Assignment – 4

Name : Ganesh konagalla

Student Id : 700756412

Github Link :

https://github.com/ganeshkonagalla123/Neural-Networks/tree/main/Assignment4

1st Program Code :

```
# Imports
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target
```

```
# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Split the dataset into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=87)
```

```
# Neural network model with additional dense layers
model = Sequential()
model.add(Dense(20, input_dim=X_train.shape[1], activation='relu'))  # Adjusted input_dim for Breast Cancer dataset
model.add(Dense(64, activation='relu'))  # Additional dense layer
model.add(Dense(64, activation='relu'))  # Additional dense layer
model.add(Dense(1, activation='sigmoid'))  # Output layer for binary classification
```

```
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

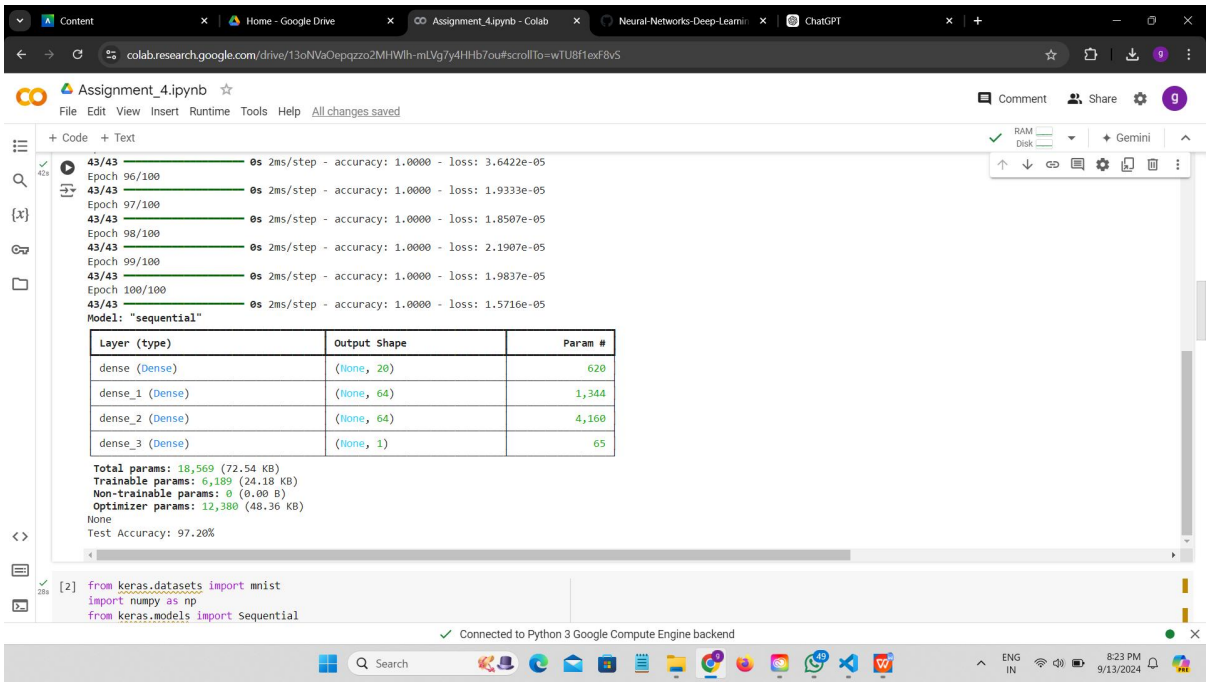
```
# Train the model
model.fit(X_train, Y_train, epochs=100, batch_size=10, verbose=1)
```

```
# Model summary
print(model.summary())
```

```
# Evaluate the model on the test data
loss, accuracy = model.evaluate(X_test, Y_test, verbose=0)
```

```
print(f'Test Accuracy: {accuracy*100:.2f}%')
```

Output :



2nd Program Code :

```
from keras.datasets import mnist
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
# Preprocess the data
dimData = np.prod(train_images.shape[1:])
train_data = train_images.reshape(train_images.shape[0], dimData).astype('float32') / 255
test_data = test_images.reshape(test_images.shape[0], dimData).astype('float32') / 255
train_labels_one_hot = to_categorical(train_labels)
test_labels_one_hot = to_categorical(test_labels)
```

```
# Creating the network
model = Sequential([
    Dense(512, activation='relu', input_shape=(dimData,)),
    Dense(512, activation='relu'),
    Dense(10, activation='softmax')
])
```

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

```
# Train the model
history = model.fit(train_data, train_labels_one_hot, batch_size=256, epochs=10, verbose=1,
                    validation_data=(test_data, test_labels_one_hot))
```

```
# Plotting the accuracy and loss
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()
```

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.legend()
plt.show()
```

```
# Predicting a single image
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
```

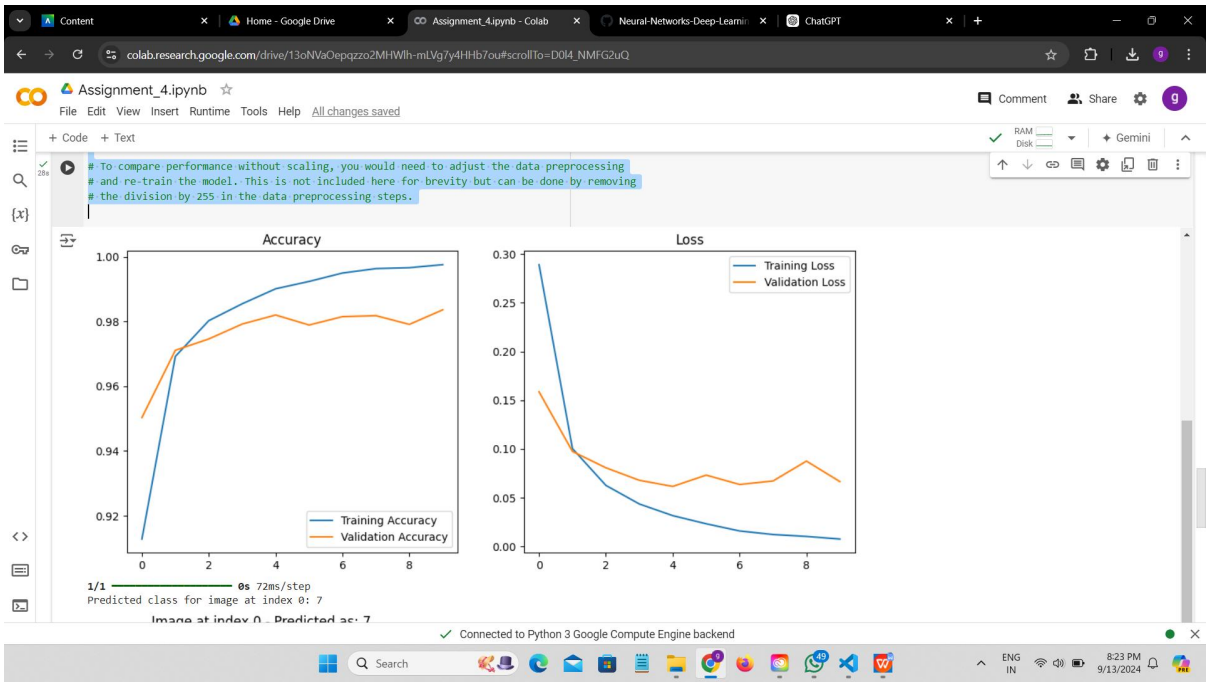
```
# Choose an image from the test set
image_index = 0  # Change this to see different predictions
predicted_class = predict_single_image(test_images[image_index])
print(f'Predicted class for image at index {image_index}: {predicted_class}')
plt.imshow(test_images[image_index], cmap='gray')
plt.title(f'Image at index {image_index} - Predicted as: {predicted_class}')
plt.show()
```

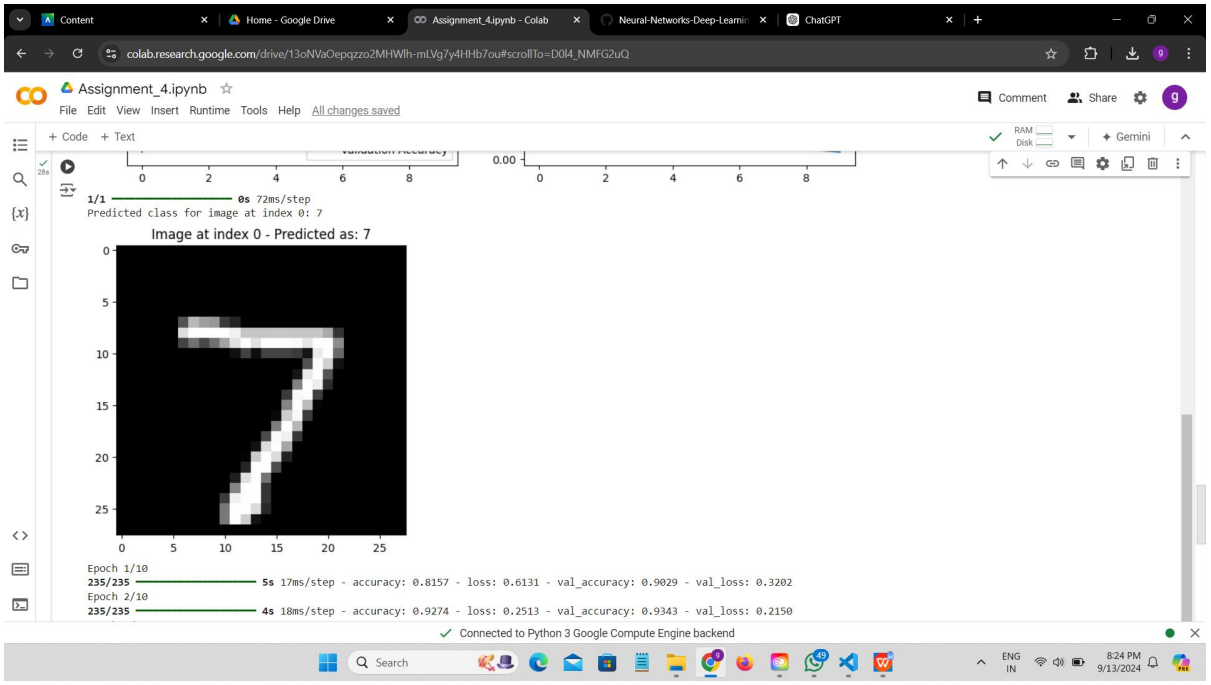
```
# Modify the model to use different activation functions and fewer layers for comparison
# You can adjust the following model architecture as needed
model_tanh = Sequential([
    Dense(512, activation='tanh', input_shape=(dimData,)),
    Dense(10, activation='softmax')
])
```

```
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,
               validation_data=(test_data, test_labels_one_hot))
```

```
# To compare performance without scaling, you would need to adjust the data preprocessing
# and re-train the model. This is not included here for brevity but can be done by removing
# the division by 255 in the data preprocessing steps.
```

Output :




Content
Home - Google Drive
Assignment_4.ipynb - Colab
Neural-Networks-Deep-Learnin
ChatGPT
colab.research.google.com/drive/13oNVaOepqzzo2MHWlh-mLVg7y4HHb7ou#scrollTo=D0l4_NMFG2uQ
Assignment_4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
1/1 0s 72ms/step
Predicted class for image at index 0: 7
Image at index 0 - Predicted as: 7
Epoch 1/10
235/235 5s 17ms/step - accuracy: 0.8157 - loss: 0.6131 - val_accuracy: 0.9029 - val_loss: 0.3202
Epoch 2/10
235/235 4s 18ms/step - accuracy: 0.9274 - loss: 0.2513 - val_accuracy: 0.9343 - val_loss: 0.2150
Connected to Python 3 Google Compute Engine backend
Search
ENG
IN
8:24 PM
9/13/2024