Assignment-1 Neural Networks

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.0)
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>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
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) (24.1)
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/py
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorfl
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0
```

Load IMDB dataset with the top 10,000 most frequent words for training and testing sentiment analysis

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

```
print(train_data,train_data.shape)
```

```
[list([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, 3
 list([1, 194, 1153, 194, 8255, 78, 228, 5, 6, 1463, 4369, 5012, 134, 26, 4, 715, 8, 118, 1634, 14, 394, 20, 13, 119, 954, 189, 102,
 list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 5974, 54, 61, 369, 13, 71, 149, 14, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75,
 ...
 list([1, 11, 6, 230, 245, 6401, 9, 6, 1225, 446, 2, 45, 2174, 84, 8322, 4007, 21, 4, 912, 84, 2, 325, 725, 134, 2, 1715, 84, 5, 36,
 list([1, 1446, 7079, 69, 72, 3305, 13, 610, 930, 8, 12, 582, 23, 5, 16, 484, 685, 54, 349, 11, 4120, 2959, 45, 58, 1466, 13, 197, 1
 list([1, 17, 6, 194, 337, 7, 4, 204, 22, 45, 254, 8, 106, 14, 123, 4, 2, 270, 2, 5, 2, 2, 732, 2098, 101, 405, 39, 14, 1034, 4, 131
```

```
train_labels[0]
```

```
1
```

```
len(train_labels)
```

```
25000
```

```
test_labels[0]
```

```
0
```

```
max([max(sequence_647) for sequence_647 in test_data])
```

```
9999
```

Decode the first review from the training data by converting word indices back to words using the reverse word index

```
word_index_647 = imdb.get_word_index()
reverse_word_index_647 = dict(
    [(value, key) for (key, value) in word_index_647.items()])
decoded_review = " ".join(
    [reverse_word_index_647.get(i - 3, "?") for i in train_data[0]])
```

Converts each sequence of integers into a binary vector of a specified dimension where the index is set to 1 if the word is present in the sequence

```
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.])
```

```
x_test[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

model647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model647.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:]


## model planned to train with 20 epoch with batch size of 256
history = model647.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
                   batch_size=512,
                   validation_data=(x_val, y_val))
```

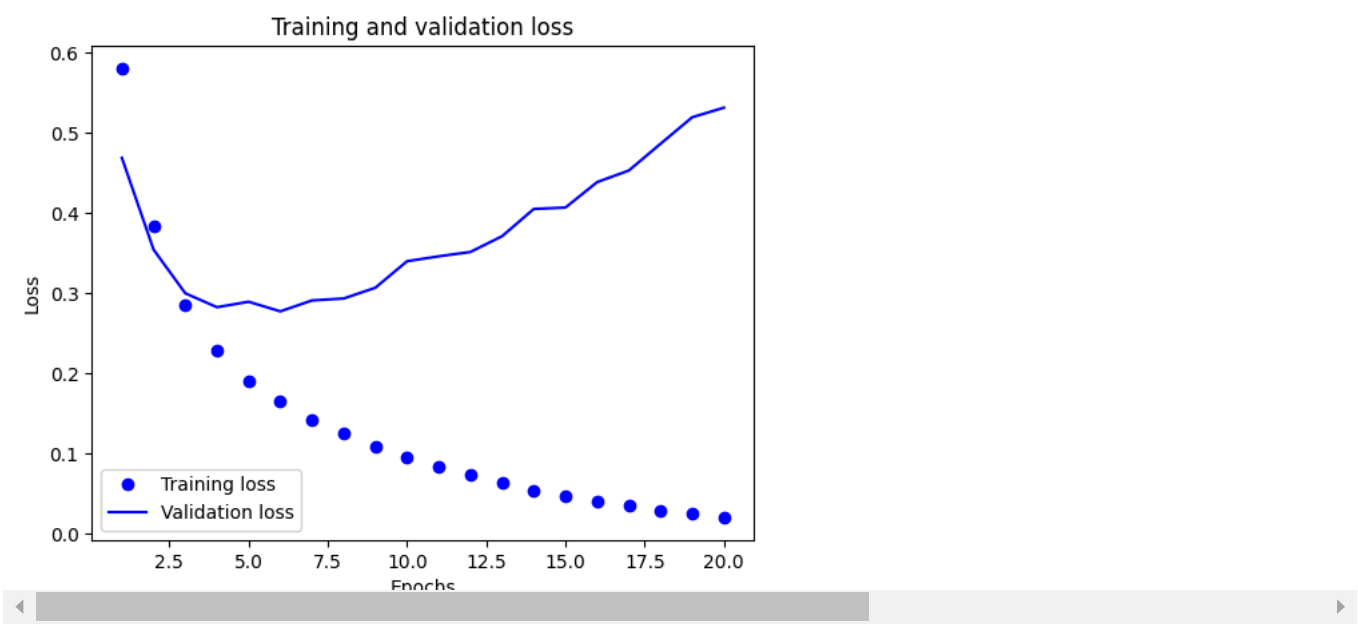
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 101ms/step - accuracy: 0.6817 - loss: 0.6398 - val_accuracy: 0.8355 - val_loss: 0.4689
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - accuracy: 0.8744 - loss: 0.4105 - val_accuracy: 0.8742 - val_loss: 0.3543
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9044 - loss: 0.2983 - val_accuracy: 0.8864 - val_loss: 0.2999
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9242 - loss: 0.2319 - val_accuracy: 0.8875 - val_loss: 0.2824
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9371 - loss: 0.1933 - val_accuracy: 0.8823 - val_loss: 0.2893
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9457 - loss: 0.1672 - val_accuracy: 0.8867 - val_loss: 0.2773
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9591 - loss: 0.1373 - val_accuracy: 0.8855 - val_loss: 0.2909
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 116ms/step - accuracy: 0.9653 - loss: 0.1176 - val_accuracy: 0.8847 - val_loss: 0.2933
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - accuracy: 0.9725 - loss: 0.1011 - val_accuracy: 0.8849 - val_loss: 0.3068
Epoch 10/20
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9739 - loss: 0.0925 - val_accuracy: 0.8812 - val_loss: 0.3399
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9765 - loss: 0.0824 - val_accuracy: 0.8757 - val_loss: 0.3460
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9812 - loss: 0.0704 - val_accuracy: 0.8779 - val_loss: 0.3514
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9864 - loss: 0.0609 - val_accuracy: 0.8805 - val_loss: 0.3711
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9906 - loss: 0.0493 - val_accuracy: 0.8763 - val_loss: 0.4051
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9910 - loss: 0.0460 - val_accuracy: 0.8757 - val_loss: 0.4069
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9930 - loss: 0.0362 - val_accuracy: 0.8713 - val_loss: 0.4387
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9946 - loss: 0.0327 - val_accuracy: 0.8737 - val_loss: 0.4531
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9957 - loss: 0.0271 - val_accuracy: 0.8704 - val_loss: 0.4865
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 58ms/step - accuracy: 0.9966 - loss: 0.0229 - val_accuracy: 0.8667 - val_loss: 0.5196
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9980 - loss: 0.0191 - val_accuracy: 0.8723 - val_loss: 0.5315
```

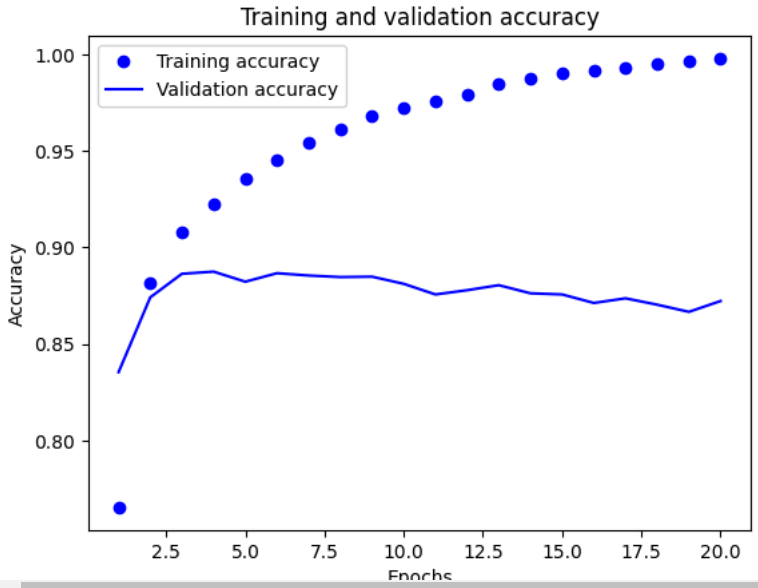
```
history_dict647 = history.history
history_dict647.keys()
```

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

```
#Plotting the training loss vs validation loss
import matplotlib.pyplot as plot647
history_dict647 = history.history
loss_values = history_dict647["loss"]
val_loss_values = history_dict647["val_loss"]
epochs = range(1, len(loss_values) + 1)
plot647.plot(epochs, loss_values, "bo", label="Training loss")
plot647.plot(epochs, val_loss_values, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```




```
#Plotting training accuracy vs validatition accuracy
plot647.clf()
acc = history_dict647["accuracy"]
val_acc = history_dict647["val_accuracy"]
plot647.plot(epochs, acc, "bo", label="Training accuracy")
plot647.plot(epochs, val_acc, "b", label="Validation accuracy")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```




```
model647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model647.fit(x_train, y_train, epochs=4, batch_size=512)
results = model647.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.7424 - loss: 0.5532
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8985 - loss: 0.2896
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 0.9248 - loss: 0.2153
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.9332 - loss: 0.1848
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8832 - loss: 0.2855
```

```
results
```

```
[0.2849089503288269, 0.8844799995422363]
```

```
model647.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.19164747],
       [0.99976003],
       [0.5941049 ],
       ...,
       [0.10002616],
       [0.07153597],
       [0.51432663]], dtype=float32)
```

```
model_647_layer = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model_647_layer.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

x_val647 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val647 = y_train[:10000]
partial_y_train = y_train[10000:]


history_layer647 = model_647_layer.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
```

```
                    batch_size=512,
                    validation_data=(x_val647, y_val647))
```

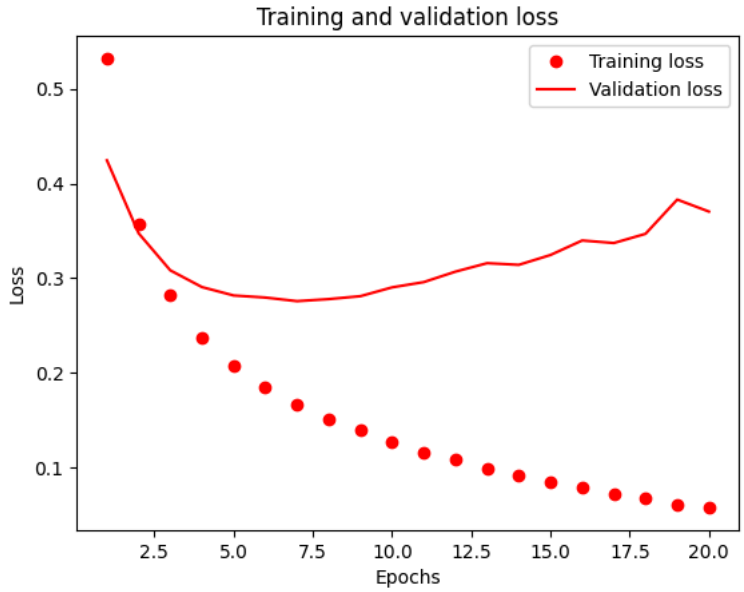
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 89ms/step - accuracy: 0.7040 - loss: 0.5990 - val_accuracy: 0.8585 - val_loss: 0.4245
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.8852 - loss: 0.3752 - val_accuracy: 0.8757 - val_loss: 0.3475
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9104 - loss: 0.2898 - val_accuracy: 0.8855 - val_loss: 0.3083
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9216 - loss: 0.2438 - val_accuracy: 0.8873 - val_loss: 0.2907
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9342 - loss: 0.2061 - val_accuracy: 0.8873 - val_loss: 0.2818
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9479 - loss: 0.1808 - val_accuracy: 0.8881 - val_loss: 0.2796
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9477 - loss: 0.1641 - val_accuracy: 0.8869 - val_loss: 0.2758
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9566 - loss: 0.1471 - val_accuracy: 0.8862 - val_loss: 0.2780
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9550 - loss: 0.1434 - val_accuracy: 0.8858 - val_loss: 0.2810
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9642 - loss: 0.1258 - val_accuracy: 0.8856 - val_loss: 0.2903
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 0.9684 - loss: 0.1156 - val_accuracy: 0.8851 - val_loss: 0.2959
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.9702 - loss: 0.1073 - val_accuracy: 0.8835 - val_loss: 0.3069
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9749 - loss: 0.0981 - val_accuracy: 0.8765 - val_loss: 0.3159
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9777 - loss: 0.0923 - val_accuracy: 0.8799 - val_loss: 0.3141
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9794 - loss: 0.0834 - val_accuracy: 0.8792 - val_loss: 0.3245
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9817 - loss: 0.0777 - val_accuracy: 0.8751 - val_loss: 0.3398
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9846 - loss: 0.0711 - val_accuracy: 0.8791 - val_loss: 0.3372
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9881 - loss: 0.0651 - val_accuracy: 0.8793 - val_loss: 0.3469
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.9893 - loss: 0.0596 - val_accuracy: 0.8747 - val_loss: 0.3830
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.9885 - loss: 0.0587 - val_accuracy: 0.8746 - val_loss: 0.3704
```

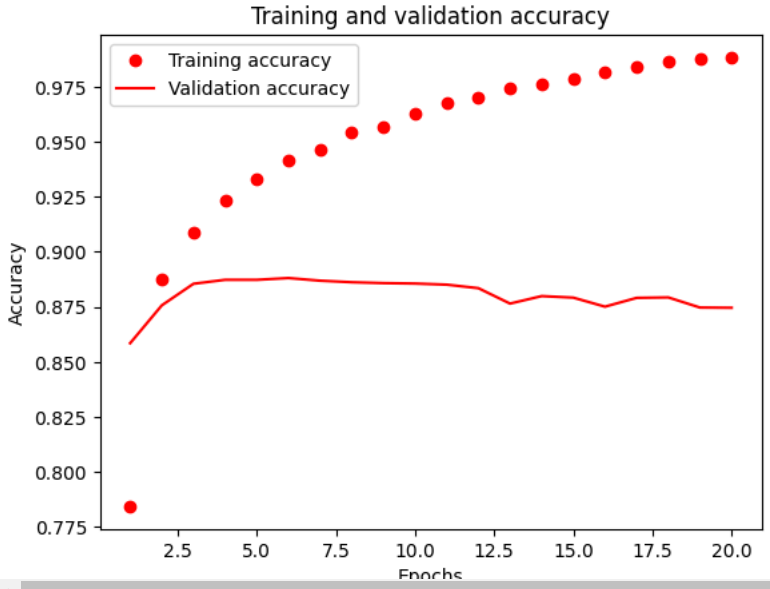
Extract the history of model training from history_layer647 and display the keys of the history dictionary

```
history_dict647 = history_layer647.history
history_dict647.keys()
```

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

```
import matplotlib.pyplot as plot647
history_dict647 = history_layer647.history
loss_value647 = history_dict647["loss"]
val_loss_value647 = history_dict647["val_loss"]
epochs647 = range(1, len(loss_value647) + 1)

#Plotting graph of Training and Validation loss
plot647.plot(epochs647, loss_value647, "ro", label="Training loss")
plot647.plot(epochs647, val_loss_value647, "r", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()

#Plotting graph of Training and Validation Accuracy
plot647.clf()
accuracy647 = history_dict647["accuracy"]
val_accuracy1 = history_dict647["val_accuracy"]
plot647.plot(epochs647, accuracy647, "ro", label="Training accuracy")
plot647.plot(epochs647, val_accuracy1, "r", label="Validation accuracy")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
model_647_layer = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model_647_layer.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model_647_layer.fit(x_train, y_train, epochs=5, batch_size=512)
result_647_layer = model_647_layer.evaluate(x_test, y_test)
```

```
Epoch 1/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.7519 - loss: 0.5362
Epoch 2/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.9017 - loss: 0.3006
Epoch 3/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.9220 - loss: 0.2356
Epoch 4/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.9275 - loss: 0.2066
Epoch 5/5
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - accuracy: 0.9374 - loss: 0.1811
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8871 - loss: 0.2804
```

```
print(result_647_layer)
```

```
[0.27885910868644714, 0.8884000182151794]
```

```
model_647_layer.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.23275131],
       [0.9998543 ],
```

```
            [0.8417611 ],
            ...,
            [0.1360763 ],
            [0.08439167],
            [0.5835538 ]], dtype=float32)


model_3_layers_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_3_layers_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
x_val3_647 = x_train[:10000]
partial_x_train_647 = x_train[10000:]

y_val3_647 = y_train[:10000]
partial_y_train_647 = y_train[10000:]

history_3_layers_647 = model_3_layers_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val3_647, y_val3_647))
```

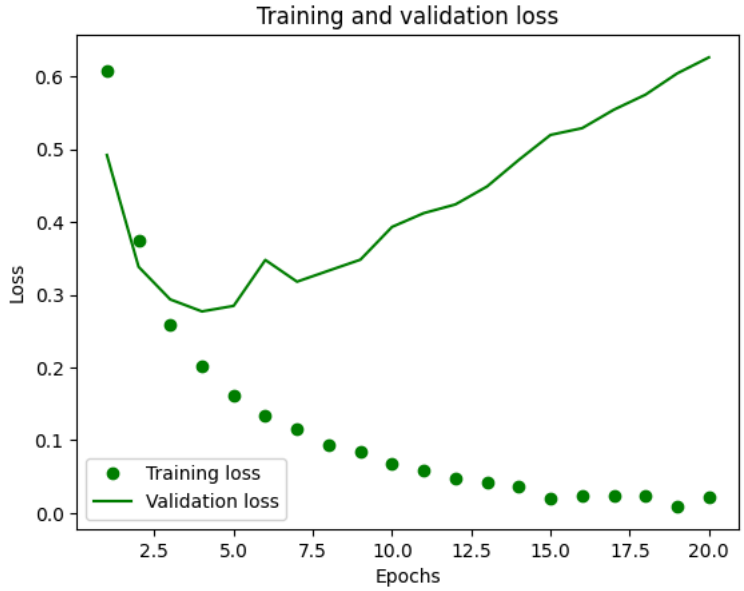
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 84ms/step - accuracy: 0.5821 - loss: 0.6564 - val_accuracy: 0.8516 - val_loss: 0.4921
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8833 - loss: 0.4087 - val_accuracy: 0.8768 - val_loss: 0.3384
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9183 - loss: 0.2641 - val_accuracy: 0.8878 - val_loss: 0.2936
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9356 - loss: 0.1994 - val_accuracy: 0.8887 - val_loss: 0.2771
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9509 - loss: 0.1582 - val_accuracy: 0.8863 - val_loss: 0.2847
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9564 - loss: 0.1338 - val_accuracy: 0.8688 - val_loss: 0.3478
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9611 - loss: 0.1186 - val_accuracy: 0.8794 - val_loss: 0.3180
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9752 - loss: 0.0898 - val_accuracy: 0.8796 - val_loss: 0.3331
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9781 - loss: 0.0770 - val_accuracy: 0.8795 - val_loss: 0.3482
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.9857 - loss: 0.0585 - val_accuracy: 0.8737 - val_loss: 0.3933
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - accuracy: 0.9844 - loss: 0.0564 - val_accuracy: 0.8743 - val_loss: 0.4122
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9904 - loss: 0.0411 - val_accuracy: 0.8772 - val_loss: 0.4242
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9920 - loss: 0.0370 - val_accuracy: 0.8751 - val_loss: 0.4490
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9954 - loss: 0.0272 - val_accuracy: 0.8732 - val_loss: 0.4854
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9977 - loss: 0.0189 - val_accuracy: 0.8713 - val_loss: 0.5198
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9940 - loss: 0.0274 - val_accuracy: 0.8732 - val_loss: 0.5291
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9955 - loss: 0.0197 - val_accuracy: 0.8745 - val_loss: 0.5545
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9972 - loss: 0.0153 - val_accuracy: 0.8716 - val_loss: 0.5752
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9992 - loss: 0.0085 - val_accuracy: 0.8719 - val_loss: 0.6044
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9954 - loss: 0.0174 - val_accuracy: 0.8702 - val_loss: 0.6265
```

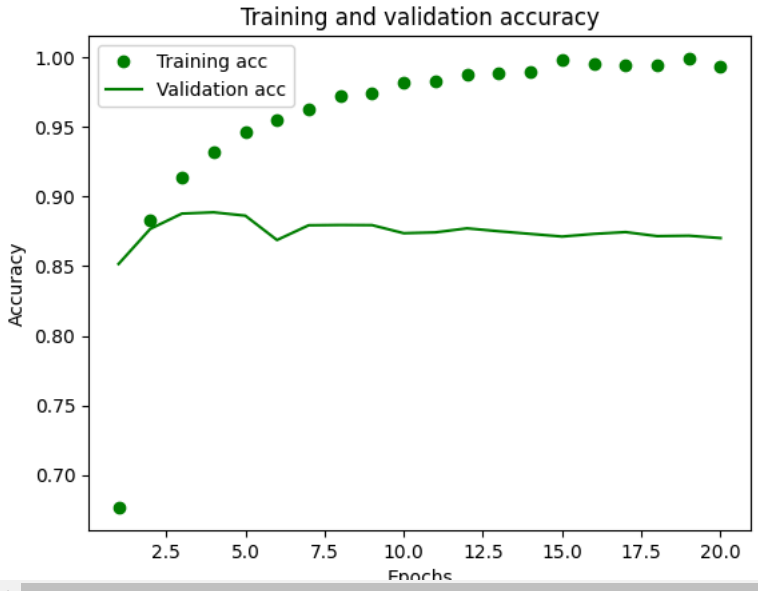
```
history_dict_3_647 = history_3_layers_647.history
history_dict_3_647.keys()
```

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

```
loss_val647 = history_dict_3_647["loss"]
val_loss_val3 = history_dict_3_647["val_loss"]
epochs3 = range(1, len(loss_val647) + 1)
plot647.plot(epochs3, loss_val647, "go", label="Training loss")
plot647.plot(epochs3, val_loss_val3, "g", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
accuracy3 = history_dict_3_647["accuracy"]
val_accuracy3 = history_dict_3_647["val_accuracy"]
plot647.plot(epochs3, accuracy3, "go", label="Training acc")
plot647.plot(epochs3, val_accuracy3, "g", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
model_3_layers_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model_3_layers_647.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model_3_layers_647.fit(x_train, y_train, epochs=3, batch_size=512)
results_3_layers = model_3_layers_647.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 26ms/step - accuracy: 0.7184 - loss: 0.5720
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.9004 - loss: 0.2734
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.9290 - loss: 0.2031
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8837 - loss: 0.2841
```

```
print(result_647_layer)
```

```
[0.27885910868644714, 0.8884000182151794]
```

```
model_647_layer.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.23275131],
       [0.9998543 ],
       [0.8417611 ],
       ...,
       [0.1360763 ],
       [0.08439167],
       [0.5835538 ]], dtype=float32)
```

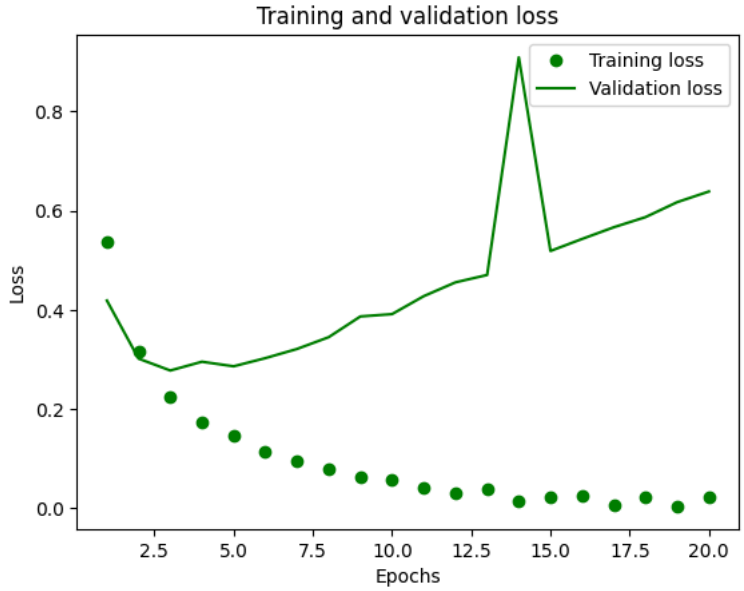
```
model_3_layers_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_3_layers_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
x_val3_647 = x_train[:10000]
partial_x_train_647 = x_train[10000:]

y_val3_647 = y_train[:10000]
partial_y_train_647 = y_train[10000:]

history_3_layers_647 = model_3_layers_647.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
                   batch_size=512,
                   validation_data=(x_val3_647, y_val3_647))
```

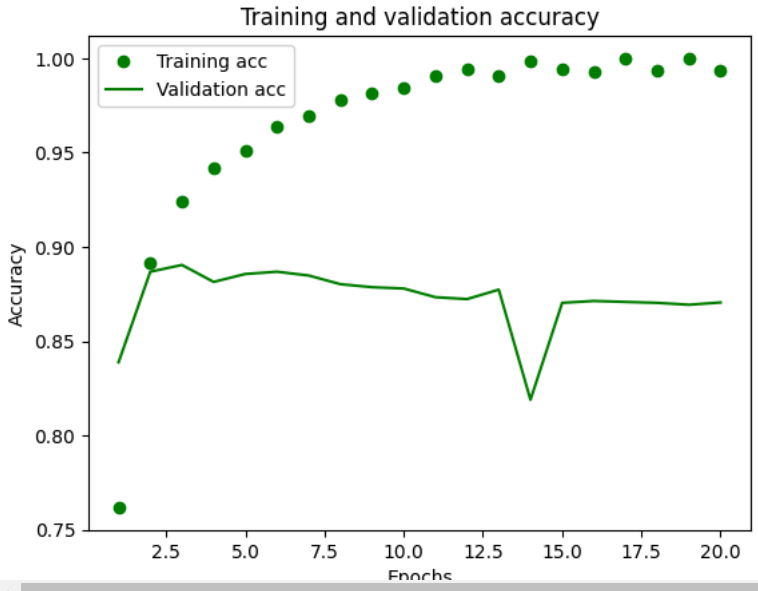
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 77ms/step - accuracy: 0.6845 - loss: 0.6106 - val_accuracy: 0.8388 - val_loss: 0.4187
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.8878 - loss: 0.3361 - val_accuracy: 0.8868 - val_loss: 0.3016
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - accuracy: 0.9280 - loss: 0.2245 - val_accuracy: 0.8904 - val_loss: 0.2776
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9447 - loss: 0.1746 - val_accuracy: 0.8814 - val_loss: 0.2953
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9522 - loss: 0.1480 - val_accuracy: 0.8856 - val_loss: 0.2861
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9682 - loss: 0.1105 - val_accuracy: 0.8868 - val_loss: 0.3025
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9718 - loss: 0.0940 - val_accuracy: 0.8848 - val_loss: 0.3211
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9803 - loss: 0.0737 - val_accuracy: 0.8802 - val_loss: 0.3448
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9847 - loss: 0.0597 - val_accuracy: 0.8786 - val_loss: 0.3866
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9889 - loss: 0.0475 - val_accuracy: 0.8779 - val_loss: 0.3914
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9937 - loss: 0.0335 - val_accuracy: 0.8733 - val_loss: 0.4276
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.9952 - loss: 0.0277 - val_accuracy: 0.8723 - val_loss: 0.4554
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9937 - loss: 0.0288 - val_accuracy: 0.8773 - val_loss: 0.4703
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9992 - loss: 0.0120 - val_accuracy: 0.8189 - val_loss: 0.9092
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9805 - loss: 0.0570 - val_accuracy: 0.8703 - val_loss: 0.5187
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9965 - loss: 0.0151 - val_accuracy: 0.8713 - val_loss: 0.5430
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9997 - loss: 0.0055 - val_accuracy: 0.8708 - val_loss: 0.5666
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9943 - loss: 0.0192 - val_accuracy: 0.8703 - val_loss: 0.5870
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 1.0000 - loss: 0.0036 - val_accuracy: 0.8693 - val_loss: 0.6173
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9975 - loss: 0.0095 - val_accuracy: 0.8705 - val_loss: 0.6387
```

```
history_dict_3_647 = history_3_layers_647.history
history_dict_3_647.keys()
```

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

```
loss_val647 = history_dict_3_647["loss"]
val_loss_val3 = history_dict_3_647["val_loss"]
epochs3 = range(1, len(loss_val647) + 1)
plot647.plot(epochs3, loss_val647, "go", label="Training loss")
plot647.plot(epochs3, val_loss_val3, "g", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()


plot647.clf()
accuracy3 = history_dict_3_647["accuracy"]
val_accuracy3 = history_dict_3_647["val_accuracy"]
plot647.plot(epochs3, accuracy3, "go", label="Training acc")
plot647.plot(epochs3, val_accuracy3, "g", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
model_3_layers_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model_3_layers_647.compile(optimizer='rmsprop',
```

```
              loss='binary_crossentropy',
              metrics=['accuracy'])

model_3_layers_647.fit(x_train, y_train, epochs=3, batch_size=512)
results_3_layers = model_3_layers_647.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 26ms/step - accuracy: 0.6554 - loss: 0.6055
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.8951 - loss: 0.3255
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9272 - loss: 0.2145
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8844 - loss: 0.2905
```

```
print(results_3_layers)
```

```
[0.2883072793483734, 0.8863199949264526]
```

```
model_3_layers_647.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
array([[0.22604847],
       [0.99998087],
       [0.74288774],
       ...,
       [0.13457565],
       [0.1173477 ],
       [0.6008797 ]], dtype=float32)
```

```
model_32_units_647 = keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
#model compilation
model_32_units_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
#model validation
x_val_32_647 = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_32_647 = y_train[:10000]
partial_y_train = y_train[10000:]
```

```
history_32_units_647 = model_32_units_647.fit(partial_x_train,
                   partial_y_train,
                   epochs=20,
                   batch_size=512,
                   validation_data=(x_val_32_647, y_val_32_647))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 71ms/step - accuracy: 0.6865 - loss: 0.5992 - val_accuracy: 0.8603 - val_loss: 0.3665
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.8888 - loss: 0.3071 - val_accuracy: 0.8775 - val_loss: 0.3045
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.9269 - loss: 0.2097 - val_accuracy: 0.8627 - val_loss: 0.3379
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.9397 - loss: 0.1671 - val_accuracy: 0.8758 - val_loss: 0.3191
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.9607 - loss: 0.1234 - val_accuracy: 0.8850 - val_loss: 0.2972
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9698 - loss: 0.0949 - val_accuracy: 0.8830 - val_loss: 0.3225
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9771 - loss: 0.0757 - val_accuracy: 0.8688 - val_loss: 0.4254
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9760 - loss: 0.0721 - val_accuracy: 0.8731 - val_loss: 0.3810
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9902 - loss: 0.0395 - val_accuracy: 0.8809 - val_loss: 0.3958
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.9901 - loss: 0.0351 - val_accuracy: 0.8779 - val_loss: 0.4202
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 68ms/step - accuracy: 0.9940 - loss: 0.0257 - val_accuracy: 0.8790 - val_loss: 0.4515
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9986 - loss: 0.0113 - val_accuracy: 0.8729 - val_loss: 0.5091
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9995 - loss: 0.0091 - val_accuracy: 0.8776 - val_loss: 0.5164
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9994 - loss: 0.0055 - val_accuracy: 0.8649 - val_loss: 0.5890
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9998 - loss: 0.0044 - val_accuracy: 0.8785 - val_loss: 0.5707
Epoch 16/20
```

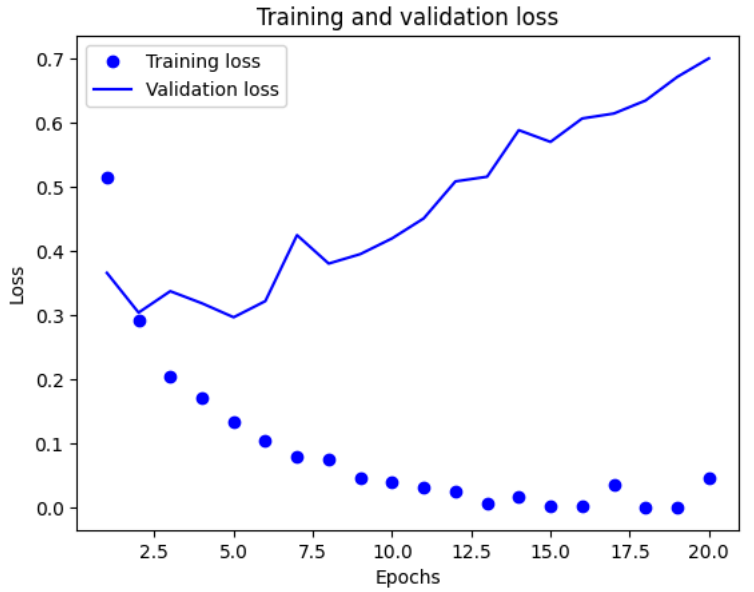
```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9995 - loss: 0.0023 - val_accuracy: 0.8759 - val_loss: 0.6072
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - accuracy: 0.9934 - loss: 0.0254 - val_accuracy: 0.8785 - val_loss: 0.6149
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 75ms/step - accuracy: 0.9997 - loss: 0.0014 - val_accuracy: 0.8775 - val_loss: 0.6352
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 1.0000 - loss: 9.7252e-04 - val_accuracy: 0.8772 - val_loss: 0.6719
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9963 - loss: 0.0140 - val_accuracy: 0.8703 - val_loss: 0.7009
```

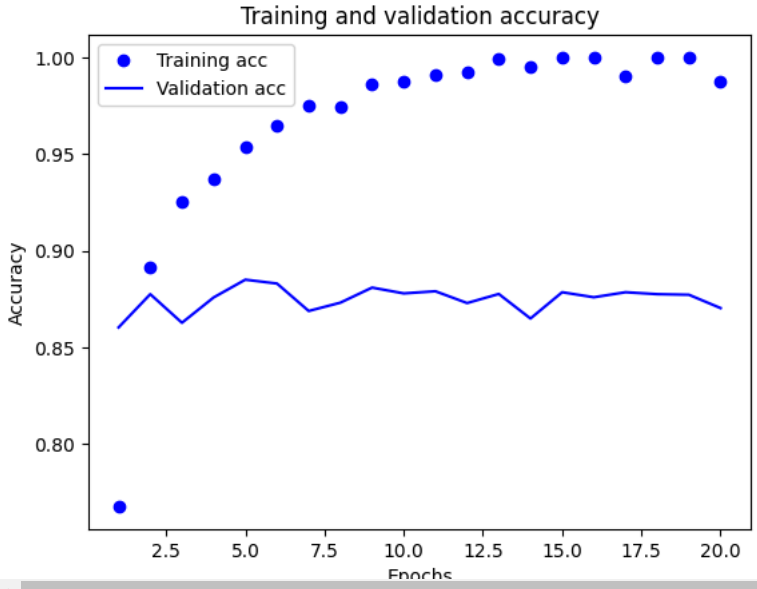
```
history_dict_32_647 = history_32_units_647.history
history_dict_32_647.keys()
```

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

```
loss_value_32_647 = history_dict_32_647["loss"]
val_loss_value_32_647 = history_dict_32_647["val_loss"]
epochs_32 = range(1, len(loss_value_32_647) + 1)
plot647.plot(epochs_32, loss_value_32_647, "bo", label="Training loss")
plot647.plot(epochs_32, val_loss_value_32_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
accuracy_32 = history_dict_32_647["accuracy"]
val_accuracy_32 = history_dict_32_647["val_accuracy"]
plot647.plot(epochs_32, accuracy_32, "bo", label="Training acc")
plot647.plot(epochs_32, val_accuracy_32, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
history_32_units_647 = model_32_units_647.fit(x_train, y_train, epochs=3, batch_size=512)
results_32_units_647 = model_32_units_647.evaluate(x_test, y_test)
results_32_units_647
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9485 - loss: 0.2264
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 34ms/step - accuracy: 0.9684 - loss: 0.1004
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - accuracy: 0.9817 - loss: 0.0654
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8628 - loss: 0.4347
[0.42939987778663635, 0.8667600154876709]
```

```
model_64_units_647 = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_64_units_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
# validation
x_val_64_2 = x_train[:10000]
partial_x_train_64_2 = x_train[10000:]

y_val_64_2 = y_train[:10000]
partial_y_train_64_2 = y_train[10000:]

history_64_647 = model_64_units_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
```

```
                    batch_size=512,
                    validation_data=(x_val_64_2, y_val_64_2))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 91ms/step - accuracy: 0.6485 - loss: 0.5988 - val_accuracy: 0.8406 - val_loss: 0.3907
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 96ms/step - accuracy: 0.8961 - loss: 0.3047 - val_accuracy: 0.8790 - val_loss: 0.2992
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 65ms/step - accuracy: 0.9146 - loss: 0.2296 - val_accuracy: 0.8762 - val_loss: 0.3083
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9323 - loss: 0.1819 - val_accuracy: 0.8866 - val_loss: 0.2807
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9456 - loss: 0.1496 - val_accuracy: 0.8837 - val_loss: 0.2888
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.9608 - loss: 0.1184 - val_accuracy: 0.8649 - val_loss: 0.3644
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 125ms/step - accuracy: 0.9658 - loss: 0.1038 - val_accuracy: 0.8678 - val_loss: 0.3705
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.9754 - loss: 0.0813 - val_accuracy: 0.8728 - val_loss: 0.3579
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9791 - loss: 0.0683 - val_accuracy: 0.8825 - val_loss: 0.3755
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.9839 - loss: 0.0548 - val_accuracy: 0.8764 - val_loss: 0.4223
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9858 - loss: 0.0504 - val_accuracy: 0.8781 - val_loss: 0.4140
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 117ms/step - accuracy: 0.9920 - loss: 0.0328 - val_accuracy: 0.8769 - val_loss: 0.4296
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 66ms/step - accuracy: 0.9971 - loss: 0.0188 - val_accuracy: 0.8794 - val_loss: 0.4422
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.9985 - loss: 0.0122 - val_accuracy: 0.8765 - val_loss: 0.4940
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.9966 - loss: 0.0166 - val_accuracy: 0.8774 - val_loss: 0.5061
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 63ms/step - accuracy: 0.9999 - loss: 0.0061 - val_accuracy: 0.8769 - val_loss: 0.5495
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 122ms/step - accuracy: 0.9937 - loss: 0.0224 - val_accuracy: 0.8754 - val_loss: 0.5386
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 1.0000 - loss: 0.0039 - val_accuracy: 0.8711 - val_loss: 0.5883
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 68ms/step - accuracy: 0.9967 - loss: 0.0123 - val_accuracy: 0.8756 - val_loss: 0.5775
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - accuracy: 1.0000 - loss: 0.0030 - val_accuracy: 0.8741 - val_loss: 0.6116
```

```
history_dict_64_647 = history_64_647.history
history_dict_64_647.keys()
```

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

```
model_64_units_647 = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_64_units_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
# validation
x_val_64_2 = x_train[:10000]
partial_x_train_64_2 = x_train[10000:]

y_val_64_2 = y_train[:10000]
partial_y_train_64_2 = y_train[10000:]

history_64_647 = model_64_units_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_64_2, y_val_64_2))
```

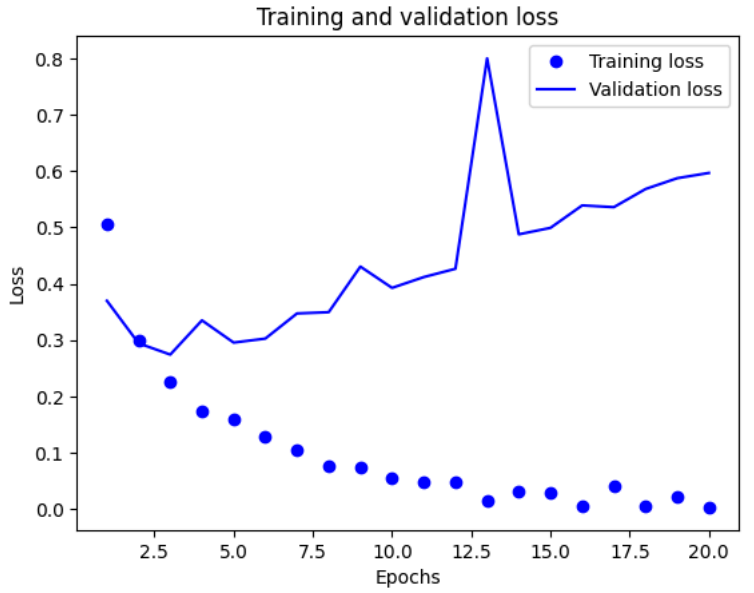
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 129ms/step - accuracy: 0.6915 - loss: 0.5900 - val_accuracy: 0.8577 - val_loss: 0.3698
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.8855 - loss: 0.3108 - val_accuracy: 0.8828 - val_loss: 0.2942
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 125ms/step - accuracy: 0.9226 - loss: 0.2263 - val_accuracy: 0.8891 - val_loss: 0.2741
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 159ms/step - accuracy: 0.9411 - loss: 0.1703 - val_accuracy: 0.8666 - val_loss: 0.3350
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 93ms/step - accuracy: 0.9345 - loss: 0.1654 - val_accuracy: 0.8831 - val_loss: 0.2954
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 66ms/step - accuracy: 0.9609 - loss: 0.1175 - val_accuracy: 0.8852 - val_loss: 0.3025
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9702 - loss: 0.0969 - val_accuracy: 0.8774 - val_loss: 0.3473
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 95ms/step - accuracy: 0.9795 - loss: 0.0725 - val_accuracy: 0.8802 - val_loss: 0.3495
```

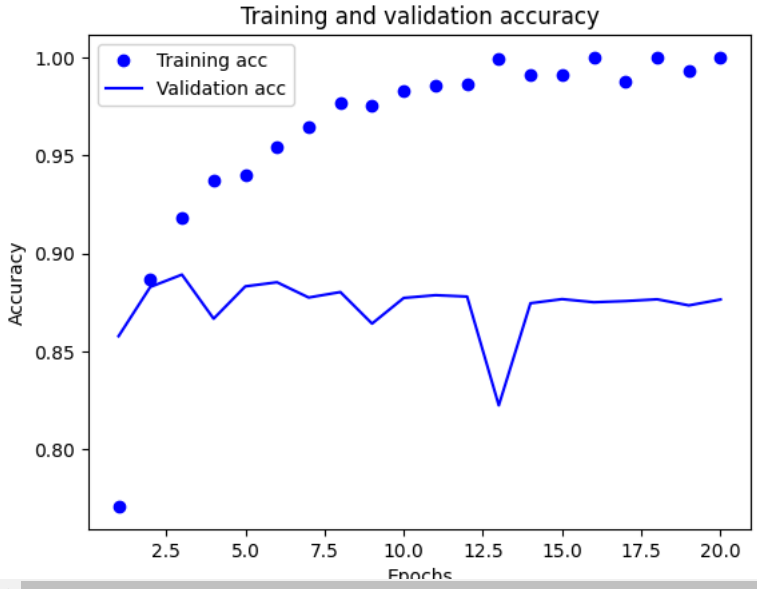
```
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 91ms/step - accuracy: 0.9732 - loss: 0.0787 - val_accuracy: 0.8641 - val_loss: 0.4305
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 69ms/step - accuracy: 0.9889 - loss: 0.0451 - val_accuracy: 0.8772 - val_loss: 0.3927
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.9909 - loss: 0.0367 - val_accuracy: 0.8786 - val_loss: 0.4119
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 66ms/step - accuracy: 0.9937 - loss: 0.0297 - val_accuracy: 0.8779 - val_loss: 0.4266
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 123ms/step - accuracy: 0.9994 - loss: 0.0141 - val_accuracy: 0.8224 - val_loss: 0.8006
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9723 - loss: 0.0725 - val_accuracy: 0.8745 - val_loss: 0.4878
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 69ms/step - accuracy: 0.9954 - loss: 0.0184 - val_accuracy: 0.8766 - val_loss: 0.4994
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.9998 - loss: 0.0058 - val_accuracy: 0.8750 - val_loss: 0.5392
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9927 - loss: 0.0263 - val_accuracy: 0.8756 - val_loss: 0.5362
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 124ms/step - accuracy: 0.9998 - loss: 0.0041 - val_accuracy: 0.8765 - val_loss: 0.5684
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.9992 - loss: 0.0049 - val_accuracy: 0.8734 - val_loss: 0.5876
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 1.0000 - loss: 0.0027 - val_accuracy: 0.8764 - val_loss: 0.5970
```

```
history_dict_64_647 = history_64_647.history
history_dict_64_647.keys()
```

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

```
loss_value64 = history_dict_64_647["loss"]
val_loss_value64 = history_dict_64_647["val_loss"]
epochs_64 = range(1, len(loss_value64) + 1)
plot647.plot(epochs_64, loss_value64, "bo", label="Training loss")
plot647.plot(epochs_64, val_loss_value64, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()

plot647.clf()
accuracy_64 = history_dict_64_647["accuracy"]
val_accuracy_64 = history_dict_64_647["val_accuracy"]
plot647.plot(epochs_64, accuracy_64, "bo", label="Training acc")
plot647.plot(epochs_64, val_accuracy_64, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
history_64_647 = model_64_units_647.fit(x_train, y_train, epochs=3, batch_size=512)
results_64_units_647 = model_64_units_647.evaluate(x_test, y_test)
results_64_units_647
```

```
Epoch 1/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 68ms/step - accuracy: 0.9375 - loss: 0.2510
Epoch 2/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 52ms/step - accuracy: 0.9694 - loss: 0.1020
Epoch 3/3
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 52ms/step - accuracy: 0.9834 - loss: 0.0609
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8605 - loss: 0.4092
[0.40178459882736206, 0.8645200133323669]
```

```
model_64_units_647.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step
array([[0.04641519],
       [0.9999956 ],
       [0.9260501 ],
       ...,
       [0.12760481],
       [0.02454367],
       [0.94523245]], dtype=float32)
```

```
model_128units_647 = keras.Sequential([
    layers.Dense(128, activation="relu"),
    layers.Dense(128, activation="relu"),
    layers.Dense(128, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_128units_647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
```

```
              metrics=["accuracy"])
# validation
x_val_128_647 = x_train[:10000]
partial_x_train_647 = x_train[10000:]

y_val_128_647 = y_train[:10000]
partial_y_train_647 = y_train[10000:]

history_128_3 = model_128units_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_128_647, y_val_128_647))
```

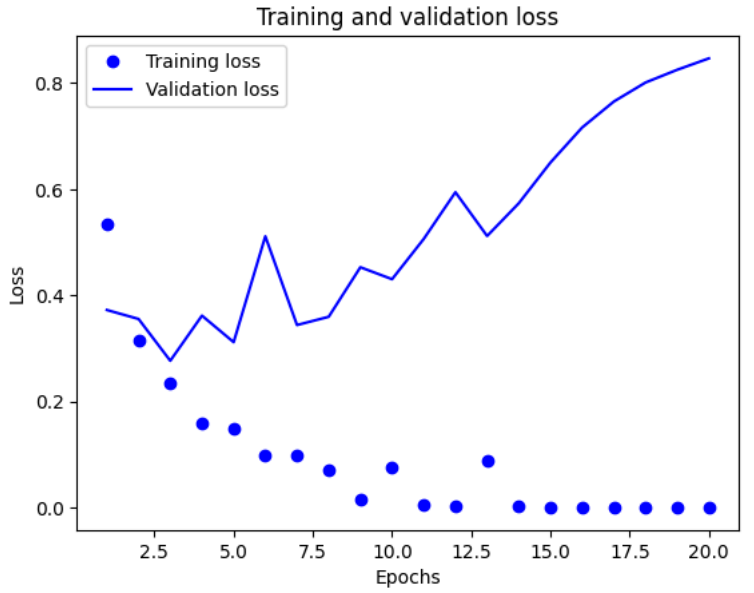
```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 168ms/step - accuracy: 0.6332 - loss: 0.6176 - val_accuracy: 0.8464 - val_loss: 0.3723
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 101ms/step - accuracy: 0.8525 - loss: 0.3497 - val_accuracy: 0.8511 - val_loss: 0.3553
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 98ms/step - accuracy: 0.9123 - loss: 0.2313 - val_accuracy: 0.8885 - val_loss: 0.2765
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 126ms/step - accuracy: 0.9415 - loss: 0.1608 - val_accuracy: 0.8628 - val_loss: 0.3616
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 101ms/step - accuracy: 0.9338 - loss: 0.1657 - val_accuracy: 0.8823 - val_loss: 0.3116
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 112ms/step - accuracy: 0.9714 - loss: 0.0912 - val_accuracy: 0.8486 - val_loss: 0.5113
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 146ms/step - accuracy: 0.9686 - loss: 0.0881 - val_accuracy: 0.8836 - val_loss: 0.3439
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 117ms/step - accuracy: 0.9936 - loss: 0.0286 - val_accuracy: 0.8805 - val_loss: 0.3594
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.9976 - loss: 0.0178 - val_accuracy: 0.8817 - val_loss: 0.4529
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 101ms/step - accuracy: 0.9961 - loss: 0.0197 - val_accuracy: 0.8782 - val_loss: 0.4303
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 6s 141ms/step - accuracy: 0.9998 - loss: 0.0061 - val_accuracy: 0.8812 - val_loss: 0.5063
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 105ms/step - accuracy: 0.9999 - loss: 0.0024 - val_accuracy: 0.8803 - val_loss: 0.5943
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 109ms/step - accuracy: 0.9905 - loss: 0.0500 - val_accuracy: 0.8804 - val_loss: 0.5118
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 169ms/step - accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.8800 - val_loss: 0.5736
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 1.0000 - loss: 8.1785e-04 - val_accuracy: 0.8800 - val_loss: 0.6504
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 101ms/step - accuracy: 1.0000 - loss: 3.7031e-04 - val_accuracy: 0.8801 - val_loss: 0.7164
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 109ms/step - accuracy: 1.0000 - loss: 1.8756e-04 - val_accuracy: 0.8794 - val_loss: 0.7654
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 100ms/step - accuracy: 1.0000 - loss: 1.2349e-04 - val_accuracy: 0.8797 - val_loss: 0.8012
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 102ms/step - accuracy: 1.0000 - loss: 8.9405e-05 - val_accuracy: 0.8794 - val_loss: 0.8253
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 100ms/step - accuracy: 1.0000 - loss: 6.6273e-05 - val_accuracy: 0.8796 - val_loss: 0.8468
```

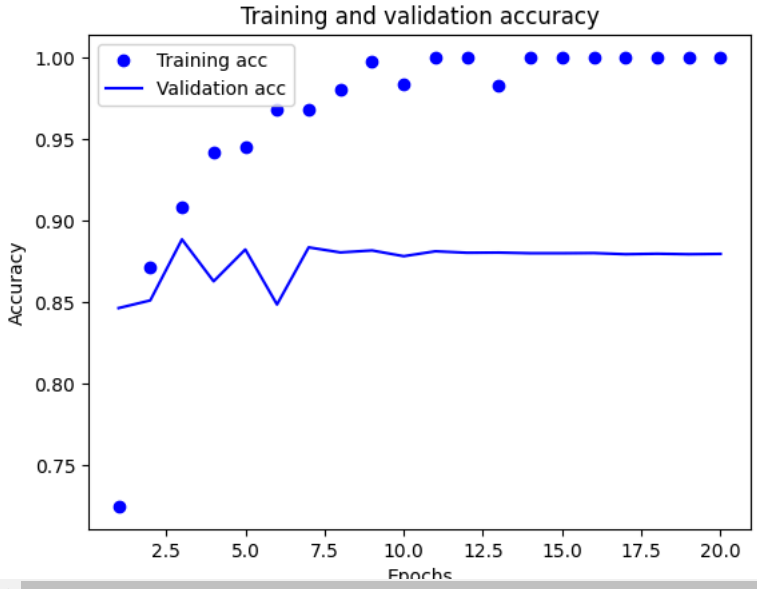
```
history_dict_128_3 = history_128_3.history
history_dict_128_3.keys()
```

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

```
loss_value128_3 = history_dict_128_3["loss"]
val_loss_value128_3 = history_dict_128_3["val_loss"]
epochs_128 = range(1, len(loss_value128_3) + 1)
plot647.plot(epochs_128, loss_value128_3, "bo", label="Training loss")
plot647.plot(epochs_128, val_loss_value128_3, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
accuracy_128 = history_dict_128_3["accuracy"]
val_accuracy_128 = history_dict_128_3["val_accuracy"]
plot647.plot(epochs_128, accuracy_128, "bo", label="Training acc")
plot647.plot(epochs_128, val_accuracy_128, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
history_128_3 = model_128units_647.fit(x_train, y_train, epochs=2, batch_size=512)
results_128_units_3 = model_128units_647.evaluate(x_test, y_test)
results_128_units_3
```

```
Epoch 1/2
49/49 ━━━━━━━━━━━━━━━━━━━━ 4s 85ms/step - accuracy: 0.9296 - loss: 0.3675
Epoch 2/2
49/49 ━━━━━━━━━━━━━━━━━━━━ 5s 77ms/step - accuracy: 0.9713 - loss: 0.0916
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.8733 - loss: 0.3646
[0.3663959801197052, 0.8749200105667114]
```

```
model_128units_647.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step
array([[0.00854527],
       [0.9999993 ],
       [0.56626767],
       ...,
       [0.00737874],
       [0.00681615],
       [0.8902984 ]], dtype=float32)
```

MSE Loss Function model with 16 units and 3-layers

```
MSE_model_16_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
# compilation of model
```

```
MSE_model_16_647.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
# validation of model
x_val_MSE_16 = x_train[:10000]
partial_x_train_16 = x_train[10000:]

y_val_MSE_16 = y_train[:10000]
partial_y_train_16 = y_train[10000:]
# Model Fit

history_MSE_647 = MSE_model_16_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_MSE_16, y_val_MSE_16))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 64ms/step - accuracy: 0.6514 - loss: 0.2306 - val_accuracy: 0.8363 - val_loss: 0.1570
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8584 - loss: 0.1370 - val_accuracy: 0.8722 - val_loss: 0.1078
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9026 - loss: 0.0883 - val_accuracy: 0.8762 - val_loss: 0.0972
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9245 - loss: 0.0685 - val_accuracy: 0.8817 - val_loss: 0.0890
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9327 - loss: 0.0584 - val_accuracy: 0.8704 - val_loss: 0.0945
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9383 - loss: 0.0516 - val_accuracy: 0.8779 - val_loss: 0.0913
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9562 - loss: 0.0418 - val_accuracy: 0.8858 - val_loss: 0.0853
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9622 - loss: 0.0369 - val_accuracy: 0.8850 - val_loss: 0.0853
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - accuracy: 0.9677 - loss: 0.0325 - val_accuracy: 0.8795 - val_loss: 0.0863
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9741 - loss: 0.0264 - val_accuracy: 0.8817 - val_loss: 0.0885
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9775 - loss: 0.0242 - val_accuracy: 0.8740 - val_loss: 0.0907
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9768 - loss: 0.0235 - val_accuracy: 0.8760 - val_loss: 0.0955
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9798 - loss: 0.0223 - val_accuracy: 0.8776 - val_loss: 0.0924
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9872 - loss: 0.0158 - val_accuracy: 0.8768 - val_loss: 0.0942
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9872 - loss: 0.0147 - val_accuracy: 0.8760 - val_loss: 0.0958
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9886 - loss: 0.0132 - val_accuracy: 0.8747 - val_loss: 0.0979
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9870 - loss: 0.0142 - val_accuracy: 0.8767 - val_loss: 0.0980
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9907 - loss: 0.0105 - val_accuracy: 0.8755 - val_loss: 0.0994
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 53ms/step - accuracy: 0.9916 - loss: 0.0099 - val_accuracy: 0.8734 - val_loss: 0.1014
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 57ms/step - accuracy: 0.9930 - loss: 0.0082 - val_accuracy: 0.8731 - val_loss: 0.1040
```

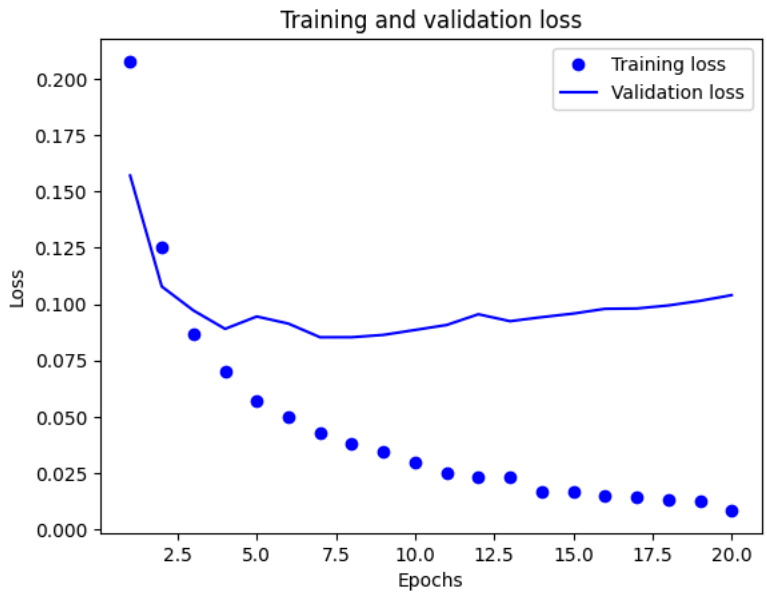
```
historydict_MSE_647 = history_MSE_647.history
historydict_MSE_647.keys()
```

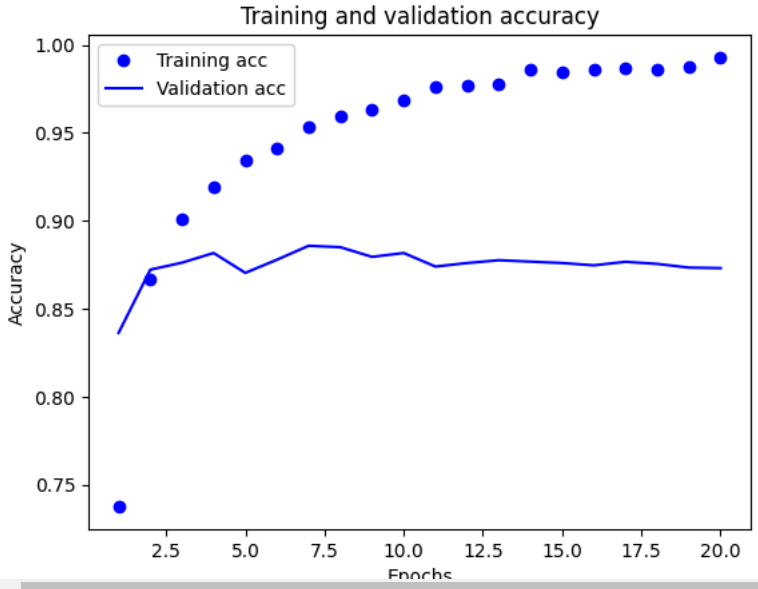
```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
import matplotlib.pyplot as plot647
loss_value_MSE_16_3 = historydict_MSE_647["loss"]
val_loss_value_MSE_16_3 = historydict_MSE_647["val_loss"]
epochs_MSE = range(1, len(loss_value_MSE_16_3) + 1)
plot647.plot(epochs_MSE, loss_value_MSE_16_3, "bo", label="Training loss")
plot647.plot(epochs_MSE, val_loss_value_MSE_16_3, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_MSE = historydict_MSE_647["accuracy"]
val_acc_MSE = historydict_MSE_647["val_accuracy"]
plot647.plot(epochs_MSE, acc_MSE, "bo", label="Training acc")
plot647.plot(epochs_MSE, val_acc_MSE, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
```

```
plot647.legend()
plot647.show()
```






```
MSE_model_16_647.fit(x_train, y_train, epochs=8, batch_size=512)
results_MSE_647 = MSE_model_16_647.evaluate(x_test, y_test)
results_MSE_647
```

```
Epoch 1/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9416 - loss: 0.0497
Epoch 2/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 30ms/step - accuracy: 0.9578 - loss: 0.0371
Epoch 3/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9645 - loss: 0.0319
Epoch 4/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - accuracy: 0.9697 - loss: 0.0286
Epoch 5/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9750 - loss: 0.0239
Epoch 6/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9761 - loss: 0.0235
Epoch 7/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9798 - loss: 0.0198
Epoch 8/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9814 - loss: 0.0188
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8644 - loss: 0.1124
[0.11023791134357452, 0.8670799732208252]
```

```
MSE_model_16_647.predict(x_test)
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step
array([[0.02738004],
       [1.        ],
       [0.15260053],
       ...,
       [0.04235547],
```

```
            [0.01303633],
            [0.7670964 ]], dtype=float32)
```

```
tanh_647 = keras.Sequential([
    layers.Dense(16, activation="tanh"),
    layers.Dense(1, activation="sigmoid")
])

tanh_647.compile(optimizer='rmsprop',
              loss='mse',
              metrics=['accuracy'])

x_val_tanh = x_train[:10000]
partial_x_train = x_train[10000:]

y_val_tanh = y_train[:10000]
partial_y_train = y_train[10000:]


historytanh_model = tanh_647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val_tanh, y_val_tanh))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.6904 - loss: 0.2055 - val_accuracy: 0.8613 - val_loss: 0.1313
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.8916 - loss: 0.1132 - val_accuracy: 0.8815 - val_loss: 0.1046
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9126 - loss: 0.0866 - val_accuracy: 0.8865 - val_loss: 0.0940
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.9255 - loss: 0.0718 - val_accuracy: 0.8761 - val_loss: 0.0941
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9374 - loss: 0.0617 - val_accuracy: 0.8865 - val_loss: 0.0858
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9470 - loss: 0.0542 - val_accuracy: 0.8768 - val_loss: 0.0916
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9458 - loss: 0.0512 - val_accuracy: 0.8845 - val_loss: 0.0861
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9536 - loss: 0.0452 - val_accuracy: 0.8877 - val_loss: 0.0834
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9598 - loss: 0.0423 - val_accuracy: 0.8855 - val_loss: 0.0846
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9632 - loss: 0.0379 - val_accuracy: 0.8727 - val_loss: 0.0944
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9678 - loss: 0.0357 - val_accuracy: 0.8725 - val_loss: 0.0920
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9708 - loss: 0.0317 - val_accuracy: 0.8777 - val_loss: 0.0869
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9736 - loss: 0.0298 - val_accuracy: 0.8672 - val_loss: 0.1000
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - accuracy: 0.9716 - loss: 0.0311 - val_accuracy: 0.8714 - val_loss: 0.0934
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9784 - loss: 0.0258 - val_accuracy: 0.8800 - val_loss: 0.0880
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9814 - loss: 0.0234 - val_accuracy: 0.8765 - val_loss: 0.0935
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9842 - loss: 0.0217 - val_accuracy: 0.8766 - val_loss: 0.0905
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9852 - loss: 0.0194 - val_accuracy: 0.8747 - val_loss: 0.0924
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9860 - loss: 0.0189 - val_accuracy: 0.8776 - val_loss: 0.0919
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9886 - loss: 0.0165 - val_accuracy: 0.8732 - val_loss: 0.0942
```

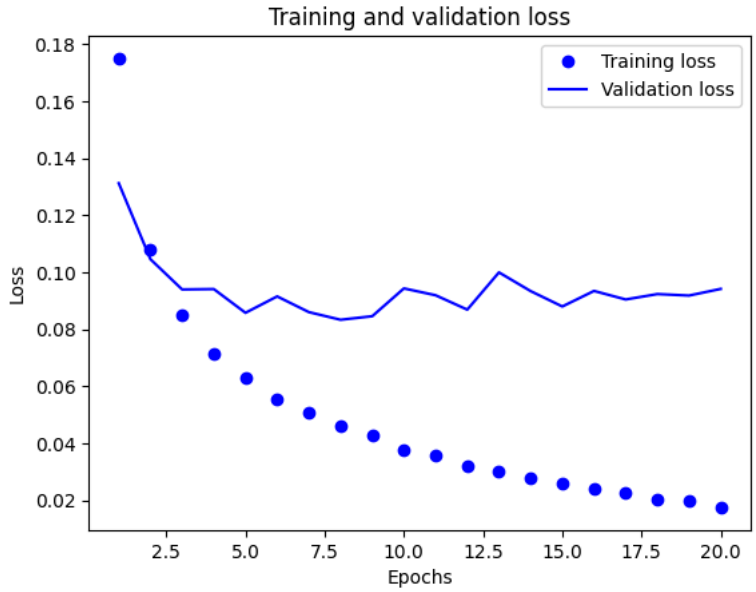
```
historydict_tanh_647 = historytanh_model.history
historydict_tanh_647.keys()
```

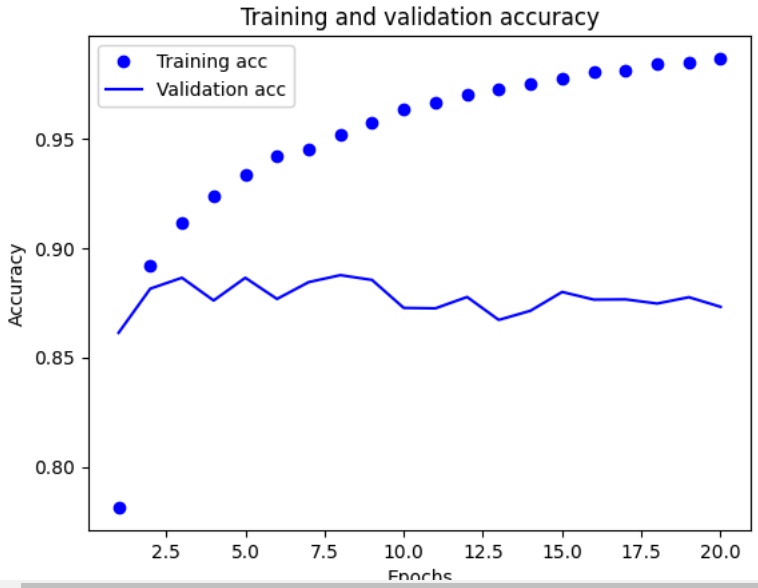
```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
loss_value_tanh_647= historydict_tanh_647["loss"]
val_loss_value_tanh_647 = historydict_tanh_647["val_loss"]
epochs_tanh = range(1, len(loss_value_tanh_647) + 1)
plot647.plot(epochs_tanh, loss_value_tanh_647, "bo", label="Training loss")
plot647.plot(epochs_tanh, val_loss_value_tanh_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_tanh = historydict_tanh_647["accuracy"]
```

```
val_acc_tanh = historydict_tanh_647["val_accuracy"]
plot647.plot(epochs_tanh, acc_tanh, "bo", label="Training acc")
plot647.plot(epochs_tanh, val_acc_tanh, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
tanh_647.fit(x_train, y_train, epochs=8, batch_size=512)
results_tanh_647 = tanh_647.evaluate(x_test, y_test)
results_tanh_647
```

```
Epoch 1/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.9432 - loss: 0.0486
Epoch 2/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9560 - loss: 0.0391
Epoch 3/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9597 - loss: 0.0361
Epoch 4/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9659 - loss: 0.0323
Epoch 5/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9667 - loss: 0.0317
Epoch 6/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9700 - loss: 0.0290
Epoch 7/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9731 - loss: 0.0273
Epoch 8/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.9763 - loss: 0.0249
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8633 - loss: 0.1067
[0.10421992093324661, 0.8677999973297119]
```

Adam Operator with 16 units and 3-layers

```
adam_647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
     layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

adam_647.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

x_adam_647 = x_train[:10000]
partial_x_train_16 = x_train[10000:]

y_adam_647 = y_train[:10000]
partial_y_train_16 = y_train[10000:]


historyadam_647 = adam_647.fit(partial_x_train_16,
                    partial_y_train_16,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_adam_647, y_adam_647))
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 5s 97ms/step - accuracy: 0.5659 - loss: 0.6566 - val_accuracy: 0.8370 - val_loss: 0.4950
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 36ms/step - accuracy: 0.8862 - loss: 0.4025 - val_accuracy: 0.8845 - val_loss: 0.2941
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9402 - loss: 0.1889 - val_accuracy: 0.8831 - val_loss: 0.2868
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9668 - loss: 0.1182 - val_accuracy: 0.8801 - val_loss: 0.3231
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9833 - loss: 0.0686 - val_accuracy: 0.8757 - val_loss: 0.3780
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9908 - loss: 0.0505 - val_accuracy: 0.8736 - val_loss: 0.4215
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9952 - loss: 0.0327 - val_accuracy: 0.8702 - val_loss: 0.4689
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9983 - loss: 0.0174 - val_accuracy: 0.8702 - val_loss: 0.5180
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 44ms/step - accuracy: 0.9993 - loss: 0.0122 - val_accuracy: 0.8710 - val_loss: 0.5599
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9995 - loss: 0.0084 - val_accuracy: 0.8698 - val_loss: 0.5946
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9993 - loss: 0.0079 - val_accuracy: 0.8684 - val_loss: 0.6258
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9999 - loss: 0.0041 - val_accuracy: 0.8684 - val_loss: 0.6549
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9996 - loss: 0.0036 - val_accuracy: 0.8690 - val_loss: 0.6843
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 1.0000 - loss: 0.0022 - val_accuracy: 0.8690 - val_loss: 0.7094
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.8683 - val_loss: 0.7305
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 1.0000 - loss: 0.0014 - val_accuracy: 0.8685 - val_loss: 0.7512
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.8683 - val_loss: 0.7680
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 1.0000 - loss: 0.0010 - val_accuracy: 0.8687 - val_loss: 0.7867
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 36ms/step - accuracy: 1.0000 - loss: 8.9264e-04 - val_accuracy: 0.8678 - val_loss: 0.8024
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 1.0000 - loss: 7.6895e-04 - val_accuracy: 0.8679 - val_loss: 0.8165
```

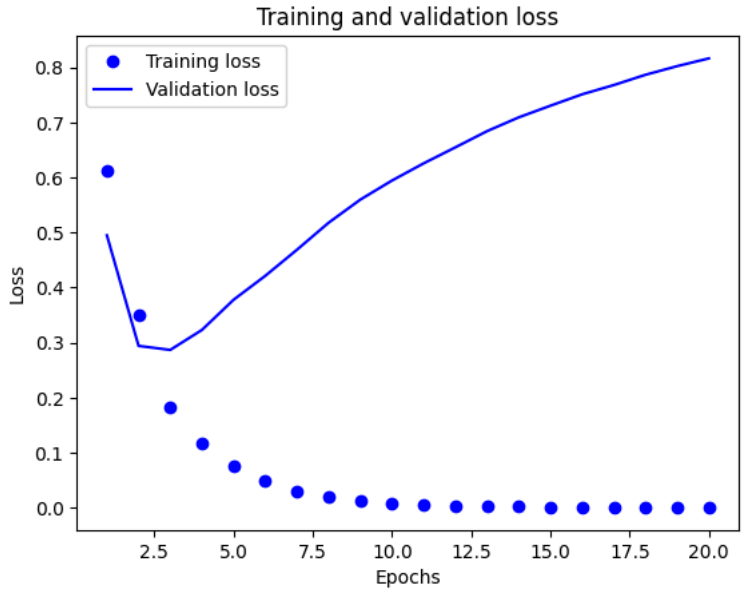
```
historydict_adam_647 = historyadam_647.history
historydict_adam_647.keys()
```

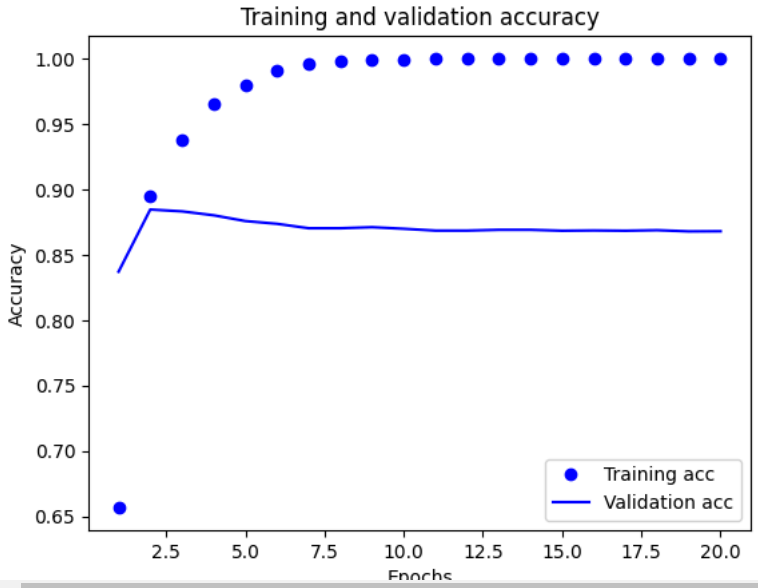
```
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
loss_value_adam_647 = historydict_adam_647["loss"]
val_loss_value_adam_647 = historydict_adam_647["val_loss"]
epochs_adam = range(1, len(loss_value_adam_647) + 1)
plot647.plot(epochs_adam, loss_value_adam_647, "bo", label="Training loss")
plot647.plot(epochs_adam, val_loss_value_adam_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_adam = historydict_adam_647["accuracy"]
```

```
val_acc_adam = historydict_adam_647["val_accuracy"]
plot647.plot(epochs_adam, acc_adam, "bo", label="Training acc")
plot647.plot(epochs_adam, val_acc_adam, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
adam_647.fit(x_train, y_train, epochs=4, batch_size=512)
results_adam = adam_647.evaluate(x_test, y_test)
results_adam
```

```
Epoch 1/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9351 - loss: 0.2823
Epoch 2/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - accuracy: 0.9621 - loss: 0.1176
Epoch 3/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - accuracy: 0.9796 - loss: 0.0721
Epoch 4/4
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9901 - loss: 0.0430
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8579 - loss: 0.5440
[0.538260817527771, 0.8588399887084961]
```

Regularization model with 16 units and 2-layers

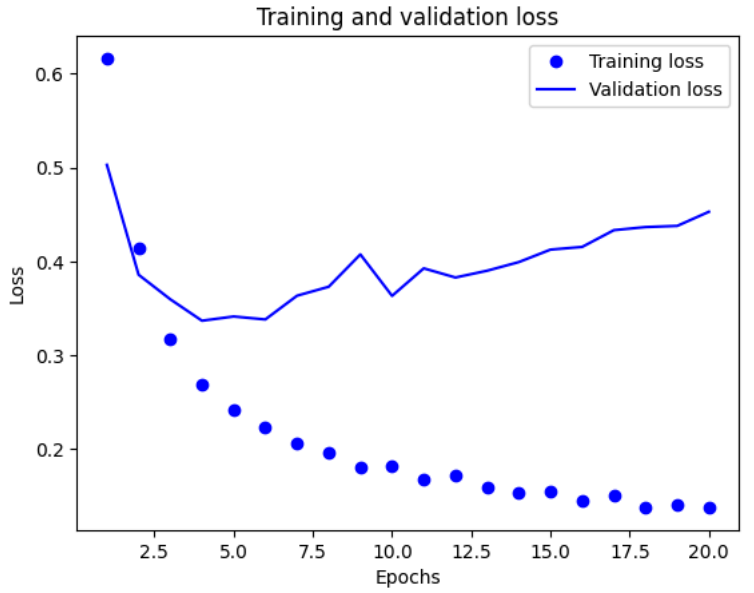
```
from tensorflow.keras import regularizers
regularization647 = keras.Sequential([
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.001)),
    layers.Dense(1, activation="sigmoid")
])
```

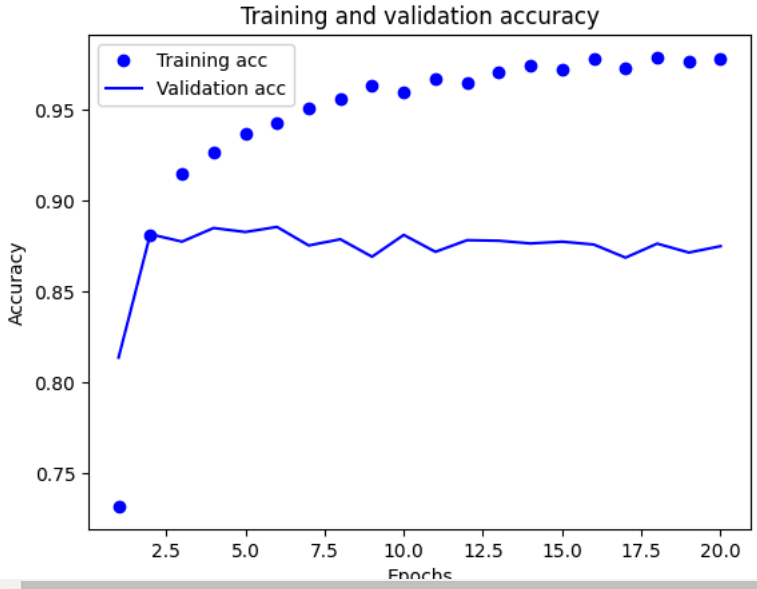
```
regularization647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

history_regularization647 = regularization647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
historydict_regularization647 = history_regularization647.history
historydict_regularization647.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 91ms/step - accuracy: 0.6416 - loss: 0.6754 - val_accuracy: 0.8138 - val_loss: 0.5027
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 55ms/step - accuracy: 0.8786 - loss: 0.4362 - val_accuracy: 0.8817 - val_loss: 0.3857
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9156 - loss: 0.3234 - val_accuracy: 0.8776 - val_loss: 0.3596
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9282 - loss: 0.2709 - val_accuracy: 0.8851 - val_loss: 0.3368
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9373 - loss: 0.2428 - val_accuracy: 0.8829 - val_loss: 0.3413
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9481 - loss: 0.2168 - val_accuracy: 0.8857 - val_loss: 0.3381
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9551 - loss: 0.2004 - val_accuracy: 0.8755 - val_loss: 0.3634
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9647 - loss: 0.1829 - val_accuracy: 0.8789 - val_loss: 0.3729
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9694 - loss: 0.1755 - val_accuracy: 0.8693 - val_loss: 0.4074
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9616 - loss: 0.1791 - val_accuracy: 0.8813 - val_loss: 0.3632
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - accuracy: 0.9717 - loss: 0.1609 - val_accuracy: 0.8720 - val_loss: 0.3925
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 51ms/step - accuracy: 0.9693 - loss: 0.1677 - val_accuracy: 0.8784 - val_loss: 0.3828
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9754 - loss: 0.1504 - val_accuracy: 0.8781 - val_loss: 0.3900
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9809 - loss: 0.1415 - val_accuracy: 0.8766 - val_loss: 0.3992
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9786 - loss: 0.1446 - val_accuracy: 0.8776 - val_loss: 0.4125
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9824 - loss: 0.1386 - val_accuracy: 0.8760 - val_loss: 0.4154
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9821 - loss: 0.1370 - val_accuracy: 0.8688 - val_loss: 0.4332
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9853 - loss: 0.1283 - val_accuracy: 0.8765 - val_loss: 0.4365
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9844 - loss: 0.1299 - val_accuracy: 0.8716 - val_loss: 0.4377
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9882 - loss: 0.1228 - val_accuracy: 0.8751 - val_loss: 0.4528
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
loss_valu_647 = historydict_regularization647["loss"]
val_loss_value_r_647 = historydict_regularization647["val_loss"]
epochs_r = range(1, len(loss_valu_647) + 1)
plot647.plot(epochs_r, loss_valu_647, "bo", label="Training loss")
plot647.plot(epochs_r, val_loss_value_r_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_r = historydict_regularization647["accuracy"]
val_acc_r = historydict_regularization647["val_accuracy"]
plot647.plot(epochs_r, acc_r, "bo", label="Training acc")
plot647.plot(epochs_r, val_acc_r, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
regularization647.fit(x_train, y_train, epochs=8, batch_size=512)
results_regularization_647 = regularization647.evaluate(x_test, y_test)
results_regularization_647
```

```
Epoch 1/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.9369 - loss: 0.2602
Epoch 2/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - accuracy: 0.9509 - loss: 0.2038
Epoch 3/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 30ms/step - accuracy: 0.9542 - loss: 0.1930
Epoch 4/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - accuracy: 0.9592 - loss: 0.1775
Epoch 5/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.9601 - loss: 0.1764
Epoch 6/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9655 - loss: 0.1641
Epoch 7/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.9685 - loss: 0.1595
Epoch 8/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 39ms/step - accuracy: 0.9686 - loss: 0.1592
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8611 - loss: 0.4436
[0.4401252865791321, 0.864359974861145]
```

Dropout function with 16 units and 3-layers

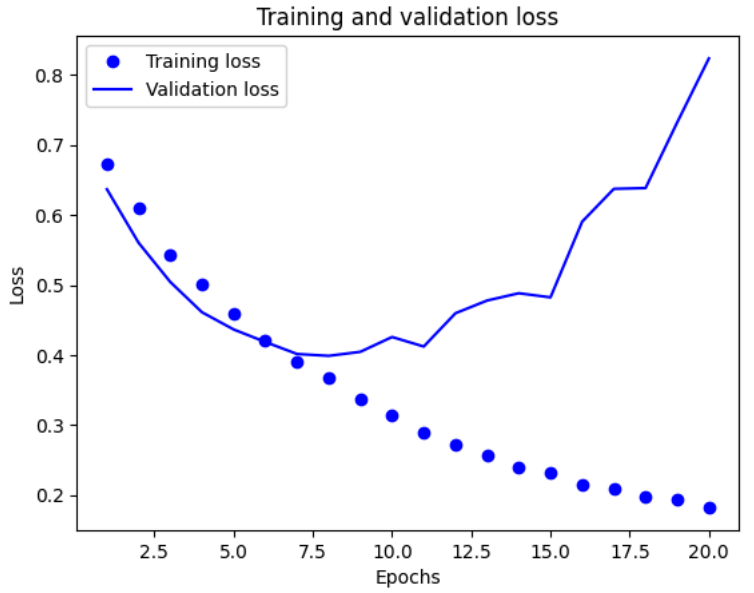
```
from tensorflow.keras import regularizers
Dropout647 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
```

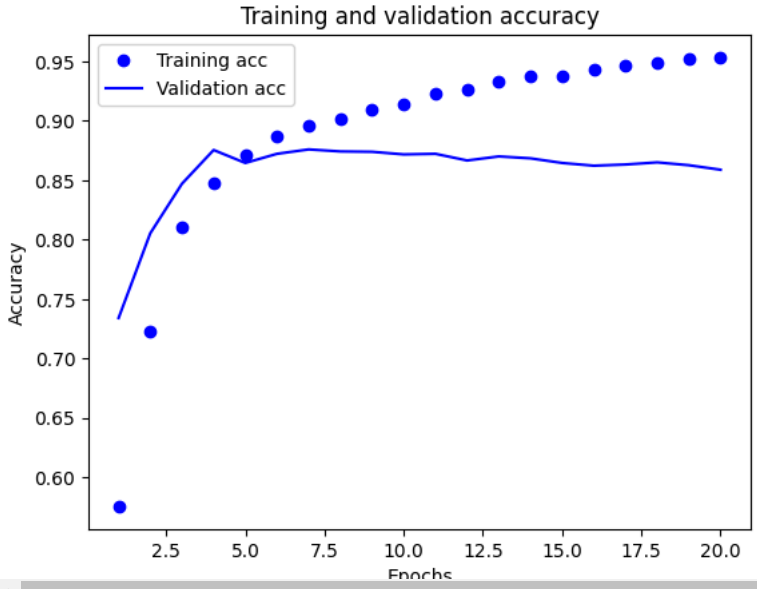
```
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
Dropout647.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])

history_Dropout_647 = Dropout647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
historydict_Dropout_647 = history_Dropout_647.history
historydict_Dropout_647.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 75ms/step - accuracy: 0.5375 - loss: 0.6864 - val_accuracy: 0.7341 - val_loss: 0.6369
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.7033 - loss: 0.6212 - val_accuracy: 0.8054 - val_loss: 0.5604
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.8026 - loss: 0.5546 - val_accuracy: 0.8471 - val_loss: 0.5044
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.8453 - loss: 0.5092 - val_accuracy: 0.8755 - val_loss: 0.4611
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 54ms/step - accuracy: 0.8711 - loss: 0.4656 - val_accuracy: 0.8646 - val_loss: 0.4365
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 38ms/step - accuracy: 0.8860 - loss: 0.4254 - val_accuracy: 0.8723 - val_loss: 0.4187
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.8965 - loss: 0.3913 - val_accuracy: 0.8760 - val_loss: 0.4015
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.8983 - loss: 0.3724 - val_accuracy: 0.8743 - val_loss: 0.3988
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9159 - loss: 0.3349 - val_accuracy: 0.8740 - val_loss: 0.4046
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9161 - loss: 0.3123 - val_accuracy: 0.8718 - val_loss: 0.4257
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9258 - loss: 0.2880 - val_accuracy: 0.8723 - val_loss: 0.4121
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9275 - loss: 0.2739 - val_accuracy: 0.8667 - val_loss: 0.4597
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9323 - loss: 0.2582 - val_accuracy: 0.8701 - val_loss: 0.4778
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.9380 - loss: 0.2364 - val_accuracy: 0.8685 - val_loss: 0.4883
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - accuracy: 0.9389 - loss: 0.2295 - val_accuracy: 0.8646 - val_loss: 0.4824
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9468 - loss: 0.2095 - val_accuracy: 0.8623 - val_loss: 0.5905
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.9446 - loss: 0.2147 - val_accuracy: 0.8633 - val_loss: 0.6374
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.9450 - loss: 0.2081 - val_accuracy: 0.8651 - val_loss: 0.6385
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.9521 - loss: 0.1953 - val_accuracy: 0.8627 - val_loss: 0.7323
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9532 - loss: 0.1844 - val_accuracy: 0.8589 - val_loss: 0.8238
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
loss_val_647 = historydict_Dropout_647["loss"]
val_loss_val_d_647 = historydict_Dropout_647["val_loss"]
epochs_d = range(1, len(loss_val_647) + 1)
plot647.plot(epochs_d, loss_val_647, "bo", label="Training loss")
plot647.plot(epochs_d, val_loss_val_d_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_d = historydict_Dropout_647["accuracy"]
val_acc_d = historydict_Dropout_647["val_accuracy"]
plot647.plot(epochs_d, acc_d, "bo", label="Training acc")
plot647.plot(epochs_d, val_acc_d, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```






```
Dropout647.fit(x_train, y_train, epochs=8, batch_size=512)
results_Dropout647 = Dropout647.evaluate(x_test, y_test)
results_Dropout647
```

```
Epoch 1/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 39ms/step - accuracy: 0.9006 - loss: 0.3851
Epoch 2/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - accuracy: 0.9072 - loss: 0.3205
Epoch 3/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - accuracy: 0.9160 - loss: 0.2928
Epoch 4/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.9256 - loss: 0.2600
Epoch 5/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9275 - loss: 0.2503
Epoch 6/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 3s 29ms/step - accuracy: 0.9295 - loss: 0.2453
Epoch 7/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 28ms/step - accuracy: 0.9317 - loss: 0.2314
Epoch 8/8
49/49 ━━━━━━━━━━━━━━━━━━━━ 2s 31ms/step - accuracy: 0.9354 - loss: 0.2198
782/782 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8561 - loss: 0.5605
[0.5733245015144348, 0.8596000075340271]
```

Training model with hyper tuned parameters with 32 units and 3 -layers

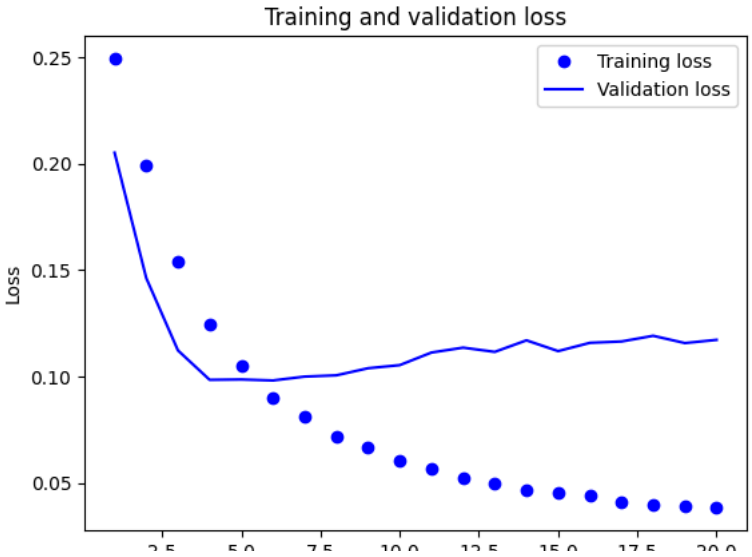
```
from tensorflow.keras import regularizers
Hyper647 = keras.Sequential([
    layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(32, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu",kernel_regularizer=regularizers.l2(0.0001)),
```

```
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
Hyper647.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])

history_Hyper647 = Hyper647.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
history_dictHyper647 = history_Hyper647.history
history_dictHyper647.keys()
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 75ms/step - accuracy: 0.5316 - loss: 0.2579 - val_accuracy: 0.8156 - val_loss: 0.2052
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.6984 - loss: 0.2112 - val_accuracy: 0.8459 - val_loss: 0.1462
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 74ms/step - accuracy: 0.8075 - loss: 0.1615 - val_accuracy: 0.8715 - val_loss: 0.1123
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.8618 - loss: 0.1254 - val_accuracy: 0.8836 - val_loss: 0.0984
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - accuracy: 0.8863 - loss: 0.1060 - val_accuracy: 0.8847 - val_loss: 0.0986
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9123 - loss: 0.0892 - val_accuracy: 0.8858 - val_loss: 0.0981
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9201 - loss: 0.0805 - val_accuracy: 0.8869 - val_loss: 0.1000
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - accuracy: 0.9311 - loss: 0.0730 - val_accuracy: 0.8878 - val_loss: 0.1006
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 76ms/step - accuracy: 0.9385 - loss: 0.0683 - val_accuracy: 0.8881 - val_loss: 0.1039
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.9467 - loss: 0.0602 - val_accuracy: 0.8860 - val_loss: 0.1053
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9526 - loss: 0.0564 - val_accuracy: 0.8804 - val_loss: 0.1113
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - accuracy: 0.9553 - loss: 0.0527 - val_accuracy: 0.8800 - val_loss: 0.1136
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9587 - loss: 0.0508 - val_accuracy: 0.8820 - val_loss: 0.1116
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9637 - loss: 0.0463 - val_accuracy: 0.8773 - val_loss: 0.1170
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9657 - loss: 0.0445 - val_accuracy: 0.8832 - val_loss: 0.1119
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 56ms/step - accuracy: 0.9678 - loss: 0.0430 - val_accuracy: 0.8783 - val_loss: 0.1158
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 65ms/step - accuracy: 0.9686 - loss: 0.0410 - val_accuracy: 0.8817 - val_loss: 0.1165
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.9729 - loss: 0.0381 - val_accuracy: 0.8791 - val_loss: 0.1191
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9729 - loss: 0.0380 - val_accuracy: 0.8811 - val_loss: 0.1157
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.9710 - loss: 0.0395 - val_accuracy: 0.8795 - val_loss: 0.1172
dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

```
loss_va_h_647 = history_dictHyper647["loss"]
val_loss_va_h_647 = history_dictHyper647["val_loss"]
epochs_h = range(1, len(loss_va_h_647) + 1)
plot647.plot(epochs_h, loss_va_h_647, "bo", label="Training loss")
plot647.plot(epochs_h, val_loss_va_h_647, "b", label="Validation loss")
plot647.title("Training and validation loss")
plot647.xlabel("Epochs")
plot647.ylabel("Loss")
plot647.legend()
plot647.show()
```

```
plot647.clf()
acc_h = history_dictHyper647["accuracy"]
val_acc_h = history_dictHyper647["val_accuracy"]
plot647.plot(epochs_h, acc_h, "bo", label="Training acc")
plot647.plot(epochs_h, val_acc_h, "b", label="Validation acc")
plot647.title("Training and validation accuracy")
plot647.xlabel("Epochs")
plot647.ylabel("Accuracy")
plot647.legend()
plot647.show()
```


Training and validation loss
Training loss
Validation loss
Loss
0.25
0.20
0.15
0.10
0.05