|  |
| --- |
| **\*\***ASSIGNMENT 1: NEURAL NETWORKS**\*\***  Name:Rishitha Rapuri  Student id: 811283595      *#The IMDb dataset—a collection of favorable and negative movie reviews—was used for this investigation. The dataset was large, co* |

In [83]:

|  |
| --- |
| *#Install Tensorflow* |

In [ ]:

|  |
| --- |
| **!**pip install tensorflow |

In [80]:

Requirement already satisfied: tensorflow in c:\users\rishi\downloads\anaconda\lib\site-packages (2.15.0)

Requirement already satisfied: tensorflow-intel==2.15.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow)

(2.15.0)

Requirement already satisfied: absl-py>=1.0.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.1

5.0->tensorflow) (2.1.0) Requirement already satisfied: astunparse>=1.6.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel== 2.15.0->tensorflow) (1.6.3) Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-inte l==2.15.0->tensorflow) (23.5.26)

Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.5.4)

Requirement already satisfied: google-pasta>=0.1.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel

==2.15.0->tensorflow) (0.2.0)

Requirement already satisfied: h5py>=2.9.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0

->tensorflow) (3.9.0)

Requirement already satisfied: libclang>=13.0.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==

2.15.0->tensorflow) (16.0.6) Requirement already satisfied: ml-dtypes~=0.2.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==

2.15.0->tensorflow) (0.2.0) Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-inte l==2.15.0->tensorflow) (1.24.3)

Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel== 2.15.0->tensorflow) (3.3.0) Requirement already satisfied: packaging in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0-> 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 c:\users\rish i\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (4.25.3)

Requirement already satisfied: setuptools in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0-

>tensorflow) (68.0.0)

Requirement already satisfied: six>=1.12.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0

->tensorflow) (1.16.0)

Requirement already satisfied: termcolor>=1.1.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==

2.15.0->tensorflow) (2.4.0) Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflowintel==2.15.0->tensorflow) (4.7.1)

Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel

==2.15.0->tensorflow) (1.14.1)

Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.31.0)

Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel

==2.15.0->tensorflow) (1.62.0)

Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-i ntel==2.15.0->tensorflow) (2.15.2)

Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from t ensorflow-intel==2.15.0->tensorflow) (2.15.0)

Requirement already satisfied: keras<2.16,>=2.15.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorflow-intel ==2.15.0->tensorflow) (2.15.0)

Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from astunparse>=1.6.0

->tensorflow-intel==2.15.0->tensorflow) (0.38.4) Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorboard<2. 16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.28.1)

Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorb oard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (1.2.0)

Requirement already satisfied: markdown>=2.6.8 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorboard<2.16,>= 2.15->tensorflow-intel==2.15.0->tensorflow) (3.4.1) Requirement already satisfied: requests<3,>=2.21.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorboard<2.1

6,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.31.0)

Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (fro m tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.7.2)

Requirement already satisfied: werkzeug>=1.0.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from tensorboard<2.16,>=

2.15->tensorflow-intel==2.15.0->tensorflow) (2.2.3) Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from google-auth<

3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (5.3.2) Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from google-auth<3,

>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.2.8)

Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\rishi\downloads\anaconda\lib\site-packages (from google-auth<3,>=1.6.3>tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (4.9)

Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from google-auth

-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (1.3.1)

Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\rishi\downloads\anaconda\lib\site-packages (from requests<3, >=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.0.4)

Requirement already satisfied: idna<4,>=2.5 in c:\users\rishi\downloads\anaconda\lib\site-packages (from requests<3,>=2.21.0->te nsorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.4)

Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from requests<3,>=2.2 1.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (1.26.16) Requirement already satisfied: certifi>=2017.4.17 in c:\users\rishi\downloads\anaconda\lib\site-packages (from requests<3,>=2.2

1.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2023.7.22) Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\rishi\downloads\anaconda\lib\site-packages (from werkzeug>=1.0.1->t ensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.1.1)

Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\rishi\downloads\anaconda\lib\site-packages (from pyasn1-modules>

=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.4.8) Requirement already satisfied: oauthlib>=3.0.0 in c:\users\rishi\downloads\anaconda\lib\site-packages (from requests-oauthlib>=

0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.2.2)

|  |
| --- |
| *#The IMDB Dataset*  *#We'll be working with the "IMDB dataset," which is a collection of 50,000 highly polarized evaluations from the Internet Movie D*  **from** tensorflow.keras.datasets **import** imdb  (train\_data, train\_labels), (test\_data, test\_labels) **=** imdb**.**load\_data( num\_words**=**10000) |

In [2]:

WARNING:tensorflow:From C:\Users\rishi\Downloads\anaconda\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse\_ softmax\_cross\_entropy is deprecated. Please use tf.compat.v1.losses.sparse\_softmax\_cross\_entropy instead.

In [3]: 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, 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, 20

25, 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, 4

8, 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, 7

1, 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])

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, 5, 207, 110, 3103, 21, 14, 69, 188, 8, 30, 23, 7, 4, 249, 126, 93, 4, 114, 9, 2300, 1523, 5, 647, 4, 116, 9, 35, 8163, 4, 2

29, 9, 340, 1322, 4, 118, 9, 4, 130, 4901, 19, 4, 1002, 5, 89, 29, 952, 46, 37, 4, 455, 9, 45, 43, 38, 1543, 1905, 398, 4, 1649,

26, 6853, 5, 163, 11, 3215, 2, 4, 1153, 9, 194, 775, 7, 8255, 2, 349, 2637, 148, 605, 2, 8003, 15, 123, 125, 68, 2, 6853, 15, 34

9, 165, 4362, 98, 5, 4, 228, 9, 43, 2, 1157, 15, 299, 120, 5, 120, 174, 11, 220, 175, 136, 50, 9, 4373, 228, 8255, 5, 2, 656, 24 5, 2350, 5, 4, 9837, 131, 152, 491, 18, 2, 32, 7464, 1212, 14, 9, 6, 371, 78, 22, 625, 64, 1382, 9, 8, 168, 145, 23, 4, 1690, 1

5, 16, 4, 1355, 5, 28, 6, 52, 154, 462, 33, 89, 78, 285, 16, 145, 95])

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, 43, 1829, 296, 4, 86, 320, 35, 534, 19, 263, 4821, 1301, 4, 1873, 33, 89, 78, 12, 66, 16, 4, 360, 7, 4, 58, 316, 334, 11, 4,

1716, 43, 645, 662, 8, 257, 85, 1200, 42, 1228, 2578, 83, 68, 3912, 15, 36, 165, 1539, 278, 36, 69, 2, 780, 8, 106, 14, 6905, 13 38, 18, 6, 22, 12, 215, 28, 610, 40, 6, 87, 326, 23, 2300, 21, 23, 22, 12, 272, 40, 57, 31, 11, 4, 22, 47, 6, 2307, 51, 9, 170,

23, 595, 116, 595, 1352, 13, 191, 79, 638, 89, 2, 14, 9, 8, 106, 607, 624, 35, 534, 6, 227, 7, 129, 113])

...

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, 28, 57, 1099, 21, 8, 140, 8, 703, 5, 2, 84, 56, 18, 1644, 14, 9, 31, 7, 4, 9406, 1209, 2295, 2, 1008, 18, 6, 20, 207, 110, 5

63, 12, 8, 2901, 2, 8, 97, 6, 20, 53, 4767, 74, 4, 460, 364, 1273, 29, 270, 11, 960, 108, 45, 40, 29, 2961, 395, 11, 6, 4065, 50

0, 7, 2, 89, 364, 70, 29, 140, 4, 64, 4780, 11, 4, 2678, 26, 178, 4, 529, 443, 2, 5, 27, 710, 117, 2, 8123, 165, 47, 84, 37, 13

1, 818, 14, 595, 10, 10, 61, 1242, 1209, 10, 10, 288, 2260, 1702, 34, 2901, 2, 4, 65, 496, 4, 231, 7, 790, 5, 6, 320, 234, 2766, 234, 1119, 1574, 7, 496, 4, 139, 929, 2901, 2, 7750, 5, 4241, 18, 4, 8497, 2, 250, 11, 1818, 7561, 4, 4217, 5408, 747, 1115, 37

2, 1890, 1006, 541, 9303, 7, 4, 59, 2, 4, 3586, 2])

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, 19 7, 12, 16, 43, 23, 2, 5, 62, 30, 145, 402, 11, 4131, 51, 575, 32, 61, 369, 71, 66, 770, 12, 1054, 75, 100, 2198, 8, 4, 105, 37,

69, 147, 712, 75, 3543, 44, 257, 390, 5, 69, 263, 514, 105, 50, 286, 1814, 23, 4, 123, 13, 161, 40, 5, 421, 4, 116, 16, 897, 13,

2, 40, 319, 5872, 112, 6700, 11, 4803, 121, 25, 70, 3468, 4, 719, 3798, 13, 18, 31, 62, 40, 8, 7200, 4, 2, 7, 14, 123, 5, 942, 2 5, 8, 721, 12, 145, 5, 202, 12, 160, 580, 202, 12, 6, 52, 58, 2, 92, 401, 728, 12, 39, 14, 251, 8, 15, 251, 5, 2, 12, 38, 84, 8

0, 124, 12, 9, 23])

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,

1310, 9, 115, 50, 305, 12, 47, 4, 168, 5, 235, 7, 38, 111, 699, 102, 7, 4, 4039, 9245, 9, 24, 6, 78, 1099, 17, 2345, 2, 21, 27, 9685, 6139, 5, 2, 1603, 92, 1183, 4, 1310, 7, 4, 204, 42, 97, 90, 35, 221, 109, 29, 127, 27, 118, 8, 97, 12, 157, 21, 6789, 2,

9, 6, 66, 78, 1099, 4, 631, 1191, 5, 2642, 272, 191, 1070, 6, 7585, 8, 2197, 2, 2, 544, 5, 383, 1271, 848, 1468, 2, 497, 2, 8, 1 597, 8778, 2, 21, 60, 27, 239, 9, 43, 8368, 209, 405, 10, 10, 12, 764, 40, 4, 248, 20, 12, 16, 5, 174, 1791, 72, 7, 51, 6, 1739,

22, 4, 204, 131, 9])] (25000,)

In [5]: train\_labels[0]

|  |  |
| --- | --- |
| Out[5]: | 1 |

In [6]: len(train\_labels)

|  |  |
| --- | --- |
| Out[6]: | 25000 |

In [7]: test\_labels[0]

|  |  |
| --- | --- |
| Out[7]: | 0 |

In [8]: max([max(sequence) **for** sequence **in** test\_data])

|  |  |
| --- | --- |
| Out[8]: | 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]]) |

In [9]:

In [10]: decoded\_review

"? this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you c

Out[10]: ould just imagine being there robert ? is an amazing actor and now the same being director ? father came from the same scottish island as myself so i loved the fact there was a real connection with this film the witty remarks throughout the film were great it was just brilliant so much that i bought the film as soon as it was released for ? and would recommend it to everyone to watc h and the fly fishing was amazing really cried at the end it was so sad and you know what they say if you cry at a film it must have been good and this definitely was also ? to the two little boy's that played the ? of norman and paul they were just brilli ant children are often left out of the ? list i think because the stars that play them all grown up are such a big profile for t he whole film but these children are amazing and should be praised for what they have done don't you think the whole story was s o lovely because it was true and was someone's life after all that was shared with us all"

In [12]: *##Preparation of the Data* **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

In [13]: *#Data Vectorization* x\_train **=** vectorize\_sequences(train\_data) x\_test **=** vectorize\_sequences(test\_data)

In [14]: x\_train[0]

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

Out[14]:

In [15]: x\_test[0]

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

Out[15]:

In [16]: *#label vectorization*

y\_train **=** np**.**asarray(train\_labels)**.**astype("float32") y\_test **=** np**.**asarray(test\_labels)**.**astype("float32")

|  |
| --- |
| *#Building the model using relu and compiling it*  *#Our input data is simply vectors, and our labels are scalars (1s and 0s): this is the simplest configuration you'll ever see. A #Finally, select a loss function and an optimizer. Because we are dealing with a binary classification problem and our network's*  **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") ]) |

In [17]:

WARNING:tensorflow:From C:\Users\rishi\Downloads\anaconda\Lib\site-packages\keras\src\backend.py:873: The name tf.get\_default\_gr aph is deprecated. Please use tf.compat.v1.get\_default\_graph instead.

In [18]: model**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])

WARNING:tensorflow:From C:\Users\rishi\Downloads\anaconda\Lib\site-packages\keras\src\optimizers\\_\_init\_\_.py:309: The name tf.tr ain.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

In [19]: *#Validating the approach*

*#To assess the model's accuracy on data it has never seen before, we will build a "validation set" by separating 10,000 samples f* x\_val **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:] y\_val **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]

In [ ]: *#We will now train our model for 20 epochs (20 iterations over all data in the x\_train and y\_train tensors), in 512-sample mini-b*

In [20]: *#Training the model*

history **=** model**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512, validation\_data**=**(x\_val, y\_val)) Epoch 1/20 WARNING:tensorflow:From C:\Users\rishi\Downloads\anaconda\Lib\site-packages\keras\src\utils\tf\_utils.py:492: The name tf.ragged.

RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\rishi\Downloads\anaconda\Lib\site-packages\keras\src\engine\base\_layer\_utils.py:384: The name t

f.executing\_eagerly\_outside\_functions is deprecated. Please use tf.compat.v1.executing\_eagerly\_outside\_functions instead.

30/30 [==============================] - 7s 185ms/step - loss: 0.5249 - accuracy: 0.7817 - val\_loss: 0.3987 - val\_accuracy: 0.86

89

Epoch 2/20

30/30 [==============================] - 1s 32ms/step - loss: 0.3305 - accuracy: 0.8922 - val\_loss: 0.3217 - val\_accuracy: 0.879

3

Epoch 3/20 30/30 [==============================] - 1s 28ms/step - loss: 0.2470 - accuracy: 0.9187 - val\_loss: 0.2844 - val\_accuracy: 0.889

5

Epoch 4/20 30/30 [==============================] - 1s 28ms/step - loss: 0.1993 - accuracy: 0.9339 - val\_loss: 0.3065 - val\_accuracy: 0.875

2

Epoch 5/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1686 - accuracy: 0.9443 - val\_loss: 0.2905 - val\_accuracy: 0.884

8

Epoch 6/20

30/30 [==============================] - 0s 14ms/step - loss: 0.1418 - accuracy: 0.9563 - val\_loss: 0.2890 - val\_accuracy: 0.882

8

Epoch 7/20 30/30 [==============================] - 0s 14ms/step - loss: 0.1222 - accuracy: 0.9620 - val\_loss: 0.3055 - val\_accuracy: 0.884

8

Epoch 8/20 30/30 [==============================] - 1s 24ms/step - loss: 0.1044 - accuracy: 0.9691 - val\_loss: 0.3066 - val\_accuracy: 0.885

0

Epoch 9/20 30/30 [==============================] - 1s 28ms/step - loss: 0.0914 - accuracy: 0.9728 - val\_loss: 0.3268 - val\_accuracy: 0.877

4

Epoch 10/20

30/30 [==============================] - 1s 31ms/step - loss: 0.0756 - accuracy: 0.9793 - val\_loss: 0.3422 - val\_accuracy: 0.876

5

Epoch 11/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0671 - accuracy: 0.9819 - val\_loss: 0.3594 - val\_accuracy: 0.881

3

Epoch 12/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0567 - accuracy: 0.9857 - val\_loss: 0.3739 - val\_accuracy: 0.877

4

Epoch 13/20 30/30 [==============================] - 1s 33ms/step - loss: 0.0460 - accuracy: 0.9900 - val\_loss: 0.4106 - val\_accuracy: 0.870

4

Epoch 14/20

30/30 [==============================] - 0s 16ms/step - loss: 0.0399 - accuracy: 0.9919 - val\_loss: 0.4417 - val\_accuracy: 0.865

4

Epoch 15/20 30/30 [==============================] - 0s 15ms/step - loss: 0.0344 - accuracy: 0.9939 - val\_loss: 0.4524 - val\_accuracy: 0.869

5

Epoch 16/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0296 - accuracy: 0.9952 - val\_loss: 0.4614 - val\_accuracy: 0.872

1

Epoch 17/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0266 - accuracy: 0.9949 - val\_loss: 0.4841 - val\_accuracy: 0.872

7

Epoch 18/20

30/30 [==============================] - 0s 14ms/step - loss: 0.0163 - accuracy: 0.9988 - val\_loss: 0.5782 - val\_accuracy: 0.858

4

Epoch 19/20 30/30 [==============================] - 1s 20ms/step - loss: 0.0187 - accuracy: 0.9972 - val\_loss: 0.5314 - val\_accuracy: 0.871

0

Epoch 20/20 30/30 [==============================] - 1s 28ms/step - loss: 0.0121 - accuracy: 0.9997 - val\_loss: 0.5502 - val\_accuracy: 0.871

0

In [ ]: *#It should be noted that the model.fit() method produces a History object. This object contains a member history, which is a dict*

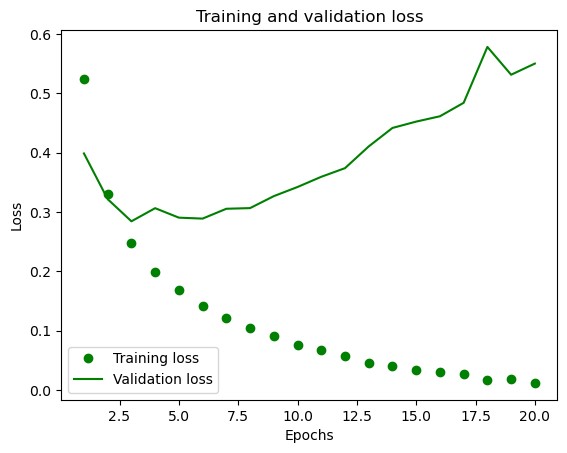
In [21]: history\_dict **=** history**.**history history\_dict**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[21]:

|  |
| --- |
| *#Plotting the training loss and validation loss* **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, "go", label**=**"Training loss") plt**.**plot(epochs, val\_loss\_values, "g", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show() |

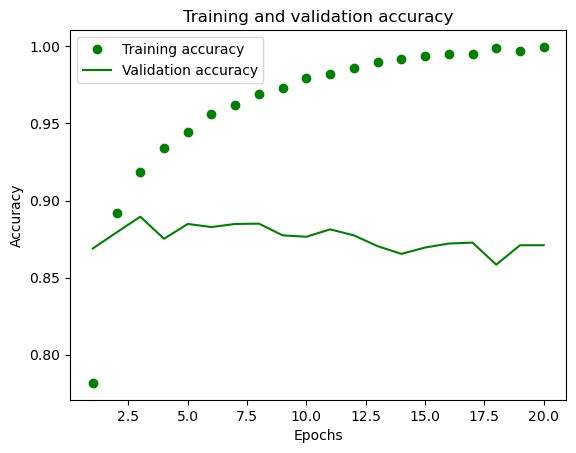
In [22]:



|  |
| --- |
| *#Plotting the training accuracy and validatition accuracy* plt**.**clf() acc **=** history\_dict["accuracy"] val\_acc **=** history\_dict["val\_accuracy"]  plt**.**plot(epochs, acc, "go", label**=**"Training accuracy") plt**.**plot(epochs, val\_acc, "g", label**=**"Validation accuracy") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

|  |
| --- |
| *#Model retraining from the scratch* 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) |

In [23]:



In [ ]:

*#The dots represent training loss and accuracy, whereas the solid lines represent validation loss and accuracy.To avoid overfitti*

In [24]:

Epoch 1/4 49/49 [==============================] - 3s 22ms/step - loss: 0.4405 - accuracy: 0.8266 Epoch 2/4 49/49 [==============================] - 1s 18ms/step - loss: 0.2616 - accuracy: 0.9060

Epoch 3/4 49/49 [==============================] - 1s 19ms/step - loss: 0.2078 - accuracy: 0.9239 Epoch 4/4

49/49 [==============================] - 1s 16ms/step - loss: 0.1806 - accuracy: 0.9343

782/782 [==============================] - 4s 4ms/step - loss: 0.2863 - accuracy: 0.8850

In [25]: results

[0.2862660884857178, 0.8849999904632568] Out[25]:

In [ ]: *#Our basic technique yields 88% accuracy with a loss of 0.2831*

In [26]: *#Using a trained model to make predictions about new data* model**.**predict(x\_test)

782/782 [==============================] - 4s 3ms/step

array([[0.21887662],

Out[26]:

[0.99978924],

1. 84688747], ...,

[0.08453077],

[0.07949591],

[0.6130646 ]], dtype=float32)

|  |
| --- |
| *#We were using 2 hidden layers. Try to use 1 or 3 hidden layers and see how it affects validation and test accuracy.*  *#Try to use layers with more hidden units or less hidden units: 32 units, 64 units... #Try to use the mse loss function instead of binary\_crossentropy.*  *#Try to use the tanh activation (an activation that was popular in the early days of neural networks) instead of relu.*  *#Try to use any technique studied in class, and these include regularization, dropout, etc., to get your model to perform better* |

In [ ]: *#Additional Experiments* In [ ]:

|  |
| --- |
| *#Constructing a neural network with single hidden layer* model\_1\_layer **=** keras**.**Sequential([ layers**.**Dense(16, activation**=**"relu"), layers**.**Dense(1, activation**=**"sigmoid")  ])  model\_1\_layer**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  x\_val1 **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val1 **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  history1\_layer **=** model\_1\_layer**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val1, y\_val1)) |

In [27]:

Epoch 1/20 30/30 [==============================] - 6s 168ms/step - loss: 0.5119 - accuracy: 0.7835 - val\_loss: 0.3964 - val\_accuracy: 0.86

26

Epoch 2/20 30/30 [==============================] - 1s 29ms/step - loss: 0.3308 - accuracy: 0.8925 - val\_loss: 0.3237 - val\_accuracy: 0.883

2

Epoch 3/20 30/30 [==============================] - 1s 26ms/step - loss: 0.2584 - accuracy: 0.9187 - val\_loss: 0.2973 - val\_accuracy: 0.885

5

Epoch 4/20

30/30 [==============================] - 1s 29ms/step - loss: 0.2191 - accuracy: 0.9283 - val\_loss: 0.2825 - val\_accuracy: 0.888

1

Epoch 5/20 30/30 [==============================] - 1s 26ms/step - loss: 0.1898 - accuracy: 0.9389 - val\_loss: 0.2755 - val\_accuracy: 0.888

6

Epoch 6/20 30/30 [==============================] - 1s 25ms/step - loss: 0.1675 - accuracy: 0.9478 - val\_loss: 0.2783 - val\_accuracy: 0.888

4

Epoch 7/20 30/30 [==============================] - 1s 26ms/step - loss: 0.1518 - accuracy: 0.9544 - val\_loss: 0.2800 - val\_accuracy: 0.886

7

Epoch 8/20

30/30 [==============================] - 1s 27ms/step - loss: 0.1373 - accuracy: 0.9593 - val\_loss: 0.2823 - val\_accuracy: 0.886

8

Epoch 9/20 30/30 [==============================] - 1s 25ms/step - loss: 0.1245 - accuracy: 0.9633 - val\_loss: 0.2879 - val\_accuracy: 0.885

3

Epoch 10/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1136 - accuracy: 0.9691 - val\_loss: 0.3113 - val\_accuracy: 0.877

2

Epoch 11/20 30/30 [==============================] - 1s 26ms/step - loss: 0.1039 - accuracy: 0.9719 - val\_loss: 0.3060 - val\_accuracy: 0.878

9

Epoch 12/20

30/30 [==============================] - 1s 23ms/step - loss: 0.0954 - accuracy: 0.9754 - val\_loss: 0.3062 - val\_accuracy: 0.883

3

Epoch 13/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0889 - accuracy: 0.9771 - val\_loss: 0.3143 - val\_accuracy: 0.882

0

Epoch 14/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0808 - accuracy: 0.9811 - val\_loss: 0.3421 - val\_accuracy: 0.871

6

Epoch 15/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0756 - accuracy: 0.9839 - val\_loss: 0.3397 - val\_accuracy: 0.880

7

Epoch 16/20

30/30 [==============================] - 1s 31ms/step - loss: 0.0700 - accuracy: 0.9851 - val\_loss: 0.3402 - val\_accuracy: 0.880

9

Epoch 17/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0646 - accuracy: 0.9870 - val\_loss: 0.3477 - val\_accuracy: 0.876

6

Epoch 18/20 30/30 [==============================] - 1s 24ms/step - loss: 0.0593 - accuracy: 0.9881 - val\_loss: 0.3614 - val\_accuracy: 0.873

9

Epoch 19/20 30/30 [==============================] - 1s 24ms/step - loss: 0.0532 - accuracy: 0.9913 - val\_loss: 0.3738 - val\_accuracy: 0.877

3

Epoch 20/20

30/30 [==============================] - 1s 26ms/step - loss: 0.0497 - accuracy: 0.9923 - val\_loss: 0.3949 - val\_accuracy: 0.868

1

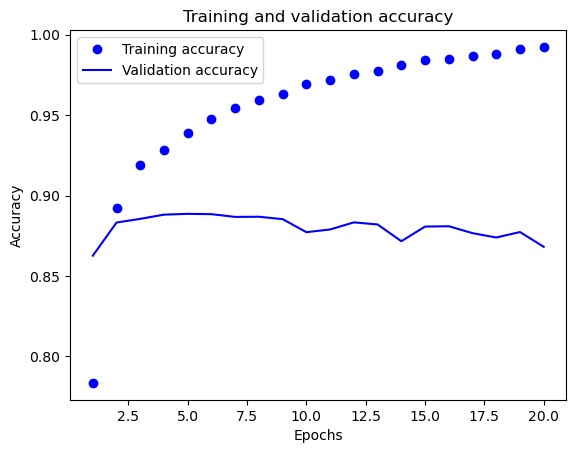
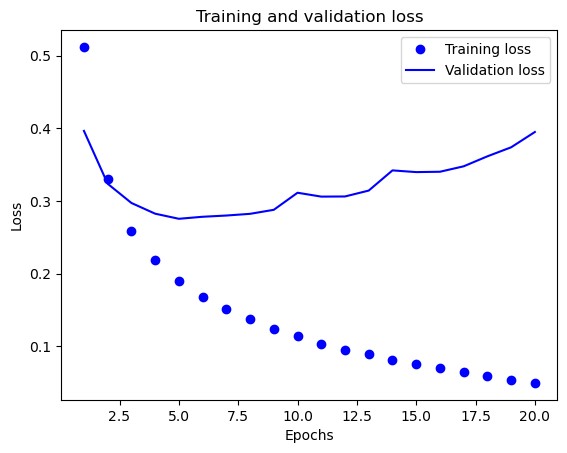
In [28]: history\_dict1 **=** history1\_layer**.**history history\_dict1**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[28]:

|  |
| --- |
| **import** matplotlib.pyplot **as** plt history\_dict1 **=** history1\_layer**.**history loss\_value1 **=** history\_dict1["loss"] val\_loss\_value1 **=** history\_dict1["val\_loss"] epochs1 **=** range(1, len(loss\_value1) **+** 1)  *#Plotting graph of Training and Validation loss*  plt**.**plot(epochs1, loss\_value1, "bo", label**=**"Training loss") plt**.**plot(epochs1, val\_loss\_value1, "b", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  *#Plotting graph of Training and Validation Accuracy* plt**.**clf()  accuracy1 **=** history\_dict1["accuracy"] val\_accuracy1 **=** history\_dict1["val\_accuracy"]  plt**.**plot(epochs1, accuracy1, "bo", label**=**"Training accuracy") plt**.**plot(epochs1, val\_accuracy1, "b", label**=**"Validation accuracy") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [29]:



In [ ]: *#As you can see, the fewer layers begin overfitting later than the reference model. Let's now employ the five epochs.*

|  |
| --- |
| *#Creating the model*  model\_1\_layer **=** keras**.**Sequential([ layers**.**Dense(16, activation**=**"relu"), layers**.**Dense(1, activation**=**"sigmoid")  ])  model\_1\_layer**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  model\_1\_layer**.**fit(x\_train, y\_train, epochs**=**5, batch\_size**=**512) result\_1\_layer **=** model\_1\_layer**.**evaluate(x\_test, y\_test) |

In [30]:

Epoch 1/5 49/49 [==============================] - 2s 18ms/step - loss: 0.4605 - accuracy: 0.8215 Epoch 2/5 49/49 [==============================] - 1s 16ms/step - loss: 0.2923 - accuracy: 0.9018

Epoch 3/5 49/49 [==============================] - 1s 18ms/step - loss: 0.2367 - accuracy: 0.9176 Epoch 4/5 49/49 [==============================] - 1s 19ms/step - loss: 0.2061 - accuracy: 0.9288

Epoch 5/5

49/49 [==============================] - 1s 16ms/step - loss: 0.1861 - accuracy: 0.9344

782/782 [==============================] - 3s 3ms/step - loss: 0.2809 - accuracy: 0.8878

In [31]: print(result\_1\_layer)

[0.28089815378189087, 0.8878399729728699]

In [ ]: *##The loss on the test set is 0.280%, and the accuracy is 88.78%.*

In [32]: model\_1\_layer**.**predict(x\_test)

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

array([[0.26410767],

Out[32]:

[0.9994294 ],

[0.8488912 ], ...,

[0.13977474],

1. 12857798],

[0.6421129 ]], dtype=float32)

|  |
| --- |
| *#Building a neural network with 3 hidden layers* model\_3\_layers **=** 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**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"]) x\_val3 **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val3 **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  history\_3\_layers **=** model\_3\_layers**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val3, y\_val3)) |

In [33]:

Epoch 1/20

30/30 [==============================] - 6s 158ms/step - loss: 0.5326 - accuracy: 0.7771 - val\_loss: 0.3914 - val\_accuracy: 0.87

09

Epoch 2/20 30/30 [==============================] - 1s 21ms/step - loss: 0.3157 - accuracy: 0.8941 - val\_loss: 0.3053 - val\_accuracy: 0.885

8

Epoch 3/20 30/30 [==============================] - 1s 17ms/step - loss: 0.2313 - accuracy: 0.9201 - val\_loss: 0.3087 - val\_accuracy: 0.871

6

Epoch 4/20 30/30 [==============================] - 1s 31ms/step - loss: 0.1821 - accuracy: 0.9389 - val\_loss: 0.2818 - val\_accuracy: 0.884

6

Epoch 5/20

30/30 [==============================] - 1s 31ms/step - loss: 0.1512 - accuracy: 0.9479 - val\_loss: 0.2845 - val\_accuracy: 0.885

5

Epoch 6/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1261 - accuracy: 0.9567 - val\_loss: 0.3492 - val\_accuracy: 0.873

1

Epoch 7/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1087 - accuracy: 0.9638 - val\_loss: 0.3161 - val\_accuracy: 0.884

0

Epoch 8/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0891 - accuracy: 0.9707 - val\_loss: 0.3536 - val\_accuracy: 0.875

1

Epoch 9/20

30/30 [==============================] - 1s 25ms/step - loss: 0.0748 - accuracy: 0.9769 - val\_loss: 0.3668 - val\_accuracy: 0.876

1

Epoch 10/20 30/30 [==============================] - 1s 30ms/step - loss: 0.0654 - accuracy: 0.9801 - val\_loss: 0.3853 - val\_accuracy: 0.876

4

Epoch 11/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0548 - accuracy: 0.9842 - val\_loss: 0.4147 - val\_accuracy: 0.876

5

Epoch 12/20 30/30 [==============================] - 1s 28ms/step - loss: 0.0444 - accuracy: 0.9879 - val\_loss: 0.4410 - val\_accuracy: 0.877

6

Epoch 13/20

30/30 [==============================] - 1s 26ms/step - loss: 0.0393 - accuracy: 0.9887 - val\_loss: 0.4680 - val\_accuracy: 0.875

6

Epoch 14/20 30/30 [==============================] - 1s 30ms/step - loss: 0.0359 - accuracy: 0.9901 - val\_loss: 0.5005 - val\_accuracy: 0.874

1

Epoch 15/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0170 - accuracy: 0.9981 - val\_loss: 0.5405 - val\_accuracy: 0.870

5

Epoch 16/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0231 - accuracy: 0.9952 - val\_loss: 0.5516 - val\_accuracy: 0.873

3

Epoch 17/20

30/30 [==============================] - 1s 33ms/step - loss: 0.0248 - accuracy: 0.9941 - val\_loss: 0.5796 - val\_accuracy: 0.871

8

Epoch 18/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0080 - accuracy: 0.9997 - val\_loss: 0.6265 - val\_accuracy: 0.866

0

Epoch 19/20 30/30 [==============================] - 1s 40ms/step - loss: 0.0196 - accuracy: 0.9941 - val\_loss: 0.6321 - val\_accuracy: 0.872

1

Epoch 20/20 30/30 [==============================] - 1s 28ms/step - loss: 0.0066 - accuracy: 0.9993 - val\_loss: 0.8076 - val\_accuracy: 0.854

6

In [34]: history\_dict\_3 **=** history\_3\_layers**.**history

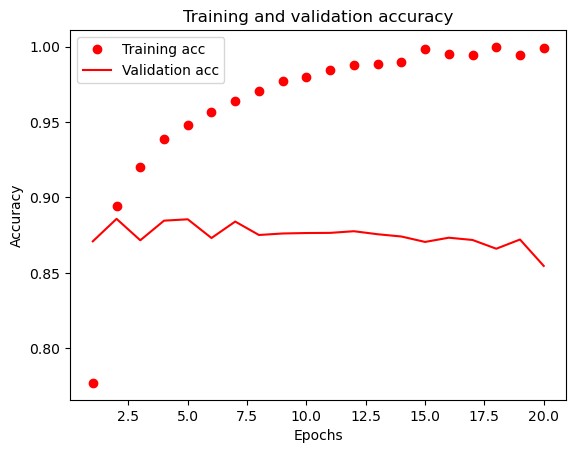
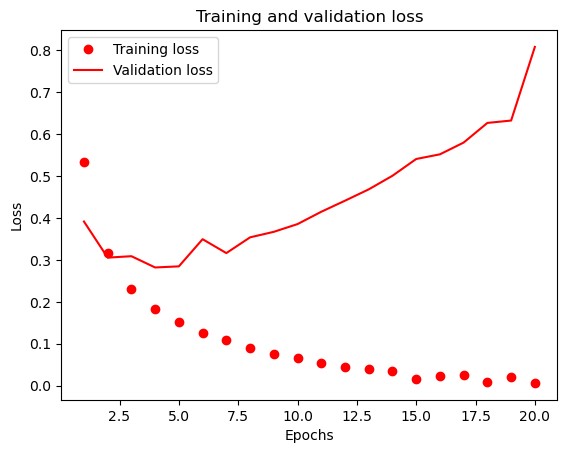
history\_dict\_3**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[34]:

|  |
| --- |
| loss\_val3 **=** history\_dict\_3["loss"] val\_loss\_val3 **=** history\_dict\_3["val\_loss"] epochs3 **=** range(1, len(loss\_val3) **+** 1)  plt**.**plot(epochs3, loss\_val3, "ro", label**=**"Training loss") plt**.**plot(epochs3, val\_loss\_val3, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf() *#clear figure*  accuracy3 **=** history\_dict\_3["accuracy"] val\_accuracy3 **=** history\_dict\_3["val\_accuracy"] plt**.**plot(epochs3, accuracy3, "ro", label**=**"Training acc") plt**.**plot(epochs3, val\_accuracy3, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [35]:



In [ ]: *#As we can see, the more layers starts overfitting so Let's use three epochs*

In [36]: model\_3\_layers **=** 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**.**compile(optimizer**=**'rmsprop', loss**=**'binary\_crossentropy', metrics**=**['accuracy'])

model\_3\_layers**.**fit(x\_train, y\_train, epochs**=**3, batch\_size**=**512) results\_3\_layers **=** model\_3\_layers**.**evaluate(x\_test, y\_test)

Epoch 1/3 49/49 [==============================] - 3s 23ms/step - loss: 0.5037 - accuracy: 0.8034 Epoch 2/3 49/49 [==============================] - 1s 19ms/step - loss: 0.2831 - accuracy: 0.8984

Epoch 3/3

49/49 [==============================] - 1s 16ms/step - loss: 0.2192 - accuracy: 0.9192

782/782 [==============================] - 3s 4ms/step - loss: 0.2978 - accuracy: 0.8805

In [37]: print(results\_3\_layers)

[0.29780739545822144, 0.8804799914360046]

In [38]: model\_3\_layers**.**predict(x\_test)

|  |  |
| --- | --- |
|  | 782/782 [==============================] - 3s 3ms/step |
| Out[38]: In [ ]: | array([[0.17930758],  [0.999612 ],  [0.6804914 ], ...,  [0.06404474],  [0.06450368],  [0.34289587]], dtype=float32)  *#The accuracy of the model does not rise significantly as the number of layers increases. However, compared to the other two, the* |

In [ ]: *#Creating the Neural Network with 32 Hidden units & 3 layers.*

|  |
| --- |
| model\_32\_units **=** 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**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  *#model validation* x\_val\_32 **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val\_32 **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  history\_32\_units **=** model\_32\_units**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val\_32, y\_val\_32)) |

In [39]:

Epoch 1/20 30/30 [==============================] - 7s 181ms/step - loss: 0.5143 - accuracy: 0.7608 - val\_loss: 0.3534 - val\_accuracy: 0.87

58

Epoch 2/20 30/30 [==============================] - 1s 44ms/step - loss: 0.2923 - accuracy: 0.8918 - val\_loss: 0.3231 - val\_accuracy: 0.865

4

Epoch 3/20 30/30 [==============================] - 1s 40ms/step - loss: 0.2166 - accuracy: 0.9199 - val\_loss: 0.2774 - val\_accuracy: 0.888

6

Epoch 4/20

30/30 [==============================] - 1s 42ms/step - loss: 0.1620 - accuracy: 0.9437 - val\_loss: 0.3188 - val\_accuracy: 0.874

6

Epoch 5/20 30/30 [==============================] - 1s 40ms/step - loss: 0.1435 - accuracy: 0.9483 - val\_loss: 0.3240 - val\_accuracy: 0.877

8

Epoch 6/20 30/30 [==============================] - 1s 45ms/step - loss: 0.1090 - accuracy: 0.9635 - val\_loss: 0.4420 - val\_accuracy: 0.851

8

Epoch 7/20 30/30 [==============================] - 1s 36ms/step - loss: 0.0826 - accuracy: 0.9735 - val\_loss: 0.3573 - val\_accuracy: 0.878

3

Epoch 8/20

30/30 [==============================] - 1s 39ms/step - loss: 0.0694 - accuracy: 0.9787 - val\_loss: 0.3870 - val\_accuracy: 0.875

7

Epoch 9/20 30/30 [==============================] - 1s 33ms/step - loss: 0.0612 - accuracy: 0.9817 - val\_loss: 0.3987 - val\_accuracy: 0.877

3

Epoch 10/20 30/30 [==============================] - 1s 35ms/step - loss: 0.0472 - accuracy: 0.9865 - val\_loss: 0.4249 - val\_accuracy: 0.877

8

Epoch 11/20 30/30 [==============================] - 1s 32ms/step - loss: 0.0425 - accuracy: 0.9884 - val\_loss: 0.4664 - val\_accuracy: 0.871

5

Epoch 12/20

30/30 [==============================] - 1s 34ms/step - loss: 0.0109 - accuracy: 0.9991 - val\_loss: 0.4850 - val\_accuracy: 0.876

4

Epoch 13/20 30/30 [==============================] - 1s 36ms/step - loss: 0.0419 - accuracy: 0.9877 - val\_loss: 0.5068 - val\_accuracy: 0.874

7

Epoch 14/20 30/30 [==============================] - 1s 33ms/step - loss: 0.0052 - accuracy: 0.9998 - val\_loss: 0.5416 - val\_accuracy: 0.874

3

Epoch 15/20 30/30 [==============================] - 1s 36ms/step - loss: 0.0034 - accuracy: 0.9999 - val\_loss: 0.9564 - val\_accuracy: 0.833

9

Epoch 16/20

30/30 [==============================] - 1s 38ms/step - loss: 0.0376 - accuracy: 0.9895 - val\_loss: 0.5990 - val\_accuracy: 0.873

6

Epoch 17/20 30/30 [==============================] - 1s 36ms/step - loss: 0.0020 - accuracy: 0.9999 - val\_loss: 0.6289 - val\_accuracy: 0.874

0

Epoch 18/20 30/30 [==============================] - 1s 42ms/step - loss: 0.0306 - accuracy: 0.9916 - val\_loss: 0.6454 - val\_accuracy: 0.873

2

Epoch 19/20 30/30 [==============================] - 1s 37ms/step - loss: 0.0014 - accuracy: 0.9999 - val\_loss: 0.6629 - val\_accuracy: 0.873

2

Epoch 20/20

30/30 [==============================] - 1s 37ms/step - loss: 9.6850e-04 - accuracy: 0.9999 - val\_loss: 0.6963 - val\_accuracy:

0.8737

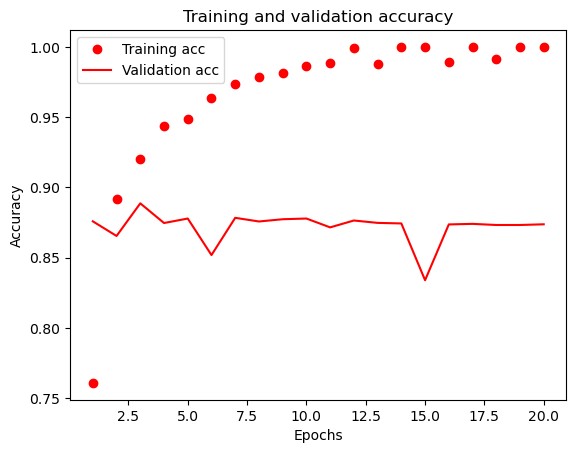
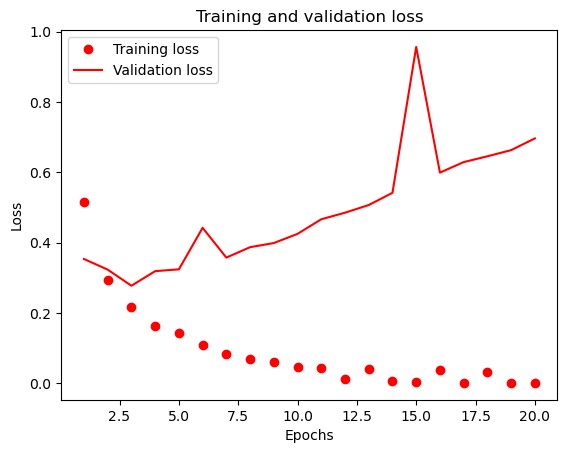
In [41]: history\_dict\_32 **=** history\_32\_units**.**history history\_dict\_32**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[41]:

|  |
| --- |
| loss\_value\_32 **=** history\_dict\_32["loss"] val\_loss\_value\_32 **=** history\_dict\_32["val\_loss"] epochs\_32 **=** range(1, len(loss\_value\_32) **+** 1)  plt**.**plot(epochs\_32, loss\_value\_32, "ro", label**=**"Training loss") plt**.**plot(epochs\_32, val\_loss\_value\_32, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf() *#clear figure*  accuracy\_32 **=** history\_dict\_32["accuracy"] val\_accuracy\_32 **=** history\_dict\_32["val\_accuracy"]  plt**.**plot(epochs\_32, accuracy\_32, "ro", label**=**"Training acc") plt**.**plot(epochs\_32, val\_accuracy\_32, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [42]:



In [43]: history\_32\_units **=** model\_32\_units**.**fit(x\_train, y\_train, epochs**=**3, batch\_size**=**512) results\_32\_units **=** model\_32\_units**.**evaluate(x\_test, y\_test) results\_32\_units

Epoch 1/3 49/49 [==============================] - 2s 29ms/step - loss: 0.2308 - accuracy: 0.9444 Epoch 2/3 49/49 [==============================] - 1s 27ms/step - loss: 0.1108 - accuracy: 0.9652

Epoch 3/3

49/49 [==============================] - 1s 25ms/step - loss: 0.0699 - accuracy: 0.9783

782/782 [==============================] - 3s 3ms/step - loss: 0.4468 - accuracy: 0.8680

[0.4467557370662689, 0.8679999709129333] Out[43]:

In [44]: model\_32\_units**.**predict(x\_test)

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

array([[0.00850865],

Out[44]:

[0.9999906 ],

[0.2733062 ], ...,

[0.01452067],

[0.00449701],

[0.9948406 ]], dtype=float32)

In [ ]: validation set accuracy **is** 86.79**%**

In [ ]: *#Having the model with 64 units & 2 layers.*

In [45]: model\_64\_units **=** keras**.**Sequential([ layers**.**Dense(64, activation**=**"relu"), layers**.**Dense(64, activation**=**"relu"), layers**.**Dense(1, activation**=**"sigmoid")

])

model\_64\_units**.**compile(optimizer**=**"rmsprop",

|  |
| --- |
| loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  *# validation*  x\_val\_64 **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val\_64 **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  history\_64 **=** model\_64\_units**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val\_64, y\_val\_64)) |

Epoch 1/20 30/30 [==============================] - 7s 196ms/step - loss: 0.5188 - accuracy: 0.7451 - val\_loss: 0.3501 - val\_accuracy: 0.86

85

Epoch 2/20

30/30 [==============================] - 2s 64ms/step - loss: 0.3034 - accuracy: 0.8845 - val\_loss: 0.2865 - val\_accuracy: 0.888

6

Epoch 3/20 30/30 [==============================] - 2s 57ms/step - loss: 0.2210 - accuracy: 0.9159 - val\_loss: 0.2748 - val\_accuracy: 0.889

3

Epoch 4/20 30/30 [==============================] - 2s 54ms/step - loss: 0.1916 - accuracy: 0.9280 - val\_loss: 0.2814 - val\_accuracy: 0.885

2

Epoch 5/20 30/30 [==============================] - 2s 53ms/step - loss: 0.1448 - accuracy: 0.9498 - val\_loss: 0.3270 - val\_accuracy: 0.873

7

Epoch 6/20

30/30 [==============================] - 2s 61ms/step - loss: 0.1406 - accuracy: 0.9481 - val\_loss: 0.3021 - val\_accuracy: 0.884

0

Epoch 7/20 30/30 [==============================] - 2s 57ms/step - loss: 0.1064 - accuracy: 0.9617 - val\_loss: 0.3214 - val\_accuracy: 0.882

9

Epoch 8/20 30/30 [==============================] - 2s 58ms/step - loss: 0.0948 - accuracy: 0.9672 - val\_loss: 0.3578 - val\_accuracy: 0.880

1

Epoch 9/20 30/30 [==============================] - 2s 56ms/step - loss: 0.0749 - accuracy: 0.9762 - val\_loss: 0.3684 - val\_accuracy: 0.877

5

Epoch 10/20

30/30 [==============================] - 2s 57ms/step - loss: 0.0572 - accuracy: 0.9826 - val\_loss: 0.4075 - val\_accuracy: 0.873

2

Epoch 11/20 30/30 [==============================] - 2s 53ms/step - loss: 0.0531 - accuracy: 0.9849 - val\_loss: 0.4199 - val\_accuracy: 0.876

1

Epoch 12/20 30/30 [==============================] - 2s 56ms/step - loss: 0.0437 - accuracy: 0.9878 - val\_loss: 0.4342 - val\_accuracy: 0.877

7

Epoch 13/20 30/30 [==============================] - 2s 55ms/step - loss: 0.0312 - accuracy: 0.9913 - val\_loss: 0.4808 - val\_accuracy: 0.874

6

Epoch 14/20

30/30 [==============================] - 2s 55ms/step - loss: 0.0389 - accuracy: 0.9886 - val\_loss: 0.4751 - val\_accuracy: 0.877

3

Epoch 15/20 30/30 [==============================] - 2s 58ms/step - loss: 0.0088 - accuracy: 0.9997 - val\_loss: 0.5220 - val\_accuracy: 0.875

7

Epoch 16/20 30/30 [==============================] - 2s 54ms/step - loss: 0.0368 - accuracy: 0.9883 - val\_loss: 0.5337 - val\_accuracy: 0.874

6

Epoch 17/20 30/30 [==============================] - 2s 56ms/step - loss: 0.0049 - accuracy: 0.9999 - val\_loss: 0.5839 - val\_accuracy: 0.874

3

Epoch 18/20

30/30 [==============================] - 2s 60ms/step - loss: 0.0358 - accuracy: 0.9890 - val\_loss: 0.5794 - val\_accuracy: 0.874

9

Epoch 19/20 30/30 [==============================] - 2s 52ms/step - loss: 0.0032 - accuracy: 0.9999 - val\_loss: 0.6093 - val\_accuracy: 0.874

6

Epoch 20/20 30/30 [==============================] - 2s 61ms/step - loss: 0.0295 - accuracy: 0.9923 - val\_loss: 0.6615 - val\_accuracy: 0.862

8

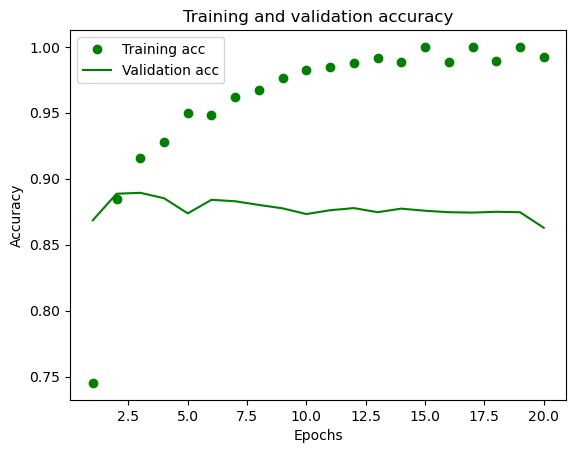
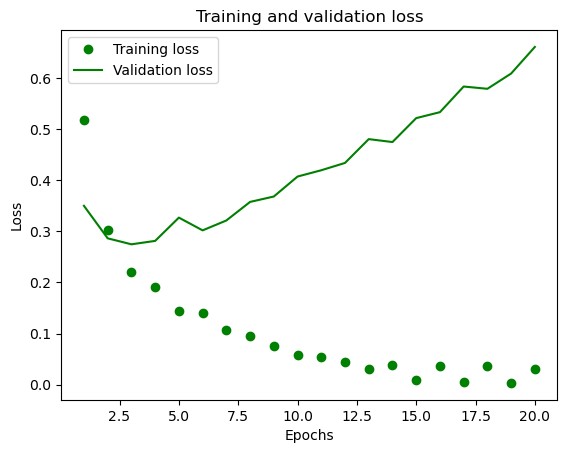
In [46]: history\_dict\_64 **=** history\_64**.**history history\_dict\_64**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[46]:

In [47]: loss\_value64 **=** history\_dict\_64["loss"] val\_loss\_value64 **=** history\_dict\_64["val\_loss"] epochs\_64 **=** range(1, len(loss\_value64) **+** 1) plt**.**plot(epochs\_64, loss\_value64, "go", label**=**"Training loss") plt**.**plot(epochs\_64, val\_loss\_value64, "g", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()

|  |
| --- |
| plt**.**clf()  accuracy\_64 **=** history\_dict\_64["accuracy"] val\_accuracy\_64 **=** history\_dict\_64["val\_accuracy"]  plt**.**plot(epochs\_64, accuracy\_64, "go", label**=**"Training acc") plt**.**plot(epochs\_64, val\_accuracy\_64, "g", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |



In [48]: history\_64 **=** model\_64\_units**.**fit(x\_train, y\_train, epochs**=**3, batch\_size**=**512) results\_64\_units **=** model\_64\_units**.**evaluate(x\_test, y\_test) results\_64\_units

Epoch 1/3 49/49 [==============================] - 2s 33ms/step - loss: 0.1829 - accuracy: 0.9486

Epoch 2/3

49/49 [==============================] - 2s 39ms/step - loss: 0.1048 - accuracy: 0.9676

Epoch 3/3

49/49 [==============================] - 2s 41ms/step - loss: 0.0664 - accuracy: 0.9805

782/782 [==============================] - 3s 4ms/step - loss: 0.4069 - accuracy: 0.8696

[0.4069437086582184, 0.8695600032806396] Out[48]:

In [49]: model\_64\_units**.**predict(x\_test)

782/782 [==============================] - 3s 4ms/step

array([[0.00927142],

Out[49]:

[0.99999964],

1. 38905373], ...,

[0.01108076],

[0.00626881],

[0.8885038 ]], dtype=float32)

In [ ]: validation set accuracy **=** 86.95**%**

|  |
| --- |
| model\_128units **=** keras**.**Sequential([ layers**.**Dense(128, activation**=**"relu"), layers**.**Dense(128, activation**=**"relu"), layers**.**Dense(128, activation**=**"relu"), layers**.**Dense(1, activation**=**"sigmoid")  ])  model\_128units**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  *# validation*  x\_val\_128 **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val\_128 **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  history\_128 **=** model\_128units**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val\_128, y\_val\_128)) |

In [ ]: *#Training the model with 128 units & 3 layers* In [50]:

Epoch 1/20 30/30 [==============================] - 13s 215ms/step - loss: 0.5440 - accuracy: 0.7387 - val\_loss: 0.3335 - val\_accuracy: 0.8

716

Epoch 2/20

30/30 [==============================] - 3s 113ms/step - loss: 0.3026 - accuracy: 0.8785 - val\_loss: 0.3291 - val\_accuracy: 0.86

55

Epoch 3/20 30/30 [==============================] - 3s 92ms/step - loss: 0.2294 - accuracy: 0.9100 - val\_loss: 0.2727 - val\_accuracy: 0.888

0

Epoch 4/20 30/30 [==============================] - 2s 83ms/step - loss: 0.1879 - accuracy: 0.9299 - val\_loss: 0.2869 - val\_accuracy: 0.886

3

Epoch 5/20 30/30 [==============================] - 3s 86ms/step - loss: 0.1232 - accuracy: 0.9565 - val\_loss: 0.3393 - val\_accuracy: 0.876

8

Epoch 6/20

30/30 [==============================] - 3s 94ms/step - loss: 0.1032 - accuracy: 0.9660 - val\_loss: 0.4578 - val\_accuracy: 0.849

7

Epoch 7/20 30/30 [==============================] - 2s 71ms/step - loss: 0.0969 - accuracy: 0.9691 - val\_loss: 0.3464 - val\_accuracy: 0.882

2

Epoch 8/20 30/30 [==============================] - 2s 62ms/step - loss: 0.0632 - accuracy: 0.9809 - val\_loss: 0.3487 - val\_accuracy: 0.882

5

Epoch 9/20 30/30 [==============================] - 3s 86ms/step - loss: 0.0134 - accuracy: 0.9977 - val\_loss: 0.4890 - val\_accuracy: 0.880

3

Epoch 10/20

30/30 [==============================] - 2s 83ms/step - loss: 0.0911 - accuracy: 0.9796 - val\_loss: 0.4243 - val\_accuracy: 0.878

4

Epoch 11/20 30/30 [==============================] - 3s 83ms/step - loss: 0.0049 - accuracy: 0.9997 - val\_loss: 0.5246 - val\_accuracy: 0.879

4

Epoch 12/20 30/30 [==============================] - 2s 81ms/step - loss: 0.0016 - accuracy: 0.9999 - val\_loss: 0.6665 - val\_accuracy: 0.875

0

Epoch 13/20 30/30 [==============================] - 3s 90ms/step - loss: 0.0811 - accuracy: 0.9845 - val\_loss: 0.5120 - val\_accuracy: 0.879

3

Epoch 14/20

30/30 [==============================] - 3s 90ms/step - loss: 0.0014 - accuracy: 1.0000 - val\_loss: 0.6026 - val\_accuracy: 0.879

1

Epoch 15/20

30/30 [==============================] - 3s 83ms/step - loss: 5.2742e-04 - accuracy: 1.0000 - val\_loss: 0.7309 - val\_accuracy:

0.8760 Epoch 16/20 30/30 [==============================] - 2s 82ms/step - loss: 0.0978 - accuracy: 0.9837 - val\_loss: 0.5553 - val\_accuracy: 0.879

5

Epoch 17/20

30/30 [==============================] - 3s 88ms/step - loss: 8.8555e-04 - accuracy: 1.0000 - val\_loss: 0.6250 - val\_accuracy:

0.8796 Epoch 18/20

30/30 [==============================] - 3s 89ms/step - loss: 3.7647e-04 - accuracy: 1.0000 - val\_loss: 0.7046 - val\_accuracy:

0.8791 Epoch 19/20

30/30 [==============================] - 3s 90ms/step - loss: 1.8995e-04 - accuracy: 1.0000 - val\_loss: 0.7666 - val\_accuracy:

0.8785 Epoch 20/20

30/30 [==============================] - 3s 84ms/step - loss: 1.1486e-04 - accuracy: 1.0000 - val\_loss: 0.8068 - val\_accuracy:

0.8787

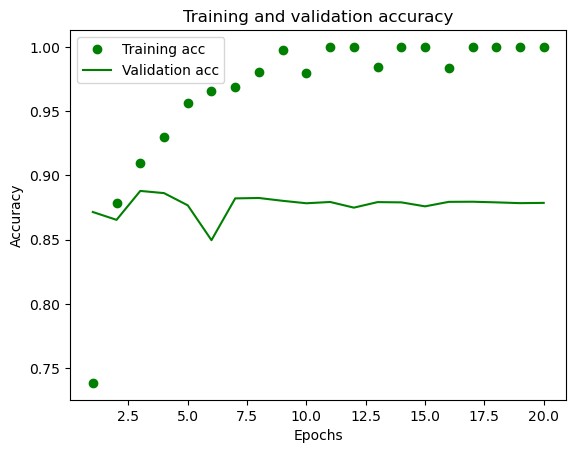
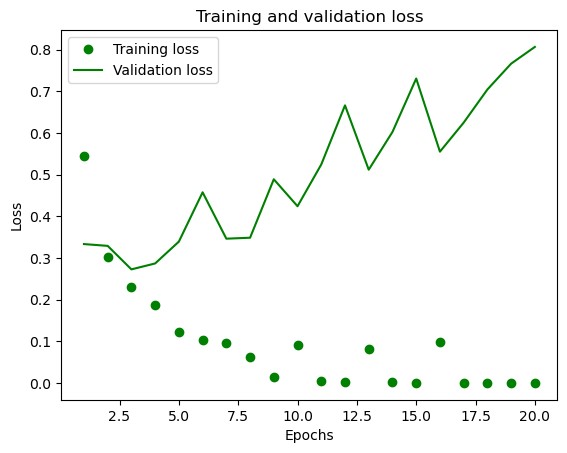
In [51]: history\_dict\_128 **=** history\_128**.**history history\_dict\_128**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[51]:

|  |
| --- |
| loss\_value128 **=** history\_dict\_128["loss"] val\_loss\_value128 **=** history\_dict\_128["val\_loss"] epochs\_128 **=** range(1, len(loss\_value128) **+** 1)  plt**.**plot(epochs\_128, loss\_value128, "go", label**=**"Training loss") plt**.**plot(epochs\_128, val\_loss\_value128, "g", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf()  accuracy\_128 **=** history\_dict\_128["accuracy"] val\_accuracy\_128 **=** history\_dict\_128["val\_accuracy"]  plt**.**plot(epochs\_128, accuracy\_128, "go", label**=**"Training acc") plt**.**plot(epochs\_128, val\_accuracy\_128, "g", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [52]:



In [53]: history\_128 **=** model\_128units**.**fit(x\_train, y\_train, epochs**=**2, batch\_size**=**512) results\_128\_units **=** model\_128units**.**evaluate(x\_test, y\_test) results\_128\_units

Epoch 1/2 49/49 [==============================] - 3s 59ms/step - loss: 0.2112 - accuracy: 0.9436

Epoch 2/2

49/49 [==============================] - 3s 54ms/step - loss: 0.0869 - accuracy: 0.9716

782/782 [==============================] - 4s 5ms/step - loss: 0.3756 - accuracy: 0.8738

[0.3756153881549835, 0.8737599849700928] Out[53]:

In [54]: model\_128units**.**predict(x\_test)

782/782 [==============================] - 4s 5ms/step

array([[0.01668514],

Out[54]:

[1. ],

[0.8119043 ], ...,

[0.0570657 ],

[0.01450755],

[0.91401994]], dtype=float32)

In [ ]: *#MSE loss function*

|  |
| --- |
| *#with 16 units*  MSE\_model **=** 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**.**compile(optimizer**=**"rmsprop", loss**=**"mse",  metrics**=**["accuracy"])  *# validation of model* x\_val\_MSE **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_val\_MSE **=** y\_train[:10000] partial\_y\_train **=** y\_train[10000:]  *# Model Fit*  history\_MSE **=** MSE\_model**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val\_MSE, y\_val\_MSE)) |

In [55]:

Epoch 1/20 30/30 [==============================] - 7s 151ms/step - loss: 0.1959 - accuracy: 0.7194 - val\_loss: 0.1514 - val\_accuracy: 0.80

99

Epoch 2/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1106 - accuracy: 0.8822 - val\_loss: 0.1216 - val\_accuracy: 0.842

1

Epoch 3/20 30/30 [==============================] - 1s 33ms/step - loss: 0.0763 - accuracy: 0.9139 - val\_loss: 0.0887 - val\_accuracy: 0.885

0

Epoch 4/20

30/30 [==============================] - 1s 33ms/step - loss: 0.0605 - accuracy: 0.9316 - val\_loss: 0.0966 - val\_accuracy: 0.868

1

Epoch 5/20 30/30 [==============================] - 1s 30ms/step - loss: 0.0494 - accuracy: 0.9447 - val\_loss: 0.1130 - val\_accuracy: 0.849

4

Epoch 6/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0428 - accuracy: 0.9525 - val\_loss: 0.0866 - val\_accuracy: 0.882

1

Epoch 7/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0367 - accuracy: 0.9593 - val\_loss: 0.0879 - val\_accuracy: 0.881

3

Epoch 8/20

30/30 [==============================] - 1s 23ms/step - loss: 0.0289 - accuracy: 0.9696 - val\_loss: 0.0919 - val\_accuracy: 0.878

5

Epoch 9/20 30/30 [==============================] - 1s 29ms/step - loss: 0.0260 - accuracy: 0.9739 - val\_loss: 0.0934 - val\_accuracy: 0.878

5

Epoch 10/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0238 - accuracy: 0.9750 - val\_loss: 0.0940 - val\_accuracy: 0.877

0

Epoch 11/20 30/30 [==============================] - 0s 15ms/step - loss: 0.0192 - accuracy: 0.9814 - val\_loss: 0.1019 - val\_accuracy: 0.871

2

Epoch 12/20

30/30 [==============================] - 1s 17ms/step - loss: 0.0209 - accuracy: 0.9775 - val\_loss: 0.0963 - val\_accuracy: 0.876

6

Epoch 13/20 30/30 [==============================] - 1s 22ms/step - loss: 0.0166 - accuracy: 0.9833 - val\_loss: 0.0978 - val\_accuracy: 0.874

5

Epoch 14/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0132 - accuracy: 0.9871 - val\_loss: 0.0989 - val\_accuracy: 0.875

2

Epoch 15/20 30/30 [==============================] - 0s 13ms/step - loss: 0.0149 - accuracy: 0.9836 - val\_loss: 0.0998 - val\_accuracy: 0.875

0

Epoch 16/20

30/30 [==============================] - 1s 17ms/step - loss: 0.0084 - accuracy: 0.9931 - val\_loss: 0.1310 - val\_accuracy: 0.839

0

Epoch 17/20 30/30 [==============================] - 0s 16ms/step - loss: 0.0126 - accuracy: 0.9871 - val\_loss: 0.1020 - val\_accuracy: 0.873

4

Epoch 18/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0126 - accuracy: 0.9864 - val\_loss: 0.1020 - val\_accuracy: 0.873

7

Epoch 19/20 30/30 [==============================] - 0s 14ms/step - loss: 0.0069 - accuracy: 0.9940 - val\_loss: 0.1103 - val\_accuracy: 0.866

5

Epoch 20/20

30/30 [==============================] - 0s 16ms/step - loss: 0.0118 - accuracy: 0.9863 - val\_loss: 0.1035 - val\_accuracy: 0.874

2

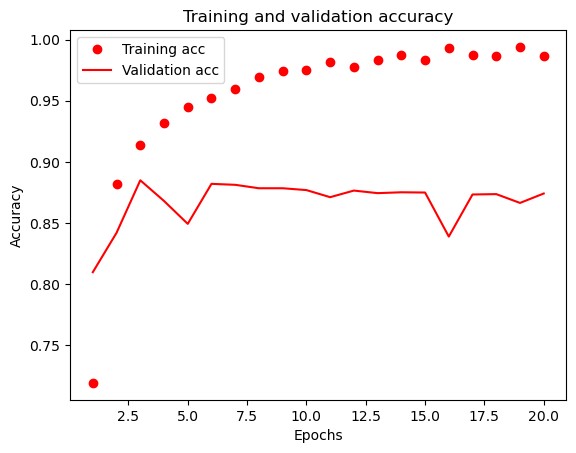
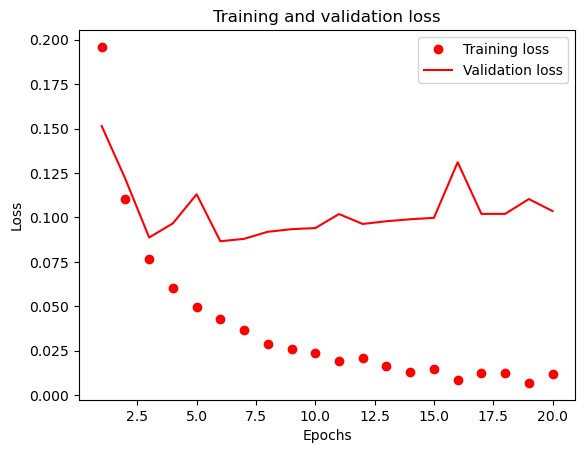
In [56]: historydict\_MSE **=** history\_MSE**.**history historydict\_MSE**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[56]:

|  |
| --- |
| **import** matplotlib.pyplot **as** plt loss\_value\_MSE **=** historydict\_MSE["loss"] val\_loss\_value\_MSE **=** historydict\_MSE["val\_loss"] epochs\_MSE **=** range(1, len(loss\_value\_MSE) **+** 1)  plt**.**plot(epochs\_MSE, loss\_value\_MSE, "ro", label**=**"Training loss") plt**.**plot(epochs\_MSE, val\_loss\_value\_MSE, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf() acc\_MSE **=** historydict\_MSE["accuracy"] val\_acc\_MSE **=** historydict\_MSE["val\_accuracy"]  plt**.**plot(epochs\_MSE, acc\_MSE, "ro", label**=**"Training acc") plt**.**plot(epochs\_MSE, val\_acc\_MSE, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [57]:



In [58]: MSE\_model**.**fit(x\_train, y\_train, epochs**=**8, batch\_size**=**512) results\_MSE **=** MSE\_model**.**evaluate(x\_test, y\_test) results\_MSE

Epoch 1/8 49/49 [==============================] - 1s 21ms/step - loss: 0.0484 - accuracy: 0.9426

Epoch 2/8 49/49 [==============================] - 0s 9ms/step - loss: 0.0383 - accuracy: 0.9558

Epoch 3/8 49/49 [==============================] - 0s 9ms/step - loss: 0.0324 - accuracy: 0.9637

Epoch 4/8 49/49 [==============================] - 0s 10ms/step - loss: 0.0281 - accuracy: 0.9689

Epoch 5/8 49/49 [==============================] - 1s 15ms/step - loss: 0.0244 - accuracy: 0.9739

Epoch 6/8 49/49 [==============================] - 1s 11ms/step - loss: 0.0246 - accuracy: 0.9730

Epoch 7/8 49/49 [==============================] - 0s 9ms/step - loss: 0.0199 - accuracy: 0.9793

Epoch 8/8

49/49 [==============================] - 1s 16ms/step - loss: 0.0193 - accuracy: 0.9800

782/782 [==============================] - 3s 3ms/step - loss: 0.1135 - accuracy: 0.8656

[0.11353883892297745, 0.8655999898910522] Out[58]:

In [59]: MSE\_model**.**predict(x\_test)

782/782 [==============================] - 3s 3ms/step array([[8.8348513e-04],

Out[59]:

[9.9853200e-01],

[7.5700812e-02], ...,

[6.4232559e-03],

[4.8348002e-04],

[8.9421880e-01]], dtype=float32)

|  |
| --- |
| tanh **=** keras**.**Sequential([  layers**.**Dense(16, activation**=**"tanh"), layers**.**Dense(1, activation**=**"sigmoid")  ])  tanh**.**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**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val\_tanh, y\_val\_tanh)) |

In [ ]: *#Tanh activation* In [60]:

Epoch 1/20 30/30 [==============================] - 5s 135ms/step - loss: 0.1758 - accuracy: 0.7883 - val\_loss: 0.1362 - val\_accuracy: 0.85

85

Epoch 2/20

30/30 [==============================] - 1s 29ms/step - loss: 0.1140 - accuracy: 0.8835 - val\_loss: 0.1140 - val\_accuracy: 0.863

7

Epoch 3/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0905 - accuracy: 0.9039 - val\_loss: 0.1014 - val\_accuracy: 0.874

8

Epoch 4/20 30/30 [==============================] - 1s 40ms/step - loss: 0.0760 - accuracy: 0.9195 - val\_loss: 0.0910 - val\_accuracy: 0.887

4

Epoch 5/20 30/30 [==============================] - 1s 33ms/step - loss: 0.0670 - accuracy: 0.9267 - val\_loss: 0.0871 - val\_accuracy: 0.888

7

Epoch 6/20

30/30 [==============================] - 1s 28ms/step - loss: 0.0594 - accuracy: 0.9365 - val\_loss: 0.0865 - val\_accuracy: 0.888

0

Epoch 7/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0544 - accuracy: 0.9420 - val\_loss: 0.0851 - val\_accuracy: 0.884

2

Epoch 8/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0491 - accuracy: 0.9489 - val\_loss: 0.0862 - val\_accuracy: 0.885

1

Epoch 9/20 30/30 [==============================] - 1s 28ms/step - loss: 0.0445 - accuracy: 0.9551 - val\_loss: 0.0896 - val\_accuracy: 0.877

0

Epoch 10/20

30/30 [==============================] - 1s 27ms/step - loss: 0.0415 - accuracy: 0.9597 - val\_loss: 0.0835 - val\_accuracy: 0.884

1

Epoch 11/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0380 - accuracy: 0.9635 - val\_loss: 0.0839 - val\_accuracy: 0.885

3

Epoch 12/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0348 - accuracy: 0.9670 - val\_loss: 0.0858 - val\_accuracy: 0.879

5

Epoch 13/20 30/30 [==============================] - 1s 35ms/step - loss: 0.0325 - accuracy: 0.9707 - val\_loss: 0.0864 - val\_accuracy: 0.879

4

Epoch 14/20

30/30 [==============================] - 1s 26ms/step - loss: 0.0301 - accuracy: 0.9724 - val\_loss: 0.0866 - val\_accuracy: 0.879

7

Epoch 15/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0279 - accuracy: 0.9751 - val\_loss: 0.0879 - val\_accuracy: 0.880

6

Epoch 16/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0259 - accuracy: 0.9777 - val\_loss: 0.0909 - val\_accuracy: 0.875

5

Epoch 17/20 30/30 [==============================] - 1s 24ms/step - loss: 0.0244 - accuracy: 0.9785 - val\_loss: 0.0921 - val\_accuracy: 0.875

2

Epoch 18/20

30/30 [==============================] - 1s 23ms/step - loss: 0.0222 - accuracy: 0.9818 - val\_loss: 0.0903 - val\_accuracy: 0.877

0

Epoch 19/20 30/30 [==============================] - 1s 30ms/step - loss: 0.0209 - accuracy: 0.9833 - val\_loss: 0.0938 - val\_accuracy: 0.873

5

Epoch 20/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0200 - accuracy: 0.9842 - val\_loss: 0.0922 - val\_accuracy: 0.876

6

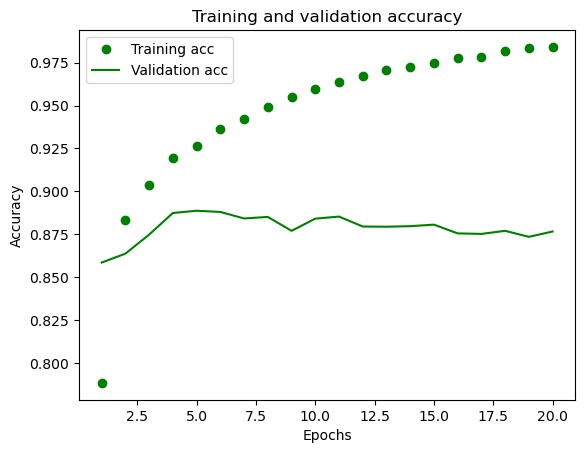
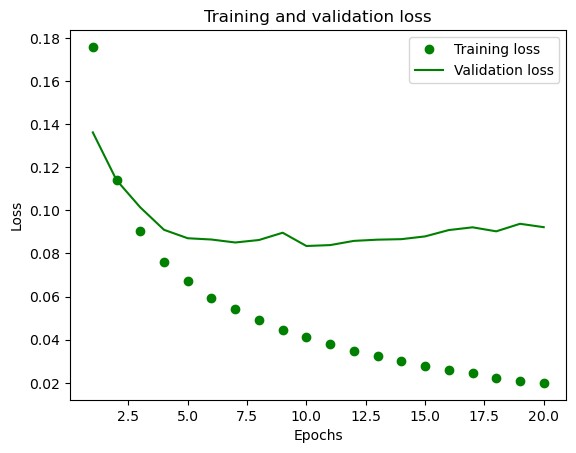
In [61]: historydict\_tanh **=** historytanh\_model**.**history historydict\_tanh**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[61]:

|  |
| --- |
| loss\_value\_tanh**=** historydict\_tanh["loss"] val\_loss\_value\_tanh **=** historydict\_tanh["val\_loss"] epochs\_tanh **=** range(1, len(loss\_value\_tanh) **+** 1)  plt**.**plot(epochs\_tanh, loss\_value\_tanh, "go", label**=**"Training loss") plt**.**plot(epochs\_tanh, val\_loss\_value\_tanh, "g", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf()  acc\_tanh **=** historydict\_tanh["accuracy"] val\_acc\_tanh **=** historydict\_tanh["val\_accuracy"] plt**.**plot(epochs\_tanh, acc\_tanh, "go", label**=**"Training acc") plt**.**plot(epochs\_tanh, val\_acc\_tanh, "g", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [62]:



|  |
| --- |
| tanh**.**fit(x\_train, y\_train, epochs**=**8, batch\_size**=**512) results\_tanh **=** tanh**.**evaluate(x\_test, y\_test) results\_tanh |

In [63]:

Out[63]:

Epoch 1/8 49/49 [==============================] - 1s 10ms/step - loss: 0.0483 - accuracy: 0.9420 Epoch 2/8 49/49 [==============================] - 1s 11ms/step - loss: 0.0413 - accuracy: 0.9524

Epoch 3/8 49/49 [==============================] - 0s 9ms/step - loss: 0.0379 - accuracy: 0.9582 Epoch 4/8 49/49 [==============================] - 0s 8ms/step - loss: 0.0347 - accuracy: 0.9625

Epoch 5/8 49/49 [==============================] - 0s 8ms/step - loss: 0.0322 - accuracy: 0.9658 Epoch 6/8 49/49 [==============================] - 1s 12ms/step - loss: 0.0303 - accuracy: 0.9688

Epoch 7/8 49/49 [==============================] - 1s 17ms/step - loss: 0.0281 - accuracy: 0.9715 Epoch 8/8

49/49 [==============================] - 1s 18ms/step - loss: 0.0267 - accuracy: 0.9727

782/782 [==============================] - 3s 3ms/step - loss: 0.1037 - accuracy: 0.8673

[0.10374121367931366, 0.8672800064086914]

|  |
| --- |
| adam **=** keras**.**Sequential([ layers**.**Dense(16, activation**=**"relu"), layers**.**Dense(16, activation**=**"relu"), layers**.**Dense(16, activation**=**"relu"), layers**.**Dense(1, activation**=**"sigmoid")  ])  adam**.**compile(optimizer**=**'adam',  loss**=**'binary\_crossentropy', metrics**=**['accuracy'])  x\_adam **=** x\_train[:10000] partial\_x\_train **=** x\_train[10000:]  y\_adam **=** y\_train[:10000]  partial\_y\_train **=** y\_train[10000:]    historyadam **=** adam**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_adam, y\_adam)) |

In [ ]: *#Adam Operator with 16 units and 3-layers* In [65]:

Epoch 1/20 30/30 [==============================] - 6s 150ms/step - loss: 0.6036 - accuracy: 0.6982 - val\_loss: 0.4877 - val\_accuracy: 0.85

25

Epoch 2/20 30/30 [==============================] - 1s 38ms/step - loss: 0.3619 - accuracy: 0.8893 - val\_loss: 0.3125 - val\_accuracy: 0.882

6

Epoch 3/20 30/30 [==============================] - 1s 32ms/step - loss: 0.2109 - accuracy: 0.9315 - val\_loss: 0.2768 - val\_accuracy: 0.890

7

Epoch 4/20

30/30 [==============================] - 1s 25ms/step - loss: 0.1409 - accuracy: 0.9553 - val\_loss: 0.2901 - val\_accuracy: 0.888

5

Epoch 5/20 30/30 [==============================] - 1s 39ms/step - loss: 0.0990 - accuracy: 0.9724 - val\_loss: 0.3242 - val\_accuracy: 0.880

6

Epoch 6/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0694 - accuracy: 0.9835 - val\_loss: 0.3646 - val\_accuracy: 0.879

2

Epoch 7/20 30/30 [==============================] - 1s 25ms/step - loss: 0.0485 - accuracy: 0.9899 - val\_loss: 0.4070 - val\_accuracy: 0.876

1

Epoch 8/20

30/30 [==============================] - 1s 27ms/step - loss: 0.0332 - accuracy: 0.9950 - val\_loss: 0.4534 - val\_accuracy: 0.873

5

Epoch 9/20 30/30 [==============================] - 1s 21ms/step - loss: 0.0224 - accuracy: 0.9975 - val\_loss: 0.4975 - val\_accuracy: 0.872

0

Epoch 10/20 30/30 [==============================] - 1s 29ms/step - loss: 0.0161 - accuracy: 0.9989 - val\_loss: 0.5484 - val\_accuracy: 0.869

0

Epoch 11/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0098 - accuracy: 0.9996 - val\_loss: 0.5823 - val\_accuracy: 0.870

7

Epoch 12/20

30/30 [==============================] - 1s 28ms/step - loss: 0.0069 - accuracy: 0.9999 - val\_loss: 0.6184 - val\_accuracy: 0.869

6

Epoch 13/20 30/30 [==============================] - 1s 26ms/step - loss: 0.0052 - accuracy: 0.9999 - val\_loss: 0.6534 - val\_accuracy: 0.868

0

Epoch 14/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0040 - accuracy: 0.9999 - val\_loss: 0.6805 - val\_accuracy: 0.867

6

Epoch 15/20 30/30 [==============================] - 1s 29ms/step - loss: 0.0033 - accuracy: 0.9999 - val\_loss: 0.7050 - val\_accuracy: 0.867

9

Epoch 16/20

30/30 [==============================] - 1s 28ms/step - loss: 0.0027 - accuracy: 0.9999 - val\_loss: 0.7297 - val\_accuracy: 0.867

4

Epoch 17/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0023 - accuracy: 0.9999 - val\_loss: 0.7517 - val\_accuracy: 0.867

1

Epoch 18/20 30/30 [==============================] - 1s 24ms/step - loss: 0.0020 - accuracy: 0.9999 - val\_loss: 0.7707 - val\_accuracy: 0.867

1

Epoch 19/20 30/30 [==============================] - 1s 27ms/step - loss: 0.0017 - accuracy: 0.9999 - val\_loss: 0.7906 - val\_accuracy: 0.866

7

Epoch 20/20

30/30 [==============================] - 1s 22ms/step - loss: 0.0015 - accuracy: 0.9999 - val\_loss: 0.8090 - val\_accuracy: 0.866

4

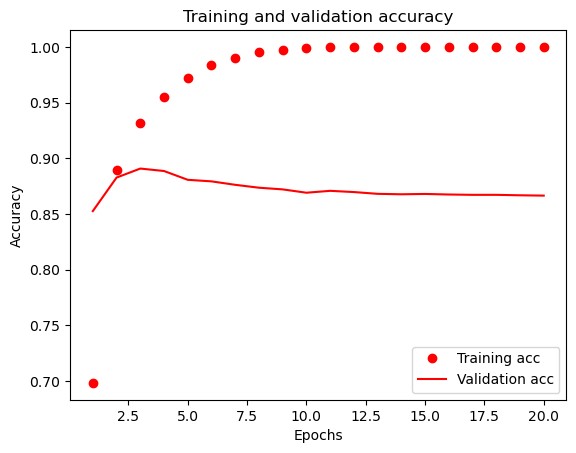
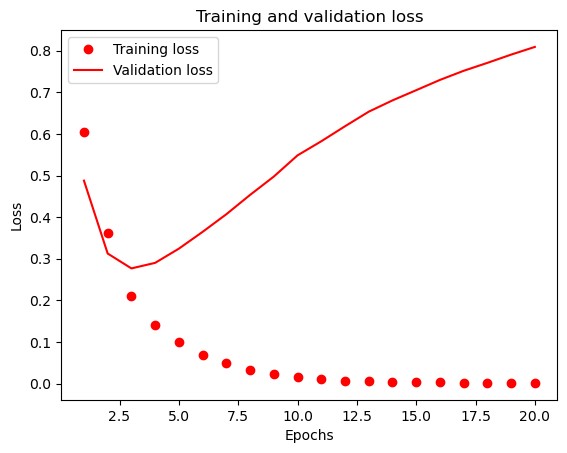
In [66]: historydict\_adam **=** historyadam**.**history historydict\_adam**.**keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

Out[66]:

|  |
| --- |
| loss\_value\_adam **=** historydict\_adam["loss"] val\_loss\_value\_adam **=** historydict\_adam["val\_loss"] epochs\_adam **=** range(1, len(loss\_value\_adam) **+** 1)  plt**.**plot(epochs\_adam, loss\_value\_adam, "ro", label**=**"Training loss") plt**.**plot(epochs\_adam, val\_loss\_value\_adam, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf()  acc\_adam **=** historydict\_adam["accuracy"] val\_acc\_adam **=** historydict\_adam["val\_accuracy"]  plt**.**plot(epochs\_adam, acc\_adam, "ro", label**=**"Training acc") plt**.**plot(epochs\_adam, val\_acc\_adam, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

In [67]:



In [68]: adam**.**fit(x\_train, y\_train, epochs**=**4, batch\_size**=**512) results\_adam **=** adam**.**evaluate(x\_test, y\_test) results\_adam

Epoch 1/4 49/49 [==============================] - 1s 21ms/step - loss: 0.2663 - accuracy: 0.9332 Epoch 2/4 49/49 [==============================] - 1s 20ms/step - loss: 0.1215 - accuracy: 0.9599

Epoch 3/4 49/49 [==============================] - 1s 22ms/step - loss: 0.0852 - accuracy: 0.9752 Epoch 4/4

49/49 [==============================] - 1s 18ms/step - loss: 0.0662 - accuracy: 0.9820

782/782 [==============================] - 3s 3ms/step - loss: 0.5339 - accuracy: 0.8584

[0.5338543057441711, 0.8583999872207642] Out[68]:

|  |
| --- |
| *#Regularization with 16 units and 2 layers.*  **from** tensorflow.keras **import** regularizers regularization **=** 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")  ])  regularization**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])  history\_regularization **=** regularization**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512, validation\_data**=**(x\_val, y\_val)) historydict\_regularization **=** history\_regularization**.**history historydict\_regularization**.**keys() |

In [69]:

Out[69]: In [70]:

Epoch 1/20 30/30 [==============================] - 6s 136ms/step - loss: 0.5784 - accuracy: 0.7779 - val\_loss: 0.4522 - val\_accuracy: 0.86

05

Epoch 2/20 30/30 [==============================] - 0s 16ms/step - loss: 0.3823 - accuracy: 0.8909 - val\_loss: 0.3673 - val\_accuracy: 0.886

0

Epoch 3/20 30/30 [==============================] - 1s 17ms/step - loss: 0.3070 - accuracy: 0.9141 - val\_loss: 0.3510 - val\_accuracy: 0.878

6

Epoch 4/20

30/30 [==============================] - 0s 15ms/step - loss: 0.2636 - accuracy: 0.9295 - val\_loss: 0.3619 - val\_accuracy: 0.872

2

Epoch 5/20 30/30 [==============================] - 0s 14ms/step - loss: 0.2387 - accuracy: 0.9401 - val\_loss: 0.3461 - val\_accuracy: 0.882

0

Epoch 6/20 30/30 [==============================] - 1s 20ms/step - loss: 0.2227 - accuracy: 0.9433 - val\_loss: 0.3784 - val\_accuracy: 0.871

4

Epoch 7/20 30/30 [==============================] - 1s 32ms/step - loss: 0.2061 - accuracy: 0.9523 - val\_loss: 0.3402 - val\_accuracy: 0.884

6

Epoch 8/20

30/30 [==============================] - 1s 31ms/step - loss: 0.1955 - accuracy: 0.9567 - val\_loss: 0.3719 - val\_accuracy: 0.873

6

Epoch 9/20 30/30 [==============================] - 1s 28ms/step - loss: 0.1909 - accuracy: 0.9568 - val\_loss: 0.3567 - val\_accuracy: 0.883

2

Epoch 10/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1836 - accuracy: 0.9614 - val\_loss: 0.3989 - val\_accuracy: 0.867

5

Epoch 11/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1761 - accuracy: 0.9636 - val\_loss: 0.3769 - val\_accuracy: 0.880

5

Epoch 12/20

30/30 [==============================] - 1s 32ms/step - loss: 0.1678 - accuracy: 0.9688 - val\_loss: 0.4151 - val\_accuracy: 0.867

2

Epoch 13/20 30/30 [==============================] - 1s 31ms/step - loss: 0.1652 - accuracy: 0.9686 - val\_loss: 0.3867 - val\_accuracy: 0.877

5

Epoch 14/20 30/30 [==============================] - 1s 25ms/step - loss: 0.1592 - accuracy: 0.9705 - val\_loss: 0.4354 - val\_accuracy: 0.870

1

Epoch 15/20 30/30 [==============================] - 1s 24ms/step - loss: 0.1563 - accuracy: 0.9721 - val\_loss: 0.4031 - val\_accuracy: 0.876

3

Epoch 16/20

30/30 [==============================] - 1s 26ms/step - loss: 0.1527 - accuracy: 0.9727 - val\_loss: 0.4117 - val\_accuracy: 0.876

7

Epoch 17/20 30/30 [==============================] - 1s 28ms/step - loss: 0.1474 - accuracy: 0.9755 - val\_loss: 0.4453 - val\_accuracy: 0.863

0

Epoch 18/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1533 - accuracy: 0.9737 - val\_loss: 0.4225 - val\_accuracy: 0.876

4

Epoch 19/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1417 - accuracy: 0.9781 - val\_loss: 0.4284 - val\_accuracy: 0.874

2

Epoch 20/20

30/30 [==============================] - 1s 25ms/step - loss: 0.1426 - accuracy: 0.9773 - val\_loss: 0.4798 - val\_accuracy: 0.866

8

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

|  |
| --- |
| loss\_valu **=** historydict\_regularization["loss"]  val\_loss\_value\_r **=** historydict\_regularization["val\_loss"] epochs\_r **=** range(1, len(loss\_valu) **+** 1)  plt**.**plot(epochs\_r, loss\_valu, "go", label**=**"Training loss") plt**.**plot(epochs\_r, val\_loss\_value\_r, "g", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf()  acc\_r **=** historydict\_regularization["accuracy"] val\_acc\_r **=** historydict\_regularization["val\_accuracy"] plt**.**plot(epochs\_r, acc\_r, "go", label**=**"Training acc") plt**.**plot(epochs\_r, val\_acc\_r, "g", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show() |

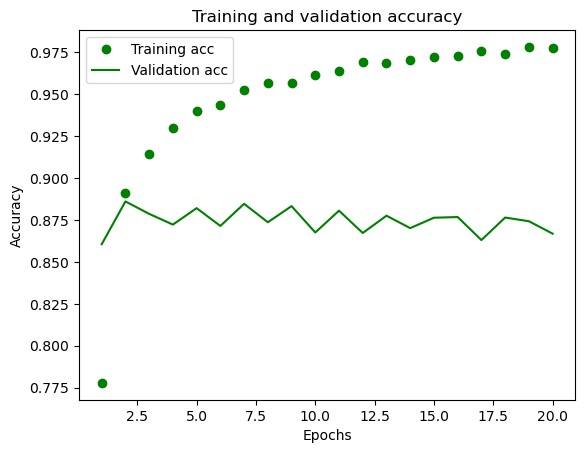
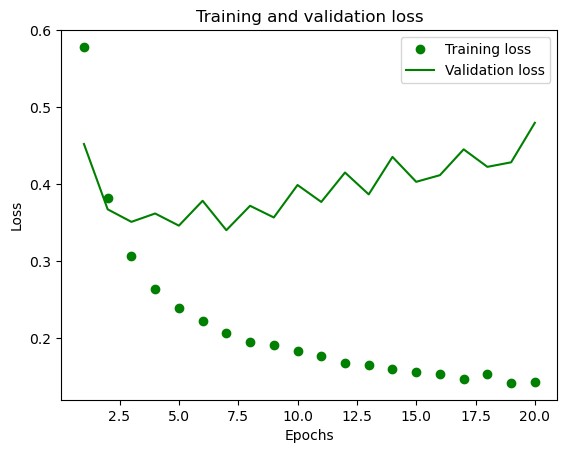
In [71]: regularization**.**fit(x\_train, y\_train, epochs**=**8, batch\_size**=**512) results\_regularization **=** regularization**.**evaluate(x\_test, y\_test) results\_regularization

Epoch 1/8 49/49 [==============================] - 1s 9ms/step - loss: 0.2578 - accuracy: 0.9334 Epoch 2/8 49/49 [==============================] - 0s 9ms/step - loss: 0.2137 - accuracy: 0.9465

Epoch 3/8 49/49 [==============================] - 1s 10ms/step - loss: 0.2037 - accuracy: 0.9466 Epoch 4/8 49/49 [==============================] - 0s 8ms/step - loss: 0.1922 - accuracy: 0.9520

Epoch 5/8 49/49 [==============================] - 1s 16ms/step - loss: 0.1897 - accuracy: 0.9532 Epoch 6/8 49/49 [==============================] - 1s 18ms/step - loss: 0.1814 - accuracy: 0.9550

Epoch 7/8 49/49 [==============================] - 1s 18ms/step - loss: 0.1803 - accuracy: 0.9563 Epoch 8/8



49/49 [==============================] - 1s 17ms/step - loss: 0.1834 - accuracy: 0.9548

782/782 [==============================] - 3s 4ms/step - loss: 0.4246 - accuracy: 0.8660

[0.42463696002960205, 0.8659600019454956] Out[71]:

In [ ]: Dropout function **with** 16 units **and** 3**-**layers

In [72]: *##Dropout* **from** tensorflow.keras **import** regularizers Dropout **=** 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),

Out[72]: In [73]:

layers**.**Dense(1, activation**=**"sigmoid")

])

Dropout**.**compile(optimizer**=**"rmsprop", loss**=**"binary\_crossentropy", metrics**=**["accuracy"])

history\_Dropout **=** Dropout**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512, validation\_data**=**(x\_val, y\_val)) historydict\_Dropout **=** history\_Dropout**.**history historydict\_Dropout**.**keys()

Epoch 1/20

30/30 [==============================] - 7s 163ms/step - loss: 0.6715 - accuracy: 0.5756 - val\_loss: 0.6221 - val\_accuracy: 0.80

73

Epoch 2/20 30/30 [==============================] - 1s 17ms/step - loss: 0.6037 - accuracy: 0.6721 - val\_loss: 0.5253 - val\_accuracy: 0.840

4

Epoch 3/20 30/30 [==============================] - 1s 19ms/step - loss: 0.5367 - accuracy: 0.7412 - val\_loss: 0.4415 - val\_accuracy: 0.872

1

Epoch 4/20 30/30 [==============================] - 0s 16ms/step - loss: 0.4772 - accuracy: 0.7904 - val\_loss: 0.3704 - val\_accuracy: 0.882

0

Epoch 5/20

30/30 [==============================] - 0s 16ms/step - loss: 0.4196 - accuracy: 0.8344 - val\_loss: 0.3248 - val\_accuracy: 0.881

6

Epoch 6/20 30/30 [==============================] - 1s 30ms/step - loss: 0.3800 - accuracy: 0.8552 - val\_loss: 0.3041 - val\_accuracy: 0.882

6

Epoch 7/20 30/30 [==============================] - 1s 30ms/step - loss: 0.3374 - accuracy: 0.8776 - val\_loss: 0.2861 - val\_accuracy: 0.886

6

Epoch 8/20 30/30 [==============================] - 1s 28ms/step - loss: 0.3068 - accuracy: 0.8911 - val\_loss: 0.2928 - val\_accuracy: 0.886

7

Epoch 9/20

30/30 [==============================] - 1s 28ms/step - loss: 0.2871 - accuracy: 0.9008 - val\_loss: 0.3245 - val\_accuracy: 0.869

4

Epoch 10/20 30/30 [==============================] - 1s 27ms/step - loss: 0.2575 - accuracy: 0.9086 - val\_loss: 0.3102 - val\_accuracy: 0.885

8

Epoch 11/20 30/30 [==============================] - 1s 29ms/step - loss: 0.2266 - accuracy: 0.9216 - val\_loss: 0.3510 - val\_accuracy: 0.883

8

Epoch 12/20 30/30 [==============================] - 1s 26ms/step - loss: 0.2163 - accuracy: 0.9243 - val\_loss: 0.3466 - val\_accuracy: 0.882

9

Epoch 13/20

30/30 [==============================] - 1s 31ms/step - loss: 0.2000 - accuracy: 0.9292 - val\_loss: 0.3602 - val\_accuracy: 0.883

0

Epoch 14/20 30/30 [==============================] - 1s 33ms/step - loss: 0.1895 - accuracy: 0.9321 - val\_loss: 0.3862 - val\_accuracy: 0.881

4

Epoch 15/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1712 - accuracy: 0.9361 - val\_loss: 0.4362 - val\_accuracy: 0.883

2

Epoch 16/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1621 - accuracy: 0.9394 - val\_loss: 0.4820 - val\_accuracy: 0.883

6

Epoch 17/20

30/30 [==============================] - 1s 28ms/step - loss: 0.1641 - accuracy: 0.9421 - val\_loss: 0.4954 - val\_accuracy: 0.883

5

Epoch 18/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1489 - accuracy: 0.9449 - val\_loss: 0.5074 - val\_accuracy: 0.882

2

Epoch 19/20 30/30 [==============================] - 1s 29ms/step - loss: 0.1437 - accuracy: 0.9465 - val\_loss: 0.5443 - val\_accuracy: 0.881

9

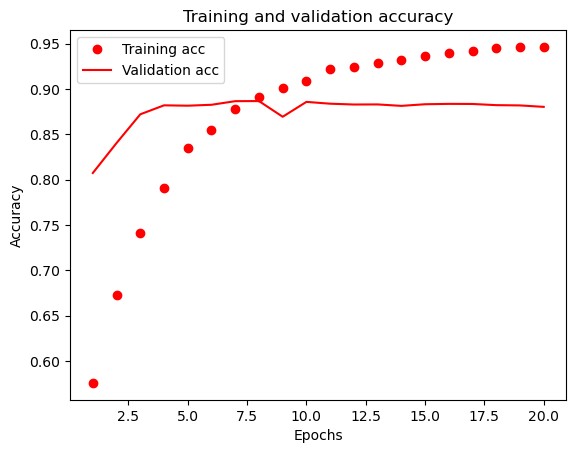
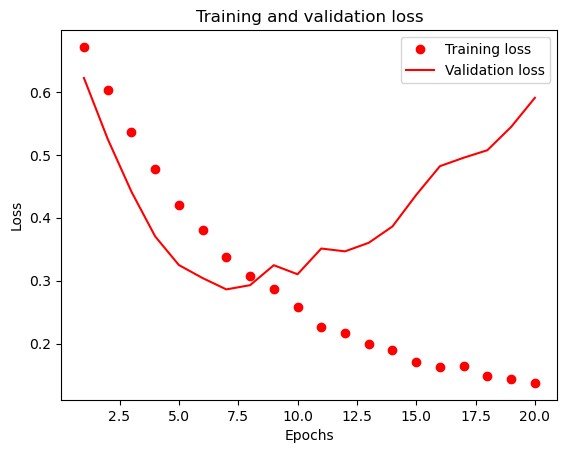
Epoch 20/20 30/30 [==============================] - 1s 27ms/step - loss: 0.1375 - accuracy: 0.9460 - val\_loss: 0.5906 - val\_accuracy: 0.880

3

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

|  |
| --- |
| loss\_val **=** historydict\_Dropout["loss"] val\_loss\_val\_d **=** historydict\_Dropout["val\_loss"] epochs\_d **=** range(1, len(loss\_val) **+** 1)  plt**.**plot(epochs\_d, loss\_val, "ro", label**=**"Training loss") plt**.**plot(epochs\_d, val\_loss\_val\_d, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() plt**.**show()  plt**.**clf()  acc\_d **=** historydict\_Dropout["accuracy"] val\_acc\_d **=** historydict\_Dropout["val\_accuracy"] plt**.**plot(epochs\_d, acc\_d, "ro", label**=**"Training acc") |

plt**.**plot(epochs\_d, val\_acc\_d, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show()



In [74]: Dropout**.**fit(x\_train, y\_train, epochs**=**8, batch\_size**=**512) results\_Dropout **=** Dropout**.**evaluate(x\_test, y\_test) results\_Dropout

Epoch 1/8 49/49 [==============================] - 1s 16ms/step - loss: 0.3112 - accuracy: 0.8938

Epoch 2/8

49/49 [==============================] - 1s 18ms/step - loss: 0.2732 - accuracy: 0.9038

Epoch 3/8 49/49 [==============================] - 1s 21ms/step - loss: 0.2421 - accuracy: 0.9097

Epoch 4/8

49/49 [==============================] - 1s 19ms/step - loss: 0.2278 - accuracy: 0.9134

Epoch 5/8 49/49 [==============================] - 1s 19ms/step - loss: 0.2121 - accuracy: 0.9208

Epoch 6/8

49/49 [==============================] - 1s 21ms/step - loss: 0.1988 - accuracy: 0.9234

Epoch 7/8 49/49 [==============================] - 1s 21ms/step - loss: 0.1968 - accuracy: 0.9238

Epoch 8/8

49/49 [==============================] - 1s 20ms/step - loss: 0.1887 - accuracy: 0.9230

782/782 [==============================] - 3s 3ms/step - loss: 0.4874 - accuracy: 0.8756

[0.4873920679092407, 0.8756399750709534] Out[74]:

In [ ]: *#Training the model with hyper tuned parameters with 32 units and 3 -layers*

In [75]: *#Training model with hyper tuned parameters* **from** tensorflow.keras **import** regularizers

Out[75]: In [76]:

|  |
| --- |
| Hyper **=** 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")  ])  Hyper**.**compile(optimizer**=**"rmsprop", loss**=**"mse", metrics**=**["accuracy"])  history\_Hyper **=** Hyper**.**fit(partial\_x\_train, partial\_y\_train, epochs**=**20, batch\_size**=**512,  validation\_data**=**(x\_val, y\_val)) history\_dictHyper **=** history\_Hyper**.**history history\_dictHyper**.**keys() |

Epoch 1/20 30/30 [==============================] - 7s 151ms/step - loss: 0.2389 - accuracy: 0.6247 - val\_loss: 0.1812 - val\_accuracy: 0.82

92

Epoch 2/20 30/30 [==============================] - 2s 54ms/step - loss: 0.1804 - accuracy: 0.7692 - val\_loss: 0.1310 - val\_accuracy: 0.847

3

Epoch 3/20 30/30 [==============================] - 1s 47ms/step - loss: 0.1412 - accuracy: 0.8346 - val\_loss: 0.1110 - val\_accuracy: 0.867

1

Epoch 4/20

30/30 [==============================] - 1s 43ms/step - loss: 0.1169 - accuracy: 0.8739 - val\_loss: 0.0992 - val\_accuracy: 0.885

1

Epoch 5/20 30/30 [==============================] - 1s 43ms/step - loss: 0.0968 - accuracy: 0.8995 - val\_loss: 0.1017 - val\_accuracy: 0.881

2

Epoch 6/20 30/30 [==============================] - 1s 41ms/step - loss: 0.0833 - accuracy: 0.9175 - val\_loss: 0.0981 - val\_accuracy: 0.889

1

Epoch 7/20 30/30 [==============================] - 1s 41ms/step - loss: 0.0750 - accuracy: 0.9261 - val\_loss: 0.1025 - val\_accuracy: 0.886

7

Epoch 8/20

30/30 [==============================] - 1s 40ms/step - loss: 0.0687 - accuracy: 0.9354 - val\_loss: 0.1044 - val\_accuracy: 0.885

6

Epoch 9/20 30/30 [==============================] - 1s 39ms/step - loss: 0.0626 - accuracy: 0.9412 - val\_loss: 0.1052 - val\_accuracy: 0.886

2

Epoch 10/20 30/30 [==============================] - 1s 24ms/step - loss: 0.0569 - accuracy: 0.9506 - val\_loss: 0.1085 - val\_accuracy: 0.885

0

Epoch 11/20 30/30 [==============================] - 1s 21ms/step - loss: 0.0527 - accuracy: 0.9542 - val\_loss: 0.1082 - val\_accuracy: 0.884

4

Epoch 12/20

30/30 [==============================] - 1s 37ms/step - loss: 0.0500 - accuracy: 0.9593 - val\_loss: 0.1104 - val\_accuracy: 0.884

7

Epoch 13/20 30/30 [==============================] - 1s 40ms/step - loss: 0.0463 - accuracy: 0.9624 - val\_loss: 0.1159 - val\_accuracy: 0.875

9

Epoch 14/20 30/30 [==============================] - 1s 40ms/step - loss: 0.0452 - accuracy: 0.9633 - val\_loss: 0.1154 - val\_accuracy: 0.878

5

Epoch 15/20 30/30 [==============================] - 1s 39ms/step - loss: 0.0437 - accuracy: 0.9662 - val\_loss: 0.1130 - val\_accuracy: 0.884

5

Epoch 16/20

30/30 [==============================] - 1s 38ms/step - loss: 0.0436 - accuracy: 0.9647 - val\_loss: 0.1122 - val\_accuracy: 0.884

0

Epoch 17/20 30/30 [==============================] - 1s 42ms/step - loss: 0.0411 - accuracy: 0.9685 - val\_loss: 0.1151 - val\_accuracy: 0.881

7

Epoch 18/20 30/30 [==============================] - 1s 41ms/step - loss: 0.0393 - accuracy: 0.9705 - val\_loss: 0.1141 - val\_accuracy: 0.882

2

Epoch 19/20 30/30 [==============================] - 1s 43ms/step - loss: 0.0397 - accuracy: 0.9686 - val\_loss: 0.1164 - val\_accuracy: 0.882

5

Epoch 20/20

30/30 [==============================] - 1s 39ms/step - loss: 0.0371 - accuracy: 0.9720 - val\_loss: 0.1165 - val\_accuracy: 0.880

8

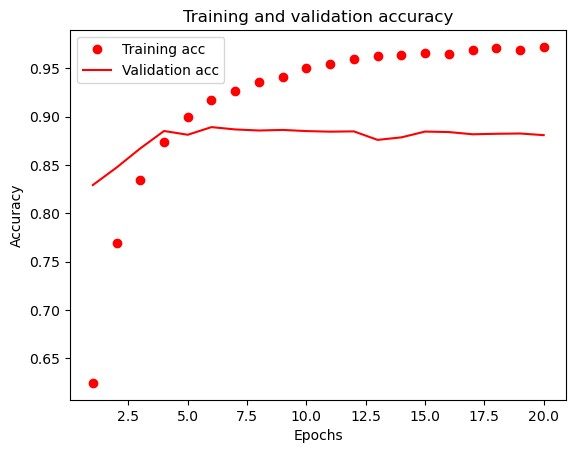
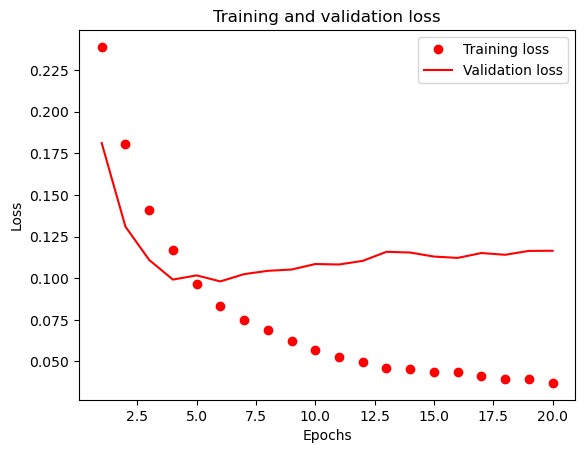
dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

|  |
| --- |
| loss\_va\_h **=** history\_dictHyper["loss"] val\_loss\_va\_h **=** history\_dictHyper["val\_loss"] epochs\_h **=** range(1, len(loss\_va\_h) **+** 1)  plt**.**plot(epochs\_h, loss\_va\_h, "ro", label**=**"Training loss") plt**.**plot(epochs\_h, val\_loss\_va\_h, "r", label**=**"Validation loss") plt**.**title("Training and validation loss") plt**.**xlabel("Epochs") plt**.**ylabel("Loss") plt**.**legend() |

plt**.**show()

plt**.**clf()

acc\_h **=** history\_dictHyper["accuracy"] val\_acc\_h **=** history\_dictHyper["val\_accuracy"] plt**.**plot(epochs\_h, acc\_h, "ro", label**=**"Training acc") plt**.**plot(epochs\_h, val\_acc\_h, "r", label**=**"Validation acc") plt**.**title("Training and validation accuracy") plt**.**xlabel("Epochs") plt**.**ylabel("Accuracy") plt**.**legend() plt**.**show()



|  |
| --- |
| Hyper**.**fit(x\_train, y\_train, epochs**=**8, batch\_size**=**512) results\_Hyper **=** Hyper**.**evaluate(x\_test, y\_test) results\_Hyper |

In [77]:

Epoch 1/8 49/49 [==============================] - 1s 14ms/step - loss: 0.0729 - accuracy: 0.9285

Epoch 2/8 49/49 [==============================] - 1s 29ms/step - loss: 0.0662 - accuracy: 0.9354 Epoch 3/8 49/49 [==============================] - 1s 29ms/step - loss: 0.0613 - accuracy: 0.9425

Epoch 4/8 49/49 [==============================] - 1s 31ms/step - loss: 0.0572 - accuracy: 0.9477 Epoch 5/8 49/49 [==============================] - 1s 28ms/step - loss: 0.0557 - accuracy: 0.9494

Epoch 6/8 49/49 [==============================] - 1s 25ms/step - loss: 0.0536 - accuracy: 0.9521 Epoch 7/8 49/49 [==============================] - 1s 18ms/step - loss: 0.0511 - accuracy: 0.9533

Epoch 8/8

49/49 [==============================] - 1s 27ms/step - loss: 0.0490 - accuracy: 0.9564

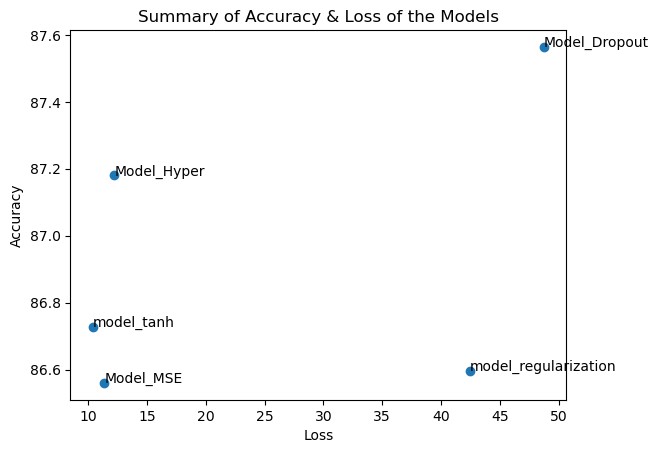
782/782 [==============================] - 4s 4ms/step - loss: 0.1219 - accuracy: 0.8718 [0.12194811552762985, 0.8718000054359436] Out[77]:

|  |
| --- |
| *#Summary*  Models\_Loss**=** np**.**array([results\_Dropout[0],results\_Hyper[0],results\_MSE[0],results\_regularization[0],results\_tanh[0]])**\***100  Models\_Loss  Models\_Accuracy**=** np**.**array([results\_Dropout[1],results\_Hyper[1],results\_MSE[1],results\_regularization[1],results\_tanh[1]])**\***100 Models\_Accuracy  Labels**=**['Model\_Dropout','Model\_Hyper','Model\_MSE','model\_regularization','model\_tanh'] plt**.**clf() |

In [78]:

|  |
| --- |
| *#Compilation*  fig, ax **=** plt**.**subplots()  ax**.**scatter(Models\_Loss,Models\_Accuracy) **for** i, txt **in** enumerate(Labels):  ax**.**annotate(txt, (Models\_Loss[i],Models\_Accuracy[i] )) plt**.**title("Summary of Accuracy & Loss of the Models") plt**.**ylabel("Accuracy") plt**.**xlabel("Loss") plt**.**show() |

<Figure size 640x480 with 0 Axes> In [79]:



In [ ]: *#summary*

We created a baseline neural network model **with** a single hidden layer of 16 units after importing the data **and** adjusting the word

We next examined the effects of changing the number of hidden units **in** the layers—32 **and** 64 units, to be precise**.** We discovered t

In [ ]: *#conclusion*

Neural network models **with** different configurations showed different patterns of sensitivity **and** loss**.** Among these, the Model\_Hyp Overall, the performance of the models was much enhanced by the addition of dropout regularization, **with** Model hyper being the mo