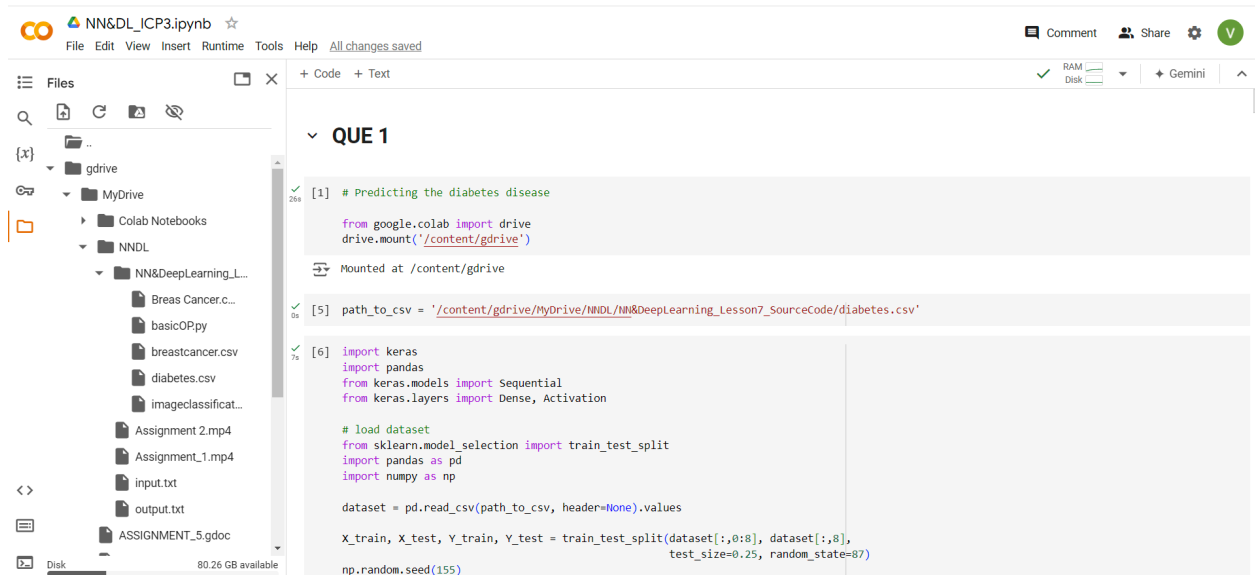


NNDL ASSIGNMENT - 3

VYSHNAVI NAGALLA - 700759215

VideoLink:

https://drive.google.com/file/d/1EZm0HkLDDcQZE44Jo_ft7dFE_3tVW-N6/view?usp=sharing



The screenshot shows the Google Colab interface for a notebook titled "NN&DL_ICP3.ipynb". The left sidebar displays the file explorer with a directory structure including "gdrive", "MyDrive", "Colab Notebooks", and "NNDL". The "NNDL" directory is expanded, showing files like "Breas Cancer.c...", "basicOPpy", "breastcancer.csv", "diabetes.csv", "imageclassificat...", "Assignment 2.mp4", "Assignment 1.mp4", "input.txt", "output.txt", and "ASSIGNMENT_5.gdoc". The main code area is titled "QUE 1" and contains the following code:

```
[1] # Predicting the diabetes disease

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

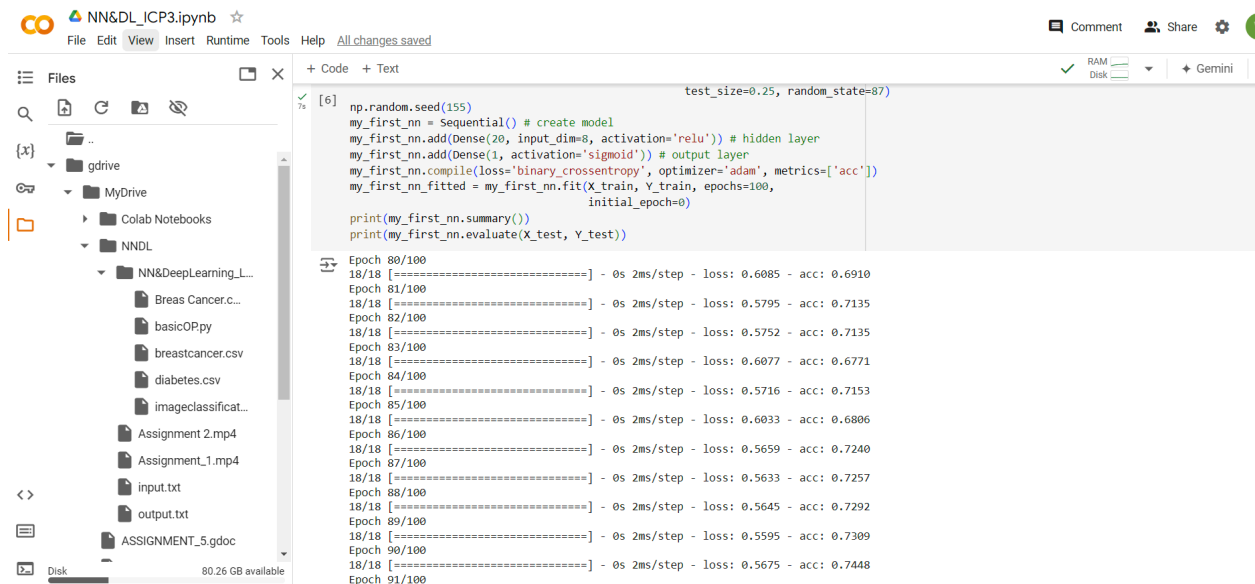
Mounted at /content/gdrive

[5] path_to_csv = '/content/gdrive/MyDrive/NNDL/NN&DeepLearning_Lesson7_SourceCode/diabetes.csv'

[6] import keras
import pandas
from keras.models import Sequential
from keras.layers import Dense, Activation

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

dataset = pd.read_csv(path_to_csv, header=None).values
X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
```



The screenshot shows the continuation of the Google Colab notebook. The code area contains the following code:

```
[6] 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(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 output of the code shows the training progress and evaluation results:

```
Epoch 80/100
18/18 [=====] - 0s 2ms/step - loss: 0.6085 - acc: 0.6910
Epoch 81/100
18/18 [=====] - 0s 2ms/step - loss: 0.5795 - acc: 0.7135
Epoch 82/100
18/18 [=====] - 0s 2ms/step - loss: 0.5752 - acc: 0.7135
Epoch 83/100
18/18 [=====] - 0s 2ms/step - loss: 0.6077 - acc: 0.6771
Epoch 84/100
18/18 [=====] - 0s 2ms/step - loss: 0.5716 - acc: 0.7153
Epoch 85/100
18/18 [=====] - 0s 2ms/step - loss: 0.6033 - acc: 0.6806
Epoch 86/100
18/18 [=====] - 0s 2ms/step - loss: 0.5659 - acc: 0.7240
Epoch 87/100
18/18 [=====] - 0s 2ms/step - loss: 0.5633 - acc: 0.7257
Epoch 88/100
18/18 [=====] - 0s 2ms/step - loss: 0.5645 - acc: 0.7292
Epoch 89/100
18/18 [=====] - 0s 2ms/step - loss: 0.5595 - acc: 0.7309
Epoch 90/100
18/18 [=====] - 0s 2ms/step - loss: 0.5675 - acc: 0.7448
Epoch 91/100
```

colab.research.google.com/drive/1r49Hnf4MQIC1ueACKD9jmfpgEcGXK3M#scrollTo=UTAJGclqVC

NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- gdrive
- MyDrive
 - Colab Notebooks
 - NNDL
 - NN&DeepLearning_L...
 - Breas Cancer.c...
 - basicOP.py
 - breastcancer.csv
 - diabetes.csv
 - imageclassificat...
 - Assignment 2.mp4
 - Assignment 1.mp4
 - input.txt
 - output.txt
 - ASSIGNMENT_5.gdoc

Disk 80.26 GB available

+ Code + Text

```
18/18 [=====] - 0s 2ms/step - loss: 0.5545 - acc: 0.7344
Epoch 92/100
18/18 [=====] - 0s 2ms/step - loss: 0.5763 - acc: 0.7292
Epoch 93/100
18/18 [=====] - 0s 3ms/step - loss: 0.5878 - acc: 0.7135
Epoch 94/100
18/18 [=====] - 0s 2ms/step - loss: 0.5615 - acc: 0.7396
Epoch 95/100
18/18 [=====] - 0s 2ms/step - loss: 0.5550 - acc: 0.7413
Epoch 96/100
18/18 [=====] - 0s 2ms/step - loss: 0.5677 - acc: 0.7153
Epoch 97/100
18/18 [=====] - 0s 4ms/step - loss: 0.5727 - acc: 0.7135
Epoch 98/100
18/18 [=====] - 0s 2ms/step - loss: 0.5748 - acc: 0.7101
Epoch 99/100
18/18 [=====] - 0s 2ms/step - loss: 0.5916 - acc: 0.6927
Epoch 100/100
18/18 [=====] - 0s 3ms/step - loss: 0.5632 - acc: 0.7205
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 20)	180
dense_1 (Dense)	(None, 1)	21

Total params: 201 (804.00 Byte)
Trainable params: 201 (804.00 Byte)
Non-trainable params: 0 (0.00 Byte)

```
None
6/6 [=====] - 0s 3ms/step - loss: 0.6195 - acc: 0.6667
```

NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- gdrive
- MyDrive
 - Colab Notebooks
 - NNDL
 - NN&DeepLearning_L...
 - Breas Cancer.c...
 - basicOP.py
 - breastcancer.csv
 - diabetes.csv
 - imageclassificat...
 - Assignment 2.mp4
 - Assignment 1.mp4
 - input.txt
 - output.txt
 - ASSIGNMENT_5.gdoc

Disk 80.26 GB available

+ Code + Text

```
6/6 [=====] - 0s 3ms/step - loss: 0.6195 - acc: 0.6667
[0.6194834113121033, 0.66666666665348816]
```

1. a. Add more Dense layers to the existing code and check how the accuracy changes.

```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer with input
my_first_nn.add(Dense(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, 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']) # compilation
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, verbose=0, initial_epoch=0) # Training
print(my_first_nn.summary()) # Summary
print(my_first_nn.evaluate(X_test, Y_test)) # Evaluating
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 20)	180
dense_3 (Dense)	(None, 20)	420

1m 24s completed at 8:36 PM

colab.research.google.com/drive/1r49Hnf4MQIC1ueACKD9jmfpgEcGXK3M#scrollTo=UTAJGclqVC

NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- gdrive
- MyDrive
 - Colab Notebooks
 - NNDL
 - NN&DeepLearning_L...
 - Breas Cancer.c...
 - basicOP.py
 - breastcancer.csv
 - diabetes.csv

Disk 80.26 GB available

+ Code + Text

```
dense_3 (dense) (None, 20) 420
[7]
dense_4 (Dense) (None, 20) 420
dense_5 (Dense) (None, 20) 420
dense_6 (Dense) (None, 20) 420
dense_7 (Dense) (None, 20) 420
dense_8 (Dense) (None, 1) 21
```

Layer (type)	Output Shape	Param #
dense_3 (dense)	(None, 20)	420
dense_4 (Dense)	(None, 20)	420
dense_5 (Dense)	(None, 20)	420
dense_6 (Dense)	(None, 20)	420
dense_7 (Dense)	(None, 20)	420
dense_8 (Dense)	(None, 1)	21

Total params: 2301 (8.99 KB)
Trainable params: 2301 (8.99 KB)
Non-trainable params: 0 (0.00 Byte)

```
None
6/6 [=====] - 0s 3ms/step - loss: 0.6079 - acc: 0.7396
[0.6078576445579529, 0.7395833134651184]
```

Files

gdrive

MyDrive

Colab Notebooks

NNDL

NN&DeepLearning_L...

Breas Cancer.c...

basicOPpy

breastcancer.csv

diabetes.csv

imageclassificat...

Assignment 2.mp4

Assignment_1.mp4

input.txt

output.txt

ASSIGNMENT_5.doc

Disk80.26 GB available

+ Code + Text

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

```
path_to_csv = '/content/gdrive/MyDrive>NNDL>NN&DeepLearning_Lesson7_SourceCode>breastcancer.csv'

#Importing packages for creating arrays
import numpy as np
import pandas as pd

#Importing packages to convert Categorical data into Numerical
from sklearn.preprocessing import LabelEncoder

#Importing packages for splitting data
from sklearn.model_selection import train_test_split

#Importing packages for keras
from keras.models import Sequential
from keras.layers import Dense, Activation

#Loading the Dataset
dataset = pd.read_csv(path_to_csv, header=0)
```

[] dataset

NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

gdrive

MyDrive

Colab Notebooks

NNDL

NN&DeepLearning_L...

Breas Cancer.c...

basicOPpy

breastcancer.csv

diabetes.csv

imageclassificat...

Assignment 2.mp4

Assignment_1.mp4

input.txt

output.txt

ASSIGNMENT_5.doc

Disk80.26 GB available

+ Code + Text

dataset

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows x 33 columns

[] #converting Categorical data into Numerical Using Label Encoding

Colab Notebook: NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files: gdrive, MyDrive, Colab Notebooks, NN&DL, NN&DeepLearning_L..., Breast Cancer.c..., basicOPpy, breastcancer.csv, diabetes.csv, imageclassificat..., Assignment 2.mp4, Assignment_1.mp4, input.txt, output.txt, ASSIGNMENT_5.gdoc

Code: + Code + Text

```
#converting Categorical data into Numerical Using Label Encoding
le=LabelEncoder()
dataset['diagnosis'] = le.fit_transform(dataset['diagnosis'])

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   column              Non-Null Count  Dtype
---  -
 0   id                  569 non-null    int64
 1   diagnosis           569 non-null    int64
 2   radius_mean         569 non-null    float64
 3   texture_mean        569 non-null    float64
 4   perimeter_mean      569 non-null    float64
 5   area_mean           569 non-null    float64
 6   smoothness_mean     569 non-null    float64
 7   compactness_mean    569 non-null    float64
 8   concavity_mean      569 non-null    float64
 9   concave points_mean 569 non-null    float64
10   symmetry_mean       569 non-null    float64
11   fractal_dimension_mean 569 non-null    float64
12   radius_se           569 non-null    float64
13   texture_se          569 non-null    float64
14   perimeter_se        569 non-null    float64
15   area_se             569 non-null    float64
16   smoothness_se       569 non-null    float64
17   compactness_se      569 non-null    float64
18   concavity_se        569 non-null    float64
19   concave points_se   569 non-null    float64
20   symmetry_se         569 non-null    float64
```

1m 24s completed at 8:36 PM

Colab Notebook: NN&DL_ICP3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files: gdrive, MyDrive, Colab Notebooks, NN&DL, NN&DeepLearning_L..., Breast Cancer.c..., basicOPpy, breastcancer.csv, diabetes.csv, imageclassificat..., Assignment 2.mp4, Assignment_1.mp4, input.txt, output.txt, ASSIGNMENT_5.gdoc

Code: + Code + Text

```
11 fractal_dimension_mean 569 non-null float64
12 radius_se             569 non-null float64
13 texture_se             569 non-null float64
14 perimeter_se           569 non-null float64
15 area_se                569 non-null float64
16 smoothness_se         569 non-null float64
17 compactness_se        569 non-null float64
18 concavity_se           569 non-null float64
19 concave points_se      569 non-null float64
20 symmetry_se            569 non-null float64
21 fractal_dimension_se   569 non-null float64
22 radius_worst           569 non-null float64
23 texture_worst          569 non-null float64
24 perimeter_worst        569 non-null float64
25 area_worst             569 non-null float64
26 smoothness_worst      569 non-null float64
27 compactness_worst      569 non-null float64
28 concavity_worst        569 non-null float64
29 concave points_worst   569 non-null float64
30 symmetry_worst         569 non-null float64
31 fractal_dimension_worst 569 non-null float64
32 unnamed: 32            0 non-null      float64
dtypes: float64(31), int64(2)
memory usage: 146.8 KB
```

```
[10] #Splitting data into Feature Matrix & Label Matrix
X_train, X_test, y_train, y_test = train_test_split(dataset.iloc[:,2:32], dataset.iloc[:,1], test_size=0.25, random_state=87)

my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
```

1m 24s completed at 8:36 PM

Colab interface showing a Jupyter Notebook titled "NN&DL_ICP3.ipynb". The file explorer on the left shows a directory structure with files like "Breas Cancer.c...", "basicOP.py", "breastcancer.csv", "diabetes.csv", "imageclassificat...", "Assignment 2.mp4", "Assignment 1.mp4", "input.txt", "output.txt", and "ASSIGNMENT_5.gdoc". The code cell [60] contains the following Python code:

```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, input_dim=30, activation='relu')) # hidden layer
my_first_nn.add(Dense(20, input_dim=30, 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']) # compilation
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, verbose=0, initial_epoch=0) # Training
print(my_first_nn.summary()) #Summary
print(my_first_nn.evaluate(X_test, Y_test)) #Evaluating
```

The output shows the model summary for "sequential_2":

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 20)	620
dense_10 (Dense)	(None, 20)	420
dense_11 (Dense)	(None, 20)	420
dense_12 (Dense)	(None, 1)	21

Total params: 1481 (5.79 KB)
Trainable params: 1481 (5.79 KB)
Non-trainable params: 0 (0.00 Byte)

5/5 [=====] - 0s 3ms/step - loss: 0.2662 - acc: 0.9301
[0.26623570919036865, 0.9300699234008789]

Colab interface showing a Jupyter Notebook titled "NN&DL_ICP3.ipynb". The file explorer on the left shows a directory structure with files like "Breas Cancer.c...", "basicOP.py", "breastcancer.csv", "diabetes.csv", "imageclassificat...", "Assignment 2.mp4", "Assignment 1.mp4", "input.txt", "output.txt", and "ASSIGNMENT_5.gdoc". The code cell [11] contains the following Python code:

```
#importing packages for Normalization
from sklearn.preprocessing import StandardScaler
```

The code cell [12] contains the following Python code:

```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=30, 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']) # compilation

sc = StandardScaler() #Create Model
X_train = sc.fit_transform(X_train) #Fit to data, then transform it.
X_test = sc.transform(X_test) # Perform standardization by centering and scaling

my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100, verbose=0, initial_epoch=0) # Training
print(my_first_nn.summary()) #Summary
print(my_first_nn.evaluate(X_test, Y_test)) #Evaluating
```

The output shows the model summary for "sequential_3":

Layer (type)	Output Shape	Param #
dense_13 (Dense)	(None, 20)	620
dense_14 (Dense)	(None, 1)	21

File Edit View Insert Runtime Tools Help All changes saved

Files

gdrive

MyDrive

Colab Notebooks

NNDL

NN&DeepLearning_L...

Breas Cancer.c...

basicOP.py

breastcancer.csv

diabetes.csv

imageclassificat...

Assignment 2.mp4

Assignment_1.mp4

input.txt

output.txt

ASSIGNMENT_5.gdoc

Disk 80.26 GB available

+ Code + Text

trainable params: 041 (2.30 KB)

[12] Non-trainable params: 0 (0.00 Byte)

None

5/5 [=====] - 0s 3ms/step - loss: 0.1086 - acc: 0.9790

[0.10860157757997513, 0.9790209531784058]

QUE 2

from keras import Sequential

from keras.datasets import mnist

import numpy as np

from keras.layers import Dense

from keras.utils import to_categorical

import matplotlib.pyplot as plt

(train_images,train_labels),(test_images, test_labels) = mnist.load_data()

#process the data

#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature

dimData = np.prod(train_images.shape[1:])

print(dimData)

train_data = train_images.reshape(train_images.shape[0],dimData)

test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1

train_data = train_data.astype('float')

test_data = test_data.astype('float')

1m 24s completed at 8:36 PM

File Edit View Insert Runtime Tools Help All changes saved

Files

gdrive

MyDrive

Colab Notebooks

NNDL

NN&DeepLearning_L...

Breas Cancer.c...

basicOP.py

breastcancer.csv

diabetes.csv

imageclassificat...

Assignment 2.mp4

Assignment_1.mp4

input.txt

output.txt

ASSIGNMENT_5.gdoc

Disk 80.26 GB available

+ Code + Text

train_data = train_images.reshape(train_images.shape[0],dimData)

test_data = test_images.reshape(test_images.shape[0],dimData)

#convert data to float and scale values between 0 and 1

train_data = train_data.astype('float')

test_data = test_data.astype('float')

#scale data

train_data /=255.0

test_data /=255.0

#change the labels from integer to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()

train_labels_one_hot = to_categorical(train_labels)

test_labels_one_hot = to_categorical(test_labels)

#creating network

model = Sequential()

model.add(Dense(512, activation='relu', input_shape=(dimData,)))

model.add(Dense(512, activation='relu'))

model.add(Dense(10, activation='softmax'))

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>

11490434/11490434 [=====] - 0s 0us/step

784

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

1m 24s completed at 8:36 PM

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

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

```
Epoch 1/10
235/235 [=====] - 7s 27ms/step - loss: 0.2944 - accuracy: 0.9108 - val_loss: 0.1323 - val_accuracy: 0.9604
Epoch 2/10
235/235 [=====] - 6s 25ms/step - loss: 0.1014 - accuracy: 0.9683 - val_loss: 0.0949 - val_accuracy: 0.9698
Epoch 3/10
235/235 [=====] - 5s 22ms/step - loss: 0.0629 - accuracy: 0.9803 - val_loss: 0.0777 - val_accuracy: 0.9750
Epoch 4/10
235/235 [=====] - 8s 34ms/step - loss: 0.0429 - accuracy: 0.9863 - val_loss: 0.0668 - val_accuracy: 0.9784
Epoch 5/10
235/235 [=====] - 5s 22ms/step - loss: 0.0314 - accuracy: 0.9901 - val_loss: 0.0667 - val_accuracy: 0.9737
Epoch 6/10
235/235 [=====] - 6s 24ms/step - loss: 0.0238 - accuracy: 0.9921 - val_loss: 0.0937 - val_accuracy: 0.9738
Epoch 7/10
235/235 [=====] - 6s 24ms/step - loss: 0.0171 - accuracy: 0.9947 - val_loss: 0.0618 - val_accuracy: 0.9830
Epoch 8/10
235/235 [=====] - 8s 33ms/step - loss: 0.0125 - accuracy: 0.9963 - val_loss: 0.0876 - val_accuracy: 0.9791
Epoch 9/10
235/235 [=====] - 8s 32ms/step - loss: 0.0096 - accuracy: 0.9972 - val_loss: 0.1402 - val_accuracy: 0.9637
Epoch 10/10
235/235 [=====] - 5s 22ms/step - loss: 0.0087 - accuracy: 0.9974 - val_loss: 0.0656 - val_accuracy: 0.9832
```

1m 24s completed at 8:36 PM

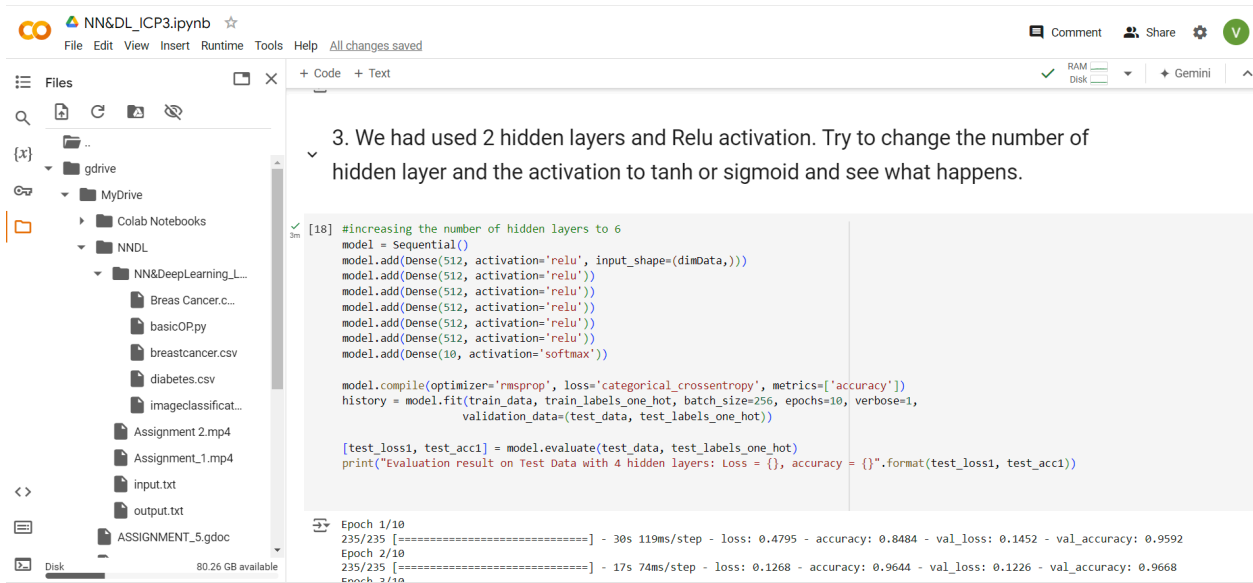
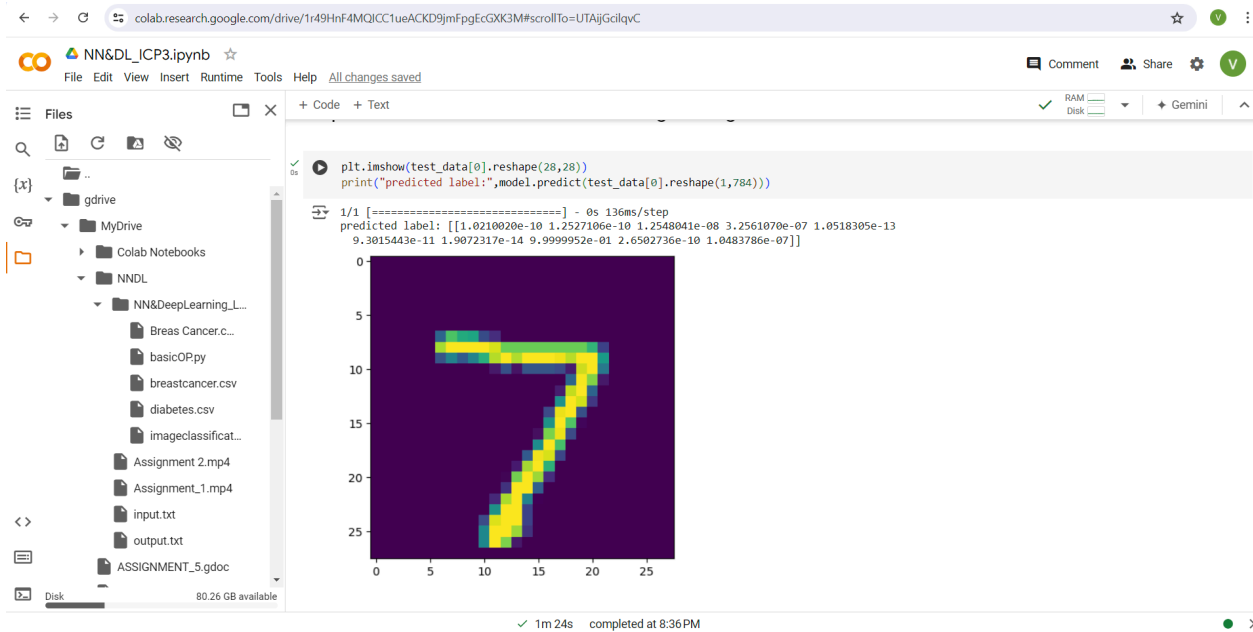
```
[15] [test_loss, test_acc] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data : Loss = {}, accuracy = {}".format(test_loss, test_acc))

history.history.keys()
```

```
313/313 [=====] - 1s 4ms/step - loss: 0.0656 - accuracy: 0.9832
Evaluation result on Test Data : Loss = 0.06558870524168015, accuracy = 0.9832000136375427
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy', 'loss', 'val_loss'], loc='upper left')
plt.show()
```





Colab interface showing the execution of a neural network training script. The file explorer on the left shows the project structure, including files like `breastcancer.csv`, `diabetes.csv`, and `imageclassificat...`. The code editor displays the training progress, showing epochs 1/10 to 10/10. The output indicates the final evaluation result on test data with 4 hidden layers: Loss = 0.09822707623243332, accuracy = 0.9811000227928162.

```
[19] #All hidden layers with tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(dimData,)))
model.add(Dense(612, activation='tanh'))
model.add(Dense(712, activation='tanh'))
model.add(Dense(812, activation='tanh'))
model.add(Dense(712, activation='tanh'))
model.add(Dense(812, activation='tanh'))
```

Colab interface showing the execution of a neural network training script. The file explorer on the left shows the project structure, including files like `breastcancer.csv`, `diabetes.csv`, and `imageclassificat...`. The code editor displays the training progress, showing epochs 1/10 to 10/10. The output indicates the final evaluation result on test data with tanh activation: Loss = {}, accuracy = {}.format(test_loss2, test_acc2).

```
model.add(Dense(512, activation='tanh'))
model.add(Dense(712, activation='tanh'))
model.add(Dense(812, activation='tanh'))
model.add(Dense(712, activation='tanh'))
model.add(Dense(812, activation='tanh'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

[test_loss2, test_acc2] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data with tanh activation: Loss = {}, accuracy = {}".format(test_loss2, test_acc2))
```

Colab Notebook: NN&DL_IP3.ipynb

Files: gdrive, MyDrive, Colab Notebooks, NN&DL, NN&DeepLearning_L...

```
Epoch 8/10
235/235 [=====] - 28s 118ms/step - loss: 0.0447 - accuracy: 0.9859 - val_loss: 0.1056 - val_accuracy: 0.9709
Epoch 9/10
235/235 [=====] - 26s 109ms/step - loss: 0.0385 - accuracy: 0.9875 - val_loss: 0.1619 - val_accuracy: 0.9611
Epoch 10/10
235/235 [=====] - 24s 101ms/step - loss: 0.0315 - accuracy: 0.9900 - val_loss: 0.0993 - val_accuracy: 0.9761
313/313 [=====] - 3s 8ms/step - loss: 0.0993 - accuracy: 0.9761
Evaluation result on Test Data with tanh activation: Loss = 0.0993187874555878, accuracy = 0.9761000275611877
```

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

```
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

print(train_images.shape[1:])
#process the data
#1. convert each image of shape 28*28 to 784 dimensional which will be fed to the network as a single feature
dimData = np.prod(train_images.shape[1:])
# print(dimData)
train_data = train_images.reshape(train_images.shape[0], dimData)
test_data = test_images.reshape(test_images.shape[0], dimData)

#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
```

Colab Notebook: NN&DL_IP3.ipynb

Files: gdrive, MyDrive, Colab Notebooks, NN&DL, NN&DeepLearning_L...

```
#change the labels from integer to one-hot encoding. to_categorical is doing the same thing as LabelEncoder()
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)

#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))

[test_loss3, test_acc3] = model.evaluate(test_data, test_labels_one_hot)
print("Evaluation result on Test Data without scaling: Loss = {}, accuracy = {}".format(test_loss3, test_acc3))
```

(28, 28)

```
Epoch 1/10
235/235 [=====] - 8s 31ms/step - loss: 6.1178 - accuracy: 0.8749 - val_loss: 1.2764 - val_accuracy: 0.8586
Epoch 2/10
235/235 [=====] - 5s 22ms/step - loss: 0.4063 - accuracy: 0.9435 - val_loss: 0.3669 - val_accuracy: 0.9369
Epoch 3/10
235/235 [=====] - 5s 23ms/step - loss: 0.2410 - accuracy: 0.9587 - val_loss: 0.3035 - val_accuracy: 0.9493
Epoch 4/10
235/235 [=====] - 7s 30ms/step - loss: 0.1871 - accuracy: 0.9666 - val_loss: 0.5185 - val_accuracy: 0.9312
Epoch 5/10
235/235 [=====] - 5s 22ms/step - loss: 0.1703 - accuracy: 0.9710 - val_loss: 0.2873 - val_accuracy: 0.9591
Epoch 6/10
235/235 [=====] - 7s 31ms/step - loss: 0.1498 - accuracy: 0.9751 - val_loss: 0.2920 - val_accuracy: 0.9636
Epoch 7/10
```

Colab Notebook: NN&DL_IP3.ipynb

Files: gdrive, MyDrive, Colab Notebooks, NN&DL, NN&DeepLearning_L...

(28, 28)

```
Epoch 1/10
235/235 [=====] - 8s 31ms/step - loss: 6.1178 - accuracy: 0.8749 - val_loss: 1.2764 - val_accuracy: 0.8586
Epoch 2/10
235/235 [=====] - 5s 22ms/step - loss: 0.4063 - accuracy: 0.9435 - val_loss: 0.3669 - val_accuracy: 0.9369
Epoch 3/10
235/235 [=====] - 5s 23ms/step - loss: 0.2410 - accuracy: 0.9587 - val_loss: 0.3035 - val_accuracy: 0.9493
Epoch 4/10
235/235 [=====] - 7s 30ms/step - loss: 0.1871 - accuracy: 0.9666 - val_loss: 0.5185 - val_accuracy: 0.9312
Epoch 5/10
235/235 [=====] - 5s 22ms/step - loss: 0.1703 - accuracy: 0.9710 - val_loss: 0.2873 - val_accuracy: 0.9591
Epoch 6/10
235/235 [=====] - 7s 31ms/step - loss: 0.1498 - accuracy: 0.9751 - val_loss: 0.2920 - val_accuracy: 0.9636
Epoch 7/10
235/235 [=====] - 5s 22ms/step - loss: 0.1388 - accuracy: 0.9778 - val_loss: 0.3159 - val_accuracy: 0.9608
Epoch 8/10
235/235 [=====] - 6s 24ms/step - loss: 0.1253 - accuracy: 0.9811 - val_loss: 0.3198 - val_accuracy: 0.9680
Epoch 9/10
235/235 [=====] - 7s 30ms/step - loss: 0.1197 - accuracy: 0.9832 - val_loss: 0.3001 - val_accuracy: 0.9745
Epoch 10/10
235/235 [=====] - 6s 27ms/step - loss: 0.1249 - accuracy: 0.9848 - val_loss: 0.3107 - val_accuracy: 0.9737
313/313 [=====] - 1s 4ms/step - loss: 0.3107 - accuracy: 0.9737
Evaluation result on Test Data without scaling: Loss = 0.3107052743434906, accuracy = 0.9736999869346619
```

