<모델훈련>

```python
import tensorflow as tf
from tensorflow.keras import layers, models

# 모델 생성
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))  # num_classes는 자음과 모음의 총 개수

# 클래스 수 설정
num_classes = 24  # 예: 14개 자음 + 10개 모음

# 모델 컴파일
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 모델 훈련
model.fit(train_images, train_labels, epochs=5, validation_data=(test_images, test_labels))

# 데이터 전처리
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)

# 모델 저장
model.save('hangul_model.h5')
```

Keras : TensorFlow의 고수준 API, 모델을 쉽게 구축하고 훈련할 수 있게 도와줌

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))

Conv2D : 2차원 합성곱 레이어 추가 -> 이미지 데이터 처리

32 : 출력 필터 수 = 레이어에서 생성할 특징 맵의 수

(3, 3) : 커널크기

activation='relu' : ReLU (Rectified Linear Unit) 활성화 함수 / 비선형성 추가

input_shape=(28, 28, 1))) : 입력 데이터의 형태 지정

➔ 28x28 크기의 흑백이미지 / 채널수 1

layers.MaxPooling2D((2, 2)) : 맥스 풀링 레이어 추가

(2,2) : 풀링 영역 크기 지정 / 입력 크기 절반으로 줄임 => 28X28 -> 14X14

64: 두번째 합성곱에서 사용하는 필터 수

layers.Flatten() : 다차원 배열(특징 맵) -> 1차원 배열

layers.Dense(64, activation='relu') : 완전 연결층 추가

- ➔ 64: 레이어의 노드 수
- ➔ activation='rule' : ReLU 활성화 함수 사용 -> 비선형성 추가

layers.Dense(num_classes, activation='softmax') : 최종 출력층 추가

num_classes: 인식할 자음과 모음의 개수 (자음: 14, 모음 : 10)

activation='softmax' : 소프트맥스 활성화 함수 사용 -> 각 클래스에 대한 확률 출력

optimizer='adam' : Adam 옵티마이저 사용 -> 효율적이고 널리 사용되는 알고리즘

loss='sparse_categorical_crossentropy' : 손실 함수로 희소 범주형 교차 엔트로피 사용 -> 다중 클래스 분류 문제에 적함

metrics=['accuracy'] : 모델의 성능을 평가할 지표 (정확도 사용)

train_images : 훈련 데이터 이미지

train_labels : 훈련 데이터의 정답 레이블

epochs=5 : 전체 데이터셋을 훈련하는 반복 횟수 (5번 훈련)

validation_data=(test_images, test_labels) : 검증 데이터로 모델의 성능 평가

HDF5형식 -> 나중에 모델을 불러와 사용