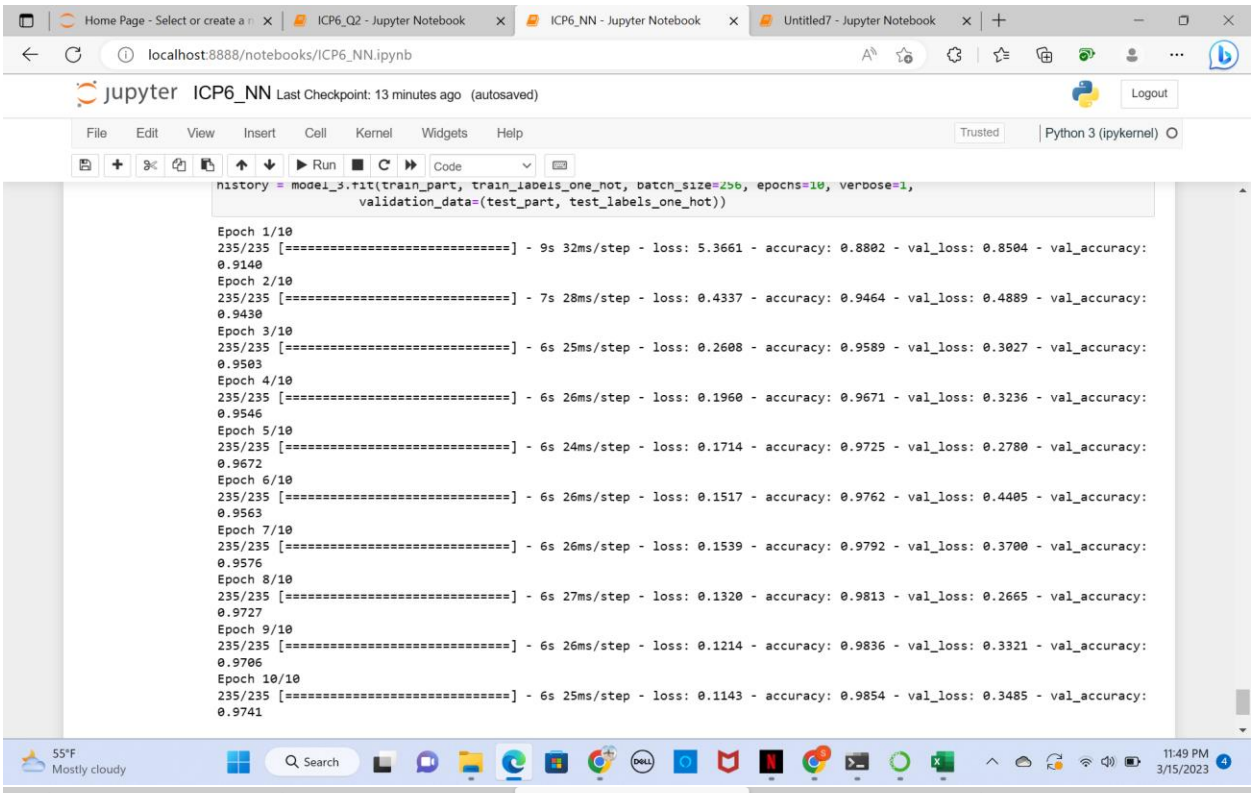
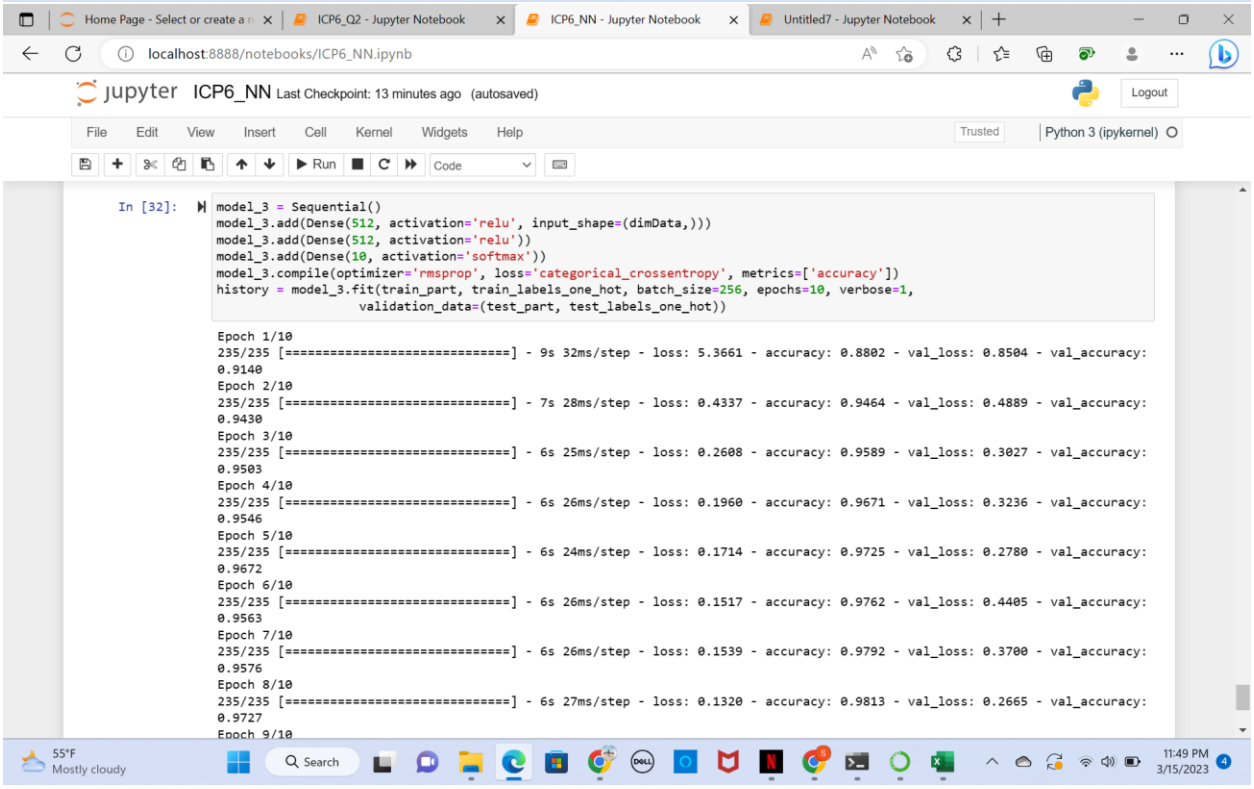
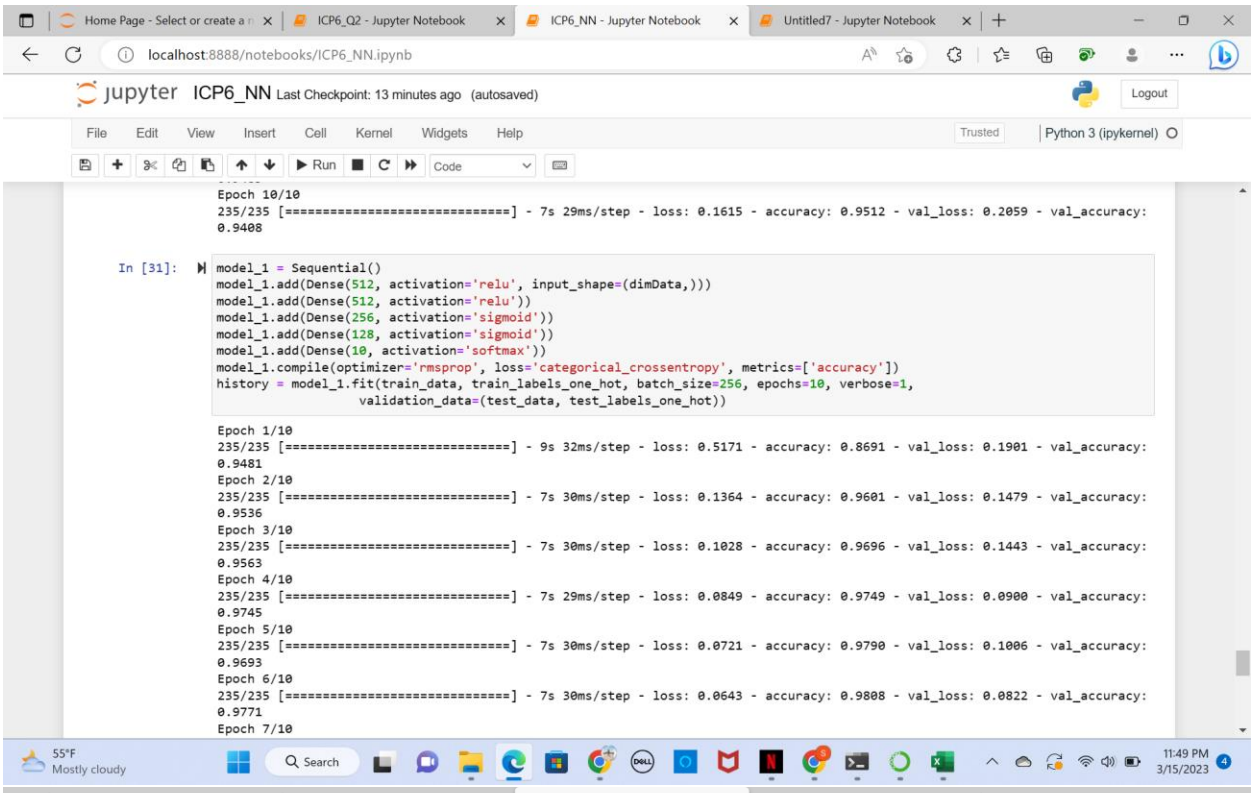
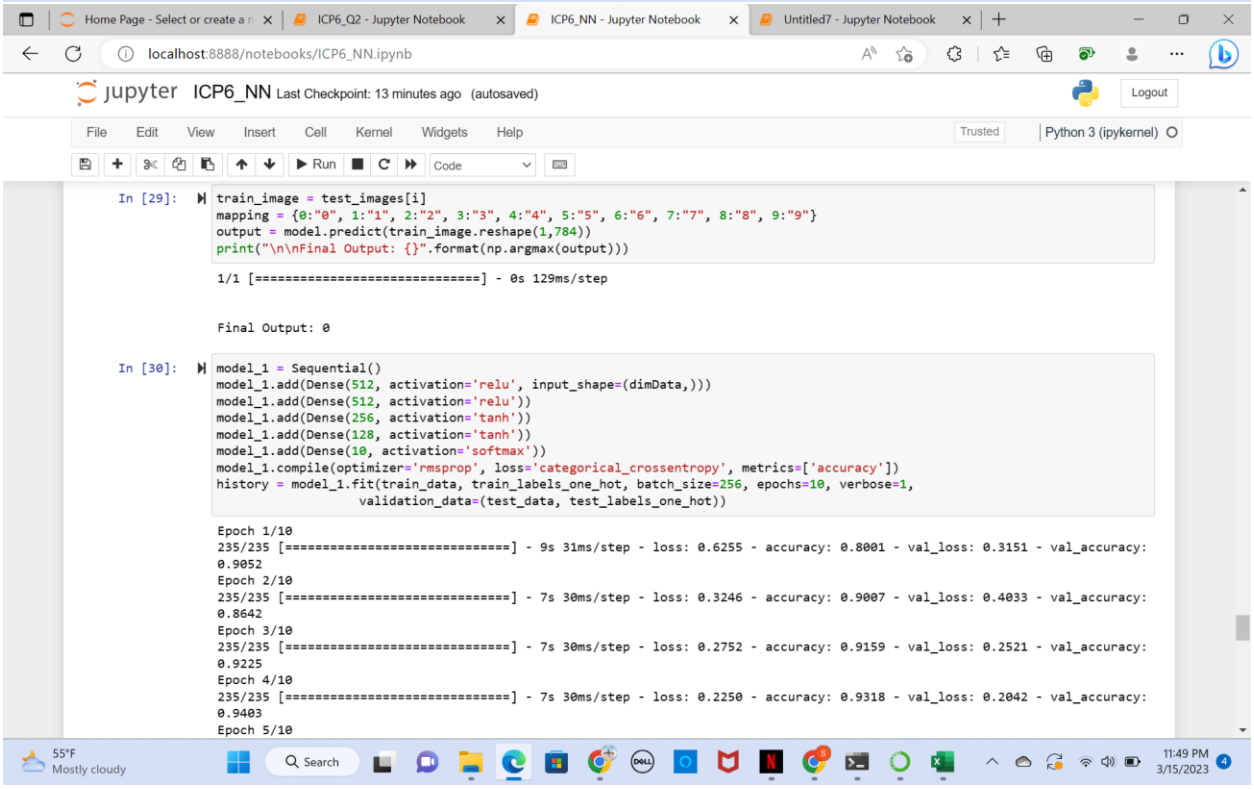
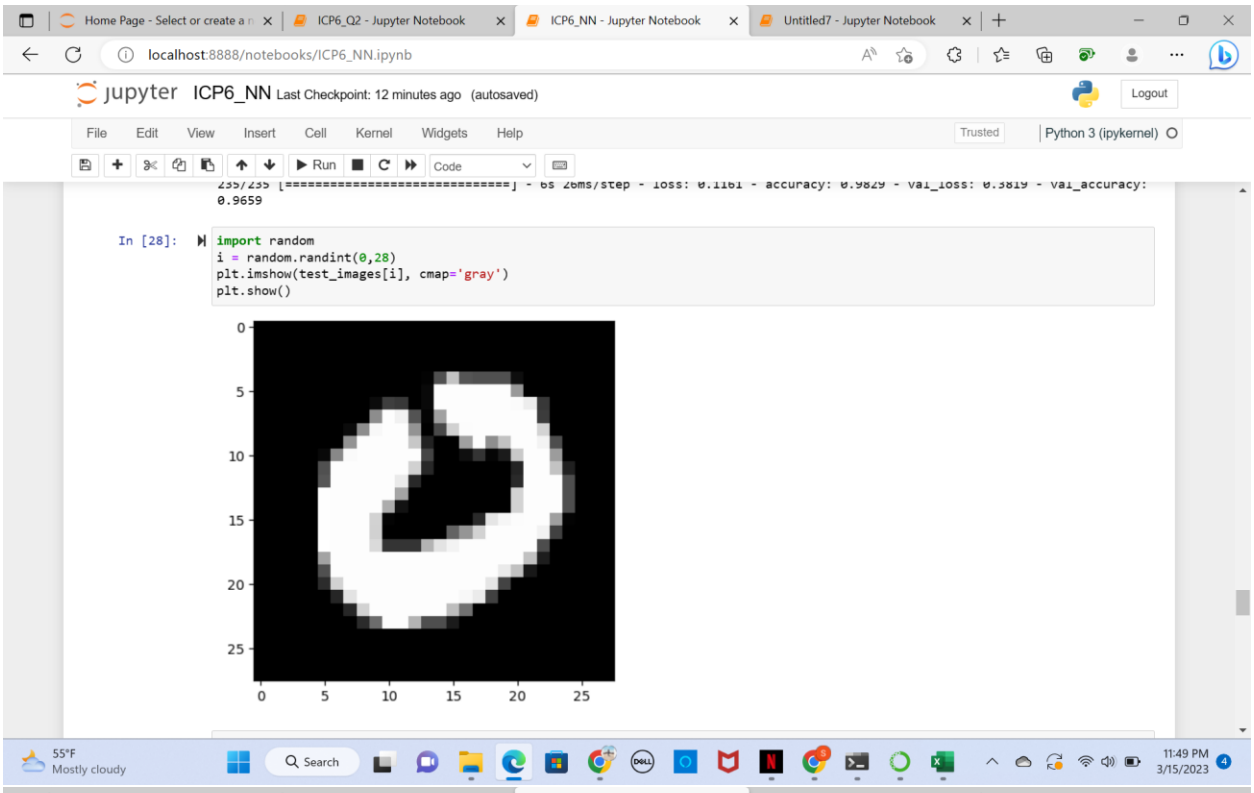
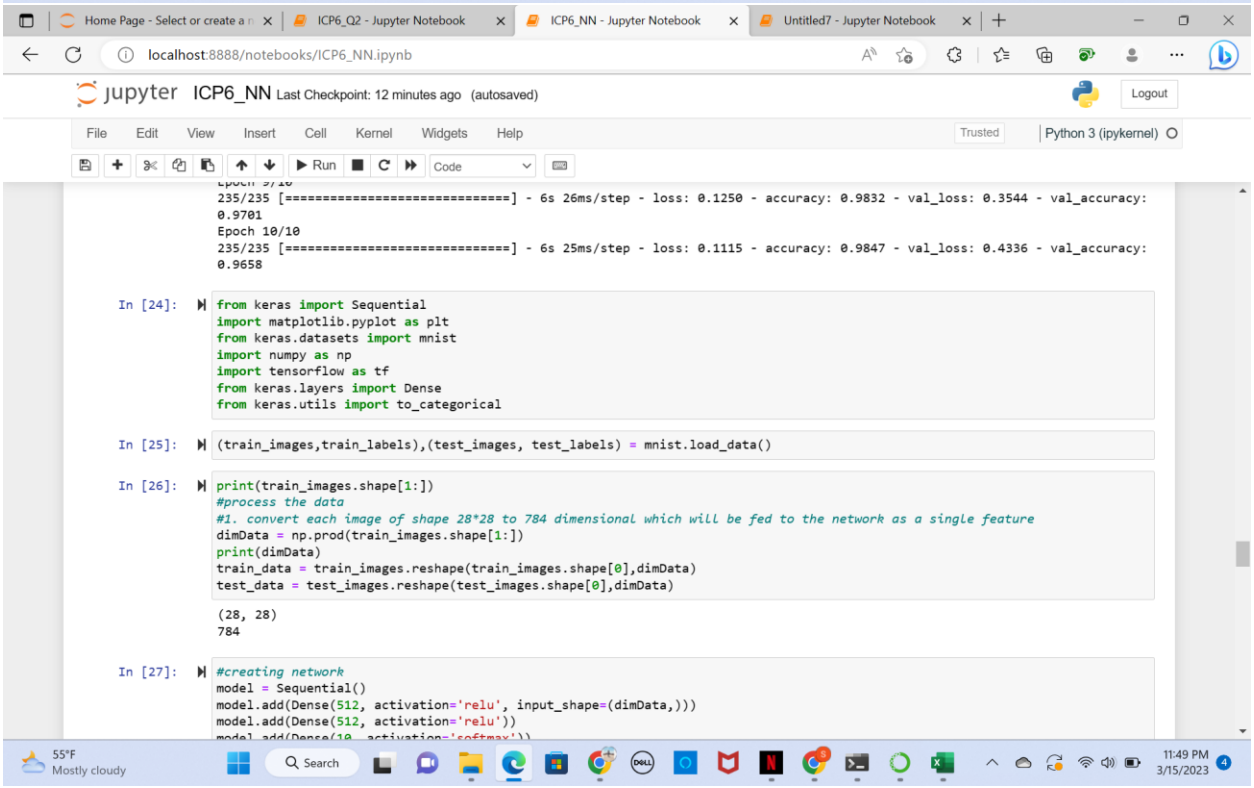

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
history = model_3.fit(train_part, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
validation_data=(test_part, test_labels_one_hot))
Epoch 1/10
235/235 [==========] - 9s 32ms/step - loss: 5.3661 - accuracy: 0.8802 - val_loss: 0.8504 - val_accuracy: 0.9140
Epoch 2/10
235/235 [==========] - 7s 28ms/step - loss: 0.4337 - accuracy: 0.9464 - val_loss: 0.4889 - val_accuracy: 0.9430
Epoch 3/10
235/235 [==========] - 6s 25ms/step - loss: 0.2608 - accuracy: 0.9589 - val_loss: 0.3027 - val_accuracy: 0.9503
Epoch 4/10
235/235 [==========] - 6s 26ms/step - loss: 0.1960 - accuracy: 0.9671 - val_loss: 0.3236 - val_accuracy: 0.9546
Epoch 5/10
235/235 [==========] - 6s 24ms/step - loss: 0.1714 - accuracy: 0.9725 - val_loss: 0.2780 - val_accuracy: 0.9672
Epoch 6/10
235/235 [==========] - 6s 26ms/step - loss: 0.1517 - accuracy: 0.9762 - val_loss: 0.4405 - val_accuracy: 0.9563
Epoch 7/10
235/235 [==========] - 6s 26ms/step - loss: 0.1539 - accuracy: 0.9792 - val_loss: 0.3700 - val_accuracy: 0.9576
Epoch 8/10
235/235 [==========] - 6s 27ms/step - loss: 0.1320 - accuracy: 0.9813 - val_loss: 0.2665 - val_accuracy: 0.9727
Epoch 9/10
235/235 [==========] - 6s 26ms/step - loss: 0.1214 - accuracy: 0.9836 - val_loss: 0.3321 - val_accuracy: 0.9706
Epoch 10/10
235/235 [==========] - 6s 25ms/step - loss: 0.1143 - accuracy: 0.9854 - val_loss: 0.3485 - val_accuracy: 0.9741
55°F Mostly cloudy
Search
11:49 PM 3/15/2023

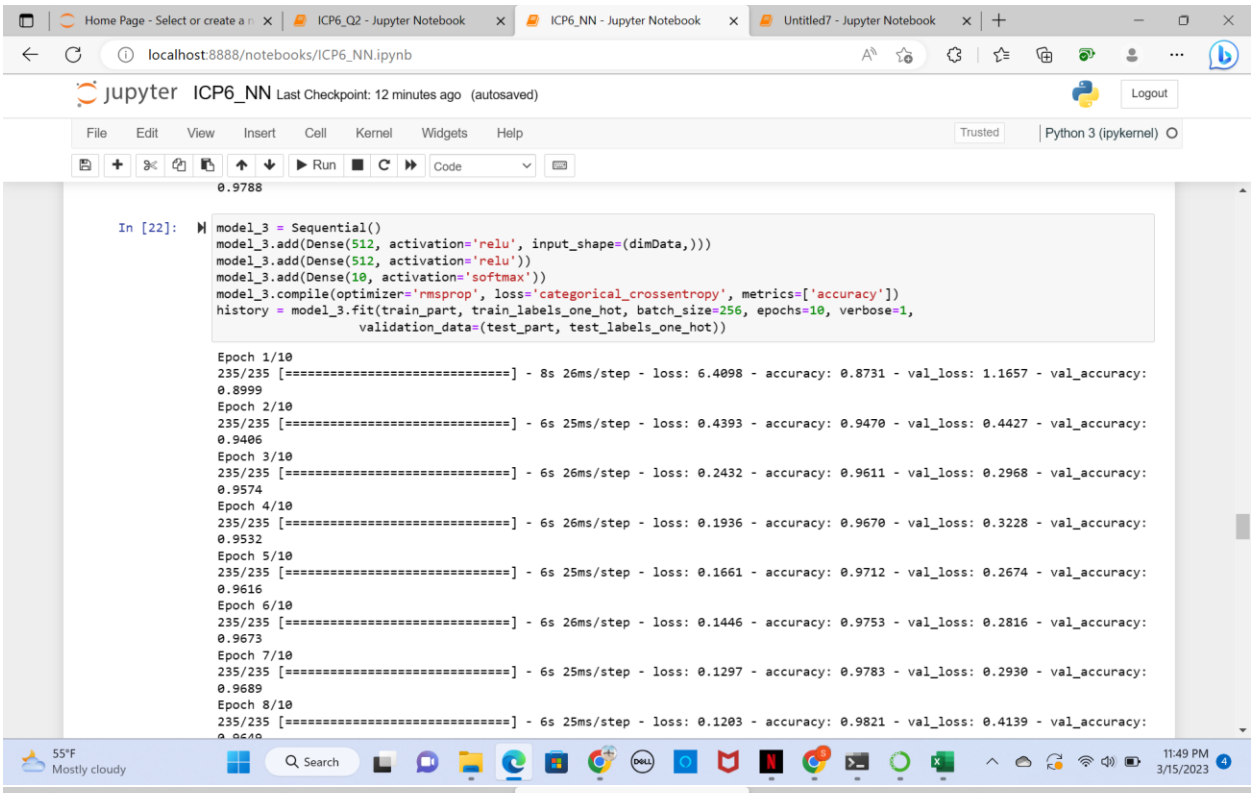

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [32]:
model_3 = Sequential()
model_3.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_3.add(Dense(512, activation='relu'))
model_3.add(Dense(10, activation='softmax'))
model_3.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_3.fit(train_part, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
validation_data=(test_part, test_labels_one_hot))
Epoch 1/10
235/235 [==========] - 9s 32ms/step - loss: 5.3661 - accuracy: 0.8802 - val_loss: 0.8504 - val_accuracy: 0.9140
Epoch 2/10
235/235 [==========] - 7s 28ms/step - loss: 0.4337 - accuracy: 0.9464 - val_loss: 0.4889 - val_accuracy: 0.9430
Epoch 3/10
235/235 [==========] - 6s 25ms/step - loss: 0.2608 - accuracy: 0.9589 - val_loss: 0.3027 - val_accuracy: 0.9503
Epoch 4/10
235/235 [==========] - 6s 26ms/step - loss: 0.1960 - accuracy: 0.9671 - val_loss: 0.3236 - val_accuracy: 0.9546
Epoch 5/10
235/235 [==========] - 6s 24ms/step - loss: 0.1714 - accuracy: 0.9725 - val_loss: 0.2780 - val_accuracy: 0.9672
Epoch 6/10
235/235 [==========] - 6s 26ms/step - loss: 0.1517 - accuracy: 0.9762 - val_loss: 0.4405 - val_accuracy: 0.9563
Epoch 7/10
235/235 [==========] - 6s 26ms/step - loss: 0.1539 - accuracy: 0.9792 - val_loss: 0.3700 - val_accuracy: 0.9576
Epoch 8/10
235/235 [==========] - 6s 27ms/step - loss: 0.1320 - accuracy: 0.9813 - val_loss: 0.2665 - val_accuracy: 0.9727
Epoch 9/10
55°F Mostly cloudy
Search
11:49 PM 3/15/2023

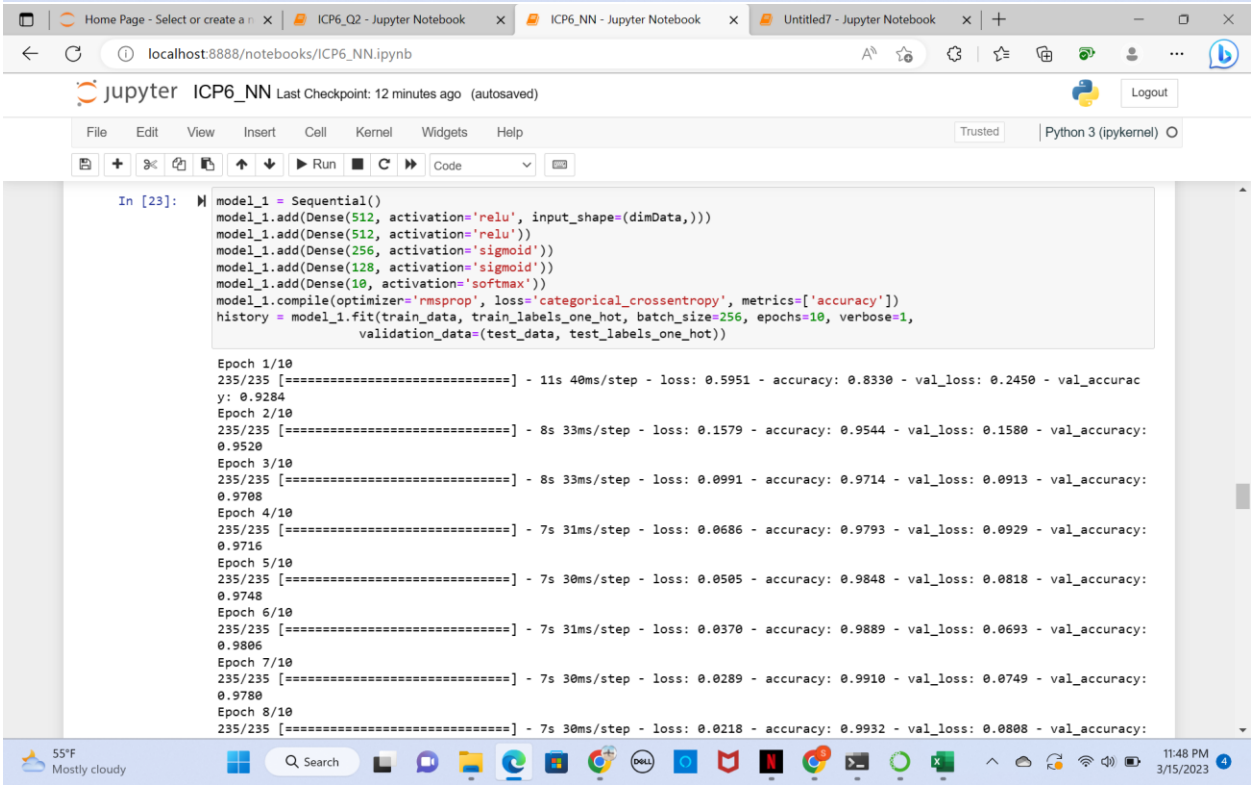

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Epoch 10/10
235/235 [==============================] - 7s 29ms/step - loss: 0.1615 - accuracy: 0.9512 - val_loss: 0.2059 - val_accuracy: 0.9408
In [31]:
model_1 = Sequential()
model_1.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_1.add(Dense(512, activation='relu'))
model_1.add(Dense(256, activation='sigmoid'))
model_1.add(Dense(128, activation='sigmoid'))
model_1.add(Dense(10, activation='softmax'))
model_1.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_1.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 [==============================] - 9s 32ms/step - loss: 0.5171 - accuracy: 0.8691 - val_loss: 0.1901 - val_accuracy: 0.9481
Epoch 2/10
235/235 [==============================] - 7s 30ms/step - loss: 0.1364 - accuracy: 0.9601 - val_loss: 0.1479 - val_accuracy: 0.9536
Epoch 3/10
235/235 [==============================] - 7s 30ms/step - loss: 0.1028 - accuracy: 0.9696 - val_loss: 0.1443 - val_accuracy: 0.9563
Epoch 4/10
235/235 [==============================] - 7s 29ms/step - loss: 0.0849 - accuracy: 0.9749 - val_loss: 0.0900 - val_accuracy: 0.9745
Epoch 5/10
235/235 [==============================] - 7s 30ms/step - loss: 0.0721 - accuracy: 0.9790 - val_loss: 0.1006 - val_accuracy: 0.9693
Epoch 6/10
235/235 [==============================] - 7s 30ms/step - loss: 0.0643 - accuracy: 0.9808 - val_loss: 0.0822 - val_accuracy: 0.9771
Epoch 7/10
55°F Mostly cloudy
Search
11:49 PM 3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 13 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [29]:
train_image = test_images[i]
mapping = {0:"0", 1:"1", 2:"2", 3:"3", 4:"4", 5:"5", 6:"6", 7:"7", 8:"8", 9:"9"}
output = model.predict(train_image.reshape(1,784))
print("\n\nFinal Output: {}".format(np.argmax(output)))
1/1 [==============================] - 0s 129ms/step
Final Output: 0
In [30]:
model_1 = Sequential()
model_1.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_1.add(Dense(512, activation='relu'))
model_1.add(Dense(256, activation='tanh'))
model_1.add(Dense(128, activation='tanh'))
model_1.add(Dense(10, activation='softmax'))
model_1.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_1.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 [==============================] - 9s 31ms/step - loss: 0.6255 - accuracy: 0.8001 - val_loss: 0.3151 - val_accuracy: 0.9052
Epoch 2/10
235/235 [==============================] - 7s 30ms/step - loss: 0.3246 - accuracy: 0.9007 - val_loss: 0.4033 - val_accuracy: 0.8642
Epoch 3/10
235/235 [==============================] - 7s 30ms/step - loss: 0.2752 - accuracy: 0.9159 - val_loss: 0.2521 - val_accuracy: 0.9225
Epoch 4/10
235/235 [==============================] - 7s 30ms/step - loss: 0.2250 - accuracy: 0.9318 - val_loss: 0.2042 - val_accuracy: 0.9403
Epoch 5/10
55°F Mostly cloudy
Search
11:49 PM 3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
0.9659
In [28]:
import random
i = random.randint(0,28)
plt.imshow(test_images[i], cmap='gray')
plt.show()
0
5
10
15
20
25
55°F
Mostly cloudy
Search
11:49 PM
3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
235/235 [==============================] - 6s 26ms/step - loss: 0.1250 - accuracy: 0.9832 - val_loss: 0.3544 - val_accuracy:
0.9701
Epoch 10/10
235/235 [==============================] - 6s 25ms/step - loss: 0.1115 - accuracy: 0.9847 - val_loss: 0.4336 - val_accuracy:
0.9658
In [24]:
from keras import Sequential
import matplotlib.pyplot as plt
from keras.datasets import mnist
import numpy as np
import tensorflow as tf
from keras.layers import Dense
from keras.utils import to_categorical
In [25]:
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()
In [26]:
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)
(28, 28)
784
In [27]:
#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
55°F
Mostly cloudy
Search
11:49 PM
3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
0.9788
In [22]:
model_3 = Sequential()
model_3.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_3.add(Dense(512, activation='relu'))
model_3.add(Dense(10, activation='softmax'))
model_3.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_3.fit(train_part, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
validation_data=(test_part, test_labels_one_hot))
Epoch 1/10
235/235 [==========] - 8s 26ms/step - loss: 6.4098 - accuracy: 0.8731 - val_loss: 1.1657 - val_accuracy: 0.8999
Epoch 2/10
235/235 [==========] - 6s 25ms/step - loss: 0.4393 - accuracy: 0.9470 - val_loss: 0.4427 - val_accuracy: 0.9406
Epoch 3/10
235/235 [==========] - 6s 26ms/step - loss: 0.2432 - accuracy: 0.9611 - val_loss: 0.2968 - val_accuracy: 0.9574
Epoch 4/10
235/235 [==========] - 6s 26ms/step - loss: 0.1936 - accuracy: 0.9670 - val_loss: 0.3228 - val_accuracy: 0.9532
Epoch 5/10
235/235 [==========] - 6s 25ms/step - loss: 0.1661 - accuracy: 0.9712 - val_loss: 0.2674 - val_accuracy: 0.9616
Epoch 6/10
235/235 [==========] - 6s 26ms/step - loss: 0.1446 - accuracy: 0.9753 - val_loss: 0.2816 - val_accuracy: 0.9673
Epoch 7/10
235/235 [==========] - 6s 25ms/step - loss: 0.1297 - accuracy: 0.9783 - val_loss: 0.2930 - val_accuracy: 0.9689
Epoch 8/10
235/235 [==========] - 6s 25ms/step - loss: 0.1203 - accuracy: 0.9821 - val_loss: 0.4139 - val_accuracy:
55°F Mostly cloudy
Search
11:49 PM 3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [23]:
model_1 = Sequential()
model_1.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_1.add(Dense(512, activation='relu'))
model_1.add(Dense(256, activation='sigmoid'))
model_1.add(Dense(128, activation='sigmoid'))
model_1.add(Dense(10, activation='softmax'))
model_1.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_1.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 [==========] - 11s 40ms/step - loss: 0.5951 - accuracy: 0.8330 - val_loss: 0.2450 - val_accuracy: 0.9284
Epoch 2/10
235/235 [==========] - 8s 33ms/step - loss: 0.1579 - accuracy: 0.9544 - val_loss: 0.1580 - val_accuracy: 0.9520
Epoch 3/10
235/235 [==========] - 8s 33ms/step - loss: 0.0991 - accuracy: 0.9714 - val_loss: 0.0913 - val_accuracy: 0.9708
Epoch 4/10
235/235 [==========] - 7s 31ms/step - loss: 0.0686 - accuracy: 0.9793 - val_loss: 0.0929 - val_accuracy: 0.9716
Epoch 5/10
235/235 [==========] - 7s 30ms/step - loss: 0.0505 - accuracy: 0.9848 - val_loss: 0.0818 - val_accuracy: 0.9748
Epoch 6/10
235/235 [==========] - 7s 31ms/step - loss: 0.0370 - accuracy: 0.9889 - val_loss: 0.0693 - val_accuracy: 0.9806
Epoch 7/10
235/235 [==========] - 7s 30ms/step - loss: 0.0289 - accuracy: 0.9910 - val_loss: 0.0749 - val_accuracy: 0.9780
Epoch 8/10
235/235 [==========] - 7s 30ms/step - loss: 0.0218 - accuracy: 0.9932 - val_loss: 0.0808 - val_accuracy:
55°F Mostly cloudy
Search
11:48 PM 3/15/2023

Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

0 5 10 15 20 25

In [20]:
```
train_image = test_images[i]
mapping = {0:"0", 1:"1", 2:"2", 3:"3", 4:"4", 5:"5", 6:"6", 7:"7", 8:"8", 9:"9"}
output = model.predict(train_image.reshape(1,784))
print("\n\nFinal Output: {}".format(np.argmax(output)))
```

```
1/1 [==============================] - 2s 2s/step


Final Output: 9
```

In [21]:
```
model_1 = Sequential()
model_1.add(Dense(512, activation='relu', input_shape=(dimData,)))
model_1.add(Dense(512, activation='relu'))
model_1.add(Dense(256, activation='tanh'))
model_1.add(Dense(128, activation='tanh'))
model_1.add(Dense(10, activation='softmax'))
model_1.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_1.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 [==============================] - 9s 33ms/step - loss: 0.3062 - accuracy: 0.9048 - val_loss: 0.1108 - val_accuracy:
0.9656
Epoch 2/10
235/235 [==============================] - 8s 34ms/step - loss: 0.0962 - accuracy: 0.9705 - val_loss: 0.1043 - val_accuracy:
0.9678
Epoch 3/10
235/235 [==============================] - 7s 30ms/step - loss: 0.0595 - accuracy: 0.9806 - val_loss: 0.1102 - val_accuracy:
0.9672
Epoch 4/10
235/235 [==============================] - 7s 31ms/step - loss: 0.0410 - accuracy: 0.9865 - val_loss: 0.0824 - val_accuracy:
```

55°F Mostly cloudy | Search | 11:48 PM 3/15/2023

Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved)

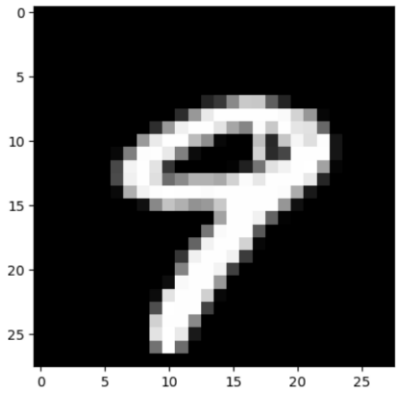
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [19]:
```
import random
i = random.randint(0,28)
plt.imshow(test_images[i], cmap='gray')
plt.show()
```



In [20]:
```
train_image = test_images[i]
```

55°F Mostly cloudy | Search | 11:48 PM 3/15/2023

Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

In [19]:
```
import random
```

55°F Mostly cloudy Search 11:48 PM 3/15/2023

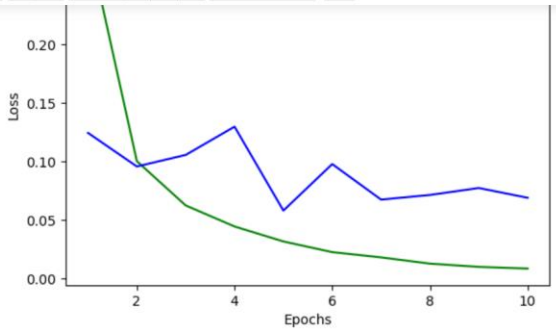
Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code



In [18]:
```
acc_train = history.history['accuracy']
acc_val = history.history['val_accuracy']
epochs = range(1,11)
plt.plot(epochs, acc_train,'g', label='Training accuracy')
plt.plot(epochs, acc_val,'b', label='validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Training and Validation accuracy

55°F Mostly cloudy Search 11:48 PM 3/15/2023

Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

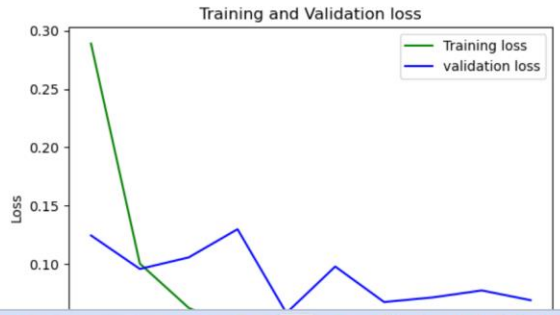
Trusted | Python 3 (ipykernel)

Run Code

```
Epoch 10/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0084 - accuracy: 0.9972 - val_loss: 0.0690 - val_accuracy:
0.9823
```

In [17]:

```
loss_train = history.history['loss']
loss_val = history.history['val_loss']
epochs = range(1,11)
plt.plot(epochs, loss_train,'g', label='Training loss')
plt.plot(epochs, loss_val,'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



55°F Mostly cloudy | Search | 11:48 PM 3/15/2023

Home Page - Select or create a n | ICP6_Q2 - Jupyter Notebook | ICP6_NN - Jupyter Notebook | Untitled7 - Jupyter Notebook

localhost:8888/notebooks/ICP6_NN.ipynb

jupyter ICP6_NN Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [15]:

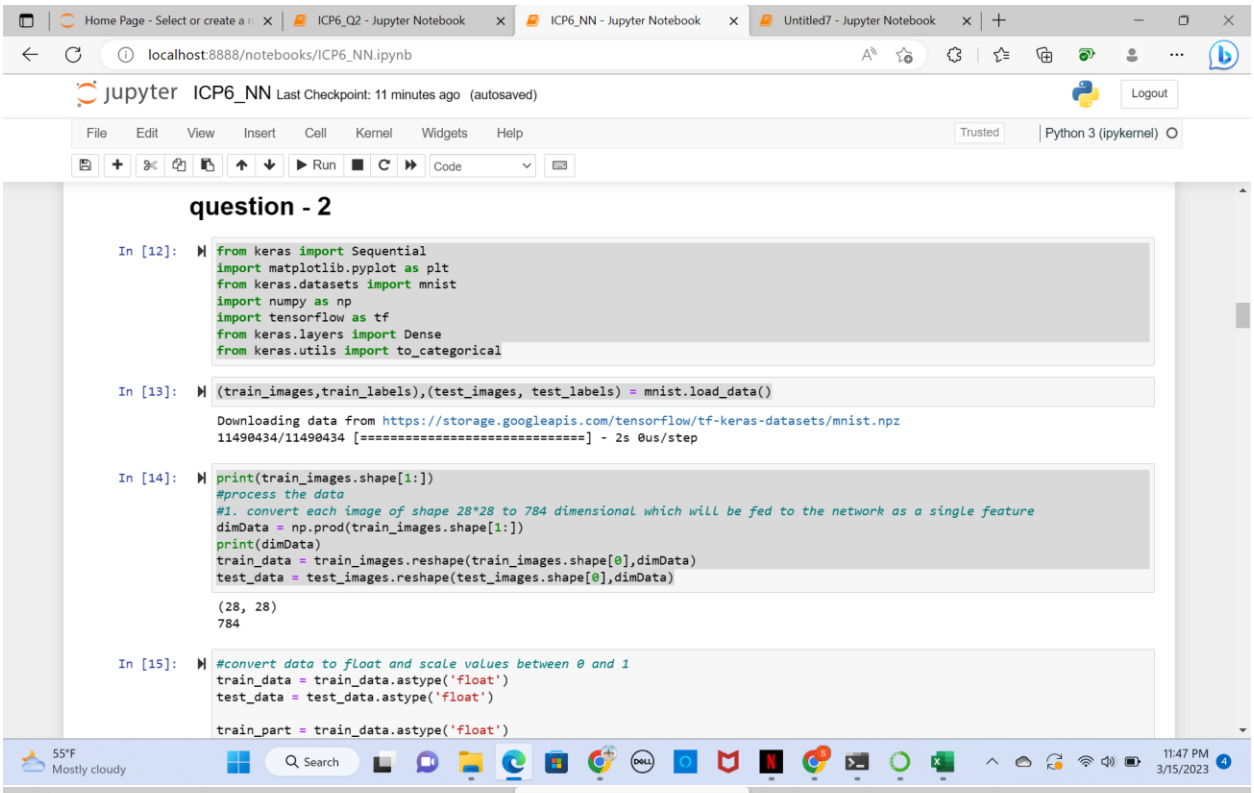
```
#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')

train_part = train_data.astype('float')
test_part = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
#change the labels frominteger 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)
```

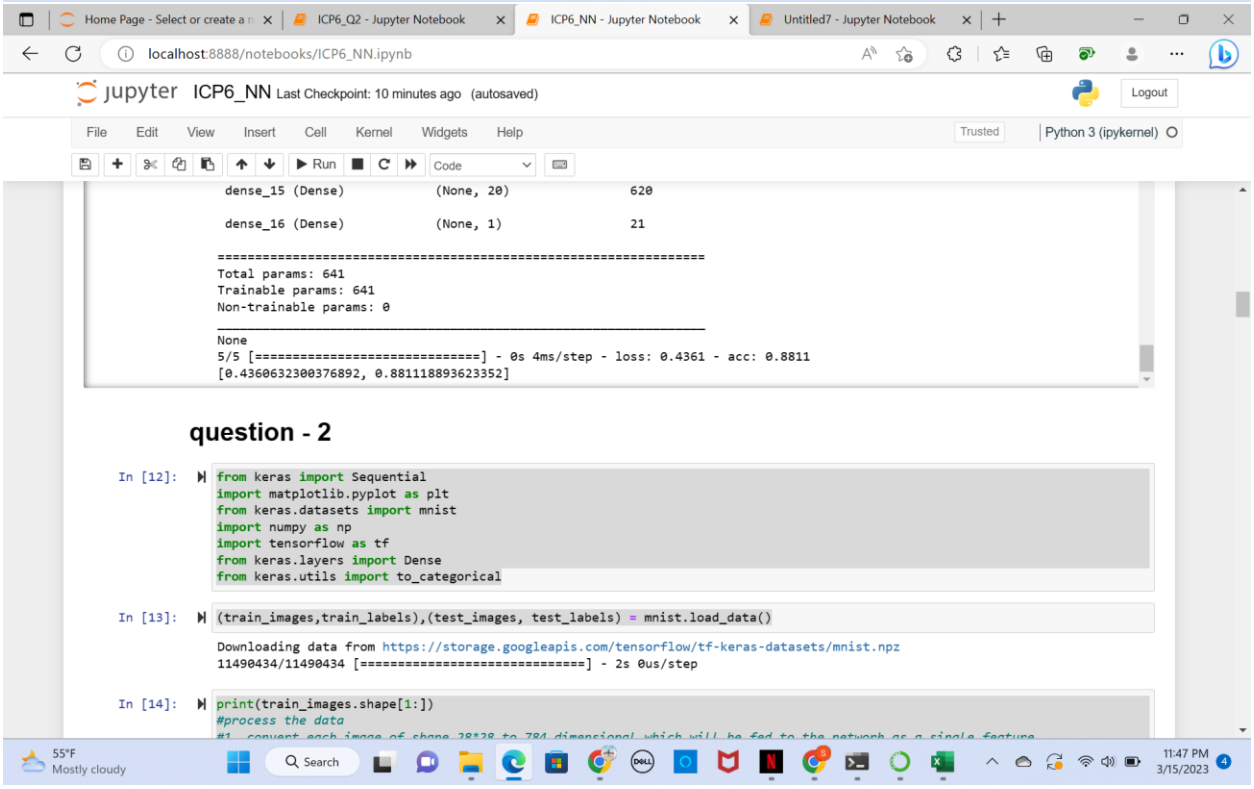
In [16]:

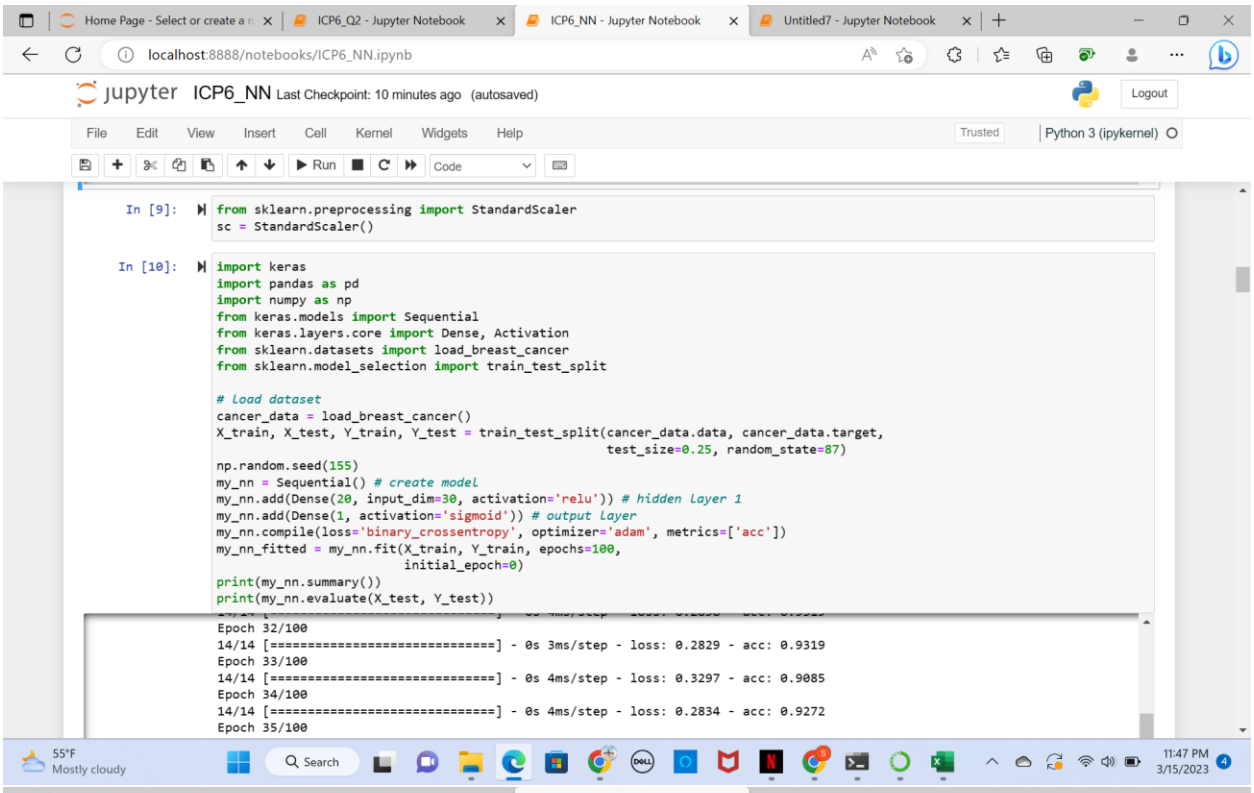
```
#creating network
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(dimData,)))
model.add(Dense(512, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

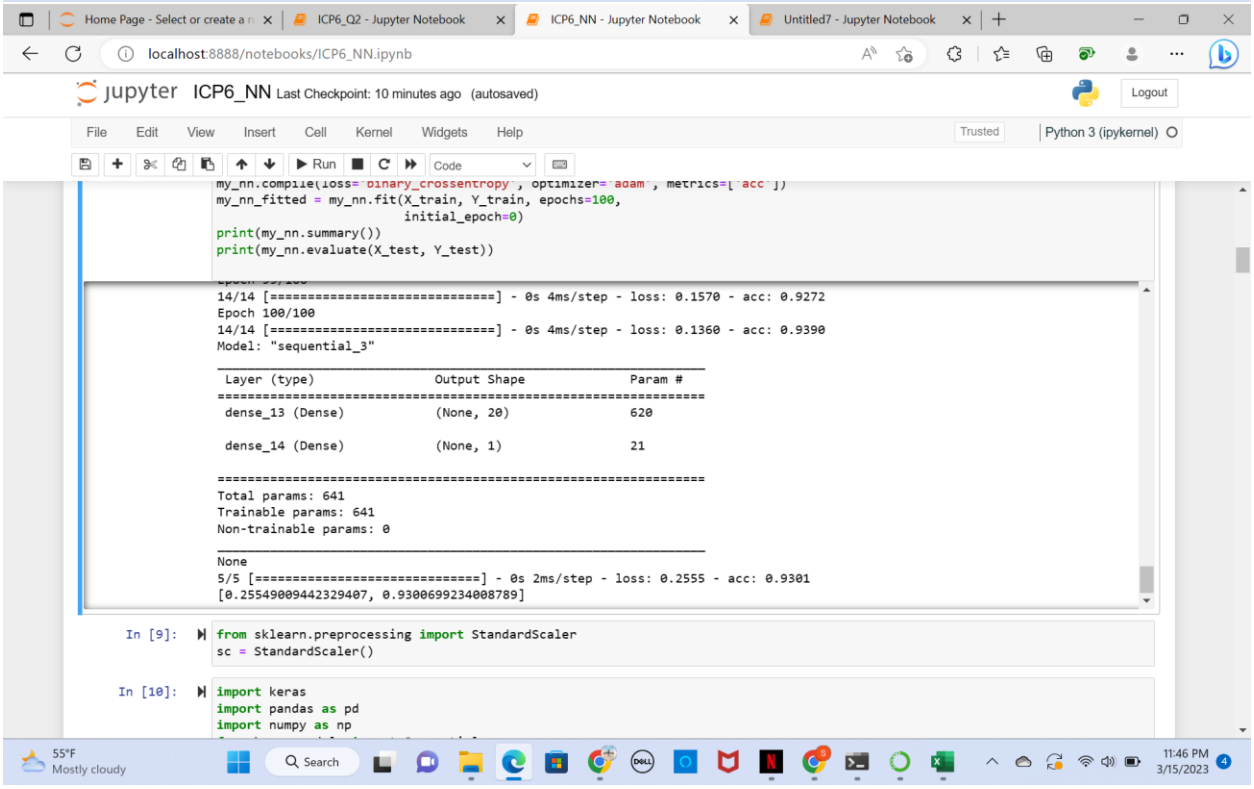
```
Epoch 1/10
235/235 [==============================] - 9s 31ms/step - loss: 0.2890 - accuracy: 0.9119 - val_loss: 0.1245 - val_accuracy:
0.9615
Epoch 2/10
235/235 [==============================] - 6s 27ms/step - loss: 0.1004 - accuracy: 0.9687 - val_loss: 0.0958 - val_accuracy:
0.9696
Epoch 3/10
235/235 [==============================] - 6s 25ms/step - loss: 0.0624 - accuracy: 0.9808 - val_loss: 0.1057 - val_accuracy:
0.9647
Epoch 4/10
235/235 [==============================] - 6s 26ms/step - loss: 0.0444 - accuracy: 0.9856 - val_loss: 0.1289 - val_accuracy:
```

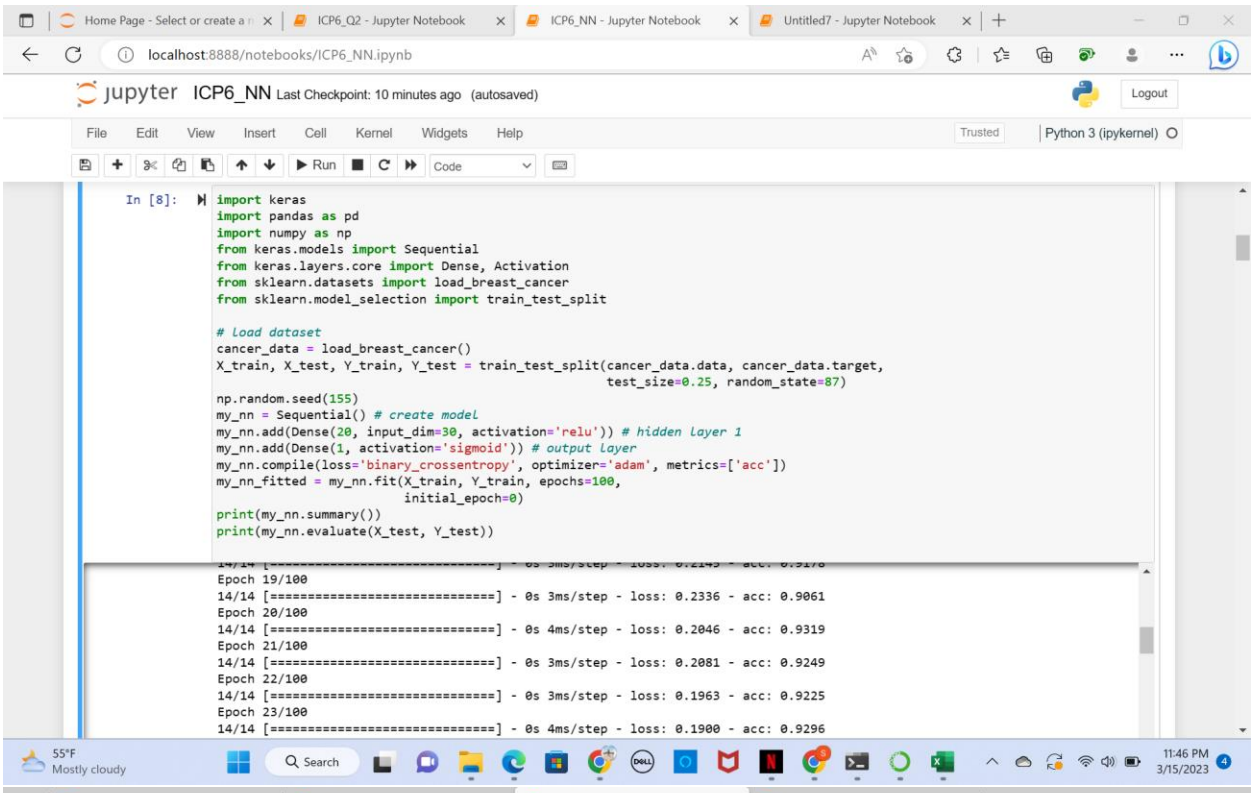
55°F Mostly cloudy | Search | 11:47 PM 3/15/2023

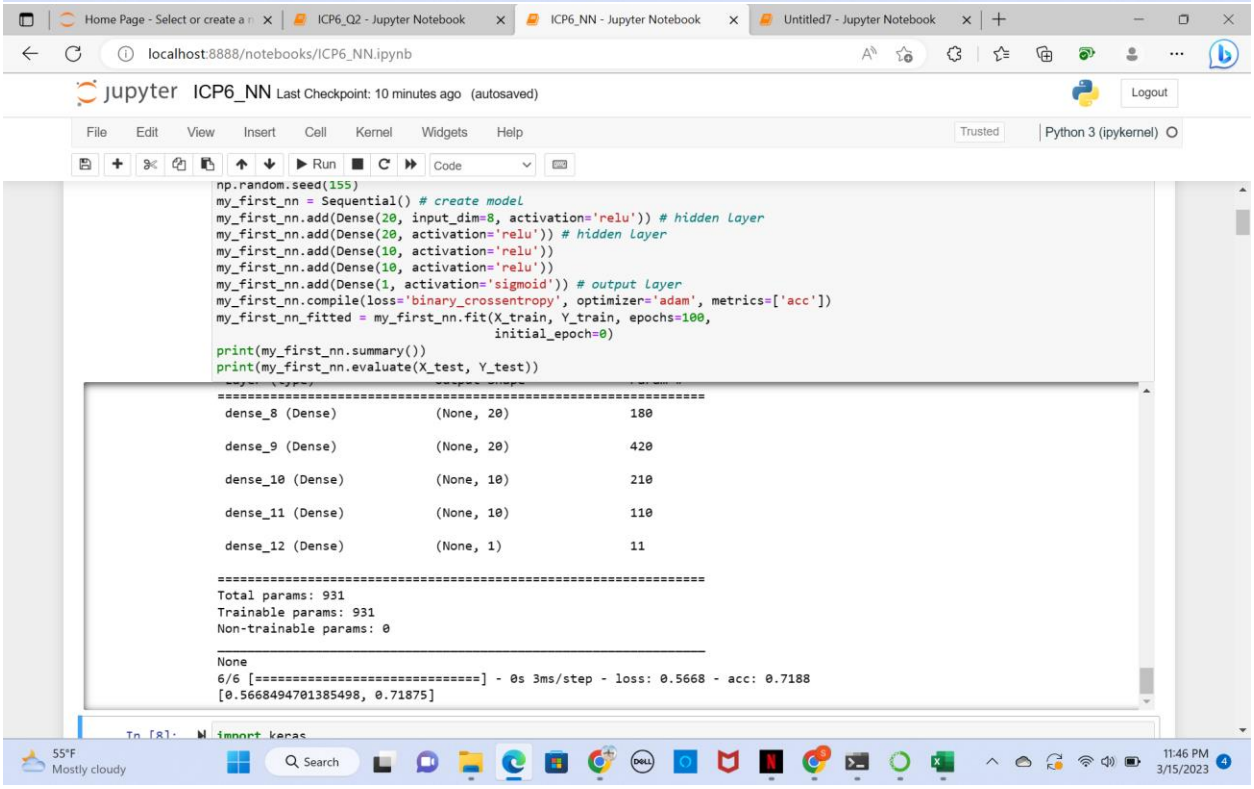

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
question - 2
In [12]:
from keras import Sequential
import matplotlib.pyplot as plt
from keras.datasets import mnist
import numpy as np
import tensorflow as tf
from keras.layers import Dense
from keras.utils import to_categorical
In [13]:
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
In [14]:
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)
(28, 28)
784
In [15]:
#convert data to float and scale values between 0 and 1
train_data = train_data.astype('float')
test_data = test_data.astype('float')
train_part = train_data.astype('float')
55°F
Mostly cloudy
Search
11:47 PM
3/15/2023

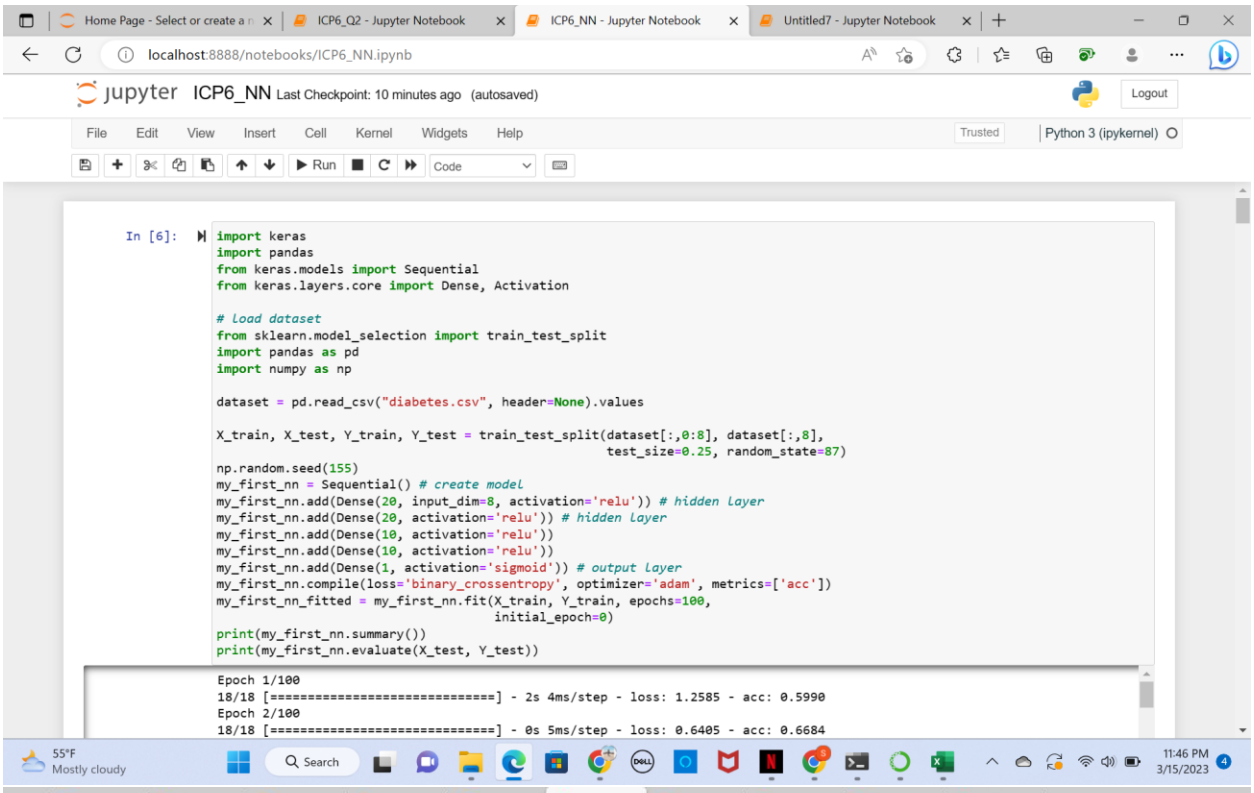

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
dense_15 (Dense) (None, 20) 620
dense_16 (Dense) (None, 1) 21
Total params: 641
Trainable params: 641
Non-trainable params: 0
None
5/5 [==============================] - 0s 4ms/step - loss: 0.4361 - acc: 0.8811
[0.4360632300376892, 0.881118893623352]
question - 2
In [12]:
from keras import Sequential
import matplotlib.pyplot as plt
from keras.datasets import mnist
import numpy as np
import tensorflow as tf
from keras.layers import Dense
from keras.utils import to_categorical
In [13]:
(train_images,train_labels),(test_images, test_labels) = mnist.load_data()
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
In [14]:
print(train_images.shape[1:])
#process the data
55°F
Mostly cloudy
Search
11:47 PM
3/15/2023

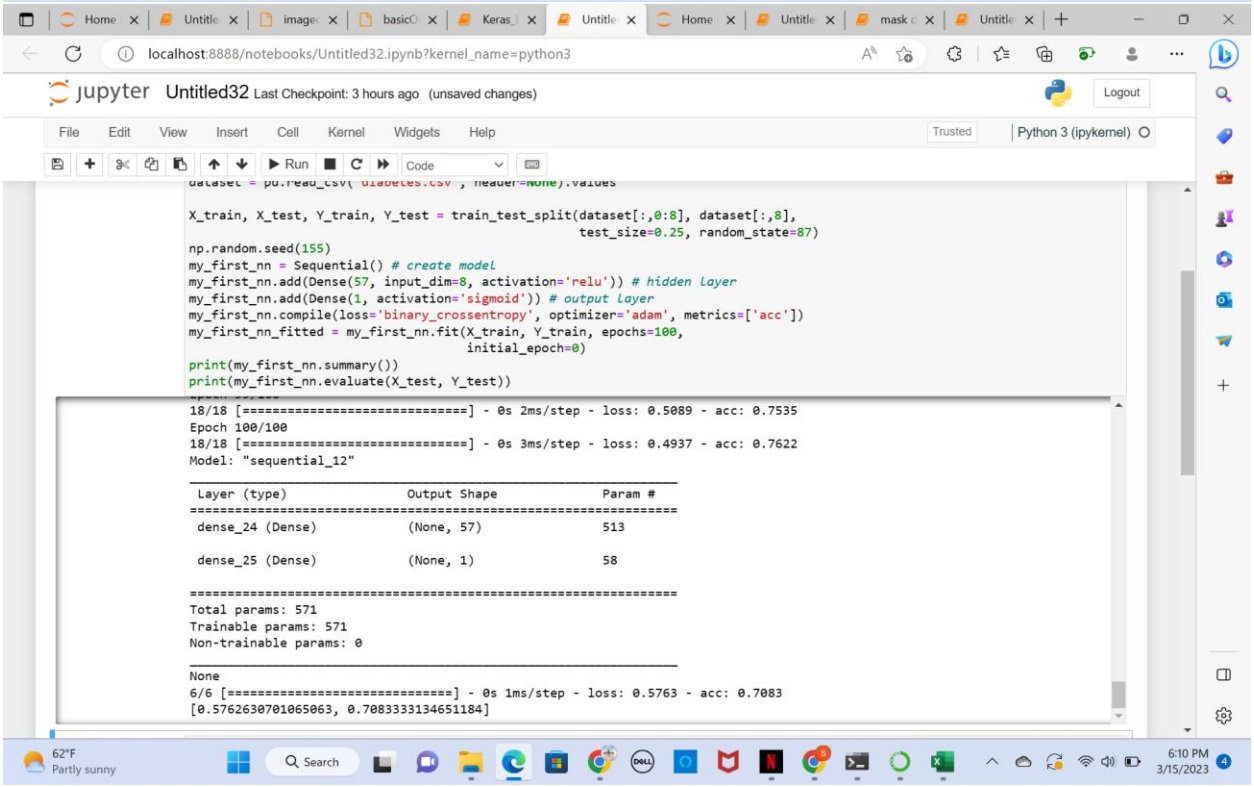

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [9]:
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
In [10]:
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))
Epoch 32/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2829 - acc: 0.9319
Epoch 33/100
14/14 [==============================] - 0s 4ms/step - loss: 0.3297 - acc: 0.9085
Epoch 34/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2834 - acc: 0.9272
Epoch 35/100
55°F
Mostly cloudy
Search
11:47 PM
3/15/2023

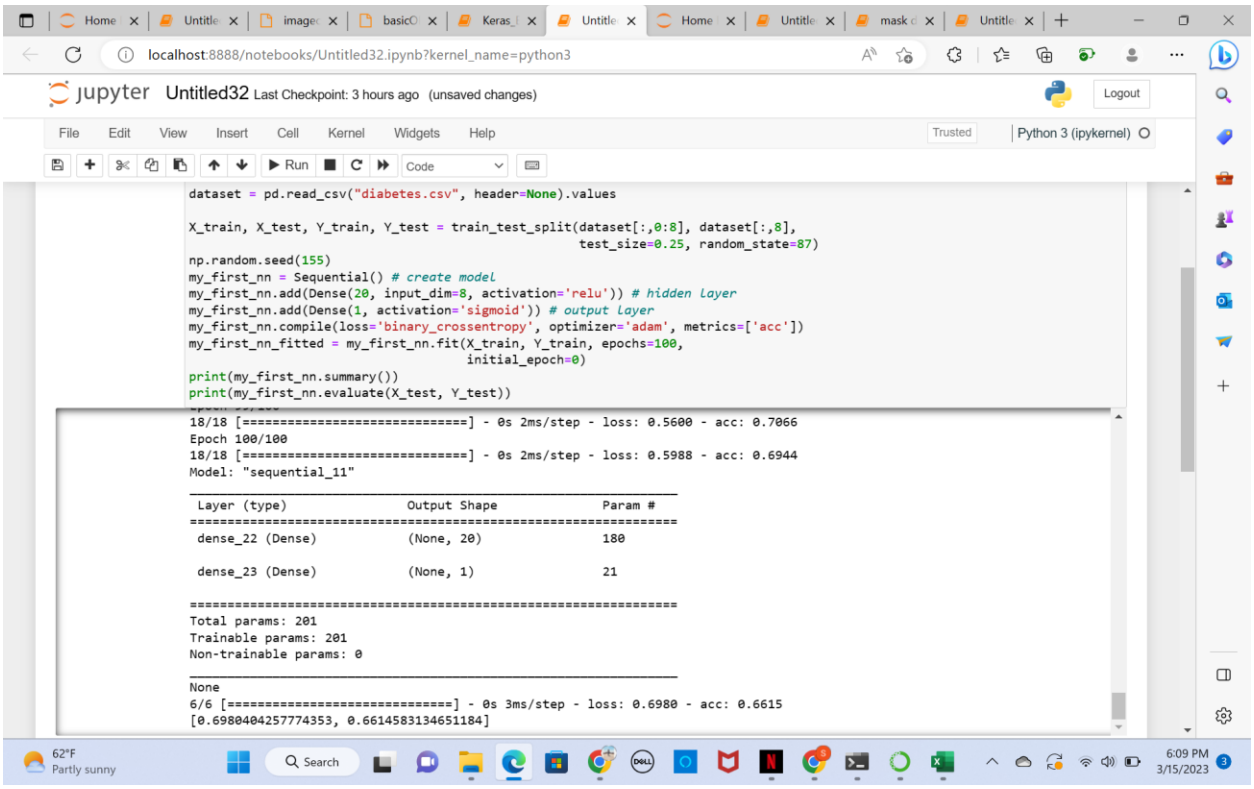

Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
my_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn.fit(X_train, Y_train, epochs=100,
initial_epoch=0)
print(my_nn.summary())
print(my_nn.evaluate(X_test, Y_test))
14/14 [==============================] - 0s 4ms/step - loss: 0.1570 - acc: 0.9272
Epoch 100/100
14/14 [==============================] - 0s 4ms/step - loss: 0.1360 - acc: 0.9390
Model: "sequential_3"
____
Layer (type) Output Shape Param #
===
dense_13 (Dense) (None, 20) 620
dense_14 (Dense) (None, 1) 21
===
Total params: 641
Trainable params: 641
Non-trainable params: 0
____
None
5/5 [==============================] - 0s 2ms/step - loss: 0.2555 - acc: 0.9301
[0.25549009442329407, 0.9300699234008789]
In [9]:
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
In [10]:
import keras
import pandas as pd
import numpy as np
55°F
Mostly cloudy
Search
11:46 PM
3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [8]:
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))
Epoch 19/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2336 - acc: 0.9061
Epoch 20/100
14/14 [==============================] - 0s 4ms/step - loss: 0.2046 - acc: 0.9319
Epoch 21/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2081 - acc: 0.9249
Epoch 22/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1963 - acc: 0.9225
Epoch 23/100
14/14 [==============================] - 0s 4ms/step - loss: 0.1900 - acc: 0.9296
55°F
Mostly cloudy
Search
11:46 PM
3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
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(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu'))
my_first_nn.add(Dense(10, activation='relu'))
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))
dense_8 (Dense) (None, 20) 180
dense_9 (Dense) (None, 20) 420
dense_10 (Dense) (None, 10) 210
dense_11 (Dense) (None, 10) 110
dense_12 (Dense) (None, 1) 11
Total params: 931
Trainable params: 931
Non-trainable params: 0
None
6/6 [==============================] - 0s 3ms/step - loss: 0.5668 - acc: 0.7188
[0.5668494701385498, 0.71875]
In [8]:
import keras
55°F
Mostly cloudy
Search
11:46 PM
3/15/2023


Home Page - Select or create a n
ICP6_Q2 - Jupyter Notebook
ICP6_NN - Jupyter Notebook
Untitled7 - Jupyter Notebook
localhost:8888/notebooks/ICP6_NN.ipynb
jupyter ICP6_NN Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [6]:
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("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(20, activation='relu')) # hidden layer
my_first_nn.add(Dense(10, activation='relu'))
my_first_nn.add(Dense(10, activation='relu'))
my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
initial_epoch=0)
print(my_first_nn.summary())
print(my_first_nn.evaluate(X_test, Y_test))
Epoch 1/100
18/18 [==============================] - 2s 4ms/step - loss: 1.2585 - acc: 0.5990
Epoch 2/100
18/18 [==============================] - 0s 5ms/step - loss: 0.6405 - acc: 0.6684
55°F
Mostly cloudy
Search
11:46 PM
3/15/2023


Home
Untitle
image
basicC
Keras_
Untitle
Home
Untitle
mask
Untitle
localhost:8888/notebooks/Untitled32.ipynb?kernel_name=python3
jupyter Untitled32 Last Checkpoint: 3 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
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(57, 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 2ms/step - loss: 0.5089 - acc: 0.7535
Epoch 100/100
18/18 [==============================] - 0s 3ms/step - loss: 0.4937 - acc: 0.7622
Model: "sequential_12"
Layer (type) Output Shape Param #
dense_24 (Dense) (None, 57) 513
dense_25 (Dense) (None, 1) 58
Total params: 571
Trainable params: 571
Non-trainable params: 0
None
6/6 [==============================] - 0s 1ms/step - loss: 0.5763 - acc: 0.7083
[0.5762630701065063, 0.7083333134651184]
62°F
Partly sunny
Search
6:10 PM
3/15/2023


localhost:8888/notebooks/Untitled32.ipynb?kernel_name=python3
jupyter Untitled32 Last Checkpoint: 3 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
dataset = pd.read_csv("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(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 2ms/step - loss: 0.5600 - acc: 0.7066
Epoch 100/100
18/18 [==============================] - 0s 2ms/step - loss: 0.5988 - acc: 0.6944
Model: "sequential_11"
Layer (type) Output Shape Param #
dense_22 (Dense) (None, 20) 180
dense_23 (Dense) (None, 1) 21
Total params: 201
Trainable params: 201
Non-trainable params: 0
None
6/6 [==============================] - 0s 3ms/step - loss: 0.6980 - acc: 0.6615
[0.6980404257774353, 0.6614583134651184]
62°F
Partly sunny
Search
6:09 PM
3/15/2023