        torch.save(model.state_dict(), 'best_siamese_triplet_4251.pt')
        print("Saved best model with loss:", best_loss)
```

3.6. Inference

실험1,2. CNN+BCEWithLogitsLoss

```
sub = pd.read_csv('./whale-categorization-playground/sample_submission.csv')
model_conv.cuda()
model_conv.eval()
for (data, target, name) in test_loader:
    data = data.cuda()
    output = model_conv(data)
    output = output.cpu().detach().numpy()
    for i, (e, n) in enumerate(list(zip(output, name))):
        sub.loc[sub['Image'] == n, 'Id'] = ' '.join(le.inverse_transform(e.argsort()[-5:][::-1]))

sub.to_csv('./submission/1203_submission.csv', index=False)
```

실험3. SiameseNetwork+TripletLoss

추론을 위해 먼저, InferenceDataset 클래스를 사용해서 필요한 이미지 데이터를 준비했다. 모델 로딩 단계에서는 SiameseNetworkWithPretrained 모델을 불러와 사전 훈련된 가중치를 적용했다. 임베딩 추출은 extract_embeddings 함수로 했다. 이 함수는 훈련 및 테스트 데이터셋에서 각 이미지의 임베딩을 추출했다. 그 다음으로는 훈련 데이터 임베딩을 바탕으로 NearestNeighbors 모델을 학습시켜 테스트 데이터셋의 각 이미지에 대한 가장 가까운 이웃을 찾았다. 마지막 단계에서는 임계값을 설정해 'new_whale' 클래스를 분류하고, 각 테스트 이미지별로 가장 가까운 이웃 5개를 뽑아 결과를 만들었다. 그리고 이 결과들을 포함한 제출 파일을 CSV 형식으로 저장했다.

```
class InferenceDataset(Dataset):
    def __init__(self, datafolder, transform=None):
        self.datafolder = datafolder
        self.image_files_list = [s for s in os.listdir(datafolder)]
        self.transform = transform

    def __len__(self):
        return len(self.image_files_list)

    def __getitem__(self, idx):
        img_name = os.path.join(self.datafolder, self.image_files_list[idx])
        image = np.array(Image.open(img_name).convert('RGB'))
        if self.transform:
            image = self.transform(image=image)["image"]
        return image, self.image_files_list[idx]
```

```
train_dataset_ = InferenceDataset('./whale-categorization-playground/train', transform=data_transforms)
train_loader_ = DataLoader(train_dataset_, batch_size=32, shuffle=False)
test_dataset_ = InferenceDataset('./whale-categorization-playground/test', transform=data_transforms)
test_loader_ = DataLoader(test_dataset_, batch_size=32, shuffle=False)
```

```python
from tqdm import tqdm

model = SiameseNetworkWithPretrained()

trained_weights = torch.load('base_siamese_triplet_4251.pt')

model.load_state_dict(trained_weights)

model.cuda()

def extract_embeddings(data_loader, model):
    model.eval()
    embeddings = []
    filenames = []
    with torch.no_grad():
        for data, file in tqdm(data_loader, desc="Extracting embeddings"):
            data = data.cuda()
            embedding = model.get_embedding(data)
            embeddings.append(embedding.cpu())
            filenames.extend(file)
    return torch.cat(embeddings).numpy(), filenames

# 임베딩 추출
train_embeddings, train_filenames = extract_embeddings(train_loader_, model)
test_embeddings, test_filenames = extract_embeddings(test_loader_, model)
```

```
Extracting embeddings: 100%|████████| 308/308 [05:51<00:00,  1.14s/it]
Extracting embeddings: 100%|████████| 488/488 [09:17<00:00,  1.14s/it]
```

```python
neigh = NearestNeighbors(n_neighbors=6)
neigh.fit(train_embeddings)

distances_test, neighbors_test = neigh.kneighbors(test_embeddings)

preds_str = []

threshold = 0.1

for i, test_filename in enumerate(test_filenames):
    sample_result = []
    sample_classes = []

    for d, n in zip(distances_test[i], neighbors_test[i]):
        train_filename = train_filenames[n]
        sample_classes.append(train_filename)
        sample_result.append((train_filename, d))

    if "new_whale" not in sample_classes and distances_test[i][0] > threshold:
        sample_result.append(("new_whale", threshold))

    sample_result.sort(key=lambda x: x[1])
    sample_result = sample_result[:5]
    preds_str.append(" ".join([x[0] for x in sample_result]))

sub = pd.read_csv('./whale-categorization-playground/sample_submission.csv')
sub['Id'] = preds_str
sub.to_csv('./submission/1204_submission_siamese_triplet_epoch44.csv', index=False)
```

3.8. ensemble

마지막으로 성능이 가장 높게 나온 실험1,2의 모델 두 개로 앙상블을 진행해서 최종 스코어를 기록할 수 있었다.

```
import pandas as pd
import torch
import numpy as np

# Load models
model_resnet50 = pretrainedmodels.resnet50()
model_resnet50.last_linear = nn.Linear(model_resnet50.last_linear.in_features, 4251)
model_resnet50.load_state_dict(torch.load('best_model.pt'))
model_resnet50.cuda()
model_resnet50.eval()

model_resnet101 = pretrainedmodels.resnet101()
model_resnet101.last_linear = nn.Linear(model_resnet101.last_linear.in_features, 4251)
model_resnet101.load_state_dict(torch.load('best_resnet101.pt'))
model_resnet101.cuda()
model_resnet101.eval()

# Inference
sub = pd.read_csv('./whale-categorization-playground/sample_submission.csv')

for (data, target, name) in test_loader:
    data = data.cuda()

    # Resnet50 predictions
    output_resnet50 = model_resnet50(data)
    output_resnet50 = output_resnet50.cpu().detach().numpy()

    # Resnet101 predictions
    output_resnet101 = model_resnet101(data)
    output_resnet101 = output_resnet101.cpu().detach().numpy()

    # Ensemble: Average predictions
    output_ensemble = (output_resnet50 + output_resnet101) / 2

    # Update submission file
    for i, (e, n) in enumerate(list(zip(output_ensemble, name))):
        sub.loc[sub['Image'] == n, 'Id'] = ' '.join(le.inverse_transform(e.argsort()[-5:][::-1]))

# Save ensemble submission
sub.to_csv('./submission/ensemble_resnet50_resnet101.csv', index=False)
```

3.7. Reference

[1]https://www.kaggle.com/datasets/martinpiotte/humpback-whale-identification-fluke-location

4. 리더보드 순위/점수 스크린샷

리더보드 순위: 5위

순위	학번	점수
1	20196513	0.58534
2	20195248	0.62427
3	20195198	0.56287
4	20195223	0.54954
5	M23048	0.50883

점수: 50.883

ensemble_resnet50_resnet101.csv
Complete (after deadline) · now

0.50883 **0.50883**