과제_2_완료_보고서

1. 해당 코드

1. Baseline 모델

Baseline 모델은 전통적인 Vanilla CNN 모델로, 주요 계층 구조는 다음과 같습니다:

- Conv2D 레이어는 특징을 추출합니다.
- MaxPooling2D 레이어는 다운샘플링을 수행합니다.
- Dropout 레이어는 과적합을 방지합니다.
- Flatten 레이어는 다차원 입력을 1차원으로 변환합니다.
- Dense 레이어는 분류를 수행합니다.

Baseline 모델 코드

```
def build_baseline_model(input_shape):
    return Sequential([
        Input(shape=input_shape),
        tf.keras.layers.Conv2D(64, (3, 3), activation='rel
u'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        Dropout(0.25),
        tf.keras.layers.Conv2D(128, (3, 3), activation='rel
u', padding='same'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        Dropout(0.25),
        tf.keras.layers.Conv2D(256, (3, 3), activation='rel
u', padding='same'),
        tf.keras.layers.Conv2D(256, (3, 3), activation='rel
u', padding='same'),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
```

```
        Dense(1000, activation='relu'),
        Dropout(0.5),
        Dense(10, activation='softmax')
    ])
```

2. 전이 학습 모델

전이 학습 모델은 MobileNetV2 사전 훈련 모델을 사용하여 일부 계층을 미세 조정하여 분류 성능을 향상시킵니다. 주요 계층 구조는 다음과 같습니다:

- MobileNetV2 사전 훈련 모델 (상단 계층 제거).
- GlobalAveragePooling2D 레이어는 글로벌 풀링을 수행합니다.
- Dense 레이어는 분류를 수행합니다.

전이 학습 모델 코드

```
def build_transfer_learning_model(input_shape):
    mobilenet = MobileNetV2(weights='imagenet', include_top=F
alse, input_shape=(224, 224, 3))
    # MobileNetV2의 첫 100개 레이어 고정
    for layer in mobilenet.layers[:100]:
        layer.trainable = False
    # 나머지 레이어는 훈련 가능
    for layer in mobilenet.layers[100:]:
        layer.trainable = True
    return Sequential([
        Input(shape=input_shape),
        Lambda(lambda image: tf.image.resize(image, (224, 22
4))),  # MobileNetV2 입력에 맞게 이미지 크기 조정
        mobilenet,
        GlobalAveragePooling2D(),
        Dense(1000, activation='relu'),
        Dense(10, activation='softmax')
    ])
```

2. 훈련 및 평가

두 모델을 동일한 초매개변수 (예: epoch와 batch size)를 사용하여 훈련 및 평가하여 공정한 비교를 보장했습니다. 또한 학습률 스케줄러를 사용하여 동적으로 학습률을 조정했습니다.

훈련 및 평가 코드

```
def build_and_train_model(model, x_train, y_train, x_val, y_v
al, epochs, batch_size, model_name):
    # 데이터 증강
    datagen = ImageDataGenerator(
        rotation_range=15,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True
    )
    datagen.fit(x_train)
    model.compile(loss='categorical_crossentropy', optimizer=
Adam(0.00002), metrics=['accuracy'])
    model.summary()
    # 학습률 스케줄러
    lr_scheduler = tf.keras.callbacks.LearningRateScheduler(l
ambda epoch: 1e-4 * 10 ** (epoch / 30))
    model.fit(datagen.flow(x_train, y_train, batch_size=batch
_size), epochs=epochs, validation_data=(x_val, y_val), verbos
e=1, callbacks=[lr_scheduler])
    model.save(f'{model_name}.keras')
    print(f"{model_name} saved as '{model_name}.keras'")
    return model
```

3. 결과 및 비교

두 모델의 훈련 로그와 최종 정확도 결과를 기록했으며, 전이 학습 모델의 성능이 Baseline 모델보다 현저히 우수했습니다.

Baseline 모델 결과

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d (MaxPooling2D)	(None, 15, 15, 64)	0
dropout (Dropout)	(None, 15, 15, 64)	0
conv2d_1 (Conv2D)	(None, 15, 15, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 128)	0
dropout_1 (Dropout)	(None, 7, 7, 128)	0
conv2d_2 (Conv2D)	(None, 7, 7, 256)	295,168
conv2d_3 (Conv2D)	(None, 7, 7, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense_2 (Dense)	(None, 1000)	2,305,000
dropout_2 (Dropout)	(None, 1000)	0
dense_3 (Dense)	(None, 10)	10,010

```
 Total params: 3,275,906 (12.50 MB)
 Trainable params: 3,275,906 (12.50 MB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 63s 79ms/step - accuracy: 0.2305 - loss: 2.0458 - val_accuracy: 0.4347 - val_loss: 1.5608 - learning_rate: 1.0000e-04
Epoch 2/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 60s 76ms/step - accuracy: 0.4240 - loss: 1.5674 - val_accuracy: 0.5197 - val_loss: 1.3471 - learning_rate: 1.0798e-04
Epoch 3/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 60s 77ms/step - accuracy: 0.4918 - loss: 1.4005 - val_accuracy: 0.5585 - val_loss: 1.2789 - learning_rate: 1.1659e-04
Epoch 4/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 60s 76ms/step - accuracy: 0.5355 - loss: 1.2991 - val_accuracy: 0.5938 - val_loss: 1.1475 - learning_rate: 1.2589e-04
baseline_model saved as 'baseline_model.keras'
Baseline model accuracy:  59.210002422332764
```

전이 학습 모델 결과

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lambda (Lambda)	(None, 224, 224, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 1000)	1,281,000
dense_1 (Dense)	(None, 10)	10,010

```
 Total params: 3,548,994 (13.54 MB)
 Trainable params: 3,152,450 (12.03 MB)
 Non-trainable params: 396,544 (1.51 MB)
Epoch 1/4
C:\Users\severin\CodLib_win\ai_learning\.env\Lib\site-packages\keras\src\trainers\data_adapters\py_d
uper().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessin
l be ignored.
  self._warn_if_super_not_called()
782/782 ━━━━━━━━━━━━━━━━━━━━ 737s 928ms/step - accuracy: 0.7097 - loss: 0.8466 - val_accuracy: 0.726
Epoch 2/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 708s 906ms/step - accuracy: 0.8650 - loss: 0.3868 - val_accuracy: 0.826
Epoch 3/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 700s 895ms/step - accuracy: 0.8940 - loss: 0.3026 - val_accuracy: 0.841
Epoch 4/4
782/782 ━━━━━━━━━━━━━━━━━━━━ 695s 888ms/step - accuracy: 0.9112 - loss: 0.2542 - val_accuracy: 0.859
transfer_learning_model saved as 'transfer_learning_model.keras'
Transfer learning model accuracy:  86.15000247955322
```

4. 실행 환경

- **Python 버전**: 3.9.12
- **TensorFlow 버전**: 2.6.0
- **실행 환경**: 로컬