저해상도 조류 이미지 분류

Contents

Chapter 01

주제

01 데이터 설명

1) 64 X 64의 저해상도 이미지 분류

입력으로 들어오는 64X64 크기의 **저해상도 조류 이미지로부터 종을 분류하는 알고리즘 개발**
학습 데이터는 64X64 크기의 15834개의 저해상도 조류 이미지
평가 데이터는 64X64 크기의 6786개의 저해상도 조류 이미지
학습데이터와 1:1로 쌍으로 구성된 256X256 고해상도 조류 이미지

Train(64X64)

Upscale(256X256)

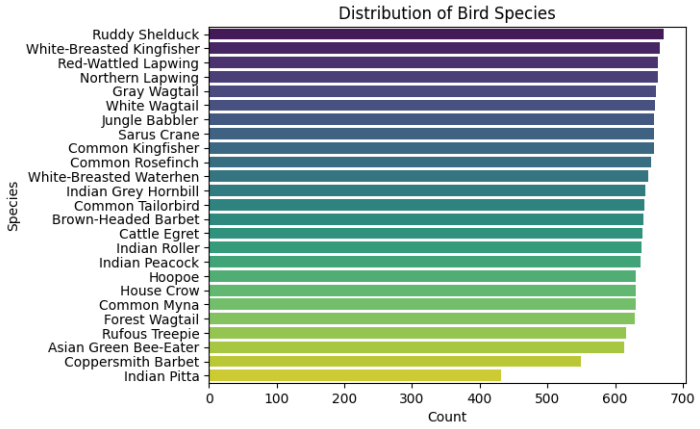
Test(64X64)

01 데이터 설명

2) EDA

총 25개의 범주를 가지는 데이터
마지막 2개의 데이터를 제외한 대부분 균일한 데이터
NULL 값이 없음

시각화



value

label	
Ruddy Shelduck	671
White-Breasted Kingfisher	666
Red-Wattled Lapwing	663
Northern Lapwing	663
Gray Wagtail	661
White Wagtail	659
Jungle Babbler	658
Sarus Crane	657
Common Kingfisher	657
Common Rosefinch	653
White-Breasted Waterhen	649
Indian Grey Hornbill	645
Common Tailorbird	643
Brown-Headed Barbet	642
Cattle Egret	641
Indian Roller	639
Indian Peacock	637
Hoopoe	631
House Crow	630
Common Myna	630
Forest Wagtail	629
Rufous Treepie	616
Asian Green Bee-Eater	613
Coppersmith Barbet	550
Indian Pitta	431

Data.info

```
df = pd.read_csv('train.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15834 entries, 0 to 15833
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   img_path          15834 non-null  object
 1   upscale_img_path  15834 non-null  object
 2   label             15834 non-null  object
dtypes: object(3)
memory usage: 371.2+ KB
```

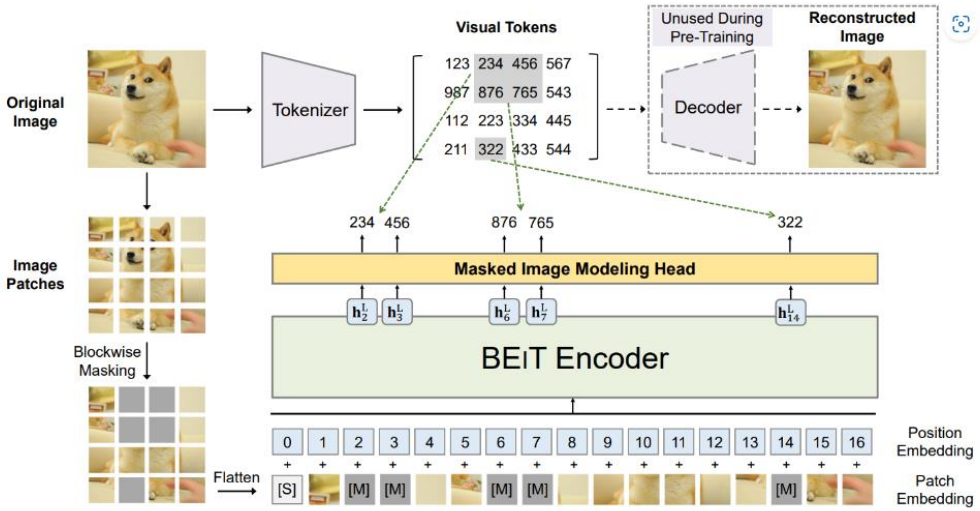
Chapter 02

분석 모델

02 분석 모델

1) BEiT 모델 설명

BERT Pre-Training of Image Transformer
이미지 데이터를 위한 Transform 기반 모델



저해상도 이미지분류에서
타 모델 대비 뛰어난 성능

- BERT의 마스크들 언어 모델링 기법을 이미지에 적용
- 이미지를 작은 패치로 분할하고, 일부 패치를 마스킹 하여 예측하는 방법

02 분석 모델

1) BEiT 학습설정

BERT Pre-Training of Image Transformer
이미지 데이터를 위한 Transform 기반 모델

- 모델 : microsoft/beit-base-patch16-224-pt22k
  (마이크로소프트에서 제공하는 BEiT 모델)
- 옵티마이저 : AdamW
- Adam옵티마이저 변형으로, 과대 적합 방지
- lr : 5e-5
- 에폭 : 5
- 손실함수: CrossEntropyLoss
  분류문제에서 자주 사용되는 손실함수, 모델의 예측과 실제 라벨 분포 간의 차이를 측정
- 데이터셋 : CustomDataset

01 분석 모델

1) BEiT 64X64 이미지의 데이터셋

Public점수 : 0.8414710386
Private점수 : 0.8508147919

```
# 데이터셋 준비
train_dataset = CustomDataset(train_df, feature_extractor, mode='train')
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True) # 배치사이즈16, 데이터 무작위로 섞어서 로드

test_dataset = CustomDataset(test_df, feature_extractor, mode='test')
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False) # train은 성능향상과 균등한 학습을 위해 셔플 트루
                                                                     # 테스트는 디버깅 편의성을 위해(샘플에대한 모델을 쉽게 추적하기위해)
```

```
# 옵티마이저
optimizer = AdamW(model.parameters(), lr=5e-5)
criterion = CrossEntropyLoss()

# 학습루프
epochs = 5
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    for batch in tqdm(train_loader):
        inputs, labels = batch
        inputs = inputs.to('cuda')
        labels = labels.to('cuda')

        optimizer.zero_grad()
        outputs = model(pixel_values=inputs)
        loss = criterion(outputs.logits, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch {epoch+1}, Loss: {running_loss / len(train_loader)}")
```

```
# 평가, 예측
model.eval()
predictions = []
ids = []

with torch.no_grad():
    for batch in tqdm(test_loader):
        inputs, id_batch = batch
        inputs = inputs.to('cuda')

        outputs = model(pixel_values=inputs)
        _, preds = torch.max(outputs.logits, dim=1)

        predictions.extend(preds.cpu().numpy())
        ids.extend(id_batch)
```

02 중간결론

1) 방법 제시

화질 224인 이유: BEiT 모델의 표준 입력 크기인 224X224를 맞춰주기 위함
사전 훈련된 가중치를 효과적으로 활용 할 것으로 기대됨

BEST 예상

256 -> 64

- Upscale(256X256)데이터를 64X64로 변환 후 훈련

64 -> 256

- Train(64X64)데이터를 256X256으로 변환 후 훈련

64 -> 224

- Train(64X64)데이터를 224X224로 변환 후 훈련
- Test(64X64) 데이터에도 224X224로 변환 후 결과 예측

Chapter 03

화질 별 결과값

03 화질 별 결과값

1) 256 -> 64

Public 점수 : 0.8414710386
Private 점수 : 0.8508147919

Public 점수 : 0.7332637447
Public 점수 : 0.7200080295

```python
class CustomDataset(Dataset):
    def __init__(self, dataframe, feature_extractor, mode='train', use_upscale=False):
        self.dataframe = dataframe
        self.feature_extractor = feature_extractor
        self.mode = mode
        self.use_upscale = use_upscale
        self.transform = transforms.Resize((64, 64))  # 64x64 크기로 리사이즈

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

    def __getitem__(self, idx):
        if self.mode == 'train':
            img_path = self.dataframe.iloc[idx]['upscale_img_path'] if self.use_upscale else self.dataframe.iloc[idx]['img_path']
            label = self.dataframe.iloc[idx]['label']
        else:
            img_path = self.dataframe.iloc[idx]['img_path']
            label = -1  # Dummy label for test mode

        image = Image.open(img_path).convert("RGB")
        image = self.transform(image)  # 64x64 크기로 리사이즈
        inputs = self.feature_extractor(images=image, return_tensors="pt")

        if self.mode == 'train':
            return inputs['pixel_values'].squeeze(0), torch.tensor(label, dtype=torch.long)
        else:
            return inputs['pixel_values'].squeeze(0), self.dataframe.iloc[idx]['id']
```

03 화질 별 결과값

2) 64 -> 256

Public 점수 : 0.8414710386
Private 점수 : 0.8508147919

Public 점수 : 0.7533914477
Public 점수 : 0.7468831411

```python
class CustomDataset(Dataset):
    def __init__(self, dataframe, feature_extractor, mode='train', use_upscale=False):
        self.dataframe = dataframe
        self.feature_extractor = feature_extractor
        self.mode = mode
        self.use_upscale = use_upscale
        self.transform = transforms.Resize((256, 256))  # 256x256 크기로 리사이즈

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

    def __getitem__(self, idx):
        if self.mode == 'train':
            img_path = self.dataframe.iloc[idx]['upscale_img_path'] if self.use_upscale else self.dataframe.iloc[idx]['img_path']
            label = self.dataframe.iloc[idx]['label']
        else:
            img_path = self.dataframe.iloc[idx]['img_path']
            label = -1  # Dummy label for test mode

        image = Image.open(img_path).convert("RGB")
        image = self.transform(image)  # 256x256 크기로 리사이즈
        inputs = self.feature_extractor(images=image, return_tensors="pt")

        if self.mode == 'train':
            return inputs['pixel_values'].squeeze(0), torch.tensor(label, dtype=torch.long)
        else:
            return inputs['pixel_values'].squeeze(0), self.dataframe.iloc[idx]['id']
```

03 화질 별 결과값

2) 64 -> 224

Public 점수 : 0.8414710386
Private 점수 : 0.8508147919

Public 점수 : 0.8894463257
Public 점수 : 0.8861093583

```python
class CustomDataset(Dataset):
    def __init__(self, dataframe, feature_extractor, mode='train', use_upscale=False):
        self.dataframe = dataframe
        self.feature_extractor = feature_extractor
        self.mode = mode
        self.use_upscale = use_upscale
        self.transform = transforms.Resize((224, 224))  # 224x224 크기로 리사이즈

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

    def __getitem__(self, idx):
        if self.mode == 'train':
            img_path = self.dataframe.iloc[idx]['upscale_img_path'] if self.use_upscale else self.dataframe.iloc[idx]['img_path']
            label = self.dataframe.iloc[idx]['label']
        else:
            img_path = self.dataframe.iloc[idx]['img_path']
            label = -1  # Dummy label for test mode

        image = Image.open(img_path).convert("RGB")
        image = self.transform(image)  # 224x224 크기로 리사이즈
        inputs = self.feature_extractor(images=image, return_tensors="pt")

        if self.mode == 'train':
            return inputs['pixel_values'].squeeze(0), torch.tensor(label, dtype=torch.long)
        else:
            return inputs['pixel_values'].squeeze(0), self.dataframe.iloc[idx]['id']
```

```python
# Test 데이터셋도 동일하게 224x224 리사이즈 모델 적용
test_dataset = CustomDataset(test_df, feature_extractor, mode='test')
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)
```

Chapter 04

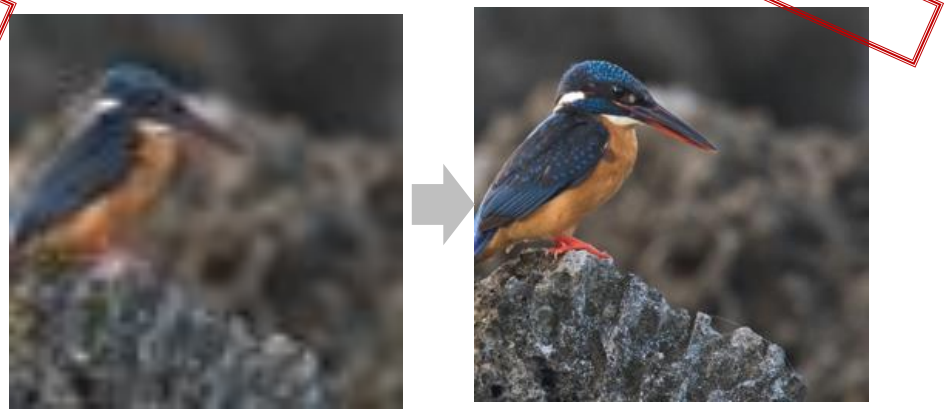
결론

04 결론

예상대로 BEiT모델 요구사항인 224X224 크기로 맞췄을 때 가장 점수가 잘 나왔습니다.
64->256, 256->64로 리사이즈 하는 방법은 리사이즈 하지 않는 것보다 오히려 성능이 내려갔습니다.

256 ->64 WORST

64 ->224 BEST



고해상도로 변환하거나 낮추는 과정에서 이미지 정보 손실이 발생했기 때문에 결과가 낮아진 걸로 보입니다. 또한, BEiT 모델의 사전 훈련 가중치가 224X224 크기에 최적화 되어 있기 때문에, 이 크기로 입력 이미지를 맞추는 것이 중요한 것으로 확인되었습니다.

감사합니다

자세한 코드는 https://github.com/seup178/bird 에 있습니다