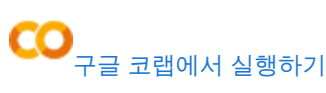


신경망 모델 훈련



손실 곡선

```
In [1]: from tensorflow import keras
from sklearn.model_selection import train_test_split

(train_input, train_target), (test_input, test_target) = keras.datasets.fashion_mnist.load_data()

train_scaled = train_input / 255.0

train_scaled, val_scaled, train_target, val_target = train_test_split(
    train_scaled, train_target, test_size=0.2, random_state=42)

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [=====] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx1-ubyte.gz
26421880/26421880 [=====] - 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [=====] - 0s 8us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422182/4422182 [=====] - 1s 8us/step

In [2]: def model_fn(a_layer=None):
    model = keras.Sequential()
    model.add(keras.layers.Flatten(input_shape=(28, 28)))
    model.add(keras.layers.Dense(100, activation='relu'))
    if a_layer: # 1f를 추가하며, 매개변수 a_layer를 받으면 층을 추가할 수 있음
        model.add(a_layer)
    model.add(keras.layers.Dense(10, activation='softmax'))
    return model

In [3]: model = model_fn()

model.summary()

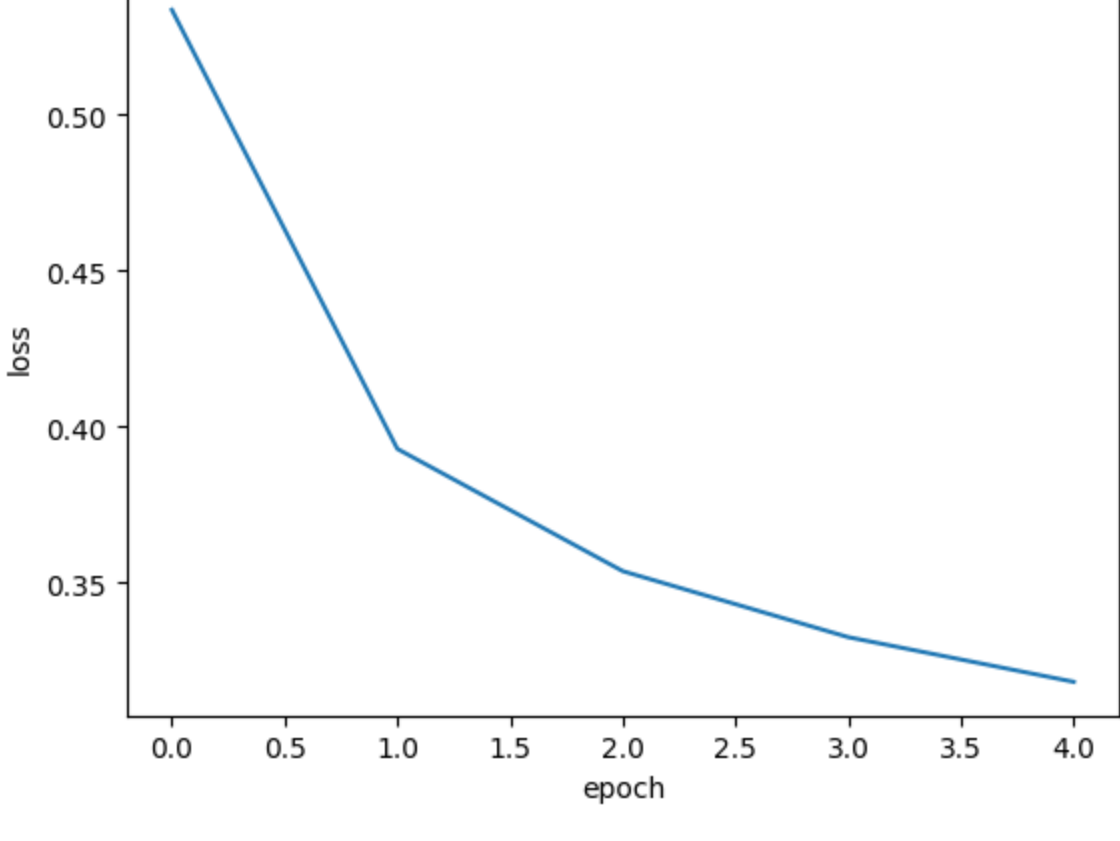
Model: "sequential"
Layer (type) Output Shape Param #
-----
flatten_1 (Flatten) (None, 784) 0
dense_1 (Dense) (None, 100) 78500
dense_1_1 (Dense) (None, 10) 1010
-----
Total params: 79510 (310.59 KB)
Trainable params: 79510 (310.59 KB)
Non-trainable params: 0 (0.00 Byte)

In [4]: model.compile(loss='sparse_categorical_crossentropy', metrics='accuracy')

history = model.fit(train_scaled, train_target, epochs=5, verbose=0) # verbose=0 : 훈련과정 출력 조절, 기본값 = 1(에포크마다 진행 막대과 함께 손실 등의 지표가 출력), 2(진행 막대를 빼고 출력)

In [5]: print(history.history.keys()) # history객체에 history키서너더에 어떤 값이 있는지 확인
dict_keys(['loss', 'accuracy'])

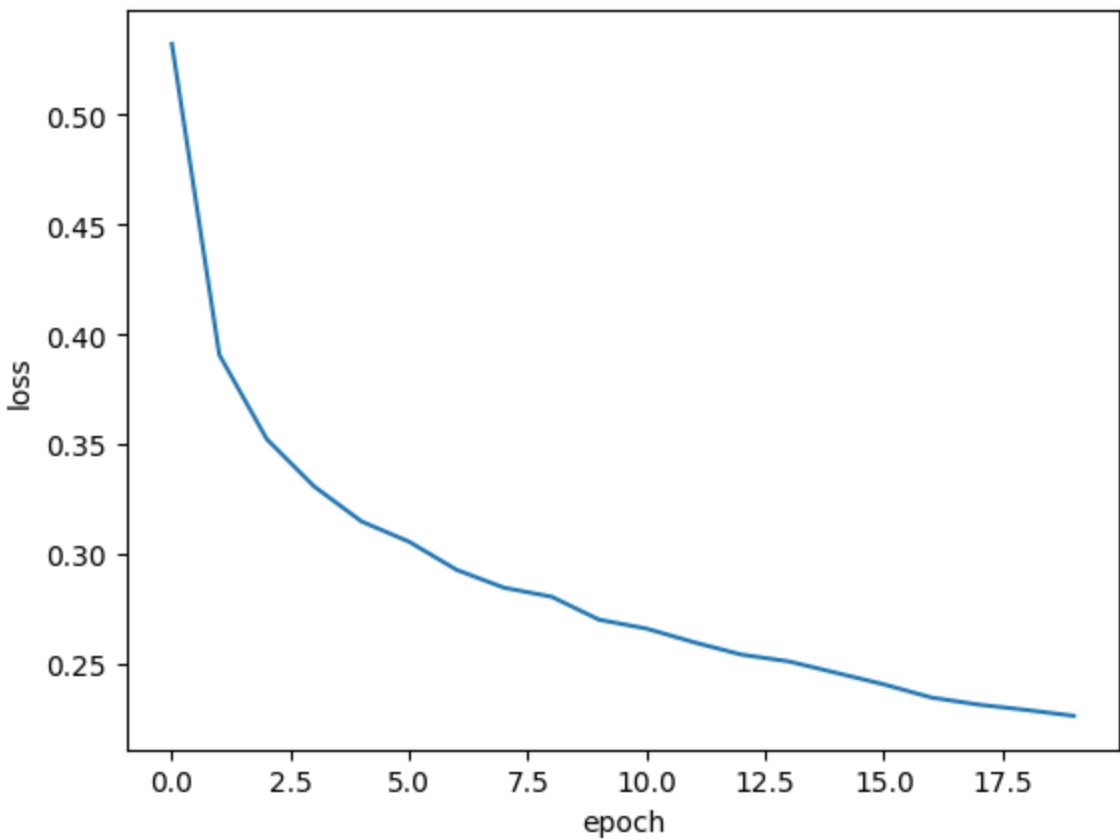
In [6]: import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.savefig('7_3-01', dpi=300)
plt.show() # 에포크마다 손실 값을 출력
```



```
In [7]: plt.plot(history.history['accuracy'])
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.savefig('7_3-02', dpi=300)
plt.show() # 에포크마다 정확도 값을 출력

In [8]: model = model_fn()
model.compile(loss='sparse_categorical_crossentropy', metrics='accuracy')
history = model.fit(train_scaled, train_target, epochs=20, verbose=0)

In [9]: plt.plot(history.history['loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.savefig('7_3-03', dpi=300)
plt.show()
```

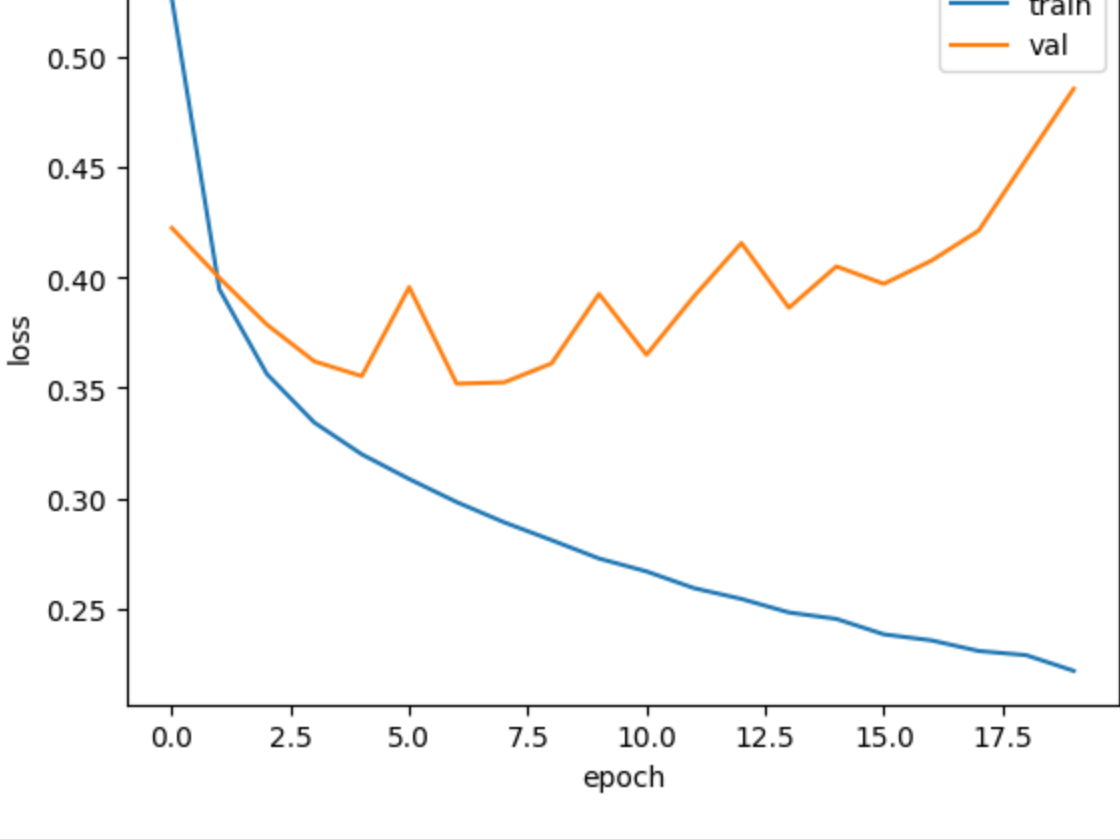


검증 손실

```
In [10]: model = model_fn()
model.compile(loss='sparse_categorical_crossentropy', metrics='accuracy')
history = model.fit(train_scaled, train_target, epochs=20, verbose=0,
                    validation_data=(val_scaled, val_target))

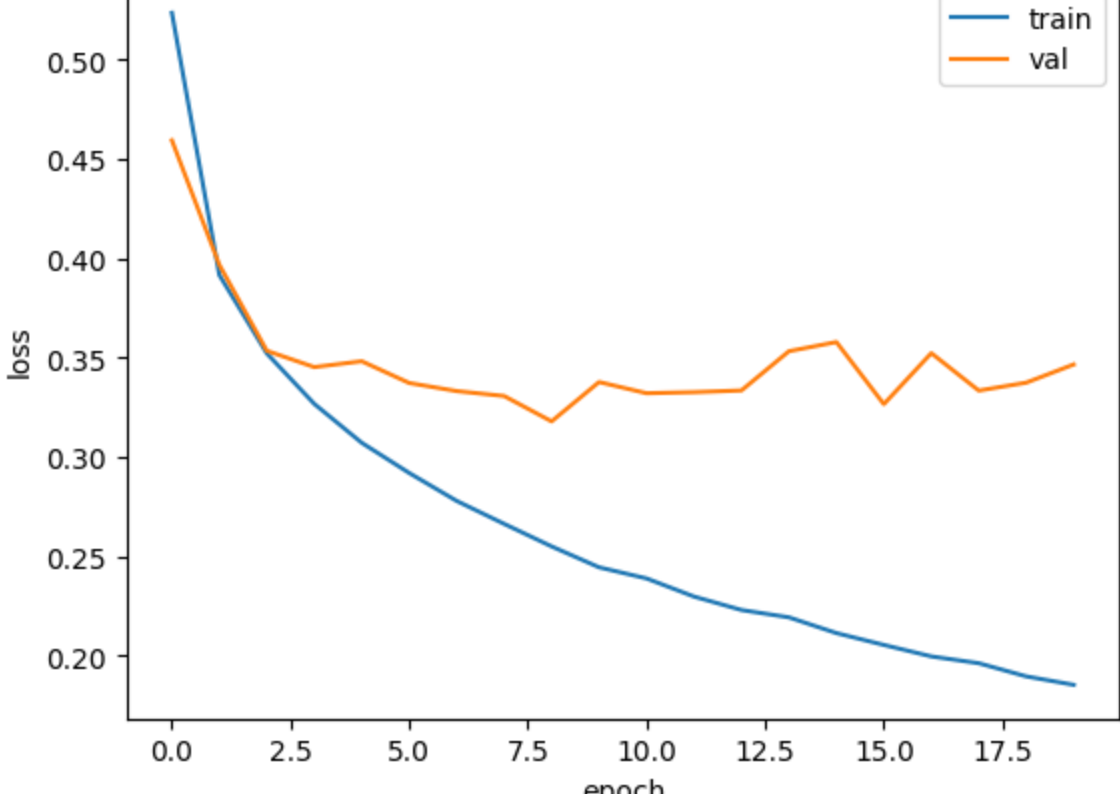
In [11]: print(history.history.keys())
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

In [12]: plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.savefig('7_3-04', dpi=300)
plt.show()
# 초기에 검증 손실이 감소하다가 다섯번째 에포크만에 다시 상승하기 시작
# 훈련 손실은 꾸준히 감소하기 때문에 과대적합 모델임을 알 수 있음
# 검증 손실이 상승하는 시점을 가능한 뒤로 늦추면 검증 세트에 대한 손실이 줄어들고, 검증 세트에 대한 정확도도 증가할 것
```



```
In [13]: model = model_fn()
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
history = model.fit(train_scaled, train_target, epochs=20, verbose=0,
                    validation_data=(val_scaled, val_target))
# fit 메서드의 결과를 history변수에 넣음, 에포크를 20으로 설정

In [14]: plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.savefig('7_3-05', dpi=300)
plt.show()
# adam을 사용하니 overfitting이 훨씬 줄어듦
```



드롭아웃

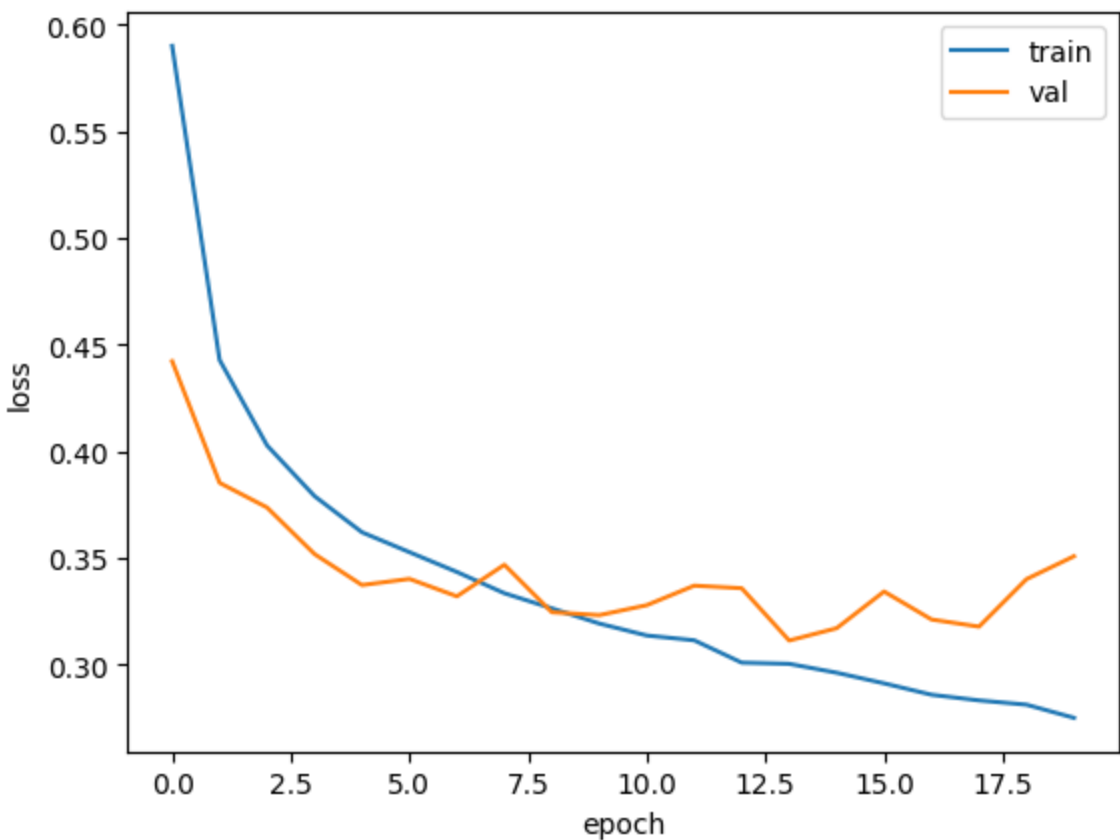
```
In [15]: model = model_fn(keras.layers.Dropout(0.3)) # 드롭아웃 0.3

model.summary()

Model: "sequential_4"
Layer (type) Output Shape Param #
-----
flatten_4 (Flatten) (None, 784) 0
dense_8 (Dense) (None, 100) 78500
dropout (Dropout) (None, 100) 0
dense_9 (Dense) (None, 10) 1010
-----
Total params: 79510 (310.59 KB)
Trainable params: 79510 (310.59 KB)
Non-trainable params: 0 (0.00 Byte)

In [16]: model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
history = model.fit(train_scaled, train_target, epochs=20, verbose=0,
                    validation_data=(val_scaled, val_target))

In [17]: plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.savefig('7_3-06', dpi=300)
plt.show() # 20번의 에포크 동안 훈련 했더니 결국 overfitting
```



모델 저장과 복원

```
In [18]: model = model_fn(keras.layers.Dropout(0.3))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
history = model.fit(train_scaled, train_target, epochs=10, verbose=0,
                    validation_data=(val_scaled, val_target))

In [19]: model.save_weights('model-weights.h5') # 모델 저장
# 훈련된 모델의 파라미터를 저장
# 기본적으로 형식폴더의 체크포인트는 포맷으로 저장하지만 파일의 확장자가 h5일 경우 HDF5 포맷으로 저장

In [20]: model.save('model-whole.h5')
```

user/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via 'model.save()'. This file format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my_model.keras')'.
saving_api.save_model(

```
In [21]: !ls -al *.h5
-rw-r--r-- 1 root root 333320 Dec 1 12:59 model-weights.h5
-rw-r--r-- 1 root root 981176 Dec 1 12:59 model-whole.h5
```

```
In [22]: model = model_fn(keras.layers.Dropout(0.3))

model.load_weights('model-weights.h5')
```

```
In [23]: import numpy as np
val_labels = np.argmax(model.predict(val_scaled), axis=-1)
# 모델에 검증 데이터를 넣은 결과를 numpy 배열로 저장
print(np.mean(val_labels == val_target))
# 위와 데이터와 예측 데이터의 결과가 일치하는 횟수를 평균으로 계산하여 정확도 출력
0.879 [=====] - 1s 2ms/step
```

```
In [24]: model = keras.models.load_model('model-whole.h5')
# 테스트를 2.3에서는 비그(https://github.com/tensorflow/tensorflow/issues/42890) 때문에 compile() 메서드를 호출해야 합니다.
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
model.evaluate(val_scaled, val_target)
```

```
375/375 [=====] - 1s 3ms/step - loss: 0.3336 - accuracy: 0.8790
[0.32273024320602417, 0.8827499747270366]
```

콜백

```
In [25]: # 콜백은 훈련 과정 중간에 어떤 작업을 수행할 수 있게 하는 객체
model = model_fn(keras.layers.Dropout(0.3))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')

checkpoint_cb = keras.callbacks.ModelCheckpoint('best-model.h5')
# model.h5에 최상의 검증 점수를 낸 모델이 저장
model.fit(train_scaled, train_target, epochs=20, verbose=0,
          validation_data=(val_scaled, val_target),
          callbacks=[checkpoint_cb])

Out[25]: <keras.callbacks.History at 0x7c334e9a34f0>
```

```
In [26]: model = keras.callbacks.load_model('best-model.h5')
# 테스트를 2.3에서는 비그(https://github.com/tensorflow/tensorflow/issues/42890) 때문에 compile() 메서드를 호출해야 합니다.
# 기본적으로 형식폴더의 체크포인트는 포맷으로 저장하지만 파일의 확장자가 h5일 경우 HDF5 포맷으로 저장
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
model.evaluate(val_scaled, val_target) # 검증 세트의 정확도를 출력
```

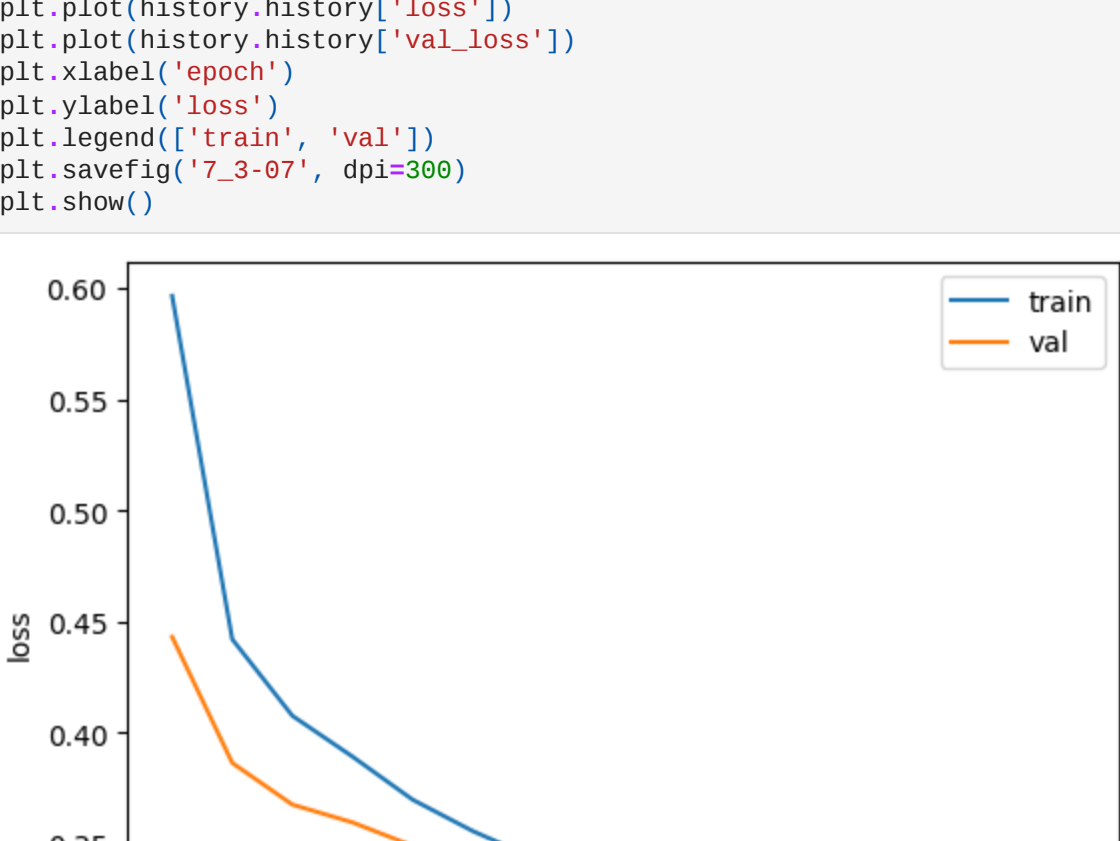
```
375/375 [=====] - 1s 2ms/step - loss: 0.3311 - accuracy: 0.8822
[0.33110445737830745, 0.8821666836738586]
```

```
In [27]: model = model_fn(keras.layers.Dropout(0.3))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')

checkpoint_cb = keras.callbacks.ModelCheckpoint('best-model.h5')
early_stopping_cb = keras.callbacks.EarlyStopping(patience=2, restore_best_weights=True)
# patience=2로 설정하여 2번 연속 검증 점수가 향상되지 않으면 훈련을 중지
# restore_best_weights=True로 설정하여, 훈련을 중지한 후의 최적 모델의 파라미터를 최상의 파라미터로 되돌림
history = model.fit(train_scaled, train_target, epochs=20, verbose=0,
                    validation_data=(val_scaled, val_target),
                    callbacks=[checkpoint_cb, early_stopping_cb])

In [28]: print(early_stopping_cb.stopped_epoch)
# 몇번째 에포크에서 훈련이 중지 되었는지 확인
# 에포크 횟수가 0부터 시작하기 때문에 +1 해서 12번째에서 훈련이 중지
15
```

```
In [29]: plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'val'])
plt.savefig('7_3-07', dpi=300)
plt.show()
```



```
In [30]: model.evaluate(val_scaled, val_target)

375/375 [=====] - 1s 3ms/step - loss: 0.3227 - accuracy: 0.8827
[0.32273024320602417, 0.8827499747270366]
```

```
Out[30]:
```

```
In [ ]:
```