EXP NO: 4 :- Build a Simple feed forward neural network to recognize handwritten characte

Aim:- To build and train a feed forward neural network model to recognize handwritten digits (0-9) using the MNIST dataset.

objective:-

1. To understand the architecture of a feed forward neural network.
2. To apply supervised learning for image classification
3. To preprocess the MNIST dataset for neural network training
4. To evaluate the model's accuracy and analyze performance

Algorithm:-

1. Import libraries:- Load required python libraries like tensorflow or keras, numpy and matplotlib
2. Load Dataset:- Load the MNIST Dataset.
3. Preprocess Data:-
   - → Normalize pixel values between 0 & 1.
   - → Flatten 28x28 images into 784-dimensional
   - → Convert labels into one hot encoded format
4. Define Model:-
   - → Input layer:- 784 neurons (flatten image)
   - → Hidden layer:- 128 neurons with ReLU activation

→ output layer:- 10 neurons with softmax activation

5. Compile Model:

choose loss function, optimizer → adam and metrics (accuracy). ↳ categorical-crossentropy

6. Train Model:- Feed training data into the model for a set number of epochs.

7. Evaluate Model:- Test the model on unseen test data and record accuracy.

8. Predict & visualize:- Make predictions on sample images to verify model performance.

Pseudo Code:-

START

Import required libraries

Load MNIST dataset

Normalize pixel values between 0 & 1

Flatten 28x28 images into vectors.

one-hot encode the labels.

Define a sequential neural network:

Input layer : size 784

Hidden layer : 128 neurons, softmax activation

Compile model with adam optimizer, categorical crossentropy loss, accuracy metric

Train model using training dataset for defined epochs.

Evaluate model on test dataset

Display accuracy and sample predictions

END.

Observations:-

- The model achieved ~97-98% accuracy on the MNIST test dataset with just 5 epochs of training.
- Accuracy improved steadily with each epoch, indicating effective learning
- Predictions on unseen data matched the actual digits in most cases.
- Errors occured mainly on digits that were poorly written or ambiguous

Conclusion:-

A simple feed forward neural network with one hidden layer can accurately classify handwritten digits from the MNIST dataset.

The performance (~98% accuracy) shows that FFNN's are effective for basic image recognization tasks after Normalization and one-hot encoding.

However, for more complex image recognization problems, deeper architectures like Convolutional Neural Network (CNNs) may be more suitable.

Output:

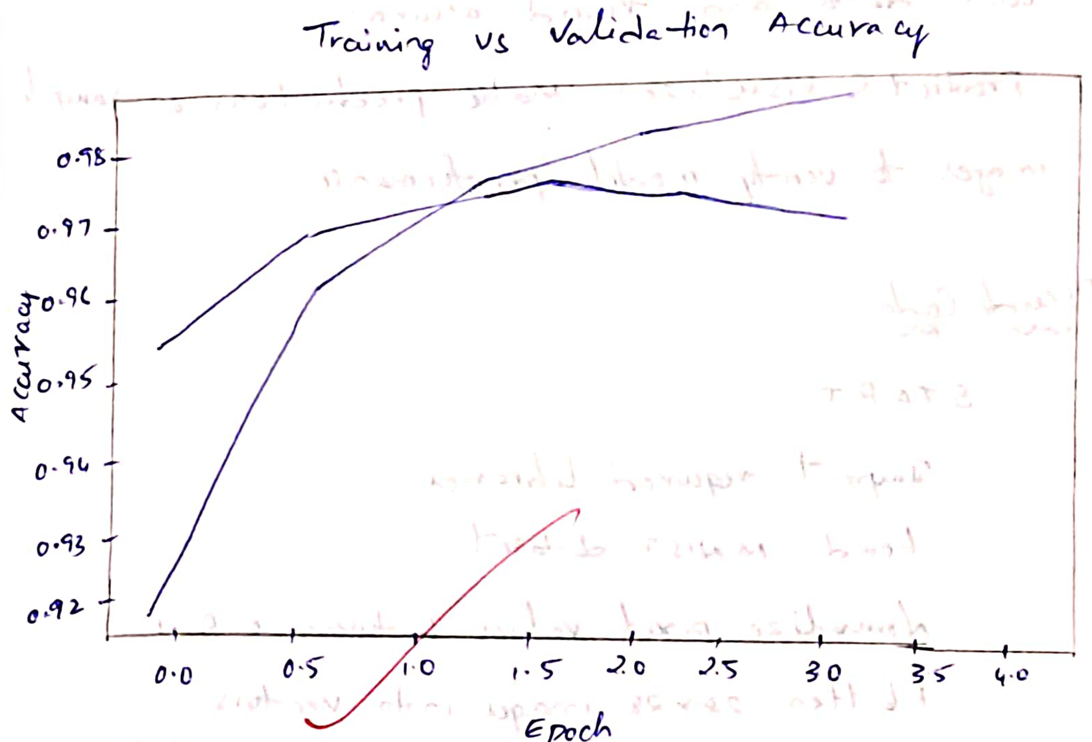
Epoch [1/5], Train Acc: 0.9176, val acc: 0.9548

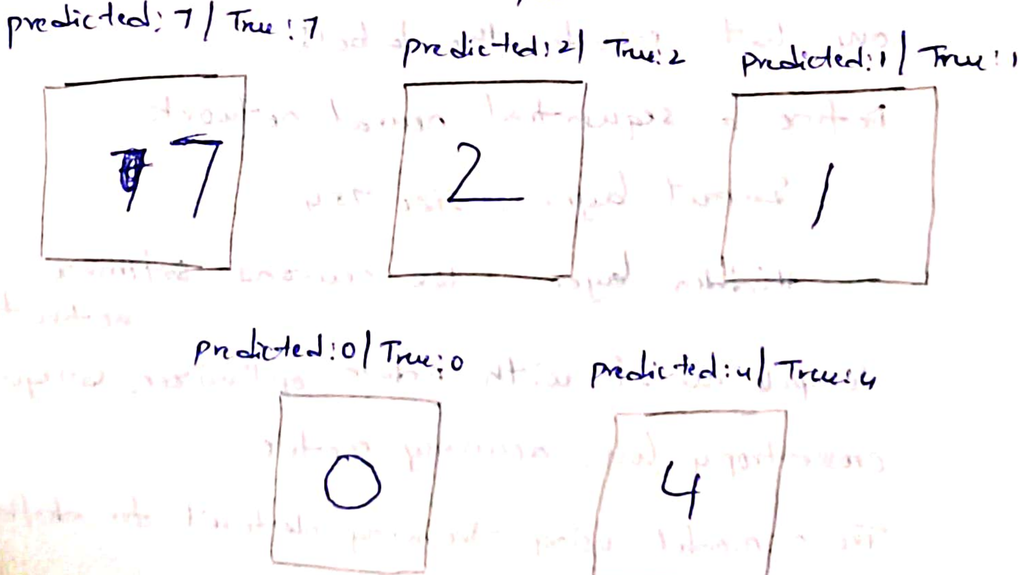
Epoch [2/5], Train Acc: 0.9624, val Acc: 0.9661

Epoch [3/5], Train Acc: 0.9738, val Acc: 0.9719

Epoch [4/5], Train Acc: 0.9802, val Acc: 0.9743

Epoch [5/5], Train Acc: 0.9833, val Acc: 0.9732





https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#scrollTo=tDrWee74Sfe

DLT LAB-4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini

RAM Disk

```
# Feed Forward Neural Network for MNIST Classification (PyTorch)

import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# 1. Device configuration (GPU if available)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# 2. Load and preprocess MNIST dataset
transform = transforms.ToTensor()  # Converts images to tensor and normalizes to [0,1]

train_dataset = datasets.MNIST(root='./data', train=True, transform=transform, download=True)
test_dataset = datasets.MNIST(root='./data', train=False, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

# 3. Define feedforward neural network
class FFNN(nn.Module):
    def __init__(self):
        super(FFNN, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(28*28, 128)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.flatten(x)
```

https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#scrollTo=tTDrWee74Sfe

DLT LAB-4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini

RAM Disk

```python
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = FFNN().to(device)

# 4. Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters())

# 5. Train the model
num_epochs = 5
train_acc_history = []
val_acc_history = []

for epoch in range(num_epochs):
    model.train()
    correct, total = 0, 0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        outputs = model(images)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    train_accuracy = correct / total
    train_acc_history.append(train_accuracy)
```

https://colab.research.google.com/drive/1lIRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#scrollTo=tTDrWee74Sfe

DLT LAB-4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini RAM Disk

```python
    # Validation
    model.eval()
    correct, total = 0, 0
    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    val_accuracy = correct / total
    val_acc_history.append(val_accuracy)

    print(f"Epoch [{epoch+1}/{num_epochs}], Train Acc: {train_accuracy:.4f}, Val Acc: {val_accuracy:.4f}")

# 6. Plot training vs validation accuracy
plt.plot(train_acc_history, label='Train Accuracy')
plt.plot(val_acc_history, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')
plt.show()

# 7. Predict and visualize some test images
import numpy as np

model.eval()
examples = enumerate(test_loader)
batch_idx, (example_data, example_targets) = next(examples)
example_data, example_targets = example_data.to(device), example_targets.to(device)

with torch.no_grad():
    output = model(example_data)
```

https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#scrollTo=tTDrWee74Sfe

DLT LAB-4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini

```
# Display first 5 predictions
for i in range(5):
    plt.imshow(example_data[i].cpu().squeeze(), cmap='gray')
    pred_label = output[i].argmax(dim=0).item()
    true_label = example_targets[i].item()
    plt.title(f"Predicted: {pred_label} | True: {true_label}")
    plt.axis('off')
    plt.show()
```

```
100%|██████████| 9.91M/9.91M [00:00<00:00, 59.7MB/s]
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.70MB/s]
100%|██████████| 1.65M/1.65M [00:00<00:00, 14.5MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 8.03MB/s]
Epoch [1/5], Train Acc: 0.9172, Val Acc: 0.9520
Epoch [2/5], Train Acc: 0.9631, Val Acc: 0.9702
Epoch [3/5], Train Acc: 0.9736, Val Acc: 0.9722
Epoch [4/5], Train Acc: 0.9799, Val Acc: 0.9755
Epoch [5/5], Train Acc: 0.9838, Val Acc: 0.9758
```

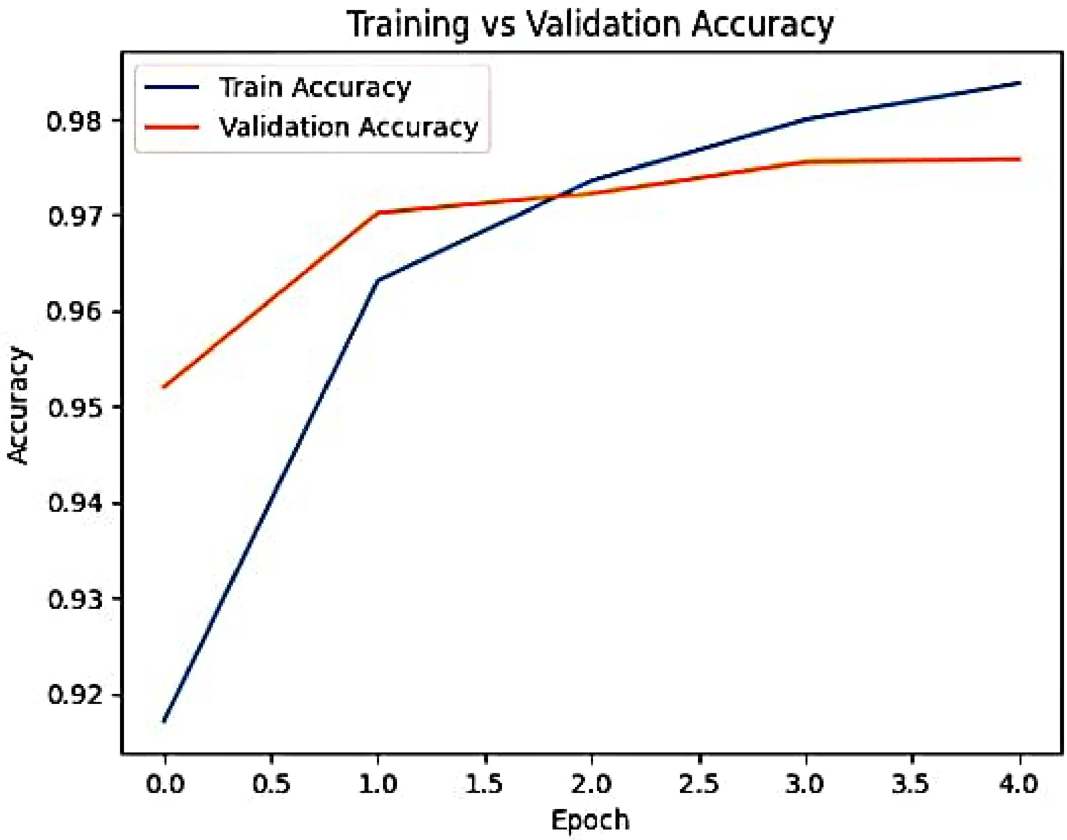
https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?auth

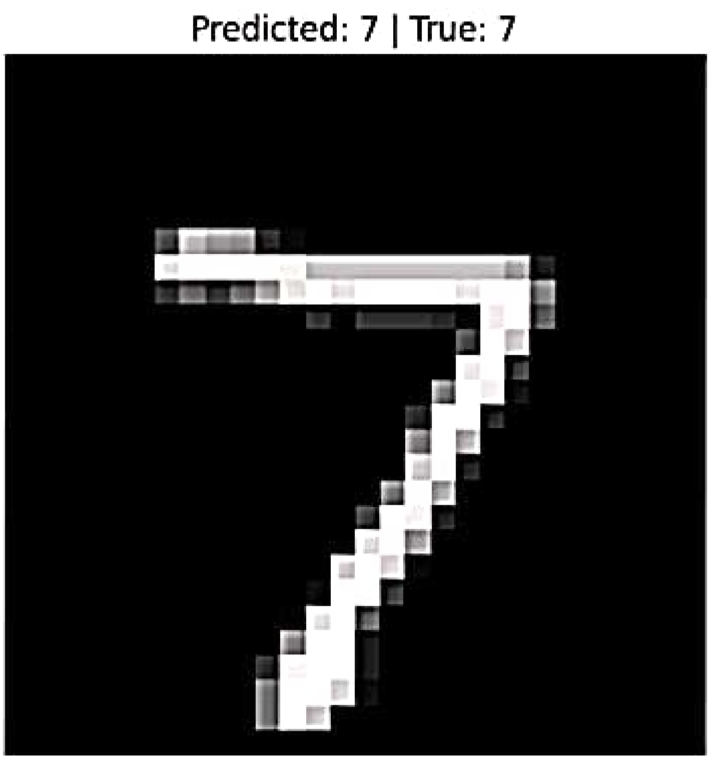
DLT LAB-4.ipynb

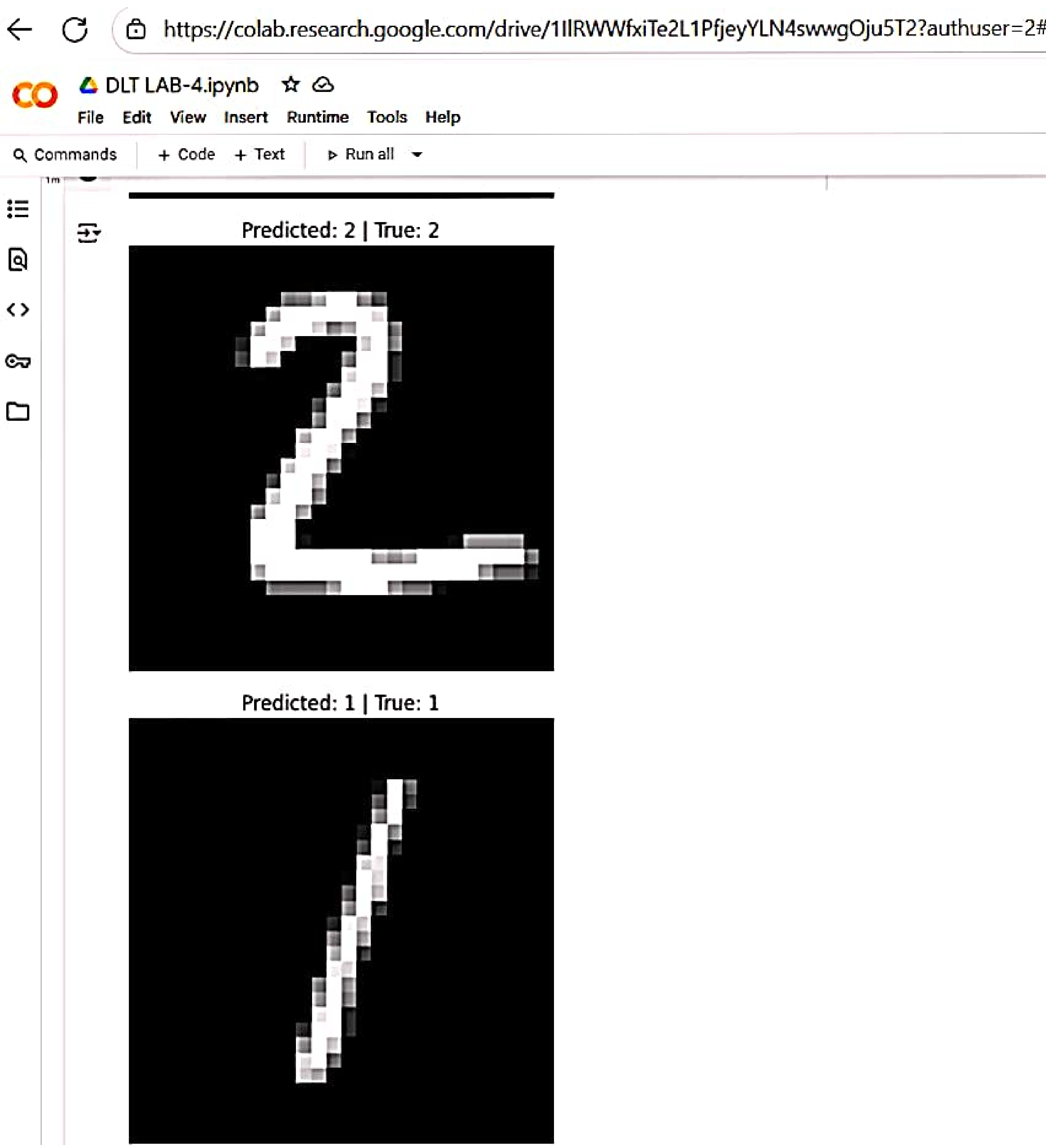
File Edit View Insert Runtime Tools Help

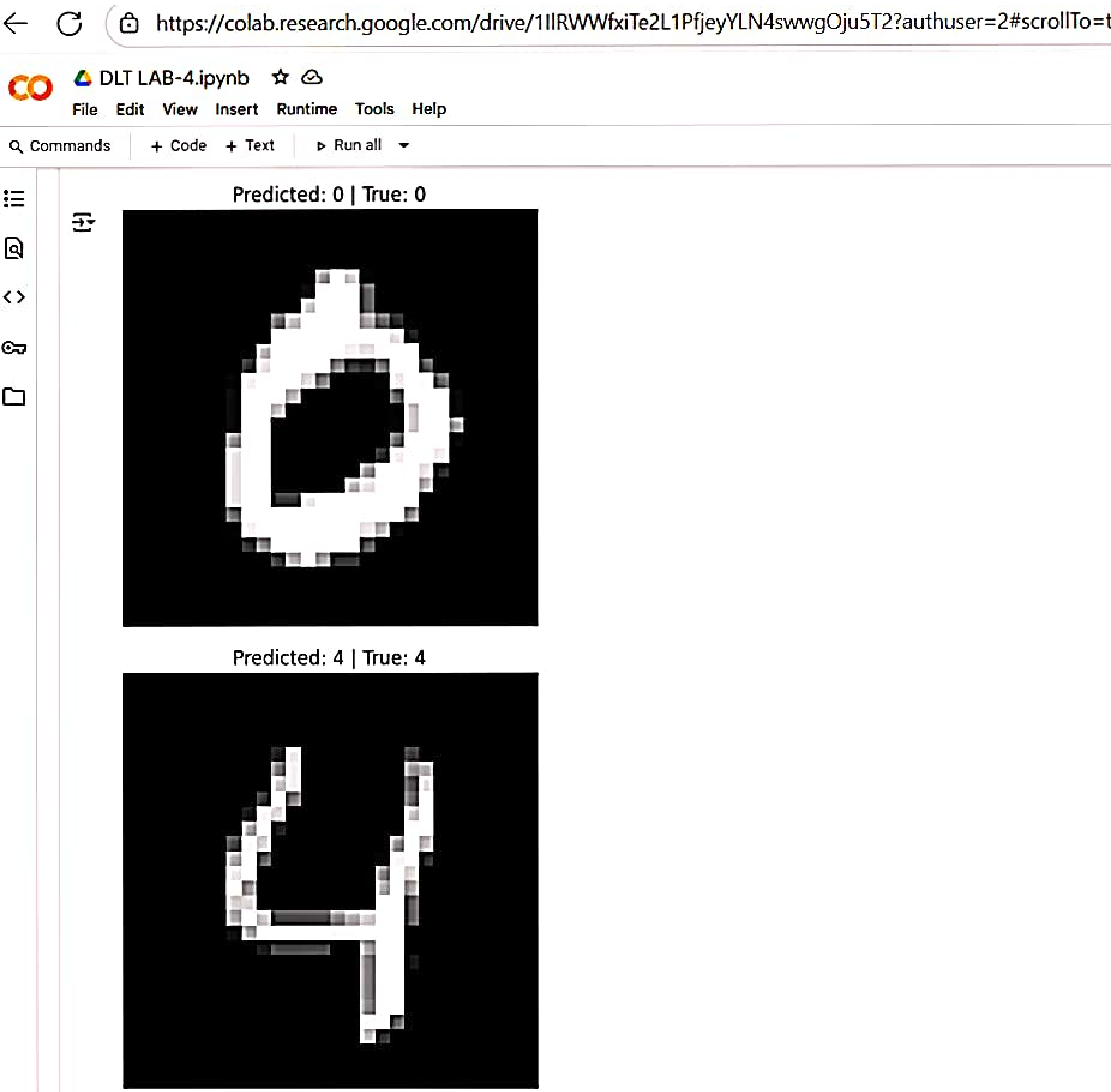
Commands + Code + Text Run all

```
Epoch [5/5], Train Acc: 0.9838, Val Acc: 0.9758
```






https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#
DLT LAB-4.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code + Text
Run all
Predicted: 2 | True: 2
Predicted: 1 | True: 1


https://colab.research.google.com/drive/1IlRWWfxiTe2L1PfjeyYLN4swwgOju5T2?authuser=2#scrollTo=t
DLT LAB-4.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Predicted: 0 | True: 0
Predicted: 4 | True: 4