```
NEURAL NETWORKS ASSIGNMENT2

NAME: Saharsha Muddagauni

Std ID: 700742679
```

In [ ]:

```
Git hub link : https://github.com/sxm26790/Assignment2-
```

Problem 1:

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

In [2]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [17]:

```
path_to_csv = '/content/gdrive/My Drive/diabetes.csv'
```

In [18]:

```
import keras
import pandas
from keras.models import Sequential
from keras.datasets import mnist
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], da
                                                     test_size=0.25, ra
np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden la
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', metr
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 67/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6244 -
acc: 0.6840
Epoch 68/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6256 -
acc: 0.6788
Epoch 69/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6228 -
acc: 0.6858
Epoch 70/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6217 -
acc: 0.6840
Epoch 71/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6215 -
acc: 0.6858
Epoch 72/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6215 -
acc: 0.6892
Epoch 73/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6216 -
acc: 0.6858
```

Question 2(problem 1):

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

In [29]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remo
unt, call drive.mount("/content/gdrive", force_remount=True).
```

In [38]:

```
path_to_csv = '/content/gdrive/My Drive/breastcancer.csv'
```

In [40]:

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

# Load dataset

cancer_dataset = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], da
                                                     test_size=0.25, ra

np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden la
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metr
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: 41.4774
- acc: 0.6615
Epoch 2/100
18/18 [==============================] - 0s 2ms/step - loss: 30.9226
- acc: 0.6615
Epoch 3/100
18/18 [==============================] - 0s 2ms/step - loss: 20.9840
- acc: 0.6615
Epoch 4/100
18/18 [==============================] - 0s 2ms/step - loss: 11.4362
- acc: 0.6562
Epoch 5/100
18/18 [==============================] - 0s 2ms/step - loss: 3.8298 -
acc: 0.6562
Epoch 6/100
18/18 [==============================] - 0s 2ms/step - loss: 2.5091 -
acc: 0.6111
Epoch 7/100
18/18 [==============================] - 0s 2ms/step - loss: 2.1508
```

Question 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()

In [46]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remo
unt, call drive.mount("/content/gdrive", force_remount=True).
```

In [43]:

```
path_to_csv = '/content/gdrive/My Drive/breastcancer.csv'
```

In [44]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [45]:
```
import keras
import pandas
from keras.models import Sequential
from keras.datasets import mnist
from keras.layers.core import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

# Load dataset

cancer_dataset = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], da
                                                     test_size=0.25, ra

np.random.seed(155)
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden la
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metr
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))
```

```
18/18 [==============================] - 0s 2ms/step - loss: 0.6311 -
acc: 0.7188
Epoch 67/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6324 -
acc: 0.7031
Epoch 68/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6241 -
acc: 0.7240
Epoch 69/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6376 -
acc: 0.6997
Epoch 70/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6167 -
acc: 0.7205
Epoch 71/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6356 -
acc: 0.7101
Epoch 72/100
18/18 [==============================] - 0s 2ms/step - loss: 0.6204 -
acc: 0.6944
Epoch 73/100
```

Problem 2:

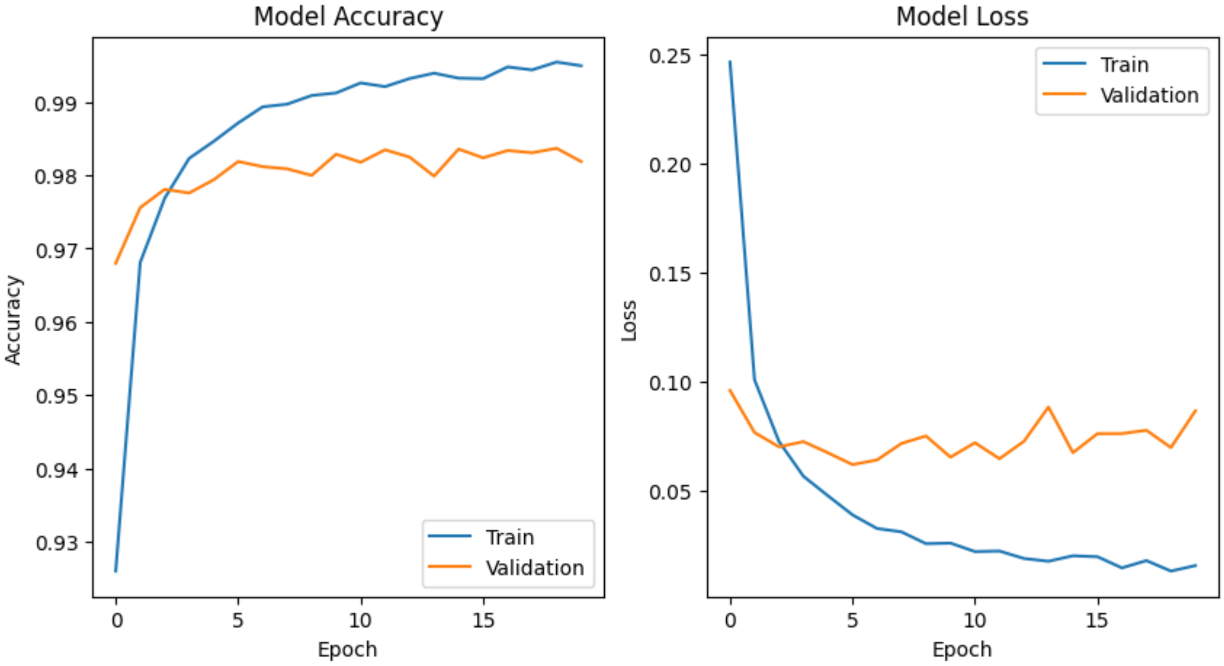
Plot the loss and accuracy for both training data and validation data using the history object in the source code.

In [47]:

```
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', metri

# train the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data
                    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-ke
ras-datasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz)
11490434/11490434 [==============================] - 0s 0us/step
Epoch 1/20
469/469 [==============================] - 15s 29ms/step - loss: 0.24
64 - accuracy: 0.9259 - val_loss: 0.0961 - val_accuracy: 0.9680
Epoch 2/20
469/469 [==============================] - 10s 21ms/step - loss: 0.10
09 - accuracy: 0.9681 - val_loss: 0.0768 - val_accuracy: 0.9756
Epoch 3/20
469/469 [==============================] - 12s 25ms/step - loss: 0.07
27 - accuracy: 0.9769 - val_loss: 0.0702 - val_accuracy: 0.9781
Epoch 4/20
469/469 [==============================] - 12s 25ms/step - loss: 0.05
67 - accuracy: 0.9823 - val_loss: 0.0726 - val_accuracy: 0.9776
Epoch 5/20
469/469 [==============================] - 12s 26ms/step - loss: 0.04
78 - accuracy: 0.9847 - val_loss: 0.0675 - val_accuracy: 0.9794
Epoch 6/20
469/469 [==============================] - 11s 23ms/step - loss: 0.03
91 - accuracy: 0.9872 - val_loss: 0.0621 - val_accuracy: 0.9819
Epoch 7/20
469/469 [==============================] - 11s 24ms/step - loss: 0.03
29 - accuracy: 0.9894 - val_loss: 0.0642 - val_accuracy: 0.9812
Epoch 8/20
469/469 [==============================] - 13s 29ms/step - loss: 0.03
13 - accuracy: 0.9897 - val_loss: 0.0718 - val_accuracy: 0.9809
Epoch 9/20
469/469 [==============================] - 12s 26ms/step - loss: 0.02
59 - accuracy: 0.9909 - val_loss: 0.0752 - val_accuracy: 0.9800
Epoch 10/20
469/469 [==============================] - 12s 26ms/step - loss: 0.02
62 - accuracy: 0.9913 - val_loss: 0.0655 - val_accuracy: 0.9829
Epoch 11/20
469/469 [==============================] - 11s 23ms/step - loss: 0.02
23 - accuracy: 0.9926 - val_loss: 0.0721 - val_accuracy: 0.9818
Epoch 12/20
469/469 [==============================] - 11s 24ms/step - loss: 0.02
25 - accuracy: 0.9921 - val_loss: 0.0648 - val_accuracy: 0.9835
Epoch 13/20
469/469 [==============================] - 12s 26ms/step - loss: 0.01
```

```
91 - accuracy: 0.9932 - val_loss: 0.0728 - val_accuracy: 0.9825
Epoch 14/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
78 - accuracy: 0.9940 - val_loss: 0.0884 - val_accuracy: 0.9799
Epoch 15/20
469/469 [==============================] - 11s 24ms/step - loss: 0.02
04 - accuracy: 0.9933 - val_loss: 0.0676 - val_accuracy: 0.9836
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.01
99 - accuracy: 0.9932 - val_loss: 0.0763 - val_accuracy: 0.9824
Epoch 17/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
48 - accuracy: 0.9948 - val_loss: 0.0763 - val_accuracy: 0.9834
Epoch 18/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
81 - accuracy: 0.9944 - val_loss: 0.0778 - val_accuracy: 0.9831
Epoch 19/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
34 - accuracy: 0.9955 - val_loss: 0.0699 - val_accuracy: 0.9837
Epoch 20/20
469/469 [==============================] - 10s 22ms/step - loss: 0.01
59 - accuracy: 0.9950 - val_loss: 0.0868 - val_accuracy: 0.9819
```



Question 2:

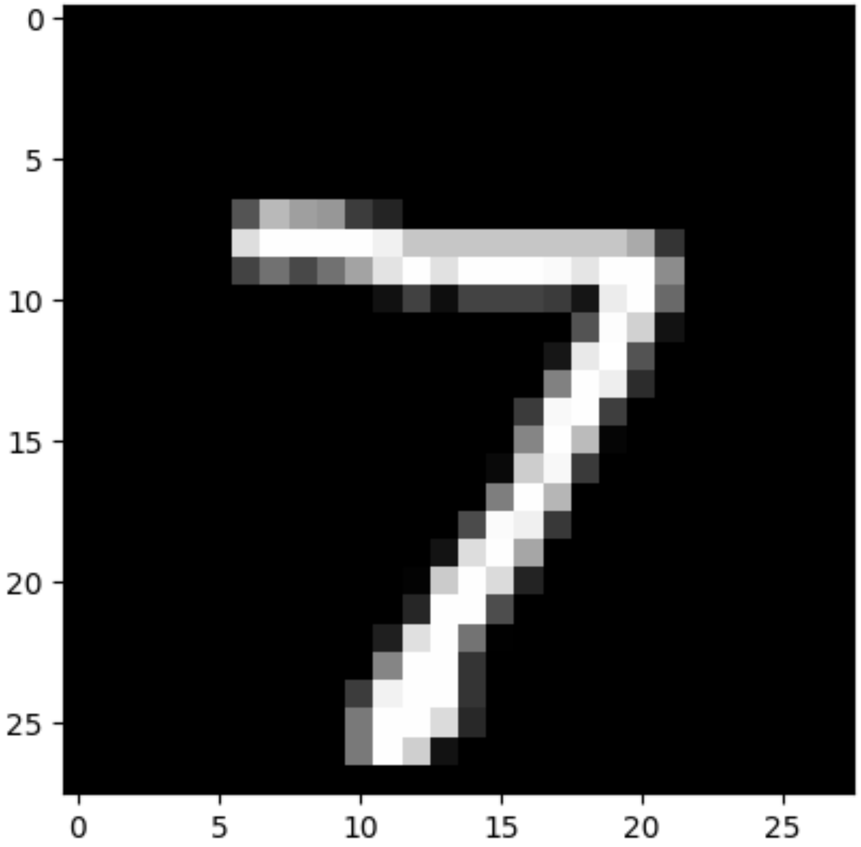
Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.

In [50]:

```
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', metri

# train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.r
          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 [==============================] - 13s 25ms/step - loss: 0.24
72 - accuracy: 0.9254 - val_loss: 0.1070 - val_accuracy: 0.9646
Epoch 2/20
469/469 [==============================] - 12s 25ms/step - loss: 0.09
92 - accuracy: 0.9696 - val_loss: 0.0854 - val_accuracy: 0.9740
Epoch 3/20
Epoch 3/20
469/469 [==============================] - 11s 23ms/step - loss: 0.06
93 - accuracy: 0.9774 - val_loss: 0.0677 - val_accuracy: 0.9794
Epoch 4/20
469/469 [==============================] - 11s 23ms/step - loss: 0.05
62 - accuracy: 0.9821 - val_loss: 0.0560 - val_accuracy: 0.9827
Epoch 5/20
469/469 [==============================] - 12s 25ms/step - loss: 0.04
55 - accuracy: 0.9857 - val_loss: 0.0655 - val_accuracy: 0.9799
Epoch 6/20
469/469 [==============================] - 12s 25ms/step - loss: 0.03
92 - accuracy: 0.9875 - val_loss: 0.0618 - val_accuracy: 0.9816
Epoch 7/20
469/469 [==============================] - 11s 23ms/step - loss: 0.03
46 - accuracy: 0.9887 - val_loss: 0.0660 - val_accuracy: 0.9827
Epoch 8/20
469/469 [==============================] - 11s 23ms/step - loss: 0.02
91 - accuracy: 0.9902 - val_loss: 0.0607 - val_accuracy: 0.9826
Epoch 9/20
469/469 [==============================] - 13s 27ms/step - loss: 0.02
66 - accuracy: 0.9910 - val_loss: 0.0733 - val_accuracy: 0.9811
Epoch 10/20
469/469 [==============================] - 12s 25ms/step - loss: 0.02
73 - accuracy: 0.9908 - val_loss: 0.0760 - val_accuracy: 0.9819
Epoch 11/20
469/469 [==============================] - 11s 24ms/step - loss: 0.02
40 - accuracy: 0.9919 - val_loss: 0.0748 - val_accuracy: 0.9814
Epoch 12/20
469/469 [==============================] - 10s 21ms/step - loss: 0.02
22 - accuracy: 0.9929 - val_loss: 0.0838 - val_accuracy: 0.9791
Epoch 13/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
85 - accuracy: 0.9937 - val_loss: 0.0716 - val_accuracy: 0.9821
Epoch 14/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
78 - accuracy: 0.9940 - val_loss: 0.0875 - val_accuracy: 0.9812
Epoch 15/20
469/469 [==============================] - 11s 23ms/step - loss: 0.02
16 - accuracy: 0.9928 - val_loss: 0.0667 - val_accuracy: 0.9829
Epoch 16/20
469/469 [==============================] - 11s 23ms/step - loss: 0.01
46 - accuracy: 0.9950 - val_loss: 0.0796 - val_accuracy: 0.9836
Epoch 17/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
58 - accuracy: 0.9951 - val_loss: 0.0758 - val_accuracy: 0.9842

Epoch 18/20
469/469 [==============================] - 12s 25ms/step - loss: 0.01
64 - accuracy: 0.9943 - val_loss: 0.0817 - val_accuracy: 0.9817
Epoch 19/20
469/469 [==============================] - 12s 26ms/step - loss: 0.01
62 - accuracy: 0.9944 - val_loss: 0.0766 - val_accuracy: 0.9835
```

```
62 - accuracy: 0.9944 - val_loss: 0.0766 - val_accuracy: 0.9835
Epoch 20/20
469/469 [==============================] - 19s 39ms/step - loss: 0.01
78 - accuracy: 0.9942 - val_loss: 0.0749 - val_accuracy: 0.9855
```

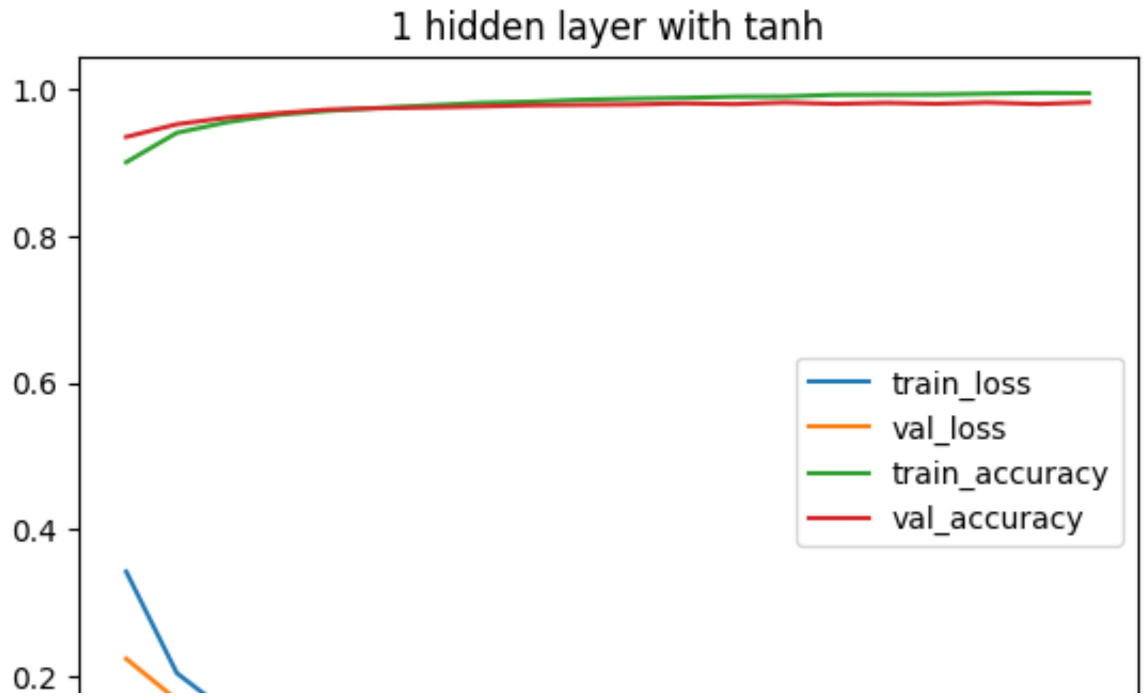


```
1/1 [==============================] - 0s 63ms/step
Model prediction: 7
```

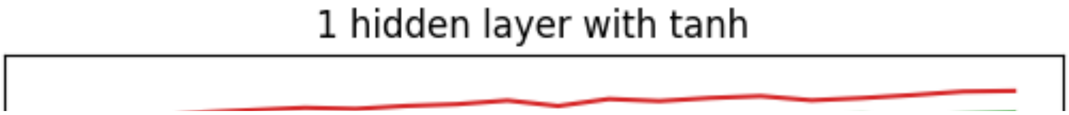
In [51]:

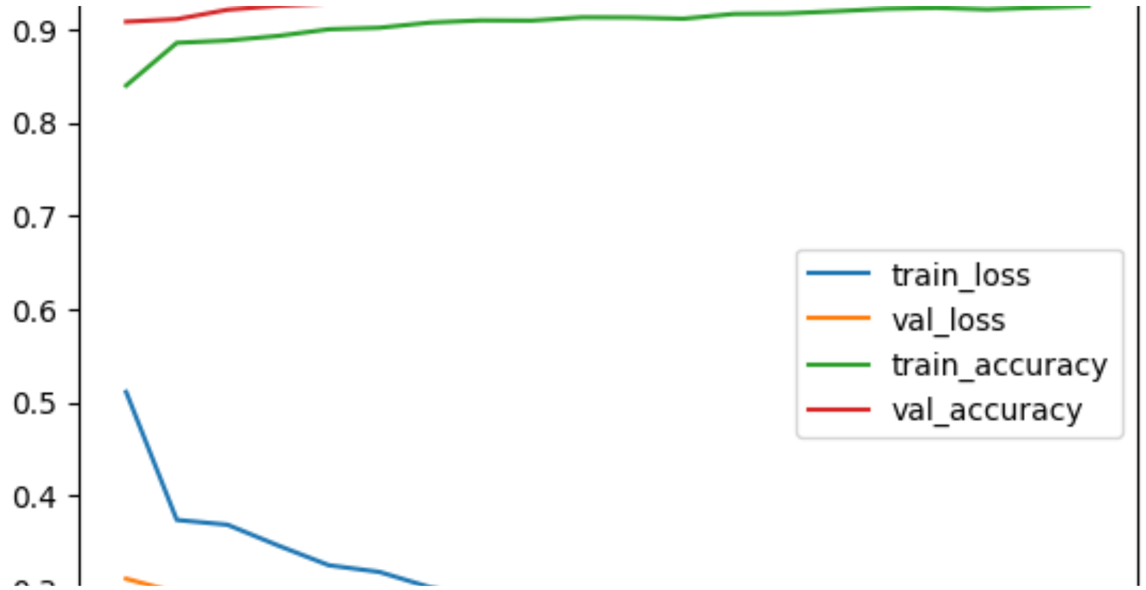
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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', m
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_
                        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, v
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name,
```



In [52]:
```
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))
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', m
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_
                        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, v
    print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name,
```




0.9
0.8
0.7
0.6
0.5
0.4
train_loss
val_loss
train_accuracy
val_accuracy