Spring 2025: Neural Networks & Deep Learning – ICP -4

Assignment -4

Name: Vanitha Chintalapudi

Student ID: 700756782

Github Link: https://github.com/VanithaChintalapudi10/Neural-network-deep-learning

Video Link: https://drive.google.com/file/d/1KLPE31AGNYkE6Q1URdoAoHflhonV6YT/view?usp=drive_link

1) 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.

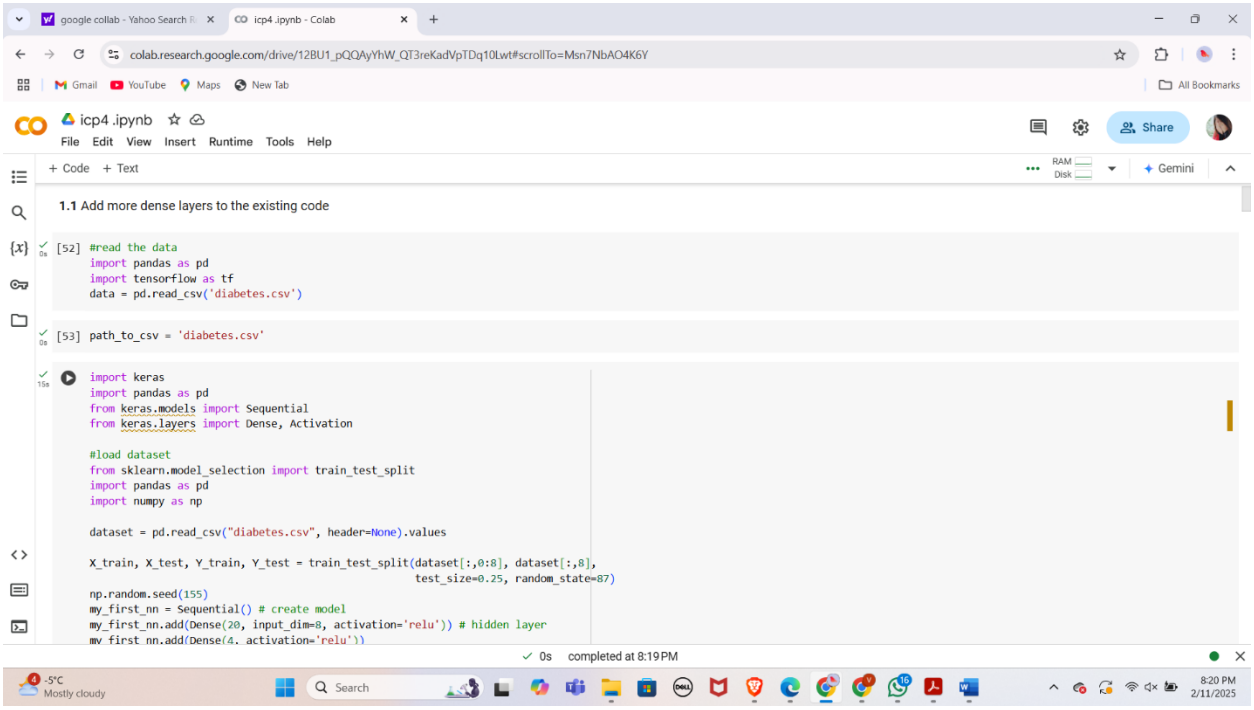
3. Normalize the data before feeding the data to the model and check how the normalization change your accuracy (code given below).

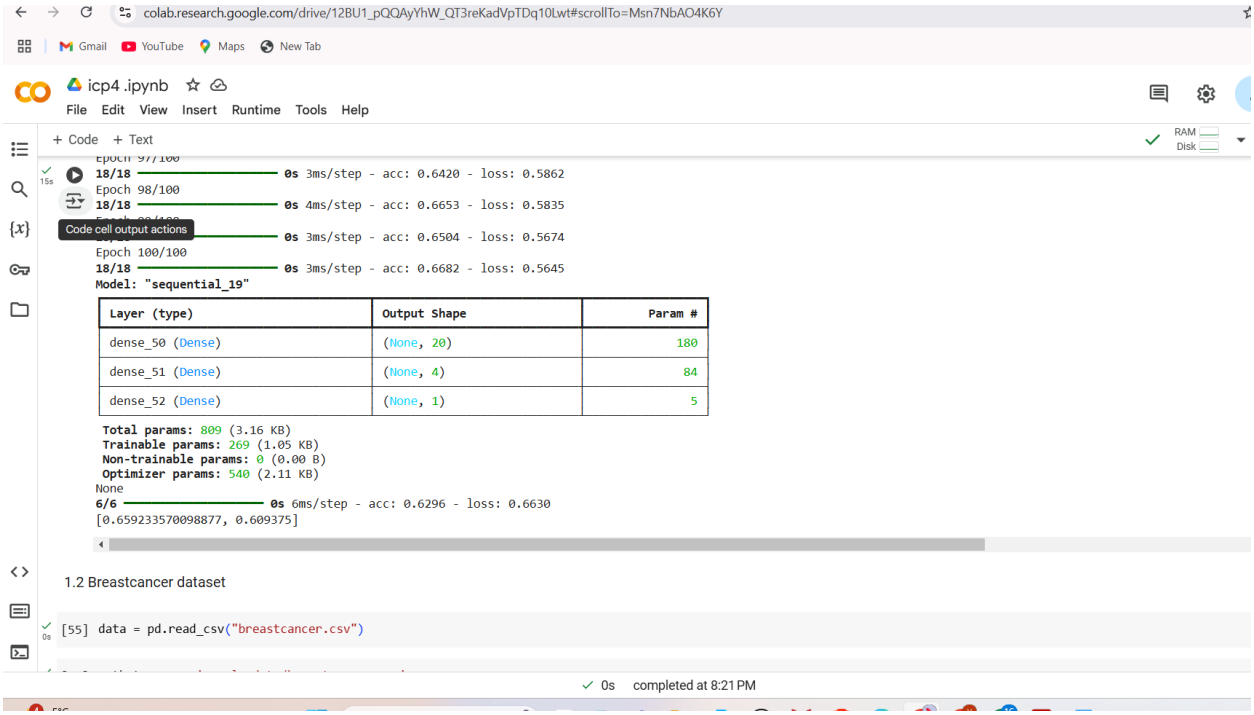
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

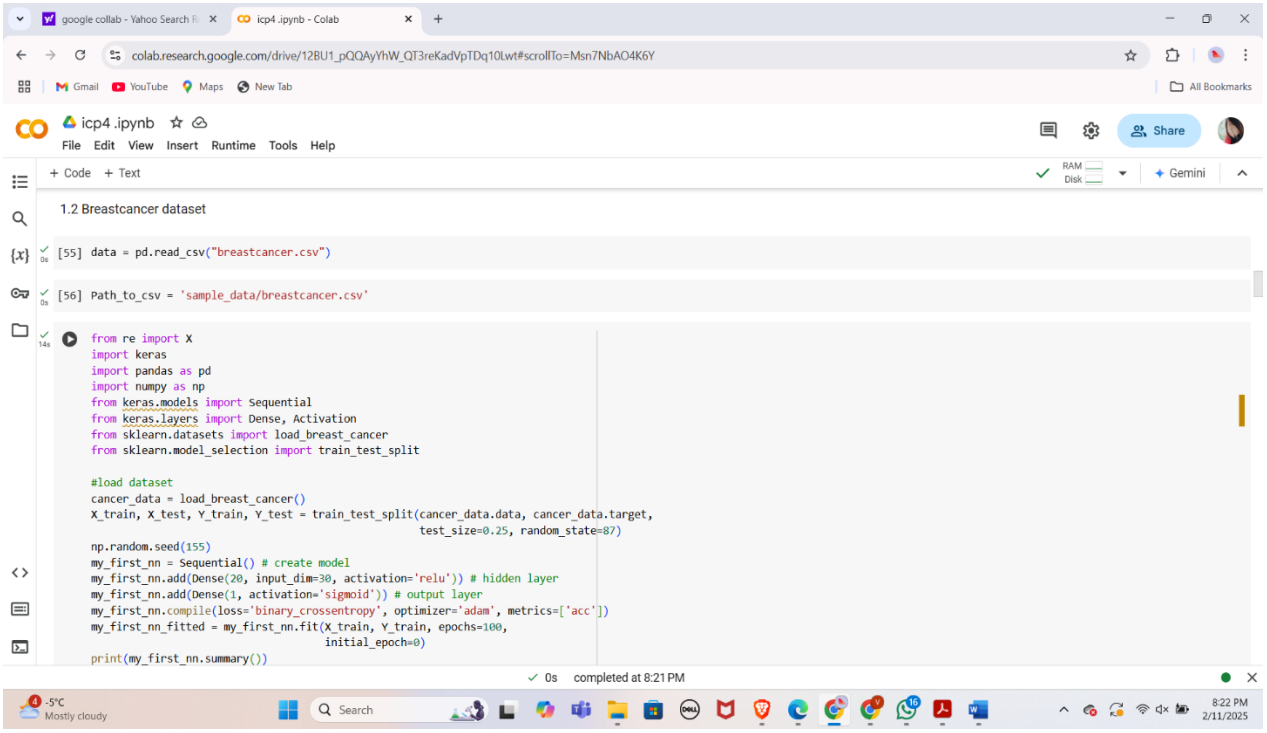
outputs:

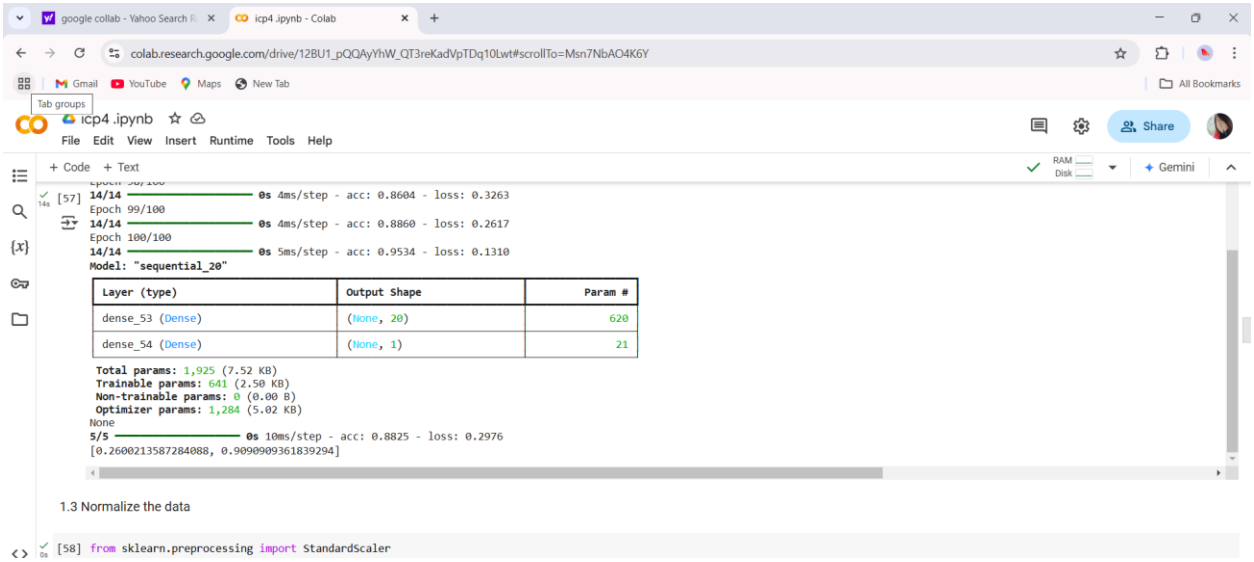
1


google collab - Yahoo Search
icp4 .ipynb - Colab
colab.research.google.com/drive/12BU1_pQQAyYhW_QT3reKadVpTDq10Lwt#scrollTo=Msn7NbAO4K6Y
Gmail
YouTube
Maps
New Tab
All Bookmarks
icp4 .ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
RAM
Disk
Gemini
1.1 Add more dense layers to the existing code
[52] #read the data
import pandas as pd
import tensorflow as tf
data = pd.read_csv('diabetes.csv')
[53] path_to_csv = 'diabetes.csv'
import keras
import pandas as pd
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("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
0s completed at 8:19 PM
-5°C
Mostly cloudy
Search
8:20 PM
2/11/2025

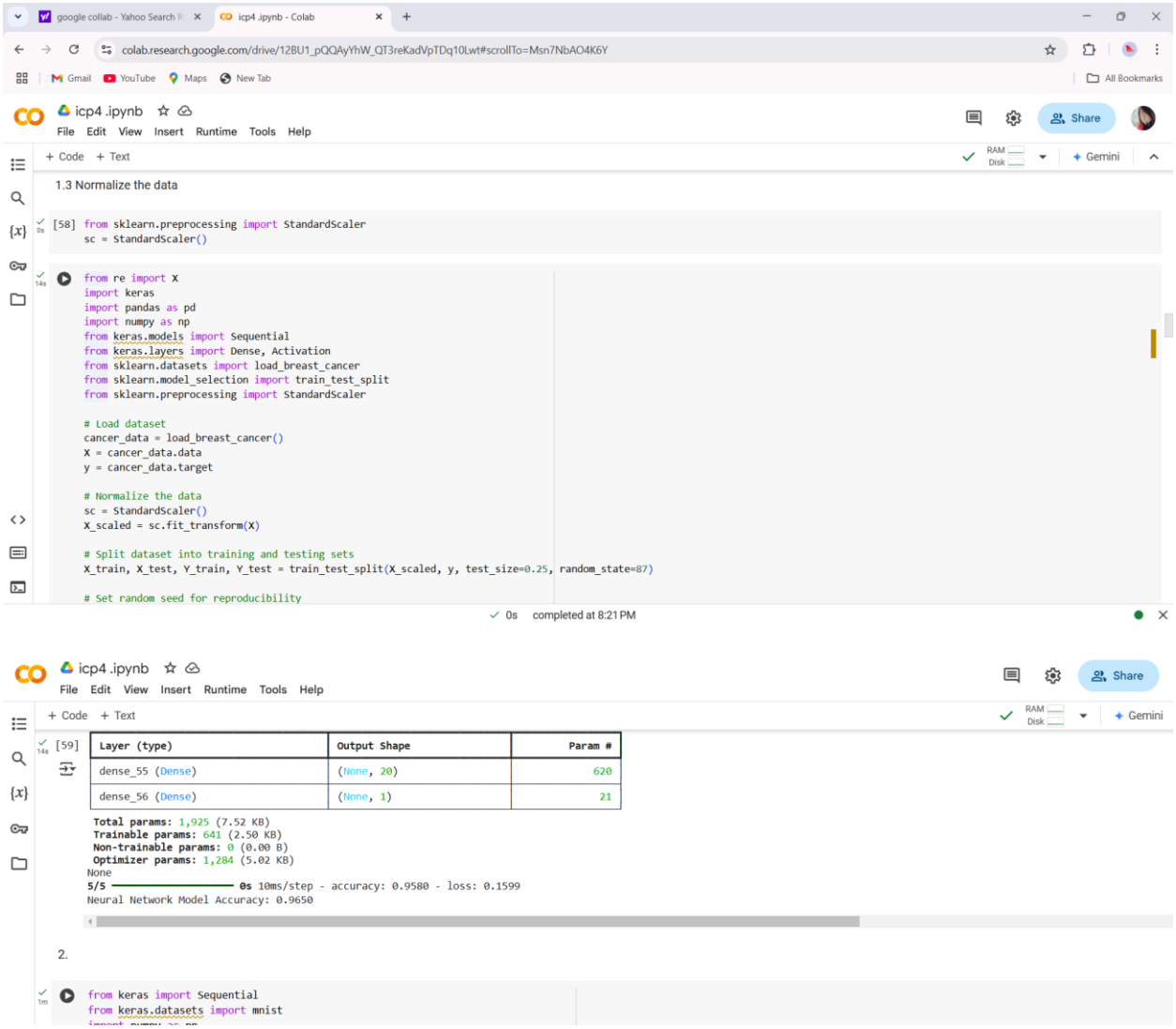

colab.research.google.com/drive/12BU1_pQQAyYhW_QT3reKadVpTDq10Lwt#scrollTo=Msn7NbAO4K6Y
Gmail
YouTube
Maps
New Tab
icp4 .ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
RAM
Disk
18/18 0s 3ms/step - acc: 0.6420 - loss: 0.5862
Epoch 98/100
18/18 0s 4ms/step - acc: 0.6653 - loss: 0.5835
Code cell output actions
0s 3ms/step - acc: 0.6504 - loss: 0.5674
Epoch 100/100
18/18 0s 3ms/step - acc: 0.6682 - loss: 0.5645
Model: "sequential_19"
Layer (type) Output Shape Param #
dense_50 (Dense) (None, 20) 180
dense_51 (Dense) (None, 4) 84
dense_52 (Dense) (None, 1) 5
Total params: 809 (3.16 KB)
Trainable params: 269 (1.05 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 540 (2.11 KB)
None
6/6 0s 6ms/step - acc: 0.6296 - loss: 0.6630
[0.659233570098877, 0.609375]
1.2 Breastcancer dataset
[55] data = pd.read_csv("breastcancer.csv")
0s completed at 8:21 PM

2


google collab - Yahoo Search
icp4 .ipynb - Colab
colab.research.google.com/drive/12BU1_pQQAyYhW_QT3reKadVpTDq10Lwt#scrollTo=Msn7NbAO4K6Y
Gmail
YouTube
Maps
New Tab
All Bookmarks
icp4 .ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
RAM
Disk
Gemini
1.2 Breastcancer dataset
[55] data = pd.read_csv("breastcancer.csv")
[56] Path_to_csv = 'sample_data/breastcancer.csv'
14s
from re import X
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
#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_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'])
my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
initial_epoch=0)
print(my_first_nn.summary())
0s completed at 8:21 PM
-5°C
Mostly cloudy
Search
8:22 PM
2/11/2025


google collab - Yahoo Search
icp4 .ipynb - Colab
colab.research.google.com/drive/12BU1_pQQAyYhW_QT3reKadVpTDq10Lwt#scrollTo=Msn7NbAO4K6Y
Tab groups
icp4 .ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
RAM
Disk
Gemini
[57] 14/14 0s 4ms/step - acc: 0.8604 - loss: 0.3263
Epoch 99/100
14/14 0s 4ms/step - acc: 0.8860 - loss: 0.2617
Epoch 100/100
14/14 0s 5ms/step - acc: 0.9534 - loss: 0.1310
Model: "sequential_20"
Layer (type) Output Shape Param #
dense_53 (Dense) (None, 20) 620
dense_54 (Dense) (None, 1) 21
Total params: 1,925 (7.52 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 1,284 (5.02 KB)
None
5/5 0s 10ms/step - acc: 0.8825 - loss: 0.2976
[0.2600213587284088, 0.9090909361839294]
1.3 Normalize the data
[58] from sklearn.preprocessing import StandardScaler

3



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.

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.

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.

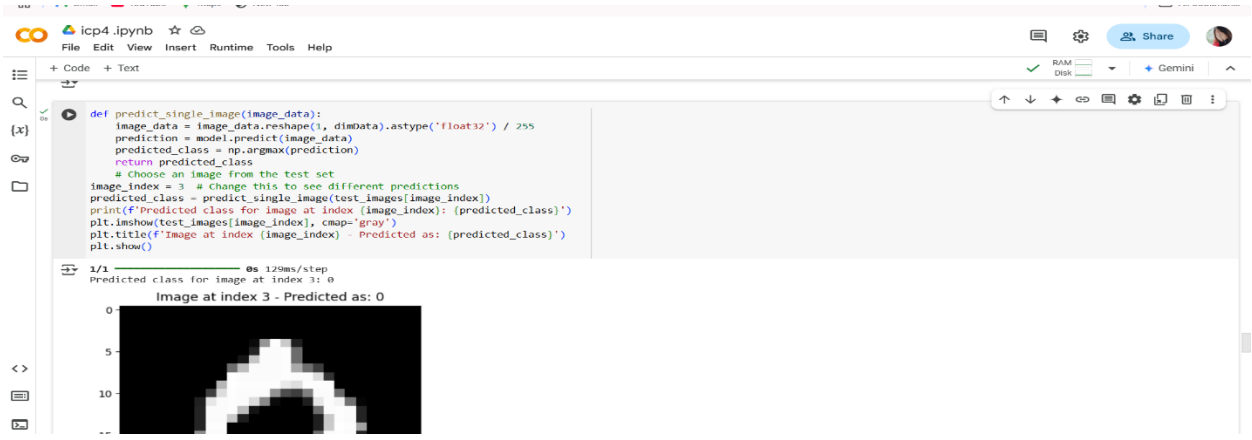
4. Run the same code without scaling the images and check the performance?

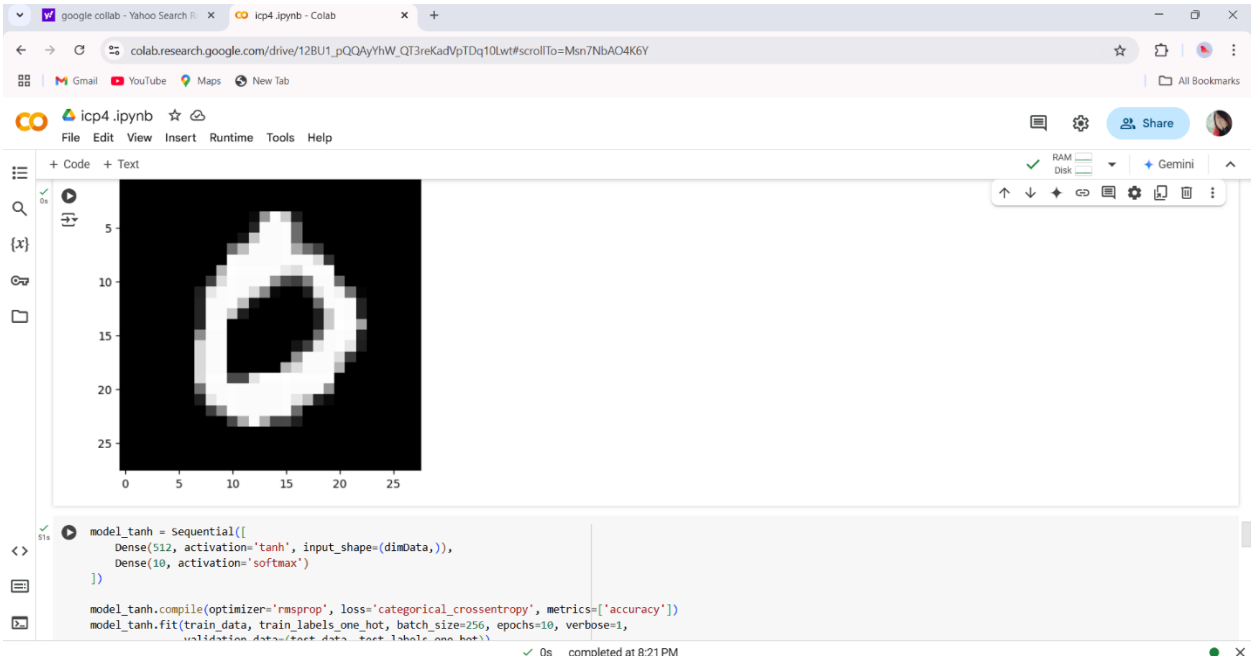
2)

1


+ Code
+ Text
2.
[60] 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')
test_data = test_data.astype('float')
#scale data
train_data /=255.0
test_data /=255.0
Accuracy
Training Accuracy
Validation Accuracy
Loss
Training Loss
Validation Loss

2


icp4.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
Gemini
def predict_single_image(image_data):
image_data = image_data.reshape(1, dimData).astype('float32') / 255
prediction = model.predict(image_data)
predicted_class = np.argmax(prediction)
return predicted_class
# Choose an image from the test set
image_index = 3 # Change this to see different predictions
predicted_class = predict_single_image(test_images[image_index])
print(f'Predicted class for image at index {image_index}: {predicted_class}')
plt.imshow(test_images[image_index], cmap='gray')
plt.title(f'Image at index {image_index} - Predicted as: {predicted_class}')
plt.show()
1/1 0s 129ms/step
Predicted class for image at index 3: 0
Image at index 3 - Predicted as: 0


google collab - Yahoo Search R
icp4 .ipynb - Colab
colab.research.google.com/drive/12BU1_pQQAyYhW_QT3reKadVpTDq10Lwt#scrollTo=Msn7NbAO4K6Y
Gmail
YouTube
Maps
New Tab
All Bookmarks
icp4 .ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
RAM
Disk
Gemini
model_tanh = Sequential([
Dense(512, activation='tanh', input_shape=(dimData,)),
Dense(10, activation='softmax')
])
model_tanh.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model_tanh.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
0s completed at 8:21 PM

3.


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))
(28, 28)
784
Epoch 1/10
235/235 8s 27ms/step - accuracy: 0.8325 - loss: 0.5323 - val_accuracy: 0.9500 - val_loss: 0.1535
Epoch 2/10
235/235 11s 31ms/step - accuracy: 0.9665 - loss: 0.1082 - val_accuracy: 0.9605 - val_loss: 0.1199
Epoch 3/10
235/235 11s 34ms/step - accuracy: 0.9801 - loss: 0.0634 - val_accuracy: 0.9784 - val_loss: 0.0715
Epoch 4/10
235/235 8s 32ms/step - accuracy: 0.9861 - loss: 0.0438 - val_accuracy: 0.9754 - val_loss: 0.0786
Epoch 5/10
235/235 12s 51ms/step - accuracy: 0.9907 - loss: 0.0306 - val_accuracy: 0.9788 - val_loss: 0.0683
Epoch 6/10
235/235 17s 38ms/step - accuracy: 0.9938 - loss: 0.0213 - val_accuracy: 0.9811 - val_loss: 0.0683
Epoch 7/10
235/235 8s 28ms/step - accuracy: 0.9953 - loss: 0.0160 - val_accuracy: 0.9833 - val_loss: 0.0550
Epoch 8/10
235/235 8s 33ms/step - accuracy: 0.9965 - loss: 0.0109 - val_accuracy: 0.9808 - val_loss: 0.0720
Epoch 9/10
235/235 9s 27ms/step - accuracy: 0.9972 - loss: 0.0087 - val_accuracy: 0.9814 - val_loss: 0.0721
Epoch 10/10
235/235 10s 28ms/step - accuracy: 0.9980 - loss: 0.0071 - val_accuracy: 0.9832 - val_loss: 0.0631