

Summer 2024: CS5720 NNDL - ICP-3

Lasya Vanga (700762893)

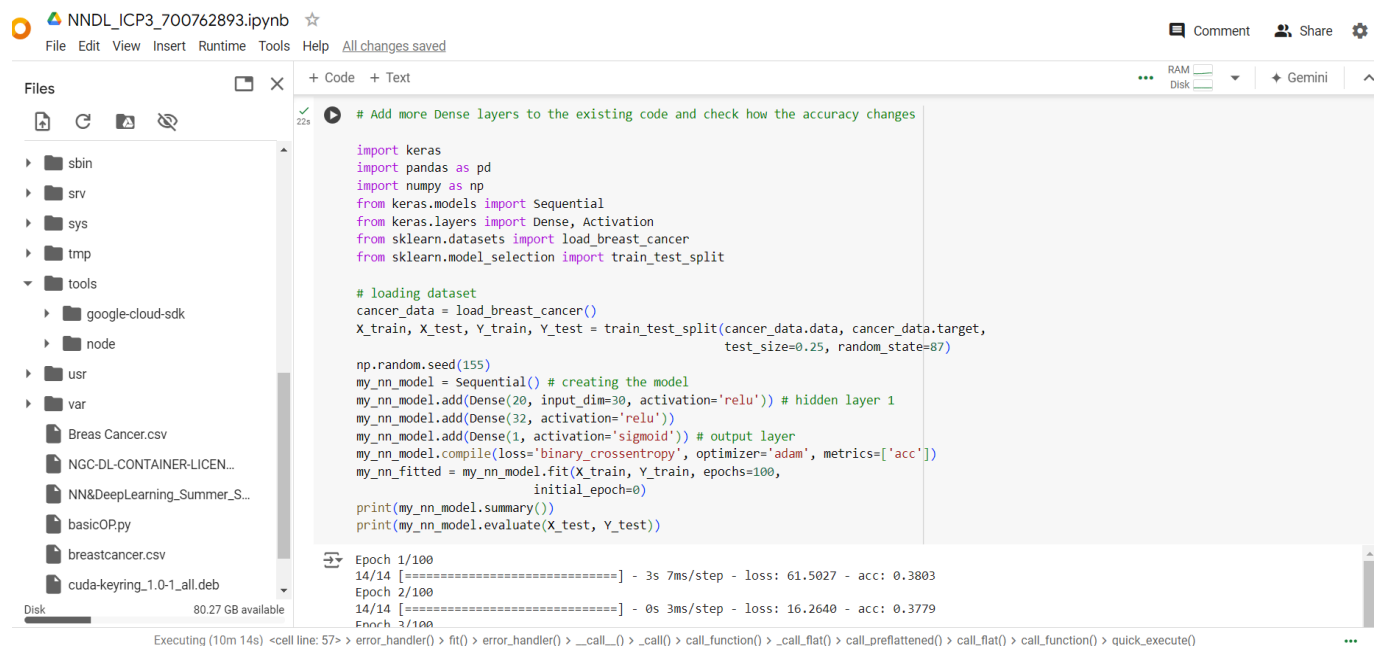
GitHub Link: <https://github.com/Lasya-vanga/NNDL-ICP3>

Use Case Description: Predicting the diabetes disease

Programming elements: Keras Basics

1) Use the use case in the class:

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



```
# Add more Dense layers to the existing code and check how the accuracy changes

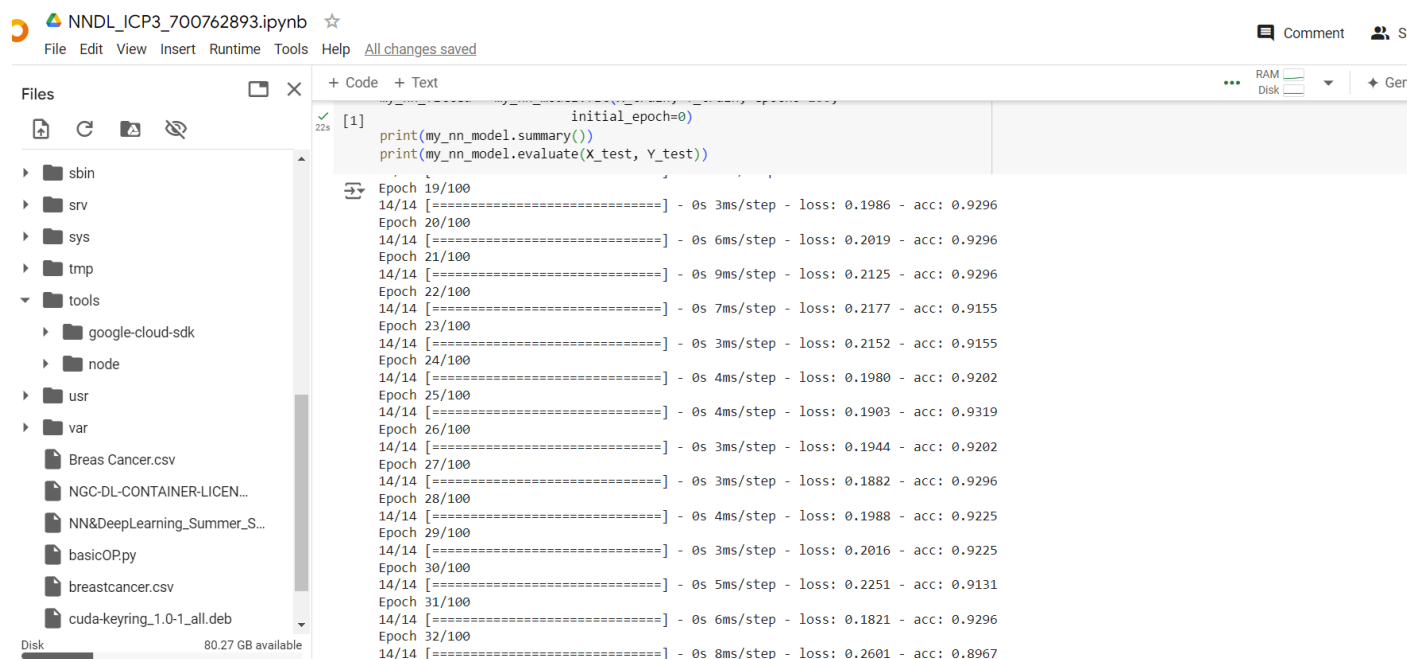
import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# loading 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_model = Sequential() # creating the model
my_nn_model.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn_model.add(Dense(32, activation='relu'))
my_nn_model.add(Dense(1, activation='sigmoid')) # output layer
my_nn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn_model.fit(X_train, Y_train, epochs=100,
                               initial_epoch=0)

print(my_nn_model.summary())
print(my_nn_model.evaluate(X_test, Y_test))
```

Epoch 1/100
14/14 [=====] - 3s 7ms/step - loss: 61.5027 - acc: 0.3803
Epoch 2/100
14/14 [=====] - 0s 3ms/step - loss: 16.2640 - acc: 0.3779
Epoch 3/100



```
[1] initial_epoch=0
print(my_nn_model.summary())
print(my_nn_model.evaluate(X_test, Y_test))
```

Epoch 19/100
14/14 [=====] - 0s 3ms/step - loss: 0.1986 - acc: 0.9296
Epoch 20/100
14/14 [=====] - 0s 6ms/step - loss: 0.2019 - acc: 0.9296
Epoch 21/100
14/14 [=====] - 0s 9ms/step - loss: 0.2125 - acc: 0.9296
Epoch 22/100
14/14 [=====] - 0s 7ms/step - loss: 0.2177 - acc: 0.9155
Epoch 23/100
14/14 [=====] - 0s 3ms/step - loss: 0.2152 - acc: 0.9155
Epoch 24/100
14/14 [=====] - 0s 4ms/step - loss: 0.1980 - acc: 0.9202
Epoch 25/100
14/14 [=====] - 0s 4ms/step - loss: 0.1903 - acc: 0.9319
Epoch 26/100
14/14 [=====] - 0s 3ms/step - loss: 0.1944 - acc: 0.9202
Epoch 27/100
14/14 [=====] - 0s 3ms/step - loss: 0.1882 - acc: 0.9296
Epoch 28/100
14/14 [=====] - 0s 4ms/step - loss: 0.1988 - acc: 0.9225
Epoch 29/100
14/14 [=====] - 0s 3ms/step - loss: 0.2016 - acc: 0.9225
Epoch 30/100
14/14 [=====] - 0s 5ms/step - loss: 0.2251 - acc: 0.9131
Epoch 31/100
14/14 [=====] - 0s 6ms/step - loss: 0.1821 - acc: 0.9296
Epoch 32/100
14/14 [=====] - 0s 8ms/step - loss: 0.2601 - acc: 0.8967

NNDL_ICP3_700762893.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
 - basicOPpy
 - breastcancer.csv
 - cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

+ Code + Text

```

Epoch 94/100
14/14 [=====] - 0s 3ms/step - loss: 0.1687 - acc: 0.9319
Epoch 95/100
14/14 [=====] - 0s 3ms/step - loss: 0.1852 - acc: 0.9272
Epoch 96/100
14/14 [=====] - 0s 3ms/step - loss: 0.2236 - acc: 0.9108
Epoch 97/100
14/14 [=====] - 0s 3ms/step - loss: 0.1748 - acc: 0.9296
Epoch 98/100
14/14 [=====] - 0s 3ms/step - loss: 0.1671 - acc: 0.9437
Epoch 99/100
14/14 [=====] - 0s 2ms/step - loss: 0.1601 - acc: 0.9272
Epoch 100/100
14/14 [=====] - 0s 2ms/step - loss: 0.1629 - acc: 0.9390
Model: "sequential"

Layer (type)                 Output Shape              Param #
-----
dense (Dense)                 (None, 20)                620
dense_1 (Dense)               (None, 32)                672
dense_2 (Dense)               (None, 1)                 33

Total params: 1325 (5.18 KB)
Trainable params: 1325 (5.18 KB)
Non-trainable params: 0 (0.00 Byte)

None
5/5 [=====] - 0s 4ms/step - loss: 0.2680 - acc: 0.9161
[0.26797401905059814, 0.9160839319229126]

```

2. 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()

NNDL_ICP3_700762893.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
 - basicOPpy
 - breastcancer.csv
 - cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

+ Code + Text

```

# Normalize the data before feeding the data to the model and check how the normalization change your accuracy

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 import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# loading 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_model = Sequential() # creating model
my_nn_model.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn_model.add(Dense(1, activation='sigmoid')) # output layer
my_nn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn_model.fit(X_train, Y_train, epochs=100,
                               initial_epoch=0)
print(my_nn_model.summary())
print(my_nn_model.evaluate(X_test, Y_test))

```

NNDL_ICP3_700762893.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sbinsrvsys
- tmp
- tools
 - google-cloud-sdk
 - node
 - usr
 - var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
- basicOPpy
- breastcancer.csv
- cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

Executing (1m 24s) <cell line: 53> > error_handler() > fit() > error_handler() > _call_() > _call() > call_function() > _call_pre flattened() > call_flat() > call_fun

```
[2] Epoch 40/100
14/14 [=====] - 0s 3ms/step - loss: 0.4692 - acc: 0.8779
Epoch 41/100
14/14 [=====] - 0s 3ms/step - loss: 0.3260 - acc: 0.9085
Epoch 42/100
14/14 [=====] - 0s 3ms/step - loss: 0.2845 - acc: 0.9061
Epoch 43/100
14/14 [=====] - 0s 3ms/step - loss: 0.2834 - acc: 0.9155
Epoch 44/100
14/14 [=====] - 0s 2ms/step - loss: 0.2856 - acc: 0.9108
Epoch 45/100
14/14 [=====] - 0s 2ms/step - loss: 0.2816 - acc: 0.9178
Epoch 46/100
14/14 [=====] - 0s 2ms/step - loss: 0.3129 - acc: 0.8897
Epoch 47/100
14/14 [=====] - 0s 2ms/step - loss: 0.3113 - acc: 0.8991
Epoch 48/100
14/14 [=====] - 0s 2ms/step - loss: 0.3203 - acc: 0.9061
Epoch 49/100
14/14 [=====] - 0s 2ms/step - loss: 0.3343 - acc: 0.9085
Epoch 50/100
14/14 [=====] - 0s 2ms/step - loss: 0.2842 - acc: 0.9061
Epoch 51/100
14/14 [=====] - 0s 2ms/step - loss: 0.2597 - acc: 0.9131
Epoch 52/100
14/14 [=====] - 0s 3ms/step - loss: 0.2963 - acc: 0.9178
Epoch 53/100
14/14 [=====] - 0s 2ms/step - loss: 0.2375 - acc: 0.9178
Epoch 54/100
14/14 [=====] - 0s 2ms/step - loss: 0.2392 - acc: 0.9202
Epoch 55/100
14/14 [=====] - 0s 2ms/step - loss: 0.2341 - acc: 0.9296
```

NNDL_ICP3_700762893.ipynb

File Edit View Insert Runtime Tools Help All changes saved

es

- sbinsrvsys
- tmp
- tools
 - google-cloud-sdk
 - node
 - usr
 - var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
- basicOPpy
- breastcancer.csv
- cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

```
Epoch 1/100
14/14 [=====] - 1s 2ms/step - loss: 14.2940 - acc: 0.1948
Epoch 2/100
14/14 [=====] - 0s 2ms/step - loss: 8.8873 - acc: 0.1690
Epoch 3/100
14/14 [=====] - 0s 3ms/step - loss: 4.0844 - acc: 0.3216
Epoch 4/100
14/14 [=====] - 0s 2ms/step - loss: 2.0391 - acc: 0.5469
Epoch 5/100
14/14 [=====] - 0s 2ms/step - loss: 1.3860 - acc: 0.6690
Epoch 6/100
14/14 [=====] - 0s 2ms/step - loss: 0.9560 - acc: 0.7113
Epoch 7/100
14/14 [=====] - 0s 3ms/step - loss: 0.9584 - acc: 0.7230
Epoch 8/100
14/14 [=====] - 0s 3ms/step - loss: 0.6838 - acc: 0.8122
Epoch 9/100
14/14 [=====] - 0s 2ms/step - loss: 0.6553 - acc: 0.7934
Epoch 10/100
14/14 [=====] - 0s 2ms/step - loss: 0.6589 - acc: 0.8122
Epoch 11/100
14/14 [=====] - 0s 2ms/step - loss: 0.6457 - acc: 0.8192
Epoch 12/100
14/14 [=====] - 0s 2ms/step - loss: 0.5700 - acc: 0.8169
Epoch 13/100
14/14 [=====] - 0s 2ms/step - loss: 0.5127 - acc: 0.8592
Epoch 14/100
14/14 [=====] - 0s 2ms/step - loss: 0.5607 - acc: 0.8310
Epoch 15/100
14/14 [=====] - 0s 2ms/step - loss: 0.6043 - acc: 0.8310
Epoch 16/100
14/14 [=====] - 0s 2ms/step - loss: 0.4739 - acc: 0.8685
```

NNDL_ICP3_700762893.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sbinsrvsys
- tmp
- tools
 - google-cloud-sdk
 - node
 - usr
 - var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
- basicOPpy
- breastcancer.csv
- cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

RAM 100% Disk 100%

```
Epoch 93/100
14/14 [=====] - 0s 2ms/step - loss: 0.1799 - acc: 0.9343
Epoch 94/100
14/14 [=====] - 0s 2ms/step - loss: 0.1619 - acc: 0.9413
Epoch 95/100
14/14 [=====] - 0s 2ms/step - loss: 0.1523 - acc: 0.9460
Epoch 96/100
14/14 [=====] - 0s 2ms/step - loss: 0.1519 - acc: 0.9437
Epoch 97/100
14/14 [=====] - 0s 2ms/step - loss: 0.1475 - acc: 0.9460
Epoch 98/100
14/14 [=====] - 0s 2ms/step - loss: 0.1512 - acc: 0.9413
Epoch 99/100
14/14 [=====] - 0s 2ms/step - loss: 0.1714 - acc: 0.9366
Epoch 100/100
14/14 [=====] - 0s 2ms/step - loss: 0.1776 - acc: 0.9225
Model: "sequential_1"
```

Layer (type)	Output shape	Param #
dense_3 (Dense)	(None, 20)	620
dense_4 (Dense)	(None, 1)	21

Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)

```
None
5/5 [=====] - 0s 4ms/step - loss: 0.2171 - acc: 0.9231
[0.21713323891162872, 0.9230769276618958]
```

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.

```
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

# converting 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)

# creating 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'])
```

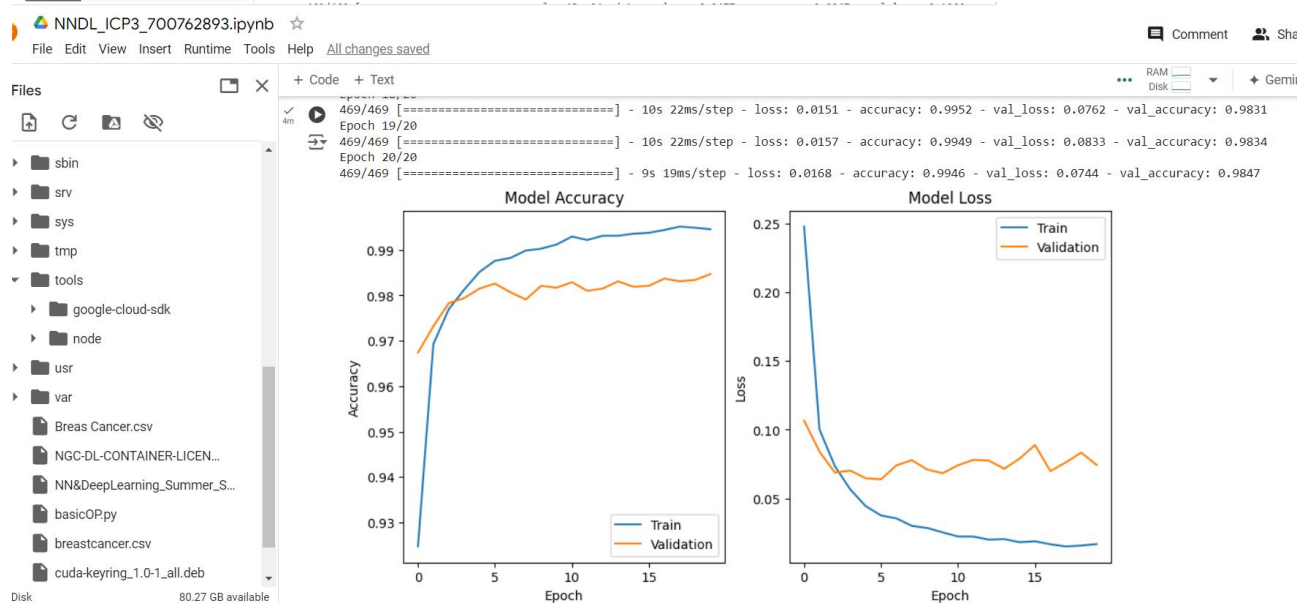
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# training 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)

# to 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()
```



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.

```
File Edit View Insert Runtime Tools Help All changes saved
```

Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
 - basicOPpy
 - breastcancer.csv
 - cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

```
+ Code + Text
```

```
# 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.
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

# loading 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

# converting 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)

# to 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'])
```

```
File Edit View Insert Runtime Tools Help All changes saved
```

Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
 - basicOPpy
 - breastcancer.csv
 - cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

```
+ Code + Text
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# to 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)

# to plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# making 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 [=====] - 12s 22ms/step - loss: 0.2458 - accuracy: 0.9255 - val_loss: 0.1079 - val_accuracy: 0.9661
Epoch 2/20
469/469 [=====] - 9s 19ms/step - loss: 0.1019 - accuracy: 0.9686 - val_loss: 0.0784 - val_accuracy: 0.9751
Epoch 3/20
469/469 [=====] - 11s 23ms/step - loss: 0.0748 - accuracy: 0.9769 - val_loss: 0.0776 - val_accuracy: 0.9775
Epoch 4/20
469/469 [=====] - 11s 23ms/step - loss: 0.0568 - accuracy: 0.9816 - val_loss: 0.0680 - val_accuracy: 0.9783
Epoch 5/20
469/469 [=====] - 10s 22ms/step - loss: 0.0439 - accuracy: 0.9859 - val_loss: 0.0769 - val_accuracy: 0.9791
Epoch 6/20
469/469 [=====] - 10s 21ms/step - loss: 0.0413 - accuracy: 0.9870 - val_loss: 0.0654 - val_accuracy: 0.9804
Epoch 7/20
469/469 [=====] - 11s 23ms/step - loss: 0.0327 - accuracy: 0.9890 - val_loss: 0.0671 - val_accuracy: 0.9798
Epoch 8/20
469/469 [=====] - 10s 22ms/step - loss: 0.0312 - accuracy: 0.9900 - val_loss: 0.0657 - val_accuracy: 0.9829
Epoch 9/20
469/469 [=====] - 10s 22ms/step - loss: 0.0302 - accuracy: 0.9896 - val_loss: 0.0703 - val_accuracy: 0.9819
```

```
File Edit View Insert Runtime Tools Help
```

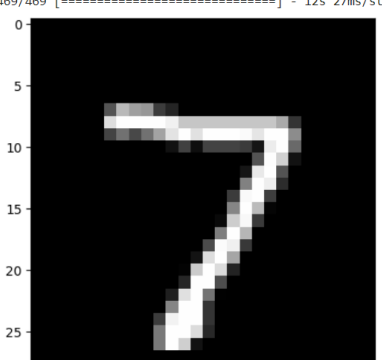
Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
 - Breas Cancer.csv
 - NGC-DL-CONTAINER-LICEN...
 - NN&DeepLearning_Summer_S...
 - basicOPpy
 - breastcancer.csv
 - cuda-keyring_1.0-1_all.deb

Disk 80.27 GB available

```
+ Code + Text
```

```
Epoch 18/20
469/469 [=====] - 10s 20ms/step - loss: 0.0152 - accuracy: 0.9951 - val_loss: 0.0764 - val_accuracy: 0.9822
Epoch 19/20
469/469 [=====] - 12s 25ms/step - loss: 0.0168 - accuracy: 0.9947 - val_loss: 0.0753 - val_accuracy: 0.9840
Epoch 20/20
469/469 [=====] - 12s 27ms/step - loss: 0.0166 - accuracy: 0.9945 - val_loss: 0.0671 - val_accuracy: 0.9857
```



```
1/1 [=====] - 0s 124ms/step
Model prediction: 7
```

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.

The following code is shown in the Jupyter Notebook:

```
# 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 wh
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.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))

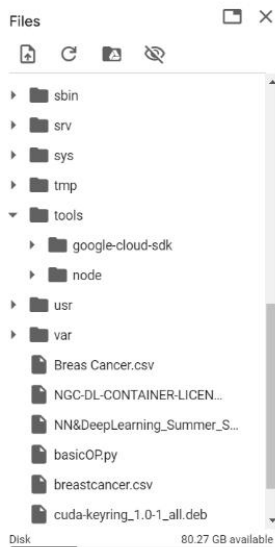
# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test), epochs=20, batch_size=128, 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))
```

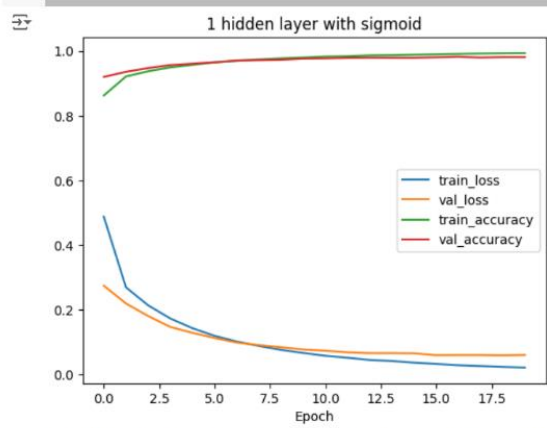
The plot shows the training and validation loss and accuracy over 20 epochs for the following models:

- 1 hidden layer with tanh
- 1 hidden layer with sigmoid
- 2 hidden layers with tanh
- 2 hidden layers with sigmoid

The plot indicates that the 2 hidden layers with tanh model achieved the lowest test loss (0.0606) and the highest test accuracy (0.9815).



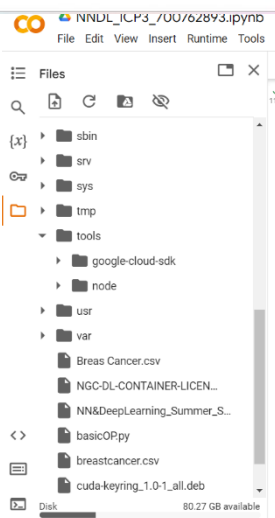
```
loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```



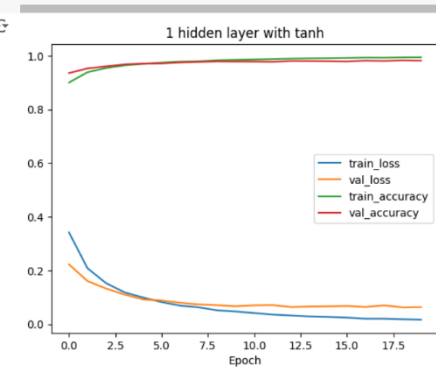
1 hidden layer with sigmoid - Test loss: 0.0606, Test accuracy: 0.9815

2 hidden layers with tanh

10m 36s completed at 5:56 PM

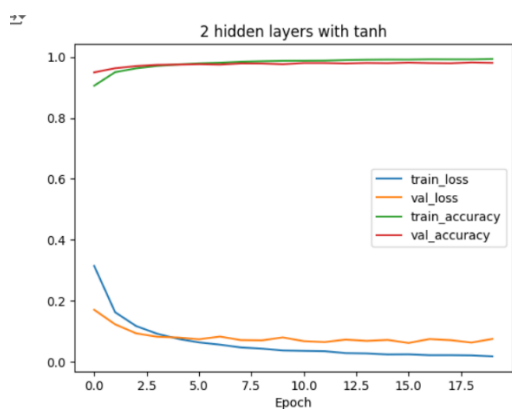


```
loss, accuracy = model.evaluate(x_test.reshape(-1, 784), y_test, verbose=0)
print('{} - Test loss: {:.4f}, Test accuracy: {:.4f}'.format(name, loss, accuracy))
```

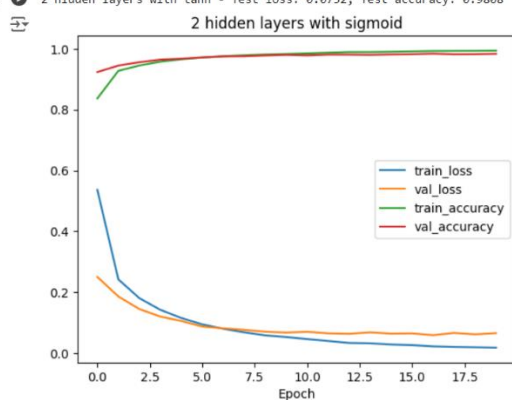


1 hidden layer with tanh - Test loss: 0.0634, Test accuracy: 0.9820

10m 36s completed at 5:56 PM



2 hidden layers with tanh - Test loss: 0.0752, Test accuracy: 0.9808



2 hidden layers with sigmoid - Test loss: 0.0660, Test accuracy: 0.9834

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

The image displays two screenshots of a Jupyter Notebook interface, showing code for training models on the MNIST dataset without image scaling.

Top Screenshot:

- File Explorer:** Shows the file system structure, including folders like `sbin`, `srv`, `sys`, `tmp`, `tools`, `google-cloud-sdk`, `node`, `usr`, `var`, and files like `Breas Cancer.csv`, `NGC-DL-CONTAINER-LICEN...`, `NN&DeepLearning_Summer_S...`, `basicOP.py`, `breastcancer.csv`, and `cuda-keyring_1.0-1_all.deb`.
- Code:** The code defines a list of models to train. It includes imports for `keras`, `mnist`, `Sequential`, `Dense`, `Dropout`, `matplotlib.pyplot` as `plt`, and `numpy` as `np`. It loads the MNIST dataset, converts class labels to binary class matrices, and creates a list of models to train. The models include a model with 1 hidden layer and tanh activation, and a model with 1 hidden layer and sigmoid activation.

Bottom Screenshot:

- File Explorer:** Similar to the top screenshot, showing the file system structure.
- Code:** The code continues the training process. It adds more models to the list, including a model with 2 hidden layers and tanh activation, and a model with 2 hidden layers and sigmoid activation. It then trains each model and plots loss and accuracy curves.

NNDL_ICP3_700762893.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

- sbin
- srv
- sys
- tmp
- tools
 - google-cloud-sdk
 - node
- usr
- var
- Breast Cancer.csv
- NGC-DL-CONTAINER-LICEN...
- NN&DeepLearning_Summer_S...
- hasinOPrv

Code

```

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))

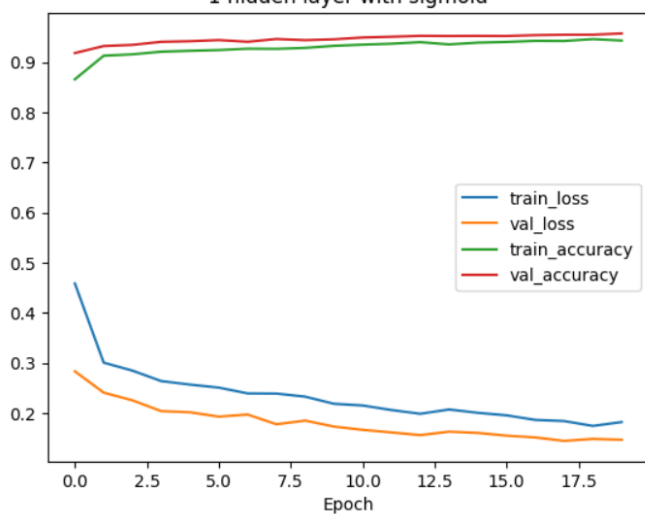
# train each model and plot loss and accuracy curves
for name, model in models:
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                        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))

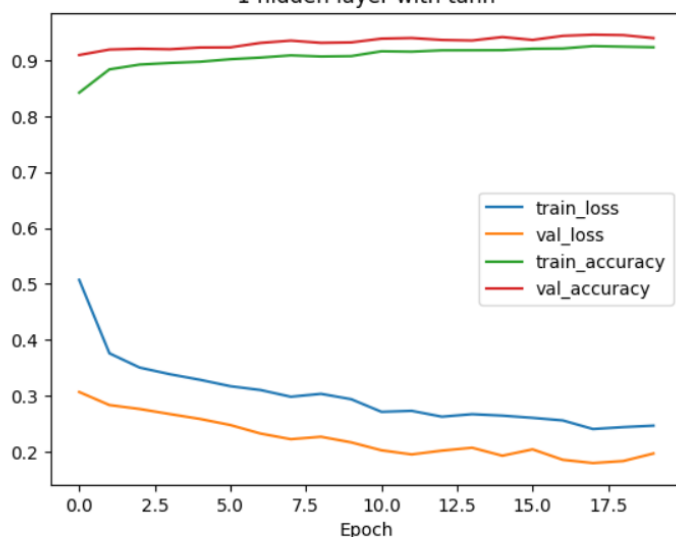
```

1 hidden layer with sigmoid



1 hidden layer with sigmoid - Test loss: 0.1469, Test accuracy: 0.9573

1 hidden layer with tanh



1 hidden layer with tanh - Test loss: 0.1967, Test accuracy: 0.9399

