Neural Networks & Deep Learning: ICP2

Name: Prathin Reddy Junnuthula
ID:700741496

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.

```
#read the data
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))
```

The given code implements a neural network using Keras to predict diabetes based on a dataset. The neural network consists of multiple Dense layers, and the code trains the model on the training data and evaluates its performance on the test data. The model's accuracy is recorded after training and testing.

To improve the model's accuracy, two additional Dense layers have been added to the neural network, increasing its complexity. After adding the new layers, the code trains the updated model on the training data and evaluates its performance on the test data again to see how the accuracy changes with the added layers.

Neural Networks & Deep Learning: ICP2

Name: Prathin Reddy Junnuthula
ID:700741496

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

```
#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))
```

The given code implements a neural network using Keras to classify breast cancer data. It uses the breast cancer dataset from sklearn, splits it into training and testing sets, and builds a neural network with two Dense layers (with 20 and 10 neurons, respectively) followed by an output Dense layer (with 1 neuron) using the ReLU activation function. The model is trained on the training data and evaluated on the test data to calculate the loss and accuracy.

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

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()

```
#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))
```

The given code implements a neural network using Keras to classify breast cancer data after preprocessing it with StandardScaler. It scales the input features, splits the data into training and testing sets, and builds a neural network with two Dense layers (with 20 and 10 neurons, respectively) followed by an output Dense layer (with 1 neuron) using the ReLU activation function for the hidden layers and the sigmoid activation function for the output layer. The model is trained on the training data and evaluated on the test data to calculate the loss and accuracy of the model in predicting breast cancer.

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

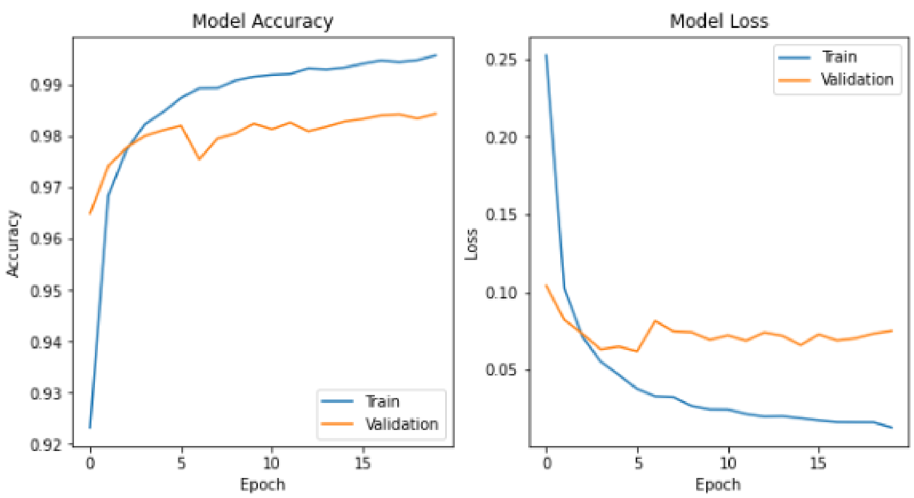
PROBLEM 2

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

```python
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()
```

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

OUTPUT



The code provided plots the model's training and testing accuracy as well as the training and testing loss over epochs. These plots help visualize the performance of the neural network during training and provide insights into potential overfitting or underfitting.

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.**

Neural Networks & Deep Learning: ICP2

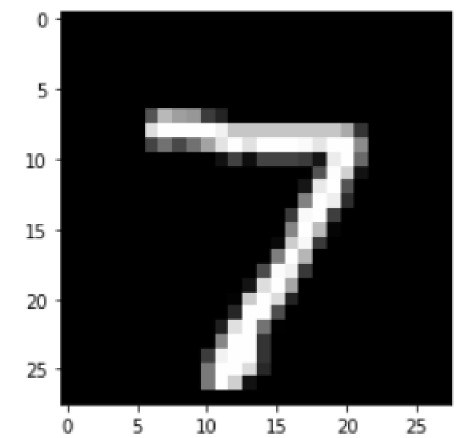
Name: Prathin Reddy Junnuthula
ID:700741496

```
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))
```

Neural Networks & Deep Learning: ICP2

Name: Prathin Reddy Junnuthula
ID:700741496

OUTPUT




```
1/1 [==============================] - 0s 120ms/step
Model prediction: 7
```

The code displays the first test image from the "test_images" dataset using a grayscale colormap. Then, it uses the trained neural network model to predict the label of the displayed image and prints the predicted class label.

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.**

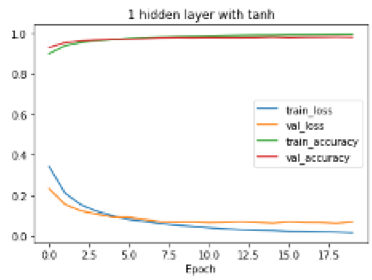
Neural Networks & Deep Learning: ICP2

Name: Prathin Reddy Junnuthula
ID:700741496

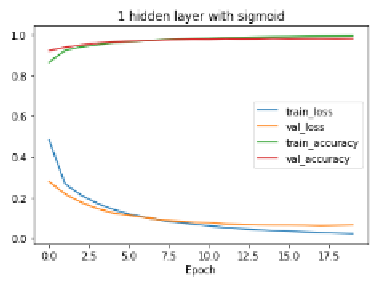
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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))
```

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

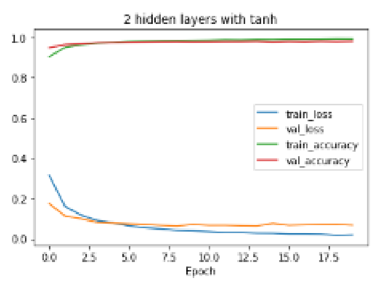
OUTPUT



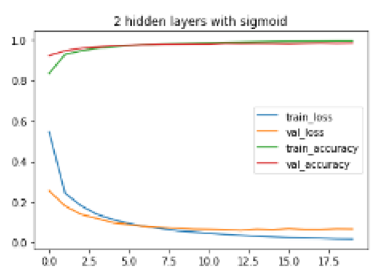

```
1 hidden layer with tanh - Test loss: 0.0716, Test accuracy: 0.9809
```




```
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
```




```
2 hidden layers with tanh - Test loss: 0.0686, Test accuracy: 0.9808
```




```
2 hidden layers with sigmoid - Test loss: 0.0663, Test accuracy: 0.9830
```

The provided code trains two neural network models with different activation functions (tanh and sigmoid) and the same architecture. The models have four Dense layers with decreasing neurons in each layer. The final Dense layer has 10 neurons with a softmax activation function for multi-class classification. The models are trained using RMSprop optimizer and categorical

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

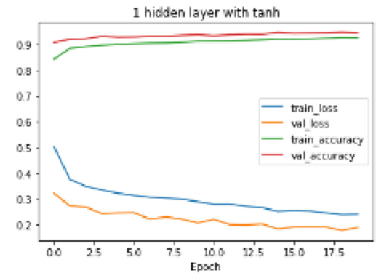
cross-entropy loss for 10 epochs, and the training progress is recorded in the "history" variable for each model.

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

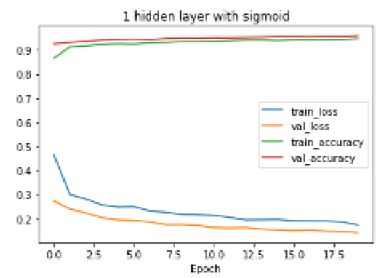
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# 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))
```

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

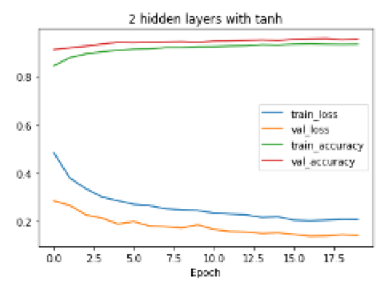
OUTPUT



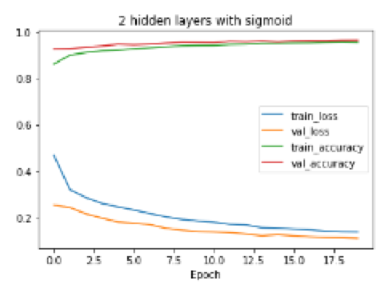

```
1 hidden layer with tanh - Test loss: 0.1895, Test accuracy: 0.9439
```




```
1 hidden layer with sigmoid - Test loss: 0.1420, Test accuracy: 0.9582
```




```
2 hidden layers with tanh - Test loss: 0.1422, Test accuracy: 0.9563
```




```
2 hidden layers with sigmoid - Test loss: 0.1095, Test accuracy: 0.9652
```

The provided code prepares the image data and their corresponding labels for training and testing. It then builds a neural network model with two hidden Dense layers, each having 512

Neural Networks & Deep Learning: ICP2
Name: Prathin Reddy Junnuthula
ID:700741496

neurons with the ReLU activation function, and an output Dense layer with 10 neurons and a softmax activation function for multi-class classification. The model is trained using RMSprop optimizer and categorical cross-entropy loss for 10 epochs. The training progress is recorded in the "history" variable.

GitHub URL: https://github.com/Prathin-offi/NNDL_ICP-2.git

Video URL:
https://drive.google.com/file/d/1K5AoDuUZcyQi49yXCymXc4mRTf2qs5dr/view?usp=sharing