In [1]:

```
import tensorflow as tf
```

In [2]:

```
tf.__version__
```

Out[2]:

```
'2.0.0'
```

In [3]:

```
from tensorflow.keras import layers, optimizers, metrics, datasets, Sequential, models
```

In [4]:

```
import os
```

In [5]:

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

In [39]:

```
(x_train, y_train), (x_test, y_test) = datasets.imdb.load_data(num_words=10000)
```

In [7]:

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[7]:

```
((25000,), (25000,), (25000,), (25000,))
```

In [8]:

```
x_train[0][:10]
```

Out[8]:

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65]
```

In [9]:

```
y_train[0]
```

Out[9]:

```
1
```

In [10]:

```
max([max(sequence) for sequence in x_train])
```

Out[10]:

9999

数字和单词映射表，索引减3，因为0，1，2为padding、start of sequence、unknown保留的索引

In [11]:

```
word_index = datasets.imdb.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in x_train[0]])
```

In [12]:

```
decoded_review
```

Out[12]:

```
"? this film was just brilliant casting location scenery story direction eve
ryone's really suited the part they played and you could just imagine being
there robert ? is an amazing actor and now the same being director ? father
came from the same scottish island as myself so i loved the fact there was a
real connection with this film the witty remarks throughout the film were gr
eat it was just brilliant so much that i bought the film as soon as it was r
eleased for ? and would recommend it to everyone to watch and the fly fishin
g was amazing really cried at the end it was so sad and you know what they s
ay if you cry at a film it must have been good and this definitely was also
? to the two little boy's that played the ? of norman and paul they were jus
t brilliant children are often left out of the ? list i think because the st
ars that play them all grown up are such a big profile for the whole film bu
t these children are amazing and should be praised for what they have done d
on't you think the whole story was so lovely because it was true and was som
eone's life after all that was shared with us all"
```

In [13]:

```
import numpy as np
```

向量化

In [14]:

```
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
```

In [15]:

```
x_train = vectorize_sequences(x_train)
x_test = vectorize_sequences(x_test)
```

In [16]:

```
x_train.shape
```

Out[16]:

```
(25000, 10000)
```

In [17]:

```
x_train[0].shape
```

Out[17]:

```
(10000,)
```

In [18]:

```
x_train[0]
```

Out[18]:

```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [19]:

```
y_train = np.asarray(y_train).astype('float32')
y_test = np.asarray(y_test).astype('float32')
```

In [20]:

```
y_train[0]
```

Out[20]:

```
1.0
```

原始网络

In [21]:

```
model1 = models.Sequential()
model1.add(layers.Dense(16, activation='relu', input_shape=(10000, )))
model1.add(layers.Dense(16, activation='relu'))
model1.add(layers.Dense(1, activation='sigmoid'))
```

In [22]:

```
model1.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
```

留出验证集

In [23]:

```
x_val = x_train[:10000]
x_train = x_train[10000:]

y_val = y_train[:10000]
y_train = y_train[10000:]
```

In [24]:

```
history1 = model1.fit(x_train, y_train, epochs=20, batch_size=512, validation_data=(x_val,
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 1s 98us/sample - loss: 0.5
085 - accuracy: 0.7811 - val_loss: 0.3847 - val_accuracy: 0.8586
Epoch 2/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.2
965 - accuracy: 0.9029 - val_loss: 0.3093 - val_accuracy: 0.8803
Epoch 3/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.2
159 - accuracy: 0.9295 - val_loss: 0.2905 - val_accuracy: 0.8832
Epoch 4/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.1
658 - accuracy: 0.9472 - val_loss: 0.2785 - val_accuracy: 0.8893
Epoch 5/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.1
383 - accuracy: 0.9557 - val_loss: 0.3312 - val_accuracy: 0.8689
Epoch 6/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
143 - accuracy: 0.9661 - val_loss: 0.2966 - val_accuracy: 0.8851
Epoch 7/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.0
944 - accuracy: 0.9725 - val_loss: 0.3438 - val_accuracy: 0.8730
Epoch 8/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.0
770 - accuracy: 0.9785 - val_loss: 0.3389 - val_accuracy: 0.8774
Epoch 9/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.0
637 - accuracy: 0.9839 - val_loss: 0.3690 - val_accuracy: 0.8740
Epoch 10/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
544 - accuracy: 0.9851 - val_loss: 0.4162 - val_accuracy: 0.8670
Epoch 11/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.0
432 - accuracy: 0.9901 - val_loss: 0.4247 - val_accuracy: 0.8686
Epoch 12/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.0
364 - accuracy: 0.9922 - val_loss: 0.4352 - val_accuracy: 0.8750
Epoch 13/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
284 - accuracy: 0.9947 - val_loss: 0.4860 - val_accuracy: 0.8740
Epoch 14/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
254 - accuracy: 0.9951 - val_loss: 0.4942 - val_accuracy: 0.8726
Epoch 15/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
173 - accuracy: 0.9977 - val_loss: 0.5350 - val_accuracy: 0.8651
Epoch 16/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
170 - accuracy: 0.9967 - val_loss: 0.5542 - val_accuracy: 0.8691
Epoch 17/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
087 - accuracy: 0.9996 - val_loss: 0.5887 - val_accuracy: 0.8633
Epoch 18/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
111 - accuracy: 0.9982 - val_loss: 0.6186 - val_accuracy: 0.8673
Epoch 19/20
```

```
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
100 - accuracy: 0.9985 - val_loss: 0.6538 - val_accuracy: 0.8660
Epoch 20/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
041 - accuracy: 0.9998 - val_loss: 0.6808 - val_accuracy: 0.8640
```

In [25]:

```
history_dict1 = history1.history
history_dict1.keys()
```

Out[25]:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

绘图

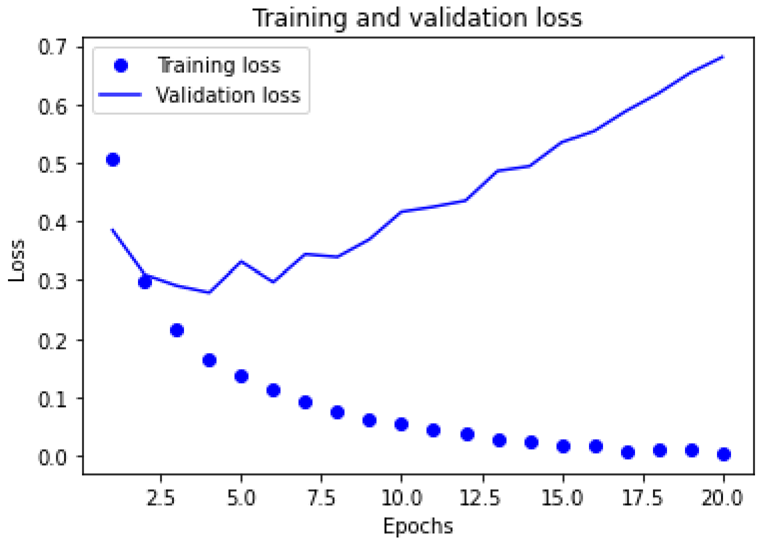
In [26]:

```
loss1 = history_dict1['loss']
val_loss1 = history_dict1['val_loss']

epochs1 = range(1, len(loss1) + 1)

plt.plot(epochs1, loss1, 'bo', label='Training loss')
plt.plot(epochs1, val_loss1, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

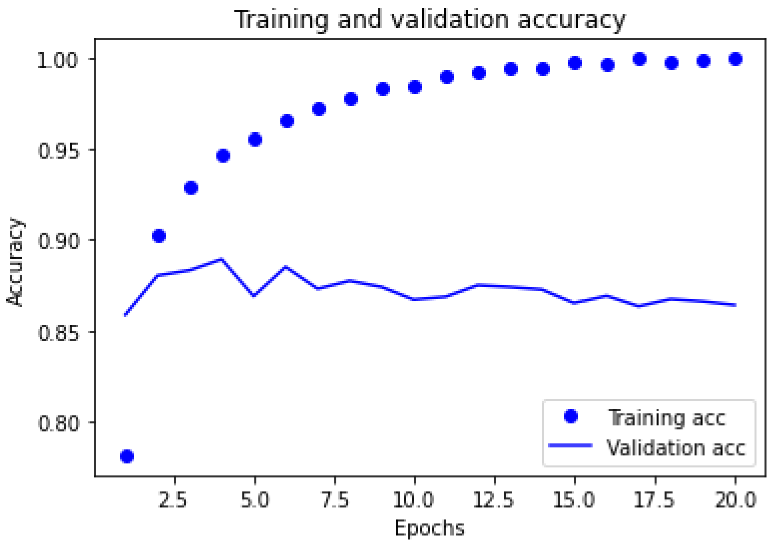


In [27]:

```
# plt.clf()      # 清除图像
acc1 = history_dict1['accuracy']
val_acc1 = history_dict1['val_accuracy']

plt.plot(epochs1, acc1, 'bo', label='Training acc')
plt.plot(epochs1, val_acc1, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [28]:

```
test_loss1, test_acc1 = model1.evaluate(x_test, y_test, verbose=0)
```

In [29]:

```
test_loss1, test_acc1
```

Out[29]:

```
(0.7380681242799759, 0.85072)
```

In [30]:

```
model1.predict(x_test)
```

Out[30]:

```
array([[0.00702149],
       [1.        ],
       [0.9980731 ],
       ...,
       [0.00798124],
       [0.01157042],
       [0.89454895]], dtype=float32)
```

尝试更小的网络

In [31]:

```
model2 = models.Sequential()
model2.add(layers.Dense(4, activation='relu', input_shape=(10000,)))
model2.add(layers.Dense(4, activation='relu'))
model2.add(layers.Dense(1, activation='sigmoid'))

model2.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

history2 =model2.fit(x_train, y_train, epochs=20, batch_size=512, validation_data=(x_val, y
test_loss2, test_acc2 = model2.evaluate(x_test, y_test, verbose=0)
test_loss2, test_acc2
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 1s 78us/sample - loss: 0.6
074 - accuracy: 0.7551 - val_loss: 0.5352 - val_accuracy: 0.8354
Epoch 2/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.4
769 - accuracy: 0.8745 - val_loss: 0.4466 - val_accuracy: 0.8659
Epoch 3/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.3
861 - accuracy: 0.9031 - val_loss: 0.3839 - val_accuracy: 0.8773
Epoch 4/20
15000/15000 [==============================] - 1s 46us/sample - loss: 0.3
160 - accuracy: 0.9198 - val_loss: 0.3447 - val_accuracy: 0.8824
Epoch 5/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.2
644 - accuracy: 0.9315 - val_loss: 0.3111 - val_accuracy: 0.8893
Epoch 6/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.2
245 - accuracy: 0.9411 - val_loss: 0.2915 - val_accuracy: 0.8895
Epoch 7/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
942 - accuracy: 0.9480 - val_loss: 0.2804 - val_accuracy: 0.8889
Epoch 8/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
684 - accuracy: 0.9553 - val_loss: 0.2825 - val_accuracy: 0.8855
Epoch 9/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
470 - accuracy: 0.9600 - val_loss: 0.2729 - val_accuracy: 0.8885
Epoch 10/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
294 - accuracy: 0.9645 - val_loss: 0.2755 - val_accuracy: 0.8880
Epoch 11/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.1
125 - accuracy: 0.9679 - val_loss: 0.2897 - val_accuracy: 0.8867
Epoch 12/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
942 - accuracy: 0.9739 - val_loss: 0.3000 - val_accuracy: 0.8843
Epoch 13/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
763 - accuracy: 0.9799 - val_loss: 0.3389 - val_accuracy: 0.8775
Epoch 14/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
652 - accuracy: 0.9842 - val_loss: 0.3474 - val_accuracy: 0.8786
Epoch 15/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
553 - accuracy: 0.9870 - val_loss: 0.3528 - val_accuracy: 0.8803
Epoch 16/20
```

```
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
477 - accuracy: 0.9897 - val_loss: 0.3692 - val_accuracy: 0.8794
Epoch 17/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
403 - accuracy: 0.9918 - val_loss: 0.3945 - val_accuracy: 0.8766
Epoch 18/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
342 - accuracy: 0.9938 - val_loss: 0.4095 - val_accuracy: 0.8739
Epoch 19/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
291 - accuracy: 0.9953 - val_loss: 0.4251 - val_accuracy: 0.8737
Epoch 20/20
15000/15000 [==============================] - 1s 47us/sample - loss: 0.0
242 - accuracy: 0.9962 - val_loss: 0.4487 - val_accuracy: 0.8711
```

Out[31]:

```
(0.4910747938275337, 0.8576)
```

In [32]:

```
history_dict2 = history2.history
```

更小的网络绘图,比较原始网络和更小网络的验证损失和训练损失

In [33]:

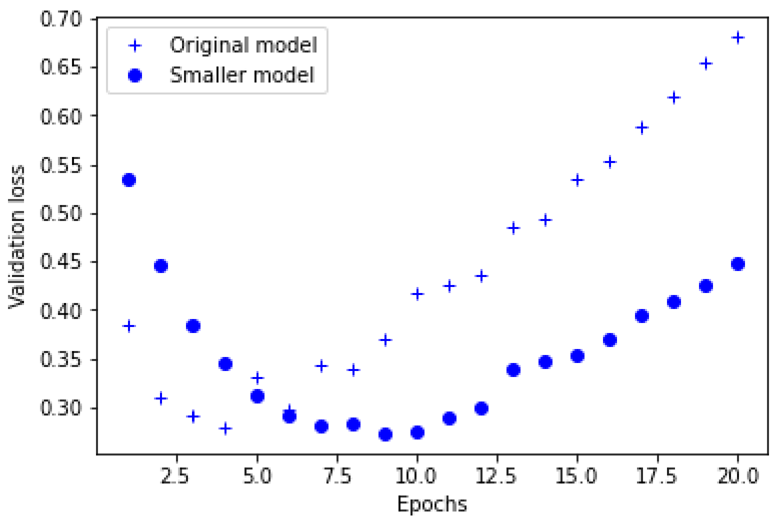
```
loss2 = history_dict2['loss']
val_loss2 = history_dict2['val_loss']

epochs2 = range(1, len(loss2) + 1)

plt.plot(epochs1, val_loss1, 'b+', label='Original model')
plt.plot(epochs2, val_loss2, 'bo', label='Smaller model')

plt.xlabel('Epochs')
plt.ylabel('Validation loss')
plt.legend()

plt.show()
```



In [34]:

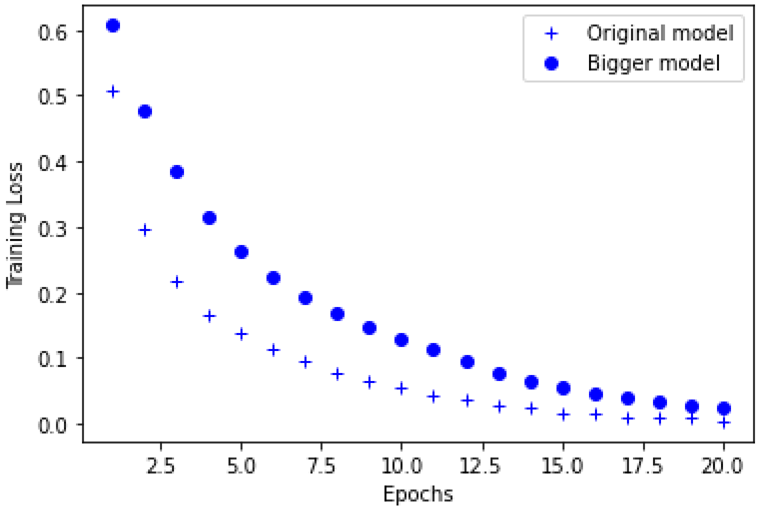
```
loss2 = history_dict2['loss']
val_loss2 = history_dict2['val_loss']

epochs2 = range(1, len(loss2) + 1)

plt.plot(epochs1, loss1, 'b+', label='Original model')
plt.plot(epochs2, loss2, 'bo', label='Bigger model')

plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.legend()

plt.show()
```



尝试更大的网络

In [35]:

```
model3 = models.Sequential()
model3.add(layers.Dense(512, activation='relu', input_shape=(10000,)))
model3.add(layers.Dense(512, activation='relu'))
model3.add(layers.Dense(1, activation='sigmoid'))

model3.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])

history3 =model3.fit(x_train, y_train, epochs=20, batch_size=512, validation_data=(x_val, y
test_loss3, test_acc3 = model3.evaluate(x_test, y_test, verbose=0)
test_loss3, test_acc3
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 1s 84us/sample - loss: 0.5
439 - accuracy: 0.7581 - val_loss: 0.3316 - val_accuracy: 0.8582
Epoch 2/20
15000/15000 [==============================] - 1s 56us/sample - loss: 0.2
462 - accuracy: 0.9030 - val_loss: 0.4513 - val_accuracy: 0.8170
Epoch 3/20
15000/15000 [==============================] - 1s 56us/sample - loss: 0.1
644 - accuracy: 0.9403 - val_loss: 0.2912 - val_accuracy: 0.8888
Epoch 4/20
15000/15000 [==============================] - 1s 56us/sample - loss: 0.1
106 - accuracy: 0.9678 - val_loss: 0.3565 - val_accuracy: 0.8804
Epoch 5/20
15000/15000 [==============================] - 1s 59us/sample - loss: 0.0
974 - accuracy: 0.9767 - val_loss: 0.3174 - val_accuracy: 0.8854
Epoch 6/20
15000/15000 [==============================] - 1s 58us/sample - loss: 0.0
061 - accuracy: 0.9995 - val_loss: 0.4903 - val_accuracy: 0.8839
Epoch 7/20
15000/15000 [==============================] - 1s 57us/sample - loss: 7.6
466e-04 - accuracy: 1.0000 - val_loss: 0.6190 - val_accuracy: 0.8855
Epoch 8/20
15000/15000 [==============================] - 1s 57us/sample - loss: 1.3
052e-04 - accuracy: 1.0000 - val_loss: 0.7124 - val_accuracy: 0.8828
Epoch 9/20
15000/15000 [==============================] - 1s 57us/sample - loss: 2.0
487e-05 - accuracy: 1.0000 - val_loss: 0.8155 - val_accuracy: 0.8849
Epoch 10/20
15000/15000 [==============================] - 1s 57us/sample - loss: 4.2
406e-06 - accuracy: 1.0000 - val_loss: 0.9105 - val_accuracy: 0.8827
Epoch 11/20
15000/15000 [==============================] - 1s 57us/sample - loss: 1.0
261e-06 - accuracy: 1.0000 - val_loss: 0.9878 - val_accuracy: 0.8850
Epoch 12/20
15000/15000 [==============================] - 1s 58us/sample - loss: 3.1
345e-07 - accuracy: 1.0000 - val_loss: 1.0675 - val_accuracy: 0.8839
Epoch 13/20
15000/15000 [==============================] - 1s 58us/sample - loss: 1.1
494e-07 - accuracy: 1.0000 - val_loss: 1.1298 - val_accuracy: 0.8846
Epoch 14/20
15000/15000 [==============================] - 1s 58us/sample - loss: 5.0
739e-08 - accuracy: 1.0000 - val_loss: 1.1772 - val_accuracy: 0.8840
Epoch 15/20
15000/15000 [==============================] - 1s 58us/sample - loss: 2.8
244e-08 - accuracy: 1.0000 - val_loss: 1.2088 - val_accuracy: 0.8842
Epoch 16/20
```

```
15000/15000 [==============================] - 1s 59us/sample - loss: 1.9
177e-08 - accuracy: 1.0000 - val_loss: 1.2309 - val_accuracy: 0.8840
Epoch 17/20
15000/15000 [==============================] - 1s 58us/sample - loss: 1.4
484e-08 - accuracy: 1.0000 - val_loss: 1.2472 - val_accuracy: 0.8843
Epoch 18/20
15000/15000 [==============================] - 1s 58us/sample - loss: 1.1
684e-08 - accuracy: 1.0000 - val_loss: 1.2612 - val_accuracy: 0.8839
Epoch 19/20
15000/15000 [==============================] - 1s 58us/sample - loss: 9.8
575e-09 - accuracy: 1.0000 - val_loss: 1.2727 - val_accuracy: 0.8835
Epoch 20/20
15000/15000 [==============================] - 1s 58us/sample - loss: 8.6
062e-09 - accuracy: 1.0000 - val_loss: 1.2817 - val_accuracy: 0.8838
```

Out[35]:

```
(1.363865372863561, 0.87364)
```

In [36]:

```
history_dict3 = history3.history
```

更大的网络绘图,比较原始网络和更大网络的验证损失和训练损失

In [37]:

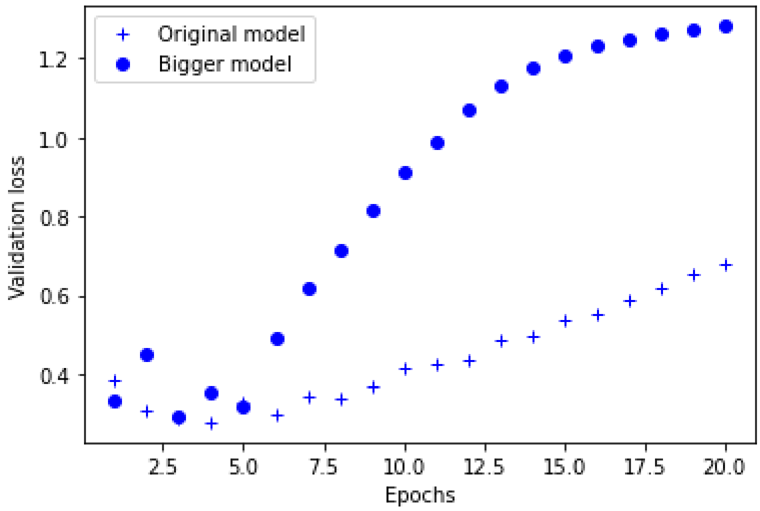
```
loss3 = history_dict3['loss']
val_loss3 = history_dict3['val_loss']

epochs3 = range(1, len(loss3) + 1)

plt.plot(epochs1, val_loss1, 'b+', label='Original model')
plt.plot(epochs3, val_loss3, 'bo', label='Bigger model')

plt.xlabel('Epochs')
plt.ylabel('Validation loss')
plt.legend()

plt.show()
```



In [38]:

```
loss3 = history_dict3['loss']
val_loss3 = history_dict3['val_loss']

epochs3 = range(1, len(loss3) + 1)

plt.plot(epochs1, loss1, 'b+', label='Original model')
plt.plot(epochs3, loss3, 'bo', label='Bigger model')

plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.legend()

plt.show()
```