Neural Networks And Deep Learning , Assignment - 2

Name - Jyothi Sai Rajesh

Kunapareddy

id - 700742325

Github link :

https://github.com/Rk-oo7/NNDL_Assignment-2.git

[1]
```
#read the data
import pandas as pd
data = pd.read_csv('sample_data/diabetes.csv')
```

```
path_to_csv = 'sample_data/diabetes.csv'
```

```
import keras
import pandas
from keras.models import Sequential
from keras.layers.core 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))
```

```
Epoch 1/100
18/18 [==============================] - 1s 2ms/step - loss: 4.2365 - acc: 0.3819
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 1.6883 - acc: 0.5712
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 1.4779 - acc: 0.5729
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 1.3767 - acc: 0.5955
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 1.3374 - acc: 0.5816
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 1.2746 - acc: 0.6128
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 1.2443 - acc: 0.5885
Epoch 8/100
18/18 [==============================] - 0s 2ms/step - loss: 1.1870 - acc: 0.6233
Epoch 9/100
18/18 [==============================] - 0s 2ms/step - loss: 1.1419 - acc: 0.6146
Epoch 10/100
18/18 [==============================] - 0s 2ms/step - loss: 1.1052 - acc: 0.6042
Epoch 11/100
18/18 [==============================] - 0s 2ms/step - loss: 1.0720 - acc: 0.6250
Epoch 12/100
18/18 [==============================] - 0s 2ms/step - loss: 1.0338 - acc: 0.6111
Epoch 13/100
18/18 [==============================] - 0s 2ms/step - loss: 1.0110 - acc: 0.6285
Epoch 14/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9916 - acc: 0.6128
Epoch 15/100
18/18 [==============================] - 0s 2ms/step - loss: 0.9251 - acc: 0.6389
Epoch 16/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8849 - acc: 0.6302
Epoch 17/100
18/18 [==============================] - 0s 2ms/step - loss: 0.8486 - acc: 0.6389
Epoch 18/100
18/18 [==============================] - 0s 3ms/step - loss: 0.8325 - acc: 0.6337
```

```
Epoch 30/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6930 - acc: 0.6736
Epoch 31/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6828 - acc: 0.6684
Epoch 32/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6843 - acc: 0.6667
Epoch 33/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6767 - acc: 0.6753
Epoch 34/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6708 - acc: 0.6736
Epoch 35/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6557 - acc: 0.6719
Epoch 36/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6583 - acc: 0.6736
Epoch 37/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6508 - acc: 0.6875
Epoch 38/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6610 - acc: 0.6806
Epoch 39/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6544 - acc: 0.6719
Epoch 40/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6655 - acc: 0.6701
Epoch 41/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6475 - acc: 0.6684
Epoch 42/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6432 - acc: 0.6875
Epoch 43/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6374 - acc: 0.6823
Epoch 44/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6463 - acc: 0.6753
Epoch 45/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6421 - acc: 0.6719
Epoch 46/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6289 - acc: 0.6875
Epoch 47/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6346 - acc: 0.6788
```

```
Epoch 57/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6182 - acc: 0.6788
Epoch 58/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6156 - acc: 0.6806
Epoch 59/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6294 - acc: 0.6719
Epoch 60/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6263 - acc: 0.6823
Epoch 61/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6125 - acc: 0.6875
Epoch 62/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6119 - acc: 0.6823
Epoch 63/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6249 - acc: 0.6858
Epoch 64/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6332 - acc: 0.6823
Epoch 65/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6239 - acc: 0.6823
Epoch 66/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6130 - acc: 0.6892
Epoch 67/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6070 - acc: 0.6875
Epoch 68/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6088 - acc: 0.6823
Epoch 69/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6039 - acc: 0.6910
Epoch 70/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6037 - acc: 0.6840
Epoch 71/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6070 - acc: 0.6806
Epoch 72/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6060 - acc: 0.6997
Epoch 73/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6037 - acc: 0.6892
Epoch 74/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6015 - acc: 0.6997
Epoch 75/100
```

```
18/18 [==============================] - 0s 3ms/step - loss: 0.6077 - acc: 0.6910
Epoch 93/100
18/18 [==============================] - 0s 3ms/step - loss: 0.5835 - acc: 0.7031
Epoch 94/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5887 - acc: 0.6997
Epoch 95/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6021 - acc: 0.7049
Epoch 96/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5998 - acc: 0.6927
Epoch 97/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5909 - acc: 0.6962
Epoch 98/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5842 - acc: 0.7031
Epoch 99/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5827 - acc: 0.7031
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5801 - acc: 0.7066
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
Trainable params: 269
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 4ms/step - loss: 0.6694 - acc: 0.6198
[0.669421911239624, 0.6197916865348816]
```

[5]
```
#read the data
data = pd.read_csv('sample_data/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.core 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: 19.7370 - acc: 0.6127
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 6.9501 - acc: 0.4812
Epoch 3/100
14/14 [==============================] - 0s 3ms/step - loss: 1.7348 - acc: 0.7934
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 1.1667 - acc: 0.7911
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.8424 - acc: 0.8521
Epoch 6/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7379 - acc: 0.8568
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6532 - acc: 0.8709
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6465 - acc: 0.8638
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5955 - acc: 0.8850
Epoch 10/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5704 - acc: 0.8685
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5624 - acc: 0.8756
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5204 - acc: 0.8803
Epoch 13/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4558 - acc: 0.8944
Epoch 14/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4492 - acc: 0.8803
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4601 - acc: 0.8897
Epoch 16/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4472 - acc: 0.8873
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4168 - acc: 0.8944
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4725 - acc: 0.8850
```

```
Epoch 30/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3543 - acc: 0.8991
Epoch 31/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3613 - acc: 0.9038
Epoch 32/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2681 - acc: 0.9131
Epoch 33/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2326 - acc: 0.9319
Epoch 34/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2121 - acc: 0.9249
Epoch 35/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2267 - acc: 0.9272
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2149 - acc: 0.9155
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2018 - acc: 0.9249
Epoch 38/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1895 - acc: 0.9272
Epoch 39/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1946 - acc: 0.9319
Epoch 40/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1964 - acc: 0.9225
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1777 - acc: 0.9296
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1775 - acc: 0.9249
Epoch 43/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1682 - acc: 0.9272
Epoch 44/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2546 - acc: 0.9108
Epoch 45/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1512 - acc: 0.9484
Epoch 46/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2322 - acc: 0.9108
Epoch 47/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1911 - acc: 0.9249
```

```
Epoch 62/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1544 - acc: 0.9343
Epoch 63/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1619 - acc: 0.9366
Epoch 64/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1846 - acc: 0.9272
Epoch 65/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1698 - acc: 0.9484
Epoch 66/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2087 - acc: 0.9413
Epoch 67/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2337 - acc: 0.9366
Epoch 68/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2058 - acc: 0.9249
Epoch 69/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1748 - acc: 0.9366
Epoch 70/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1455 - acc: 0.9437
Epoch 71/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1383 - acc: 0.9413
Epoch 72/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1682 - acc: 0.9437
Epoch 73/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1453 - acc: 0.9484
Epoch 74/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1677 - acc: 0.9366
Epoch 75/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1801 - acc: 0.9413
Epoch 76/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1579 - acc: 0.9437
Epoch 77/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1611 - acc: 0.9390
Epoch 78/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1725 - acc: 0.9366
Epoch 79/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1337 - acc: 0.9531
```

```
14/14 [==============================] - 0s 2ms/step - loss: 0.1635 - acc: 0.9460
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1418 - acc: 0.9437
Epoch 93/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1782 - acc: 0.9272
Epoch 94/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1313 - acc: 0.9437
Epoch 95/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2540 - acc: 0.9131
Epoch 96/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2844 - acc: 0.9085
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3514 - acc: 0.9014
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2980 - acc: 0.9085
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2324 - acc: 0.9343
Epoch 100/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2574 - acc: 0.9155
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                620

 dense_4 (Dense)             (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 5ms/step - loss: 0.2828 - acc: 0.9091
[0.2827788293361664, 0.9090909361839294]
```

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

```
path_to_csv = 'sample_data/breastcancer.csv'
```

+ Code + Text

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

[11]
```
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core 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: 20.3918 - acc: 0.6901
Epoch 2/100
14/14 [==============================] - 0s 2ms/step - loss: 2.5206 - acc: 0.7089
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 1.3414 - acc: 0.8638
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 1.1539 - acc: 0.9178
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.8934 - acc: 0.8826
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 0.8359 - acc: 0.9155
Epoch 7/100
14/14 [==============================] - 0s 3ms/step - loss: 0.8043 - acc: 0.8944
Epoch 8/100
14/14 [==============================] - 0s 3ms/step - loss: 0.7640 - acc: 0.8920
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.7307 - acc: 0.9131
Epoch 10/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6863 - acc: 0.8991
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6604 - acc: 0.9131
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6406 - acc: 0.9085
Epoch 13/100
14/14 [==============================] - 0s 3ms/step - loss: 0.6393 - acc: 0.9038
Epoch 14/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6232 - acc: 0.8850
Epoch 15/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5970 - acc: 0.9202
Epoch 16/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5695 - acc: 0.9085
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5701 - acc: 0.9225
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5587 - acc: 0.9249
```

```
Epoch 36/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5177 - acc: 0.9131
Epoch 37/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5723 - acc: 0.9155
Epoch 38/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4906 - acc: 0.9155
Epoch 39/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4728 - acc: 0.9178
Epoch 40/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4706 - acc: 0.9296
Epoch 41/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4635 - acc: 0.9108
Epoch 42/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5159 - acc: 0.9061
Epoch 43/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4764 - acc: 0.9225
Epoch 44/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4825 - acc: 0.9108
Epoch 45/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5405 - acc: 0.9202
Epoch 46/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5594 - acc: 0.9085
Epoch 47/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5211 - acc: 0.8991
Epoch 48/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4974 - acc: 0.8920
Epoch 49/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4434 - acc: 0.9108
Epoch 50/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4792 - acc: 0.9038
Epoch 51/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4748 - acc: 0.9131
Epoch 52/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4350 - acc: 0.9131
Epoch 53/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5195 - acc: 0.9038
Epoch 54/100
```

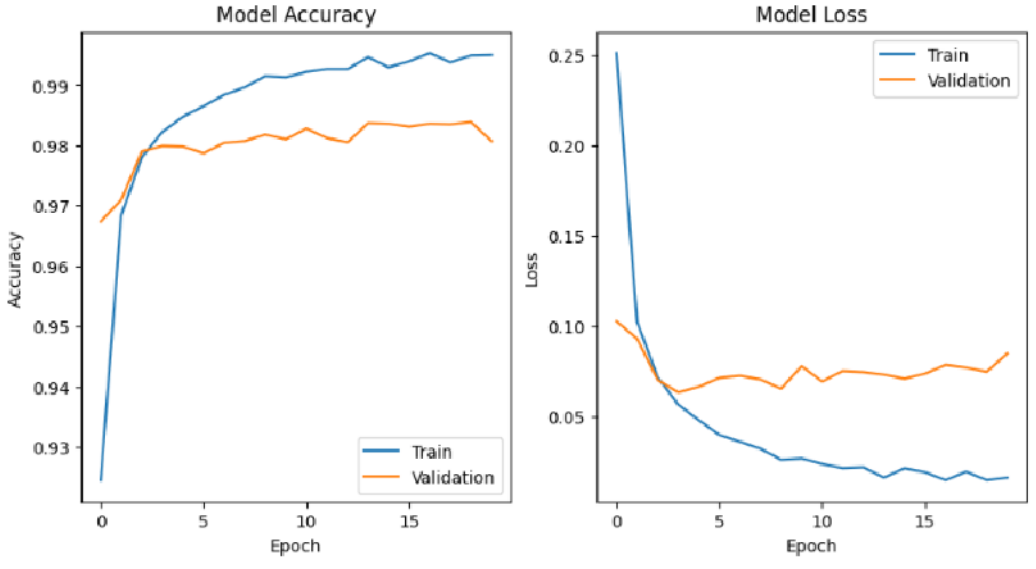
```
Epoch 64/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3612 - acc: 0.9272
Epoch 65/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4328 - acc: 0.9202
Epoch 66/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3791 - acc: 0.9108
Epoch 67/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5151 - acc: 0.9131
Epoch 68/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4648 - acc: 0.9155
Epoch 69/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4280 - acc: 0.9085
Epoch 70/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3856 - acc: 0.9202
Epoch 71/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3958 - acc: 0.9085
Epoch 72/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3577 - acc: 0.9131
Epoch 73/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3580 - acc: 0.9296
Epoch 74/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3333 - acc: 0.9272
Epoch 75/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3360 - acc: 0.9178
Epoch 76/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3394 - acc: 0.9178
Epoch 77/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3310 - acc: 0.9225
Epoch 78/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3434 - acc: 0.9155
Epoch 79/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3955 - acc: 0.9061
Epoch 80/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3916 - acc: 0.9108
Epoch 81/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3725 - acc: 0.9108
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.3157 - acc: 0.9202
Epoch 92/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2924 - acc: 0.9319
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3270 - acc: 0.9155
Epoch 94/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2955 - acc: 0.9249
Epoch 95/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2711 - acc: 0.9319
Epoch 96/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2604 - acc: 0.9343
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2682 - acc: 0.9249
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2367 - acc: 0.9272
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2796 - acc: 0.9272
Epoch 100/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2279 - acc: 0.9413
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 20)                620

 dense_6 (Dense)             (None, 1)                 21

=================================================================
Total params: 641
Trainable params: 641
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 2ms/step - loss: 0.6030 - acc: 0.8601
[0.6029807925224304, 0.8601398468017578]
```

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
i
#  ndarray: x_train
   ndarray with shape (60000, 28, 28)
(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)
```

```python
# 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 [==============================] - 1s 0us/step
Epoch 1/20
469/469 [==============================] - 13s 24ms/step - loss: 0.2512 - accuracy: 0.9245 - val_loss: 0.1025 - val_accuracy: 0.9674
Epoch 2/20
469/469 [==============================] - 11s 24ms/step - loss: 0.1022 - accuracy: 0.9685 - val_loss: 0.0928 - val_accuracy: 0.9710
Epoch 3/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0716 - accuracy: 0.9780 - val_loss: 0.0700 - val_accuracy: 0.9790
Epoch 4/20
469/469 [==============================] - 18s 38ms/step - loss: 0.0563 - accuracy: 0.9822 - val_loss: 0.0631 - val_accuracy: 0.9799
Epoch 5/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0476 - accuracy: 0.9847 - val_loss: 0.0662 - val_accuracy: 0.9798
Epoch 6/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0395 - accuracy: 0.9865 - val_loss: 0.0711 - val_accuracy: 0.9787
Epoch 7/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0357 - accuracy: 0.9884 - val_loss: 0.0722 - val_accuracy: 0.9804
Epoch 8/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0318 - accuracy: 0.9896 - val_loss: 0.0702 - val_accuracy: 0.9807
Epoch 9/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0257 - accuracy: 0.9915 - val_loss: 0.0651 - val_accuracy: 0.9818
Epoch 10/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0265 - accuracy: 0.9912 - val_loss: 0.0776 - val_accuracy: 0.9810
Epoch 11/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0234 - accuracy: 0.9922 - val_loss: 0.0691 - val_accuracy: 0.9828
Epoch 12/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0211 - accuracy: 0.9926 - val_loss: 0.0750 - val_accuracy: 0.9812
Epoch 13/20
469/469 [==============================] - 14s 31ms/step - loss: 0.0216 - accuracy: 0.9926 - val_loss: 0.0743 - val_accuracy: 0.9805
Epoch 14/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0158 - accuracy: 0.9946 - val_loss: 0.0729 - val_accuracy: 0.9837
Epoch 15/20
469/469 [==============================] - 12s 25ms/step - loss: 0.0210 - accuracy: 0.9930 - val_loss: 0.0706 - val_accuracy: 0.9836
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0189 - accuracy: 0.9940 - val_loss: 0.0734 - val_accuracy: 0.9831
Epoch 17/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0144 - accuracy: 0.9953 - val_loss: 0.0784 - val_accuracy: 0.9836
Epoch 18/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0191 - accuracy: 0.9938 - val_loss: 0.0769 - val_accuracy: 0.9835
Epoch 19/20
469/469 [==============================] - 10s 22ms/step - loss: 0.0144 - accuracy: 0.9949 - val_loss: 0.0745 - val_accuracy: 0.9839
Epoch 20/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0158 - accuracy: 0.9950 - val_loss: 0.0846 - val_accuracy: 0.9807
```

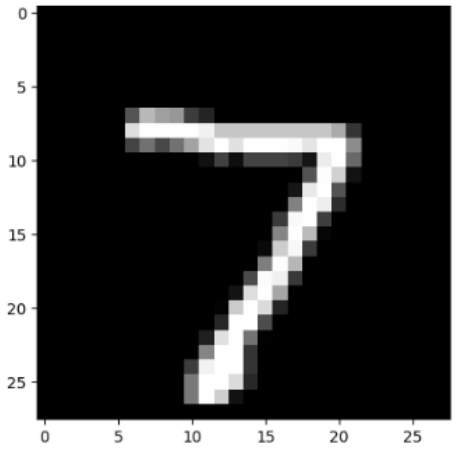


```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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'])
```

```python
# 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 [==============================] - 11s 21ms/step - loss: 0.2466 - accuracy: 0.9268 - val_loss: 0.1033 - val_accuracy: 0.9685
Epoch 2/20
469/469 [==============================] - 16s 33ms/step - loss: 0.0982 - accuracy: 0.9698 - val_loss: 0.0917 - val_accuracy: 0.9724
Epoch 3/20
469/469 [==============================] - 14s 30ms/step - loss: 0.0718 - accuracy: 0.9778 - val_loss: 0.0675 - val_accuracy: 0.9780
Epoch 4/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0550 - accuracy: 0.9829 - val_loss: 0.0700 - val_accuracy: 0.9780
Epoch 5/20
469/469 [==============================] - 12s 26ms/step - loss: 0.0447 - accuracy: 0.9854 - val_loss: 0.0727 - val_accuracy: 0.9804
Epoch 6/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0388 - accuracy: 0.9872 - val_loss: 0.0585 - val_accuracy: 0.9822
Epoch 7/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0331 - accuracy: 0.9888 - val_loss: 0.0795 - val_accuracy: 0.9770
Epoch 8/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0309 - accuracy: 0.9894 - val_loss: 0.0683 - val_accuracy: 0.9817
Epoch 9/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0279 - accuracy: 0.9904 - val_loss: 0.0656 - val_accuracy: 0.9830
Epoch 10/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0245 - accuracy: 0.9919 - val_loss: 0.0652 - val_accuracy: 0.9825
Epoch 11/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0198 - accuracy: 0.9937 - val_loss: 0.0654 - val_accuracy: 0.9849
```

```
Epoch 17/20
469/469 [==============================] - 10s 21ms/step - loss: 0.0169 - accuracy: 0.9941 - val_loss: 0.0770 - val_accuracy: 0.9838
Epoch 18/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0171 - accuracy: 0.9945 - val_loss: 0.0712 - val_accuracy: 0.9829
Epoch 19/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0142 - accuracy: 0.9950 - val_loss: 0.0777 - val_accuracy: 0.9839
Epoch 20/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0166 - accuracy: 0.9946 - val_loss: 0.0727 - val_accuracy: 0.9855
```

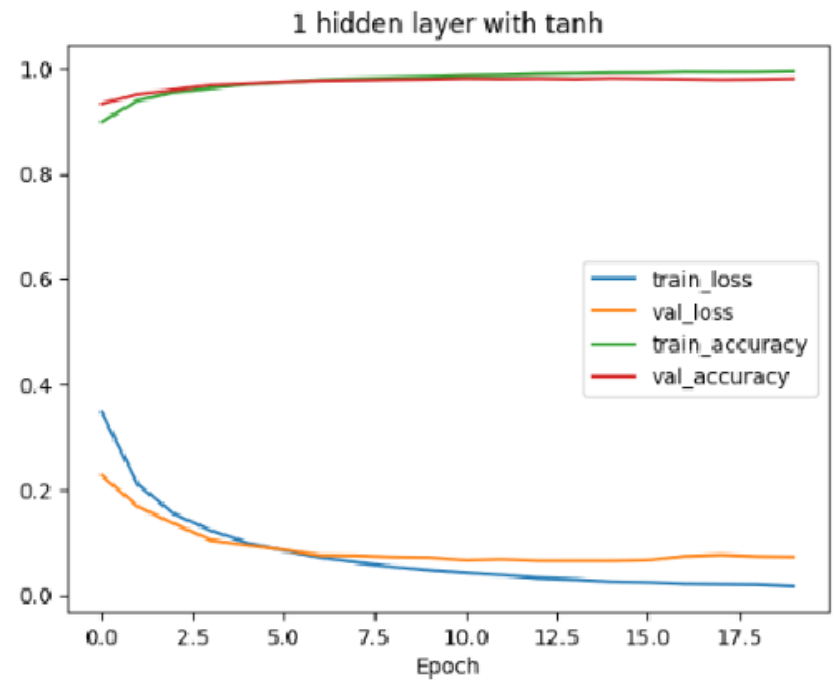



```
1/1 [==============================] - 0s 91ms/step
Model prediction: 7
```

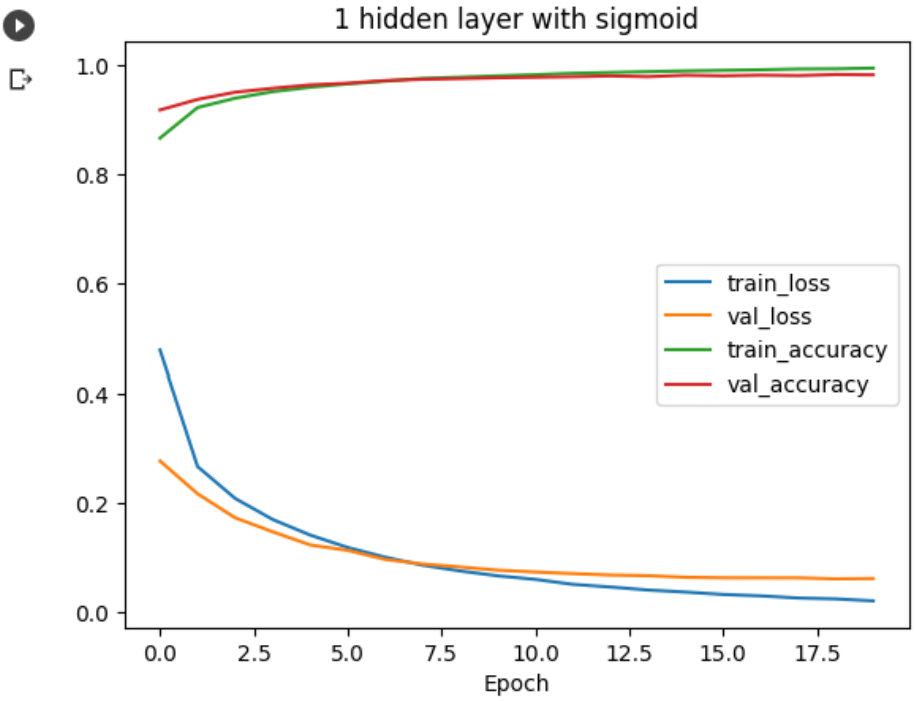
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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')
```

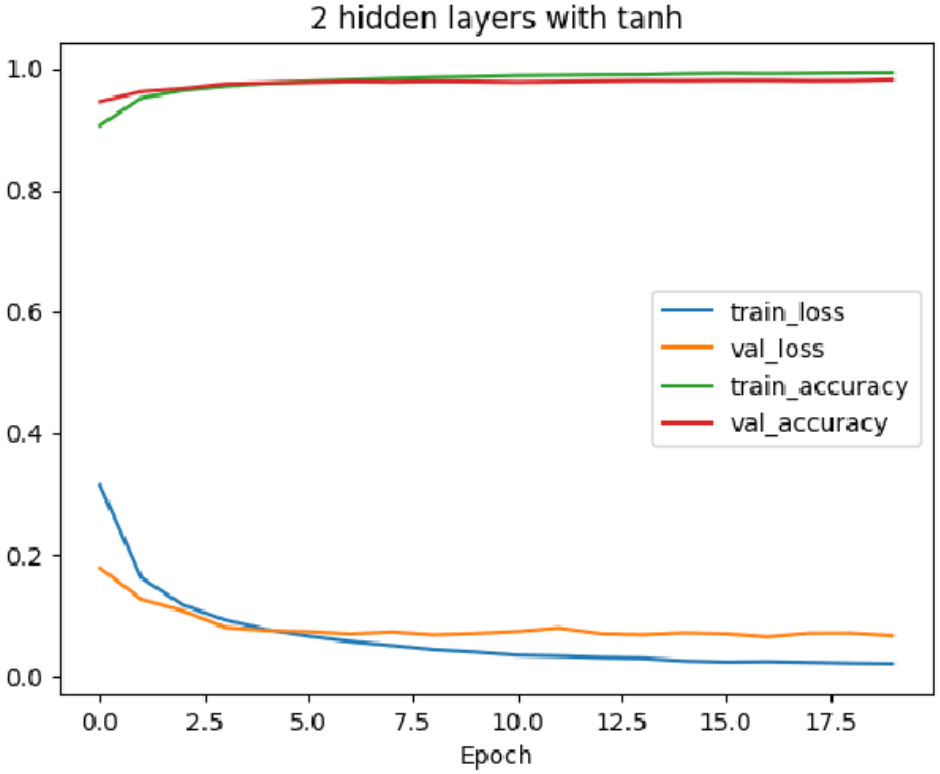
```
    plt.legend()
    plt.show()

    # 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))
```



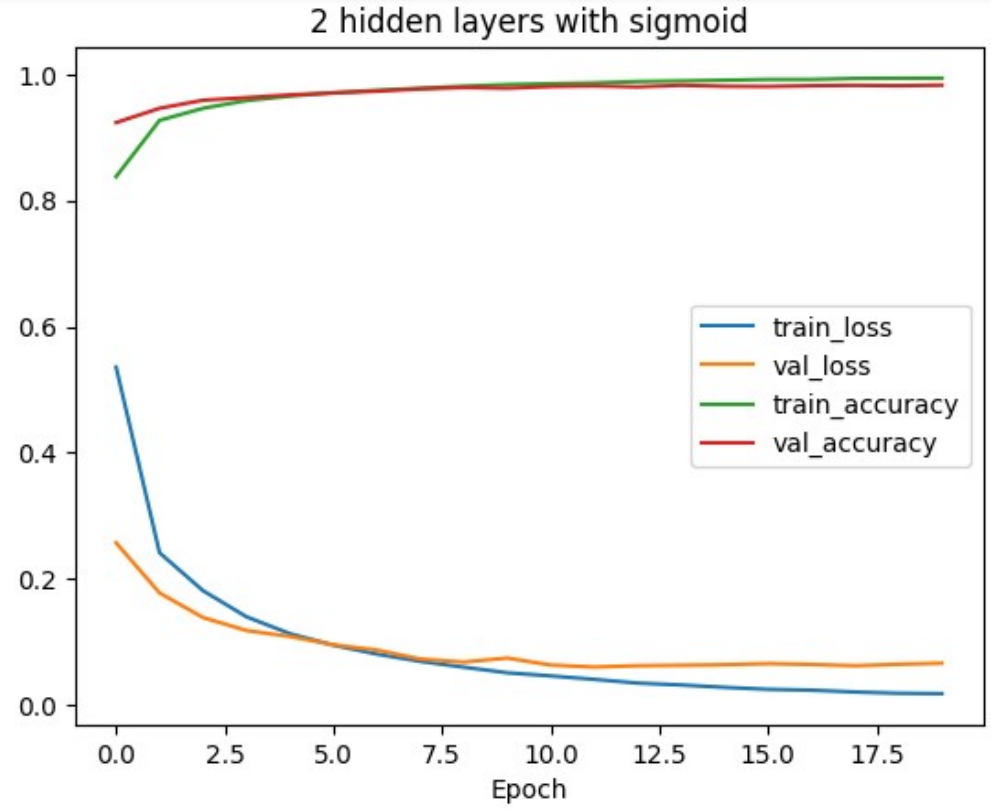

```
1 hidden layer with tanh - Test loss: 0.0723, Test accuracy: 0.9805
```




```
1 hidden layer with sigmoid - Test loss: 0.0621, Test accuracy: 0.9819
```




```
2 hidden layers with tanh - Test loss: 0.0660, Test accuracy: 0.9819
```

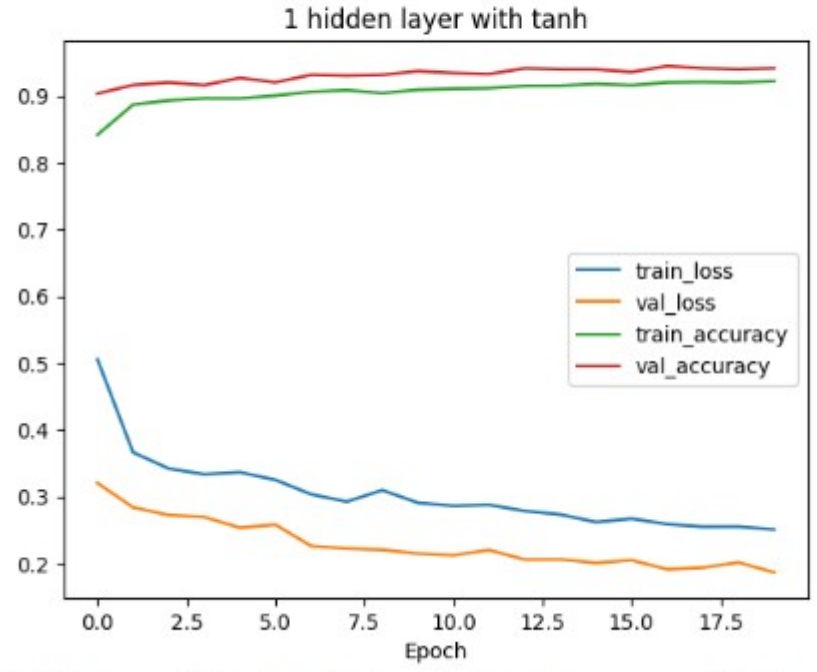



```
2 hidden layers with sigmoid - Test loss: 0.0664, Test accuracy: 0.9827
```

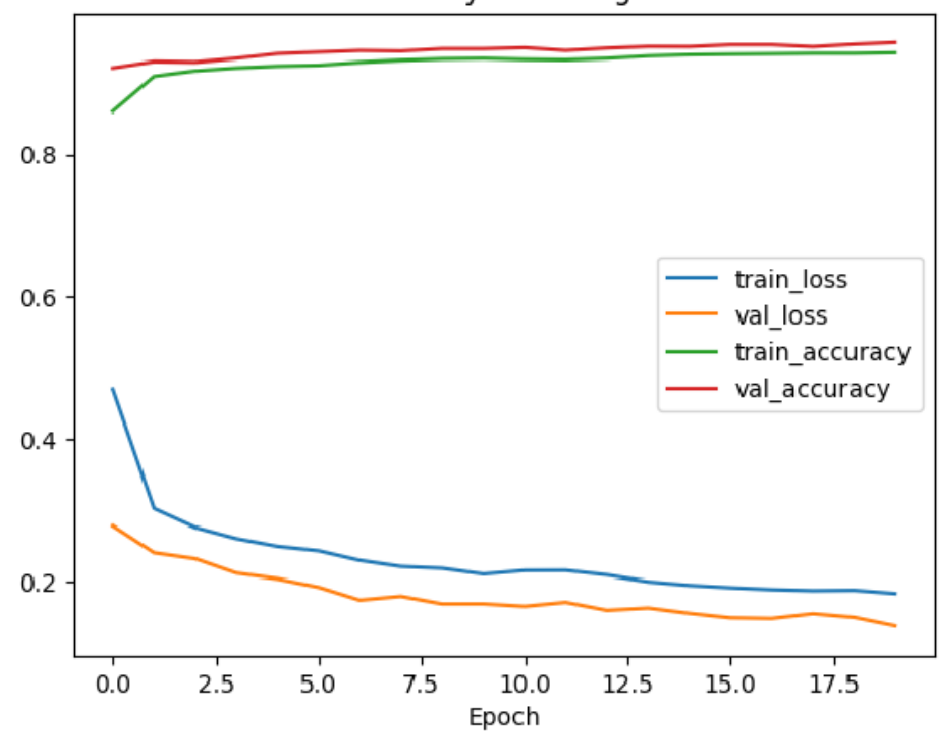
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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))
```

```python
# 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()
```

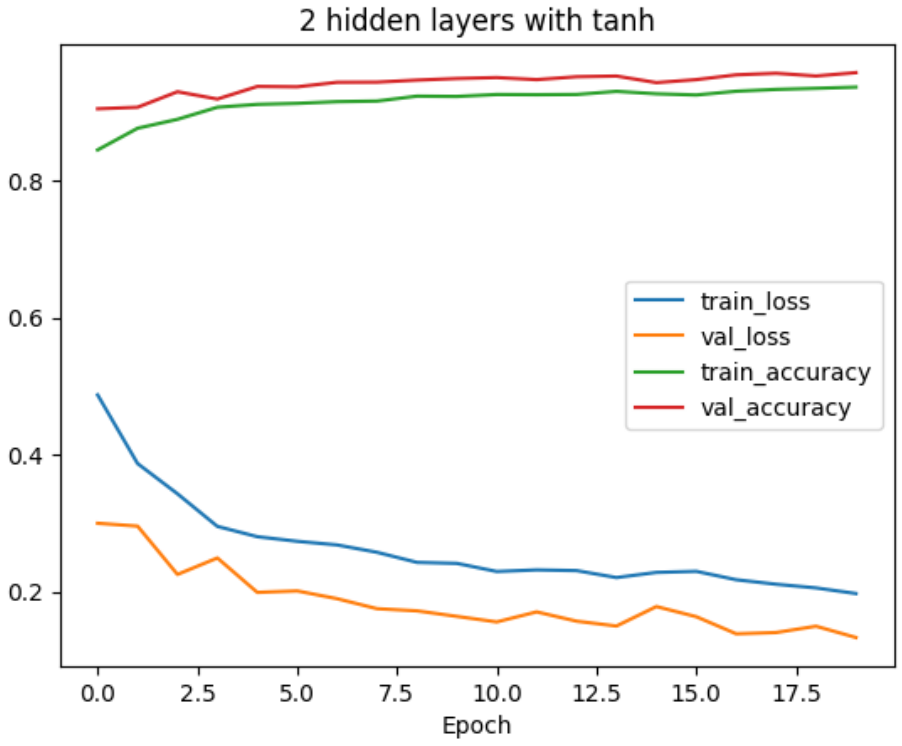
```python
    # evaluate the model on test data
    loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```



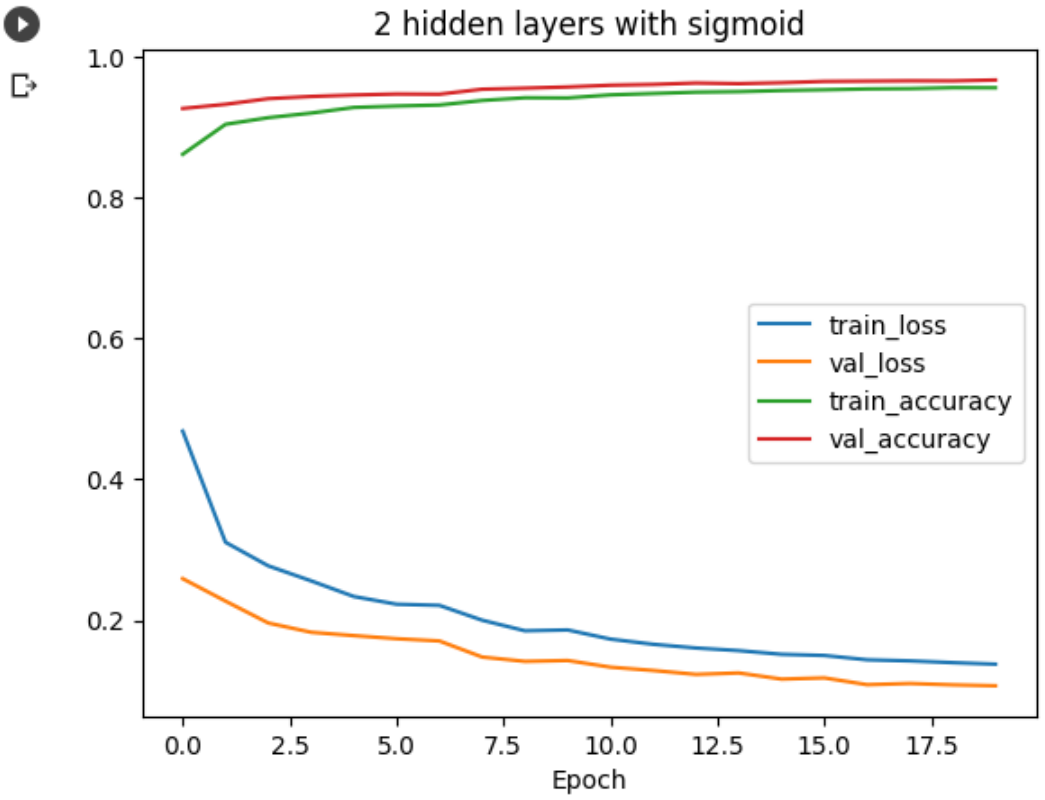

```
1 hidden layer with tanh - Test loss: 0.1869, Test accuracy: 0.9415
```




```
1 hidden layer with sigmoid - Test loss: 0.1361, Test accuracy: 0.9584
```




```
2 hidden layers with tanh - Test loss: 0.1327, Test accuracy: 0.9582
```




```
2 hidden layers with sigmoid - Test loss: 0.1072, Test accuracy: 0.9667
```