NEURAL NETWORKS & DEEP LEARNING: ICP2

Name: Reshma Maddala

ID: 700740808

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()

1]:
```
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
```

3]:
```
#read the data
data = pd.read_csv('sample_data/diabetes.csv')
path_to_csv = 'sample_data/diabetes.csv'
```

4]:
```
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 [==============================] - 6s 5ms/step - loss: 0.9374 - acc: 0.3802
Epoch 2/100
18/18 [==============================] - 0s 4ms/step - loss: 0.7141 - acc: 0.5399
Epoch 3/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6752 - acc: 0.6615
Epoch 4/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6619 - acc: 0.6632
Epoch 5/100
18/18 [==============================] - 0s 4ms/step - loss: 0.6557 - acc: 0.6632
Epoch 6/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6545 - acc: 0.6632
Epoch 7/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6482 - acc: 0.6632
Epoch 8/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6472 - acc: 0.6632
Epoch 9/100
18/18 [==============================] - 0s 3ms/step - loss: 0.6441 - acc: 0.6632
Epoch 10/100
```

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

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

```
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))
```

```
14/14 [==============================] - 0s 4ms/step - loss: 0.1872 - acc: 0.9225
Epoch 100/100
14/14 [==============================] - 0s 5ms/step - loss: 0.1853 - acc: 0.9366
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 3ms/step - loss: 0.2894 - acc: 0.8881
[0.28943607211112976, 0.8881118893623352]
```

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

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

```
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.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))
```

```
14/14 [==============================] - 0s 3ms/step - loss: 0.2531 - acc: 0.9366
Epoch 100/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2515 - acc: 0.9272
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 4ms/step - loss: 0.6826 - acc: 0.8811
[0.6825757622718811, 0.881118893623352]
```

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.

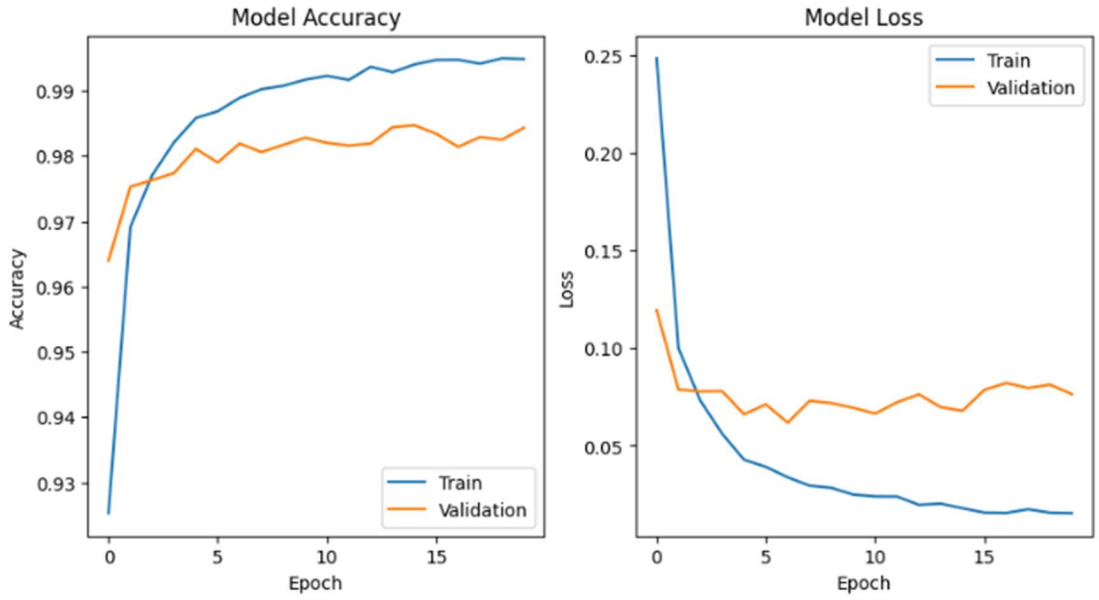
3. We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to tanh or sigmoid and see what happens.

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')
```

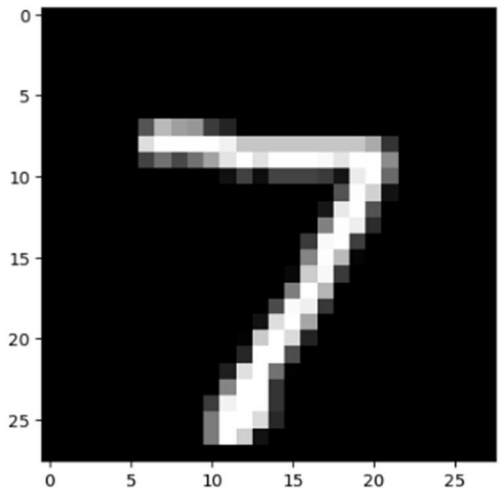
```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')

plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
Epoch 1/20
469/469 [==============================] - 4s 5ms/step - loss: 0.2483 - accuracy: 0.9253 - val_loss: 0.1195 - val_accuracy:
0.9640
Epoch 2/20
469/469 [==============================] - 2s 5ms/step - loss: 0.1000 - accuracy: 0.9691 - val_loss: 0.0789 - val_accuracy:
0.9753
Epoch 3/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0732 - accuracy: 0.9771 - val_loss: 0.0779 - val_accuracy:
0.9763
Epoch 4/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0562 - accuracy: 0.9821 - val_loss: 0.0780 - val_accuracy:
0.9774
Epoch 5/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0429 - accuracy: 0.9858 - val_loss: 0.0662 - val_accuracy:
0.9811
Epoch 6/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0392 - accuracy: 0.9869 - val_loss: 0.0712 - val_accuracy:
0.9790
Epoch 7/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0339 - accuracy: 0.9889 - val_loss: 0.0619 - val_accuracy:
0.9819
Epoch 8/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0296 - accuracy: 0.9902 - val_loss: 0.0731 - val_accuracy:
0.9806
Epoch 9/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0285 - accuracy: 0.9908 - val_loss: 0.0719 - val_accuracy:
0.9817
Epoch 10/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0251 - accuracy: 0.9917 - val_loss: 0.0695 - val_accuracy:
0.9828
Epoch 11/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0241 - accuracy: 0.9923 - val_loss: 0.0665 - val_accuracy:
0.9820
Epoch 12/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0241 - accuracy: 0.9916 - val_loss: 0.0723 - val_accuracy:
0.9816
Epoch 13/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0198 - accuracy: 0.9937 - val_loss: 0.0764 - val_accuracy:
0.9819
0.9845
```



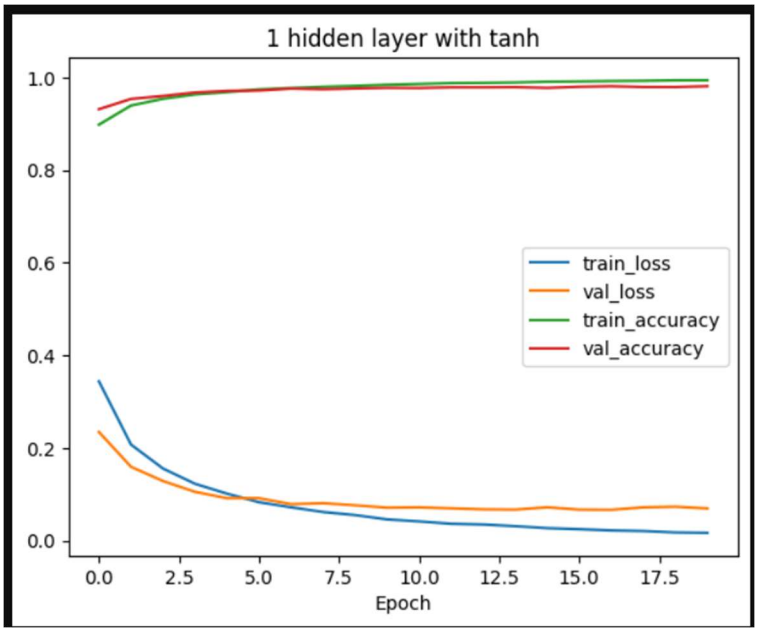
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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'])

# 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))
```

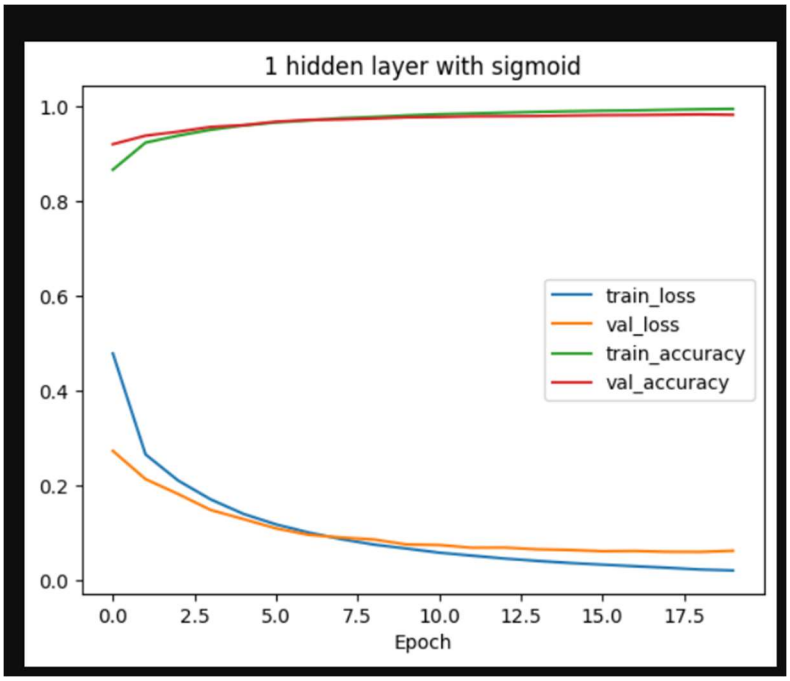
```
Epoch 13/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0196 - accuracy: 0.9934 - val_loss: 0.0750 - val_accuracy:
0.9811
Epoch 14/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0188 - accuracy: 0.9937 - val_loss: 0.0721 - val_accuracy:
0.9823
Epoch 15/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0186 - accuracy: 0.9936 - val_loss: 0.0770 - val_accuracy:
0.9814
Epoch 16/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0174 - accuracy: 0.9941 - val_loss: 0.0789 - val_accuracy:
0.9825
Epoch 17/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0169 - accuracy: 0.9944 - val_loss: 0.0997 - val_accuracy:
0.9797
Epoch 18/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0190 - accuracy: 0.9937 - val_loss: 0.0865 - val_accuracy:
0.9816
Epoch 19/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0169 - accuracy: 0.9943 - val_loss: 0.0769 - val_accuracy:
0.9824
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0157 - accuracy: 0.9948 - val_loss: 0.0775 - val_accuracy:
0.9826
```



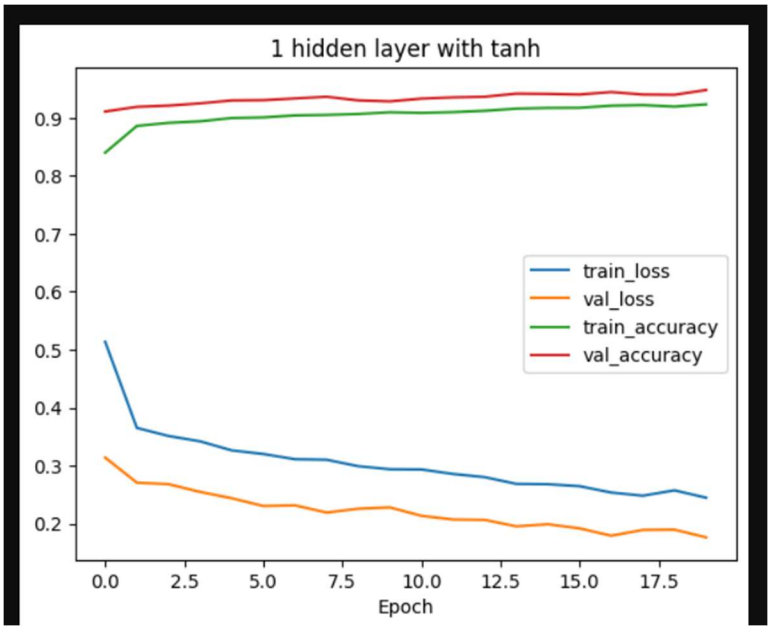

```
1/1 [==============================] - 0s 80ms/step
Model prediction: 7
```

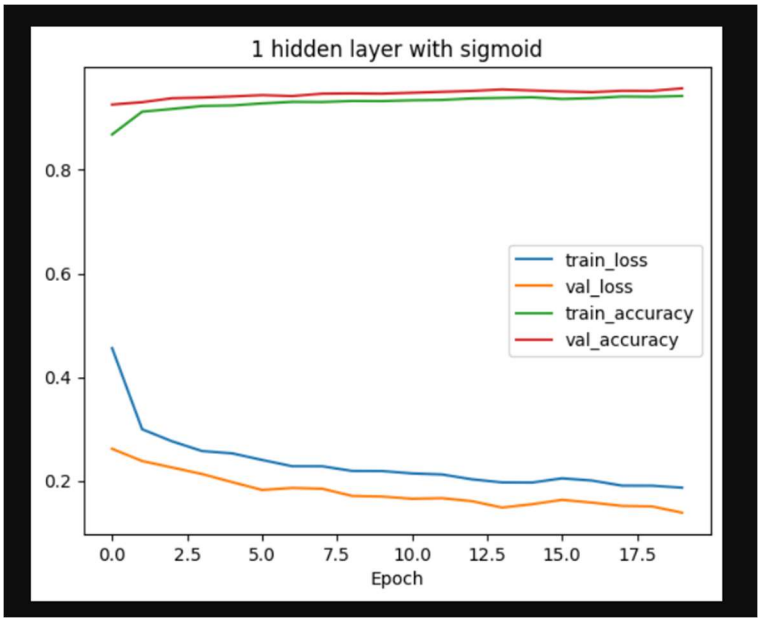
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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.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
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy


1 hidden layer with sigmoid
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch
train_loss
val_loss
train_accuracy
val_accuracy

```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))

# 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))
```





GITHUB REPO LINK: https://github.com/maddalareshma/NNDL-ICP2