meghana-aml1-1

September 23, 2024

Introduction: This project aims to investigate different methods for enhancing the performance of a neural network model using the IMDb dataset. We will adapt an existing neural network and evaluate the outcomes of various strategies, including altering the number of hidden layers and units, adjusting the loss function and activation function, and implementing regularization techniques such as dropout.

Dataset: The dataset utilized is the IMDb collection, which features movie reviews classified as positive or negative. It comprises 25,000 reviews for training and an additional 25,000 for testing

[1]:
```
from numpy.random import seed
seed(123)
from tensorflow.keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(
    num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/imdb.npz
17464789/17464789          1s
0us/step
```

[2]:
```
train_data
```

[2]:
```
array([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, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12,
16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5,
62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130,
12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28,
77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5,
723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381,
15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476,
26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38,
1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32]),
       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, 229, 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, 349, 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, 245, 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, 15, 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, 1338,
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, 563, 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, 500, 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, 131, 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, 372, 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, 197, 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, 25, 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, 80, 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, 1597, 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])],
      dtype=object)
```

[ ]:
```
train_labels[0]
```

[ ]: 1

[ ]:
```
len(train_labels)
```

[ ]: 25000

[ ]:
```
len(train_labels)
```

[ ]: 25000

[ ]:
```
test_data
```

[ ]:
```
array([list([1, 591, 202, 14, 31, 6, 717, 10, 10, 2, 2, 5, 4, 360, 7, 4, 177,
5760, 394, 354, 4, 123, 9, 1035, 1035, 1035, 10, 10, 13, 92, 124, 89, 488, 7944,
100, 28, 1668, 14, 31, 23, 27, 7479, 29, 220, 468, 8, 124, 14, 286, 170, 8, 157,
46, 5, 27, 239, 16, 179, 2, 38, 32, 25, 7944, 451, 202, 14, 6, 717]),
       list([1, 14, 22, 3443, 6, 176, 7, 5063, 88, 12, 2679, 23, 1310, 5, 109,
943, 4, 114, 9, 55, 606, 5, 111, 7, 4, 139, 193, 273, 23, 4, 172, 270, 11, 7216,
2, 4, 8463, 2801, 109, 1603, 21, 4, 22, 3861, 8, 6, 1193, 1330, 10, 10, 4, 105,
987, 35, 841, 2, 19, 861, 1074, 5, 1987, 2, 45, 55, 221, 15, 670, 5304, 526, 14,
1069, 4, 405, 5, 2438, 7, 27, 85, 108, 131, 4, 5045, 5304, 3884, 405, 9, 3523,
133, 5, 50, 13, 104, 51, 66, 166, 14, 22, 157, 9, 4, 530, 239, 34, 8463, 2801,
45, 407, 31, 7, 41, 3778, 105, 21, 59, 299, 12, 38, 950, 5, 4521, 15, 45, 629,
488, 2733, 127, 6, 52, 292, 17, 4, 6936, 185, 132, 1988, 5304, 1799, 488, 2693,
47, 6, 392, 173, 4, 2, 4378, 270, 2352, 4, 1500, 7, 4, 65, 55, 73, 11, 346, 14,
20, 9, 6, 976, 2078, 7, 5293, 861, 2, 5, 4182, 30, 3127, 2, 56, 4, 841, 5, 990,
692, 8, 4, 1669, 398, 229, 10, 10, 13, 2822, 670, 5304, 14, 9, 31, 7, 27, 111,
108, 15, 2033, 19, 7836, 1429, 875, 551, 14, 22, 9, 1193, 21, 45, 4829, 5, 45,
252, 8, 2, 6, 565, 921, 3639, 39, 4, 529, 48, 25, 181, 8, 67, 35, 1732, 22, 49,
238, 60, 135, 1162, 14, 9, 290, 4, 58, 10, 10, 472, 45, 55, 878, 8, 169, 11,
374, 5687, 25, 203, 28, 8, 818, 12, 125, 4, 3077]),
       list([1, 111, 748, 4368, 1133, 2, 2, 4, 87, 1551, 1262, 7, 31, 318, 9459,
7, 4, 498, 5076, 748, 63, 29, 5161, 220, 686, 2, 5, 17, 12, 575, 220, 2507, 17,
6, 185, 132, 2, 16, 53, 928, 11, 2, 74, 4, 438, 21, 27, 2, 589, 8, 22, 107, 2,
2, 997, 1638, 8, 35, 2076, 9019, 11, 22, 231, 54, 29, 1706, 29, 100, 2, 2425,
34, 2, 8738, 2, 5, 2, 98, 31, 2122, 33, 6, 58, 14, 3808, 1638, 8, 4, 365, 7,
2789, 3761, 356, 346, 4, 2, 1060, 63, 29, 93, 11, 5421, 11, 2, 33, 6, 58, 54,
1270, 431, 748, 7, 32, 2580, 16, 11, 94, 2, 10, 10, 4, 993, 2, 7, 4, 1766, 2634,
2164, 2, 8, 847, 8, 1450, 121, 31, 7, 27, 86, 2663, 2, 16, 6, 465, 993, 2006, 2,
573, 17, 2, 42, 4, 2, 37, 473, 6, 711, 6, 8869, 7, 328, 212, 70, 30, 258, 11,
220, 32, 7, 108, 21, 133, 12, 9, 55, 465, 849, 3711, 53, 33, 2071, 1969, 37, 70,
1144, 4, 5940, 1409, 74, 476, 37, 62, 91, 1329, 169, 4, 1330, 2, 146, 655, 2212,
5, 258, 12, 184, 2, 546, 5, 849, 2, 7, 4, 22, 1436, 18, 631, 1386, 797, 7, 4,
8712, 71, 348, 425, 4320, 1061, 19, 2, 5, 2, 11, 661, 8, 339, 2, 4, 2455, 2, 7,
4, 1962, 10, 10, 263, 787, 9, 270, 11, 6, 9466, 4, 2, 2, 121, 4, 5437, 26, 4434,
19, 68, 1372, 5, 28, 446, 6, 318, 7149, 8, 67, 51, 36, 70, 81, 8, 4392, 2294,
36, 1197, 8, 2, 2, 18, 6, 711, 4, 9909, 26, 2, 1125, 11, 14, 636, 720, 12, 426,
```

```
28, 77, 776, 8, 97, 38, 111, 7489, 6175, 168, 1239, 5189, 137, 2, 18, 27, 173,
9, 2399, 17, 6, 2, 428, 2, 232, 11, 4, 8014, 37, 272, 40, 2708, 247, 30, 656, 6,
2, 54, 2, 3292, 98, 6, 2840, 40, 558, 37, 6093, 98, 4, 2, 1197, 15, 14, 9, 57,
4893, 5, 4659, 6, 275, 711, 7937, 2, 3292, 98, 6, 2, 10, 10, 6639, 19, 14, 2,
267, 162, 711, 37, 5900, 752, 98, 4, 2, 2378, 90, 19, 6, 2, 7, 2, 1810, 2, 4,
4770, 3183, 930, 8, 508, 90, 4, 1317, 8, 4, 2, 17, 2, 3965, 1853, 4, 1494, 8,
4468, 189, 4, 2, 6287, 5774, 4, 4770, 5, 95, 271, 23, 6, 7742, 6063, 2, 5437,
33, 1526, 6, 425, 3155, 2, 4535, 1636, 7, 4, 4669, 2, 469, 4, 4552, 54, 4, 150,
5664, 2, 280, 53, 2, 2, 18, 339, 29, 1978, 27, 7885, 5, 2, 68, 1830, 19, 6571,
2, 4, 1515, 7, 263, 65, 2132, 34, 6, 5680, 7489, 43, 159, 29, 9, 4706, 9, 387,
73, 195, 584, 10, 10, 1069, 4, 58, 810, 54, 14, 6078, 117, 22, 16, 93, 5, 1069,
4, 192, 15, 12, 16, 93, 34, 6, 1766, 2, 33, 4, 5673, 7, 15, 2, 9252, 3286, 325,
12, 62, 30, 776, 8, 67, 14, 17, 6, 2, 44, 148, 687, 2, 203, 42, 203, 24, 28, 69,
2, 6676, 11, 330, 54, 29, 93, 2, 21, 845, 2, 27, 1099, 7, 819, 4, 22, 1407, 17,
6, 2, 787, 7, 2460, 2, 2, 100, 30, 4, 3737, 3617, 3169, 2321, 42, 1898, 11, 4,
3814, 42, 101, 704, 7, 101, 999, 15, 1625, 94, 2926, 180, 5, 9, 9101, 34, 2, 45,
6, 1429, 22, 60, 6, 1220, 31, 11, 94, 6408, 96, 21, 94, 749, 9, 57, 975]),
       …,
       list([1, 13, 1408, 15, 8, 135, 14, 9, 35, 32, 46, 394, 20, 62, 30, 5093,
21, 45, 184, 78, 4, 1492, 910, 769, 2290, 2515, 395, 4257, 5, 1454, 11, 119, 2,
89, 1036, 4, 116, 218, 78, 21, 407, 100, 30, 128, 262, 15, 7, 185, 2280, 284,
1842, 2, 37, 315, 4, 226, 20, 272, 2942, 40, 29, 152, 60, 181, 8, 30, 50, 553,
362, 80, 119, 12, 21, 846, 5518]),
       list([1, 11, 119, 241, 9, 4, 840, 20, 12, 468, 15, 94, 3684, 562, 791,
39, 4, 86, 107, 8, 97, 14, 31, 33, 4, 2960, 7, 743, 46, 1028, 9, 3531, 5, 4,
768, 47, 8, 79, 90, 145, 164, 162, 50, 6, 501, 119, 7, 9, 4, 78, 232, 15, 16,
224, 11, 4, 333, 20, 4, 985, 200, 5, 2, 5, 9, 1861, 8, 79, 357, 4, 20, 47, 220,
57, 206, 139, 11, 12, 5, 55, 117, 212, 13, 1276, 92, 124, 51, 45, 1188, 71, 536,
13, 520, 14, 20, 6, 2302, 7, 470]),
       list([1, 6, 52, 7465, 430, 22, 9, 220, 2594, 8, 28, 2, 519, 3227, 6, 769,
15, 47, 6, 3482, 4067, 8, 114, 5, 33, 222, 31, 55, 184, 704, 5586, 2, 19, 346,
3153, 5, 6, 364, 350, 4, 184, 5586, 9, 133, 1810, 11, 5417, 2, 21, 4, 7298, 2,
570, 50, 2005, 2643, 9, 6, 1249, 17, 6, 2, 2, 21, 17, 6, 1211, 232, 1138, 2249,
29, 266, 56, 96, 346, 194, 308, 9, 194, 21, 29, 218, 1078, 19, 4, 78, 173, 7,
27, 2, 5698, 3406, 718, 2, 9, 6, 6907, 17, 210, 5, 3281, 5677, 47, 77, 395, 14,
172, 173, 18, 2740, 2931, 4517, 82, 127, 27, 173, 11, 6, 392, 217, 21, 50, 9,
57, 65, 12, 2, 53, 40, 35, 390, 7, 11, 4, 3567, 7, 4, 314, 74, 6, 792, 22, 2,
19, 714, 727, 5205, 382, 4, 91, 6533, 439, 19, 14, 20, 9, 1441, 5805, 1118, 4,
756, 25, 124, 4, 31, 12, 16, 93, 804, 34, 2005, 2643])],
      dtype=object)
```

[ ]:
```python
test_labels[0]
```

[ ]: 0

[ ]:
```python
max([max(sequence) for sequence in test_data])
```

[ ]: 9999

1 Translating Reviews into Text

[ ]:
```python
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]])
```

[ ]:
```python
decoded_review
```

[ ]:
```
"? this film was just brilliant casting location scenery story direction
everyone's really suited the part they played and you could just imagine being
there robert ? is an amazing actor and now the same being director ? father came
from the same scottish island as myself so i loved the fact there was a real
connection with this film the witty remarks throughout the film were 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 watch 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 brilliant 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 the whole film but these children are amazing and
should be praised for what they have done don't you think the whole story was so
lovely because it was true and was someone's life after all that was shared with
us all"
```

Data preparation

[ ]:
```python
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
```

Data Vectorization

[ ]:
```python
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

[ ]:
```python

```

[ ]:
```python
x_train[0]
```

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

[ ]:
```
x_test[0]
```

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

Label Vectorization

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

Building model using relu and compiling it

[ ]:
```
from tensorflow import keras
from tensorflow.keras import layers
seed(123)
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"])
```

[ ]:
```
seed(123)
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

[ ]:
```
seed(123)
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 6s 51ms/step - loss: 0.5371 - accuracy:
0.7781 - val_loss: 0.4241 - val_accuracy: 0.8535
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3391 - accuracy:
0.8912 - val_loss: 0.3309 - val_accuracy: 0.8753
Epoch 3/20
30/30 [==============================] - 1s 19ms/step - loss: 0.2471 - accuracy:
0.9202 - val_loss: 0.3044 - val_accuracy: 0.8776
```

```
Epoch 4/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2014 - accuracy:
0.9331 - val_loss: 0.2785 - val_accuracy: 0.8876
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1697 - accuracy:
0.9449 - val_loss: 0.2768 - val_accuracy: 0.8878
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1436 - accuracy:
0.9539 - val_loss: 0.2863 - val_accuracy: 0.8864
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1239 - accuracy:
0.9599 - val_loss: 0.3006 - val_accuracy: 0.8835
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1081 - accuracy:
0.9675 - val_loss: 0.3041 - val_accuracy: 0.8830
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0929 - accuracy:
0.9722 - val_loss: 0.3170 - val_accuracy: 0.8814
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0804 - accuracy:
0.9783 - val_loss: 0.3343 - val_accuracy: 0.8796
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0669 - accuracy:
0.9827 - val_loss: 0.3539 - val_accuracy: 0.8782
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0595 - accuracy:
0.9847 - val_loss: 0.4004 - val_accuracy: 0.8691
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0485 - accuracy:
0.9897 - val_loss: 0.3984 - val_accuracy: 0.8756
Epoch 14/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0439 - accuracy:
0.9899 - val_loss: 0.4157 - val_accuracy: 0.8756
Epoch 15/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0388 - accuracy:
0.9923 - val_loss: 0.4697 - val_accuracy: 0.8690
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0277 - accuracy:
0.9962 - val_loss: 0.4978 - val_accuracy: 0.8658
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0260 - accuracy:
0.9963 - val_loss: 0.4789 - val_accuracy: 0.8721
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0220 - accuracy:
0.9971 - val_loss: 0.5088 - val_accuracy: 0.8712
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0197 - accuracy:
0.9969 - val_loss: 0.5240 - val_accuracy: 0.8703
```

```
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0175 - accuracy:
0.9976 - val_loss: 0.5515 - val_accuracy: 0.8684
```

The training process began with a loss of 0.5371 and an accuracy of 0.7781 on the training set, while the validation set had a loss of 0.4241 and a validation accuracy of 0.8535.

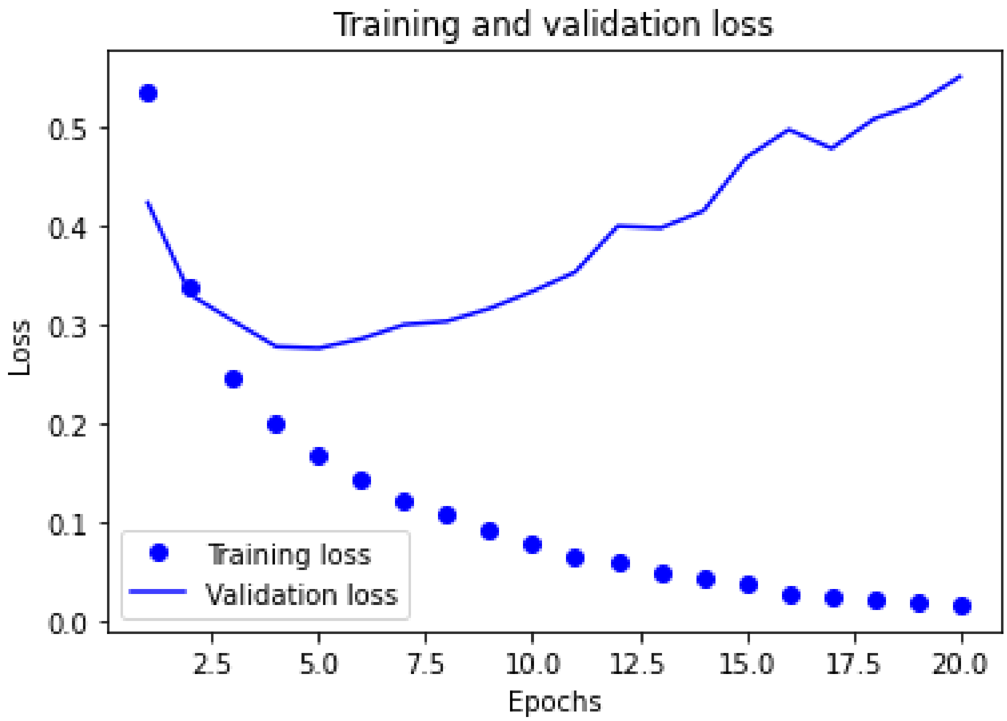
As training continued, both loss and accuracy on the training set improved, culminating in a loss of 0.0175 and an accuracy of 0.9976 by the end of the 20th epoch. However, on the validation set, the model recorded a loss of 0.5515 and an accuracy of 0.8684 at the same epoch, indicating that the model is overfitting to the training data.

[ ]:
```
history_dict = history.history
history_dict.keys()
```

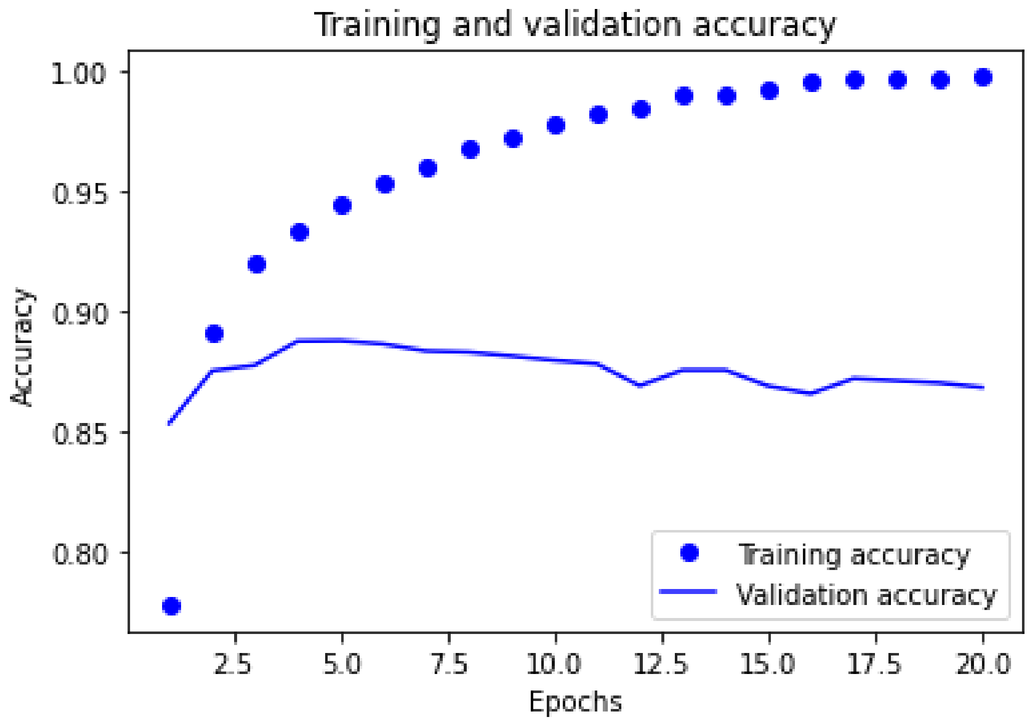
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Plotting the training 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, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



The two graphs indicate that the model's ability to predict new data diminishes after a certain number of epochs, likely due to overfitting the training data. To enhance the model's performance, further analysis may be required, including adjusting hyperparameters or implementing regularization techniques.

Retraining the model

[ ]:
```
np.random.seed(123)
model = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 1s 10ms/step - loss: 0.5137 - accuracy:
0.7962
Epoch 2/4
49/49 [==============================] - 1s 11ms/step - loss: 0.3133 - accuracy:
0.8990
```

```
Epoch 3/4
49/49 [==============================] - 1s 11ms/step - loss: 0.2372 - accuracy:
0.9186
Epoch 4/4
49/49 [==============================] - 1s 11ms/step - loss: 0.1987 - accuracy:
0.9308
782/782 [==============================] - 2s 2ms/step - loss: 0.2828 -
accuracy: 0.8884
```

[ ]:
```
results
```

[ ]:
```
[0.2828458249568939, 0.8883600234985352]
```

The neural network model has achieved an accuracy of 88.84% on the test dataset. The loss value on the test dataset is 0.2828.

[ ]:
```
model.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.28335577],
       [0.9999572 ],
       [0.9212047 ],
       …,
       [0.12263743],
       [0.11844525],
       [0.61341363]], dtype=float32)
```

Building a neural network with 1 hidden layer

[ ]:
```
seed(123)
model1 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

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

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

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


history1 = model1.fit(partial_x_train,
```

```
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.5055 - accuracy:
0.7939 - val_loss: 0.4066 - val_accuracy: 0.8527
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.3314 - accuracy:
0.8942 - val_loss: 0.3244 - val_accuracy: 0.8827
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2629 - accuracy:
0.9143 - val_loss: 0.2949 - val_accuracy: 0.8896
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2208 - accuracy:
0.9291 - val_loss: 0.2809 - val_accuracy: 0.8891
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1928 - accuracy:
0.9399 - val_loss: 0.2761 - val_accuracy: 0.8878
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1716 - accuracy:
0.9453 - val_loss: 0.2765 - val_accuracy: 0.8876
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1539 - accuracy:
0.9527 - val_loss: 0.2848 - val_accuracy: 0.8860
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1407 - accuracy:
0.9582 - val_loss: 0.2950 - val_accuracy: 0.8794
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1277 - accuracy:
0.9632 - val_loss: 0.2849 - val_accuracy: 0.8859
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1177 - accuracy:
0.9670 - val_loss: 0.2946 - val_accuracy: 0.8817
Epoch 11/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1068 - accuracy:
0.9703 - val_loss: 0.3005 - val_accuracy: 0.8806
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0996 - accuracy:
0.9732 - val_loss: 0.3050 - val_accuracy: 0.8819
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0907 - accuracy:
0.9769 - val_loss: 0.3119 - val_accuracy: 0.8812
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0835 - accuracy:
0.9801 - val_loss: 0.3252 - val_accuracy: 0.8821
```

```
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0781 - accuracy:
0.9818 - val_loss: 0.3313 - val_accuracy: 0.8815
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0717 - accuracy:
0.9844 - val_loss: 0.3454 - val_accuracy: 0.8751
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0654 - accuracy:
0.9870 - val_loss: 0.3604 - val_accuracy: 0.8775
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0621 - accuracy:
0.9883 - val_loss: 0.3581 - val_accuracy: 0.8761
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0558 - accuracy:
0.9895 - val_loss: 0.3928 - val_accuracy: 0.8742
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0519 - accuracy:
0.9916 - val_loss: 0.3905 - val_accuracy: 0.8715
```

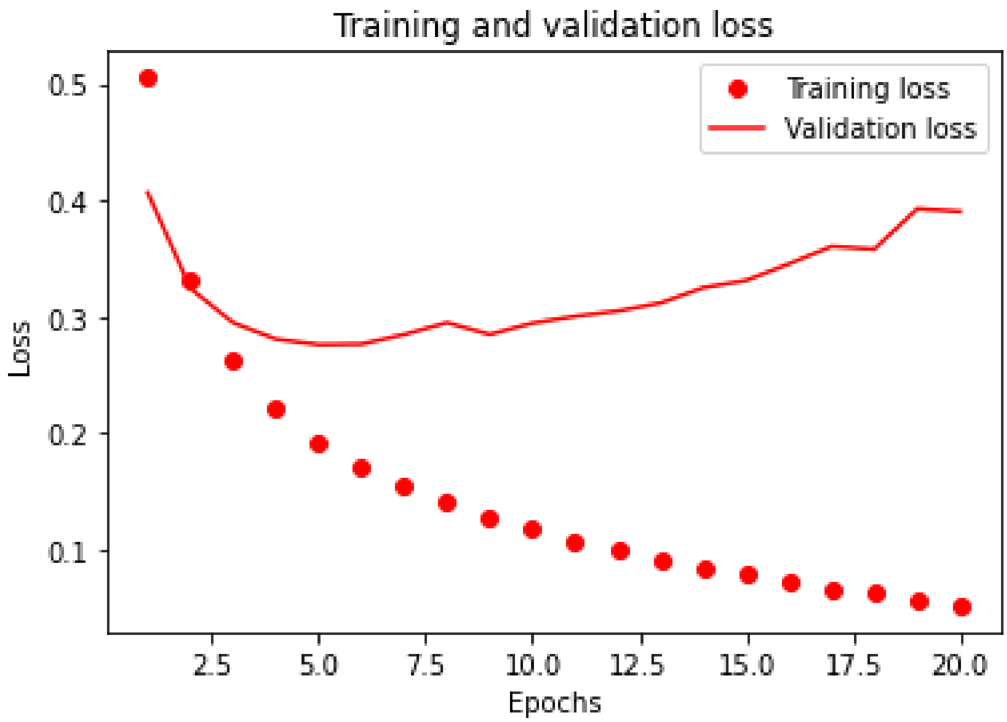
[ ]:
```
history_dict = history1.history
history_dict.keys()
```

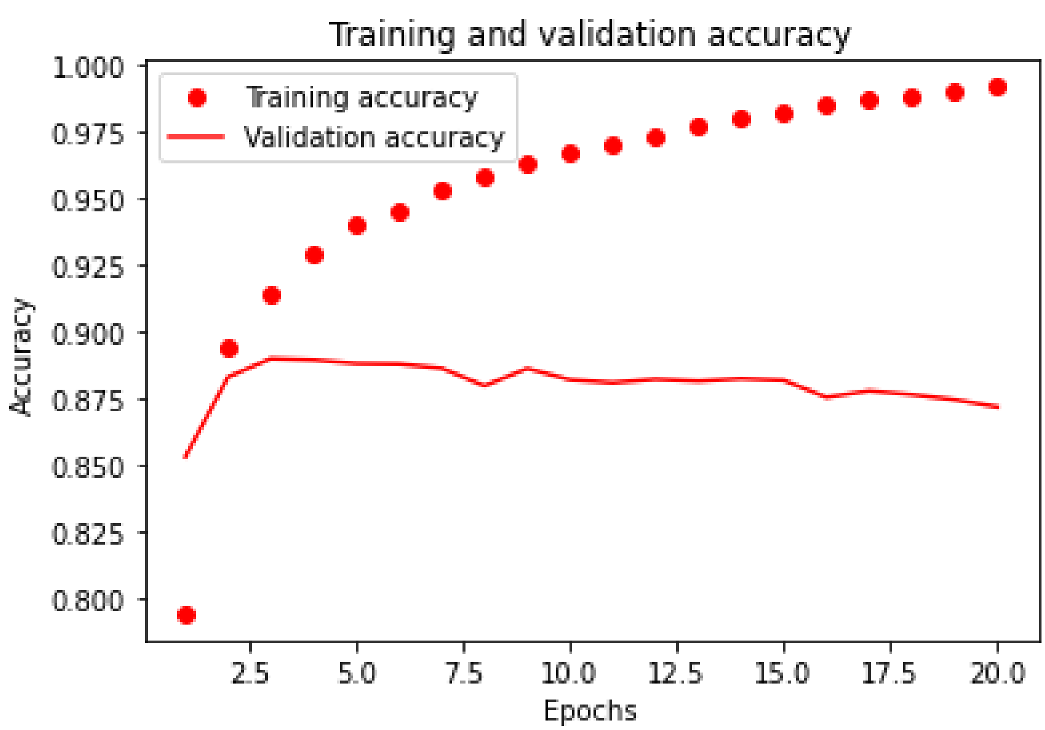
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[ ]:
```
import matplotlib.pyplot as plt
history_dict = history1.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
#Plotting graph between Training and Validation loss
plt.plot(epochs, loss_values, "ro", label="Training loss")
plt.plot(epochs, val_loss_values, "r", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

#Plotting graph between Training and Validation Accuracy
plt.clf()
acc = history_dict["accuracy"]
val_acc = history_dict["val_accuracy"]
plt.plot(epochs, acc, "ro", label="Training accuracy")
plt.plot(epochs, val_acc, "r", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
```

```
plt.legend()
plt.show()
```





[ ]:
```
np.random.seed(123)
model1 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

model1.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model1.fit(x_train, y_train, epochs=5, batch_size=512)
results1 = model1.evaluate(x_test, y_test)
```

```
Epoch 1/5
49/49 [==============================] - 1s 11ms/step - loss: 0.4636 - accuracy:
0.8196
Epoch 2/5
49/49 [==============================] - 1s 11ms/step - loss: 0.2947 - accuracy:
0.9029
Epoch 3/5
49/49 [==============================] - 1s 11ms/step - loss: 0.2384 - accuracy:
0.9177
Epoch 4/5
49/49 [==============================] - 1s 11ms/step - loss: 0.2060 - accuracy:
0.9287
Epoch 5/5
49/49 [==============================] - 1s 10ms/step - loss: 0.1854 - accuracy:
0.9355
782/782 [==============================] - 2s 2ms/step - loss: 0.2787 -
accuracy: 0.8882
```

[ ]:
```
results1
```

[ ]:
```
[0.27873992919921875, 0.8882399797439575]
```

The loss on the test set is 0.2787, and the accuracy is 88.82%.

[ ]:
```
model1.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.24893306],
       [0.99891126],
       [0.78139806],
       …,
       [0.12196511],
```

```
       [0.09458669],
       [0.56346315]], dtype=float32)
```

Building a neural network with 3 hidden layers

```
[ ]: np.random.seed(123)
model_3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_3.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

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

history3 = model_3.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.5542 - accuracy:
0.7633 - val_loss: 0.4196 - val_accuracy: 0.8527
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3277 - accuracy:
0.8941 - val_loss: 0.3379 - val_accuracy: 0.8627
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2379 - accuracy:
0.9177 - val_loss: 0.2820 - val_accuracy: 0.8872
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1869 - accuracy:
0.9362 - val_loss: 0.2919 - val_accuracy: 0.8835
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1558 - accuracy:
0.9467 - val_loss: 0.2881 - val_accuracy: 0.8862
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1294 - accuracy:
0.9573 - val_loss: 0.2966 - val_accuracy: 0.8834
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1143 - accuracy:
0.9617 - val_loss: 0.3393 - val_accuracy: 0.8748
```

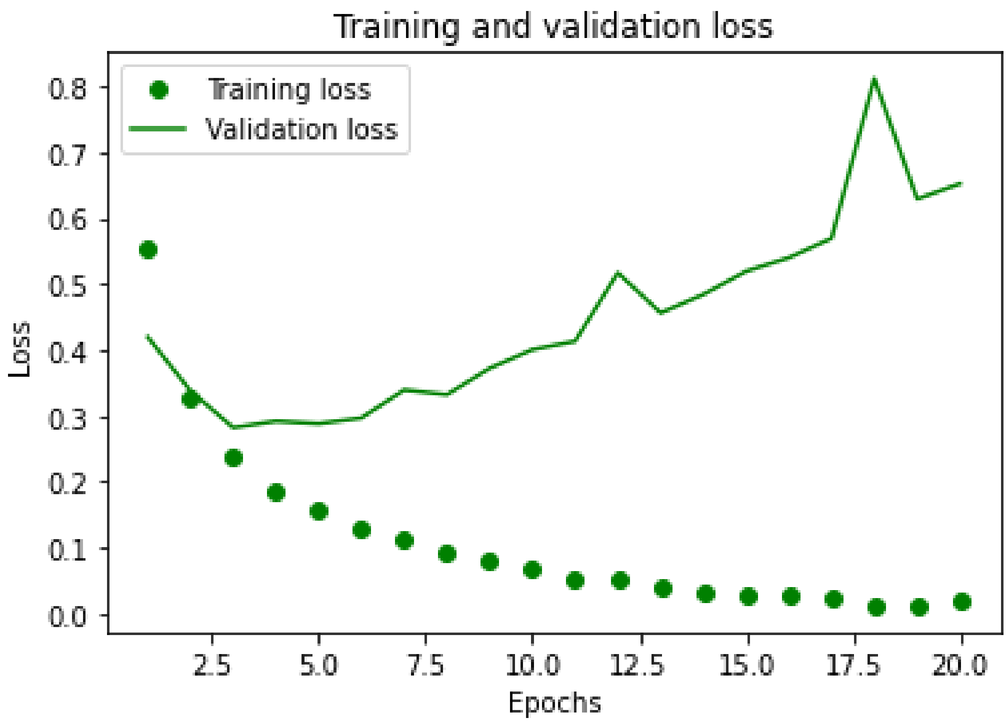
```
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0928 - accuracy:
0.9701 - val_loss: 0.3321 - val_accuracy: 0.8822
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0798 - accuracy:
0.9761 - val_loss: 0.3722 - val_accuracy: 0.8771
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0671 - accuracy:
0.9813 - val_loss: 0.4005 - val_accuracy: 0.8692
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0524 - accuracy:
0.9871 - val_loss: 0.4132 - val_accuracy: 0.8737
Epoch 12/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0517 - accuracy:
0.9847 - val_loss: 0.5173 - val_accuracy: 0.8535
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0417 - accuracy:
0.9887 - val_loss: 0.4562 - val_accuracy: 0.8745
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0335 - accuracy:
0.9917 - val_loss: 0.4843 - val_accuracy: 0.8731
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0270 - accuracy:
0.9937 - val_loss: 0.5195 - val_accuracy: 0.8705
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0261 - accuracy:
0.9939 - val_loss: 0.5398 - val_accuracy: 0.8711
Epoch 17/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0247 - accuracy:
0.9933 - val_loss: 0.5692 - val_accuracy: 0.8692
Epoch 18/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0118 - accuracy:
0.9987 - val_loss: 0.8118 - val_accuracy: 0.8348
Epoch 19/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0121 - accuracy:
0.9979 - val_loss: 0.6290 - val_accuracy: 0.8693
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0189 - accuracy:
0.9949 - val_loss: 0.6521 - val_accuracy: 0.8676
```

[ ]:
```
history_dict3 = history3.history
history_dict3.keys()
```

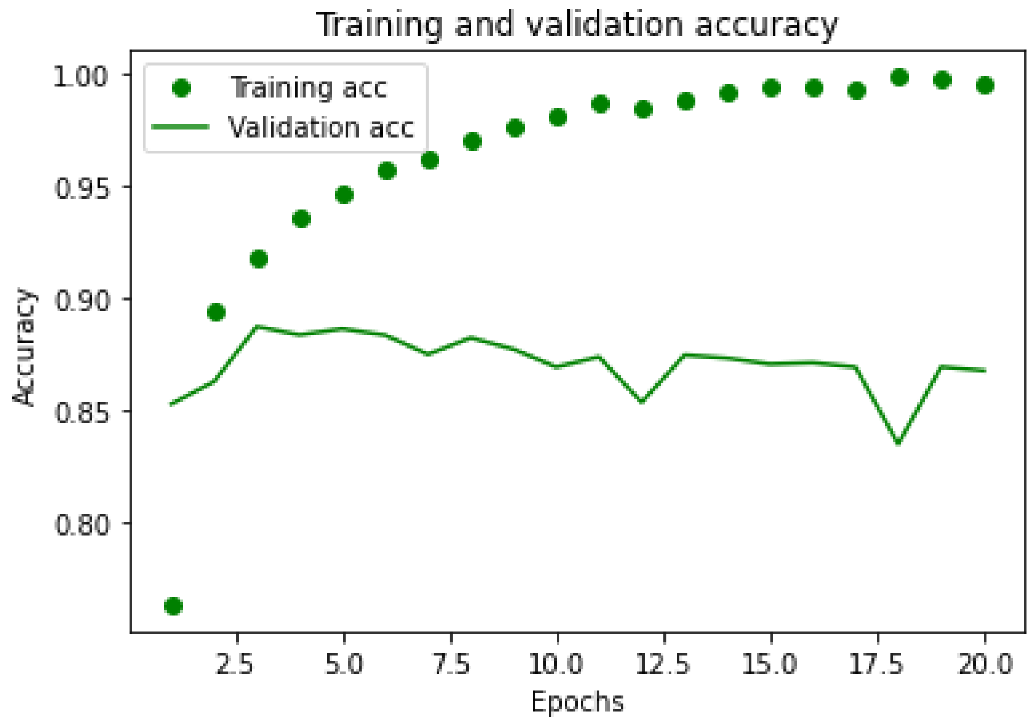
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[ ]:
```
loss_values = history_dict3["loss"]
val_loss_values = history_dict3["val_loss"]
```

```python
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()
```



[ ]:
```python
plt.clf()
acc = history_dict3["accuracy"]
val_acc = history_dict3["val_accuracy"]
plt.plot(epochs, acc, "go", label="Training acc")
plt.plot(epochs, val_acc, "g", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
np.random.seed(123)
model_3 = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])


model_3.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model_3.fit(x_train, y_train, epochs=3, batch_size=512)
results_3 = model_3.evaluate(x_test, y_test)
```

```
Epoch 1/3
49/49 [==============================] - 2s 11ms/step - loss: 0.4895 - accuracy:
0.7990
Epoch 2/3
49/49 [==============================] - 1s 11ms/step - loss: 0.2729 - accuracy:
0.9022
Epoch 3/3
49/49 [==============================] - 1s 11ms/step - loss: 0.2141 - accuracy:
```

```
0.9208
782/782 [==============================] - 2s 2ms/step - loss: 0.2839 -
accuracy: 0.8866
```

The loss on the test set is 0.2839, and the accuracy is 88.66%.

[ ]:
```
results_3
```

[ ]:
```
[0.2839148938655853, 0.8866000175476074]
```

[ ]:
```
model_3.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.26020768],
       [0.9987081 ],
       [0.7886101 ],
       …,
       [0.1106073 ],
       [0.0808426 ],
       [0.5625424 ]], dtype=float32)
```

Changing the number of layers does not lead to a significant increase in the model's accuracy; however, the model with three layers demonstrates higher accuracy compared to the other two configurations.

When determining the overall architecture of a neural network, it is essential to decide on the number of units in the hidden layers. Although these layers do not directly interact with the external environment, they play a crucial role in influencing the final outcomes.

Building Neural Network with 32 units.

[ ]:
```
np.random.seed(123)
model_32 = keras.Sequential([
    layers.Dense(32, activation="relu"),
    layers.Dense(32, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
#model compilation
model_32.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
#model validation
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

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

np.random.seed(123)
```

```
history32 = model_32.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

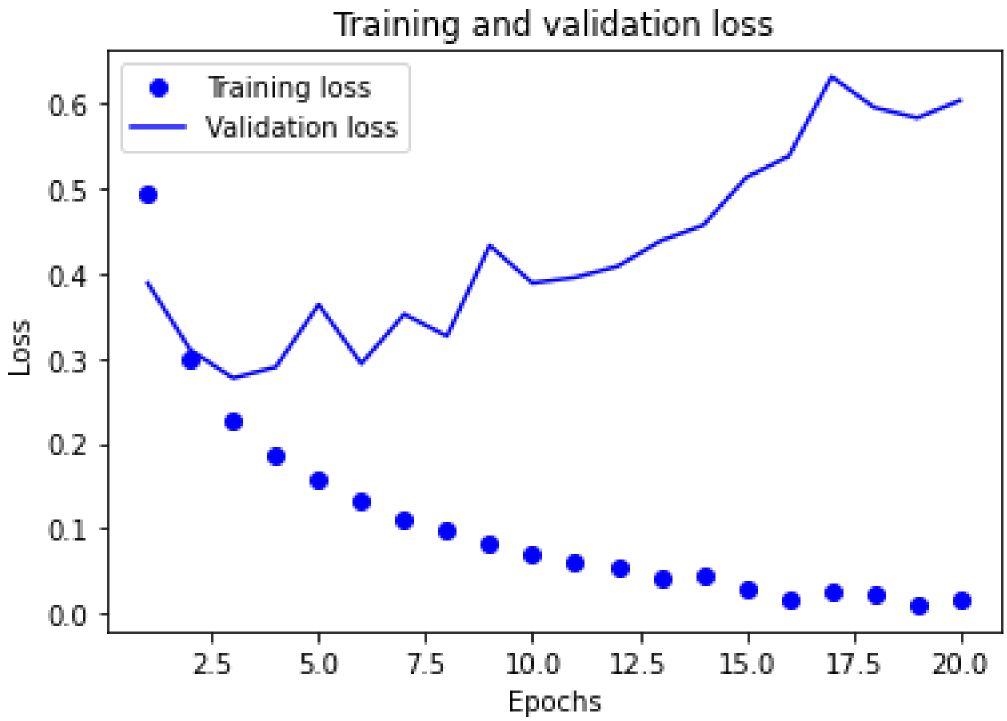
```
Epoch 1/20
30/30 [==============================] - 2s 47ms/step - loss: 0.4958 - accuracy:
0.7890 - val_loss: 0.3888 - val_accuracy: 0.8463
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2993 - accuracy:
0.8946 - val_loss: 0.3098 - val_accuracy: 0.8770
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2268 - accuracy:
0.9203 - val_loss: 0.2769 - val_accuracy: 0.8898
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1850 - accuracy:
0.9354 - val_loss: 0.2900 - val_accuracy: 0.8855
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1573 - accuracy:
0.9442 - val_loss: 0.3635 - val_accuracy: 0.8583
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1320 - accuracy:
0.9555 - val_loss: 0.2938 - val_accuracy: 0.8840
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1104 - accuracy:
0.9651 - val_loss: 0.3522 - val_accuracy: 0.8753
Epoch 8/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0981 - accuracy:
0.9691 - val_loss: 0.3263 - val_accuracy: 0.8808
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0822 - accuracy:
0.9749 - val_loss: 0.4331 - val_accuracy: 0.8573
Epoch 10/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0705 - accuracy:
0.9789 - val_loss: 0.3890 - val_accuracy: 0.8709
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0607 - accuracy:
0.9823 - val_loss: 0.3955 - val_accuracy: 0.8775
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0529 - accuracy:
0.9840 - val_loss: 0.4088 - val_accuracy: 0.8776
Epoch 13/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0418 - accuracy:
0.9883 - val_loss: 0.4385 - val_accuracy: 0.8749
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0433 - accuracy:
```

```
0.9874 - val_loss: 0.4572 - val_accuracy: 0.8743
Epoch 15/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0290 - accuracy:
0.9935 - val_loss: 0.5133 - val_accuracy: 0.8718
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0175 - accuracy:
0.9985 - val_loss: 0.5386 - val_accuracy: 0.8655
Epoch 17/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0243 - accuracy:
0.9946 - val_loss: 0.6322 - val_accuracy: 0.8535
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0221 - accuracy:
0.9943 - val_loss: 0.5958 - val_accuracy: 0.8674
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0099 - accuracy:
0.9995 - val_loss: 0.5834 - val_accuracy: 0.8715
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0159 - accuracy:
0.9961 - val_loss: 0.6040 - val_accuracy: 0.8714
```

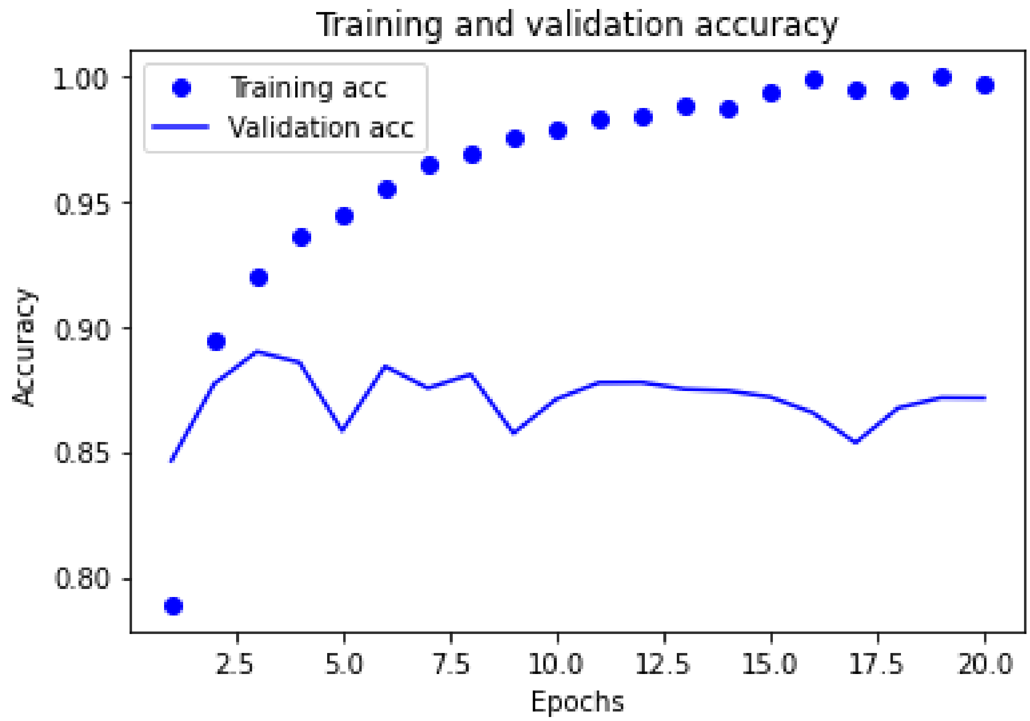
[ ]:
```python
history_dict32 = history32.history
history_dict32.keys()
```

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

[ ]:
```python
loss_values = history_dict32["loss"]
val_loss_values = history_dict32["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```
plt.clf()
acc = history_dict32["accuracy"]
val_acc = history_dict32["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
history_32 = model_32.fit(x_train, y_train, epochs=3, batch_size=512)
results_32 = model_32.evaluate(x_test, y_test)
results_32
```

```
Epoch 1/3
49/49 [==============================] - 1s 11ms/step - loss: 0.1957 - accuracy:
0.9456
Epoch 2/3
49/49 [==============================] - 1s 11ms/step - loss: 0.1225 - accuracy:
0.9625
Epoch 3/3
49/49 [==============================] - 1s 11ms/step - loss: 0.0902 - accuracy:
0.9723
782/782 [==============================] - 2s 3ms/step - loss: 0.4155 -
accuracy: 0.8649
```

[ ]:
```
[0.41551604866981506, 0.8648800253868103]
```

[ ]:
```
model_32.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.02048531],
       [0.9999927 ],
```

```
        [0.08758123],
        …,
        [0.03793093],
        [0.02606053],
        [0.78134537]], dtype=float32)
```

The accuracy on the validation set is 86.48

Traing the model with 64 units

```
[ ]: np.random.seed(123)
model_64 = keras.Sequential([
    layers.Dense(64, activation="relu"),
    layers.Dense(64, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_64.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
# validation
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

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

np.random.seed(123)
history64 = model_64.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

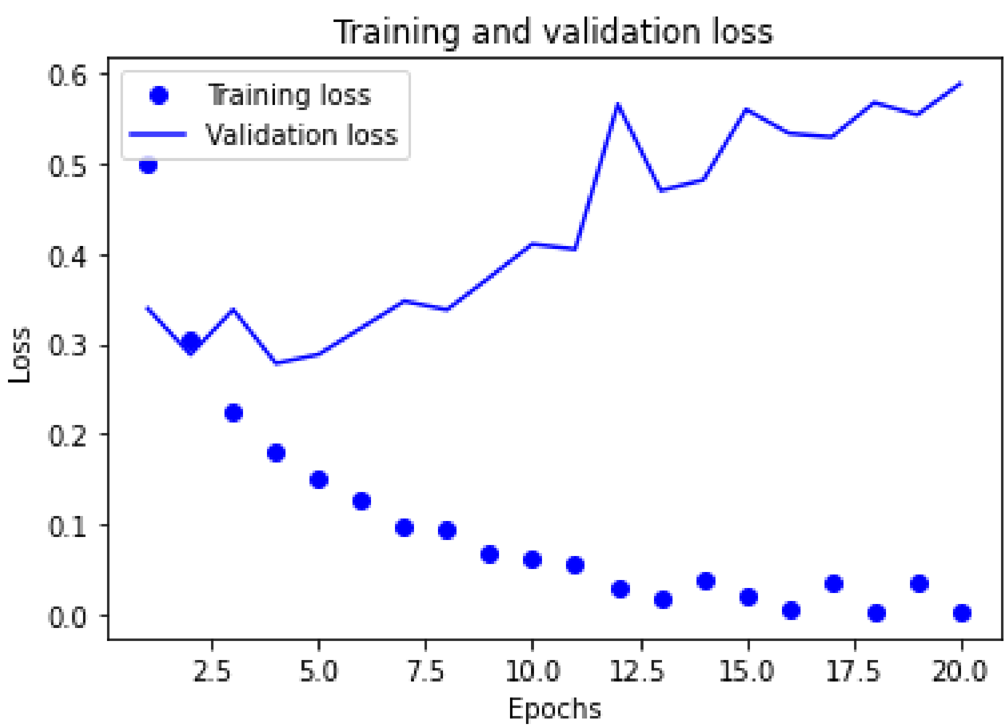
```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.4990 - accuracy:
0.7729 - val_loss: 0.3392 - val_accuracy: 0.8687
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3043 - accuracy:
0.8841 - val_loss: 0.2880 - val_accuracy: 0.8858
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2248 - accuracy:
0.9147 - val_loss: 0.3379 - val_accuracy: 0.8602
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1793 - accuracy:
0.9330 - val_loss: 0.2782 - val_accuracy: 0.8870
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1502 - accuracy:
0.9438 - val_loss: 0.2882 - val_accuracy: 0.8868
```

```
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1259 - accuracy:
0.9547 - val_loss: 0.3176 - val_accuracy: 0.8832
Epoch 7/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0964 - accuracy:
0.9670 - val_loss: 0.3473 - val_accuracy: 0.8810
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0951 - accuracy:
0.9665 - val_loss: 0.3377 - val_accuracy: 0.8813
Epoch 9/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0671 - accuracy:
0.9785 - val_loss: 0.3737 - val_accuracy: 0.8810
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0605 - accuracy:
0.9831 - val_loss: 0.4105 - val_accuracy: 0.8779
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0567 - accuracy:
0.9832 - val_loss: 0.4046 - val_accuracy: 0.8762
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0283 - accuracy:
0.9933 - val_loss: 0.5657 - val_accuracy: 0.8483
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0167 - accuracy:
0.9979 - val_loss: 0.4699 - val_accuracy: 0.8769
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0378 - accuracy:
0.9882 - val_loss: 0.4817 - val_accuracy: 0.8773
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0215 - accuracy:
0.9943 - val_loss: 0.5598 - val_accuracy: 0.8620
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0071 - accuracy:
0.9997 - val_loss: 0.5334 - val_accuracy: 0.8768
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0344 - accuracy:
0.9894 - val_loss: 0.5291 - val_accuracy: 0.8758
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0042 - accuracy:
0.9998 - val_loss: 0.5674 - val_accuracy: 0.8763
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0354 - accuracy:
0.9893 - val_loss: 0.5539 - val_accuracy: 0.8759
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0030 - accuracy:
0.9999 - val_loss: 0.5879 - val_accuracy: 0.8767
```

[ ]:
```
history_dict64 = history64.history
history_dict64.keys()
```

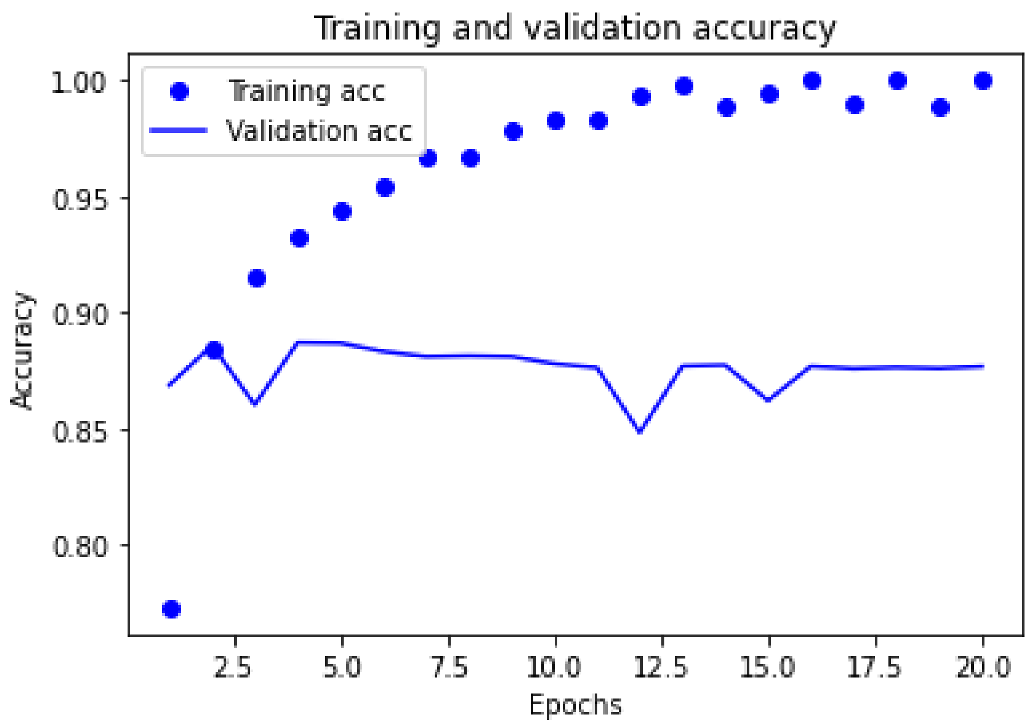
[ ]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

[ ]:
```
loss_values = history_dict64["loss"]
val_loss_values = history_dict64["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```
plt.clf()
acc = history_dict64["accuracy"]
val_acc = history_dict64["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
```

```
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
history_64 = model_64.fit(x_train, y_train, epochs=3, batch_size=512)
results_64 = model_64.evaluate(x_test, y_test)
results_64
```

```
Epoch 1/3
49/49 [==============================] - 1s 11ms/step - loss: 0.1760 - accuracy:
0.9479
Epoch 2/3
49/49 [==============================] - 1s 11ms/step - loss: 0.0966 - accuracy:
0.9698
Epoch 3/3
49/49 [==============================] - 1s 10ms/step - loss: 0.0604 - accuracy:
0.9832
782/782 [==============================] - 2s 2ms/step - loss: 0.4104 -
accuracy: 0.8676
```

[ ]: [0.4103541672229767, 0.8675600290298462]

[ ]:
```
model_64.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.01911188],
       [0.9999995 ],
       [0.6092722 ],
       …,
       [0.02825702],
       [0.02160722],
       [0.8528232 ]], dtype=float32)
```

The accuracy on the validation set is 86.75%

Training the model with 128 units

[ ]:
```python
np.random.seed(123)
model_128 = keras.Sequential([
    layers.Dense(128, activation="relu"),
    layers.Dense(128, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
model_128.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
# validation
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

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

np.random.seed(123)
history128 = model_128.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

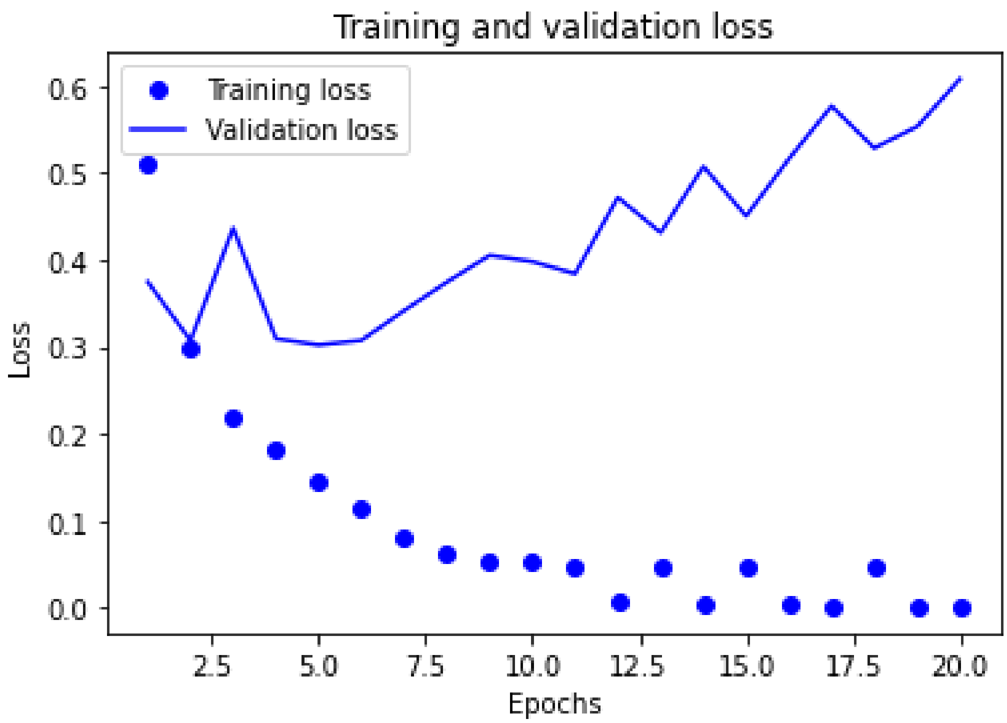
```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.5092 - accuracy:
0.7605 - val_loss: 0.3750 - val_accuracy: 0.8388
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2999 - accuracy:
0.8839 - val_loss: 0.3073 - val_accuracy: 0.8725
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2183 - accuracy:
0.9188 - val_loss: 0.4367 - val_accuracy: 0.8213
Epoch 4/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1810 - accuracy:
0.9280 - val_loss: 0.3095 - val_accuracy: 0.8743
```

```
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1443 - accuracy:
0.9465 - val_loss: 0.3023 - val_accuracy: 0.8830
Epoch 6/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1156 - accuracy:
0.9588 - val_loss: 0.3075 - val_accuracy: 0.8837
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0795 - accuracy:
0.9741 - val_loss: 0.3419 - val_accuracy: 0.8792
Epoch 8/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0625 - accuracy:
0.9813 - val_loss: 0.3747 - val_accuracy: 0.8792
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0530 - accuracy:
0.9843 - val_loss: 0.4053 - val_accuracy: 0.8758
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0535 - accuracy:
0.9845 - val_loss: 0.3981 - val_accuracy: 0.8784
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0462 - accuracy:
0.9862 - val_loss: 0.3843 - val_accuracy: 0.8782
Epoch 12/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0084 - accuracy:
0.9997 - val_loss: 0.4720 - val_accuracy: 0.8788
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0466 - accuracy:
0.9875 - val_loss: 0.4314 - val_accuracy: 0.8793
Epoch 14/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0044 - accuracy:
0.9999 - val_loss: 0.5075 - val_accuracy: 0.8779
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0480 - accuracy:
0.9859 - val_loss: 0.4505 - val_accuracy: 0.8789
Epoch 16/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0035 - accuracy:
1.0000 - val_loss: 0.5163 - val_accuracy: 0.8794
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0018 - accuracy:
1.0000 - val_loss: 0.5771 - val_accuracy: 0.8770
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0472 - accuracy:
0.9878 - val_loss: 0.5288 - val_accuracy: 0.8759
Epoch 19/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0022 - accuracy:
1.0000 - val_loss: 0.5537 - val_accuracy: 0.8779
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0011 - accuracy:
1.0000 - val_loss: 0.6081 - val_accuracy: 0.8775
```

[ ]:
```
history_dict128 = history128.history
history_dict128.keys()
```

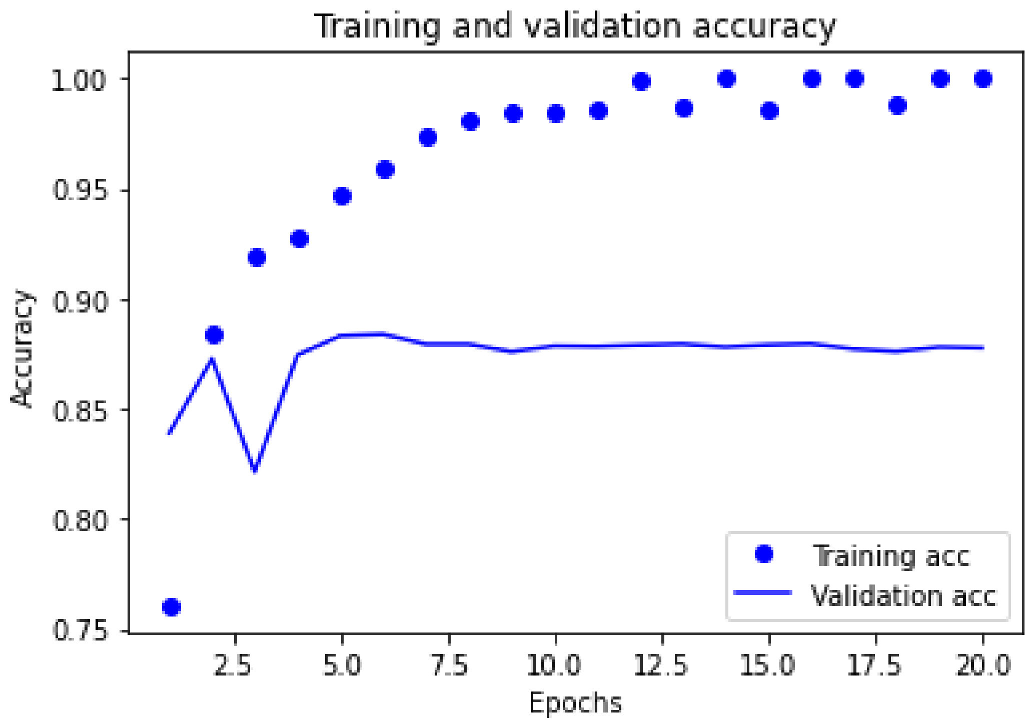
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[ ]:
```
loss_values = history_dict128["loss"]
val_loss_values = history_dict128["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```
plt.clf()
acc = history_dict128["accuracy"]
val_acc = history_dict128["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
```

```
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
history_128 = model_128.fit(x_train, y_train, epochs=2, batch_size=512)
results_128 = model_128.evaluate(x_test, y_test)
results_128
```

```
Epoch 1/2
49/49 [==============================] - 1s 12ms/step - loss: 0.1713 - accuracy:
0.9470
Epoch 2/2
49/49 [==============================] - 1s 12ms/step - loss: 0.0857 - accuracy:
0.9730
782/782 [==============================] - 2s 2ms/step - loss: 0.3647 -
accuracy: 0.8738
```

[ ]: [0.3647419810295105, 0.8738399744033813]

[ ]:
```
model_128.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]: array([[0.0530677],
        [0.9999995],
        [0.9354145],
        …,
        [0.02437645],
        [0.00841208],
        [0.9205662]], dtype=float32)

The accuracy on the validation set is 87.38%

MSE Loss Function

[ ]:
```
np.random.seed(123)
model_MSE = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
#Model compilation
model_MSE.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
# validation
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
# Model Fit
np.random.seed(123)
history_model_MSE = model_MSE.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

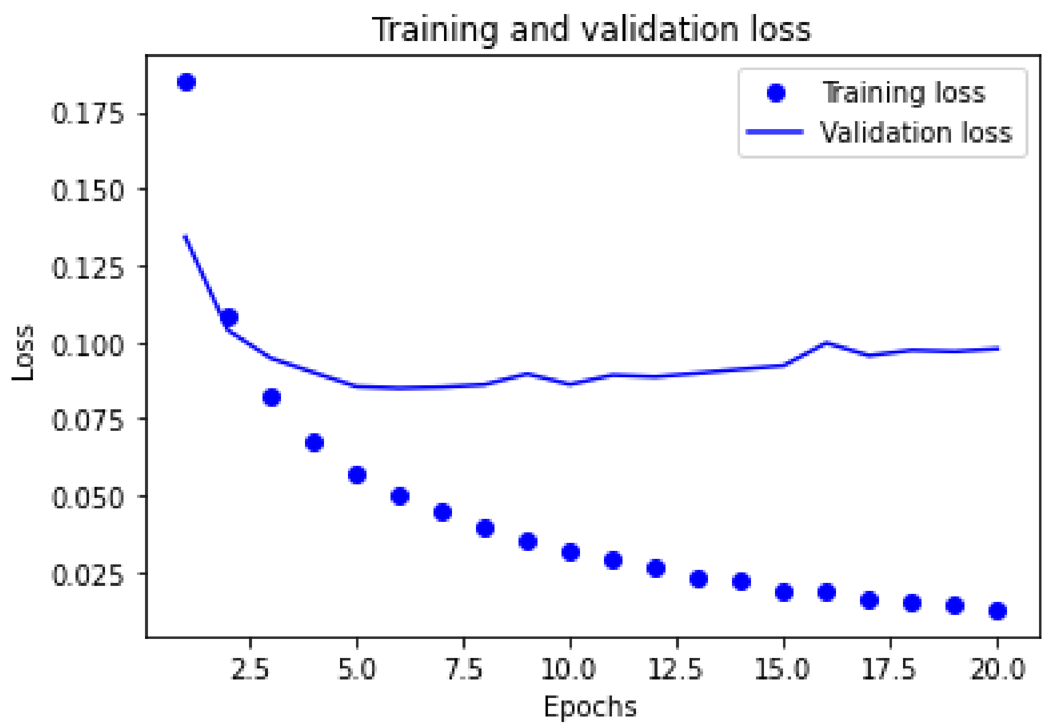
```
Epoch 1/20
30/30 [==============================] - 3s 48ms/step - loss: 0.1849 - accuracy:
0.7725 - val_loss: 0.1343 - val_accuracy: 0.8569
Epoch 2/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1088 - accuracy:
0.8837 - val_loss: 0.1039 - val_accuracy: 0.8750
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0827 - accuracy:
0.9077 - val_loss: 0.0948 - val_accuracy: 0.8783
Epoch 4/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0680 - accuracy:
0.9240 - val_loss: 0.0901 - val_accuracy: 0.8818
```

```
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0573 - accuracy:
0.9373 - val_loss: 0.0855 - val_accuracy: 0.8860
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0501 - accuracy:
0.9460 - val_loss: 0.0850 - val_accuracy: 0.8839
Epoch 7/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0448 - accuracy:
0.9544 - val_loss: 0.0853 - val_accuracy: 0.8840
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0398 - accuracy:
0.9595 - val_loss: 0.0861 - val_accuracy: 0.8810
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0357 - accuracy:
0.9634 - val_loss: 0.0896 - val_accuracy: 0.8801
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0321 - accuracy:
0.9687 - val_loss: 0.0862 - val_accuracy: 0.8805
Epoch 11/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0293 - accuracy:
0.9733 - val_loss: 0.0893 - val_accuracy: 0.8794
Epoch 12/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0264 - accuracy:
0.9750 - val_loss: 0.0887 - val_accuracy: 0.8783
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0228 - accuracy:
0.9805 - val_loss: 0.0899 - val_accuracy: 0.8793
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0226 - accuracy:
0.9797 - val_loss: 0.0912 - val_accuracy: 0.8772
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0189 - accuracy:
0.9845 - val_loss: 0.0924 - val_accuracy: 0.8759
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0191 - accuracy:
0.9843 - val_loss: 0.0998 - val_accuracy: 0.8679
Epoch 17/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0165 - accuracy:
0.9870 - val_loss: 0.0956 - val_accuracy: 0.8758
Epoch 18/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0157 - accuracy:
0.9877 - val_loss: 0.0974 - val_accuracy: 0.8722
Epoch 19/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0141 - accuracy:
0.9889 - val_loss: 0.0969 - val_accuracy: 0.8761
Epoch 20/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0128 - accuracy:
0.9899 - val_loss: 0.0978 - val_accuracy: 0.8756
```

[ ]:
```
history_dict_MSE = history_model_MSE.history
history_dict_MSE.keys()
```

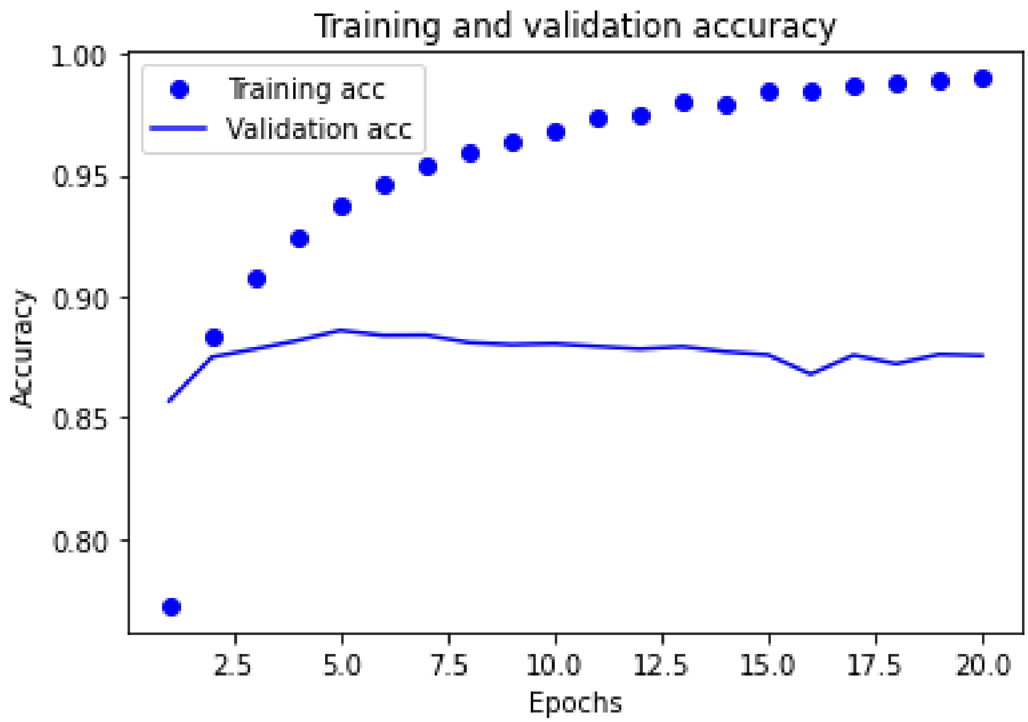
[ ]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

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



[ ]:
```
plt.clf()
acc = history_dict_MSE["accuracy"]
val_acc = history_dict_MSE["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
```

```
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
model_MSE.fit(x_train, y_train, epochs=8, batch_size=512)
results_MSE = model_MSE.evaluate(x_test, y_test)
results_MSE
```

```
Epoch 1/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0468 - accuracy:
0.9443
Epoch 2/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0392 - accuracy:
0.9548
Epoch 3/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0344 - accuracy:
0.9614
Epoch 4/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0309 - accuracy:
0.9670
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0286 - accuracy:
0.9706
```

```
Epoch 6/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0276 - accuracy:
0.9712
Epoch 7/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0243 - accuracy:
0.9761
Epoch 8/8
49/49 [==============================] - 1s 10ms/step - loss: 0.0237 - accuracy:
0.9759
782/782 [==============================] - 2s 2ms/step - loss: 0.1102 -
accuracy: 0.8645
```

[ ]:
```
[0.11019179970026016, 0.8644800186157227]
```

[ ]:
```
model_MSE.predict(x_test)
```

```
782/782 [==============================] - 1s 2ms/step
```

[ ]:
```
array([[0.0129396 ],
       [0.99995804],
       [0.34060687],
       …,
       [0.03023529],
       [0.01194245],
       [0.8410266 ]], dtype=float32)
```

Tanh Activation Function

[ ]:
```
np.random.seed(123)
model_tanh = keras.Sequential([
    layers.Dense(16, activation="tanh"),
    layers.Dense(16, activation="tanh"),
    layers.Dense(1, activation="sigmoid")
])

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

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

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

np.random.seed(123)

history_tanh = model_tanh.fit(partial_x_train,
```

```
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

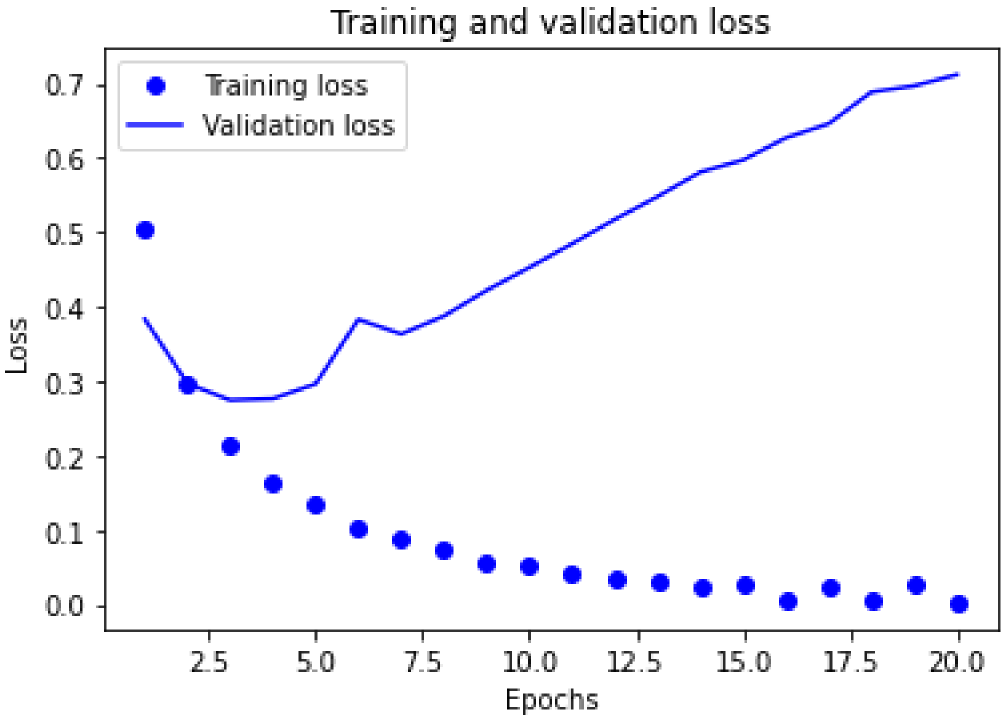
```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.5031 - accuracy:
0.7875 - val_loss: 0.3834 - val_accuracy: 0.8543
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2959 - accuracy:
0.8966 - val_loss: 0.2970 - val_accuracy: 0.8830
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2146 - accuracy:
0.9241 - val_loss: 0.2746 - val_accuracy: 0.8887
Epoch 4/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1648 - accuracy:
0.9441 - val_loss: 0.2766 - val_accuracy: 0.8851
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1339 - accuracy:
0.9529 - val_loss: 0.2966 - val_accuracy: 0.8866
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1011 - accuracy:
0.9682 - val_loss: 0.3830 - val_accuracy: 0.8683
Epoch 7/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0890 - accuracy:
0.9707 - val_loss: 0.3634 - val_accuracy: 0.8788
Epoch 8/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0738 - accuracy:
0.9757 - val_loss: 0.3876 - val_accuracy: 0.8780
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0573 - accuracy:
0.9812 - val_loss: 0.4222 - val_accuracy: 0.8739
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0528 - accuracy:
0.9838 - val_loss: 0.4530 - val_accuracy: 0.8740
Epoch 11/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0419 - accuracy:
0.9882 - val_loss: 0.4847 - val_accuracy: 0.8710
Epoch 12/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0329 - accuracy:
0.9902 - val_loss: 0.5174 - val_accuracy: 0.8694
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0307 - accuracy:
0.9915 - val_loss: 0.5483 - val_accuracy: 0.8696
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0248 - accuracy:
0.9932 - val_loss: 0.5812 - val_accuracy: 0.8649
```

```
Epoch 15/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0265 - accuracy:
0.9927 - val_loss: 0.5974 - val_accuracy: 0.8659
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0071 - accuracy:
0.9995 - val_loss: 0.6270 - val_accuracy: 0.8659
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0253 - accuracy:
0.9928 - val_loss: 0.6459 - val_accuracy: 0.8662
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0041 - accuracy:
0.9997 - val_loss: 0.6886 - val_accuracy: 0.8600
Epoch 19/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0289 - accuracy:
0.9917 - val_loss: 0.6969 - val_accuracy: 0.8652
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0026 - accuracy:
0.9999 - val_loss: 0.7120 - val_accuracy: 0.8638
```

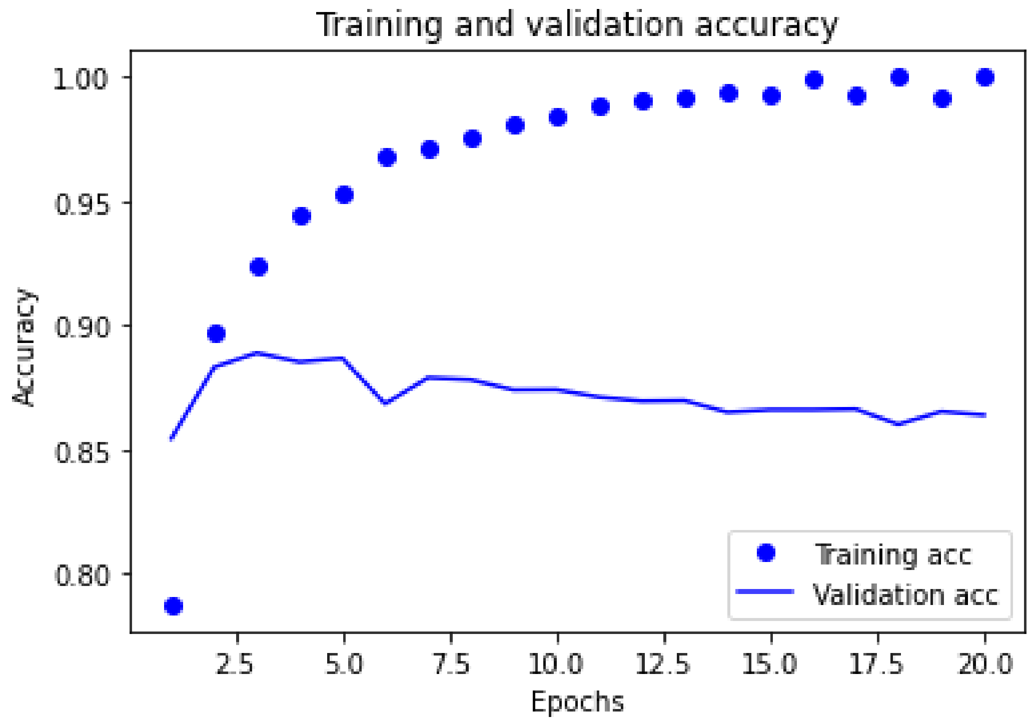
[ ]:
```
history_dict_tanh = history_tanh.history
history_dict_tanh.keys()
```

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

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



[ ]:
```
plt.clf()
acc = history_dict_tanh["accuracy"]
val_acc = history_dict_tanh["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```




```
[ ]: model_tanh.fit(x_train, y_train, epochs=8, batch_size=512)
results_tanh = model_tanh.evaluate(x_test, y_test)
results_tanh
```

```
Epoch 1/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2664 - accuracy:
0.9434
Epoch 2/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1440 - accuracy:
0.9584
Epoch 3/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1192 - accuracy:
0.9630
Epoch 4/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0992 - accuracy:
0.9679
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0887 - accuracy:
0.9712
Epoch 6/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0738 - accuracy:
0.9775
Epoch 7/8
49/49 [==============================] - 1s 10ms/step - loss: 0.0695 - accuracy:
```

```
0.9782
Epoch 8/8
49/49 [==============================] - 1s 10ms/step - loss: 0.0570 - accuracy:
0.9827
782/782 [==============================] - 2s 3ms/step - loss: 0.6204 -
accuracy: 0.8520
```

[ ]: [0.6204051375389099, 0.8520399928092957]

Adam Optimizer Function

[ ]:
```
np.random.seed(123)
model_adam = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])

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

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

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

np.random.seed(123)

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

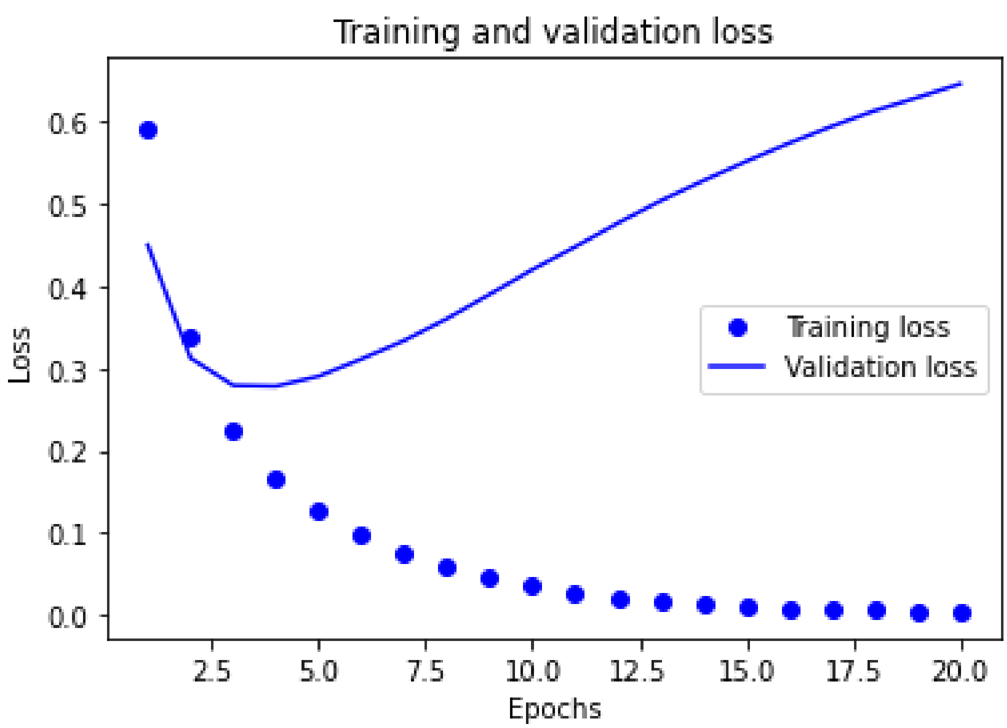
```
Epoch 1/20
30/30 [==============================] - 3s 48ms/step - loss: 0.5916 - accuracy:
0.7279 - val_loss: 0.4499 - val_accuracy: 0.8422
Epoch 2/20
30/30 [==============================] - 1s 17ms/step - loss: 0.3395 - accuracy:
0.8938 - val_loss: 0.3126 - val_accuracy: 0.8824
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2242 - accuracy:
0.9277 - val_loss: 0.2792 - val_accuracy: 0.8900
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1653 - accuracy:
0.9483 - val_loss: 0.2783 - val_accuracy: 0.8885
```

```
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1256 - accuracy:
0.9631 - val_loss: 0.2904 - val_accuracy: 0.8851
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0980 - accuracy:
0.9735 - val_loss: 0.3113 - val_accuracy: 0.8828
Epoch 7/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0764 - accuracy:
0.9832 - val_loss: 0.3339 - val_accuracy: 0.8789
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0587 - accuracy:
0.9884 - val_loss: 0.3605 - val_accuracy: 0.8791
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0462 - accuracy:
0.9927 - val_loss: 0.3901 - val_accuracy: 0.8745
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0359 - accuracy:
0.9952 - val_loss: 0.4200 - val_accuracy: 0.8748
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0279 - accuracy:
0.9968 - val_loss: 0.4477 - val_accuracy: 0.8741
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0212 - accuracy:
0.9987 - val_loss: 0.4767 - val_accuracy: 0.8710
Epoch 13/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0164 - accuracy:
0.9994 - val_loss: 0.5039 - val_accuracy: 0.8717
Epoch 14/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0128 - accuracy:
0.9997 - val_loss: 0.5284 - val_accuracy: 0.8703
Epoch 15/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0101 - accuracy:
0.9997 - val_loss: 0.5518 - val_accuracy: 0.8685
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0081 - accuracy:
0.9999 - val_loss: 0.5739 - val_accuracy: 0.8677
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0067 - accuracy:
0.9999 - val_loss: 0.5945 - val_accuracy: 0.8670
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0055 - accuracy:
0.9999 - val_loss: 0.6134 - val_accuracy: 0.8665
Epoch 19/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0046 - accuracy:
0.9999 - val_loss: 0.6295 - val_accuracy: 0.8659
Epoch 20/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0040 - accuracy:
0.9999 - val_loss: 0.6459 - val_accuracy: 0.8662
```

[ ]:
```
history_dict_adam = history_adam.history
history_dict_adam.keys()
```

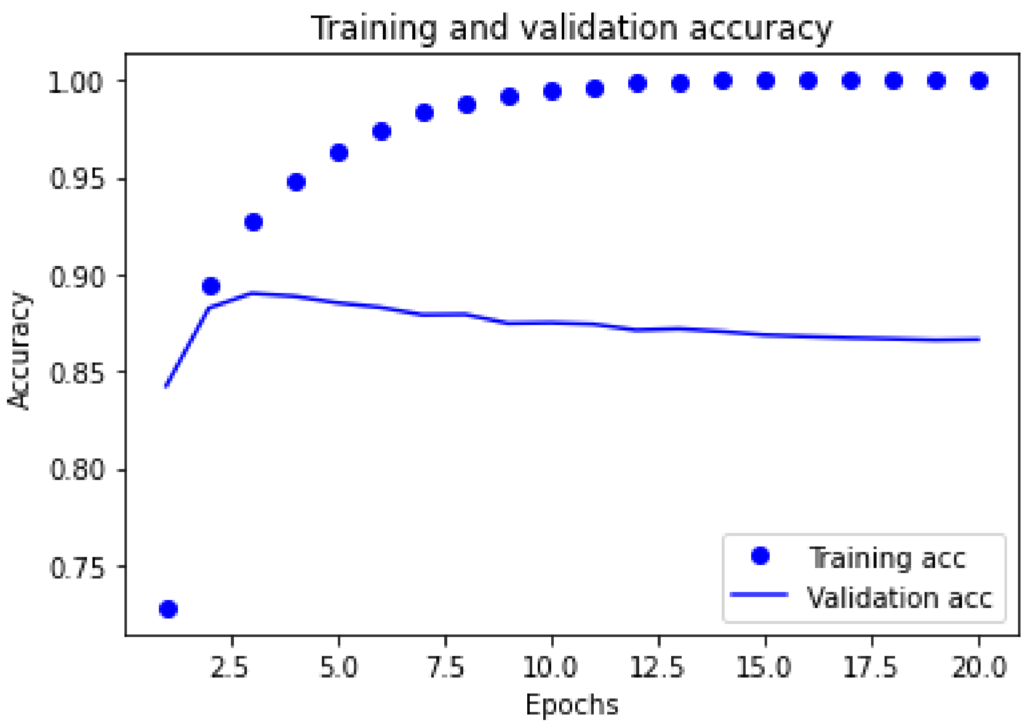
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

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



[ ]:
```
plt.clf()
acc = history_dict_adam["accuracy"]
val_acc = history_dict_adam["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
```

```
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
model_adam.fit(x_train, y_train, epochs=4, batch_size=512)
results_adam = model_adam.evaluate(x_test, y_test)
results_adam
```

```
Epoch 1/4
49/49 [==============================] - 1s 11ms/step - loss: 0.2282 - accuracy:
0.9388
Epoch 2/4
49/49 [==============================] - 1s 11ms/step - loss: 0.1090 - accuracy:
0.9665
Epoch 3/4
49/49 [==============================] - 1s 11ms/step - loss: 0.0719 - accuracy:
0.9793
Epoch 4/4
49/49 [==============================] - 1s 11ms/step - loss: 0.0520 - accuracy:
0.9878
782/782 [==============================] - 2s 3ms/step - loss: 0.4984 -
accuracy: 0.8578
```

[ ]:
```
[0.49841028451919556, 0.8578000068664551]
```

Regularization

[ ]:
```
from tensorflow.keras import regularizers
np.random.seed(123)
model_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")
])
model_regularization.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
np.random.seed(123)
history_model_regularization = model_regularization.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
history_dict_regularization = history_model_regularization.history
history_dict_regularization.keys()
```

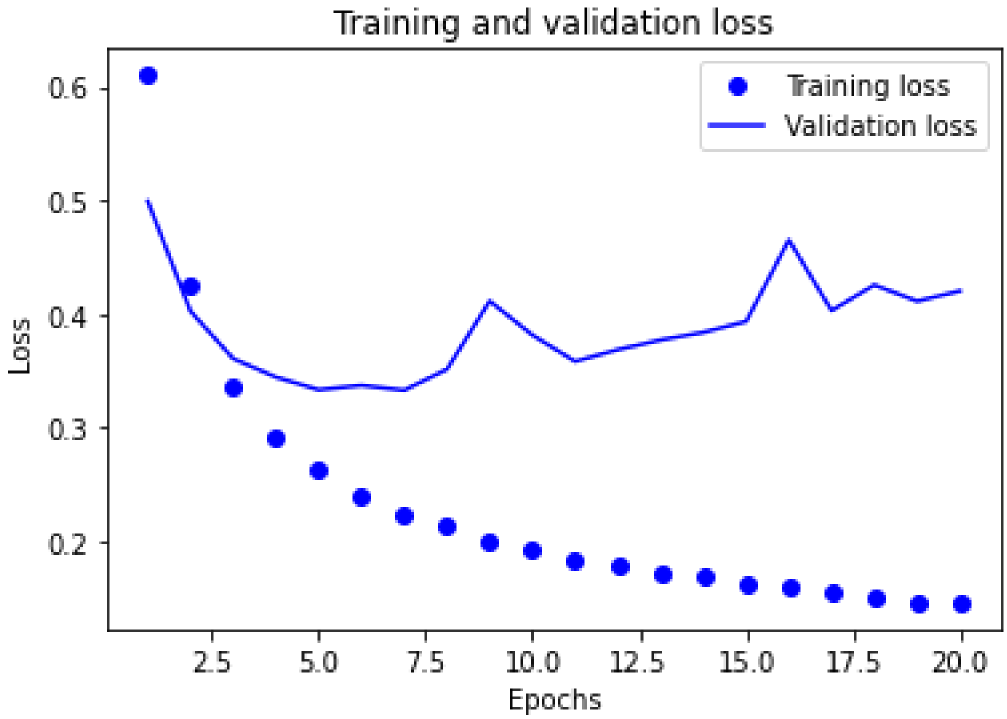
```
Epoch 1/20
30/30 [==============================] - 2s 48ms/step - loss: 0.6102 - accuracy:
0.7638 - val_loss: 0.4993 - val_accuracy: 0.8420
Epoch 2/20
30/30 [==============================] - 1s 19ms/step - loss: 0.4243 - accuracy:
0.8827 - val_loss: 0.4025 - val_accuracy: 0.8700
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3368 - accuracy:
0.9069 - val_loss: 0.3606 - val_accuracy: 0.8799
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2902 - accuracy:
0.9198 - val_loss: 0.3443 - val_accuracy: 0.8835
Epoch 5/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2621 - accuracy:
0.9314 - val_loss: 0.3332 - val_accuracy: 0.8873
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2404 - accuracy:
0.9393 - val_loss: 0.3371 - val_accuracy: 0.8856
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2232 - accuracy:
0.9483 - val_loss: 0.3328 - val_accuracy: 0.8848
Epoch 8/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2130 - accuracy:
0.9507 - val_loss: 0.3512 - val_accuracy: 0.8829
Epoch 9/20
```

```
30/30 [==============================] - 1s 17ms/step - loss: 0.2002 - accuracy:
0.9557 - val_loss: 0.4114 - val_accuracy: 0.8625
Epoch 10/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1932 - accuracy:
0.9584 - val_loss: 0.3814 - val_accuracy: 0.8712
Epoch 11/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1839 - accuracy:
0.9624 - val_loss: 0.3582 - val_accuracy: 0.8801
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1791 - accuracy:
0.9648 - val_loss: 0.3686 - val_accuracy: 0.8780
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1725 - accuracy:
0.9675 - val_loss: 0.3771 - val_accuracy: 0.8774
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1690 - accuracy:
0.9672 - val_loss: 0.3838 - val_accuracy: 0.8791
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 0.1617 - accuracy:
0.9721 - val_loss: 0.3934 - val_accuracy: 0.8780
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1602 - accuracy:
0.9713 - val_loss: 0.4651 - val_accuracy: 0.8551
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1543 - accuracy:
0.9745 - val_loss: 0.4030 - val_accuracy: 0.8763
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1508 - accuracy:
0.9757 - val_loss: 0.4256 - val_accuracy: 0.8660
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1460 - accuracy:
0.9781 - val_loss: 0.4113 - val_accuracy: 0.8744
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1459 - accuracy:
0.9773 - val_loss: 0.4203 - val_accuracy: 0.8726
```

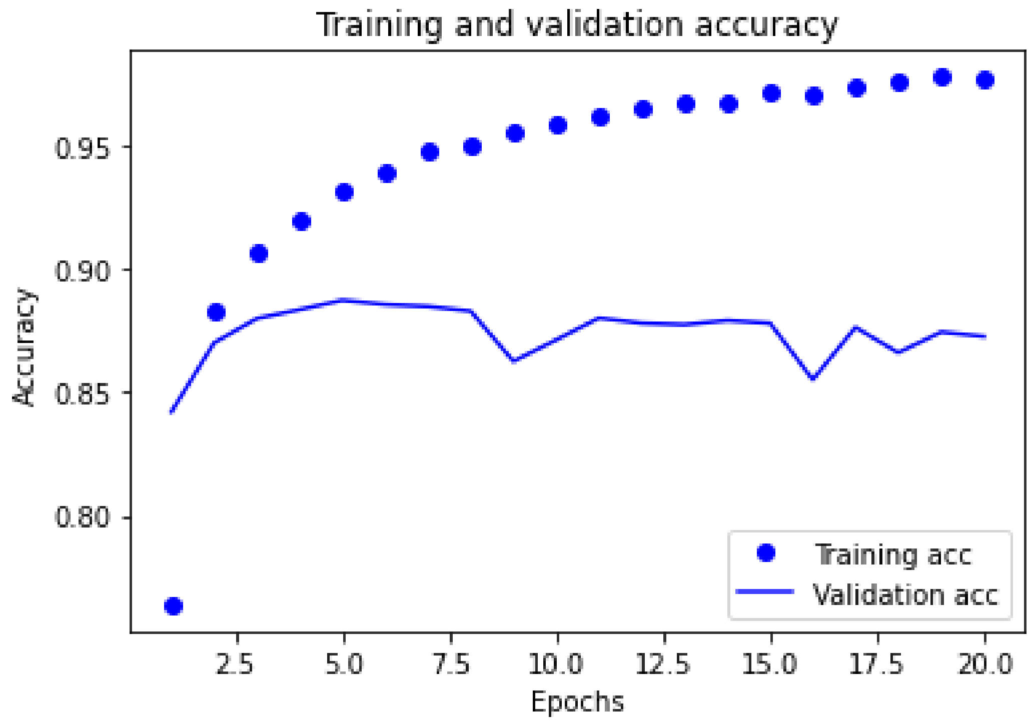
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[ ]:
```python
loss_values = history_dict_regularization["loss"]
val_loss_values = history_dict_regularization["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
```

```
plt.show()
```



[ ]:
```
plt.clf()
acc = history_dict_regularization["accuracy"]
val_acc = history_dict_regularization["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
model_regularization.fit(x_train, y_train, epochs=8, batch_size=512)
results_regularization = model_regularization.evaluate(x_test, y_test)
results_regularization
```

```
Epoch 1/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2496 - accuracy:
0.9352
Epoch 2/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2135 - accuracy:
0.9472
Epoch 3/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2030 - accuracy:
0.9478
Epoch 4/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1902 - accuracy:
0.9539
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1876 - accuracy:
0.9553
Epoch 6/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1848 - accuracy:
0.9558
Epoch 7/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1812 - accuracy:
```

```
0.9576
Epoch 8/8
49/49 [==============================] - 1s 11ms/step - loss: 0.1782 - accuracy:
0.9587
782/782 [==============================] - 2s 3ms/step - loss: 0.4255 -
accuracy: 0.8675
```

[ ]:
```
[0.42552879452705383, 0.8675199747085571]
```

The loss on test set is 0.4255 and accuracy is 86.75%.

Dropout

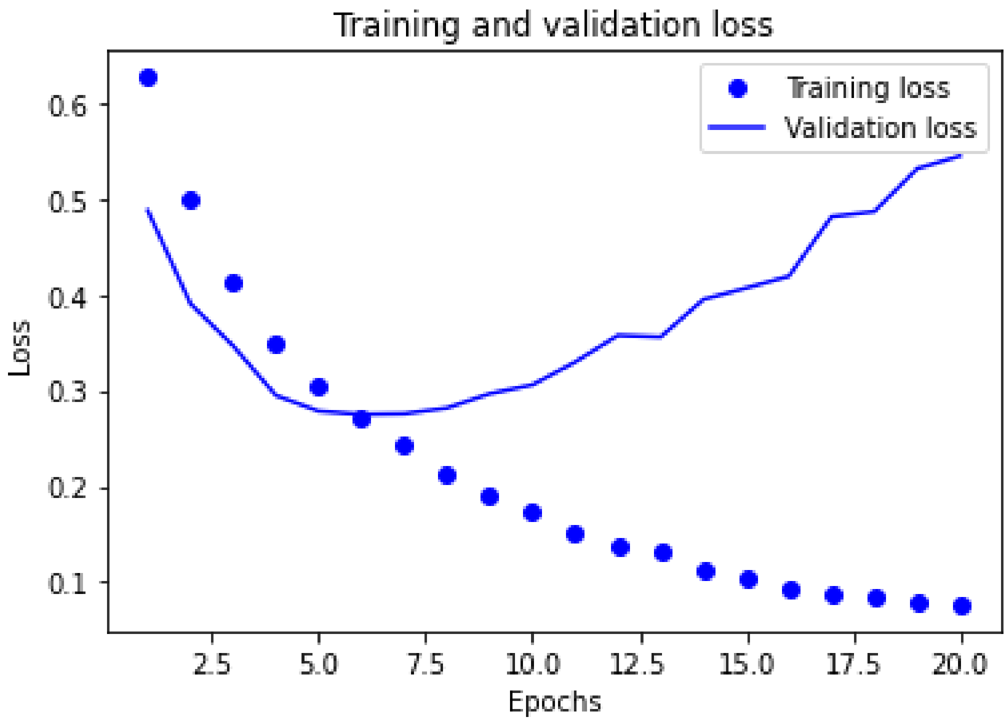
[ ]:
```
from tensorflow.keras import regularizers
np.random.seed(123)
model_Dropout = keras.Sequential([
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(16, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(1, activation="sigmoid")
])
model_Dropout.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
np.random.seed(123)
history_model_Dropout = model_Dropout.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
history_dict_Dropout = history_model_Dropout.history
history_dict_Dropout.keys()
```

```
Epoch 1/20
30/30 [==============================] - 2s 47ms/step - loss: 0.6270 - accuracy:
0.6458 - val_loss: 0.4878 - val_accuracy: 0.8423
Epoch 2/20
30/30 [==============================] - 1s 17ms/step - loss: 0.4985 - accuracy:
0.7700 - val_loss: 0.3906 - val_accuracy: 0.8727
Epoch 3/20
30/30 [==============================] - 1s 17ms/step - loss: 0.4121 - accuracy:
0.8235 - val_loss: 0.3465 - val_accuracy: 0.8778
Epoch 4/20
30/30 [==============================] - 1s 18ms/step - loss: 0.3497 - accuracy:
0.8602 - val_loss: 0.2947 - val_accuracy: 0.8860
Epoch 5/20
30/30 [==============================] - 1s 17ms/step - loss: 0.3056 - accuracy:
0.8867 - val_loss: 0.2784 - val_accuracy: 0.8888
```

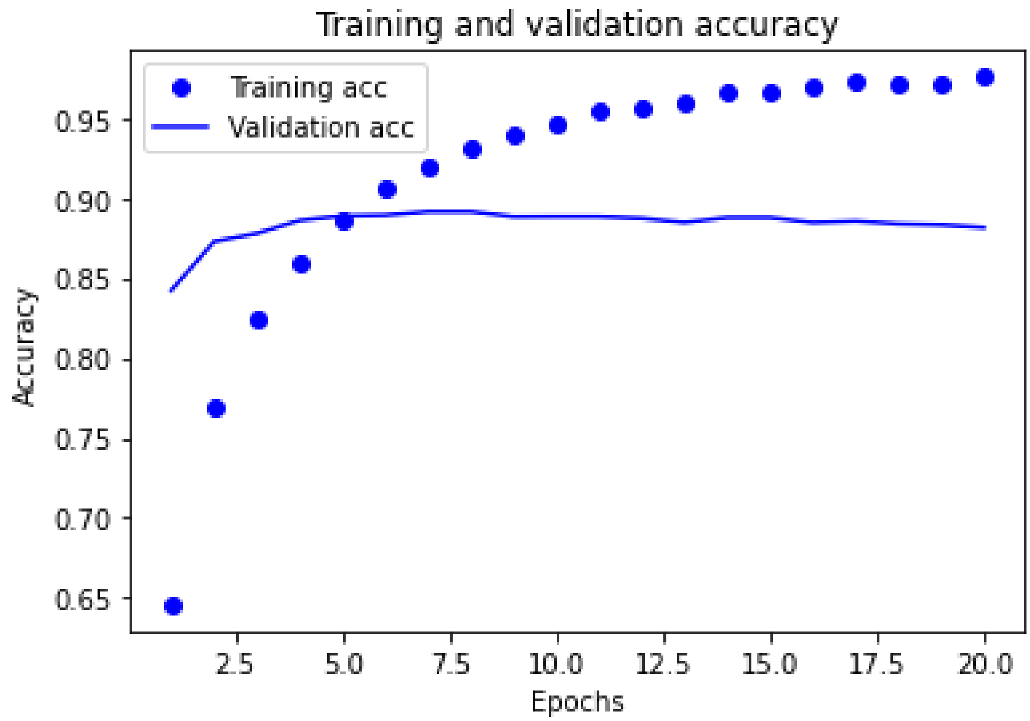
```
Epoch 6/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2723 - accuracy:
0.9054 - val_loss: 0.2750 - val_accuracy: 0.8892
Epoch 7/20
30/30 [==============================] - 1s 17ms/step - loss: 0.2435 - accuracy:
0.9198 - val_loss: 0.2757 - val_accuracy: 0.8915
Epoch 8/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2120 - accuracy:
0.9309 - val_loss: 0.2816 - val_accuracy: 0.8915
Epoch 9/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1892 - accuracy:
0.9391 - val_loss: 0.2966 - val_accuracy: 0.8883
Epoch 10/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1724 - accuracy:
0.9461 - val_loss: 0.3058 - val_accuracy: 0.8884
Epoch 11/20
30/30 [==============================] - 0s 17ms/step - loss: 0.1505 - accuracy:
0.9538 - val_loss: 0.3296 - val_accuracy: 0.8884
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1376 - accuracy:
0.9557 - val_loss: 0.3575 - val_accuracy: 0.8873
Epoch 13/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1307 - accuracy:
0.9592 - val_loss: 0.3558 - val_accuracy: 0.8847
Epoch 14/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1134 - accuracy:
0.9655 - val_loss: 0.3951 - val_accuracy: 0.8878
Epoch 15/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1054 - accuracy:
0.9665 - val_loss: 0.4070 - val_accuracy: 0.8878
Epoch 16/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0936 - accuracy:
0.9700 - val_loss: 0.4192 - val_accuracy: 0.8846
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0881 - accuracy:
0.9727 - val_loss: 0.4814 - val_accuracy: 0.8855
Epoch 18/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0847 - accuracy:
0.9715 - val_loss: 0.4867 - val_accuracy: 0.8838
Epoch 19/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0802 - accuracy:
0.9720 - val_loss: 0.5315 - val_accuracy: 0.8832
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0764 - accuracy:
0.9759 - val_loss: 0.5447 - val_accuracy: 0.8815
```

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

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



[ ]:
```
plt.clf()
acc = history_dict_Dropout["accuracy"]
val_acc = history_dict_Dropout["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
model_Dropout.fit(x_train, y_train, epochs=8, batch_size=512)
results_Dropout = model_Dropout.evaluate(x_test, y_test)
results_Dropout
```

```
Epoch 1/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2544 - accuracy:
0.9212
Epoch 2/8
49/49 [==============================] - 1s 11ms/step - loss: 0.2008 - accuracy:
0.9370
Epoch 3/8
49/49 [==============================] - 1s 10ms/step - loss: 0.1790 - accuracy:
0.9431
Epoch 4/8
49/49 [==============================] - 0s 10ms/step - loss: 0.1683 - accuracy:
0.9469
Epoch 5/8
49/49 [==============================] - 1s 10ms/step - loss: 0.1559 - accuracy:
0.9511
Epoch 6/8
49/49 [==============================] - 1s 10ms/step - loss: 0.1422 - accuracy:
0.9541
Epoch 7/8
49/49 [==============================] - 1s 10ms/step - loss: 0.1387 - accuracy:
```

```
0.9553
Epoch 8/8
49/49 [==============================] - 1s 10ms/step - loss: 0.1295 - accuracy:
0.9564
782/782 [==============================] - 2s 2ms/step - loss: 0.4659 -
accuracy: 0.8722
```

[ ]:
```
[0.465873658657074, 0.872160017490387]
```

The loss on the test set is 0.4659 and accuracy is 0.8722.

Training model with hyper tuned parameters

[ ]:
```
from tensorflow.keras import regularizers
np.random.seed(123)
model_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")
])
model_Hyper.compile(optimizer="rmsprop",
              loss="mse",
              metrics=["accuracy"])
np.random.seed(123)
history_model_Hyper = model_Hyper.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
history_dict_Hyper = history_model_Hyper.history
history_dict_Hyper.keys()
```

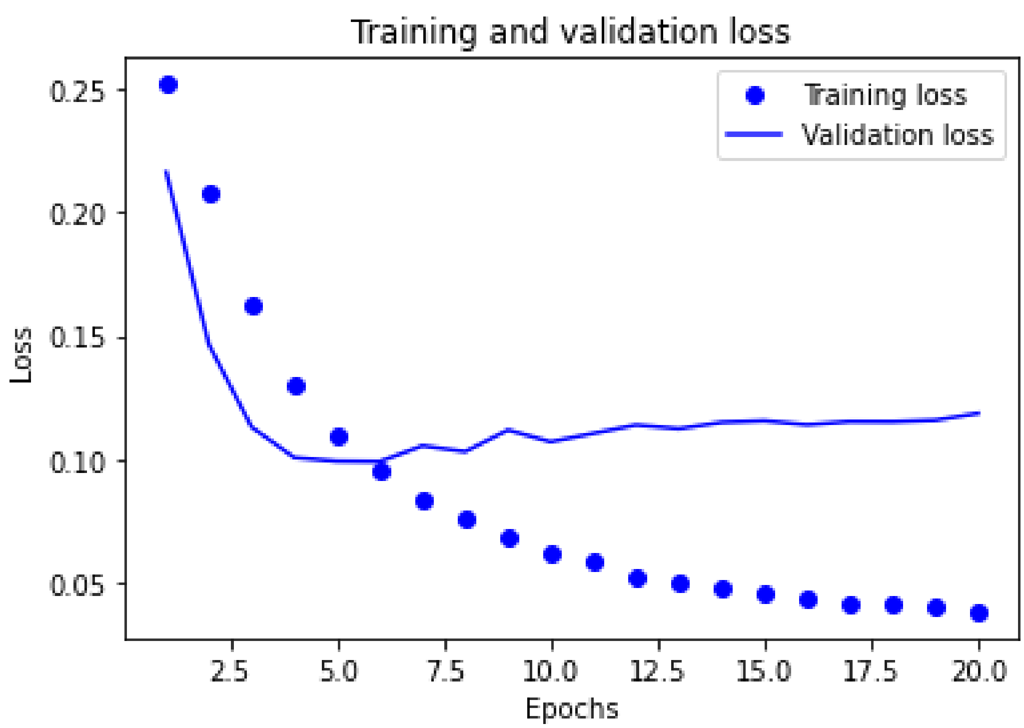
```
Epoch 1/20
30/30 [==============================] - 3s 49ms/step - loss: 0.2518 - accuracy:
0.5781 - val_loss: 0.2163 - val_accuracy: 0.7967
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2078 - accuracy:
0.7137 - val_loss: 0.1465 - val_accuracy: 0.8561
Epoch 3/20
30/30 [==============================] - 1s 18ms/step - loss: 0.1625 - accuracy:
0.8007 - val_loss: 0.1131 - val_accuracy: 0.8700
```

```
Epoch 4/20
30/30 [==============================] - 1s 17ms/step - loss: 0.1302 - accuracy:
0.8539 - val_loss: 0.1008 - val_accuracy: 0.8837
Epoch 5/20
30/30 [==============================] - 1s 19ms/step - loss: 0.1101 - accuracy:
0.8803 - val_loss: 0.0993 - val_accuracy: 0.8825
Epoch 6/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0960 - accuracy:
0.9000 - val_loss: 0.0992 - val_accuracy: 0.8830
Epoch 7/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0837 - accuracy:
0.9177 - val_loss: 0.1056 - val_accuracy: 0.8811
Epoch 8/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0757 - accuracy:
0.9277 - val_loss: 0.1032 - val_accuracy: 0.8841
Epoch 9/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0688 - accuracy:
0.9355 - val_loss: 0.1119 - val_accuracy: 0.8752
Epoch 10/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0617 - accuracy:
0.9452 - val_loss: 0.1072 - val_accuracy: 0.8831
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0593 - accuracy:
0.9485 - val_loss: 0.1106 - val_accuracy: 0.8818
Epoch 12/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0525 - accuracy:
0.9575 - val_loss: 0.1140 - val_accuracy: 0.8800
Epoch 13/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0507 - accuracy:
0.9582 - val_loss: 0.1125 - val_accuracy: 0.8811
Epoch 14/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0477 - accuracy:
0.9610 - val_loss: 0.1150 - val_accuracy: 0.8802
Epoch 15/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0455 - accuracy:
0.9643 - val_loss: 0.1157 - val_accuracy: 0.8781
Epoch 16/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0439 - accuracy:
0.9657 - val_loss: 0.1141 - val_accuracy: 0.8802
Epoch 17/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0412 - accuracy:
0.9692 - val_loss: 0.1154 - val_accuracy: 0.8801
Epoch 18/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0421 - accuracy:
0.9680 - val_loss: 0.1153 - val_accuracy: 0.8806
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0409 - accuracy:
0.9693 - val_loss: 0.1159 - val_accuracy: 0.8807
```

```
Epoch 20/20
30/30 [==============================] - 1s 17ms/step - loss: 0.0385 - accuracy:
0.9705 - val_loss: 0.1187 - val_accuracy: 0.8778
```

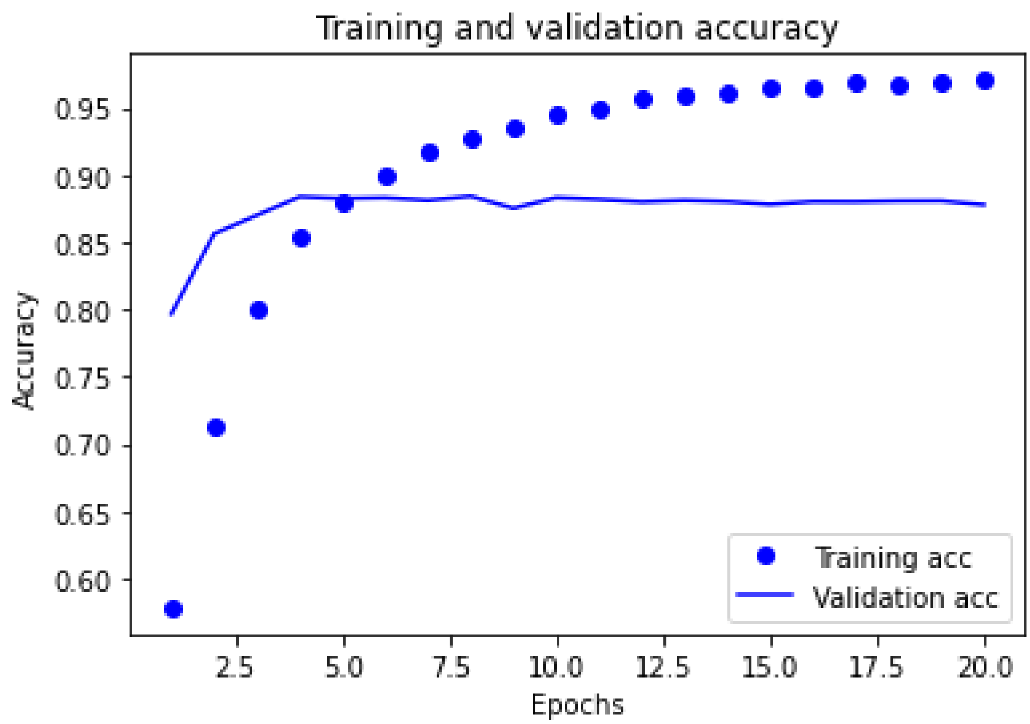
[ ]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[ ]:
```python
loss_values = history_dict_Hyper["loss"]
val_loss_values = history_dict_Hyper["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "bo", label="Training loss")
plt.plot(epochs, val_loss_values, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```python
plt.clf()
acc = history_dict_Hyper["accuracy"]
val_acc = history_dict_Hyper["val_accuracy"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
```

```
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[ ]:
```
model_Hyper.fit(x_train, y_train, epochs=8, batch_size=512)
results_Hyper = model_Hyper.evaluate(x_test, y_test)
results_Hyper
```

```
Epoch 1/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0736 - accuracy:
0.9285
Epoch 2/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0665 - accuracy:
0.9365
Epoch 3/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0627 - accuracy:
0.9411
Epoch 4/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0580 - accuracy:
0.9464
Epoch 5/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0559 - accuracy:
0.9488
Epoch 6/8
```

```
49/49 [==============================] - 1s 11ms/step - loss: 0.0524 - accuracy:
0.9529
Epoch 7/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0505 - accuracy:
0.9552
Epoch 8/8
49/49 [==============================] - 1s 11ms/step - loss: 0.0489 - accuracy:
0.9572
782/782 [==============================] - 2s 2ms/step - loss: 0.1127 -
accuracy: 0.8807
```

[ ]:
```
[0.11273709684610367, 0.8806800246238708]
```

Summary

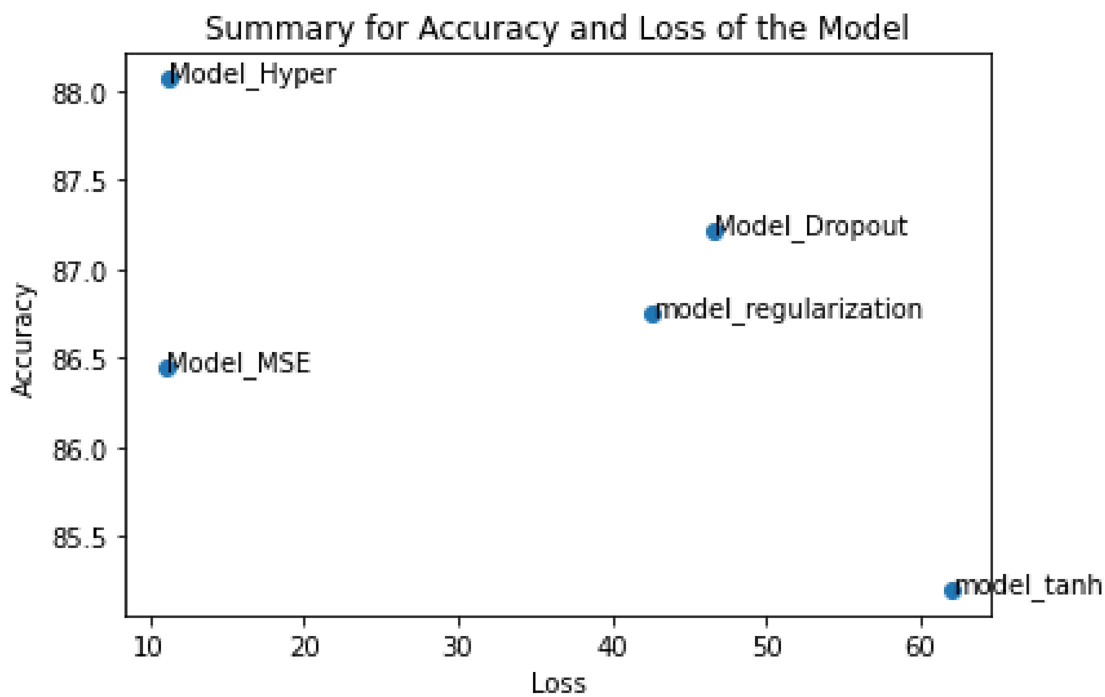
[ ]:
```
All_Models_Loss= np.
  ↪array([results_Dropout[0],results_Hyper[0],results_MSE[0],results_regularization[0],results_
All_Models_Loss
All_Models_Accuracy= np.
  ↪array([results_Dropout[1],results_Hyper[1],results_MSE[1],results_regularization[1],results_
All_Models_Accuracy
Labels=['Model_Dropout','Model_Hyper','Model_MSE','model_regularization','model_tanh']
plt.clf()
```

```
<Figure size 432x288 with 0 Axes>
```

Compilation

[ ]:
```
fig, ax = plt.subplots()
ax.scatter(All_Models_Loss,All_Models_Accuracy)
for i, txt in enumerate(Labels):
    ax.annotate(txt, (All_Models_Loss[i],All_Models_Accuracy[i] ))
plt.title("Summary for Accuracy and Loss of the Model")
plt.ylabel("Accuracy")
plt.xlabel("Loss")

plt.show()
```



Approach: We began by loading the data and defining the maximum number of words to include in each review, as well as the maximum length of the reviews. Next, we created a baseline neural network model featuring one hidden layer with 16 units, using binary cross-entropy as the loss function and ReLU as the activation function for the hidden layer.

We then explored various methods to enhance the model's performance. Initially, we experimented with different numbers of hidden layers by constructing models with one and three hidden layers. After training and evaluating these models on both the training and test datasets, we observed that the model with three hidden layers yielded slightly better validation and test accuracy than the one with a single hidden layer.

Following this, we tested configurations with varying numbers of hidden units: specifically, 32, 64, and 128 units. We trained and assessed these models, plotting the validation accuracy for each. Our findings indicated that increasing the number of hidden units generally improved validation and test accuracy, although excessively high numbers could lead to overfitting.

We also experimented with the mean squared error (MSE) loss function instead of binary cross-entropy. After training and evaluating the MSE model, we found that it did not significantly impact performance compared to the baseline.

Conclusion: To address overfitting, we implemented dropout regularization, creating a new model with dropout layers and training it on the datasets. This approach resulted in higher validation accuracy compared to the baseline model. It can be concluded that different variations of the neural network models yield varying levels of accuracy and loss. The Model_Hyper achieved the highest accuracy and loss, indicating that using three dense layers with a dropout rate of 0.5 optimizes performance on the IMDb dataset. The MSE loss function produced the lowest loss value compared to binary cross-entropy, while the tanh activation function resulted in lower accuracy due to the vanishing gradient issue. The Adam optimizer was effective in model computation. Regularization techniques helped reduce overfitting and resulted in lower losses, with the L-2 model showing slightly improved accuracy. Although the dropout method decreased the loss function, it did not significantly influence accuracy. According to the graph, Model_Hyper exhibited the highest accuracy with a reasonably low loss. Model_MSE had the lowest loss but was less accurate than Model_Hyper. Model_tanh showed lower accuracy than the other models, and model_regularization had high loss and low accuracy relative to the others. Thus, we conclude that Model_Hyper is the best-performing model among those evaluated.