# NEURAL NETWORKS AND DEEP LEARNING

ASSIGNMENT – 6

NAME – MANDA SUVIDHA REDDY

EMAIL – SXM95990@UCMO.EDU

STUDENT ID – 700729599

1. Use the use case in the class:

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

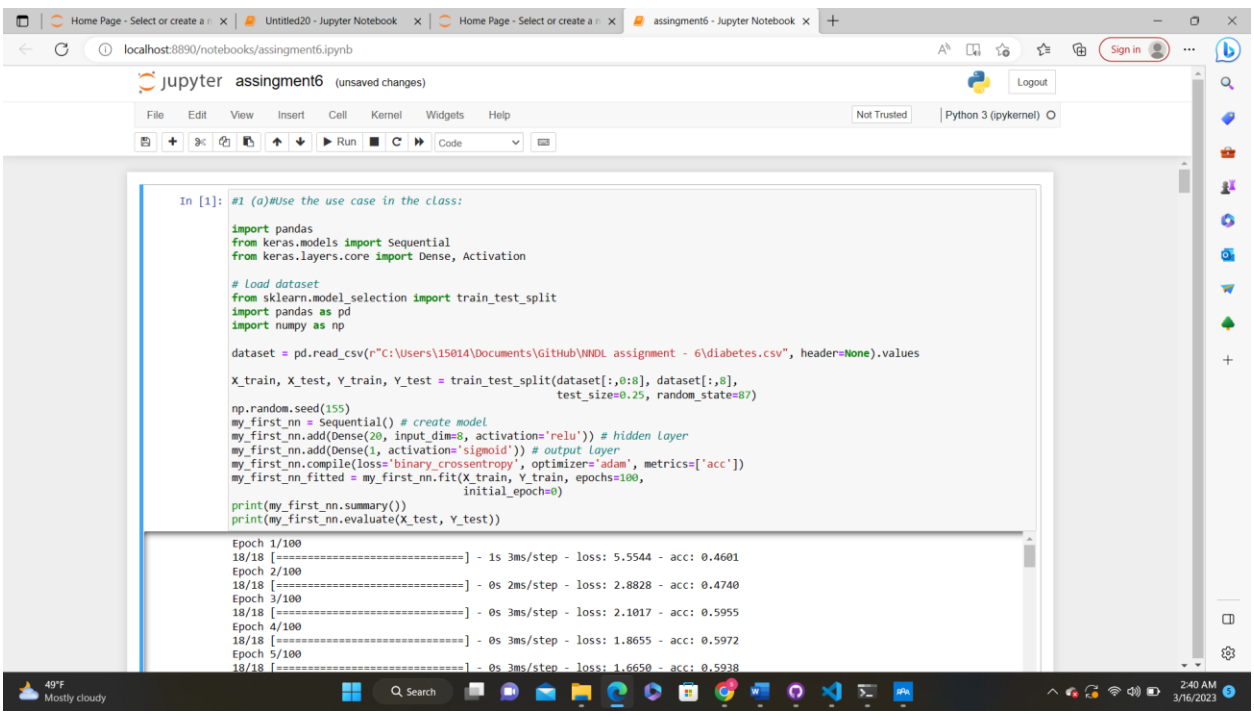
2. Change the data source to Breast Cancer dataset * available in the source code folder and make required changes. Report accuracy of the model.

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()

Breast Cancer dataset is designated to predict if a patient has Malignant (M) or Benign = B cancer.



Home Page - Select or create a n | Untitled20 - Jupyter Notebook | Home Page - Select or create a n | assingment6 - Jupyter Notebook

localhost:8890/notebooks/assingment6.ipynb

jupyter assingment6 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
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))
```

```
18/18 [==============================] - 0s 997us/step - loss: 0.5501 - acc: 0.7344
Epoch 100/100
18/18 [==============================] - 0s 994us/step - loss: 0.5461 - acc: 0.7361
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 20)                180

 dense_1 (Dense)             (None, 1)                 21

=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 2ms/step - loss: 0.5877 - acc: 0.6927
[0.5877408981323242, 0.6927083134651184]
```

In [2]:
```
#1(a). Add more Dense layers to the existing code and check how the accuracy changes.
# added more dense layers to the above existing code
import pandas
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# load dataset
```

49°F Mostly cloudy | Search | 2:41 AM 3/16/2023

Home Page - Select or create a n | Untitled20 - Jupyter Notebook | Home Page - Select or create a n | assingment6 - Jupyter Notebook

localhost:8890/notebooks/assingment6.ipynb

jupyter assingment6 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

In [2]:
```
#1(a). Add more Dense layers to the existing code and check how the accuracy changes.
# added more dense layers to the above existing code
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(r"C:\Users\15014\Documents\GitHub\NNDL assignment - 6\diabetes.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(10, activation='relu')) # additional hidden layer with node 10
my_first_nn.add(Dense(5, activation='relu')) # additional hidden layer with node 5
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))
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 20)                180

 dense_3 (Dense)             (None, 10)                210

 dense_4 (Dense)             (None, 5)                 55
```

49°F Mostly cloudy | Search | 2:41 AM 3/16/2023

```
my_first_nn = Sequential() # create model
my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu')) # additional hidden layer with node 10
my_first_nn.add(Dense(5, activation='relu')) # additional hidden layer with node 5
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))
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 20)                180

 dense_3 (Dense)             (None, 10)                210

 dense_4 (Dense)             (None, 5)                 55

 dense_5 (Dense)             (None, 1)                 6

=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6385 - acc: 0.6510
[0.6384992003440857, 0.6510416865348816]
```

In [3]:
```
#1(b) Change the data source to Breast Cancer dataset * available in the source code folder and make required
#changes. Report accuracy of the model.
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
```

In [3]:
```
#1(b) Change the data source to Breast Cancer dataset * available in the source code folder and make required
#changes. Report accuracy of the model.
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
#read the data
data = pd.read_csv(r'C:\Users\15014\Documents\GitHub\NNDL assignment - 6\breastcancer.csv')

data = load_breast_cancer()
X = data.data
Y = data.target
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
Epoch 1/100
14/14 [==============================] - 1s 3ms/step - loss: 11.5298 - accuracy: 0.4178
Epoch 2/100
14/14 [==============================] - 0s 3ms/step - loss: 1.6208 - accuracy: 0.7207
Epoch 3/100
14/14 [==============================] - 0s 3ms/step - loss: 0.5905 - accuracy: 0.8169
Epoch 4/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4143 - accuracy: 0.8615
Epoch 5/100
```

Home Page - Select or create a n | Untitled20 - Jupyter Notebook | Home Page - Select or create a n | assingment6 - Jupyter Notebook

localhost:8890/notebooks/assingment6.ipynb

jupyter assingment6 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=87)
np.random.seed(155)
model = Sequential()
model.add(Dense(20, input_dim=30, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, initial_epoch=0)
loss, accuracy = model.evaluate(X_test, Y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
Epoch 93/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1570 - accuracy: 0.9390
Epoch 94/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1724 - accuracy: 0.9366
Epoch 95/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1510 - accuracy: 0.9413
Epoch 96/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1846 - accuracy: 0.9319
Epoch 97/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1529 - accuracy: 0.9366
Epoch 98/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1444 - accuracy: 0.9319
Epoch 99/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1456 - accuracy: 0.9343
Epoch 100/100
14/14 [==============================] - 0s 1ms/step - loss: 0.1761 - accuracy: 0.9202
5/5 [==============================] - 0s 3ms/step - loss: 0.3655 - accuracy: 0.8531
Test Loss: 0.36550605297088623
Test Accuracy: 0.8531468510627747
```

In [4]:

```
#1(c) 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()
```

49°F Mostly cloudy | Search | 2:41 AM 3/16/2023

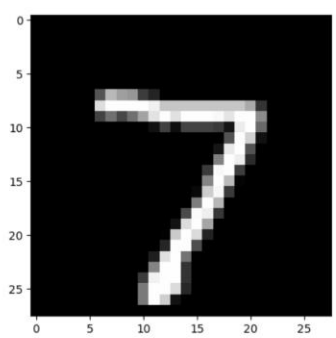
Home Page - Select or create a n | Untitled20 - Jupyter Notebook | Home Page - Select or create a n | assingment6 - Jupyter Notebook

localhost:8890/notebooks/assingment6.ipynb

jupyter assingment6 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
0.9840
Epoch 18/20
469/469 [==============================] - 14s 29ms/step - loss: 0.0172 - accuracy: 0.9944 - val_loss: 0.0756 - val_accuracy:
0.9832
Epoch 19/20
469/469 [==============================] - 13s 27ms/step - loss: 0.0139 - accuracy: 0.9954 - val_loss: 0.0757 - val_accuracy:
0.9836
Epoch 20/20
469/469 [==============================] - 11s 24ms/step - loss: 0.0131 - accuracy: 0.9955 - val_loss: 0.0679 - val_accuracy:
0.9829
```




```
1/1 [==============================] - 0s 82ms/step
```

49°F Mostly cloudy | Search | 2:42 AM 3/16/2023

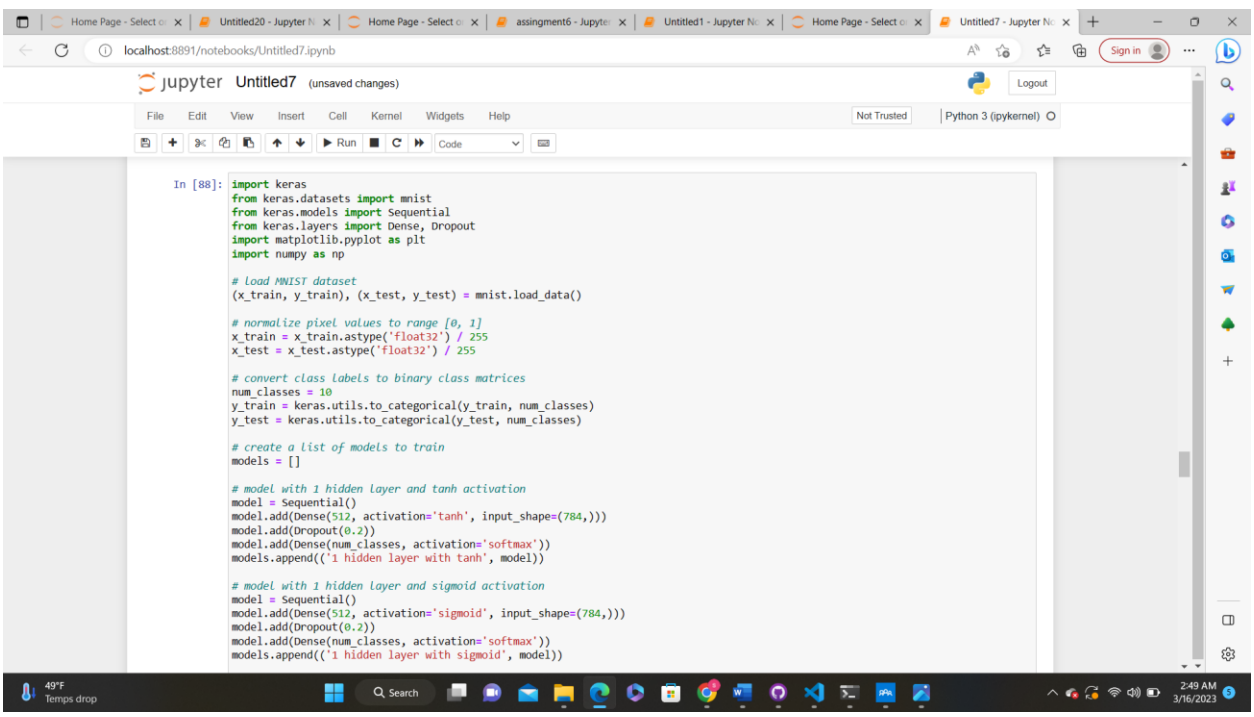
2. Use Image Classification on the hand written digits data set (mnist)

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

b. 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.

c. 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.

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



localhost:8891/notebooks/Untitled7.ipynb

jupyter Untitled7 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

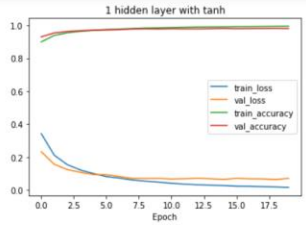
```python
# model with 2 hidden layers and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('2 hidden 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))
```

localhost:8891/notebooks/Untitled7.ipynb

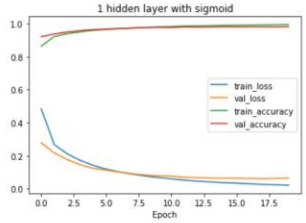
jupyter Untitled7 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

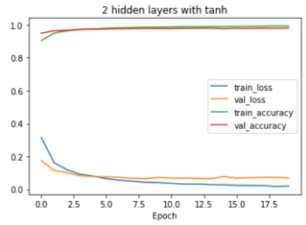



```
1 hidden layer with tanh - Test loss: 0.0716, Test accuracy: 0.9809
```

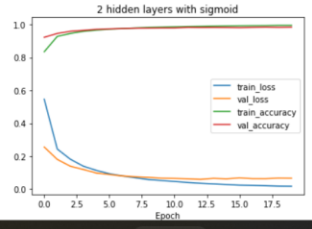



```
1 hidden layer with sigmoid - Test loss: 0.0642, Test accuracy: 0.9809
```

```
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
```




In [89]:

```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))
```

localhost:8891/notebooks/Untitled7.ipynb

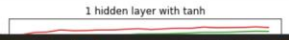
jupyter Untitled7 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

```
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



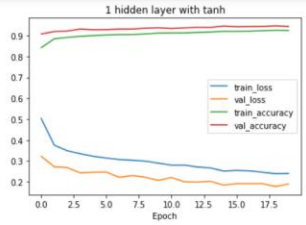
localhost:8891/notebooks/Untitled7.ipynb

jupyter Untitled7 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

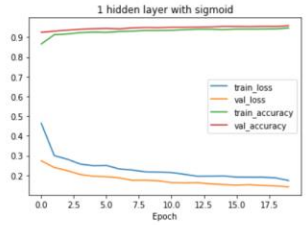
Not Trusted | Python 3 (ipykernel)

1 hidden layer with tanh

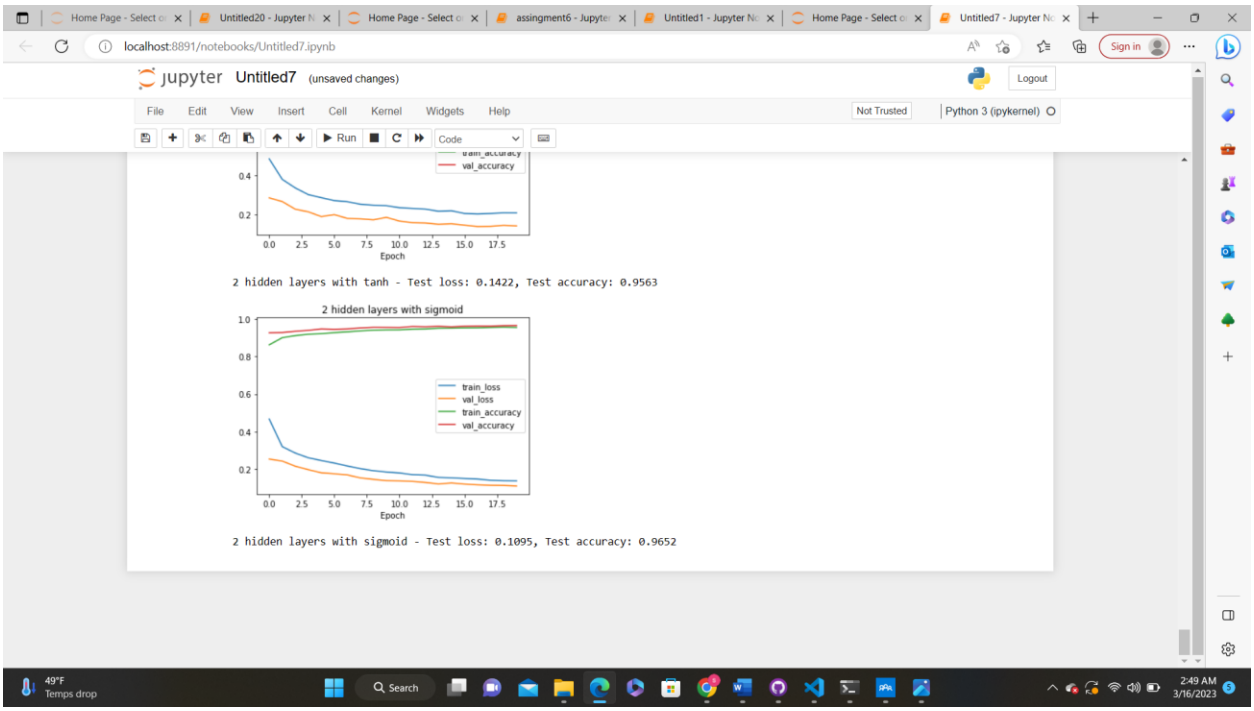



```
1 hidden layer with tanh - Test loss: 0.1895, Test accuracy: 0.9439
```

1 hidden layer with sigmoid




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




GIT REPO LINK -