Overfitting and Underfitting

透過加入提前停止回調來防止過度擬合，從而改善訓練結果。

設定環境

```
# Setup plotting
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)
plt.rc('animation', html='html5')

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex4 import *
```

首先載入 Spotify 資料集。任務是根據各種音訊特徵（例如“tempo”、“danceability”和“mode”）來預測歌曲的流行度。

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.model_selection import GroupShuffleSplit

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import callbacks

spotify = pd.read_csv('../input/dl-course-data/spotify.csv')

X = spotify.copy().dropna()
y = X.pop('track_popularity')
artists = X['track_artist']

features_num = ['danceability', 'energy', 'key', 'loudness', 'mode',
                'speechiness', 'acousticness', 'instrumentalness',
                'liveness', 'valence', 'tempo', 'duration_ms']
features_cat = ['playlist_genre']

preprocessor = make_column_transformer(
    (StandardScaler(), features_num),
    (OneHotEncoder(), features_cat),
)

# We'll do a "grouped" split to keep all of an artist's songs in one
# split or the other. This is to help prevent signal leakage.
def group_split(X, y, group, train_size=0.75):
    splitter = GroupShuffleSplit(train_size=train_size)
    train, test = next(splitter.split(X, y, groups=group))
    return (X.iloc[train], X.iloc[test], y.iloc[train], y.iloc[test])

X_train, X_valid, y_train, y_valid = group_split(X, y, artists)

X_train = preprocessor.fit_transform(X_train)
X_valid = preprocessor.transform(X_valid)
y_train = y_train / 100 # popularity is on a scale 0-100, so this rescales to 0-1.
y_valid = y_valid / 100

input_shape = [X_train.shape[1]]
print("Input shape: {}".format(input_shape))
```

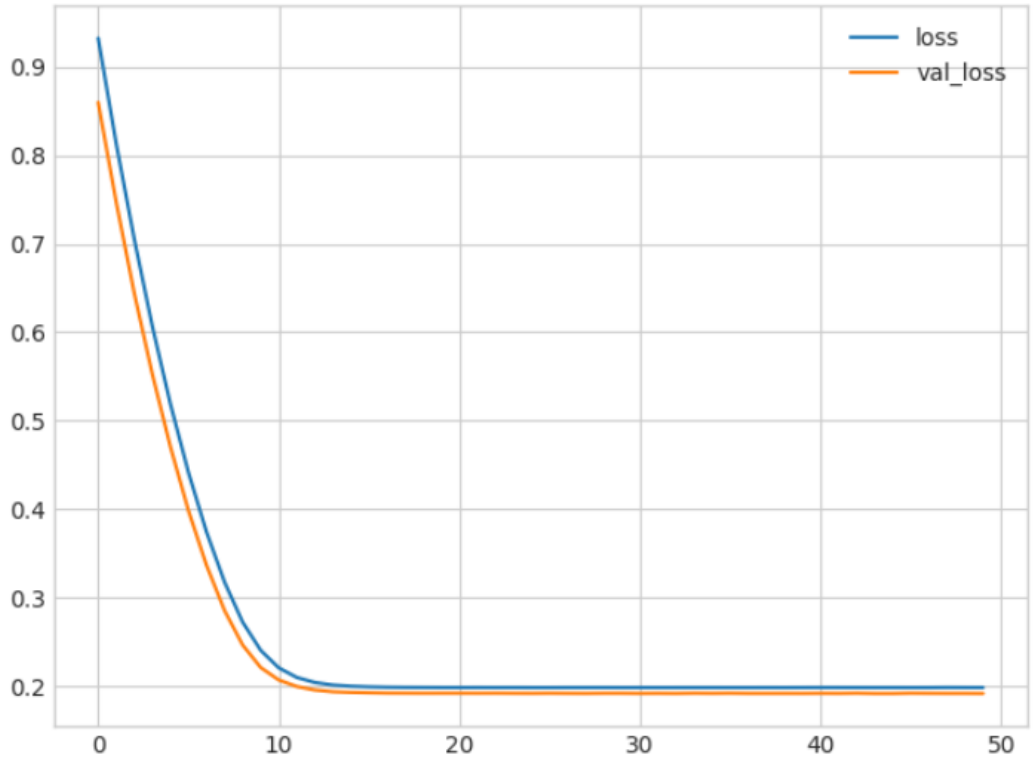
```
Input shape: [18]
```

從最簡單的網路——線性模型開始。此模型的容量較低。

運行下一個 cell，即可在 Spotify 資料集上訓練線性模型。

```
model = keras.Sequential([
    layers.Dense(1, input_shape=input_shape),
])
model.compile(
    optimizer='adam',
    loss='mae',
)
history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=50,
    verbose=0, # suppress output since we'll plot the curves
)
history_df = pd.DataFrame(history.history)
history_df.loc[0:, ['loss', 'val_loss']].plot()
print("Minimum Validation Loss: {:0.4f}".format(history_df['val_loss'].min()));
```

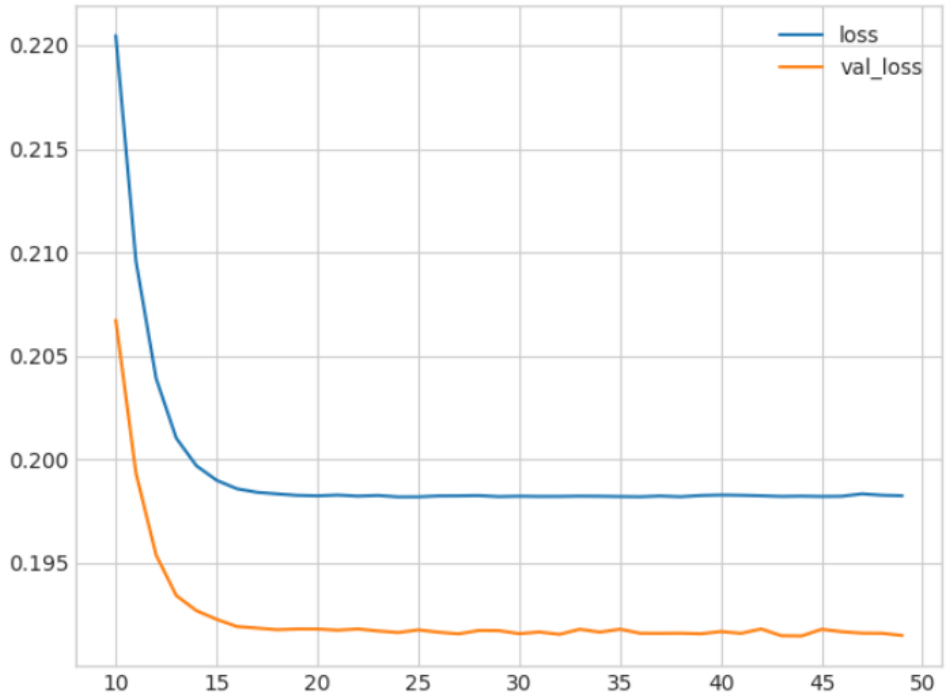
```
Minimum Validation Loss: 0.1915
```



曲線就像你在這裡看到的那樣，呈現出"hockey stick" 狀的圖案並不罕見。這使得訓練的最後部分難以看清，所以我們從第 10 個週期開始：

```
# Start the plot at epoch 10
history_df.loc[10:, ['loss', 'val_loss']].plot()
print("Minimum Validation Loss: {:0.4f}".format(history_df['val_loss'].min()));
```

```
Minimum Validation Loss: 0.1915
```



1) Evaluate Baseline

你覺得這個模型是欠擬合、過度擬合(underfitting, overfitting)，還是恰到好處？

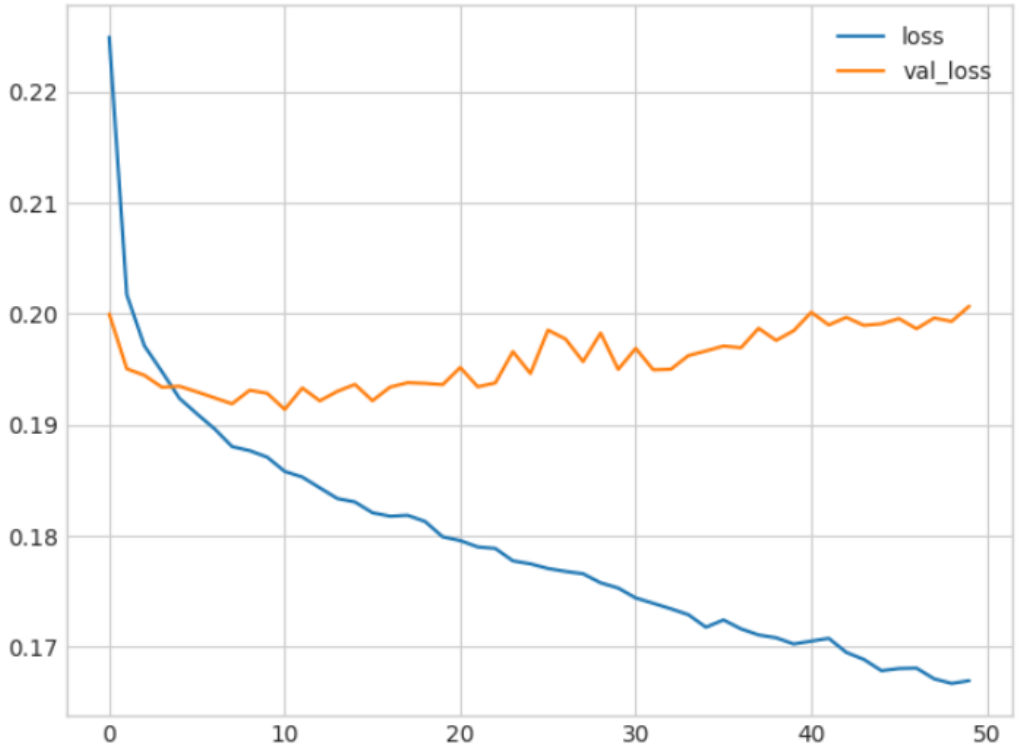
這些曲線之間的差距很小，而且驗證損失從未增加，因此網路更有可能是欠擬合而不是過度擬合。值得嘗試增加容量來驗證是否確實如此。

現在讓我們為網路增加一些容量。我們將新增三個隱藏層，每個層包含 128 個單元。運行下一個單元來訓練網路並查看學習曲線。

```
model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=input_shape),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])
model.compile(
    optimizer='adam',
    loss='mae',
)
history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=50,
)
history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss', 'val_loss']].plot()
print("Minimum Validation Loss: {:0.4f}".format(history_df['val_loss'].min()));
```

```
Epoch 1/50
48/48 [==============================] - 1s 5ms/step - loss: 0.2249 - val_loss: 0.2000
Epoch 2/50
48/48 [==============================] - 0s 3ms/step - loss: 0.2017 - val_loss: 0.1951
Epoch 3/50
```

```
Epoch 50/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1670 - val_loss: 0.2007
Minimum Validation Loss: 0.1914
```



2) Add Capacity

你對這些曲線的評價是什麼？欠擬合、過擬合(Underfitting, overfitting)，還是恰到好處？

現在，驗證損失很早就開始上升，而訓練損失則持續下降。這表示網路已經開始過擬合。此時，我們需要嘗試一些方法來防止過度擬合，例如減少單元數量或使用類似提前停止的方法。（下一課我們會講到另一種方法！）

3)Define Early Stopping Callback

現在定義一個早期停止回調，等待 5 個時期(epochs)（耐心）以使驗證損失至少發生 0.001（min_delta）的變化，並保持具有最佳損失的權重（restore_best_weights）。

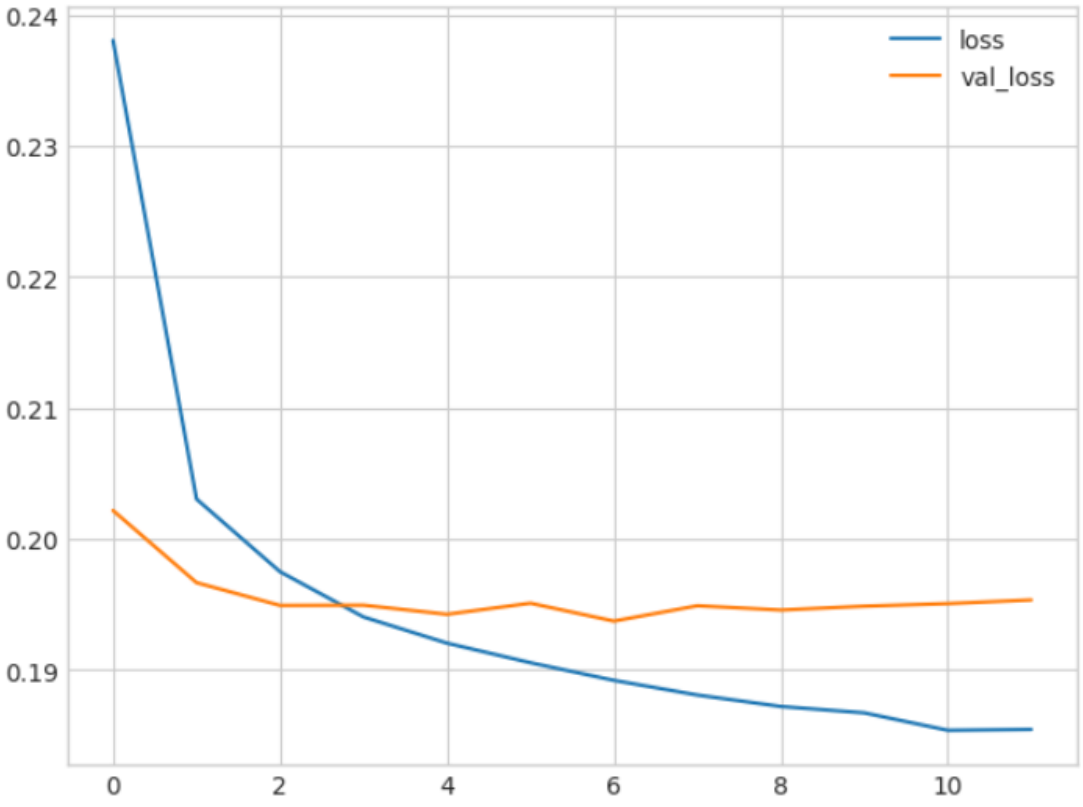
```
from tensorflow.keras import callbacks

# YOUR CODE HERE: define an early stopping callback
early_stopping = callbacks.EarlyStopping(
    patience = 5,
    min_delta = 0.001,
    restore_best_weights = True,
)
```

現在運行此 cell 來訓練模型並取得學習曲線。注意 model.fit 中的呼叫參數。

```
model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=input_shape),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])
model.compile(
    optimizer='adam',
    loss='mae',
)
history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    batch_size=512,
    epochs=50,
    callbacks=[early_stopping]
)
history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss', 'val_loss']].plot()
print("Minimum Validation Loss: {:0.4f}".format(history_df['val_loss'].min()));
```

```
Epoch 1/50
48/48 [==============================] - 1s 6ms/step - loss: 0.2380 - val_loss: 0.2021
Epoch 2/50
48/48 [==============================] - 0s 3ms/step - loss: 0.2030 - val_loss: 0.1966
Epoch 3/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1975 - val_loss: 0.1949
Epoch 4/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1940 - val_loss: 0.1949
Epoch 5/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1920 - val_loss: 0.1942
Epoch 6/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1905 - val_loss: 0.1951
Epoch 7/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1892 - val_loss: 0.1937
Epoch 8/50
48/48 [==============================] - 0s 4ms/step - loss: 0.1881 - val_loss: 0.1949
Epoch 9/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1872 - val_loss: 0.1946
Epoch 10/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1867 - val_loss: 0.1948
Epoch 11/50
48/48 [==============================] - 0s 3ms/step - loss: 0.1854 - val_loss: 0.1950
Epoch 12/50
48/48 [==============================] - 0s 4ms/step - loss: 0.1854 - val_loss: 0.1953
Minimum Validation Loss: 0.1937
```



4）訓練和解釋

與沒有提前停止的訓練相比，這是一種進步嗎？

一旦網路開始過擬合，提前停止回調就會停止訓練。此外，透過引入 restore_best_weights，我們仍然可以保留驗證損失最低的模型。