```
!pip install keras
```

```
Requirement already satisfied: keras in
/usr/local/lib/python3.10/dist-packages (2.13.1)
```

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.13.0)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.1.21 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.57.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: keras<2.14,>=2.13.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.1)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: numpy<=1.24.3,>=1.22 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorboard<2.14,>=2.13 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.0)
Requirement already satisfied: tensorflow-estimator<2.14,>=2.13.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.13.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions<4.6.0,>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.15.0)
```

```
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.33.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13-
>tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13-
>tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13-
>tensorflow) (3.4.4)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13-
>tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.14,>=2.13->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.14,>=2.13-
>tensorflow) (2.3.7)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.14,>=2.13->tensorflow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.14,>=2.13->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.14,>=2.13->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<1.1,>=0.5->tensorboard<2.14,>=2.13->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.14,>=2.13->tensorflow) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.14,>=2.13->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.14,>=2.13->tensorflow) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.14,>=2.13->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.14,>=2.13->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.14,>=2.13->tensorflow) (0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5-
>tensorboard<2.14,>=2.13->tensorflow) (3.2.2)
```

```python
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

```python
train_data[0]
```

```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
 19,
 14,
 22,
 4,
 1920,
 4613,
 469,
 4,
 22,
 71,
 87,
 12,
 16,
 43,
 530,
 38,
 76,
 15,
 13,
 1247,
 4,
 22,
 17,
 515,
 17,
 12,
 16,
 626,
 18,
 2,
 5,
 62,
 386,
 12,
 8,
 316,
 8,
 106,
 5,
 4,
 2223,
 5244,
 16,
 480,
 66,
 3785,
 33,
 4,
 130,
 12,
 16,
 38,
 619,
 5,
 25,
 124,
 51,
 36,
 135,
 48,
 25,
 1415,
 33,
 6,
 22,
 12,
 215,
 28,
 77,
 52,
 5,
 14,
 407,
 16,
 82,
 2,
 8,
 4,
 107,
 117,
 5952,
 15,
 256,
 4,
 2,
 7,
 3766,
 5,
 723,
 36,
 71,
 43,
 530,
 476,
 26,
 400,
 317,
 46,
 7,
 4,
 2,
 1029,
 13,
 104,
 88,
 4,
 381,
 15,
 297,
 98,
 32,
 2071,
 56,
 26,
 141,
 6,
 194,
 7486,
 18,
 4,
 226,
 22,
 21,
 134,
 476,
 26,
 480,
 5,
 144,
 30,
 5535,
 18,
 51,
 36,
 28,
 224,
 92,
 25,
 104,
 4,
 226,
 65,
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 32]
```

```
train_labels[0]
```

```
1
```

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

```
9999
```

```
word_index = 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 train_data[0]])
```

```
import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        for j in sequence:
            results[i, j] = 1.
    return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

```
x_train[0]
```

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

```
y_train = np.asarray(train_labels).astype("float32")
y_test = np.asarray(test_labels).astype("float32")
```

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

```
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
```

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

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

```
Epoch 1/20
30/30 [==============================] - 9s 68ms/step - loss: 0.5213 -
accuracy: 0.7765 - val_loss: 0.3915 - val_accuracy: 0.8661
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.3195 -
accuracy: 0.8955 - val_loss: 0.3560 - val_accuracy: 0.8547
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.2386 -
accuracy: 0.9226 - val_loss: 0.2839 - val_accuracy: 0.8897
Epoch 4/20
```

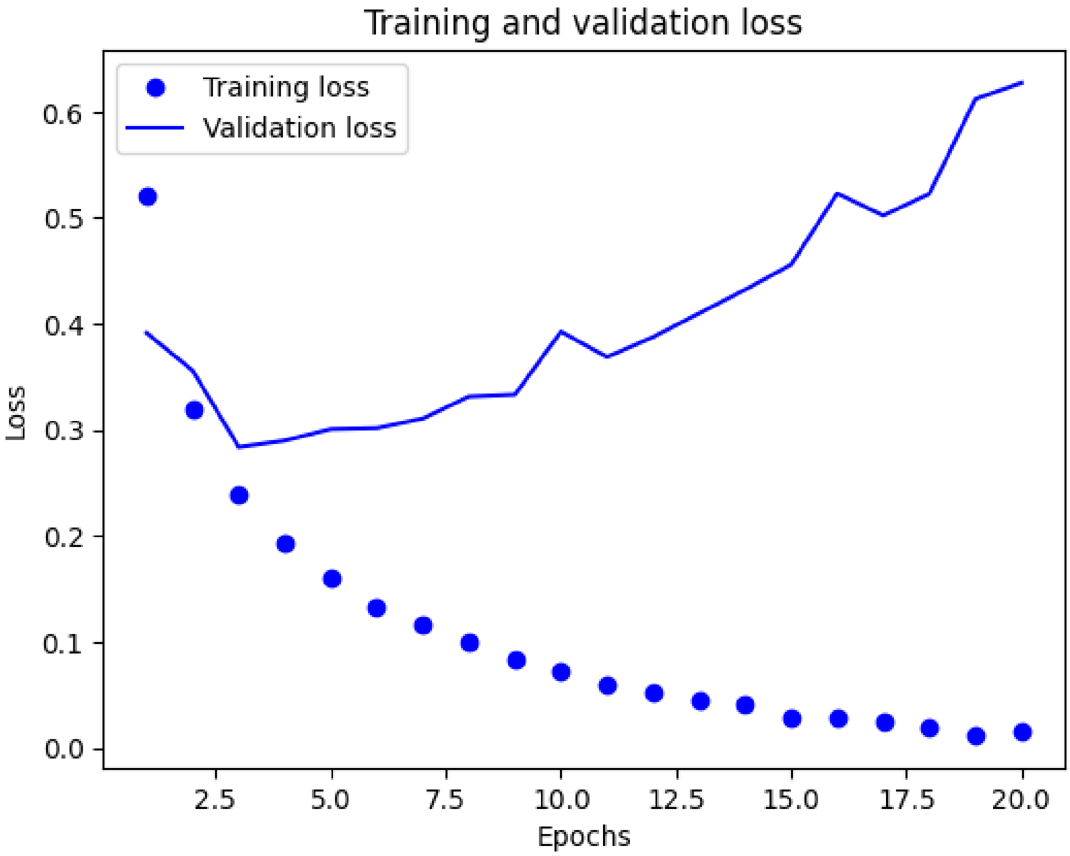
```
30/30 [==============================] - 1s 25ms/step - loss: 0.1928 -
accuracy: 0.9351 - val_loss: 0.2900 - val_accuracy: 0.8830
Epoch 5/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1611 -
accuracy: 0.9473 - val_loss: 0.3007 - val_accuracy: 0.8799
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1328 -
accuracy: 0.9583 - val_loss: 0.3017 - val_accuracy: 0.8817
Epoch 7/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1171 -
accuracy: 0.9639 - val_loss: 0.3108 - val_accuracy: 0.8788
Epoch 8/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1003 -
accuracy: 0.9697 - val_loss: 0.3314 - val_accuracy: 0.8754
Epoch 9/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0843 -
accuracy: 0.9767 - val_loss: 0.3333 - val_accuracy: 0.8823
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0721 -
accuracy: 0.9807 - val_loss: 0.3927 - val_accuracy: 0.8667
Epoch 11/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0598 -
accuracy: 0.9855 - val_loss: 0.3688 - val_accuracy: 0.8787
Epoch 12/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0524 -
accuracy: 0.9872 - val_loss: 0.3876 - val_accuracy: 0.8778
Epoch 13/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0451 -
accuracy: 0.9900 - val_loss: 0.4100 - val_accuracy: 0.8754
Epoch 14/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0415 -
accuracy: 0.9907 - val_loss: 0.4323 - val_accuracy: 0.8732
Epoch 15/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0285 -
accuracy: 0.9949 - val_loss: 0.4560 - val_accuracy: 0.8738
Epoch 16/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0277 -
accuracy: 0.9945 - val_loss: 0.5230 - val_accuracy: 0.8640
Epoch 17/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0243 -
accuracy: 0.9951 - val_loss: 0.5023 - val_accuracy: 0.8716
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0196 -
accuracy: 0.9974 - val_loss: 0.5226 - val_accuracy: 0.8712
Epoch 19/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0122 -
accuracy: 0.9995 - val_loss: 0.6121 - val_accuracy: 0.8613
Epoch 20/20
```

```
30/30 [==============================] - 1s 22ms/step - loss: 0.0154 -
accuracy: 0.9981 - val_loss: 0.6273 - val_accuracy: 0.8606
```

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

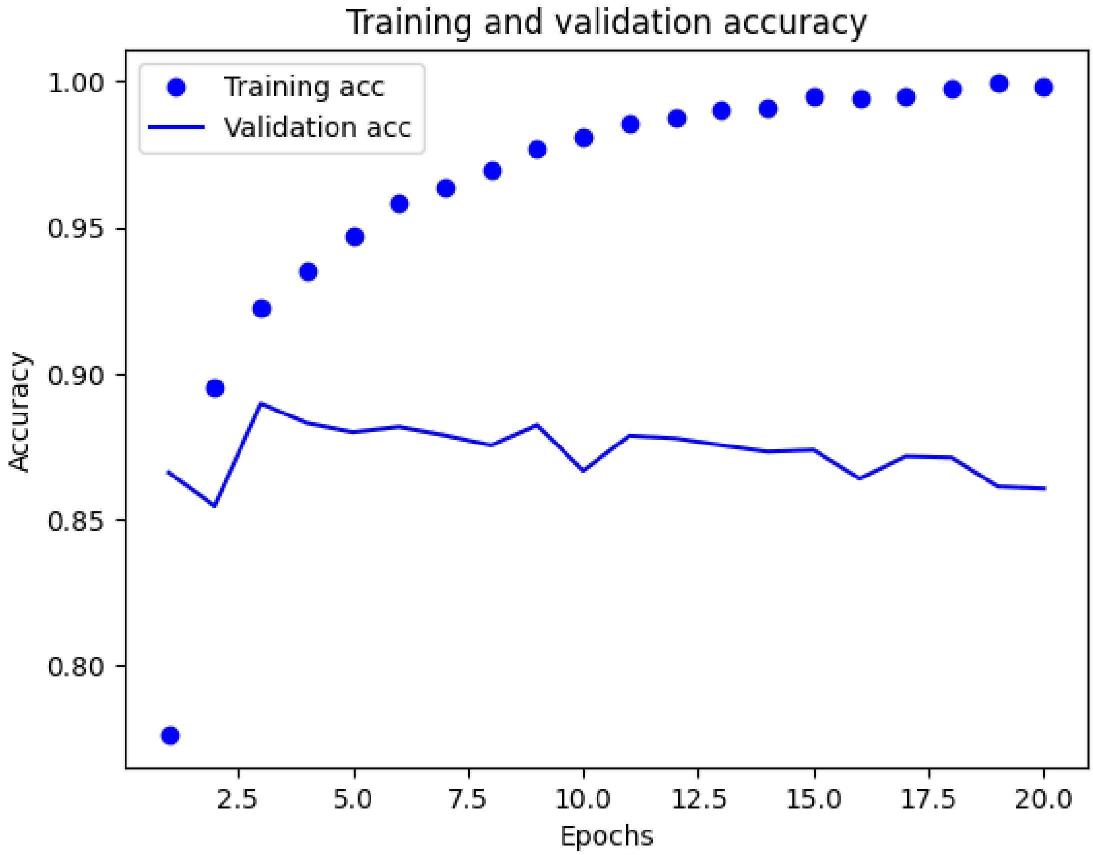
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



```
plt.clf()
acc = history_dict["accuracy"]
```

```
val_acc = history_dict["val_accuracy"]
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("Accuracy")
plt.legend()
plt.show()
```



```
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 7s 33ms/step - loss: 0.5148 -
```

```
accuracy: 0.7978
Epoch 2/4
49/49 [==============================] - 1s 28ms/step - loss: 0.3099 -
accuracy: 0.8937
Epoch 3/4
49/49 [==============================] - 1s 14ms/step - loss: 0.2377 -
accuracy: 0.9145
Epoch 4/4
49/49 [==============================] - 1s 13ms/step - loss: 0.2009 -
accuracy: 0.9284
782/782 [==============================] - 2s 3ms/step - loss: 0.2784
- accuracy: 0.8891
```

```
results
```

```
[0.27842622995376587, 0.8891199827194214]
```

```
model.predict(x_test)
```

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

```
array([[0.27425697],
       [0.9989247 ],
       [0.90748364],
       ...,
       [0.0914925 ],
       [0.07871728],
       [0.6139318 ]], dtype=float32)
```

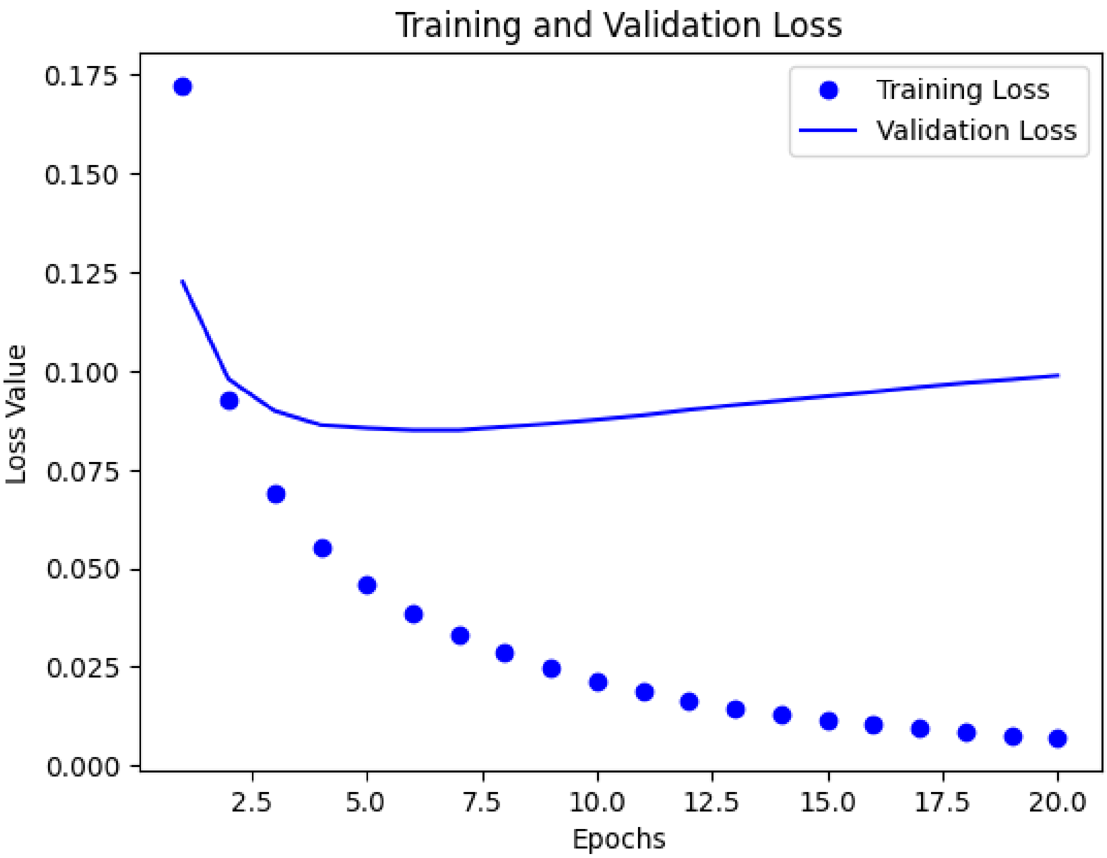
```
#we used 1 hidden layer 16 nodes

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers
```

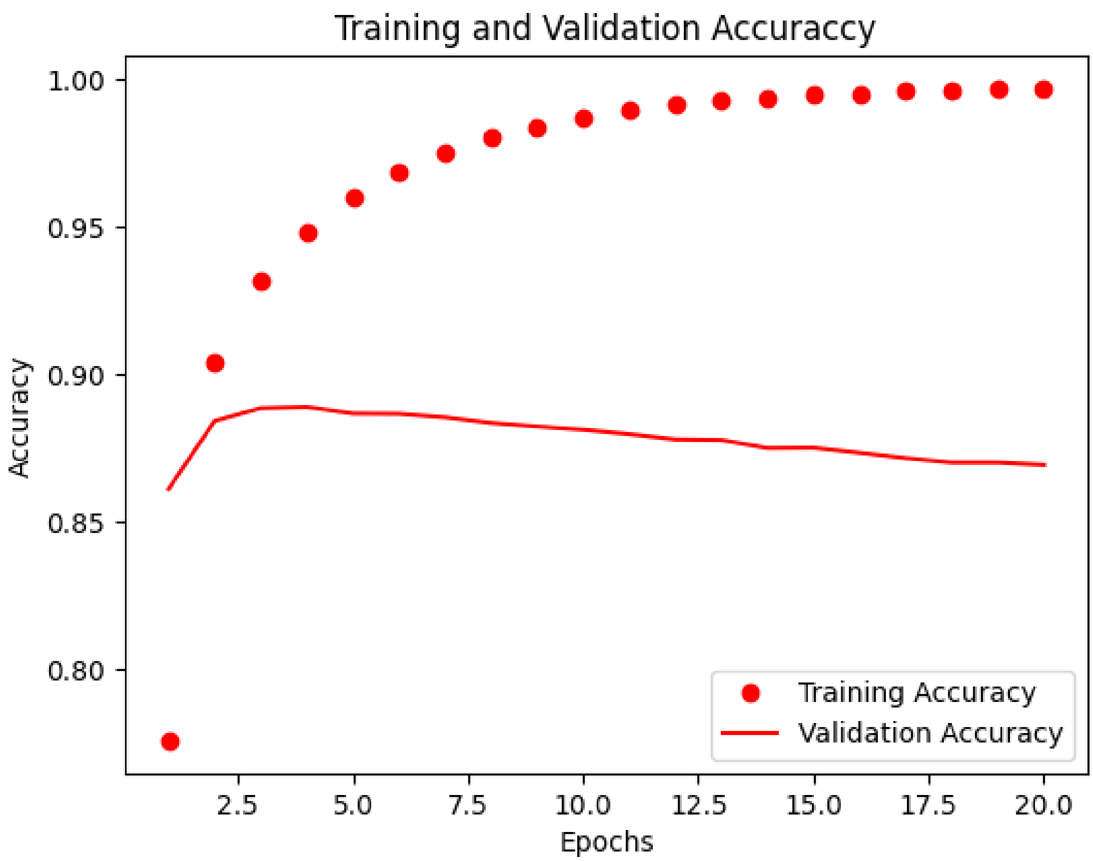
```
model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()

# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/20
30/30 [==============================] - 7s 68ms/step - loss: 0.1723 -
binary_accuracy: 0.7760 - val_loss: 0.1225 - val_binary_accuracy:
0.8612
Epoch 2/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0926 -
binary_accuracy: 0.9043 - val_loss: 0.0979 - val_binary_accuracy:
0.8842
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0691 -
binary_accuracy: 0.9319 - val_loss: 0.0898 - val_binary_accuracy:
0.8885
Epoch 4/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0552 -
binary_accuracy: 0.9480 - val_loss: 0.0863 - val_binary_accuracy:
0.8889
Epoch 5/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0458 -
binary_accuracy: 0.9600 - val_loss: 0.0855 - val_binary_accuracy:
0.8867
Epoch 6/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0387 -
binary_accuracy: 0.9686 - val_loss: 0.0850 - val_binary_accuracy:
0.8866
Epoch 7/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0331 -
binary_accuracy: 0.9747 - val_loss: 0.0849 - val_binary_accuracy:
0.8854
Epoch 8/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0285 -
```

```
binary_accuracy: 0.9803 - val_loss: 0.0858 - val_binary_accuracy:
0.8835
Epoch 9/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0248 -
binary_accuracy: 0.9838 - val_loss: 0.0866 - val_binary_accuracy:
0.8823
Epoch 10/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0215 -
binary_accuracy: 0.9869 - val_loss: 0.0876 - val_binary_accuracy:
0.8812
Epoch 11/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0188 -
binary_accuracy: 0.9892 - val_loss: 0.0887 - val_binary_accuracy:
0.8797
Epoch 12/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0166 -
binary_accuracy: 0.9913 - val_loss: 0.0901 - val_binary_accuracy:
0.8779
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0146 -
binary_accuracy: 0.9927 - val_loss: 0.0913 - val_binary_accuracy:
0.8777
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0130 -
binary_accuracy: 0.9935 - val_loss: 0.0925 - val_binary_accuracy:
0.8750
Epoch 15/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0116 -
binary_accuracy: 0.9947 - val_loss: 0.0936 - val_binary_accuracy:
0.8751
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0104 -
binary_accuracy: 0.9950 - val_loss: 0.0947 - val_binary_accuracy:
0.8734
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0093 -
binary_accuracy: 0.9958 - val_loss: 0.0959 - val_binary_accuracy:
0.8716
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0085 -
binary_accuracy: 0.9963 - val_loss: 0.0969 - val_binary_accuracy:
0.8701
Epoch 19/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0077 -
binary_accuracy: 0.9965 - val_loss: 0.0978 - val_binary_accuracy:
0.8701
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0071 -
```

```
binary_accuracy: 0.9967 - val_loss: 0.0988 - val_binary_accuracy:
0.8693
```





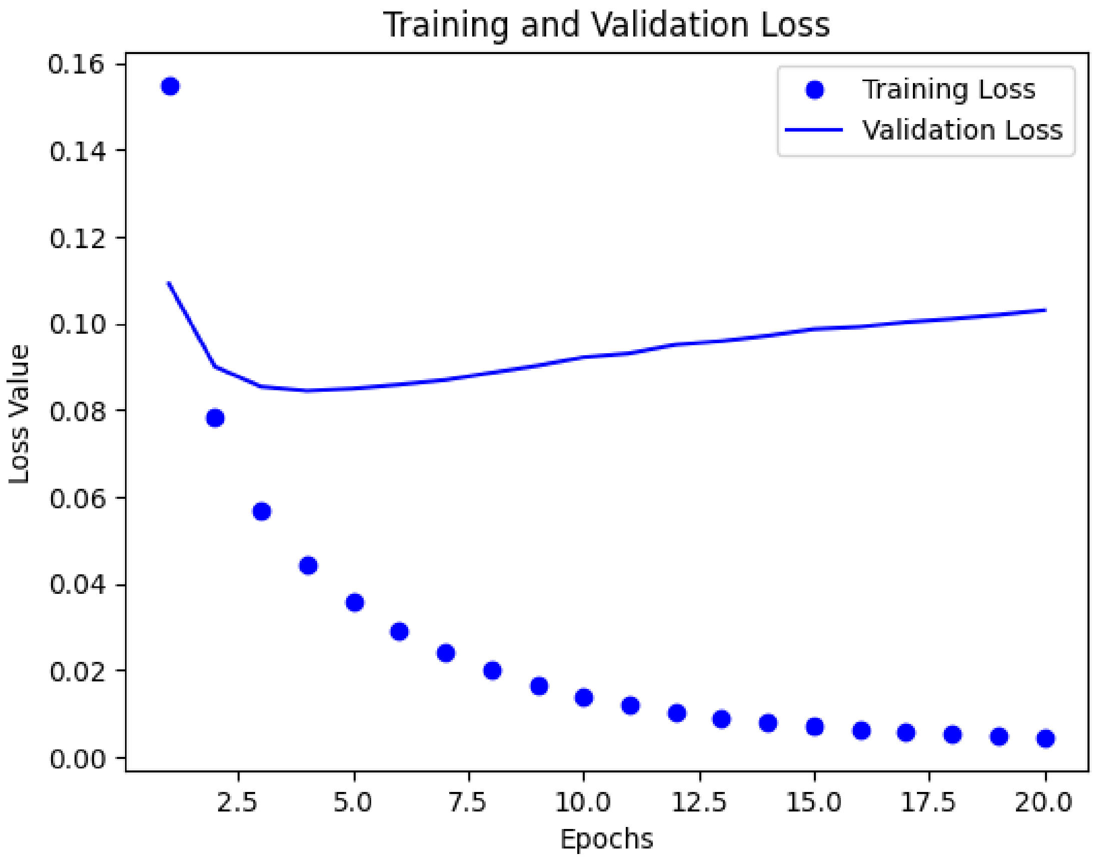
```
Epoch 1/4
49/49 [==============================] - 2s 15ms/step - loss: 0.1523 -
accuracy: 0.8216
Epoch 2/4
49/49 [==============================] - 1s 19ms/step - loss: 0.0842 -
accuracy: 0.9066
Epoch 3/4
49/49 [==============================] - 1s 17ms/step - loss: 0.0647 -
accuracy: 0.9294
Epoch 4/4
49/49 [==============================] - 1s 16ms/step - loss: 0.0534 -
accuracy: 0.9436
782/782 [==============================] - 3s 3ms/step - loss: 0.0865
- accuracy: 0.8858

[0.08645174652338028, 0.8858399987220764]

#We used 1 hidden layer 32 nodes

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
```

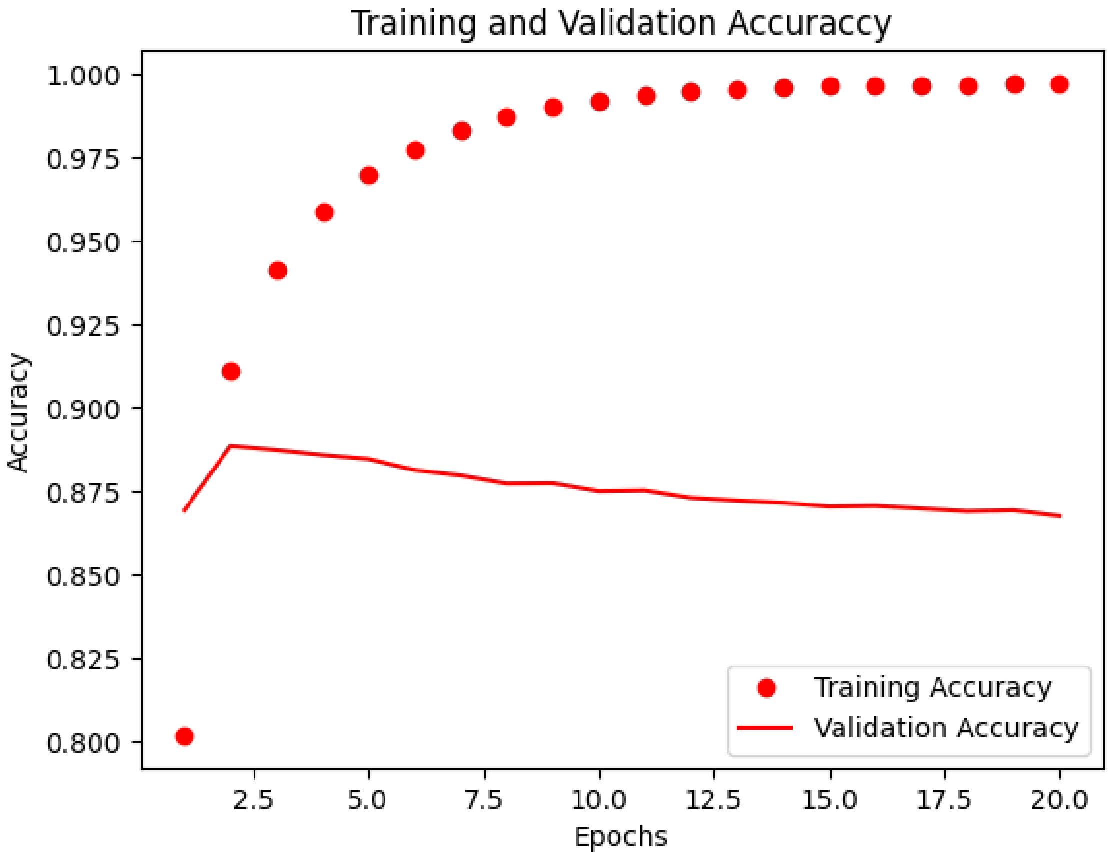
```
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```

```
Epoch 1/20
30/30 [==============================] - 8s 78ms/step - loss: 0.1549 -
binary_accuracy: 0.8019 - val_loss: 0.1091 - val_binary_accuracy:
0.8693
Epoch 2/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0783 -
binary_accuracy: 0.9114 - val_loss: 0.0900 - val_binary_accuracy:
0.8885
Epoch 3/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0569 -
binary_accuracy: 0.9416 - val_loss: 0.0854 - val_binary_accuracy:
0.8873
Epoch 4/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0442 -
```

```
binary_accuracy: 0.9587 - val_loss: 0.0844 - val_binary_accuracy:
0.8858
Epoch 5/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0357 -
binary_accuracy: 0.9699 - val_loss: 0.0850 - val_binary_accuracy:
0.8846
Epoch 6/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0292 -
binary_accuracy: 0.9775 - val_loss: 0.0859 - val_binary_accuracy:
0.8813
Epoch 7/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0240 -
binary_accuracy: 0.9833 - val_loss: 0.0869 - val_binary_accuracy:
0.8797
Epoch 8/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0200 -
binary_accuracy: 0.9870 - val_loss: 0.0886 - val_binary_accuracy:
0.8773
Epoch 9/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0165 -
binary_accuracy: 0.9899 - val_loss: 0.0902 - val_binary_accuracy:
0.8774
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0140 -
binary_accuracy: 0.9920 - val_loss: 0.0922 - val_binary_accuracy:
0.8750
Epoch 11/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0119 -
binary_accuracy: 0.9938 - val_loss: 0.0931 - val_binary_accuracy:
0.8752
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0103 -
binary_accuracy: 0.9947 - val_loss: 0.0951 - val_binary_accuracy:
0.8730
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0090 -
binary_accuracy: 0.9953 - val_loss: 0.0959 - val_binary_accuracy:
0.8722
Epoch 14/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0079 -
binary_accuracy: 0.9958 - val_loss: 0.0971 - val_binary_accuracy:
0.8715
Epoch 15/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0070 -
binary_accuracy: 0.9963 - val_loss: 0.0986 - val_binary_accuracy:
0.8704
Epoch 16/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0064 -
binary_accuracy: 0.9966 - val_loss: 0.0992 - val_binary_accuracy:
```

```
0.8706
Epoch 17/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0058 -
binary_accuracy: 0.9967 - val_loss: 0.1003 - val_binary_accuracy:
0.8698
Epoch 18/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0054 -
binary_accuracy: 0.9968 - val_loss: 0.1010 - val_binary_accuracy:
0.8690
Epoch 19/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0049 -
binary_accuracy: 0.9970 - val_loss: 0.1019 - val_binary_accuracy:
0.8693
Epoch 20/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0045 -
binary_accuracy: 0.9972 - val_loss: 0.1030 - val_binary_accuracy:
0.8676
```



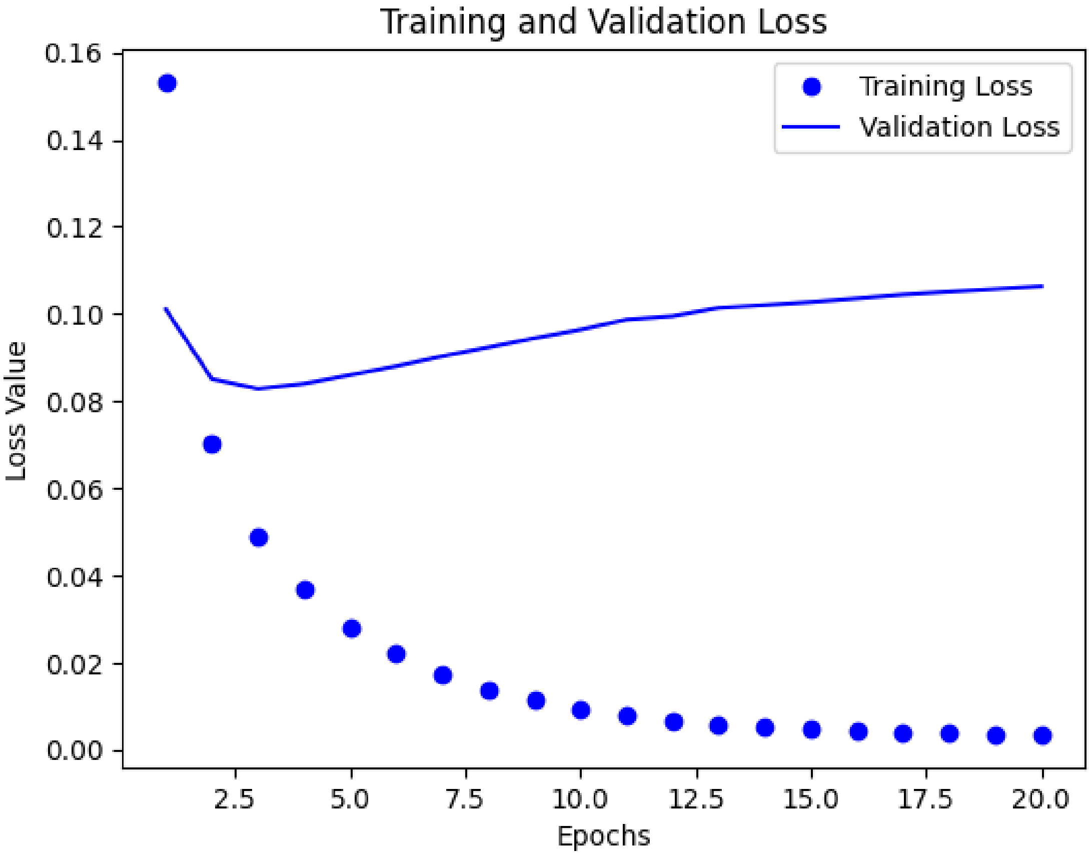


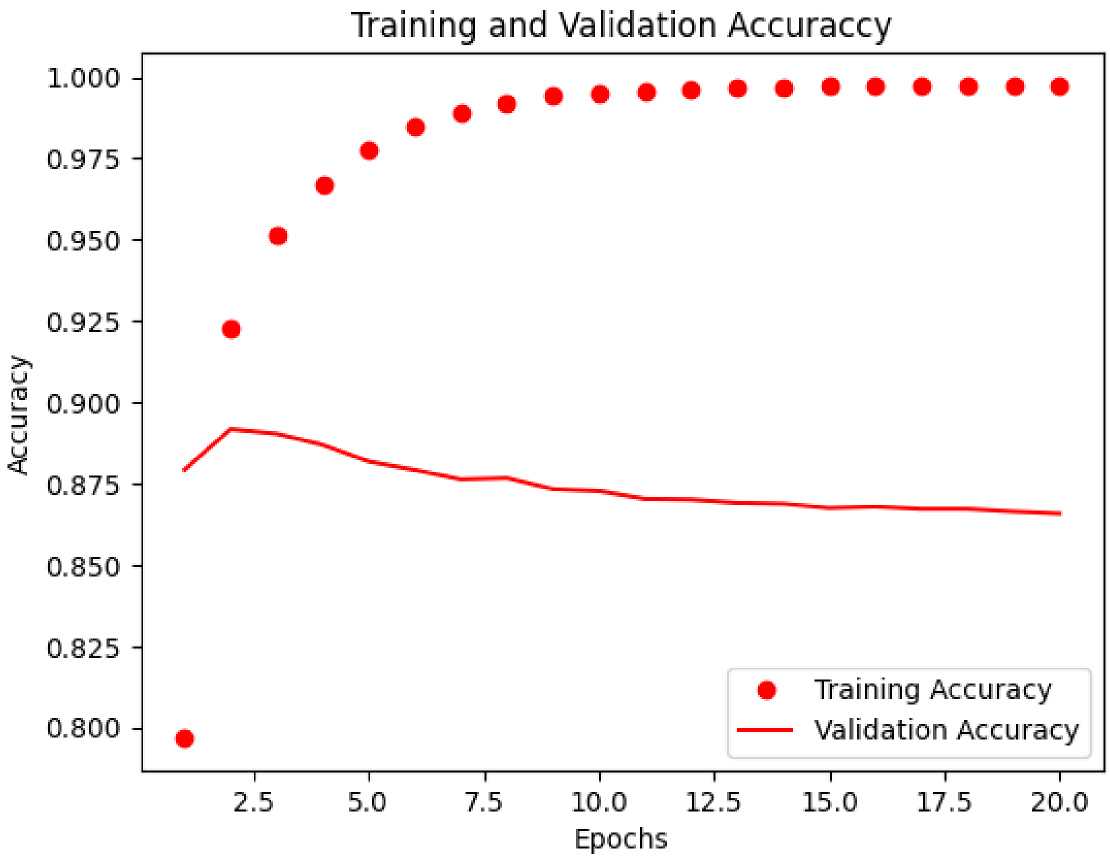
```
Epoch 1/4
49/49 [==============================] - 1s 14ms/step - loss: 0.1260 -
accuracy: 0.8423
Epoch 2/4
49/49 [==============================] - 1s 15ms/step - loss: 0.0664 -
accuracy: 0.9222
Epoch 3/4
49/49 [==============================] - 1s 13ms/step - loss: 0.0508 -
accuracy: 0.9435
Epoch 4/4
49/49 [==============================] - 1s 13ms/step - loss: 0.0412 -
accuracy: 0.9576
782/782 [==============================] - 2s 3ms/step - loss: 0.0889
- accuracy: 0.8795

[0.08893857151269913, 0.8795199990272522]

#we take 1 hiddenlayer 64 nodes

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
```

```
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results

Epoch 1/20
30/30 [==============================] - 7s 72ms/step - loss: 0.1532 -
binary_accuracy: 0.7969 - val_loss: 0.1010 - val_binary_accuracy:
0.8793
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0701 -
binary_accuracy: 0.9229 - val_loss: 0.0850 - val_binary_accuracy:
0.8917
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0489 -
binary_accuracy: 0.9511 - val_loss: 0.0828 - val_binary_accuracy:
0.8903
Epoch 4/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0369 -
```

```
binary_accuracy: 0.9669 - val_loss: 0.0840 - val_binary_accuracy:
0.8869
Epoch 5/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0280 -
binary_accuracy: 0.9777 - val_loss: 0.0860 - val_binary_accuracy:
0.8818
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0221 -
binary_accuracy: 0.9846 - val_loss: 0.0880 - val_binary_accuracy:
0.8792
Epoch 7/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0174 -
binary_accuracy: 0.9891 - val_loss: 0.0903 - val_binary_accuracy:
0.8763
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0137 -
binary_accuracy: 0.9919 - val_loss: 0.0923 - val_binary_accuracy:
0.8767
Epoch 9/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0112 -
binary_accuracy: 0.9941 - val_loss: 0.0944 - val_binary_accuracy:
0.8733
Epoch 10/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0093 -
binary_accuracy: 0.9948 - val_loss: 0.0964 - val_binary_accuracy:
0.8727
Epoch 11/20
30/30 [==============================] - 1s 38ms/step - loss: 0.0078 -
binary_accuracy: 0.9958 - val_loss: 0.0987 - val_binary_accuracy:
0.8703
Epoch 12/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0067 -
binary_accuracy: 0.9964 - val_loss: 0.0995 - val_binary_accuracy:
0.8701
Epoch 13/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0058 -
binary_accuracy: 0.9966 - val_loss: 0.1014 - val_binary_accuracy:
0.8691
Epoch 14/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0053 -
binary_accuracy: 0.9969 - val_loss: 0.1020 - val_binary_accuracy:
0.8688
Epoch 15/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0047 -
binary_accuracy: 0.9971 - val_loss: 0.1027 - val_binary_accuracy:
0.8675
Epoch 16/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0043 -
binary_accuracy: 0.9973 - val_loss: 0.1036 - val_binary_accuracy:
```

```
0.8679
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0040 -
binary_accuracy: 0.9973 - val_loss: 0.1044 - val_binary_accuracy:
0.8672
Epoch 18/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0038 -
binary_accuracy: 0.9973 - val_loss: 0.1051 - val_binary_accuracy:
0.8672
Epoch 19/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0036 -
binary_accuracy: 0.9974 - val_loss: 0.1057 - val_binary_accuracy:
0.8664
Epoch 20/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0034 -
binary_accuracy: 0.9974 - val_loss: 0.1063 - val_binary_accuracy:
0.8658
```



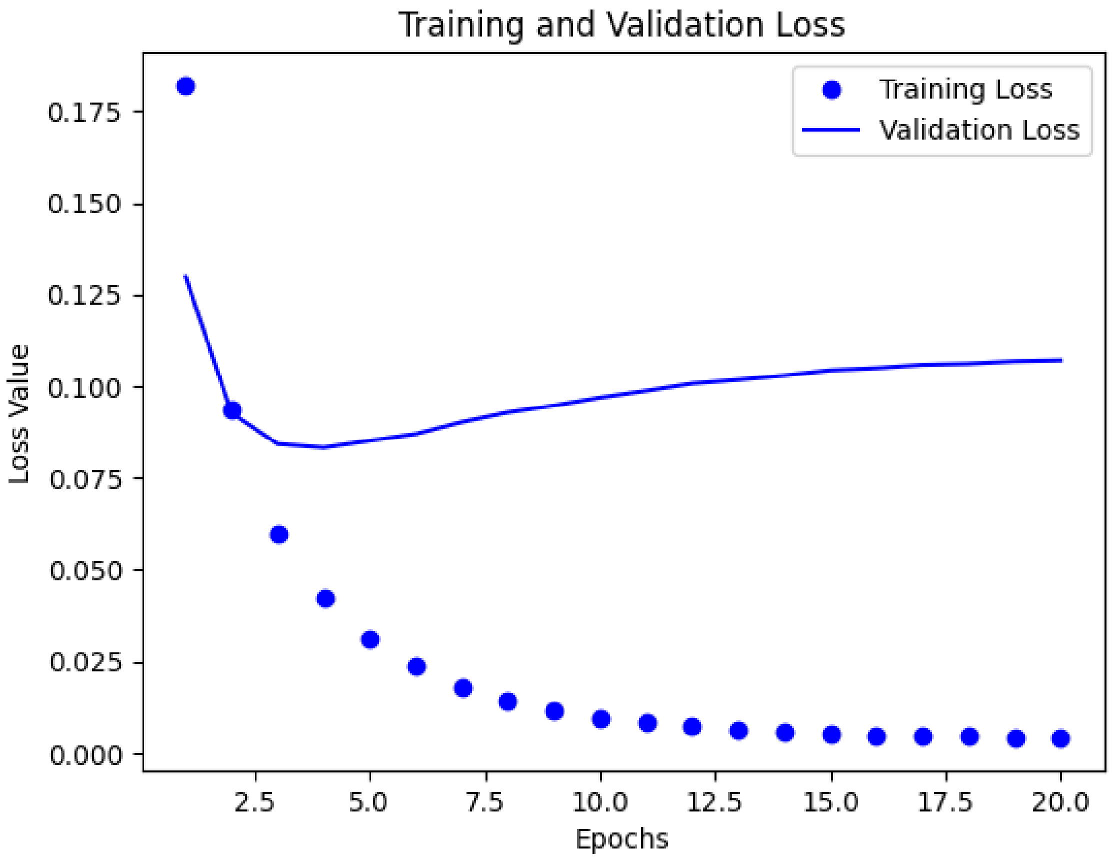


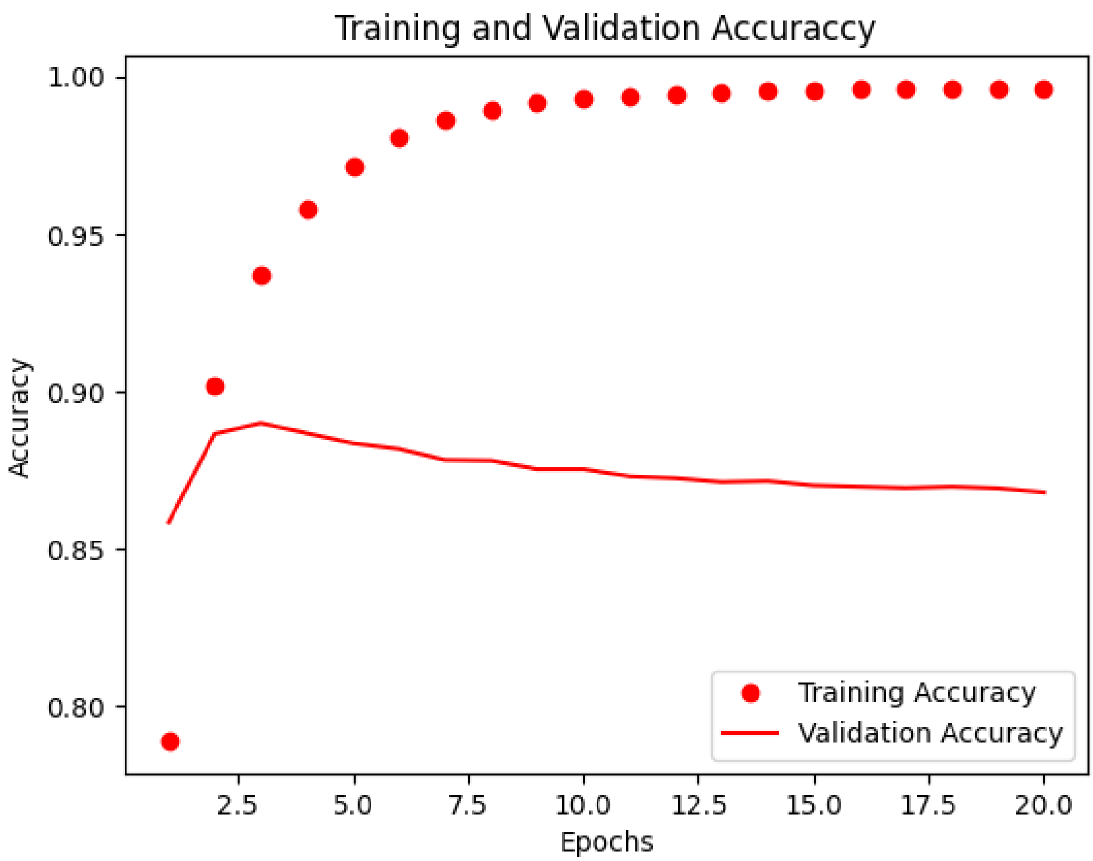
```
Epoch 1/4
49/49 [==============================] - 2s 19ms/step - loss: 0.1202 -
accuracy: 0.8481
Epoch 2/4
49/49 [==============================] - 1s 22ms/step - loss: 0.0613 -
accuracy: 0.9268
Epoch 3/4
49/49 [==============================] - 1s 14ms/step - loss: 0.0457 -
accuracy: 0.9484
Epoch 4/4
49/49 [==============================] - 1s 13ms/step - loss: 0.0361 -
accuracy: 0.9620
782/782 [==============================] - 3s 3ms/step - loss: 0.0938
- accuracy: 0.8730

[0.09382835775613785, 0.8730400204658508]

#We consider hidden layer 2 and nodes 16

from keras import models
from tensorflow.keras import layers

model = models.Sequential()
```

```
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()

# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)

results

Epoch 1/20
30/30 [==============================] - 8s 66ms/step - loss: 0.1821 -
binary_accuracy: 0.7889 - val_loss: 0.1298 - val_binary_accuracy:
0.8583
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0936 -
binary_accuracy: 0.9017 - val_loss: 0.0926 - val_binary_accuracy:
0.8864
Epoch 3/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0599 -
binary_accuracy: 0.9371 - val_loss: 0.0842 - val_binary_accuracy:
0.8898
Epoch 4/20
```

```
30/30 [==============================] - 1s 22ms/step - loss: 0.0423 -
binary_accuracy: 0.9581 - val_loss: 0.0832 - val_binary_accuracy:
0.8866
Epoch 5/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0314 -
binary_accuracy: 0.9716 - val_loss: 0.0851 - val_binary_accuracy:
0.8834
Epoch 6/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0236 -
binary_accuracy: 0.9810 - val_loss: 0.0870 - val_binary_accuracy:
0.8816
Epoch 7/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0181 -
binary_accuracy: 0.9862 - val_loss: 0.0902 - val_binary_accuracy:
0.8781
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0142 -
binary_accuracy: 0.9896 - val_loss: 0.0928 - val_binary_accuracy:
0.8779
Epoch 9/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0117 -
binary_accuracy: 0.9916 - val_loss: 0.0947 - val_binary_accuracy:
0.8752
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0096 -
binary_accuracy: 0.9929 - val_loss: 0.0968 - val_binary_accuracy:
0.8752
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0082 -
binary_accuracy: 0.9939 - val_loss: 0.0987 - val_binary_accuracy:
0.8729
Epoch 12/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0073 -
binary_accuracy: 0.9943 - val_loss: 0.1007 - val_binary_accuracy:
0.8723
Epoch 13/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0064 -
binary_accuracy: 0.9949 - val_loss: 0.1017 - val_binary_accuracy:
0.8711
Epoch 14/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0058 -
binary_accuracy: 0.9953 - val_loss: 0.1029 - val_binary_accuracy:
0.8714
Epoch 15/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0053 -
binary_accuracy: 0.9956 - val_loss: 0.1043 - val_binary_accuracy:
0.8700
Epoch 16/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0049 -
```

```
binary_accuracy: 0.9959 - val_loss: 0.1049 - val_binary_accuracy:
0.8696
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0047 -
binary_accuracy: 0.9960 - val_loss: 0.1058 - val_binary_accuracy:
0.8692
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0044 -
binary_accuracy: 0.9961 - val_loss: 0.1061 - val_binary_accuracy:
0.8696
Epoch 19/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0043 -
binary_accuracy: 0.9962 - val_loss: 0.1067 - val_binary_accuracy:
0.8691
Epoch 20/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0042 -
binary_accuracy: 0.9962 - val_loss: 0.1070 - val_binary_accuracy:
0.8678
```

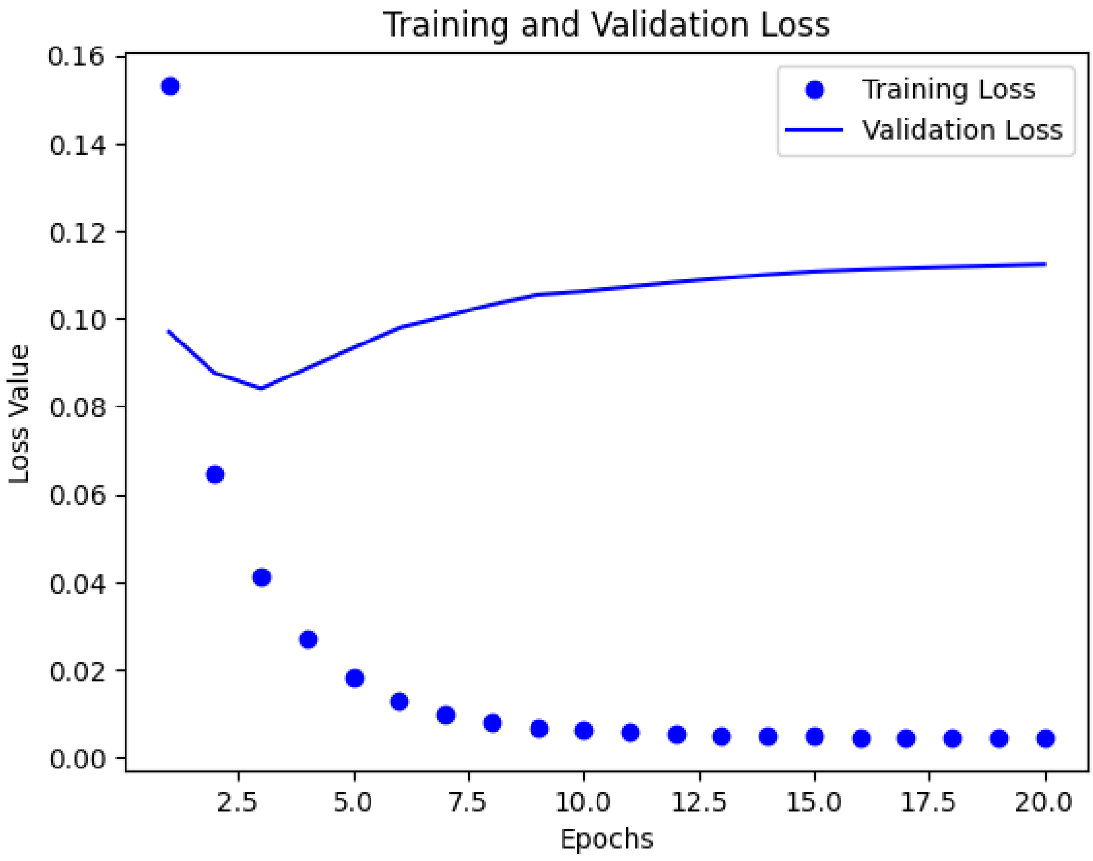


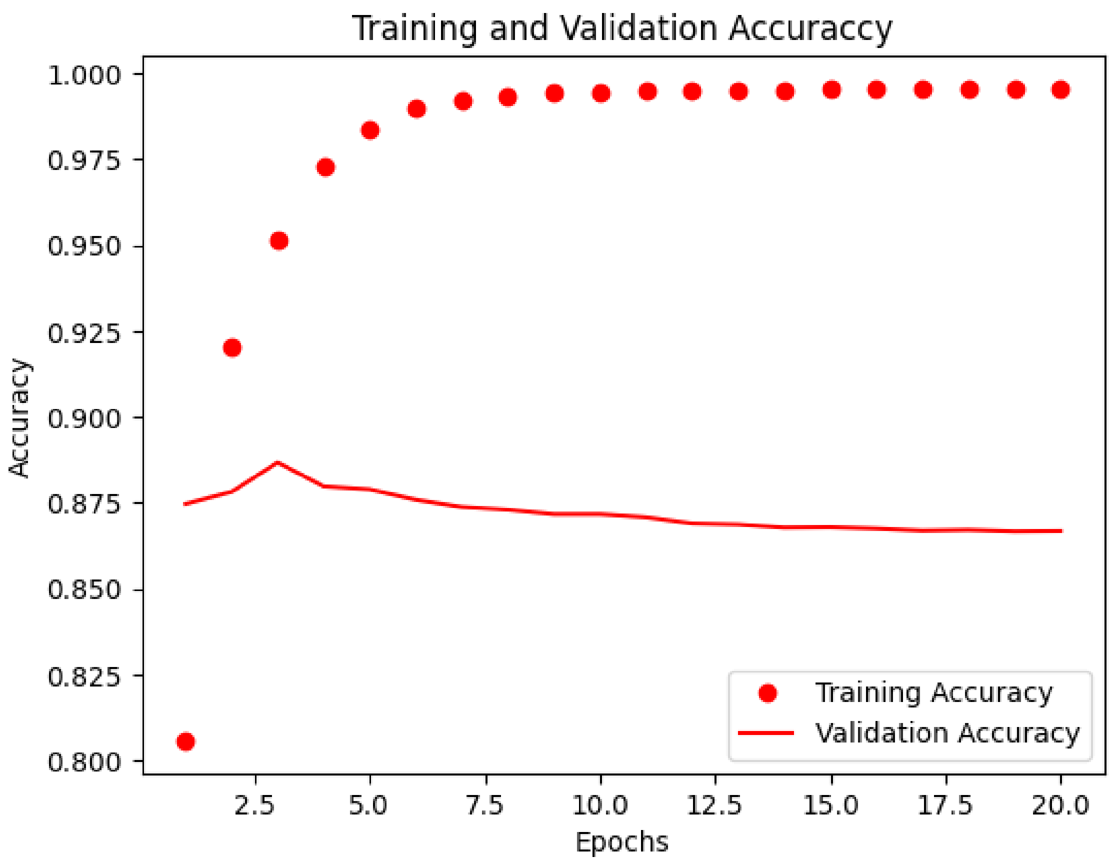


```
Epoch 1/4
49/49 [==============================] - 3s 20ms/step - loss: 0.1443 -
accuracy: 0.8192
Epoch 2/4
49/49 [==============================] - 1s 24ms/step - loss: 0.0669 -
accuracy: 0.9183
Epoch 3/4
49/49 [==============================] - 1s 22ms/step - loss: 0.0476 -
accuracy: 0.9452
Epoch 4/4
49/49 [==============================] - 1s 20ms/step - loss: 0.0368 -
accuracy: 0.9609
782/782 [==============================] - 3s 3ms/step - loss: 0.0928
- accuracy: 0.8750

[0.09282186627388, 0.8749600052833557]

#We consider hidden layer 2 32 nodes
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                     batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results

Epoch 1/20
30/30 [==============================] - 4s 65ms/step - loss: 0.1532 -
binary_accuracy: 0.8059 - val_loss: 0.0970 - val_binary_accuracy:
0.8747
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0648 -
binary_accuracy: 0.9202 - val_loss: 0.0876 - val_binary_accuracy:
0.8782
Epoch 3/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0412 -
binary_accuracy: 0.9515 - val_loss: 0.0840 - val_binary_accuracy:
0.8867
Epoch 4/20
```

```
30/30 [==============================] - 1s 32ms/step - loss: 0.0271 -
binary_accuracy: 0.9729 - val_loss: 0.0887 - val_binary_accuracy:
0.8798
Epoch 5/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0181 -
binary_accuracy: 0.9837 - val_loss: 0.0933 - val_binary_accuracy:
0.8789
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0129 -
binary_accuracy: 0.9896 - val_loss: 0.0979 - val_binary_accuracy:
0.8759
Epoch 7/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0097 -
binary_accuracy: 0.9920 - val_loss: 0.1005 - val_binary_accuracy:
0.8738
Epoch 8/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0079 -
binary_accuracy: 0.9931 - val_loss: 0.1032 - val_binary_accuracy:
0.8730
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0068 -
binary_accuracy: 0.9941 - val_loss: 0.1054 - val_binary_accuracy:
0.8717
Epoch 10/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0061 -
binary_accuracy: 0.9944 - val_loss: 0.1062 - val_binary_accuracy:
0.8717
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0057 -
binary_accuracy: 0.9947 - val_loss: 0.1072 - val_binary_accuracy:
0.8708
Epoch 12/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0054 -
binary_accuracy: 0.9949 - val_loss: 0.1083 - val_binary_accuracy:
0.8690
Epoch 13/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0051 -
binary_accuracy: 0.9951 - val_loss: 0.1092 - val_binary_accuracy:
0.8687
Epoch 14/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0050 -
binary_accuracy: 0.9952 - val_loss: 0.1100 - val_binary_accuracy:
0.8678
Epoch 15/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0048 -
binary_accuracy: 0.9954 - val_loss: 0.1107 - val_binary_accuracy:
0.8679
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0047 -
```

```
binary_accuracy: 0.9955 - val_loss: 0.1112 - val_binary_accuracy:
0.8675
Epoch 17/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0046 -
binary_accuracy: 0.9955 - val_loss: 0.1115 - val_binary_accuracy:
0.8669
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0046 -
binary_accuracy: 0.9955 - val_loss: 0.1118 - val_binary_accuracy:
0.8671
Epoch 19/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0046 -
binary_accuracy: 0.9955 - val_loss: 0.1121 - val_binary_accuracy:
0.8667
Epoch 20/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0046 -
binary_accuracy: 0.9955 - val_loss: 0.1124 - val_binary_accuracy:
0.8668
```





```
Epoch 1/2
49/49 [==============================] - 2s 14ms/step - loss: 0.1233 -
accuracy: 0.8360
Epoch 2/2
49/49 [==============================] - 1s 14ms/step - loss: 0.0560 -
accuracy: 0.9284
782/782 [==============================] - 2s 3ms/step - loss: 0.0878
- accuracy: 0.8814

[0.08784940093755722, 0.8814399838447571]
```

```
#We consider hidden layer 2 64 nodes
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])
```

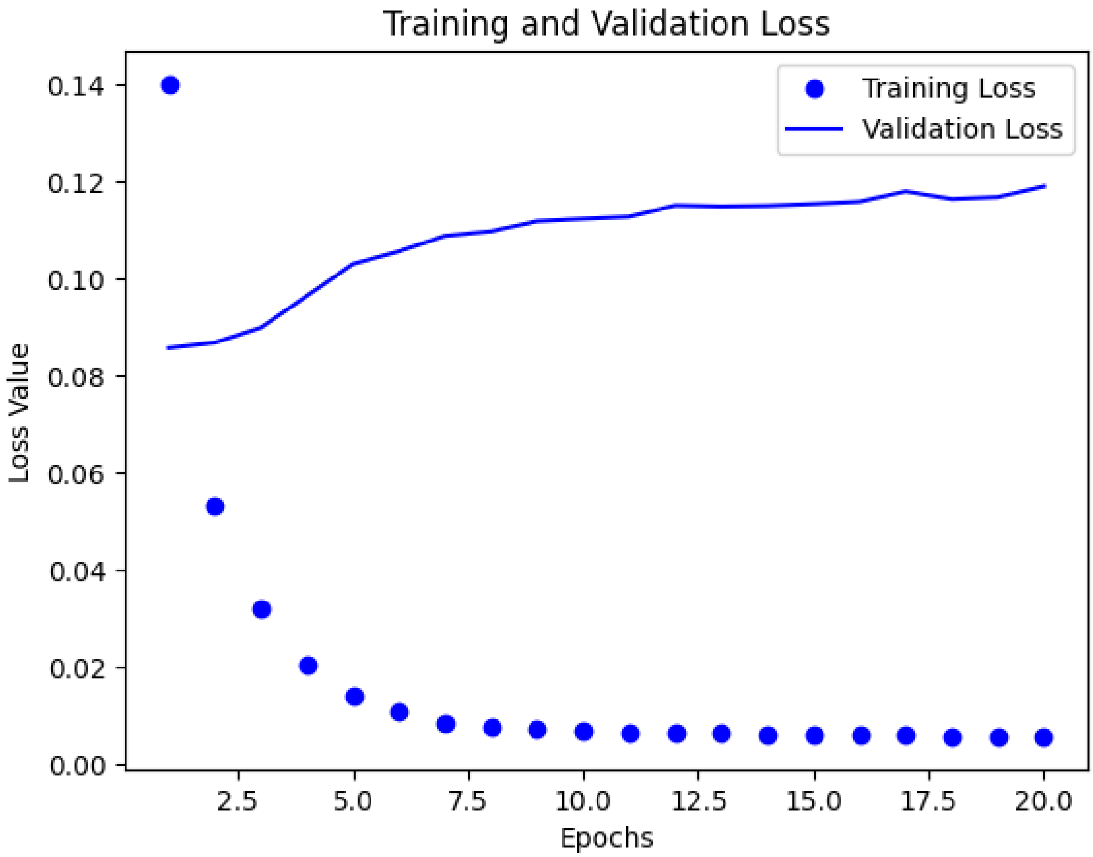
```
from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

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

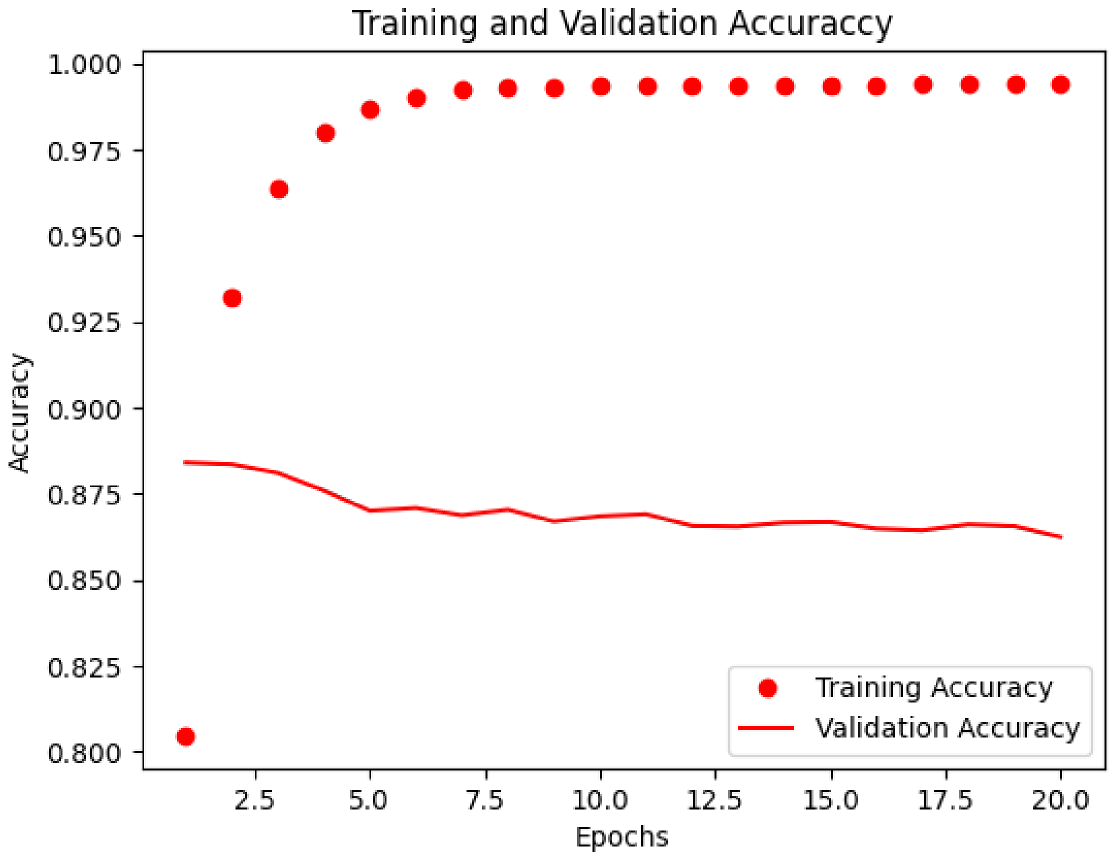
```
Epoch 1/20
30/30 [==============================] - 4s 64ms/step - loss: 0.1400 -
binary_accuracy: 0.8047 - val_loss: 0.0857 - val_binary_accuracy:
0.8840
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0532 -
binary_accuracy: 0.9321 - val_loss: 0.0868 - val_binary_accuracy:
0.8835
Epoch 3/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0321 -
binary_accuracy: 0.9638 - val_loss: 0.0898 - val_binary_accuracy:
0.8810
Epoch 4/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0206 -
binary_accuracy: 0.9799 - val_loss: 0.0964 - val_binary_accuracy:
0.8759
Epoch 5/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0141 -
binary_accuracy: 0.9869 - val_loss: 0.1029 - val_binary_accuracy:
0.8700
Epoch 6/20
```

```
30/30 [==============================] - 1s 23ms/step - loss: 0.0108 -
binary_accuracy: 0.9905 - val_loss: 0.1056 - val_binary_accuracy:
0.8708
Epoch 7/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0086 -
binary_accuracy: 0.9923 - val_loss: 0.1087 - val_binary_accuracy:
0.8687
Epoch 8/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0079 -
binary_accuracy: 0.9928 - val_loss: 0.1097 - val_binary_accuracy:
0.8703
Epoch 9/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0073 -
binary_accuracy: 0.9931 - val_loss: 0.1118 - val_binary_accuracy:
0.8669
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0068 -
binary_accuracy: 0.9934 - val_loss: 0.1123 - val_binary_accuracy:
0.8684
Epoch 11/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0066 -
binary_accuracy: 0.9935 - val_loss: 0.1127 - val_binary_accuracy:
0.8690
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0066 -
binary_accuracy: 0.9936 - val_loss: 0.1150 - val_binary_accuracy:
0.8656
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0064 -
binary_accuracy: 0.9937 - val_loss: 0.1148 - val_binary_accuracy:
0.8654
Epoch 14/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0063 -
binary_accuracy: 0.9938 - val_loss: 0.1149 - val_binary_accuracy:
0.8665
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0063 -
binary_accuracy: 0.9938 - val_loss: 0.1153 - val_binary_accuracy:
0.8667
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0062 -
binary_accuracy: 0.9939 - val_loss: 0.1158 - val_binary_accuracy:
0.8648
Epoch 17/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0061 -
binary_accuracy: 0.9939 - val_loss: 0.1179 - val_binary_accuracy:
0.8643
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0059 -
```

```
binary_accuracy: 0.9941 - val_loss: 0.1163 - val_binary_accuracy:
0.8660
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0059 -
binary_accuracy: 0.9941 - val_loss: 0.1167 - val_binary_accuracy:
0.8655
Epoch 20/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0058 -
binary_accuracy: 0.9942 - val_loss: 0.1189 - val_binary_accuracy:
0.8624

dict_keys(['loss', 'binary_accuracy', 'val_loss',
'val_binary_accuracy'])
```

```
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```



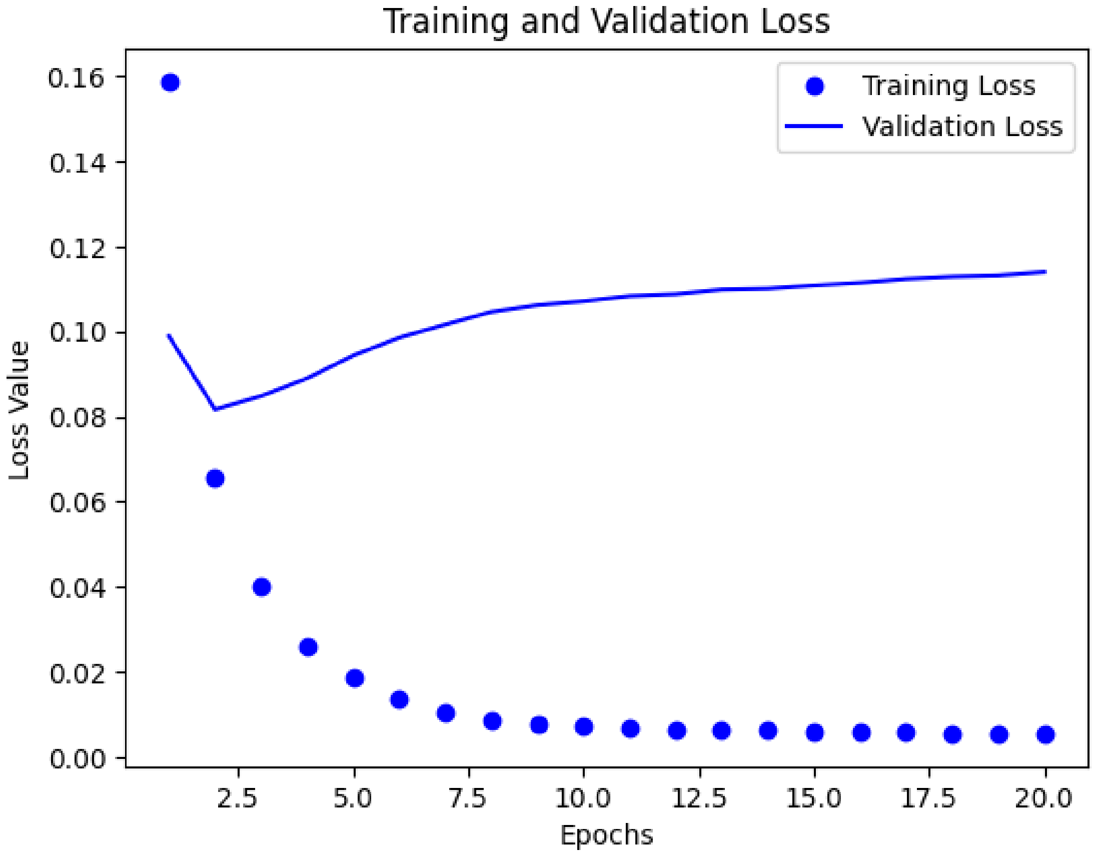


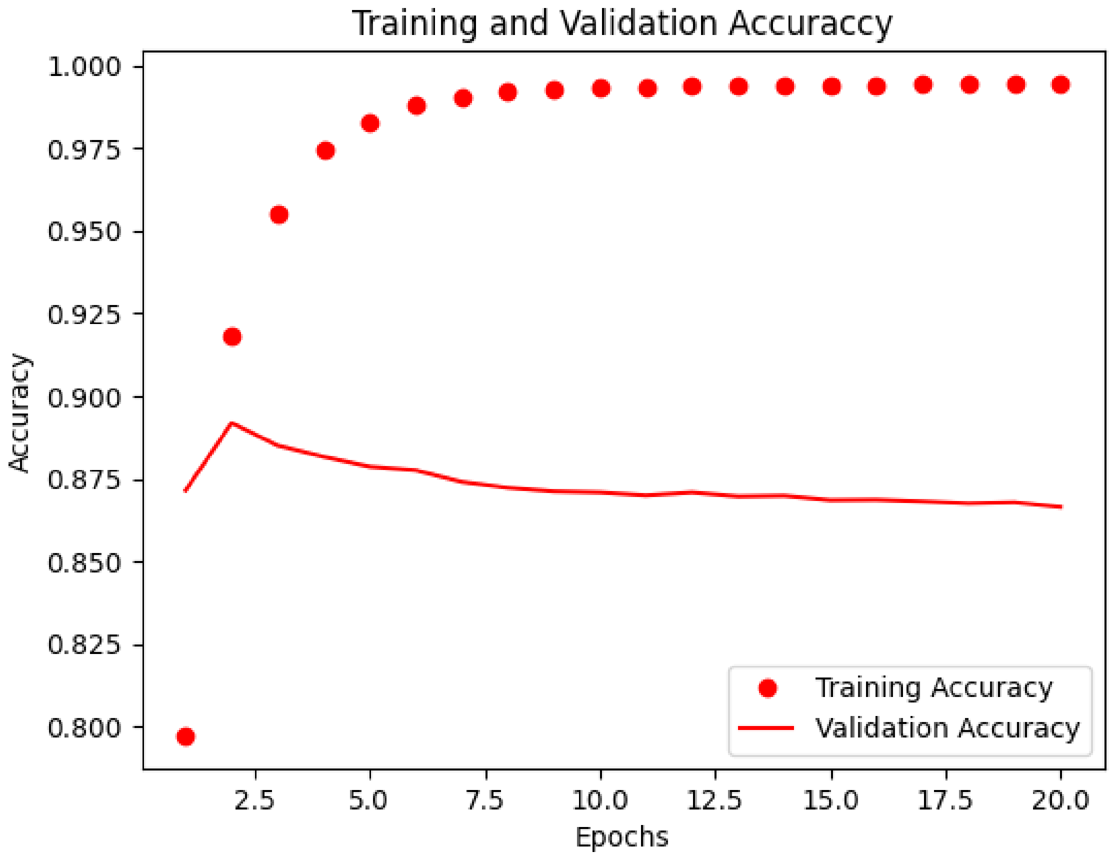
```
Epoch 1/2
49/49 [==============================] - 2s 15ms/step - loss: 0.1144 -
accuracy: 0.8438
Epoch 2/2
49/49 [==============================] - 1s 13ms/step - loss: 0.0532 -
accuracy: 0.9325
782/782 [==============================] - 2s 3ms/step - loss: 0.0911
- accuracy: 0.8786

[0.09111196547746658, 0.878600001335144]
```

```
#we consider hidden layer 3 nodes 16
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

Epoch 1/20
30/30 [==============================] - 8s 68ms/step - loss: 0.1587 -
binary_accuracy: 0.7973 - val_loss: 0.0989 - val_binary_accuracy:
0.8714
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0657 -
binary_accuracy: 0.9183 - val_loss: 0.0817 - val_binary_accuracy:
0.8918
Epoch 3/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0401 -
binary_accuracy: 0.9552 - val_loss: 0.0848 - val_binary_accuracy:
0.8850
Epoch 4/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0261 -
binary_accuracy: 0.9743 - val_loss: 0.0890 - val_binary_accuracy:
0.8816
Epoch 5/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0187 -
binary_accuracy: 0.9830 - val_loss: 0.0943 - val_binary_accuracy:
0.8786
```

```
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0135 -
binary_accuracy: 0.9879 - val_loss: 0.0985 - val_binary_accuracy:
0.8775
Epoch 7/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0106 -
binary_accuracy: 0.9907 - val_loss: 0.1016 - val_binary_accuracy:
0.8740
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0088 -
binary_accuracy: 0.9919 - val_loss: 0.1046 - val_binary_accuracy:
0.8722
Epoch 9/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0077 -
binary_accuracy: 0.9929 - val_loss: 0.1062 - val_binary_accuracy:
0.8711
Epoch 10/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0072 -
binary_accuracy: 0.9932 - val_loss: 0.1071 - val_binary_accuracy:
0.8708
Epoch 11/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0069 -
binary_accuracy: 0.9934 - val_loss: 0.1083 - val_binary_accuracy:
0.8699
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0065 -
binary_accuracy: 0.9937 - val_loss: 0.1087 - val_binary_accuracy:
0.8708
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0064 -
binary_accuracy: 0.9938 - val_loss: 0.1098 - val_binary_accuracy:
0.8696
Epoch 14/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0062 -
binary_accuracy: 0.9939 - val_loss: 0.1100 - val_binary_accuracy:
0.8698
Epoch 15/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0061 -
binary_accuracy: 0.9941 - val_loss: 0.1108 - val_binary_accuracy:
0.8685
Epoch 16/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0060 -
binary_accuracy: 0.9941 - val_loss: 0.1114 - val_binary_accuracy:
0.8686
Epoch 17/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0058 -
binary_accuracy: 0.9943 - val_loss: 0.1123 - val_binary_accuracy:
0.8681
Epoch 18/20
```

```
30/30 [==============================] - 1s 25ms/step - loss: 0.0057 -
binary_accuracy: 0.9944 - val_loss: 0.1128 - val_binary_accuracy:
0.8675
Epoch 19/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0056 -
binary_accuracy: 0.9945 - val_loss: 0.1131 - val_binary_accuracy:
0.8678
Epoch 20/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0056 -
binary_accuracy: 0.9945 - val_loss: 0.1139 - val_binary_accuracy:
0.8664

dict_keys(['loss', 'binary_accuracy', 'val_loss',
'val_binary_accuracy'])

# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


model = models.Sequential()
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(16, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=3, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```





```
Epoch 1/3
49/49 [==============================] - 3s 15ms/step - loss: 0.1396 -
accuracy: 0.8279
Epoch 2/3
49/49 [==============================] - 1s 15ms/step - loss: 0.0614 -
accuracy: 0.9235
Epoch 3/3
49/49 [==============================] - 1s 15ms/step - loss: 0.0428 -
accuracy: 0.9502
782/782 [==============================] - 2s 3ms/step - loss: 0.0913
- accuracy: 0.8774

[0.09133495390415192, 0.8773599863052368]
```

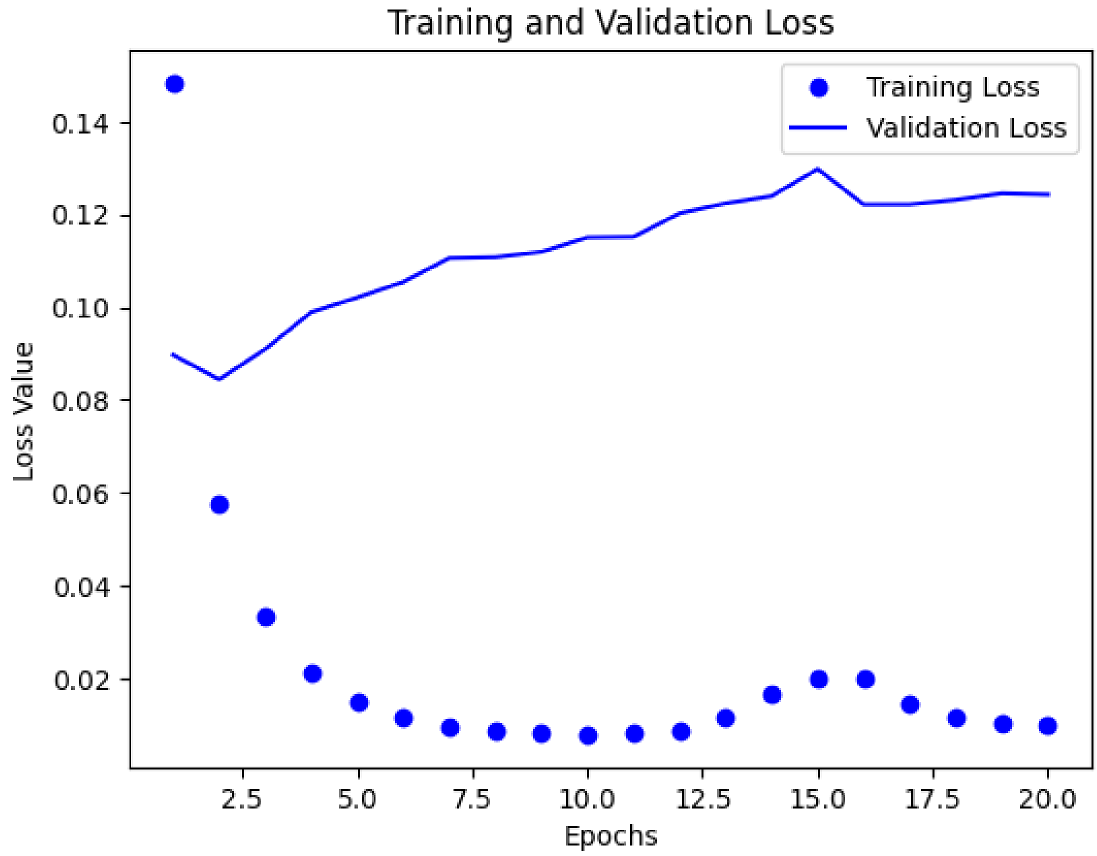
```
#Hidden layer 3 nodes 32
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

Epoch 1/20
30/30 [==============================] - 9s 94ms/step - loss: 0.1483 -
binary_accuracy: 0.8068 - val_loss: 0.0897 - val_binary_accuracy:
0.8783
Epoch 2/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0578 -
binary_accuracy: 0.9258 - val_loss: 0.0845 - val_binary_accuracy:
0.8846
Epoch 3/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0334 -
binary_accuracy: 0.9617 - val_loss: 0.0910 - val_binary_accuracy:
0.8779
Epoch 4/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0214 -
binary_accuracy: 0.9776 - val_loss: 0.0989 - val_binary_accuracy:
0.8733
Epoch 5/20
```

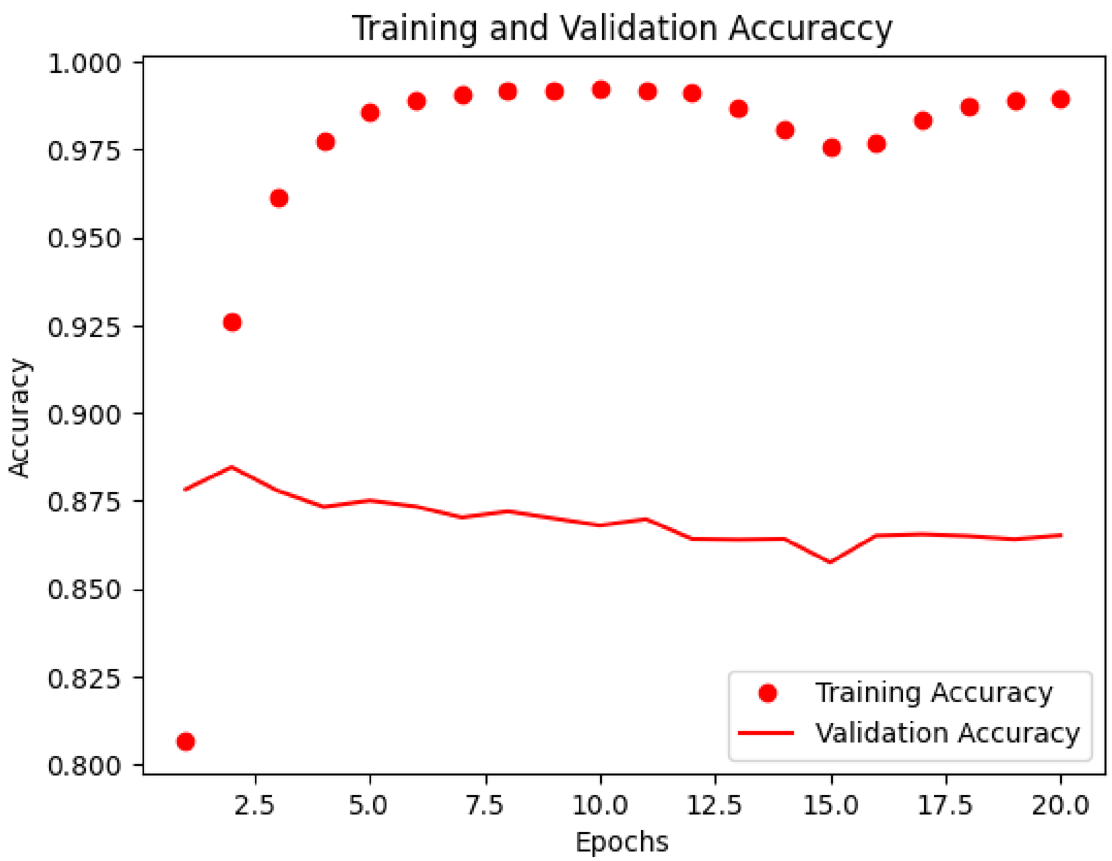
```
30/30 [==============================] - 1s 26ms/step - loss: 0.0151 -
binary_accuracy: 0.9859 - val_loss: 0.1021 - val_binary_accuracy:
0.8751
Epoch 6/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0115 -
binary_accuracy: 0.9891 - val_loss: 0.1055 - val_binary_accuracy:
0.8734
Epoch 7/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0098 -
binary_accuracy: 0.9908 - val_loss: 0.1106 - val_binary_accuracy:
0.8703
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0087 -
binary_accuracy: 0.9918 - val_loss: 0.1108 - val_binary_accuracy:
0.8720
Epoch 9/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0082 -
binary_accuracy: 0.9921 - val_loss: 0.1119 - val_binary_accuracy:
0.8700
Epoch 10/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0080 -
binary_accuracy: 0.9924 - val_loss: 0.1150 - val_binary_accuracy:
0.8680
Epoch 11/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0083 -
binary_accuracy: 0.9920 - val_loss: 0.1151 - val_binary_accuracy:
0.8698
Epoch 12/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0087 -
binary_accuracy: 0.9913 - val_loss: 0.1202 - val_binary_accuracy:
0.8642
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0115 -
binary_accuracy: 0.9868 - val_loss: 0.1224 - val_binary_accuracy:
0.8640
Epoch 14/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0169 -
binary_accuracy: 0.9805 - val_loss: 0.1239 - val_binary_accuracy:
0.8642
Epoch 15/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0201 -
binary_accuracy: 0.9759 - val_loss: 0.1297 - val_binary_accuracy:
0.8575
Epoch 16/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0199 -
binary_accuracy: 0.9768 - val_loss: 0.1221 - val_binary_accuracy:
0.8651
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0145 -
```

```
binary_accuracy: 0.9837 - val_loss: 0.1221 - val_binary_accuracy:
0.8655
Epoch 18/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0119 -
binary_accuracy: 0.9873 - val_loss: 0.1231 - val_binary_accuracy:
0.8650
Epoch 19/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0104 -
binary_accuracy: 0.9891 - val_loss: 0.1245 - val_binary_accuracy:
0.8641
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0099 -
binary_accuracy: 0.9896 - val_loss: 0.1243 - val_binary_accuracy:
0.8652
```

```
dict_keys(['loss', 'binary_accuracy', 'val_loss',
'val_binary_accuracy'])
```

```python
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")
plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(32, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=6, batch_size=512)
results = model.evaluate(x_test, y_test)

results
```





```
Epoch 1/6
49/49 [==============================] - 3s 21ms/step - loss: 0.1172 -
accuracy: 0.8416
Epoch 2/6
49/49 [==============================] - 1s 18ms/step - loss: 0.0532 -
accuracy: 0.9315
Epoch 3/6
49/49 [==============================] - 1s 22ms/step - loss: 0.0381 -
accuracy: 0.9540
Epoch 4/6
49/49 [==============================] - 1s 26ms/step - loss: 0.0289 -
accuracy: 0.9675
Epoch 5/6
49/49 [==============================] - 1s 28ms/step - loss: 0.0255 -
accuracy: 0.9715
Epoch 6/6
49/49 [==============================] - 1s 25ms/step - loss: 0.0248 -
accuracy: 0.9714
782/782 [==============================] - 4s 4ms/step - loss: 0.1190
- accuracy: 0.8622

[0.11898766458034515, 0.8622000217437744]
```

```
#We consider layer 3 64 nodes
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers

model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()

Epoch 1/20
30/30 [==============================] - 4s 64ms/step - loss: 0.1293 -
binary_accuracy: 0.8243 - val_loss: 0.0889 - val_binary_accuracy:
0.8783
Epoch 2/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0508 -
binary_accuracy: 0.9359 - val_loss: 0.0897 - val_binary_accuracy:
0.8805
```

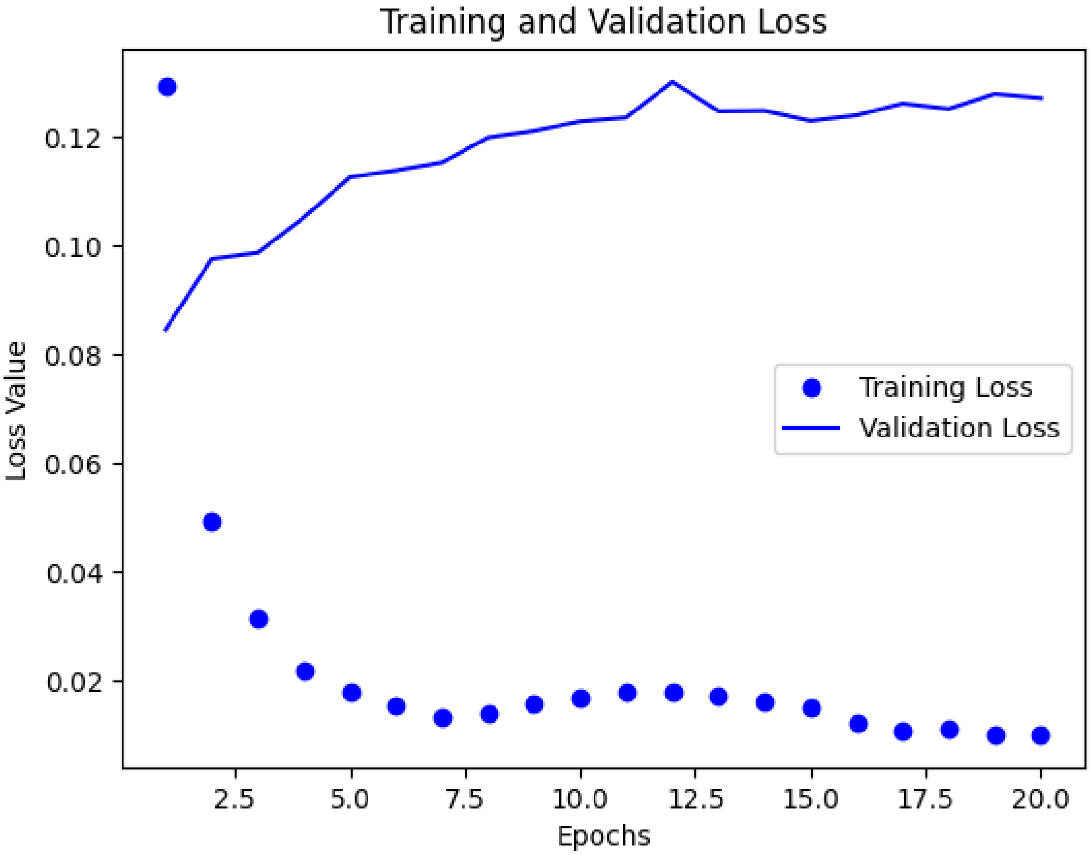
```
Epoch 3/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0307 -
binary_accuracy: 0.9653 - val_loss: 0.0978 - val_binary_accuracy:
0.8754
Epoch 4/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0218 -
binary_accuracy: 0.9765 - val_loss: 0.1033 - val_binary_accuracy:
0.8713
Epoch 5/20
30/30 [==============================] - 1s 43ms/step - loss: 0.0179 -
binary_accuracy: 0.9807 - val_loss: 0.1117 - val_binary_accuracy:
0.8678
Epoch 6/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0173 -
binary_accuracy: 0.9815 - val_loss: 0.1132 - val_binary_accuracy:
0.8681
Epoch 7/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0157 -
binary_accuracy: 0.9829 - val_loss: 0.1161 - val_binary_accuracy:
0.8681
Epoch 8/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0178 -
binary_accuracy: 0.9800 - val_loss: 0.1192 - val_binary_accuracy:
0.8651
Epoch 9/20
30/30 [==============================] - 1s 37ms/step - loss: 0.0179 -
binary_accuracy: 0.9797 - val_loss: 0.1193 - val_binary_accuracy:
0.8667
Epoch 10/20
30/30 [==============================] - 1s 41ms/step - loss: 0.0160 -
binary_accuracy: 0.9823 - val_loss: 0.1205 - val_binary_accuracy:
0.8650
Epoch 11/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0146 -
binary_accuracy: 0.9841 - val_loss: 0.1236 - val_binary_accuracy:
0.8641
Epoch 12/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0196 -
binary_accuracy: 0.9769 - val_loss: 0.1213 - val_binary_accuracy:
0.8662
Epoch 13/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0159 -
binary_accuracy: 0.9823 - val_loss: 0.1201 - val_binary_accuracy:
0.8675
Epoch 14/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0123 -
binary_accuracy: 0.9872 - val_loss: 0.1208 - val_binary_accuracy:
0.8681
Epoch 15/20
```

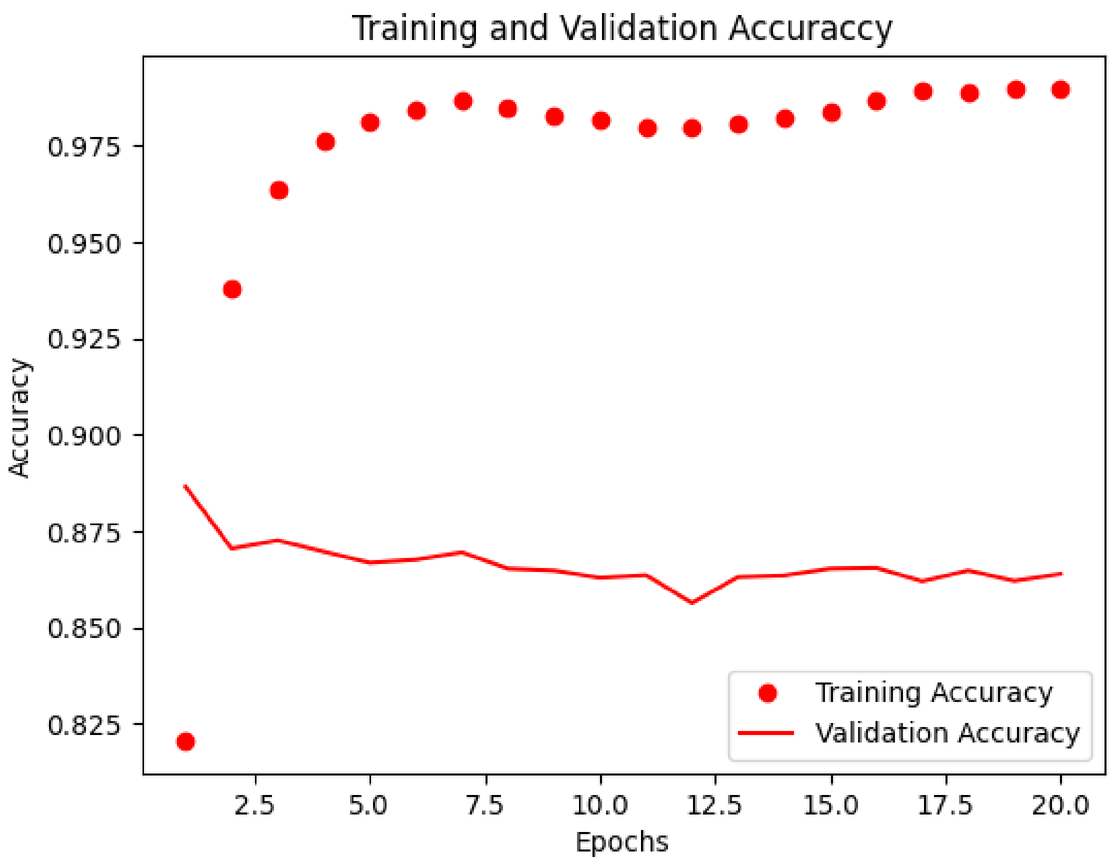
```
30/30 [==============================] - 1s 24ms/step - loss: 0.0108 -
binary_accuracy: 0.9891 - val_loss: 0.1224 - val_binary_accuracy:
0.8662
Epoch 16/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0101 -
binary_accuracy: 0.9900 - val_loss: 0.1218 - val_binary_accuracy:
0.8674
Epoch 17/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0099 -
binary_accuracy: 0.9901 - val_loss: 0.1224 - val_binary_accuracy:
0.8671
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0098 -
binary_accuracy: 0.9903 - val_loss: 0.1230 - val_binary_accuracy:
0.8678
Epoch 19/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0098 -
binary_accuracy: 0.9901 - val_loss: 0.1239 - val_binary_accuracy:
0.8674
Epoch 20/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0096 -
binary_accuracy: 0.9904 - val_loss: 0.1245 - val_binary_accuracy:
0.8658

dict_keys(['loss', 'binary_accuracy', 'val_loss',
'val_binary_accuracy'])

#We consider layer 3 64 nodes
from keras import models
from tensorflow.keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

from keras import optimizers
from keras import losses
from keras import metrics

from tensorflow import keras
from keras import optimizers
from tensorflow.keras import optimizers
from tensorflow.keras import optimizers
```

```
model.compile(optimizer='adam',
              loss = losses.mse,
              metrics = [metrics.binary_accuracy])

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

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

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

history_dict = history.history
history_dict.keys()
# Plotting the training and validation loss

import matplotlib.pyplot as plt
%matplotlib inline

loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

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

plt.plot(epochs, loss_values, 'bo', label="Training Loss")
plt.plot(epochs, val_loss_values, 'b', label="Validation Loss")

plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss Value')
plt.legend()

plt.show()


# Plotting the training and validation accuracy
# Training and Validation Accuracy

acc_values = history_dict['binary_accuracy']
val_acc_values = history_dict['val_binary_accuracy']

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

plt.plot(epochs, acc_values, 'ro', label="Training Accuracy")
plt.plot(epochs, val_acc_values, 'r', label="Validation Accuracy")

plt.title('Training and Validation Accuraccy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()


model = models.Sequential()
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(64, activation='tanh', input_shape=(10000,)))
layers.Dropout(0.5),
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',
              loss='mse',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=2, batch_size=512)
results = model.evaluate(x_test, y_test)

results

Epoch 1/20
30/30 [==============================] - 8s 68ms/step - loss: 0.1294 -
binary_accuracy: 0.8207 - val_loss: 0.0846 - val_binary_accuracy:
0.8865
Epoch 2/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0494 -
binary_accuracy: 0.9381 - val_loss: 0.0976 - val_binary_accuracy:
0.8705
Epoch 3/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0316 -
binary_accuracy: 0.9637 - val_loss: 0.0988 - val_binary_accuracy:
0.8726
Epoch 4/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0220 -
binary_accuracy: 0.9761 - val_loss: 0.1052 - val_binary_accuracy:
0.8697
Epoch 5/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0179 -
binary_accuracy: 0.9810 - val_loss: 0.1126 - val_binary_accuracy:
0.8668
Epoch 6/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0155 -
binary_accuracy: 0.9841 - val_loss: 0.1139 - val_binary_accuracy:
0.8676
Epoch 7/20
```

```
30/30 [==============================] - 1s 27ms/step - loss: 0.0133 -
binary_accuracy: 0.9865 - val_loss: 0.1153 - val_binary_accuracy:
0.8695
Epoch 8/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0139 -
binary_accuracy: 0.9849 - val_loss: 0.1199 - val_binary_accuracy:
0.8653
Epoch 9/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0158 -
binary_accuracy: 0.9826 - val_loss: 0.1211 - val_binary_accuracy:
0.8648
Epoch 10/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0169 -
binary_accuracy: 0.9815 - val_loss: 0.1229 - val_binary_accuracy:
0.8629
Epoch 11/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0181 -
binary_accuracy: 0.9796 - val_loss: 0.1236 - val_binary_accuracy:
0.8636
Epoch 12/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0179 -
binary_accuracy: 0.9799 - val_loss: 0.1301 - val_binary_accuracy:
0.8564
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0175 -
binary_accuracy: 0.9805 - val_loss: 0.1248 - val_binary_accuracy:
0.8631
Epoch 14/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0161 -
binary_accuracy: 0.9822 - val_loss: 0.1249 - val_binary_accuracy:
0.8635
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0152 -
binary_accuracy: 0.9835 - val_loss: 0.1230 - val_binary_accuracy:
0.8653
Epoch 16/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0122 -
binary_accuracy: 0.9868 - val_loss: 0.1240 - val_binary_accuracy:
0.8655
Epoch 17/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0108 -
binary_accuracy: 0.9890 - val_loss: 0.1262 - val_binary_accuracy:
0.8620
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0111 -
binary_accuracy: 0.9887 - val_loss: 0.1252 - val_binary_accuracy:
0.8648
Epoch 19/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0103 -
```

```
binary_accuracy: 0.9897 - val_loss: 0.1279 - val_binary_accuracy:
0.8621
Epoch 20/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0102 -
binary_accuracy: 0.9897 - val_loss: 0.1272 - val_binary_accuracy:
0.8639
```





```
Epoch 1/2
49/49 [==============================] - 3s 16ms/step - loss: 0.1111 -
accuracy: 0.8475
Epoch 2/2
49/49 [==============================] - 1s 15ms/step - loss: 0.0519 -
accuracy: 0.9342
782/782 [==============================] - 2s 3ms/step - loss: 0.0935
- accuracy: 0.8756

[0.09354441612958908, 0.8755999803543091]
```