```
import torch
import torchvision.models as models
import torchvision.transforms as transforms
from PIL import Image
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
import numpy as np
import os
from google.colab import drive
```

```
# Google Drive 마운트 및 CSV 파일 로드
from google.colab import drive
drive.mount('/content/drive')
csv_path = '/content/drive/MyDrive/final/survey_style_preferences.csv'
df = pd.read_csv(csv_path)
```

```
Mounted at /content/drive
```

```
# ResNet-18 모델 로드 (중간 레이어 feature vector 추출용)
class FeatureExtractor(torch.nn.Module):
    def __init__(self, model):
        super(FeatureExtractor, self).__init__()
        self.features = torch.nn.Sequential(*list(model.children())[:-1])  # avgpool까지만 사용

    def forward(self, x):
        return self.features(x).view(x.size(0), -1)  # Flatten features

# 모델 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
resnet18 = models.resnet18(pretrained=False)  # 사전학습 없이
model = FeatureExtractor(resnet18).to(device)
model.eval()
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and ma
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights
  warnings.warn(msg)
FeatureExtractor(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (5): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (6): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
```

```
# 이미지 전처리 정의
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Validation 이미지 폴더 내 모든 이미지 feature vector 추출
validation_folder = '/content/drive/MyDrive/final/validation_image'
validation_features = {}

for img_name in os.listdir(validation_folder):
    img_path = os.path.join(validation_folder, img_name)
    image = Image.open(img_path).convert('RGB')
    image = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        feature = model(image).cpu().numpy()
    validation_features[img_name] = feature  # 이미지 이름을 key로 저장
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-4-b5329a4d7a58> in <cell line: 12>()
     12 for img_name in os.listdir(validation_folder):
     13     img_path = os.path.join(validation_folder, img_name)
---> 14     image = Image.open(img_path).convert('RGB')
     15     image = transform(image).unsqueeze(0).to(device)
     16     with torch.no_grad():

                          1 frames

/usr/local/lib/python3.10/dist-packages/PIL/Image.py in copy(self)
   1272         """
   1273         self.load()
-> 1274         return self._new(self.im.copy())
   1275 
   1276     __copy__ = copy

KeyboardInterrupt: 
```

```
# 유사도를 기반으로 Validation 이미지 선호 여부 예측
results = []
threshold = 0.8  # 임계값 설정 (예: 코사인 유사도 0.8 이상이면 선호로 예측)

for _, row in df.iterrows():
    user_id = row['응답자 ID']

    # 응답자의 선호/비선호 이미지 feature 벡터 준비
    liked_vectors = []
    disliked_vectors = []

    if pd.notna(row['Validation 스타일 선호']):
        liked_images = row['Validation 스타일 선호'].split(" | ")
        liked_vectors = [validation_features[img] for img in liked_images if img in validation_features]

    if pd.notna(row['Validation 스타일 비선호']):
        disliked_images = row['Validation 스타일 비선호'].split(" | ")
        disliked_vectors = [validation_features[img] for img in disliked_images if img in validation_features]

    # 모든 Validation 이미지에 대해 응답자의 선호 여부 예측
    for val_img_name, val_feature in validation_features.items():
        val_feature = np.array(val_feature).reshape(1, -1)

        # 선호 이미지와의 유사도 계산
        liked_similarities = [cosine_similarity(val_feature, liked_vector)[0][0] for liked_vector in liked_vectors]
        disliked_similarities = [cosine_similarity(val_feature, disliked_vector)[0][0] for disliked_vector in disliked_vectors]

        # 유사도 평균을 사용하여 예측 결정
        avg_liked_similarity = np.mean(liked_similarities) if liked_similarities else 0
        avg_disliked_similarity = np.mean(disliked_similarities) if disliked_similarities else 0

        # 임계값 비교를 통해 선호 여부 예측
        prediction = '선호' if avg_liked_similarity >= threshold and avg_liked_similarity > avg_disliked_similarity else '비선호'

        # 결과 저장
        results.append({
            '응답자 ID': user_id,
            'Validation 이미지': val_img_name,
            '예측 선호 여부': prediction,
            '선호 유사도 평균': avg_liked_similarity,
            '비선호 유사도 평균': avg_disliked_similarity
        })
```

```
results
```

```
# 결과를 DataFrame으로 변환하여 확인
results_df = pd.DataFrame(results)
print(results_df)
```

```
# 결과를 CSV 파일로 저장
output_path = '/content/drive/MyDrive/final/3-2 prediction_results.csv'
results_df.to_csv(output_path, index=False, encoding='utf-8-sig')

print(f"Prediction results saved to {output_path}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-e6f853770614> in <cell line: 3>()
      1 # 결과를 CSV 파일로 저장
      2 output_path = '/content/drive/MyDrive/Colab Notebooks/지은/3-2 prediction_results.csv'
----> 3 results_df.to_csv(output_path, index=False, encoding='utf-8-sig')
      4 
      5 print(f"Prediction results saved to {output_path}")

NameError: name 'results_df' is not defined
```

선호도 데이터를 바탕으로 예측 정확도 측정

```
import pandas as pd

preference_predict = pd.read_csv('/content/drive/MyDrive/final/3-2 prediction_results.csv')
preference_true = pd.read_csv('/content/drive/MyDrive/final/스타일선호도.csv')
```

```
num_accurate_predict = 0 # 정확하게 예측한 data 수

for _, row in preference_predict.iterrows():
    user_id = row['응답자 ID']
    val_img = row['Validation 이미지']
    print(user_id)
    if row['예측 선호 여부'] == '선호':
        if val_img in preference_true[preference_true['응답자ID'] == user_id]['Validation 스타일 선호'].values:
            num_accurate_predict += 1
    else:
        if val_img in preference_true[preference_true['응답자ID'] == user_id]['Validation 스타일 비선호'].values:
            num_accurate_predict += 1

accuracy = num_accurate_predict / len(preference_predict) * 100
print(f"Accuracy: {accuracy}%")
```

```
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
```

```
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
368
```