```
from·tensorflow·import·keras
from keras.datasets import imdb
%matplotlib inline
import numpy as np
import pandas as pd
from matplotlib import cm
import matplotlib.pyplot as plt
import seaborn as sns
import os
import time


from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.callbacks import EarlyStopping
from keras import models


(X_train, y_train), (X_test, y_test) = imdb.load_data()
X = np.concatenate((X_train, X_test), axis=0)
y = np.concatenate((y_train, y_test), axis=0)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
```

```
##training data shape review
print("Training data: ")
print(X.shape)
print(y.shape)
print("Classes: ")
print(np.unique(y))
```

```
Training data:
(50000,)
(50000,)
Classes:
[0 1]
```

```
print("Number of words: ")
print(len(np.unique(np.hstack(X))))
```

```
Number of words:
88585
```

```
print("Review length: ")
result = [len(x) for x in X]
print("Mean %.2f words (%f)" % (np.mean(result), np.std(result)))
# plot review length
plt.boxplot(result)
plt.show()
```

```
     Review length:
     Mean 234.76 words (172.911495)
```

```
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=5000)
```

```
def vectorize_sequences(sequences, dimension=5000):
    # Create an all-zero matrix of shape (len(sequences), dimension)
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # set specific indices of results[i] to 1s
    return results
```

```
# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
```

```
# Our vectorized labels one-hot encoder
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')


from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
model.add(layers.Dense(32, activation='relu',))
model.add(layers.Dense(1, activation='sigmoid'))


#Set validation set aside

x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]


model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])


start_time_m1 = time.time()
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
total_time_m1 = time.time() - start_time_m1
```

```
     Epoch 1/20
     30/30 [==============================] - 3s 47ms/step - loss: 0.5519 - acc: 0.7599 - val_loss: 0.3860 - val_acc: 0.8554
     Epoch 2/20
     30/30 [==============================] - 2s 52ms/step - loss: 0.2953 - acc: 0.8919 - val_loss: 0.3018 - val_acc: 0.8791
     Epoch 3/20
     30/30 [==============================] - 1s 46ms/step - loss: 0.2138 - acc: 0.9210 - val_loss: 0.2864 - val_acc: 0.8847
     Epoch 4/20
     30/30 [==============================] - 1s 37ms/step - loss: 0.1697 - acc: 0.9391 - val_loss: 0.3020 - val_acc: 0.8773
     Epoch 5/20
     30/30 [==============================] - 1s 30ms/step - loss: 0.1386 - acc: 0.9517 - val_loss: 0.3249 - val_acc: 0.8766
     Epoch 6/20
     30/30 [==============================] - 1s 32ms/step - loss: 0.1165 - acc: 0.9607 - val_loss: 0.3521 - val_acc: 0.8724
     Epoch 7/20
     30/30 [==============================] - 1s 33ms/step - loss: 0.0943 - acc: 0.9703 - val_loss: 0.3855 - val_acc: 0.8717
     Epoch 8/20
     30/30 [==============================] - 1s 32ms/step - loss: 0.0722 - acc: 0.9799 - val_loss: 0.4203 - val_acc: 0.8635
     Epoch 9/20
     30/30 [==============================] - 1s 33ms/step - loss: 0.0542 - acc: 0.9887 - val_loss: 0.4556 - val_acc: 0.8613
     Epoch 10/20
     30/30 [==============================] - 1s 32ms/step - loss: 0.0413 - acc: 0.9931 - val_loss: 0.4969 - val_acc: 0.8611
     Epoch 11/20
     30/30 [==============================] - 1s 33ms/step - loss: 0.0302 - acc: 0.9961 - val_loss: 0.5281 - val_acc: 0.8616
     Epoch 12/20
     30/30 [==============================] - 1s 31ms/step - loss: 0.0205 - acc: 0.9983 - val_loss: 0.5751 - val_acc: 0.8594
     Epoch 13/20
```

```
30/30 [==============================] - 1s 34ms/step - loss: 0.0141 - acc: 0.9997 - val_loss: 0.6076 - val_acc: 0.8600
Epoch 14/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0095 - acc: 0.9999 - val_loss: 0.6441 - val_acc: 0.8601
Epoch 15/20
30/30 [==============================] - 1s 47ms/step - loss: 0.0066 - acc: 1.0000 - val_loss: 0.6822 - val_acc: 0.8599
Epoch 16/20
30/30 [==============================] - 2s 58ms/step - loss: 0.0048 - acc: 1.0000 - val_loss: 0.7112 - val_acc: 0.8597
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0036 - acc: 1.0000 - val_loss: 0.7393 - val_acc: 0.8596
Epoch 18/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0028 - acc: 1.0000 - val_loss: 0.7646 - val_acc: 0.8598
Epoch 19/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0022 - acc: 1.0000 - val_loss: 0.7878 - val_acc: 0.8594
Epoch 20/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0018 - acc: 1.0000 - val_loss: 0.8095 - val_acc: 0.8592
```

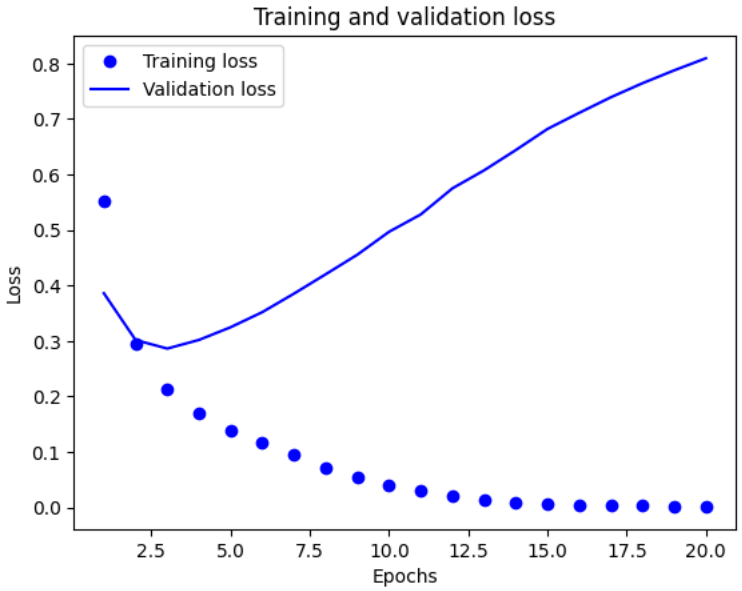
```
print("The Dense Convolutional Neural Network 1 layer took %.4f seconds to train." % (total_time_m1))
```

```
The Dense Convolutional Neural Network 1 layer took 43.5509 seconds to train.
```

```
history_dict = history.history
history_dict.keys()
```

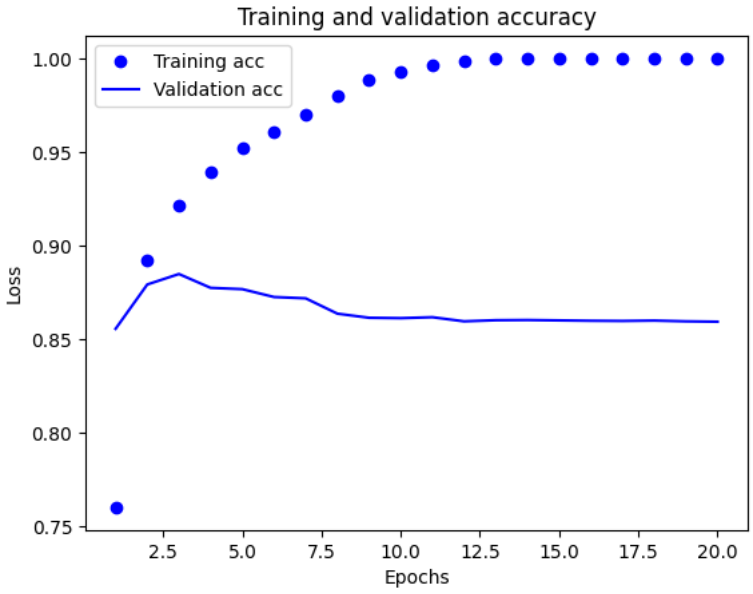
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

```
import matplotlib.pyplot as plt
%matplotlib inline

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
plt.clf()   # clear figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

plt.plot(epochs, acc, 'bo', label='Training acc')
```

```
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                160032

 dense_1 (Dense)             (None, 32)                1056

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 161,121
Trainable params: 161,121
Non-trainable params: 0
_________________________________________________________________
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, auc
#predictions
pred = model.predict(x_test)
classes_x=np.argmax(pred,axis=1)

#accuracy
accuracy_score(y_test,classes_x)
```

```
782/782 [==============================] - 3s 2ms/step
0.5
```

```
#Confusion Matrix
conf_mat = confusion_matrix(y_test, classes_x)
print(conf_mat)

conf_mat_normalized = conf_mat.astype('float') / conf_mat.sum(axis=1)[:, np.newaxis]
sns.heatmap(conf_mat_normalized)
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

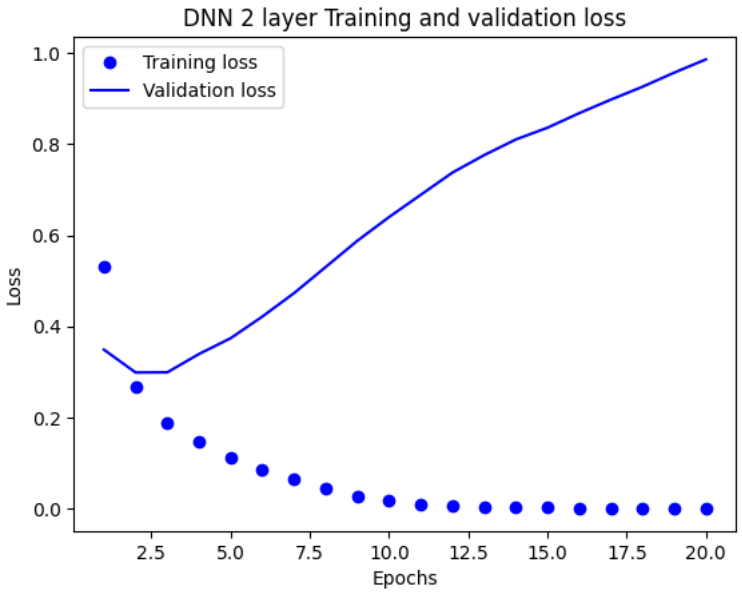
```
[[12500     0]
 [12500     0]]
Text(0.5, 23.52222222222222, 'Predicted label')
```


```
#Dense with Two Layer
model2 = models.Sequential()
model2.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(1, activation='sigmoid'))


model2.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['acc'])


start_time_m2 = time.time()
history= model2.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
total_time_m2 = time.time() - start_time_m2

print("The Dense Convolutional Neural Network 2 layers took %.4f seconds to train." % (total_time_m2))
```

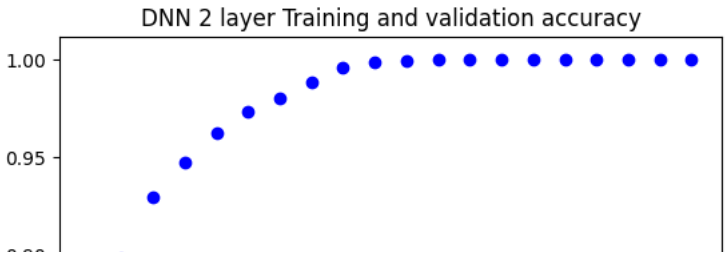
```
Epoch 1/20
30/30 [==============================] - 2s 39ms/step - loss: 0.5316 - acc: 0.7702 - val_loss: 0.3490 - val_acc: 0.8643
Epoch 2/20
30/30 [==============================] - 1s 35ms/step - loss: 0.2663 - acc: 0.8979 - val_loss: 0.2989 - val_acc: 0.8816
Epoch 3/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1894 - acc: 0.9293 - val_loss: 0.2992 - val_acc: 0.8801
Epoch 4/20
30/30 [==============================] - 1s 36ms/step - loss: 0.1461 - acc: 0.9473 - val_loss: 0.3394 - val_acc: 0.8705
Epoch 5/20
30/30 [==============================] - 1s 42ms/step - loss: 0.1121 - acc: 0.9625 - val_loss: 0.3741 - val_acc: 0.8688
Epoch 6/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0868 - acc: 0.9732 - val_loss: 0.4216 - val_acc: 0.8659
Epoch 7/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0650 - acc: 0.9803 - val_loss: 0.4729 - val_acc: 0.8668
Epoch 8/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0442 - acc: 0.9886 - val_loss: 0.5303 - val_acc: 0.8623
Epoch 9/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0273 - acc: 0.9958 - val_loss: 0.5880 - val_acc: 0.8595
Epoch 10/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0167 - acc: 0.9985 - val_loss: 0.6399 - val_acc: 0.8584
Epoch 11/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0102 - acc: 0.9996 - val_loss: 0.6888 - val_acc: 0.8585
Epoch 12/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0065 - acc: 0.9998 - val_loss: 0.7375 - val_acc: 0.8563
Epoch 13/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0045 - acc: 0.9999 - val_loss: 0.7758 - val_acc: 0.8560
Epoch 14/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0033 - acc: 0.9999 - val_loss: 0.8102 - val_acc: 0.8552
Epoch 15/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0026 - acc: 0.9999 - val_loss: 0.8360 - val_acc: 0.8569
Epoch 16/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0020 - acc: 0.9999 - val_loss: 0.8679 - val_acc: 0.8555
Epoch 17/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0015 - acc: 0.9999 - val_loss: 0.8976 - val_acc: 0.8567
Epoch 18/20
30/30 [==============================] - 1s 48ms/step - loss: 0.0012 - acc: 1.0000 - val_loss: 0.9259 - val_acc: 0.8563
```

```
Epoch 19/20
30/30 [==============================] - 1s 43ms/step - loss: 9.5346e-04 - acc: 1.0000 - val_loss: 0.9568 - val_acc: 0.8561
Epoch 20/20
30/30 [==============================] - 1s 31ms/step - loss: 7.5277e-04 - acc: 1.0000 - val_loss: 0.9860 - val_acc: 0.8558
The Dense Convolutional Neural Network 2 layers took 21.9591 seconds to train.
```

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('DNN 2 layer Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
plt.clf()   # clear figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('DNN 2 layer Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 32)                160032

 dense_4 (Dense)             (None, 32)                1056

 dense_5 (Dense)             (None, 32)                1056

 dense_6 (Dense)             (None, 1)                 33

=================================================================
Total params: 162,177
Trainable params: 162,177
Non-trainable params: 0
_________________________________________________________________
```

```
from numpy.ma.core import argmax
pred = model2.predict(x_test)
classes_x=argmax(pred,axis=-1)
#accuracy
accuracy_score(y_test,classes_x)
```

```
782/782 [==============================] - 2s 2ms/step
0.5
```