7조는 못말려

짱구는 못말려

떡잎마을 방범대 분류 모델

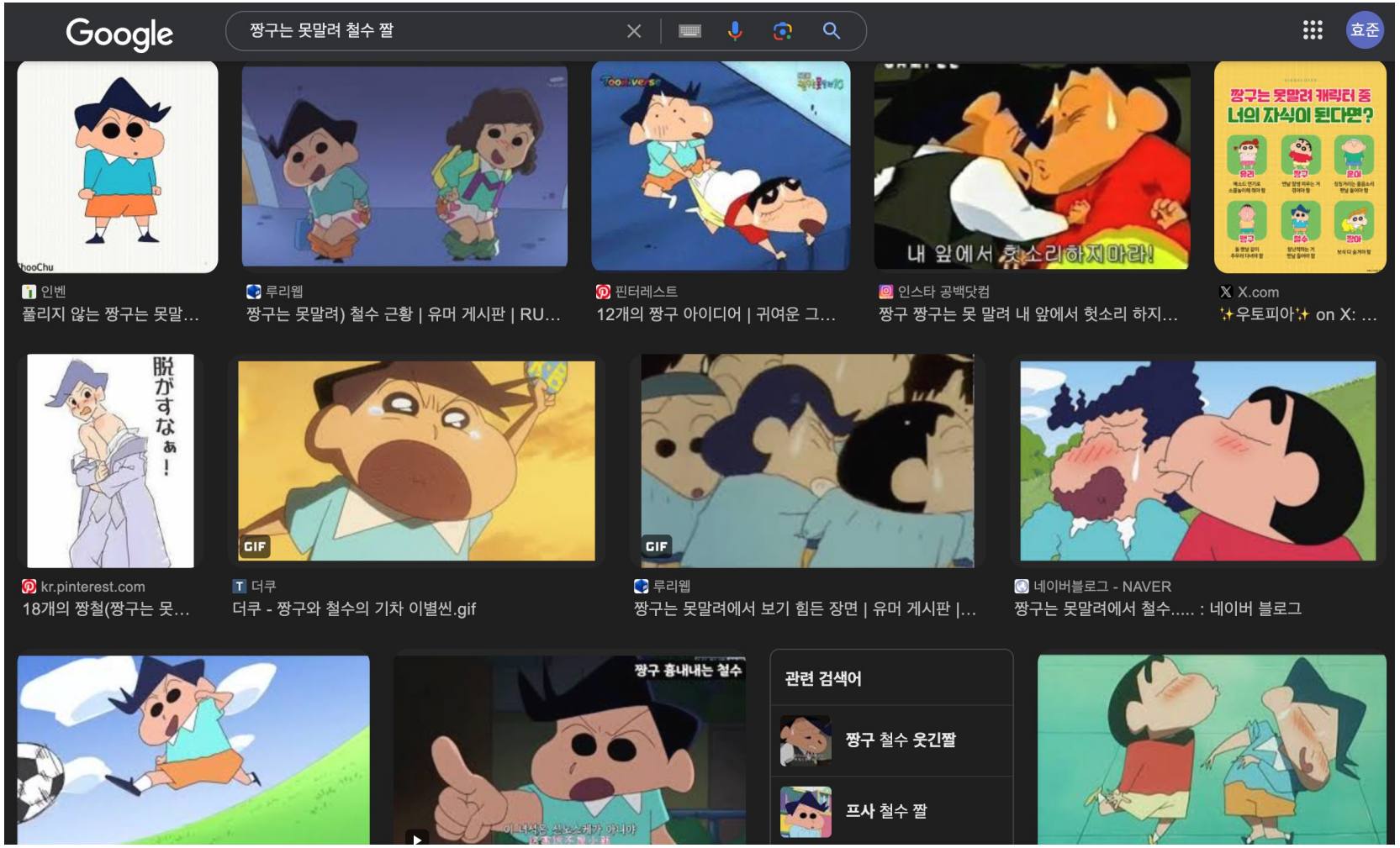
발표 순서

1. 안효준 (ResNet50)
2. 안윤호 (VGG19_BN)
3. 권도운 (convnext_base)
4. 김태헌 (efficientnet b7)

ResNet50

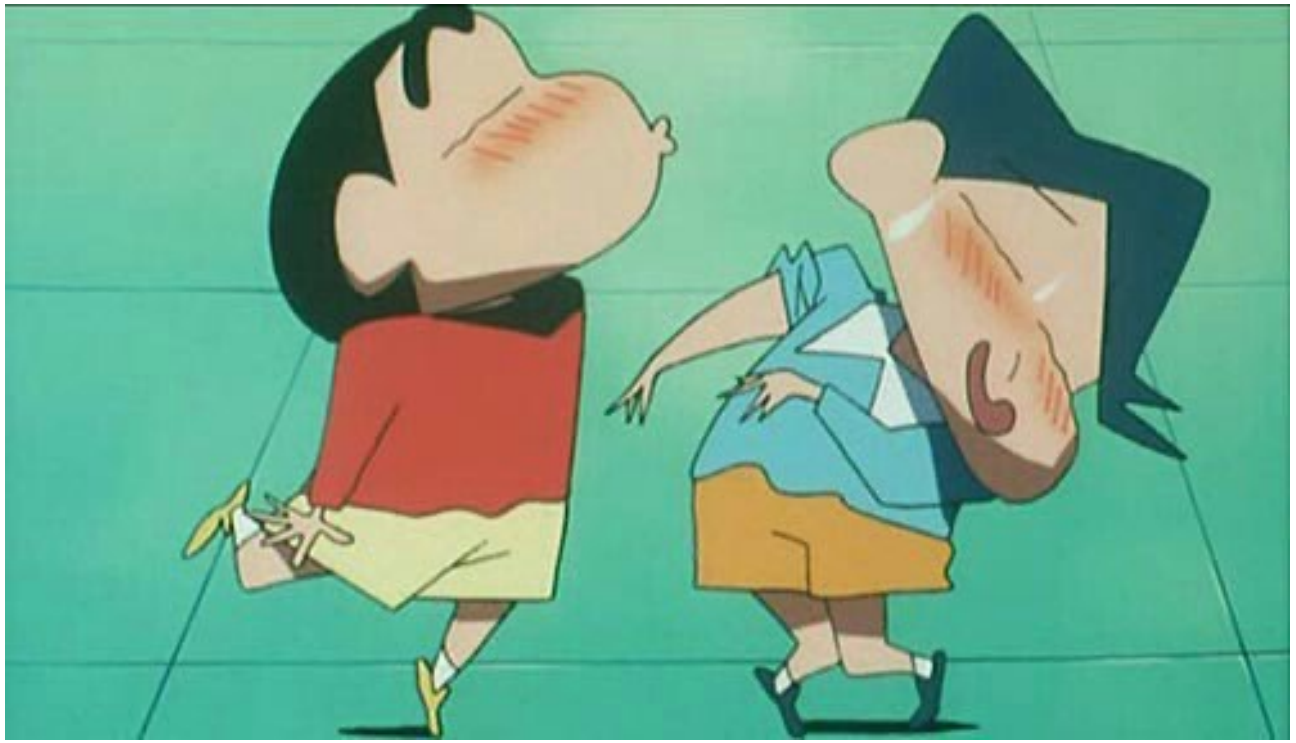
안효준

1 데이터 출처



구글에서 직접 수집

2 이미지 전처리



사진을 잘라내어 이미지 전처리 진행

3 원본 데이터 확인

```
# 원본 이미지 경로
ROOT_PATH = '/Users/anhyojun/WorkSpace/LocalData/방범대 원본'

# ImageFolder 사용
originDS = datasets.ImageFolder(root=ROOT_PATH)

print(originDS.classes)
print(f'데이터 개수 : {len(originDS)}')
```

✓ 0.0s

```
['맹구', '유리', '짱구', '훈이', '철수']
데이터 개수 : 3233
```

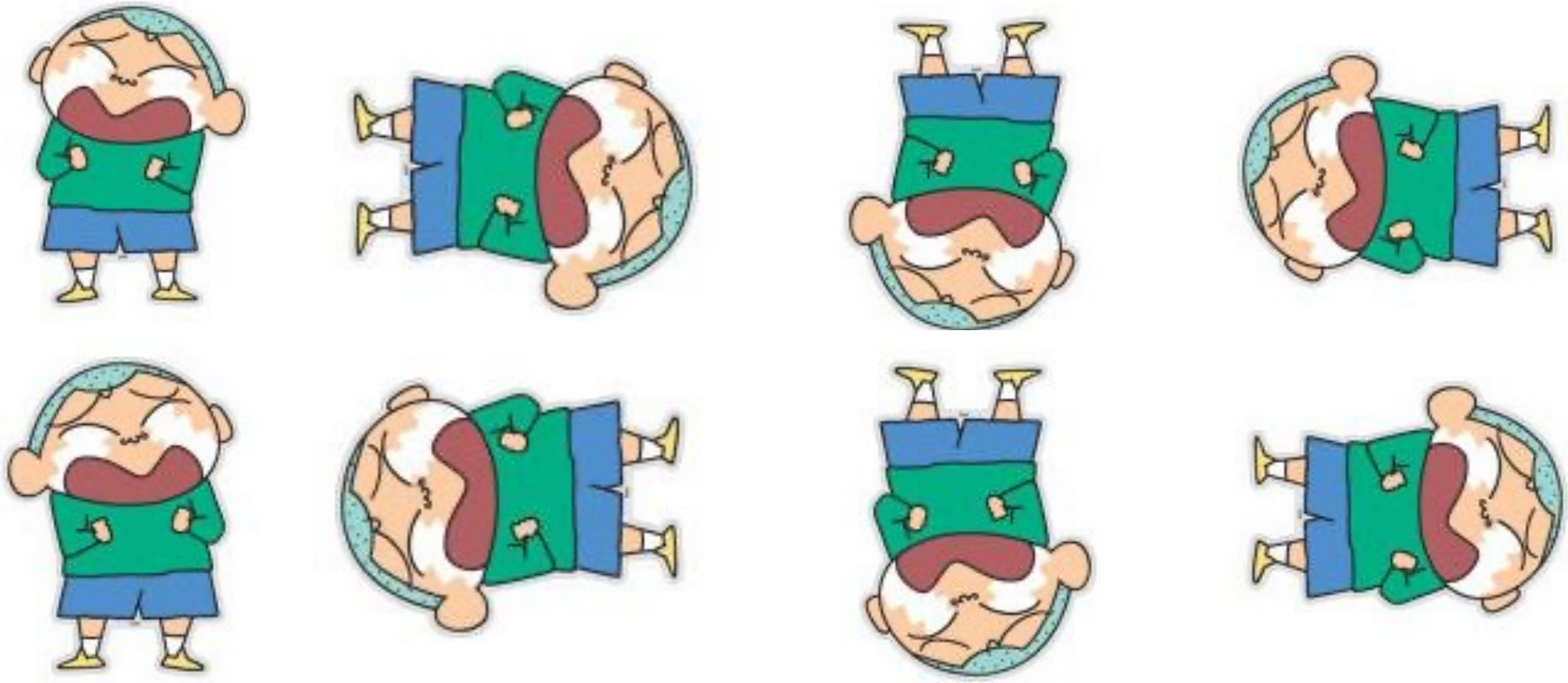
3233장의 원본 방범대 이미지 수집

3 이미지 증강

```python
# 이미지 회전해서 증강하기

# 원본 이미지 경로
ROOT_PATH = '/Users/anhyojun/WorkSpace/LocalData/방범대'
# 회전된 이미지 경로
SAVE_PATH_90 = '/Users/anhyojun/WorkSpace/LocalData/방범대_회전'

# ImageFolder 사용
originDS = datasets.ImageFolder(root=ROOT_PATH)

for i in range(len(originDS)):
    for j in [0, 90, 180, 270]:
        class_name = originDS.classes[DS[i][1]]
        FILE_PATH = os.path.join(SAVE_PATH_90, class_name, f'img_index{i}_{j}.jpg')
        FILE_PATH2 = os.path.join(SAVE_PATH_90, class_name, f'flip_img_index{i}_{j}.jpg')
        rotate_img = originDS[i][0].rotate(j, expand=True)
        flipped_img = rotate_img.transpose(Image.FLIP_LEFT_RIGHT)
        rotate_img.save(FILE_PATH)
        flipped_img.save(FILE_PATH2)
```

90, 180, 270도 회전 및 좌우 반전을 통해 이미지 증강

3 이미지 증강



1장의 사진을 8장으로 증강

4 증강 후 데이터 확인

```python
# 이미지 전처리(transform) 정의
transform = transforms.Compose([
    transforms.Resize((224, 224)),  # 이미지 크기 조정
    transforms.ToTensor(),          # 텐서로 변환
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # ImageNet 정규화 값
])

# 이미지 경로
ROOT_PATH = '/Users/anhyojun/WorkSpace/LocalData/방범대'

# ImageFolder 사용
DS = datasets.ImageFolder(root=ROOT_PATH, transform=transform)
# DataLoader
DL = DataLoader(DS, batch_size = 32, shuffle=True)

print(DS.classes)
print(f'데이터 개수 : {len(DL.dataset)}')

file_path = [i[0] for i in DS.imgs]
labels = [i[1] for i in DS.imgs]
```

✓ 0.1s

```
['맹구', '유리', '짱구', '철수', '훈이']
데이터 개수 : 25864
```

3233장에서 25864장으로 증가

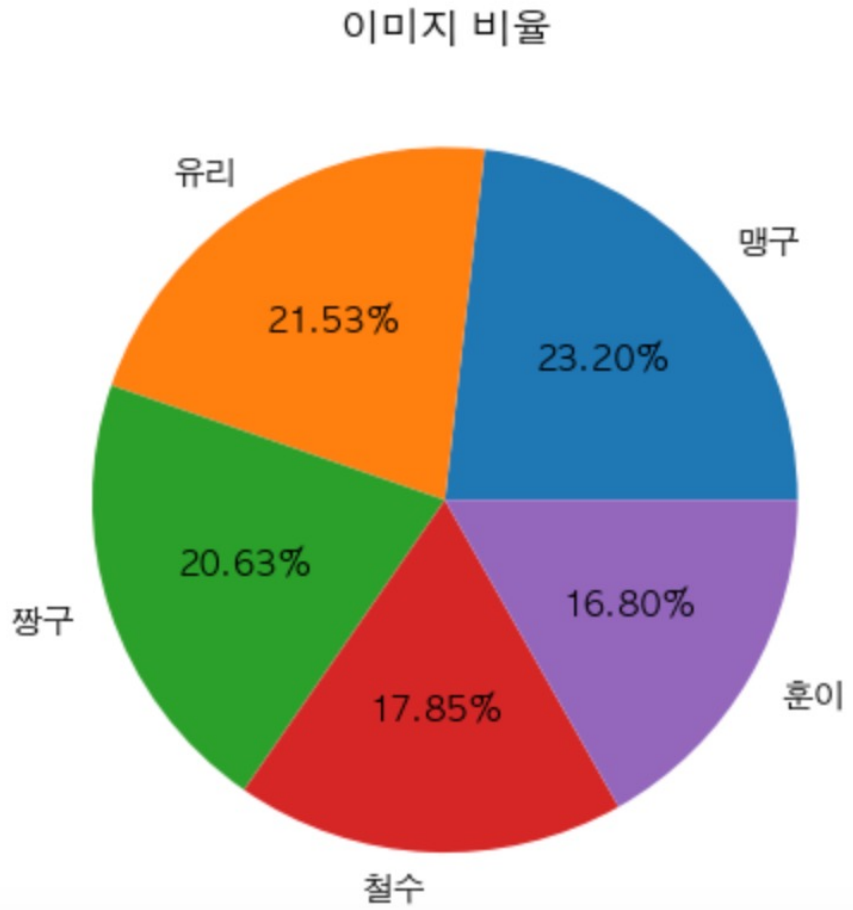
3 이미지 비율 시각화

```
labels = [DS[i][1] for i in range(len(DS))]
```

✓ 6m 41.3s

```
plt.pie(pd.DataFrame(labels).value_counts(), labels=DS.classes,
        autopct='%.2f%%')
plt.title('이미지 비율')
plt.show()
```

✓ 0.0s



비교적 균형 잡힌 비율로 이미지가 분포됨

3 훈련, 검증, 테스트 데이터 분리

```
file_path = [i[0] for i in DS.imgs]
labels = [i[1] for i in DS.imgs]
```

✓ 0.0s

```
train_path, test_path, train_labels, test_labels = train_test_split(file_path, labels, stratify=labels,
                                                    random_state = 42, train_size = 0.8)
train_path, valid_path, train_labels, valid_labels = train_test_split(train_path, train_labels,
                                                     stratify=train_labels,
                                                     random_state = 42, train_size = 0.8)
```

✓ 0.0s

파일 경로와 레이블을 분리

3 훈련, 검증, 테스트 데이터 분리

```python
class ImagePathDataset(Dataset):
    def __init__(self, file_paths, labels, transform=None):
        super().__init__()
        self.file_paths = file_paths
        self.labels = labels
        self.transform = transform

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

    def __getitem__(self, idx):
        img_path = self.file_paths[idx]
        image = Image.open(img_path).convert("RGB")

        if self.transform != None:
            image = self.transform(image)

        # 레이블 불러오기
        label = self.labels[idx]

        return image, label
```

✓ 0.0s

경로를 통해 데이터셋을 만드는 클래스 생성

```python
trainDS = ImagePathDataset(train_path, train_labels, transform=transform)
validDS = ImagePathDataset(valid_path, valid_labels, transform=transform)
testDS = ImagePathDataset(test_path, test_labels, transform=transform)

trainDL = DataLoader(trainDS, batch_size = 16)
validDL = DataLoader(validDS, batch_size = 16)
testDL = DataLoader(testDS, batch_size = 16)
```

✓ 0.0s

훈련, 검증, 테스트 데이터로더 생성

3 이미지 확인

```
def draw_shin_chan(INDEX):
    name_dict = {i : DS.classes[i] for i in range(len(DS.classes))}
    plt.rc('font', family='AppleGothic')
    a = trainDL.dataset[INDEX][0].permute(1,2,0)
    plt.axis('off')
    plt.title(name_dict[trainDL.dataset[INDEX][1]])
    plt.imshow(a)
    plt.show()

draw_shin_chan(0)
```

✓ 0.1s

색상이 정규화된 이미지를 확인 가능

4 전이학습 모델 불러오기

```
# 미리 학습된 ResNet 모델 불러오기 (weight 사용)
model = models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)

# 합성곱층 가중치 고정
for param in model.parameters():
    param.requires_grad = False

# 전결합층 가중치 변경하도록 설정
for param in model.fc.parameters():
    param.requires_grad = True
```

✓ 0.4s

전이학습 모델인 ResNet50 모델 불러오기

4 전결합층 출력값 변경

```
model
```

✓ 0.0s

```
...
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=1000, bias=True)
)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# 마지막 레이어 (full connected layer) 교체 (클래스 5개로 변경)
num_features = model.fc.in_features # 2048
model.fc = nn.Linear(num_features, 5) # 입력은 이전 층의 아웃을 그대로 받고 출력은 방법대 인원
```

✓ 0.0s

전결합층의 출력값을 방범대 인원인 5로 변경

4 훈련 진행

```
# MPS 사용 여부 확인하기
device = torch.device("mps" if torch.backends.mps.is_available() else "cpu")
print(f"device : {device}")
```

✓ 0.0s

```
device : mps
```

MAC에서 병렬 학습을 위한 MPS 사용 가능 여부 확인

```
# 훈련
EPOCH = 100
SAVE_PATH = '/Users/anhyojun/WorkSpace/KDT/김소현 강사님/프로젝트/5번째 프로젝트/model'

# 옵티마이저를 모델 파라미터에 대해 초기화
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

result = training(trainDL, validDL, model, model_type = 'multiclass', optimizer = optimizer,
                  epoch = EPOCH, endurance_cnt = 5, view_epoch = 1, num_classes = 5, SAVE_PATH = SAVE_PATH,
                  MPS = True, device = device)
```

⊗ 139m 1.4s

전이학습 모델로 훈련 진행

4 훈련 진행

```
    305 # (2) 손실 함수 계산
    306 if model_type == 'regression': # 회귀일 때

File /opt/anaconda3/envs/TORCH_CV_38/lib/python3.8/site-packages/t
   1551     return self._compiled_call_impl(*args, **kwargs)  # ty
   1552 else:
-> 1553     return self._call_impl(*args, **kwargs)

File /opt/anaconda3/envs/TORCH_CV_38/lib/python3.8/site-packages/t
   1557 # If we don't have any hooks, we want to skip the rest of
   1558 # this function, and just call forward.
   1559 if not (self._backward_hooks or self._backward_pre_hooks o
...
    453                     _pair(0), self.dilation, self.groups)
--> 454 return F.conv2d(input, weight, bias, self.stride,
    455                 self.padding, self.dilation, self.groups)

RuntimeError: MPS backend out of memory (MPS allocated: 1.12 GB, o
```

```
[EPOCH] : 4에서 모델 저장 완료.
[Loss : 4/100] Train : 0.2394, Test : 0.2657
[Score  : 4/100] Train : 0.9181, Test : 0.9051
[EPOCH] : 5에서 모델 저장 완료.
[Loss : 5/100] Train : 0.2205, Test : 0.2549
[Score  : 5/100] Train : 0.9258, Test : 0.9091
[EPOCH] : 6에서 모델 저장 완료.
[Loss : 6/100] Train : 0.2049, Test : 0.2461
[Score  : 6/100] Train : 0.9303, Test : 0.9111
[EPOCH] : 7에서 모델 저장 완료.
[Loss : 7/100] Train : 0.1917, Test : 0.2386
[Score  : 7/100] Train : 0.9360, Test : 0.9142
[EPOCH] : 8에서 모델 저장 완료.
[Loss : 8/100] Train : 0.1802, Test : 0.2321
[Score  : 8/100] Train : 0.9408, Test : 0.9173
...
[EPOCH] : 41에서 모델 저장 완료.
[Loss : 41/100] Train : 0.0546, Test : 0.1730
[Score  : 41/100] Train : 0.9887, Test : 0.9361
```

Output is truncated. View as a scrollable element or open in a

메모리 부족 오류가 떴지만 충분한 학습이 진행돼 41번 에포크 모델 사용

4 Best Model 저장 및 예측

```python
best_model = models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)

best_model.fc = nn.Linear(best_model.fc.in_features, 5)
best_model.load_state_dict(torch.load('best_model_epoch_41.pth', weights_only=True))
```

✓ 0.9s

```
<All keys matched successfully>
```

```python
accuracy_list = []
for input, target in testDL:
    best_model.eval()
    pred = best_model(input)
    pred_value = torch.argmax(pred, dim = 1)
    accuracy = sum(pred_value == target) / len(target)
    accuracy_list.append(accuracy)
```

✓ 25m 50.7s

```python
total_accuracy = sum(accuracy_list) / len(accuracy_list)
print(f"[test accuracy] : {total_accuracy.item():.4f}")
```

✓ 0.0s

```
[test accuracy] : 0.9419
```

테스트 정확도 : 94.19%

4 테스트 데이터 평가 지표

```python
import torch
from sklearn.metrics import classification_report

# pred_value_list와 target_list를 하나의 리스트로 변환
preds = torch.cat(pred_value_list).cpu().numpy()
targets = torch.cat(target_list).cpu().numpy()

# classification_report 출력
report = classification_report(targets, preds)
print(report)
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.98	0.92	0.95	1114
1	0.96	0.95	0.96	1067
2	0.95	0.89	0.92	923
3	0.96	0.95	0.95	869
4	0.88	0.99	0.93	1200
accuracy			0.94	5173
macro avg	0.95	0.94	0.94	5173
weighted avg	0.94	0.94	0.94	5173

만족스러운 결과

4 웹 페이지 시연

4 결론 및 소감

- ResNet 전이학습 모델의 강력함을 알 수 있었다
- 직접 이미지를 수집한 뒤 모델을 만들어 성취감이 높았다
- MPS 병렬 처리 방법을 새롭게 알아냈다