Assignment-6

Murali Krishna Ponnam

700755557

GitHub: https://github.com/Muralikrishna9550/Assignment_6.git

Video: https://drive.google.com/file/d/1_6u4yTINvWZ3YDX9zdJkF2rlXRODeRpw/view?usp=sharing

In class programming: 1

1. Use the use case in the class: a. Add more Dense layers to the existing code and check how the accuracy changes.

2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.

3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below). from sklearn.preprocessing import StandardScaler sc = StandardScaler() Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer

```
[2]  #read the data
     import pandas as pd
     data = pd.read_csv('diabetes.csv')
```

```
[3]  path_to_csv = 'diabetes.csv'
```

```
     import keras
     import pandas
     from keras.models import Sequential
     from keras.layers import Dense, Activation

     # load dataset
     from sklearn.model_selection import train_test_split
     import pandas as pd
     import numpy as np

     dataset = pd.read_csv(path_to_csv, header=None).values

     X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                         test_size=0.25, random_state=87)
     np.random.seed(155)
     my_first_nn = Sequential() # create model
     my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
     my_first_nn.add(Dense(4, activation='relu')) # hidden layer
     my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
     my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                          initial_epoch=0)
     print(my_first_nn.summary())
     print(my_first_nn.evaluate(X_test, Y_test))
```

(variable) my_first_nn: Sequential

```
Epoch 1/100 18/18 [==============================] - 1s 3ms/step - loss: 28.7097 - acc: 0.3385
Epoch 2/100 18/18 [==============================] - 0s 2ms/step - loss: 19.0821 - acc: 0.3403
Epoch 3/100 18/18 [==============================] - 0s 2ms/step - loss: 9.7154 - acc: 0.3368
Epoch 4/100 18/18 [==============================] - 0s 2ms/step - loss: 3.6108 - acc: 0.4549
Epoch 5/100 18/18 [==============================] - 0s 2ms/step - loss: 1.9655 - acc: 0.6337
Epoch 6/100 18/18 [==============================] - 0s 2ms/step - loss: 1.1196 - acc: 0.6215
Epoch 7/100 18/18 [==============================] - 0s 2ms/step - loss: 0.8363 - acc: 0.6528
Epoch 8/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7765 - acc: 0.6510
Epoch 9/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7546 - acc: 0.6562
Epoch 10/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7425 - acc: 0.6528
Epoch 11/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7343 - acc: 0.6528
Epoch 12/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7290 - acc: 0.6528
Epoch 13/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7224 - acc: 0.6510
Epoch 14/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7151 - acc: 0.6493
Epoch 15/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7103 - acc: 0.6493
Epoch 16/100 18/18 [==============================] - 0s 2ms/step - loss: 0.7046 - acc: 0.6493
Epoch 17/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6995 - acc: 0.6493
Epoch 18/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6934 - acc: 0.6510
Epoch 19/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6882 - acc: 0.6510
```

Epoch 20/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6838 - acc: 0.6510
Epoch 21/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6804 - acc: 0.6510
Epoch 22/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6767 - acc: 0.6528
Epoch 23/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6730 - acc: 0.6528
Epoch 24/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6703 - acc: 0.6562
Epoch 25/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6684 - acc: 0.6562
Epoch 26/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6668 - acc: 0.6562
Epoch 27/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6654 - acc: 0.6562
Epoch 28/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6639 - acc: 0.6562
Epoch 29/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6629 - acc: 0.6562
Epoch 30/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6618 - acc: 0.6545
Epoch 31/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6605 - acc: 0.6562
Epoch 32/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6596 - acc: 0.6562
Epoch 33/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6582 - acc: 0.6580
Epoch 34/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6569 - acc: 0.6597
Epoch 35/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6560 - acc: 0.6597
Epoch 36/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6551 - acc: 0.6615
Epoch 37/100 18/18 [==============================] - 0s 3ms/step - loss: 0.6544 - acc: 0.6632
Epoch 38/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6537 - acc: 0.6632
Epoch 39/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6528 - acc: 0.6632
Epoch 40/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6521 - acc: 0.6632
Epoch 41/100 18/18 [==============================] - 0s 3ms/step - loss: 0.6514 - acc: 0.6632
Epoch 42/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6508 - acc: 0.6632
Epoch 43/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6503 - acc: 0.6632
Epoch 44/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6496 - acc: 0.6615
Epoch 45/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6491 - acc: 0.6615
Epoch 46/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6486 - acc: 0.6615
Epoch 47/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6481 - acc: 0.6615
Epoch 48/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6478 - acc: 0.6615
Epoch 49/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6471 - acc: 0.6615
Epoch 50/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6466 - acc: 0.6615
Epoch 51/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6463 - acc: 0.6615
Epoch 52/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6460 - acc: 0.6615
Epoch 53/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6455 - acc: 0.6615
Epoch 54/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6453 - acc: 0.6615
Epoch 55/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6448 - acc: 0.6615
Epoch 56/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6442 - acc: 0.6615
Epoch 57/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6441 - acc: 0.6615
Epoch 58/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6436 - acc: 0.6615
Epoch 59/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6433 - acc: 0.6615
Epoch 60/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6439 - acc: 0.6615
Epoch 61/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6428 - acc: 0.6615
Epoch 62/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6424 - acc: 0.6615

Epoch 63/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6423 - acc: 0.6615
Epoch 64/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6421 - acc: 0.6615
Epoch 65/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6418 - acc: 0.6615
Epoch 66/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6414 - acc: 0.6615
Epoch 67/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6414 - acc: 0.6615
Epoch 68/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6411 - acc: 0.6615
Epoch 69/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6408 - acc: 0.6615
Epoch 70/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6405 - acc: 0.6615
Epoch 71/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6404 - acc: 0.6615
Epoch 72/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6402 - acc: 0.6615
Epoch 73/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6400 - acc: 0.6615
Epoch 74/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6399 - acc: 0.6615
Epoch 75/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6397 - acc: 0.6615
Epoch 76/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6394 - acc: 0.6615
Epoch 77/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6397 - acc: 0.6615
Epoch 78/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6393 - acc: 0.6615
Epoch 79/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6391 - acc: 0.6615
Epoch 80/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6393 - acc: 0.6615
Epoch 81/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6390 - acc: 0.6615
Epoch 82/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6390 - acc: 0.6615
Epoch 83/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6389 - acc: 0.6615
Epoch 84/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6386 - acc: 0.6615
Epoch 85/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6387 - acc: 0.6615
Epoch 86/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6383 - acc: 0.6615
Epoch 87/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6381 - acc: 0.6615
Epoch 88/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6382 - acc: 0.6615
Epoch 89/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6381 - acc: 0.6615
Epoch 90/100 18/18 [==============================] - 0s 3ms/step - loss: 0.6380 - acc: 0.6615
Epoch 91/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6383 - acc: 0.6615
Epoch 92/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6377 - acc: 0.6615
Epoch 93/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6385 - acc: 0.6615
Epoch 94/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6378 - acc: 0.6615
Epoch 95/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6377 - acc: 0.6615
Epoch 96/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6375 - acc: 0.6615
Epoch 97/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6377 - acc: 0.6615
Epoch 98/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6377 - acc: 0.6615
Epoch 99/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6374 - acc: 0.6615
Epoch 100/100 18/18 [==============================] - 0s 2ms/step - loss: 0.6374 - acc: 0.6615
Model: "sequential" _____ Layer (type)
Output Shape Param #
=== dense (Dense) (None,
20) 180 dense_1 (Dense) (None, 4) 84 dense_2 (Dense) (None, 1) 5
=== Total params: 269

(1.05 KB) Trainable params: 269 (1.05 KB) Non-trainable params: 0 (0.00 Byte)
_____ None 6/6
[==============================] - 0s 3ms/step - loss: 0.6606 - acc: 0.6198 [0.660642683506012, 0.6197916865348816]

+ Code + Text

```
[5]  #read the data
     data = pd.read_csv('breastcancer.csv')
```

```
[6]  path_to_csv = 'sample_data/breastcancer.csv'
```

```
[7]  import keras
     import pandas as pd
     import numpy as np
     from keras.models import Sequential
     from keras.layers import Dense, Activation
     from sklearn.datasets import load_breast_cancer
     from sklearn.model_selection import train_test_split

     # load dataset
     cancer_data = load_breast_cancer()
     X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                         test_size=0.25, random_state=87)
     np.random.seed(155)
     my_nn = Sequential() # create model
     my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
     my_nn.add(Dense(1, activation='sigmoid')) # output layer
     my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                              initial_epoch=0)
     print(my_nn.summary())
     print(my_nn.evaluate(X_test, Y_test))
```

Epoch 1/100 14/14 [==============================] - 1s 3ms/step - loss: 102.3078 - acc: 0.3803
Epoch 2/100 14/14 [==============================] - 0s 3ms/step - loss: 59.1148 - acc: 0.3803
Epoch 3/100 14/14 [==============================] - 0s 2ms/step - loss: 18.0792 - acc: 0.4601
Epoch 4/100 14/14 [==============================] - 0s 3ms/step - loss: 3.6943 - acc: 0.7394
Epoch 5/100 14/14 [==============================] - 0s 8ms/step - loss: 1.1049 - acc: 0.8005
Epoch 6/100 14/14 [==============================] - 0s 3ms/step - loss: 0.8925 - acc: 0.8451
Epoch 7/100 14/14 [==============================] - 0s 3ms/step - loss: 0.8376 - acc: 0.8498
Epoch 8/100 14/14 [==============================] - 0s 3ms/step - loss: 0.8064 - acc: 0.8427
Epoch 9/100 14/14 [==============================] - 0s 3ms/step - loss: 0.7676 - acc: 0.8545
Epoch 10/100 14/14 [==============================] - 0s 3ms/step - loss: 0.7416 - acc: 0.8615
Epoch 11/100 14/14 [==============================] - 0s 2ms/step - loss: 0.7417 - acc: 0.8474
Epoch 12/100 14/14 [==============================] - 0s 3ms/step - loss: 0.7564 - acc: 0.8779
Epoch 13/100 14/14 [==============================] - 0s 3ms/step - loss: 0.6699 - acc: 0.8568

Epoch 14/100 14/14 [==============================] - 0s 3ms/step - loss: 0.6799 - acc: 0.8545
Epoch 15/100 14/14 [==============================] - 0s 3ms/step - loss: 0.6825 - acc: 0.8732
Epoch 16/100 14/14 [==============================] - 0s 3ms/step - loss: 0.6358 - acc: 0.8756
Epoch 17/100 14/14 [==============================] - 0s 3ms/step - loss: 0.6047 - acc: 0.8732
Epoch 18/100 14/14 [==============================] - 0s 6ms/step - loss: 0.5634 - acc: 0.8803
Epoch 19/100 14/14 [==============================] - 0s 3ms/step - loss: 0.5546 - acc: 0.8756
Epoch 20/100 14/14 [==============================] - 0s 3ms/step - loss: 0.5656 - acc: 0.8803
Epoch 21/100 14/14 [==============================] - 0s 3ms/step - loss: 0.5279 - acc: 0.8873
Epoch 22/100 14/14 [==============================] - 0s 3ms/step - loss: 0.5286 - acc: 0.8850
Epoch 23/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4875 - acc: 0.8756
Epoch 24/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4942 - acc: 0.8967
Epoch 25/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4791 - acc: 0.8873
Epoch 26/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4725 - acc: 0.8897
Epoch 27/100 14/14 [==============================] - 0s 2ms/step - loss: 0.4402 - acc: 0.8897
Epoch 28/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4333 - acc: 0.9038
Epoch 29/100 14/14 [==============================] - 0s 2ms/step - loss: 0.4272 - acc: 0.8944
Epoch 30/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3860 - acc: 0.9108
Epoch 31/100 14/14 [==============================] - 0s 3ms/step - loss: 0.4952 - acc: 0.8920
Epoch 32/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3922 - acc: 0.8991
Epoch 33/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3771 - acc: 0.9085
Epoch 34/100 14/14 [==============================] - 0s 3ms/step - loss: 0.7166 - acc: 0.8592
Epoch 35/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3751 - acc: 0.9108
Epoch 36/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3469 - acc: 0.9014
Epoch 37/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3559 - acc: 0.9038
Epoch 38/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3142 - acc: 0.9108
Epoch 39/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3540 - acc: 0.9061
Epoch 40/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3123 - acc: 0.9061
Epoch 41/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3201 - acc: 0.9108
Epoch 42/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2783 - acc: 0.9178
Epoch 43/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2841 - acc: 0.9061
Epoch 44/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2664 - acc: 0.9085
Epoch 45/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2509 - acc: 0.9155
Epoch 46/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2837 - acc: 0.9202
Epoch 47/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2776 - acc: 0.9155
Epoch 48/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2496 - acc: 0.9202
Epoch 49/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2970 - acc: 0.9202
Epoch 50/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2562 - acc: 0.9296
Epoch 51/100 14/14 [==============================] - 0s 3ms/step - loss: 0.3467 - acc: 0.9085
Epoch 52/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2634 - acc: 0.9202
Epoch 53/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2171 - acc: 0.9272
Epoch 54/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2175 - acc: 0.9249
Epoch 55/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2488 - acc: 0.9272
Epoch 56/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2668 - acc: 0.9249

Epoch 57/100 14/14 [==============================] - 0s 4ms/step - loss: 0.2485 - acc: 0.9272
Epoch 58/100 14/14 [==============================] - 0s 5ms/step - loss: 0.2832 - acc: 0.9085
Epoch 59/100 14/14 [==============================] - 0s 4ms/step - loss: 0.2214 - acc: 0.9249
Epoch 60/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2104 - acc: 0.9272
Epoch 61/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2217 - acc: 0.9296
Epoch 62/100 14/14 [==============================] - 0s 3ms/step - loss: 0.1963 - acc: 0.9296
Epoch 63/100 14/14 [==============================] - 0s 3ms/step - loss: 0.1907 - acc: 0.9437
Epoch 64/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2267 - acc: 0.9249
Epoch 65/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2001 - acc: 0.9178
Epoch 66/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1667 - acc: 0.9390
Epoch 67/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1678 - acc: 0.9296
Epoch 68/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1771 - acc: 0.9366
Epoch 69/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1519 - acc: 0.9296
Epoch 70/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1789 - acc: 0.9249
Epoch 71/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1930 - acc: 0.9366
Epoch 72/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2408 - acc: 0.9202
Epoch 73/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2966 - acc: 0.9038
Epoch 74/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2153 - acc: 0.9343
Epoch 75/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1956 - acc: 0.9390
Epoch 76/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1452 - acc: 0.9413
Epoch 77/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1679 - acc: 0.9343
Epoch 78/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1894 - acc: 0.9343
Epoch 79/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2121 - acc: 0.9296
Epoch 80/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1513 - acc: 0.9390
Epoch 81/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1422 - acc: 0.9319
Epoch 82/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1342 - acc: 0.9390
Epoch 83/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1405 - acc: 0.9413
Epoch 84/100 14/14 [==============================] - 0s 4ms/step - loss: 0.1360 - acc: 0.9413
Epoch 85/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1949 - acc: 0.9296
Epoch 86/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2018 - acc: 0.9225
Epoch 87/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1425 - acc: 0.9484
Epoch 88/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1750 - acc: 0.9390
Epoch 89/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1313 - acc: 0.9413
Epoch 90/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1288 - acc: 0.9484
Epoch 91/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1879 - acc: 0.9272
Epoch 92/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1723 - acc: 0.9531
Epoch 93/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1232 - acc: 0.9531
Epoch 94/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1199 - acc: 0.9460
Epoch 95/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1317 - acc: 0.9531
Epoch 96/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1965 - acc: 0.9178
Epoch 97/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1730 - acc: 0.9296
Epoch 98/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1797 - acc: 0.9319
Epoch 99/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1189 - acc: 0.9413

Epoch 100/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1187 - acc: 0.9507
Model: "sequential_1" _____ Layer (type) Output Shape Param # === dense_3 (Dense) (None, 20) 620 dense_4 (Dense) (None, 1) 21 === Total params: 641 (2.50 KB) Trainable params: 641 (2.50 KB) Non-trainable params: 0 (0.00 Byte) _____ None 5/5 [==============================] - 0s 3ms/step - loss: 0.1654 - acc: 0.9371
[0.16541491448879242, 0.9370629191398621]

```
[8]  #read the data
     data = pd.read_csv('breastcancer.csv')
```

```
[9]  path_to_csv = 'breastcancer.csv'
```

```
[10] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
```

```
     import keras
     import pandas as pd
     import numpy as np
     from keras.models import Sequential
     from keras.layers import Dense, Activation
     from sklearn.datasets import load_breast_cancer
     from sklearn.model_selection import train_test_split

     # load dataset
     cancer_data = load_breast_cancer()
     X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                         test_size=0.25, random_state=87)
     np.random.seed(155)
     my_nn = Sequential() # create model
     my_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
     my_nn.add(Dense(1, activation='sigmoid')) # output layer
     my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
                              initial_epoch=0)
     print(my_nn.summary())
     print(my_nn.evaluate(X_test, Y_test))
```

Epoch 1/100 14/14 [==============================] - 1s 2ms/step - loss: 160.3251 - acc: 0.6197
Epoch 2/100 14/14 [==============================] - 0s 2ms/step - loss: 114.6389 - acc: 0.6197
Epoch 3/100 14/14 [==============================] - 0s 2ms/step - loss: 72.7626 - acc: 0.6197
Epoch 4/100 14/14 [==============================] - 0s 2ms/step - loss: 31.9955 - acc: 0.5728
Epoch 5/100 14/14 [==============================] - 0s 2ms/step - loss: 10.4390 - acc: 0.2653

Epoch 6/100 14/14 [==============================] - 0s 2ms/step - loss: 8.9202 - acc: 0.2418
Epoch 7/100 14/14 [==============================] - 0s 3ms/step - loss: 6.1245 - acc: 0.1948
Epoch 8/100 14/14 [==============================] - 0s 2ms/step - loss: 3.5213 - acc: 0.2746
Epoch 9/100 14/14 [==============================] - 0s 2ms/step - loss: 1.8175 - acc: 0.4249
Epoch 10/100 14/14 [==============================] - 0s 2ms/step - loss: 1.0556 - acc: 0.6127
Epoch 11/100 14/14 [==============================] - 0s 2ms/step - loss: 0.6782 - acc: 0.7254
Epoch 12/100 14/14 [==============================] - 0s 2ms/step - loss: 0.5148 - acc: 0.7887
Epoch 13/100 14/14 [==============================] - 0s 2ms/step - loss: 0.4347 - acc: 0.8216
Epoch 14/100 14/14 [==============================] - 0s 2ms/step - loss: 0.4257 - acc: 0.8216
Epoch 15/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3664 - acc: 0.8732
Epoch 16/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3560 - acc: 0.8568
Epoch 17/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3169 - acc: 0.8709
Epoch 18/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3153 - acc: 0.8826
Epoch 19/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2856 - acc: 0.8897
Epoch 20/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2775 - acc: 0.8944
Epoch 21/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2709 - acc: 0.8967
Epoch 22/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2642 - acc: 0.9061
Epoch 23/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2543 - acc: 0.9061
Epoch 24/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2589 - acc: 0.8944
Epoch 25/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2477 - acc: 0.8991
Epoch 26/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2414 - acc: 0.9108
Epoch 27/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2447 - acc: 0.9061
Epoch 28/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2293 - acc: 0.9131
Epoch 29/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2413 - acc: 0.9085
Epoch 30/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2250 - acc: 0.9061
Epoch 31/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2217 - acc: 0.9202
Epoch 32/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2429 - acc: 0.9061
Epoch 33/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2235 - acc: 0.9178
Epoch 34/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2176 - acc: 0.9202
Epoch 35/100 14/14 [==============================] - 0s 3ms/step - loss: 0.2322 - acc: 0.9202
Epoch 36/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2222 - acc: 0.9131
Epoch 37/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2351 - acc: 0.9202
Epoch 38/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2131 - acc: 0.9131
Epoch 39/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2054 - acc: 0.9155
Epoch 40/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2411 - acc: 0.8991
Epoch 41/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2008 - acc: 0.9225
Epoch 42/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1937 - acc: 0.9202
Epoch 43/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2065 - acc: 0.9272
Epoch 44/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1985 - acc: 0.9202
Epoch 45/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1973 - acc: 0.9225
Epoch 46/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2064 - acc: 0.9085
Epoch 47/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1893 - acc: 0.9225
Epoch 48/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1881 - acc: 0.9249

Epoch 49/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1889 - acc: 0.9272
Epoch 50/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2146 - acc: 0.9155
Epoch 51/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1807 - acc: 0.9343
Epoch 52/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1863 - acc: 0.9249
Epoch 53/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2141 - acc: 0.9225
Epoch 54/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1799 - acc: 0.9272
Epoch 55/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1725 - acc: 0.9366
Epoch 56/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1869 - acc: 0.9272
Epoch 57/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1812 - acc: 0.9272
Epoch 58/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1710 - acc: 0.9319
Epoch 59/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1879 - acc: 0.9202
Epoch 60/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1745 - acc: 0.9343
Epoch 61/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1693 - acc: 0.9225
Epoch 62/100 14/14 [==============================] - 0s 4ms/step - loss: 0.1701 - acc: 0.9343
Epoch 63/100 14/14 [==============================] - 0s 3ms/step - loss: 0.1901 - acc: 0.9225
Epoch 64/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1655 - acc: 0.9343
Epoch 65/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2049 - acc: 0.9131
Epoch 66/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1975 - acc: 0.9272
Epoch 67/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1665 - acc: 0.9413
Epoch 68/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1680 - acc: 0.9390
Epoch 69/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1778 - acc: 0.9366
Epoch 70/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1947 - acc: 0.9225
Epoch 71/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2147 - acc: 0.9131
Epoch 72/100 14/14 [==============================] - 0s 2ms/step - loss: 0.3321 - acc: 0.8967
Epoch 73/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2058 - acc: 0.9202
Epoch 74/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1865 - acc: 0.9225
Epoch 75/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1576 - acc: 0.9319
Epoch 76/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1567 - acc: 0.9390
Epoch 77/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1586 - acc: 0.9319
Epoch 78/100 14/14 [==============================] - 0s 2ms/step - loss: 0.2066 - acc: 0.9155
Epoch 79/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1565 - acc: 0.9413
Epoch 80/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1486 - acc: 0.9343
Epoch 81/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1546 - acc: 0.9390
Epoch 82/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1621 - acc: 0.9366
Epoch 83/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1539 - acc: 0.9366
Epoch 84/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1622 - acc: 0.9366
Epoch 85/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1948 - acc: 0.9296
Epoch 86/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1516 - acc: 0.9366
Epoch 87/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1571 - acc: 0.9343
Epoch 88/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1403 - acc: 0.9437
Epoch 89/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1414 - acc: 0.9413
Epoch 90/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1428 - acc: 0.9437
Epoch 91/100 14/14 [==============================] - 0s 3ms/step - loss: 0.1487 - acc: 0.9343

Epoch 92/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1601 - acc: 0.9390
Epoch 93/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1767 - acc: 0.9272
Epoch 94/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1575 - acc: 0.9413
Epoch 95/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1773 - acc: 0.9272
Epoch 96/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1891 - acc: 0.9225
Epoch 97/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1407 - acc: 0.9390
Epoch 98/100 14/14 [==============================] - 0s 3ms/step - loss: 0.1373 - acc: 0.9484
Epoch 99/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1444 - acc: 0.9296
Epoch 100/100 14/14 [==============================] - 0s 2ms/step - loss: 0.1486 - acc: 0.9319
Model: "sequential_2" _____ Layer (type) Output Shape Param # === dense_5 (Dense) (None, 20) 620 dense_6 (Dense) (None, 1) 21 === Total params: 641 (2.50 KB) Trainable params: 641 (2.50 KB) Non-trainable params: 0 (0.00 Byte) _____ None 5/5 [==============================] - 0s 3ms/step - loss: 0.2498 - acc: 0.9021
[0.24980774521827698, 0.9020978808403015]

In class programming: 2

Use Image Classification on the hand written digits data set (mnist)

1. Plot the loss and accuracy for both training data and validation data using the history object in the source code.

2. Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

4. Run the same code without scaling the images and check the performance?

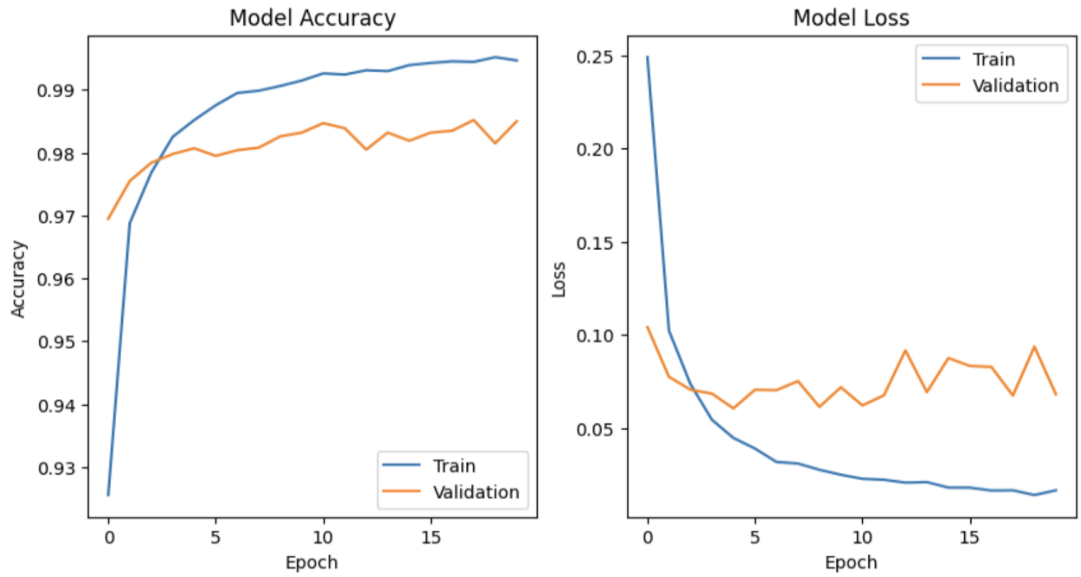
```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

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

```
# train the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                    epochs=20, batch_size=128)

# plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/20
469/469 [==============================] - 10s 21ms/step - loss: 0.2489 - accuracy: 0.9255 - val_loss: 0.1042 - val_accuracy: 0.9695
Epoch 2/20
469/469 [==============================] - 10s 21ms/step - loss: 0.1023 - accuracy: 0.9688 -
```

val_loss: 0.0775 - val_accuracy: 0.9755
Epoch 3/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0736 - accuracy: 0.9768 - val_loss: 0.0706 - val_accuracy: 0.9784
Epoch 4/20
469/469 [==============================] - 14s 31ms/step - loss: 0.0545 - accuracy: 0.9825 - val_loss: 0.0685 - val_accuracy: 0.9798
Epoch 5/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0447 - accuracy: 0.9852 - val_loss: 0.0606 - val_accuracy: 0.9807
Epoch 6/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0391 - accuracy: 0.9876 - val_loss: 0.0706 - val_accuracy: 0.9795
Epoch 7/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0319 - accuracy: 0.9895 - val_loss: 0.0704 - val_accuracy: 0.9804
Epoch 8/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0310 - accuracy: 0.9899 - val_loss: 0.0752 - val_accuracy: 0.9808
Epoch 9/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0277 - accuracy: 0.9906 - val_loss: 0.0614 - val_accuracy: 0.9826
Epoch 10/20
469/469 [==============================] - 10s 20ms/step - loss: 0.0250 - accuracy: 0.9915 - val_loss: 0.0719 - val_accuracy: 0.9832
Epoch 11/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0228 - accuracy: 0.9926 - val_loss: 0.0622 - val_accuracy: 0.9847
Epoch 12/20
469/469 [==============================] - 10s 20ms/step - loss: 0.0223 - accuracy: 0.9924 - val_loss: 0.0676 - val_accuracy: 0.9839
Epoch 13/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0208 - accuracy: 0.9931 - val_loss: 0.0917 - val_accuracy: 0.9805
Epoch 14/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0211 - accuracy: 0.9930 - val_loss: 0.0693 - val_accuracy: 0.9832
Epoch 15/20
469/469 [==============================] - 10s 20ms/step - loss: 0.0181 - accuracy: 0.9939 - val_loss: 0.0876 - val_accuracy: 0.9819
Epoch 16/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0181 - accuracy: 0.9943 - val_loss: 0.0833 - val_accuracy: 0.9832
Epoch 17/20

469/469 [==============================] - 10s 20ms/step - loss: 0.0165 - accuracy: 0.9945 - val_loss: 0.0828 - val_accuracy: 0.9835
Epoch 18/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0166 - accuracy: 0.9944 - val_loss: 0.0675 - val_accuracy: 0.9852
Epoch 19/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0141 - accuracy: 0.9952 - val_loss: 0.0937 - val_accuracy: 0.9815
Epoch 20/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0166 - accuracy: 0.9947 - val_loss: 0.0682 - val_accuracy: 0.9850

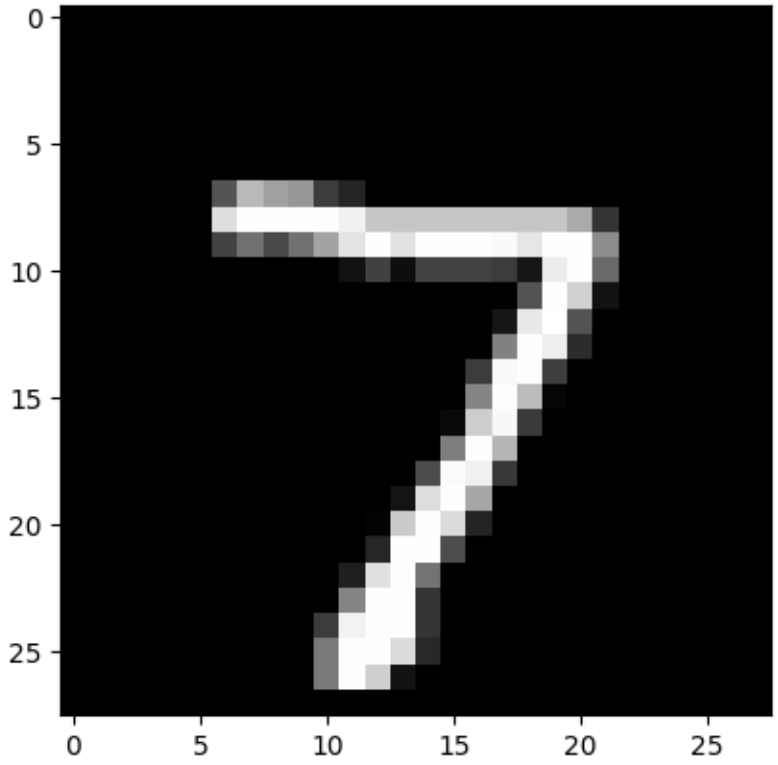


```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

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

# train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)

# plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# make a prediction on the image using the trained model
prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

```
Epoch 1/20
469/469 [==============================] - 9s 16ms/step - loss: 0.2501 - accuracy: 0.9247 - val_loss: 0.1042 - val_accuracy: 0.9654
Epoch 2/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0997 - accuracy: 0.9696 - val_loss: 0.0816 - val_accuracy: 0.9736
Epoch 3/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0711 - accuracy: 0.9772 - val_loss: 0.0737 - val_accuracy: 0.9756
Epoch 4/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0563 - accuracy: 0.9815 - val_loss: 0.0778 - val_accuracy: 0.9771
```

Epoch 5/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0464 - accuracy: 0.9847 - val_loss: 0.0615 - val_accuracy: 0.9813
Epoch 6/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0411 - accuracy: 0.9866 - val_loss: 0.0619 - val_accuracy: 0.9806
Epoch 7/20
469/469 [==============================] - 8s 16ms/step - loss: 0.0331 - accuracy: 0.9888 - val_loss: 0.0559 - val_accuracy: 0.9832
Epoch 8/20
469/469 [==============================] - 9s 18ms/step - loss: 0.0292 - accuracy: 0.9907 - val_loss: 0.0716 - val_accuracy: 0.9795
Epoch 9/20
469/469 [==============================] - 9s 20ms/step - loss: 0.0282 - accuracy: 0.9903 - val_loss: 0.0788 - val_accuracy: 0.9808
Epoch 10/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0295 - accuracy: 0.9903 - val_loss: 0.0609 - val_accuracy: 0.9832
Epoch 11/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0231 - accuracy: 0.9919 - val_loss: 0.0766 - val_accuracy: 0.9810
Epoch 12/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0220 - accuracy: 0.9924 - val_loss: 0.0692 - val_accuracy: 0.9827
Epoch 13/20
469/469 [==============================] - 8s 16ms/step - loss: 0.0205 - accuracy: 0.9926 - val_loss: 0.0689 - val_accuracy: 0.9832
Epoch 14/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0198 - accuracy: 0.9932 - val_loss: 0.0755 - val_accuracy: 0.9825
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0185 - accuracy: 0.9939 - val_loss: 0.0668 - val_accuracy: 0.9826
Epoch 16/20
469/469 [==============================] - 8s 16ms/step - loss: 0.0183 - accuracy: 0.9937 - val_loss: 0.0804 - val_accuracy: 0.9837
Epoch 17/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0173 - accuracy: 0.9941 - val_loss: 0.0707 - val_accuracy: 0.9848
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0156 - accuracy: 0.9947 - val_loss: 0.0799 - val_accuracy: 0.9847
Epoch 19/20
469/469 [==============================] - 8s 17ms/step - loss: 0.0169 - accuracy: 0.9941 -

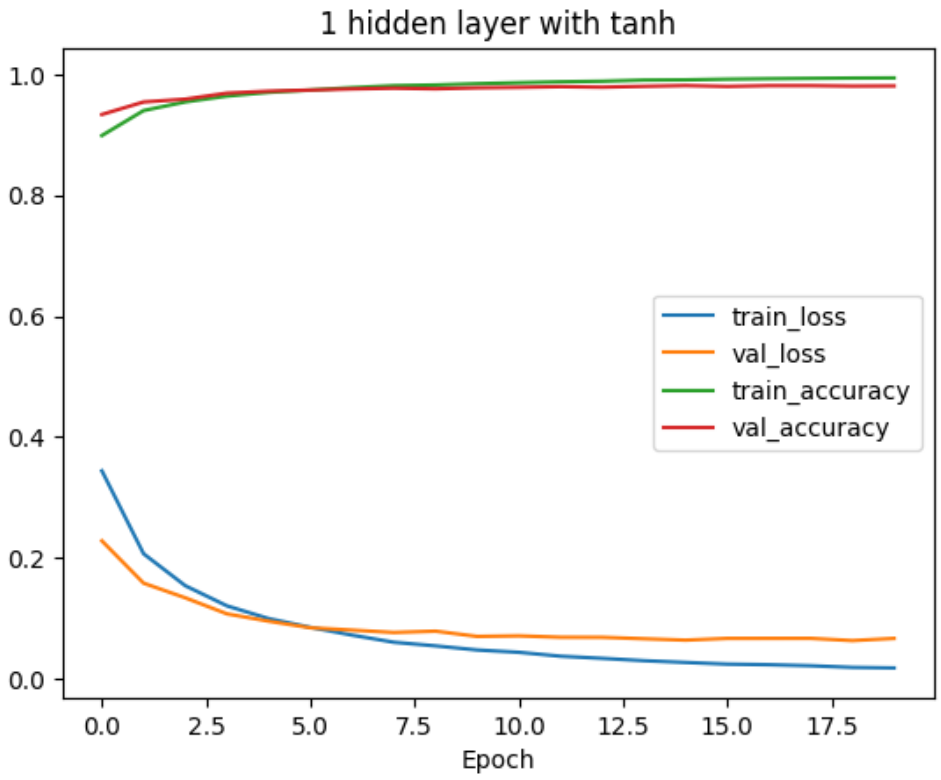
val_loss: 0.0822 - val_accuracy: 0.9821
Epoch 20/20
469/469 [==============================] - 8s 18ms/step - loss: 0.0142 - accuracy: 0.9956 - val_loss: 0.0731 - val_accuracy: 0.9848



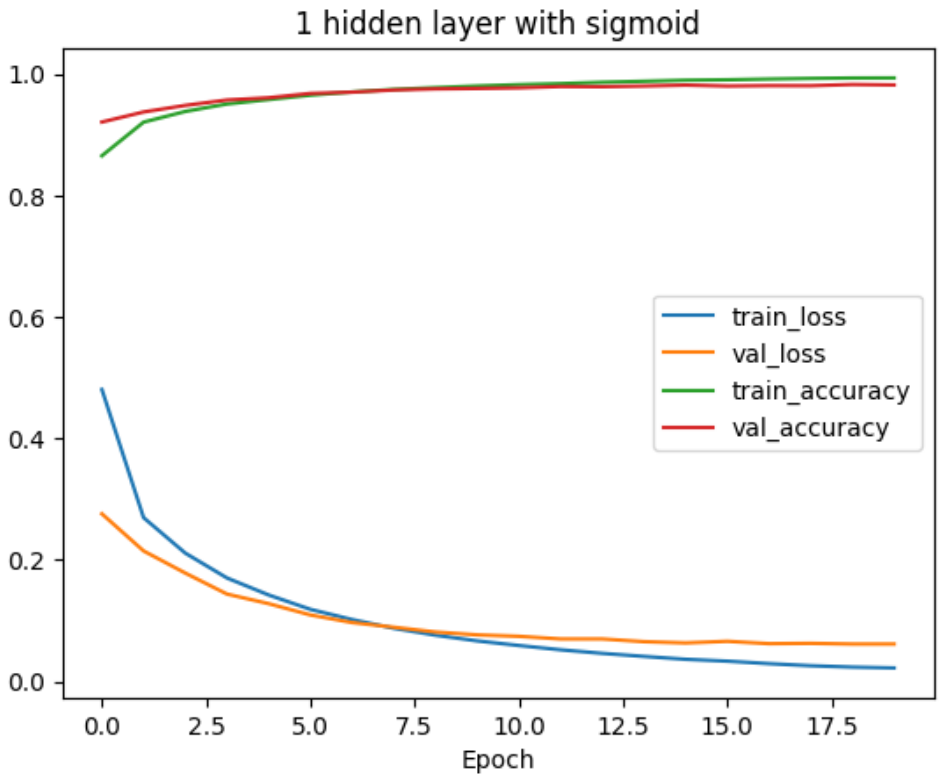
1/1 [==============================] - 0s 90ms/step
Model prediction: 7

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
```

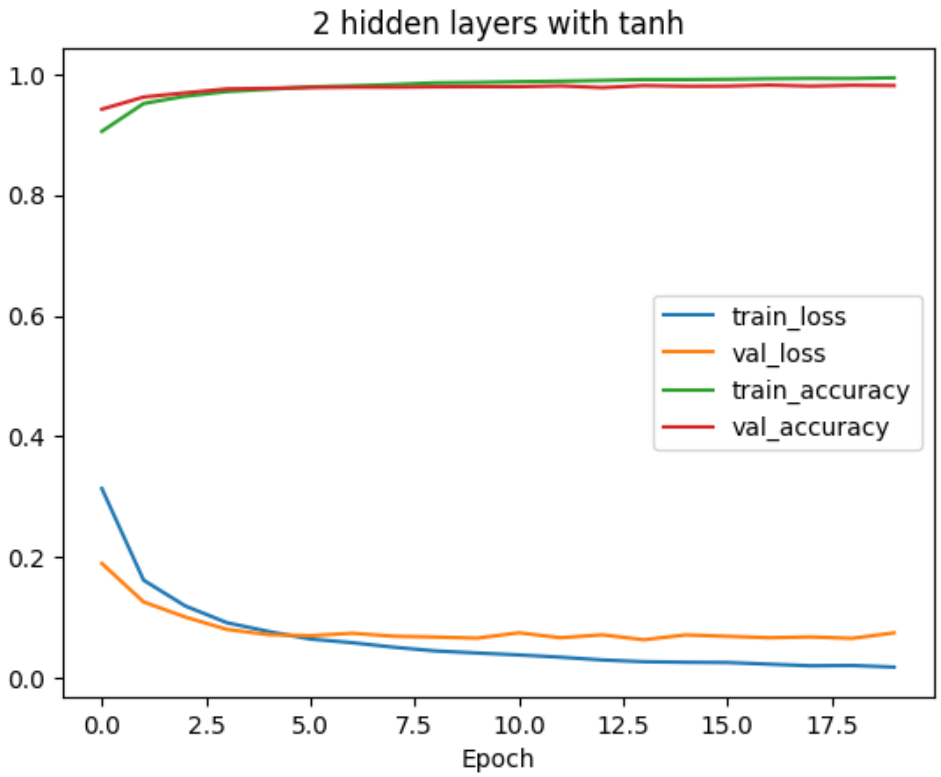
```
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))

# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```



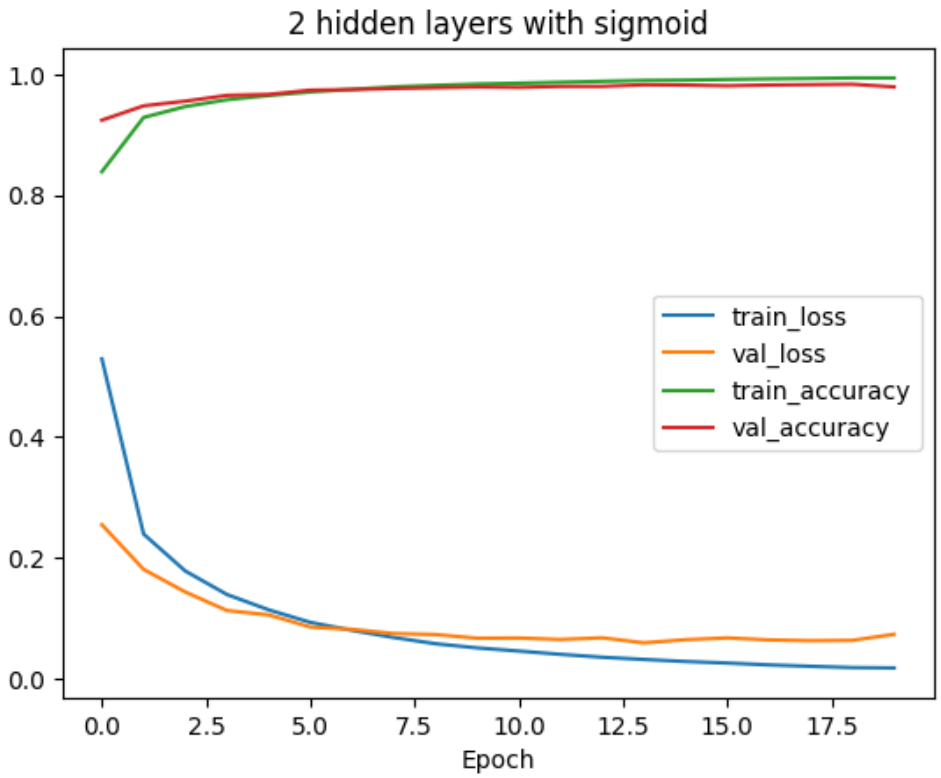
1 hidden layer with tanh - Test loss: 0.0664, Test accuracy: 0.9807



1 hidden layer with sigmoid - Test loss: 0.0616, Test accuracy: 0.9820



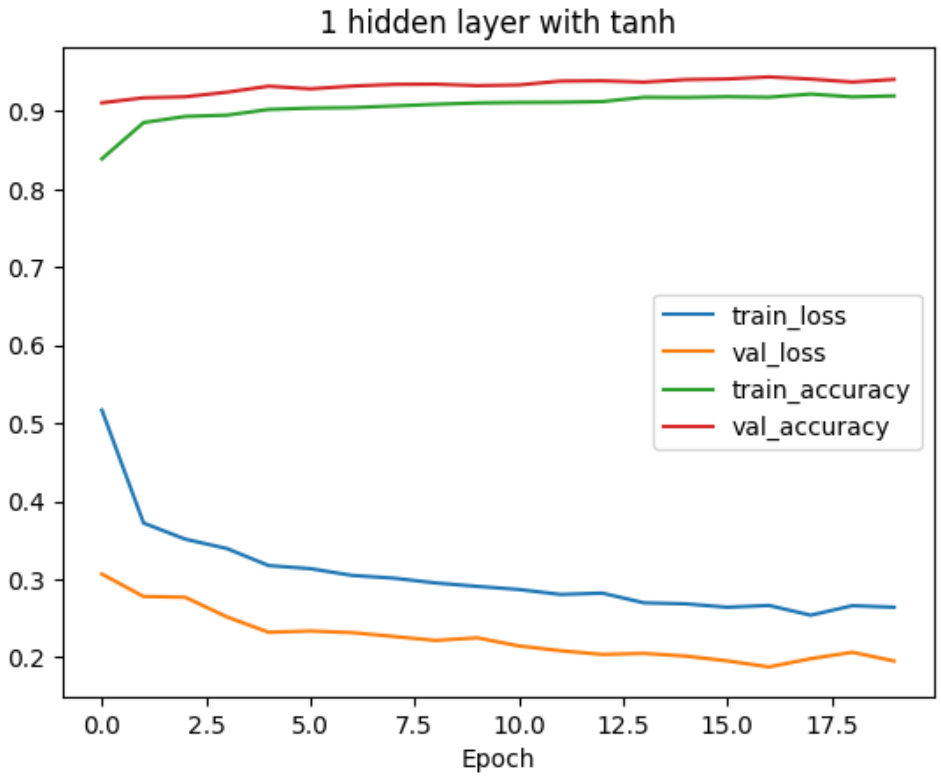
2 hidden layers with tanh - Test loss: 0.0744, Test accuracy: 0.9812



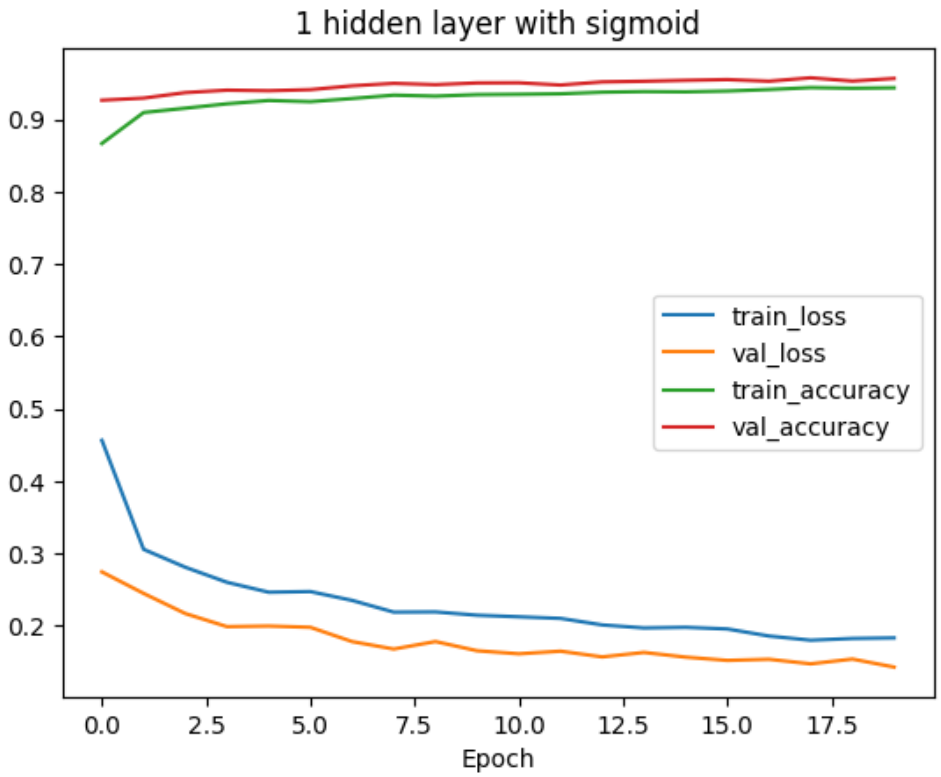
2 hidden layers with sigmoid - Test loss: 0.0733, Test accuracy: 0.9792

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with tanh', model))
```

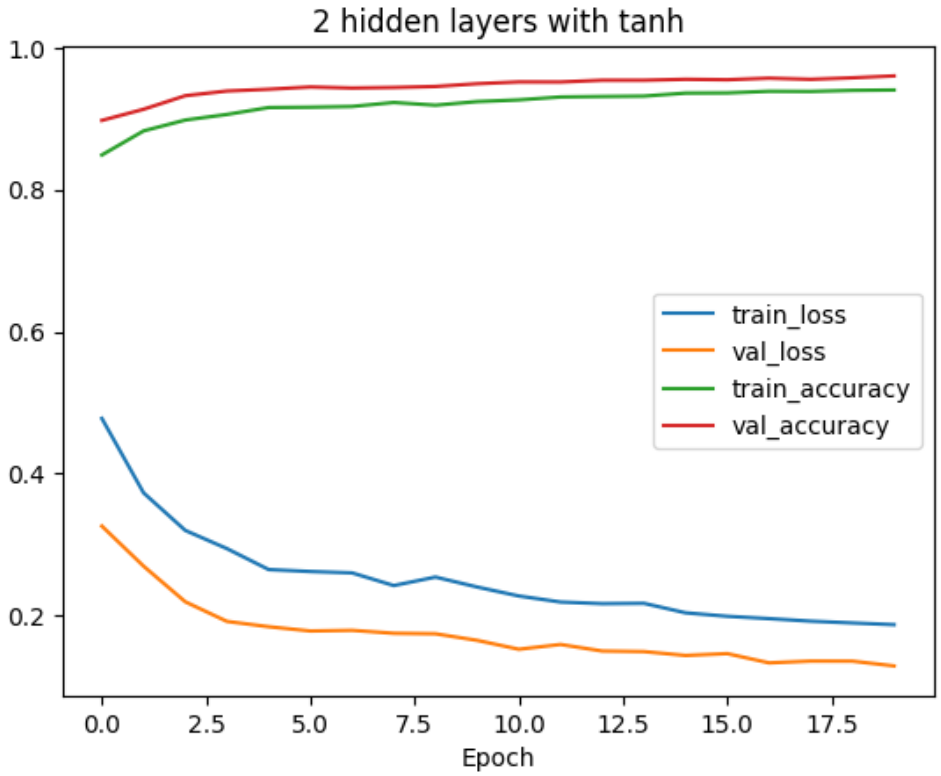
```python
# model with 2 hidden layers and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden layers with sigmoid', model))

# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                        epochs=20, batch_size=128, verbose=0)
    # plot loss and accuracy curves
    plt.plot(history.history['loss'], label='train_loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.plot(history.history['accuracy'], label='train_accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.title(name)
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()

    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```



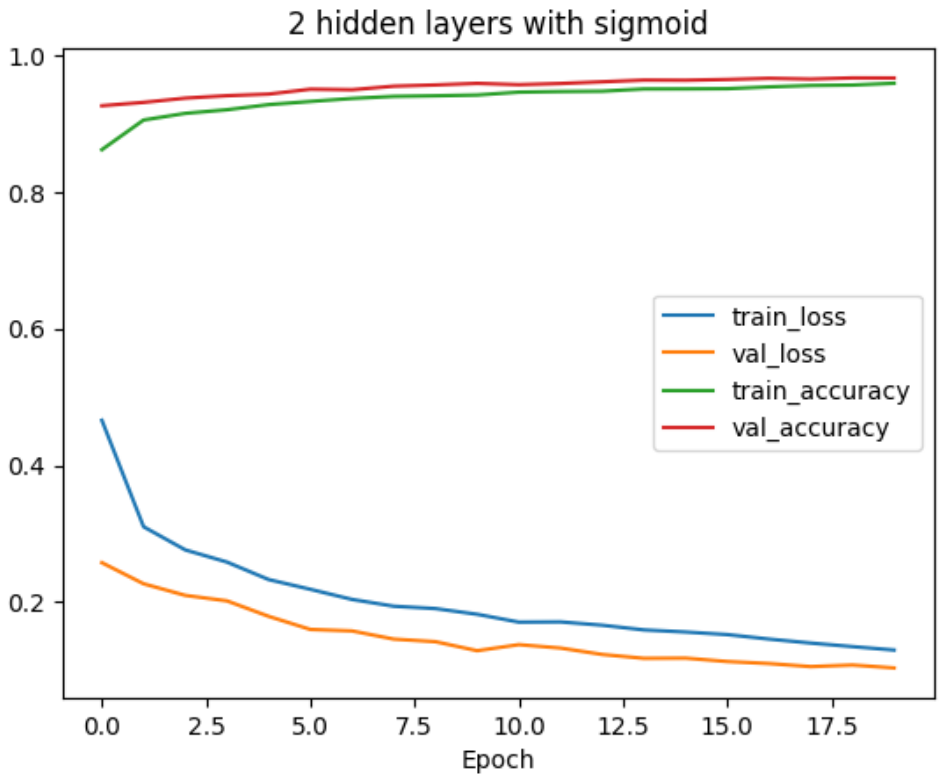
1 hidden layer with tanh - Test loss: 0.1951, Test accuracy: 0.9407



1 hidden layer with sigmoid - Test loss: 0.1426, Test accuracy: 0.9568



2 hidden layers with tanh - Test loss: 0.1289, Test accuracy: 0.9602



2 hidden layers with sigmoid - Test loss: 0.1037, Test accuracy: 0.9665