Humpback Whale Identification Challenge

영상처리와딥러닝 기말 프로젝트 솔루션

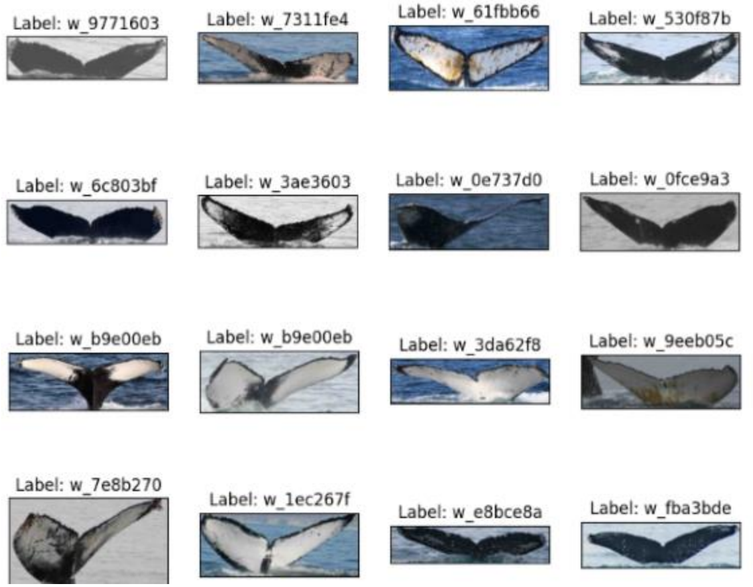
빅데이터 20195248 정유빈

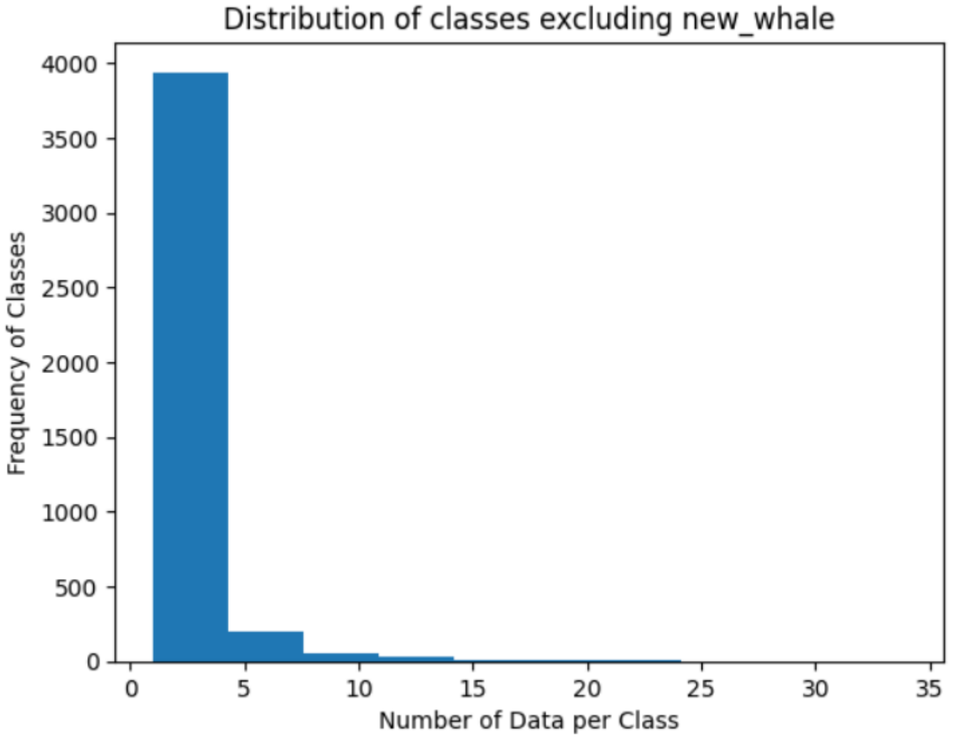
Contents

Dataset

- 혹등고래의 꼬리 이미지와 이미지에 해당하는 고래 ID 존재
- 전체 4251개의 클래스 중 약 95%는 데이터가 5장 미만

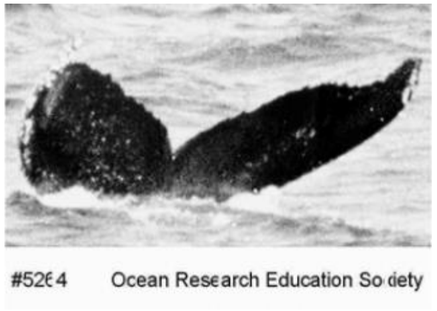
  -> 각 클래스의 대표성을 적절히 학습하기 어려움

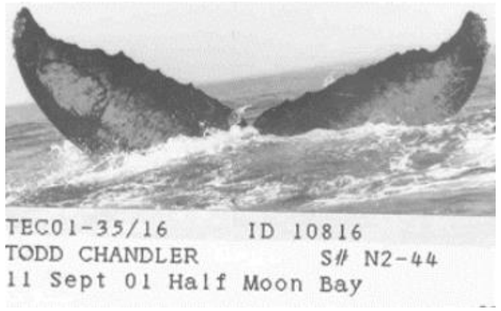


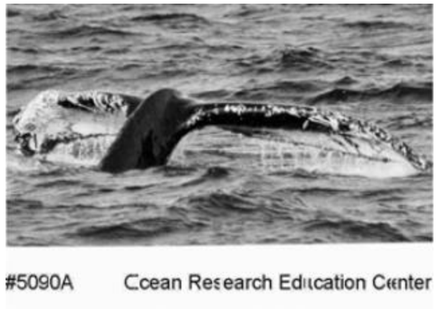


Dataset

- Train 데이터 중 학습에 영향을 줄 수 있는 이미지 존재







글씨가 포함되어 있는 이미지

고래의 꼬리보다 배경 픽셀의 비중이 더 높은 이미지

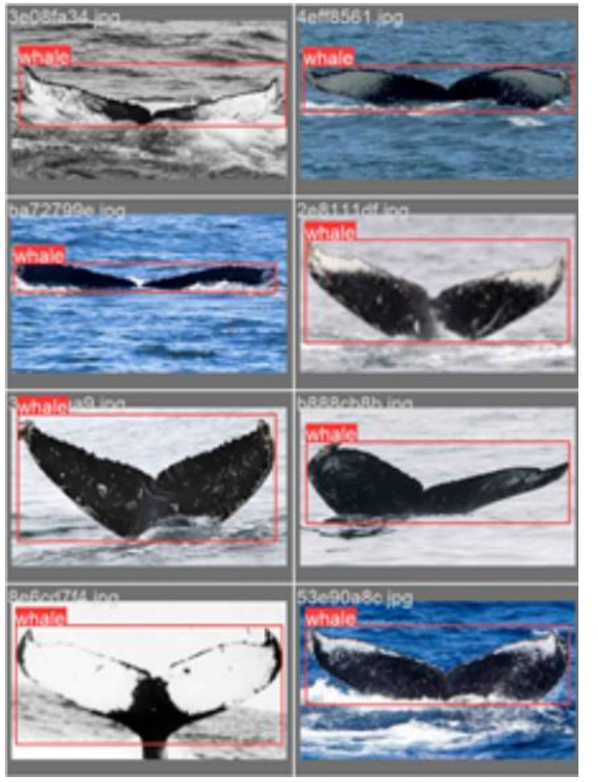
Dataset - Object detection

- 고래의 꼬리 부분만 crop 하기 위해 YOLOv5 모델 fine-tuning

Bounding box (535개)

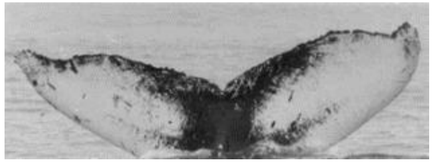
fn	bbox
00ad9219.jpg	9 10 276 1044
3b339885.jpg	27 13 459 1034
00ded600.jpg	15 14 483 1029
06e622e0.jpg	13 10 261 880
3bf857ac.jpg	13 2 420 1044
0a00c8c5.jpg	26 15 410 1018
6ad9c4b8.jpg	10 16 413 1049
3c711425.jpg	105 23 381 1022
6ae55fe5.jpg	137 20 371 1035
0aed1250.jpg	73 21 503 901
3d1f59ae.jpg	145 14 383 1025
6b43b833.jpg	15 5 308 1038

Pre-trained YOLO



Dataset - Object detection

- Train과 Test 데이터 모두 crop한 후 사용하였더니 성능이 향상되었음



submission.csv Complete (after deadline) · 7d ago	0.31331	0.31331
crop.csv Complete (after deadline) · 4d ago	0.3592	0.3592

Dataset - Data Augmentation

- 실험에 공통적으로 사용한 Data Augmentation

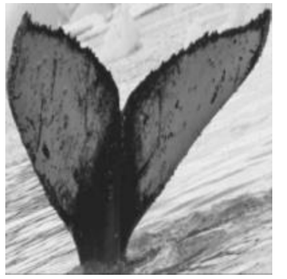
Original

HorizontalFLlip

CLAHE

ColorJitter

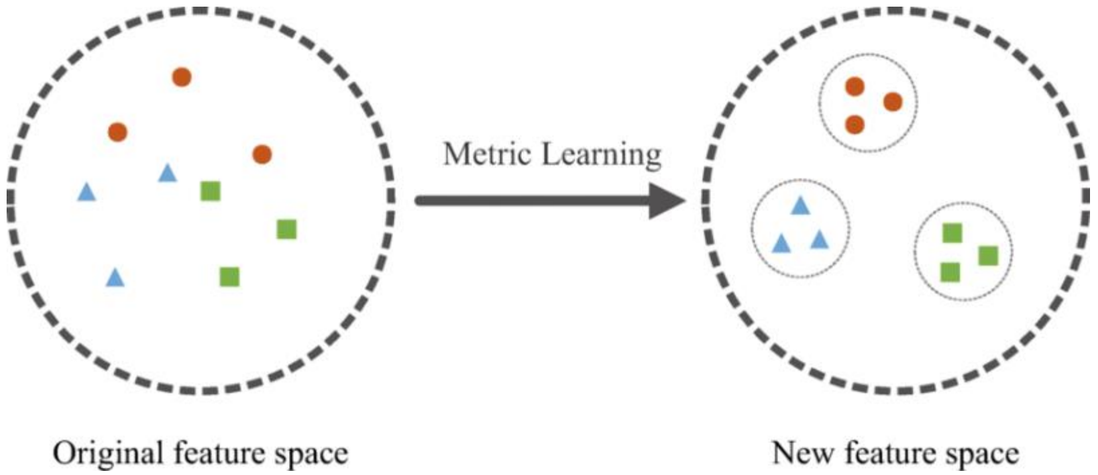
Rotate



ToGray

Metric Learning

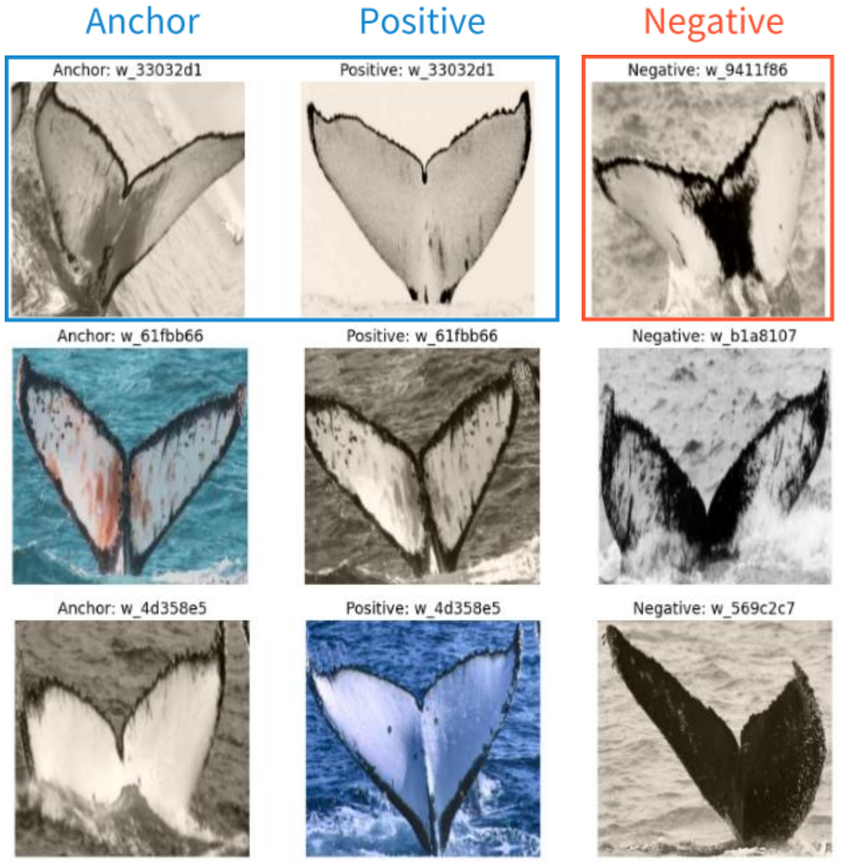
- Metric Learning
  - 거리 또는 유사성 기반 학습을 통해 클래스 간의 차이점에 집중하여 특징 추출
  - 새로운 데이터가 주어질 때 기존에 학습된 공간에 효과적으로 매핑 가능



Metric Learning

- Siamese Network + Triplet Loss + K-NN

```python
class TripletGenerator:
    def __init__(self, file_class_mapping):
        self.file_class_mapping = file_class_mapping
        self.class_to_files = defaultdict(list)

        for file, class_ in file_class_mapping.items():
            self.class_to_files[class_].append(file)

        self.classes = list(self.class_to_files.keys())

    def get_triplet(self):
        pos_class = random.choice(self.classes)
        pos_files = self.class_to_files[pos_class]

        # Anchor, Positive 선택
        if len(pos_files) > 1:
            anchor, positive = random.sample(pos_files, 2)
        else:
            anchor = positive = pos_files[0]

        # Negative 선택
        neg_class = random.choice([cls for cls in self.classes if cls != pos_class])
        negative = random.choice(self.class_to_files[neg_class])

        return anchor, positive, negative
```



Metric Learning

- Siamese Network + Triplet Loss + K-NN

Backbone - ResNet50
Output Embedding Size - 512

```python
class SiameseNetwork(nn.Module):
    def __init__(self, embedding_size=512):
        super(SiameseNetwork, self).__init__()
        self.backbone = models.resnet50(pretrained=True)
        in_features = self.backbone.fc.in_features
        self.backbone.fc = nn.Linear(in_features, embedding_size)

    def forward(self, x):
        return self.backbone(x)

model = SiameseNetwork().to(device)
triplet_loss = nn.TripletMarginLoss(margin=1.0, p=2)
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

학습된 Siamese 모델에서 train과 test embedding 각각 추출
Train 데이터로 KNN 학습 후 test를 넣어 각 데이터에 가까운 클래스 찾음

```python
from sklearn.neighbors import NearestNeighbors

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

distances_test, neighbors_test = neigh.kneighbors(test_embeddings)
```

CNN Architecture1 - ResNet

- 사전 학습된 ResNet50 모델 사용

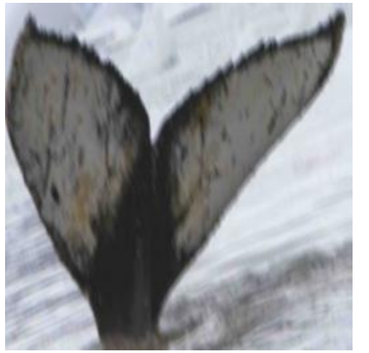
```
model_conv = pretrainedmodels.resnet50()
model_conv.last_linear = nn.Linear(model_conv.last_linear.in_features, 4251)
criterion = nn.CrossEntropyLoss()
n_epochs = 28
optimizer = optim.Adam(model_conv.parameters(), lr=0.001)
exp_lr_scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=n_epochs, eta_min=0)
```

CrossEntropyLoss 사용
Train epoch: 28
Optimizer: Adam
Scheduler: CosineAnnealingLR

CNN Architecture1 - ResNet

- Data augmentation 기법 추가

```
data_transforms = A.Compose([
                            A.Resize(256, 256),
                            A.HorizontalFlip(p=0.5),
                            A.OneOf([
                                A.CLAHE(clip_limit=0.2, tile_grid_size=(3, 3), p=0.7),
                                A.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2, p=0.5)
                            ], p=0.5),
                            A.Rotate(limit=15, p=0.5),
                            A.OneOf([
                                A.MotionBlur(blur_limit=7, allow_shifted=True, always_apply=False, p=0.5),
                                A.Downscale(scale_min=0.4, scale_max=0.8, interpolation=2, p=0.5)
                            ], p=0.3),
                            A.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
                            A.ToGray(p=0.5),
                            ToTensorV2()
])

data_transforms_test = A.Compose([
                                A.Resize(256, 256),
                                A.Normalize([0.485 , 0.456, 0.406], [0.229, 0.224 , 0.225]),
                                ToTensorV2()
])
```



MotionBlur

Downscale

submission (7).csv Complete (after deadline) · 1d ago · resnet50		0.55436	0.55436

CNN Architecture2 - EfficientNet

- 사전 학습된 EfficientNet-b0 모델 사용

* Data Augmentation은 이전과 동일

```
model_conv = timm.create_model('efficientnet_b0', pretrained=True)
num_ftrs = model_conv.classifier.in_features
model_conv.classifier = nn.Linear(num_ftrs, 4251)
# criterion = nn.BCEWithLogitsLoss()
criterion = nn.CrossEntropyLoss()
n_epochs = 28
optimizer = optim.Adam(model_conv.parameters(), lr=0.001)
exp_lr_scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=n_epochs, eta_min=0)
```

CrossEntropyLoss 사용
Train epoch: 28
Optimizer: Adam
Scheduler: CosineAnnealingLR

Ensemble

- Soft voting 기법 적용

Inference

```
model1 = torch.load('model/model_resnet50.pth') # 0.55449
model2 = torch.load('model/model_effb0.pth') # 0.53362
model3 = torch.load('model/model_effb0_2.pth') # 0.52609
model4 = torch.load('model.pth') # 0.49614

model1.eval()
model2.eval()
model3.eval()
model4.eval()

sub = pd.read_csv('sample_submission.csv')

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

    with torch.no_grad():
        output1 = model1(data).cpu().numpy()
        output2 = model2(data).cpu().numpy()
        output3 = model3(data).cpu().numpy()
        output4 = model4(data).cpu().numpy()

    final_output = (output1 + output2 + output3 + output4) / 4

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

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

모델1) ResNet50 - 0.55436

모델2) EfficientNet-b0 - 0.53723

모델3) EfficientNet-b0 - 0.52606

모델4) EfficientNet-b0 - 0.49614

모델의 예측 결과 평균

감사합니다.