SK네트웍스 **Family AI**과정 **7**기

모델링 및 평가 수집된 데이터 및 저장

☐ 개요

- 산출물 단계 : 모델링 및 평가
- 평가 산출물 : 모델링 및 평가가 수집된 데이터 및 저장
- 제출 일자 : 25.04.04
- 깃허브 경로 :https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN07-FINAL-2Team
- 작성 팀원 : 김서진

개요	프로젝트 모델링 소개 얼굴 이미지를 가지고 5가지 얼굴형(Round, Oblong, Heart, Square, Oval)으로 자동 분류하고 가상피팅을 하는 딥러닝 모델을 구축 여기서 MTCNN은 얼굴을 정밀하게 검출하는 데 사용되었고, CNN은 검출된 얼굴 이미지를 학습하여 얼굴형을 분류하는 역할을 담당 모델링 과정 요약 ① 데이터 전처리 이미지 크기 통일, RGB 변환, MTCNN을 활용한 얼굴 영역 추출 ② 모델 설계 및 학습 • CNN 구조 설계 (Conv2D, MaxPooling, Dropout, Dense Layer 등 사용) • 모델 학습 (Epoch 수: 100, Optimizer: Adam, Loss: categorical_crossentropy) • 분류 대상: Oval, Oblong, Heart, Square, Round (총 5가지) ③ 전이학습 • VGG16 (사전 학습된 가중치 활용) • 모델학습(Epoch 수: 10, Batch Size: 64, Optimizer: Adam, Loss: categorical_crossentropy) • 분류 대상: Oval, Oblong, Heart, Square, Round (총 5가지) ④ 모델 저장 모델 저장 및 재사용 가능하도록 .keras 포맷으로 저장

모델 개발 및 시행착오	인공지능 데이터 전처리 결과 보고서 바탕으로 전처리 후 모델링 내용 ① **Mediapipe** 학습결과 • CNN 모델은 468개의 랜드마크에 대한 3차원 입력 데이터를 활용 • 정규화를 통해 모델의 학습 성능을 개선 • 초기에는 StandardScaler를 활용하여 전체 데이터 평탄화 후 표준화 • 이후 원래 형태를 유지하면서 np.max 를 이용한 정규화를 다시 적용 • 모델 구조는 Conv1D 레이어를 여러 층으로 쌓아 깊이를 부여 중간에 Dropout과 MaxPooling 레이어를 삽입하여 과적합 방지 및 특성 추출을 강화 • 마지막으로 Dense 레이어를 거쳐 5개의 얼굴형 클래스를 분류하는 softmax 출력층 하지만 정확도가 19.9%로 나와 얼굴형을 정확히 분류 실패 125/125 ━━━━━━━━━━━━━━━━━━━━ 7s 55ms/step - accuracy: 0.2022 - loss: 1.6102 - val_accuracy: 0.1998 - val_loss: 1.6100 Epoch 6/50 125/125 ━━━━━━━━━━━━━━━━━━━━ 7s 53ms/step - accuracy: 0.1998 - loss: 1.6107 - val_accuracy: 0.1998 - val_loss: 1.6098 32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.2000 - loss: 1.6093 Test Loss: 1.6097451448440552 Test Accuracy: 0.19979919493198395 ② 모델 개선을 위한 데이터 증강 시도 성능을 개선해보고자 회전, 이동, 좌우 반전의 기법을 적용함. • **rotate_landmarks:** 랜드마크를 기준으로 30도 회전시켜 변형된 얼굴 형태를 생성 • **translate_landmarks:** 각 좌표를 이동하여 위치 변화에 대응 • **flip_landmarks:** 좌우 반전을 통해 다양한 상황을 반영 ③ 데이터 증강 시도 후 결과 모델 학습 과정에 EarlyStopping 기법을 적용하여 50 epoch까지 학습을 진행하도록 설정 검증 손실 값(val_loss) 개선되지 않을 시 종료 증강된 데이터로 훈련을 시켰음에도 정확도가 20%로 아까와 비슷한 결과 ``` # 증강된 데이터로 모델 훈련 # augmented_X_train_landmarks의 shape 확인 후 reshape augmented_X_train_landmarks = augmented_X_train_landmarks.reshape(-1, 468, 3) # 훈련 model.fit(augmented_X_train_landmarks, augmented_y_train, epochs=50, batch_size=32, validation_data=(X_test_landmarks, y_test), callbacks=[early_stopping]) loss, accuracy = model.evaluate(X_test_landmarks, y_test) print(f"Test Loss: {loss}") print(f"Test Accuracy: {accuracy}") ``` Epoch 1/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 55ms/step - accuracy: 0.1998 - loss: 1.9799 - val_accuracy: 0.2058 - val_loss: 1.6093 Epoch 2/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 54ms/step - accuracy: 0.1922 - loss: 1.6095 - val_accuracy: 0.1948 - val_loss: 1.6093 Epoch 3/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 53ms/step - accuracy: 0.1981 - loss: 1.6095 - val_accuracy: 0.2058 - val_loss: 1.6093 Epoch 4/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 18s 48ms/step - accuracy: 0.2004 - loss: 1.6096 - val_accuracy: 0.2048 - val_loss: 1.6093 Epoch 5/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 55ms/step - accuracy: 0.1983 - loss: 1.6094 - val_accuracy: 0.2068 - val_loss: 1.6093 Epoch 6/50 373/373 ━━━━━━━━━━━━━━━━━━━━ 20s 55ms/step - accuracy: 0.2018 - loss: 1.6094 - val_accuracy: 0.2048 - val_loss: 1.6093 32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.2036 - loss: 1.6092 Test Loss: 1.6093120574951172 Test Accuracy: 0.20582328736782074 ④ 결론 Mediapipe를 사용한 좌표 기반 데이터만으로는 랜드마크 데이터를 학습시키는 데 한계가 있음. Mediapipe를 사용하지 않고 얼굴 검출에 특화된 모델을 활용하여 학습을 진행하기로 결정

모델 선정 및 기본 모델 학습

인공지능 전처리 보고서 바탕으로 **MTCNN** 활용 전처리 후 모델링 내용

① **CNN** 모델을 구축

- 데이터 이미지를 입력 받아 5개 얼굴형으로 분류하는 모델 구축
- 드롭아웃 레이어를 추가하여 과적합을 방지
- Adam 옵티마이저와 categorical_crossentropy 손실함수를 사용하여 모델을 학습하도록 설정

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 222, 222, 8)	80
max_pooling2d_4 (MaxPooling2D)	(None, 111, 111, 8)	0
conv2d_5 (Conv2D)	(None, 109, 109, 16)	1,168
max_pooling2d_5 (MaxPooling2D)	(None, 54, 54, 16)	0
conv2d_6 (Conv2D)	(None, 52, 52, 64)	9,280
max_pooling2d_6 (MaxPooling2D)	(None, 26, 26, 64)	0
conv2d_7 (Conv2D)	(None, 24, 24, 128)	73,856
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 128)	0
flatten_1 (Flatten)	(None, 18432)	0
dense_2 (Dense)	(None, 32)	589,856
dropout_1 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 5)	165

Total params: 674,405 (2.57 MB)
Trainable params: 674,405 (2.57 MB)
Non-trainable params: 0 (0.00 B)

② **CNN** 모델 학습 결과

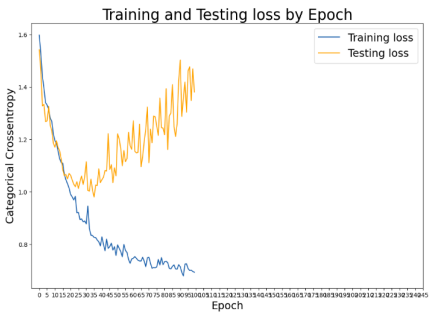
- 초기 모델 학습 결과는 **Accuracy**: 0.6779, **Loss**: 1.4418, **Test** 결과: [1.3802, 0.6784]로 정확도가 69%이다.

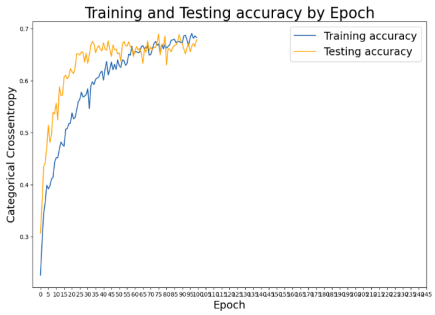
```
Epoch 100/100
125/125 ━━━━━━━━━━━━━━━━━━━━ 12s 92ms/step - accuracy: 0.6885 - loss: 0.6822 - val_accuracy: 0.6784 - val_loss: 1.3802
```

```
max(history.history['val_accuracy'])
```

```
0.6893787384033203
```

③ 학습 결과 분석





학습 단계	훈련 손실 & 검증 손실 변화	훈련 정확도 & 검증 정확도 변화
초기 (0~20 epoch)	훈련 손실과 검증 손실이 모두 감소하며 모델 학습이 원활하게 진행됨	정확도가 급격히 증가
중반 (20~50 epoch)	훈련 손실은 지속적으로 감소하지만, 검증 손실은 일정 수준에서 정체됨	훈련 정확도와 검증 정확도가 모두 60% 이상 도달
후반 (50 epoch 이후)	훈련 손실은 계속 감소하지만, 검증 손실이 증가하여 과적합 발생	훈련 정확도는 증가하지만, 검증 정확도는 일정 수준에서 변동하며 과적합 발생

④ 결론

검증 손실이 증가하고 정확도가 정체되는 현상이 나타나 일반화 성능을 개선하기 위해 추가적인 데이터 증강 또는 정규화 기법 적용이 필요 느낌
그래서 전이학습을 활용한 VGG16 모델을 적용하여 성능을 개선하기로 함.

전이학습

① **VGG16**모델 선정 이유

- 모델이 16개의 다양한 합성곱 계층으로 이루어져 얼굴의 중요한 특징이 추출 가능해 얼굴형 구분에 유리
- ImageNet 데이터셋에서 학습되어 전이학습이 비교적 빠른 편이라 이미지 분류를 효율적으로 학습하고 성능이 향상상
- 특정 도메인에 맞게 모델 조정이 가능해 적은 양의 데이터로 높은 성능

② **VGG16**모델 전이학습 과정

1. 데이터 증강

- 회전(rotation_range=20)
- 수평 뒤집기(horizontal_flip=True)

<u>학습 데이터 다양성 확보</u> 및 <u>과적합 방지</u>

```
datagen = ImageDataGenerator(rotation_range=20, horizontal_flip=True)
```

2. 최적화 알고리즘

- Adam Optimizer를 사용
- 손실 함수로는 Categorical Crossentropy를 사용

```
x = layers.Flatten()(base_model.output)

x = layers.Dense(64, activation='relu')(x)  # add 1 fully connected layer, try with 512 first
x = layers.Dropout(0.5)(x)
x = layers.Dense(5, activation='softmax')(x)  # add final layer

model_t1 = tf.keras.models.Model(base_model.input, x)


model_t1.compile(loss='categorical_crossentropy',
                 optimizer='adam',
                 metrics=['accuracy'])

model_t1.summary()
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 64)	1,605,696
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 5)	325

```
Total params: 16,320,709 (62.26 MB)
Trainable params: 1,606,021 (6.13 MB)
Non-trainable params: 14,714,688 (56.13 MB)
```

3. 학습률 및 배치 크기

- 배치 크기: 64
- 에폭: 10

```
batch_size = 64
steps_per_epoch = len(X_train_rgb) // batch_size

history_t1 = model_t1.fit(datagen.flow(X_train_rgb, y_train, batch_size=batch_size),
                          steps_per_epoch=len(X_train) // batch_size, epochs=10,
                          validation_data=(X_test_rgb, y_test))
```

```
Epoch 7/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 370s 6s/step - accuracy: 0.8232 - loss: 0.5100 - val_accuracy: 0.8798 - val_loss: 0.3856
Epoch 8/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 79s 1s/step - accuracy: 0.8438 - loss: 0.4762 - val_accuracy: 0.8778 - val_loss: 0.3850
Epoch 9/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 378s 6s/step - accuracy: 0.8438 - loss: 0.4405 - val_accuracy: 0.8858 - val_loss: 0.3481
Epoch 10/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.8281 - loss: 0.5000 - val_accuracy: 0.8838 - val_loss: 0.3485
```

```
max(history_t1.history['val_accuracy'])
```

```
0.8857715725898743
```

4. 결과

학습결과 약Loss(손실): 1.38, Accuracy(정확도): 0.89 (89%)

모델 평가 및 예측성 비교	③ **CNN vs VGG16** **CNN** 모델 예측 결과 분석 • Oval과 Oblong 구분과 Round와 Square 구분 혼동 • 학습 데이터가 충분하지 않거나, 단순한 CNN 모델이 얼굴형을 잘 구분하지 못하는 문제 발생 → 얼굴 특징을 비교적 단순한 구조로 학습하기 때문에 정확도가 낮고, 손실값도 상대적으로 큼. **VGG16** 전이학습 모델 예측 결과 분석 • 얼굴 특징을 더 세밀하게 분석하며 Oval, Oblong, Square 등 명확한 차이를 가진 얼굴형을 더 정확하게 구분 • 기존 CNN 모델이 혼동했던 사례에서 개선된 결과 → 사전 학습된 강력한 특징 추출 능력을 활용하여 높은 정확도를 보임. ④ 결론 결과적으로 **VGG16** 모델이 얼굴형 분류 문제에서 더 우수한 성능을 보이며, 최종 모델로 선택하는 것이 적절

평가 항목	CNN (MTCNN 기반)	VGG16 전이학습
정확도 (Accuracy)	약 67.8%	약 89%
손실값 (Loss)	1.38	0.38
F1-score (추가 가능)	상대적으로 낮음	상대적으로 높음
예측 속도	빠름	다소 느림
일반화 성능	오버피팅 위험 낮음	전이학습을 통한 높은 일반화