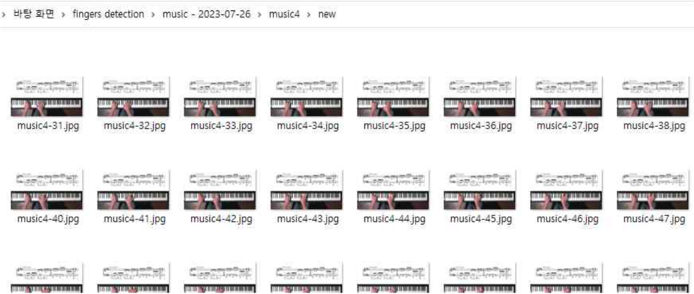
ResNet-50 Based Hand Pose Estimation Implementation

고영민

2023/09/14

Use Dataset



Input data

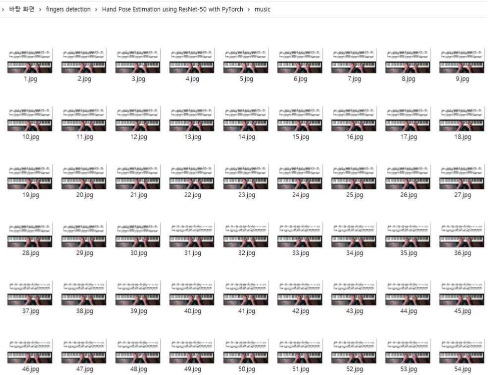
Target data

2023년 7월 26일 작업했던 music4의 이미지와 mediapipe로 추출한 스켈레톤 데이터(21(랜드마크 수) *2(양 손) *3(x, y, z)

1. Preprocessing

```
music_folder = 'music'
file_list = os.listdir(music_folder)

# 이미지 파일만 필터링 (확장자가 .jpg 인 파일)
image_files = [f for f in file_list if f.endswith('.jpg')]

# 파일 이름을 정렬하여 순서대로 처리
image_files.sort()

# 숫자를 1부터 시작하여 새로운 이름으로 변경
for i, image_file in enumerate(image_files, start=1):
    old_path = os.path.join(music_folder, image_file)

    # 현재 파일 이름에서 "output_music4-" 다음에 오는 숫자를 추출
    match = re.search(r'music4-(\d+)', image_file)
    if match:
        number = int(match.group(1))

        # 숫자를 1부터 시작하는 값으로 변경하여 새로운 이름 생성
        new_number = i
        new_file_name = f"{new_number}.jpg"

        # 새로운 파일 경로 생성
        new_path = os.path.join(music_folder, new_file_name)

        # 파일 이름 변경
        os.rename(old_path, new_path)
        print(f"Renamed {old_path} to {new_path}")

print("File renaming complete.")
```

```
File renaming complete.
```



이미지 번호를 1.jpg부터 순서대로 파일명 저장

1. Preprocessing

```python
class CustomDataset(Dataset):
    def __init__(self, csv_path, img_dir, transform=None):
        self.data = pd.read_csv(csv_path)
        self.img_dir = img_dir
        self.transform = transform

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

    def __getitem__(self, idx):
        # 이미지 파일 이름 설정 (1부터 시작하므로 idx + 1)
        img_name = os.path.join(self.img_dir, str(idx + 1) + ".jpg")
        image = Image.open(img_name)

        # 해당 행의 라벨 데이터 (x, y, z 좌표)를 추출
        labels = torch.tensor(self.data.iloc[idx].values, dtype=torch.float32)

        if self.transform:
            image = self.transform(image)

        return image, labels
```

```python
# 이미지 전처리를 원하는대로 수정
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])
])

# 데이터셋 및 DataLoader 설정
dataset = CustomDataset(csv_path="hand_result.csv", img_dir="music/", transform=transform)

batch_size=64
# 데이터를 6:2:2 비율로 분할
total_dataset_size = len(dataset)
train_size = int(0.6 * total_dataset_size)
val_size = int(0.2 * total_dataset_size)
test_size = total_dataset_size - train_size - val_size
train_dataset, val_dataset, test_dataset = random_split(dataset, [train_size, val_size, test_size])

# DataLoader 생성
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_dataloader = DataLoader(val_dataset, batch_size=batch_size)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size)
```

이미지 정규화 후 데이터를 6:2:2로 분할, 총 데이터셋 3,300개(훈련: 1979, 검증: 659개, 테스트: 661개)

1. Load Sample

```python
import matplotlib.pyplot as plt

# Load Samples
num_samples_to_display = 3
samples = []
for i in range(num_samples_to_display):
    sample_idx = i  # index
    image, labels = test_dataset[sample_idx]
    samples.append((image, labels))

# Visualization
for i, (image, labels) in enumerate(samples):
    print(f"Sample {i + 1}:")
    print("Image shape:", image.shape)
    print("Labels shape:", labels.shape)
    print("Labels:", labels)

    plt.figure(figsize=(4, 4))
    plt.imshow(image.permute(1, 2, 0))  # 이미지 형식 변경 (C, H, W) -> (H, W, C)
    plt.title(f"Sample {i + 1}")
    plt.axis("off")
    plt.show()
```

```
Sample 1:
Image shape: torch.Size([3, 224, 224])
Labels shape: torch.Size([126])
Labels: tensor([ 9.1984e+02,  5.9711e+02,  1.6400e-07,  8.8533e+02,  5.7429e+02,
        -7.7183e-03,  8.6390e+02,  5.4255e+02, -1.2022e-02,  8.4643e+02,
         5.2003e+02, -1.5319e-02,  8.2769e+02,  5.0678e+02, -1.9018e-02,
         9.1352e+02,  5.1495e+02, -1.0912e-02,  9.2157e+02,  4.7970e+02,
        -1.5198e-02,  9.2582e+02,  4.5679e+02, -1.8796e-02,  9.2878e+02,
         4.3962e+02, -2.1853e-02,  9.3928e+02,  5.1859e+02, -1.1645e-02,
         9.5643e+02,  4.8243e+02, -1.3955e-02,  9.6571e+02,  4.5922e+02,
        -1.6956e-02,  9.7340e+02,  4.4082e+02, -2.0138e-02,  9.5720e+02,
         5.2808e+02, -1.3024e-02,  9.7511e+02,  4.9545e+02, -1.7607e-02,
         9.8367e+02,  4.7523e+02, -2.2643e-02,  9.9001e+02,  4.5795e+02,
        -2.6975e-02,  9.6750e+02,  5.4179e+02, -1.4780e-02,  9.8946e+02,
         5.2804e+02, -2.1489e-02,  1.0025e+03,  5.1779e+02, -2.5356e-02,
         1.0136e+03,  5.0811e+02, -2.8099e-02,  5.1558e+02,  6.2041e+02,
         1.0800e-07,  5.5283e+02,  5.9545e+02, -5.5170e-03,  5.7016e+02,
         5.5933e+02, -8.2120e-03,  5.7950e+02,  5.3246e+02, -1.0781e-02,
         5.9623e+02,  5.1987e+02, -1.3425e-02,  5.2870e+02,  5.3102e+02,
        -7.3939e-03,  5.2500e+02,  4.9924e+02, -1.1732e-02,  5.2522e+02,
         4.8196e+02, -1.4819e-02,  5.2728e+02,  4.7119e+02, -1.7236e-02,
         5.0257e+02,  5.3523e+02, -8.2490e-03,  4.8744e+02,  4.9671e+02,
        -1.2181e-02,  4.8275e+02,  4.7385e+02, -1.4720e-02,  4.8080e+02,
         4.5763e+02, -1.6644e-02,  4.8384e+02,  5.4543e+02, -9.4930e-03,
         4.6712e+02,  5.0857e+02, -1.4299e-02,  4.6153e+02,  4.8770e+02,
        -1.7652e-02,  4.5913e+02,  4.7076e+02, -1.9958e-02,  4.7250e+02,
         5.5884e+02, -1.1085e-02,  4.5429e+02,  5.3569e+02, -1.6901e-02,
         4.4375e+02,  5.2049e+02, -1.9678e-02,  4.3612e+02,  5.0771e+02,
        -2.1051e-02])
```

Sample 1

2. Define Model

```python
# 모델 클래스 정의
class HandPoseResNet(nn.Module):
    def __init__(self, num_keypoints_per_hand=21):
        super(HandPoseResNet, self).__init__()
        # ResNet-50
        self.resnet = models.resnet50(pretrained=True)

        # 마지막 fully connected layer를 변경하여 출력 크기를 조정
        num_ftrs = self.resnet.fc.in_features
        self.resnet.fc = nn.Sequential(
            nn.Linear(num_ftrs, num_keypoints_per_hand * 3 * 2)  # (왼손 + 오른손) * (x, y, z) 좌표 (21개 * 3 * 2)
        )

    def forward(self, x):
        return self.resnet(x)
```

2. Train Model

```python
# CUDA ~ Use GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 모델을 GPU로 이동
model = HandPoseResNet(num_keypoints_per_hand=21).to(device)

# 손실 함수, 옵티마이저 설정
criterion = nn.L1Loss()  # L1 손실 함수 사용 (절대 오차)
optimizer = optim.Adam(model.parameters(), lr=0.001)

# 에포크 수 설정
num_epochs = 50

train_loss_for_plot = []
val_loss_for_plot = []

for epoch in range(num_epochs):
    # 모델을 훈련 모드로 설정
    model.train()

    # 에포크마다의 훈련 손실과 검증 손실을 저장할 리스트 초기화
    train_losses = []

    # 훈련 데이터로 모델 훈련
    train_loss = 0.0
    for inputs, labels in train_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_losses.append(loss)

    # 모델을 평가 모드로 설정
    model.eval()

    # 검증 데이터로 모델 평가
    val_losses = []

    with torch.no_grad():
        for inputs, labels in val_dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            val_loss = criterion(outputs, labels)
            val_losses.append(val_loss.item())

    # 에포크마다 결과 출력
    print(f"Epoch [{epoch+1}/{num_epochs}]")
```

```
Validation Loss: 36.8715
Epoch [45/50]
Train Loss: 20.3140
Validation Loss: 35.3376
Epoch [46/50]
Train Loss: 17.8384
Validation Loss: 34.3818
Epoch [47/50]
Train Loss: 21.0139
Validation Loss: 35.9937
Epoch [48/50]
Train Loss: 21.6885
Validation Loss: 35.7727
Epoch [49/50]
Train Loss: 18.9538
Validation Loss: 33.4900
Epoch [50/50]
Train Loss: 19.7852
Validation Loss: 37.8639
```

3. Evaluate Model

```python
from sklearn.metrics import mean_squared_error

# 모델을 평가 모드로 설정
model.eval()

# 데이터 로더 설정 (테스트)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size)

# 평균 제곱 오차를 저장할 빈 리스트 생성
test_losses = []

# 예측된 라벨과 실제 라벨을 저장할 리스트 생성
predicted_labels = []
true_labels = []

with torch.no_grad():
    for inputs, labels in test_dataloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        test_loss = criterion(outputs, labels)
        test_losses.append(test_loss.item())

        predicted_labels.extend(outputs.cpu().numpy())
        true_labels.extend(labels.cpu().numpy())

# 테스트 데이터에 대한 평균 제곱 오차 계산
test_mse = np.mean(test_losses)

# 평균 제곱 오차 출력
print(f"Test Mean Squared Error (MSE): {test_mse:.4f}")

# 예측된 라벨과 실제 라벨을 사용하여 추가 평가 지표 계산 가능
# 예를 들어, R-squared (R^2) 등을 계산하여 모델의 성능을 평가할 수 있습니다.
```

```
Test Mean Squared Error (MSE): 36.0714
```

```python
print(f'Test Dataset 갯수: {len(predicted_labels)}')
print(f'예측 레이블 수: {len(predicted_labels[0])}')
print(f'0번째 idx를 가진 testdata에 대한 랜드마크 예측 값 {predicted_labels[0]}')
```

```
Test Dataset 갯수: 661
예측 레이블 수: 126
0번째 idx를 가진 testdata에 대한 랜드마크 예측 값 [ 9.1180957e+02  6.1231854e+02  3.6083707e-01  8.7073328e+02
  5.8921234e+02 -1.5973382e-01  8.5039575e+02  5.5487164e+02
 -1.2278560e-01  8.3511572e+02  5.2663855e+02 -2.0860358e-01
  8.1704077e+02  5.0843600e+02  3.1547573e-01  8.9398486e+02
  5.2423474e+02  5.3042614e-01  9.0011761e+02  4.8880756e+02
  1.8894817e-01  9.0124109e+02  4.6765829e+02 -6.4361721e-01
  9.0246301e+02  4.5171539e+02  3.5169938e-01  9.2419940e+02
  5.2919958e+02  7.8206278e-02  9.3705951e+02  4.9203815e+02
 -1.8599395e-01  9.4202954e+02  4.6846997e+02  2.1809462e-01
  9.4426764e+02  4.4988141e+02  5.5878079e-01  9.4624774e+02
  5.3798853e+02 -2.4960598e-02  9.6033246e+02  5.0385513e+02
 -2.3267421e-01  9.6346619e+02  4.8121011e+02  3.3546770e-01
  9.6533923e+02  4.6290649e+02 -3.3491766e-01  9.6210706e+02
  5.5090448e+02 -2.8438857e-01  9.7853278e+02  5.3031555e+02
 -1.3326716e-01  9.8787354e+02  5.1591919e+02 -6.0354573e-01
  9.9167053e+02  5.0478091e+02 -3.8426813e-01  4.7322986e+02
  6.2963159e+02  4.5969984e-03  5.1553613e+02  6.0594574e+02
 -4.3430108e-01  5.3725476e+02  5.7361267e+02 -1.1382926e-01
  5.5095331e+02  5.4708478e+02 -4.2325586e-01  5.6918378e+02
  5.3182526e+02 -1.0548563e-01  4.9237387e+02  5.4170013e+02
 -1.8835047e-01  4.9365906e+02  5.0538217e+02  5.2393287e-01
  4.9886148e+02  4.9055200e+02 -3.9637703e-01  5.0557309e+02
  4.8129172e+02  2.8685525e-01  4.6273465e+02  5.4444562e+02
  1.5404665e-01  4.5386804e+02  5.0379208e+02 -6.3040853e-02
  4.5527740e+02  4.8383395e+02  6.3281703e-01  4.5817850e+02
  4.7289865e+02  5.8132567e-02  4.4062549e+02  5.5251221e+02
 -1.4651388e-01  4.2602811e+02  5.1520020e+02 -6.8872660e-02
  4.2495154e+02  4.9392242e+02 -1.2565697e-02  4.2603894e+02
  4.7747714e+02 -4.9760867e-02  4.2749600e+02  5.6695288e+02
  4.7609717e-02  4.1019940e+02  5.4343262e+02 -2.2145909e-01
  4.0131940e+02  5.2842358e+02 -2.6239222e-01  3.9696402e+02
  5.1563928e+02  2.8488675e-01]
```

Original Image

4. Test Model

```
import csv
# 평균과 표준 편차
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]

# 전처리 역변환 함수
def reverse_normalize(image):
    reverse_normalize = transforms.Compose([
        transforms.Normalize(mean=[0, 0, 0], std=[1 / std[0], 1 / std[1], 1 / std[2]]),
        transforms.Normalize(mean=[-mean[0], -mean[1], -mean[2]], std=[1, 1, 1])
    ])
    return reverse_normalize(image)

# 예측을 역 정규화하여 원래 값으로 변환
sample_idx = 10  # 확인하고 싶은 test data의 idx
original_image = reverse_normalize(test_dataset[sample_idx][0])  # 입력 이미지 역 정규화
predicted_values = predictions[sample_idx]  # 모델 예측 값

# 원본 이미지와 모델 예측 값을 출력
print(f"Original Image")
plt.imshow(original_image.permute(1, 2, 0))
plt.axis("off")
plt.show()

# 1.csv 파일 경로 설정
csv_file = '1.csv'

# 라벨 값을 1.csv 파일에 한 행에 모두 저장
with open(csv_file, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(predicted_values)

print("Model Predictions:")
print(predicted_values)
```

Original Image

현재 csv에 랜드마크값 저장하는 것까지 완료

```
Model Predictions:
[ 7,09379456e+02  6,03652893e+02 -1,69214994e-01  7,05194458e+02
  5,85007080e+02 -4,77291550e-03  7,02677185e+02  5,53661804e+02
  9,99539346e-02  7,01088806e+02  5,26306030e+02  8,98953062e-03
  6,99048950e+02  5,08830383e+02 -6,19435720e-02  7,06272949e+02
  5,21930542e+02 -9,10406262e-02  7,06255981e+02  4,86219849e+02
 -1,23236351e-01  7,05491272e+02  4,67351196e+02 -8,40492398e-02
  7,04734375e+02  4,55206635e+02 -1,01848423e-01  7,08992920e+02
  5,22713440e+02  1,97526157e-01  7,08348145e+02  4,84018768e+02
  2,86012013e-02  7,08072754e+02  4,62405701e+02 -1,70827582e-01
  7,07726013e+02  4,47646729e+02 -2,17752054e-01  7,11125732e+02
  5,29259277e+02 -1,12901837e-01  7,10013794e+02  4,93707428e+02
  4,26557511e-02  7,09340515e+02  4,73715149e+02 -1,17524944e-01
  7,08466675e+02  4,59260834e+02 -2,99888495e-02  7,13130737e+02
  5,39379578e+02 -2,22704396e-01  7,13898682e+02  5,15155151e+02
  1,42593626e-02  7,13777161e+02  5,00886841e+02 -6,18211925e-02
  7,13720337e+02  4,89944214e+02 -1,64682463e-01  6,71485291e+02
  6,10791382e+02  4,80858721e-02  6,75912537e+02  5,91391724e+02
 -1,96098629e-02  6,77612305e+02  5,59031311e+02  2,77224630e-02
  6.78404480e+02  5,30876221e+02  1,77647546e-02  6,79525452e+02
  e+02 -3,77202295e-02  6,72659851e+02  5,28725464e+02
  e-02  6,73419067e+02  4,92119354e+02  4,53800373e-02
  e+02  4,75237854e+02 -2,15347344e-03  6,74330200e+02
  e+02 -9,56692398e-02  6,69495728e+02  5,29075439e+02
  e-01  6,68300720e+02  4,88871918e+02 -9,24451128e-02
  e+02  4,68620789e+02 -1,18191205e-01  6,70248108e+02
  e+02 -5,15311211e-02  6,67077026e+02  5,35467224e+02
  e-02  6,64507874e+02  4,98549591e+02 -1,94123328e-01
  e+02  4,79224213e+02  1,31983116e-01  6,63949707e+02
  e+02 -4,46800962e-02  6,66082336e+02  5,45102234e+02
```

1.csv [C:₩Users₩User₩Desktop₩fingers detection₩Hand Pose Estimation using ResNet-50 with PyTorch₩] - 훈셀

파일 편집 보기 입력 서식 수식 데이터 도구

A1 709.37946

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	709.37946	603.6529	-0.169215	705.19446	585.0071	-0.004773	702.6772	553.6618	0.0999539	701.0888	526.306	0.0089895	699.04895	508.83038	-0.061944	706.27295	521.93054	-0.091041	706.256	486.21985	-0.123236	705.4913	467.3512	-0.084049	704.7344	455.20663
2																										