14/08/26 Exp4 : Build a Simple Feed Forward Neural Network to Recognize Handwritten Character

Aim:

To design and implement a simple feed forward neural network for recognizing handwritten character using open Source dataset.

Objective:

* To Understand FFNN
* To train an FFNN on handwritten character images.
* To evaluate the networks performance using statistical metrics.
* To visualize predictions for sample Test images.

Pseudocode:

```
1) Import necessary libraries.
2) Load the handwritten character dataset
      (MNIST Digits Dataset)
3) Normalize the image pixel values.
4) Split the dataset into training and testing
   sets
5) Define the feed forward neural network
   architecture:
      - Input layer (flattened image pixels)
      - one or more hidden dense layer
      - Output layer (number of character classes)
```

6) Compile The model with loss function, optimizer and metrics.

7) Train The model on training data

8) Evaluate the model on Test data

9) Visualize sample predictions

Observation

| Metric | Value |
|---|---|
| Accuracy | ~~92.88%~~<br>97.82% |

↓

This shows that

★ model successfully learned patterns of handwritten characters from the dataset

★ Normalization of pixels values helped improve training performance

★ Increasing No. of layers, neurons or training epochs may improve accuracy further.

Output

```
Epoch [1/5] , Loss : 1.0594
Epoch [2/5], Loss : 0.3860
Epoch [3/5] Loss :0.3290
Epoch [4/5], Loss : 0.2974
Epoch [5/5], Loss : 0.2726
```

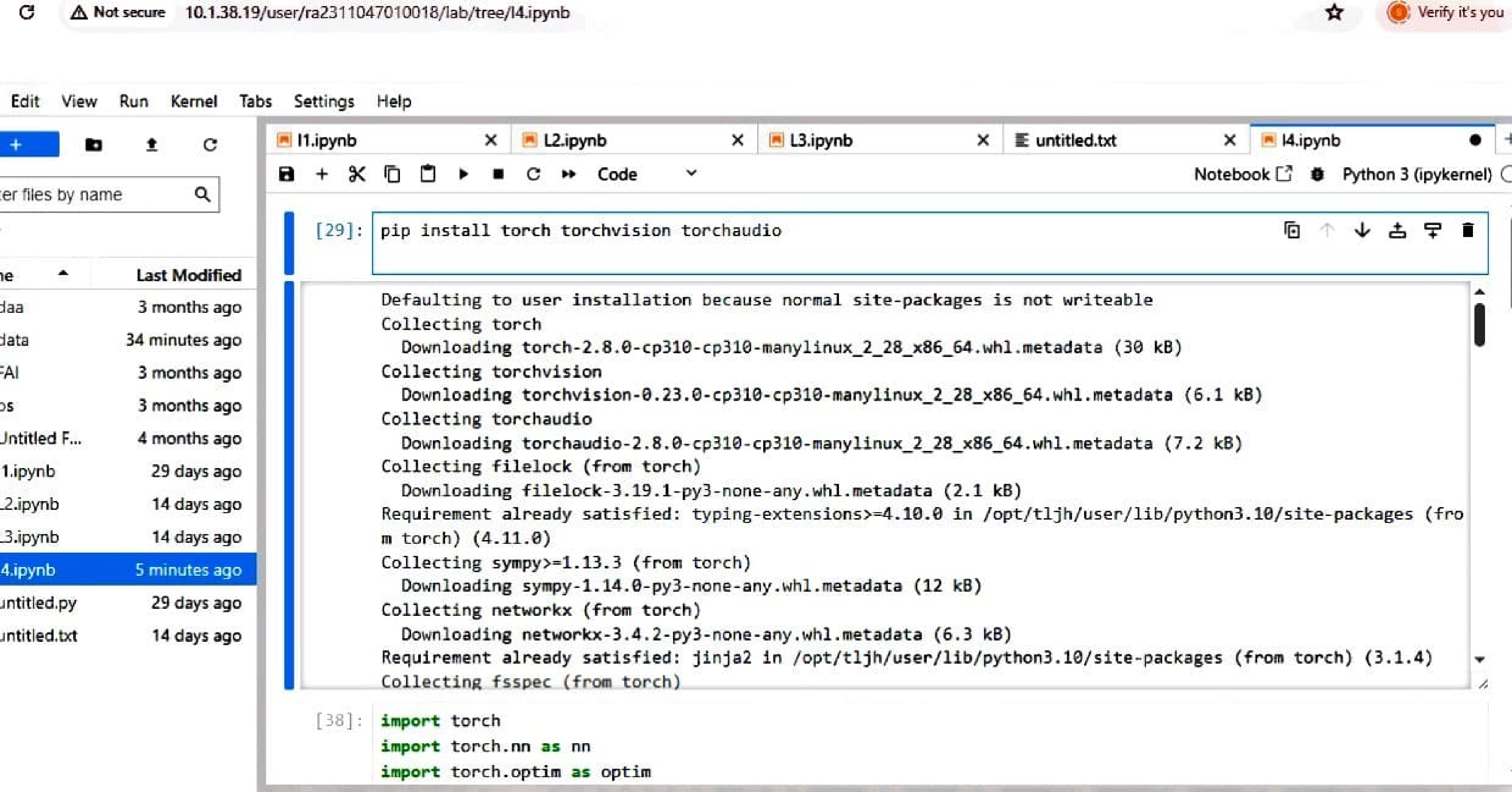
Table

| Parameter | Observation |
|---|---|
| Dataset Used | MNIST Handwritten Digits Dataset |
| Model Type | Feed Forward Neural Network (Fully connected) |
| Training Accuracy | 98. [illegible]. 82% |
| Activation Function | ReLU (Hidden Layer) |

Result

Implemented simple FFNN for recognizing hand written digits using open Source. dataset.

12/8/25



| ne | Last Modified |
|---|---|
| daa | 3 months ago |
| data | 34 minutes ago |
| FAI | 3 months ago |
| os | 3 months ago |
| Untitled F... | 4 months ago |
| 1.ipynb | 29 days ago |
| 2.ipynb | 14 days ago |
| 3.ipynb | 14 days ago |
| 4.ipynb | 5 minutes ago |
| untitled.py | 29 days ago |
| untitled.txt | 14 days ago |

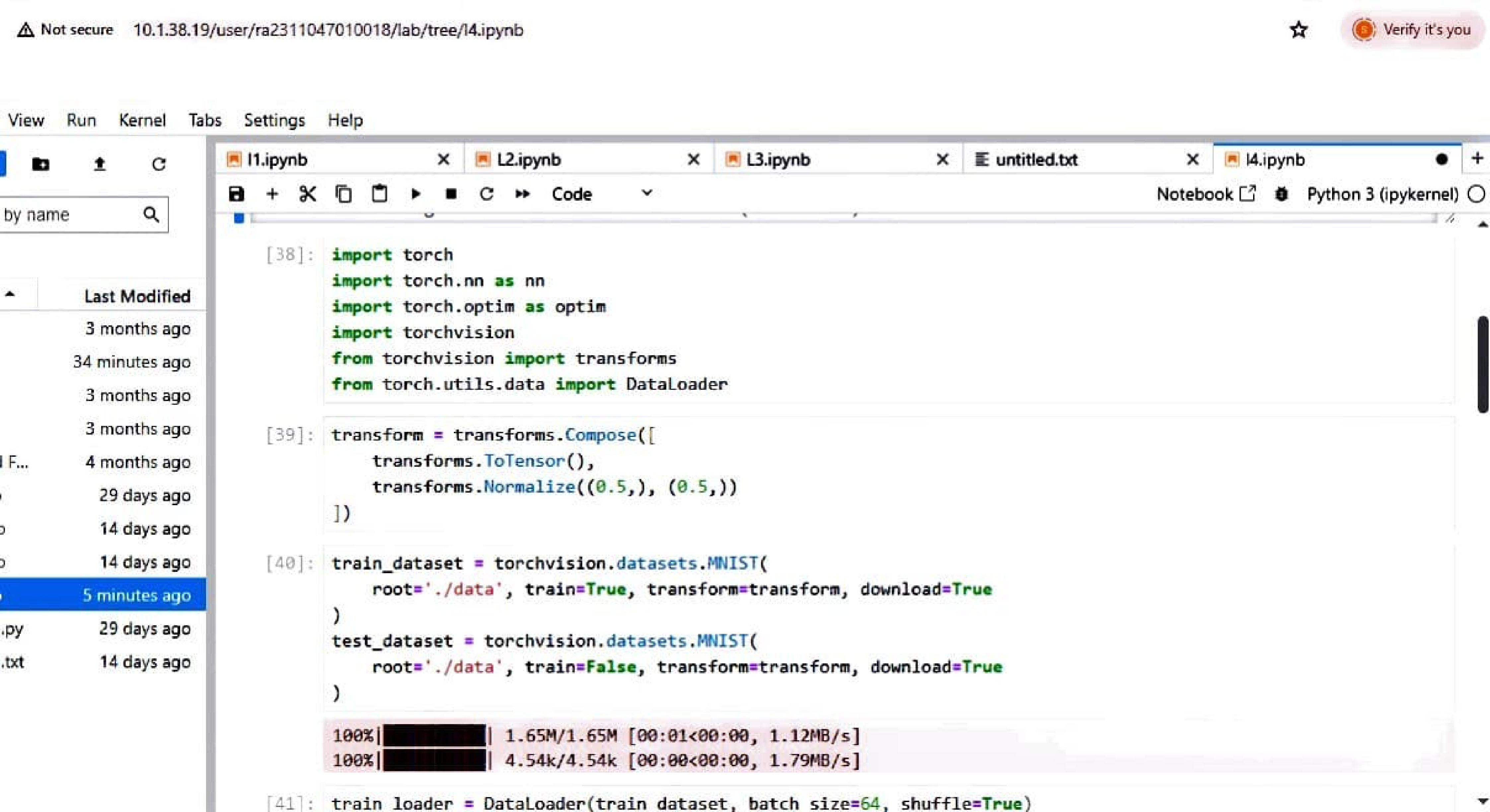
[29]:

```
pip install torch torchvision torchaudio
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting torch
  Downloading torch-2.8.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (30 kB)
Collecting torchvision
  Downloading torchvision-0.23.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (6.1 kB)
Collecting torchaudio
  Downloading torchaudio-2.8.0-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (7.2 kB)
Collecting filelock (from torch)
  Downloading filelock-3.19.1-py3-none-any.whl.metadata (2.1 kB)
Requirement already satisfied: typing-extensions>=4.10.0 in /opt/tljh/user/lib/python3.10/site-packages (fro
m torch) (4.11.0)
Collecting sympy>=1.13.3 (from torch)
  Downloading sympy-1.14.0-py3-none-any.whl.metadata (12 kB)
Collecting networkx (from torch)
  Downloading networkx-3.4.2-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: jinja2 in /opt/tljh/user/lib/python3.10/site-packages (from torch) (3.1.4)
Collecting fsspec (from torch)
```

[38]:

```
import torch
import torch.nn as nn
import torch.optim as optim
```



[38]:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader
```

[39]:

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

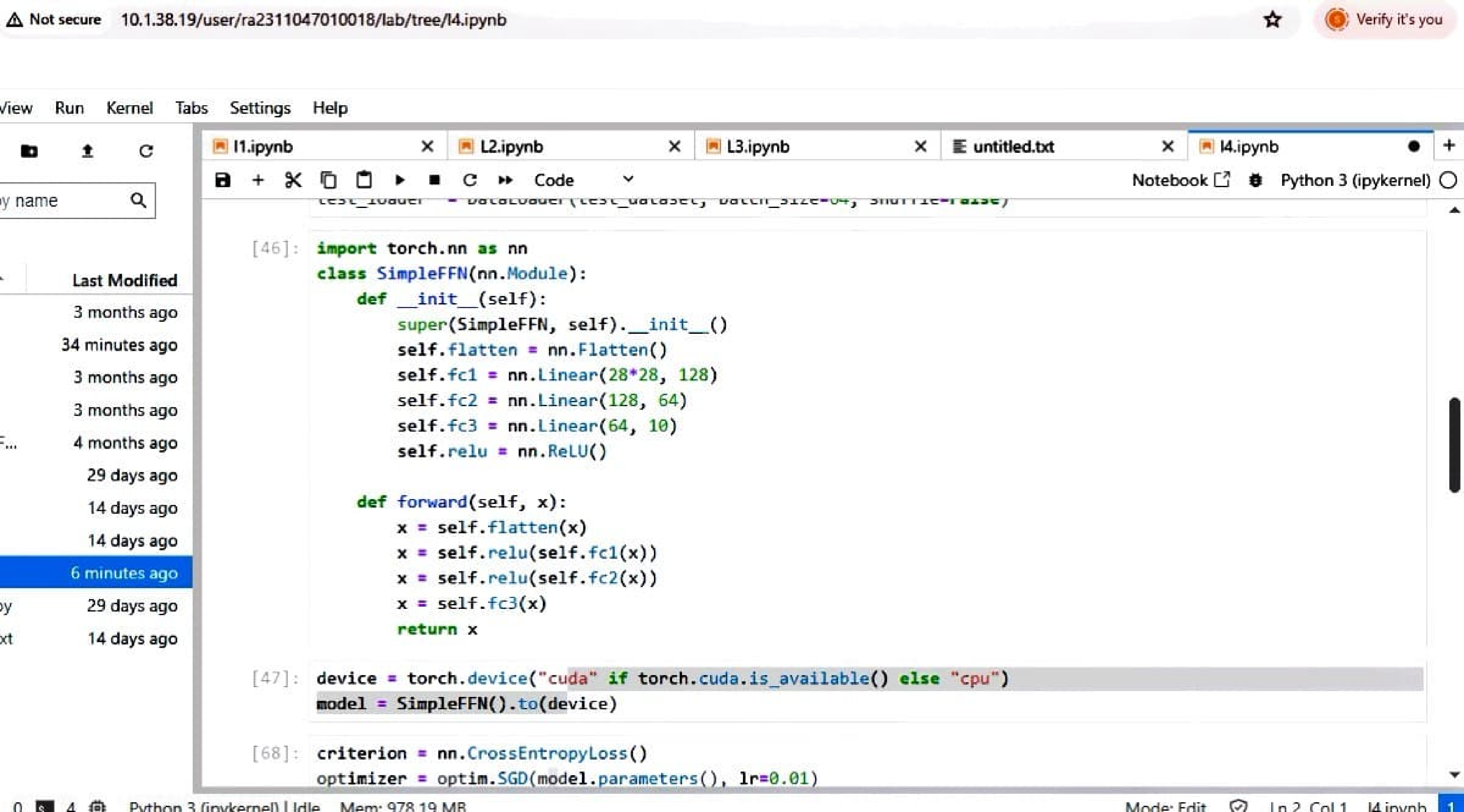
[40]:

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

```
100%|██████████| 1.65M/1.65M [00:01<00:00, 1.12MB/s]
100%|██████████| 4.54k/4.54k [00:00<00:00, 1.79MB/s]
```

[41]:

```
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
```



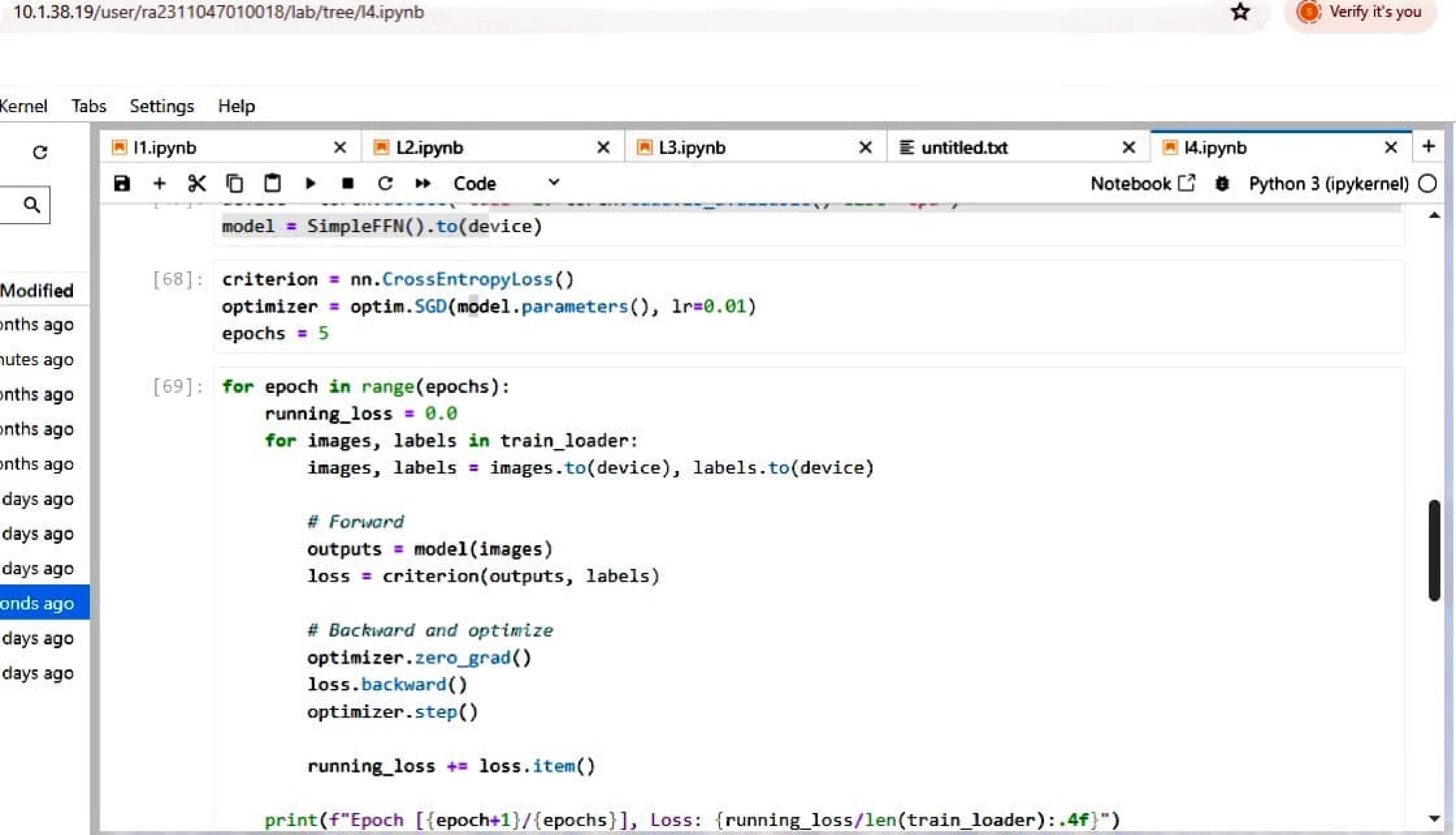
```
test_loader  = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

```
[46]: import torch.nn as nn
      class SimpleFFN(nn.Module):
          def __init__(self):
              super(SimpleFFN, self).__init__()
              self.flatten = nn.Flatten()
              self.fc1 = nn.Linear(28*28, 128)
              self.fc2 = nn.Linear(128, 64)
              self.fc3 = nn.Linear(64, 10)
              self.relu = nn.ReLU()

          def forward(self, x):
              x = self.flatten(x)
              x = self.relu(self.fc1(x))
              x = self.relu(self.fc2(x))
              x = self.fc3(x)
              return x
```

```
[47]: device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
      model = SimpleFFN().to(device)
```

```
[68]: criterion = nn.CrossEntropyLoss()
      optimizer = optim.SGD(model.parameters(), lr=0.01)
```



```
model = SimpleFFN().to(device)
```

[68]:

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
epochs = 5
```

[69]:

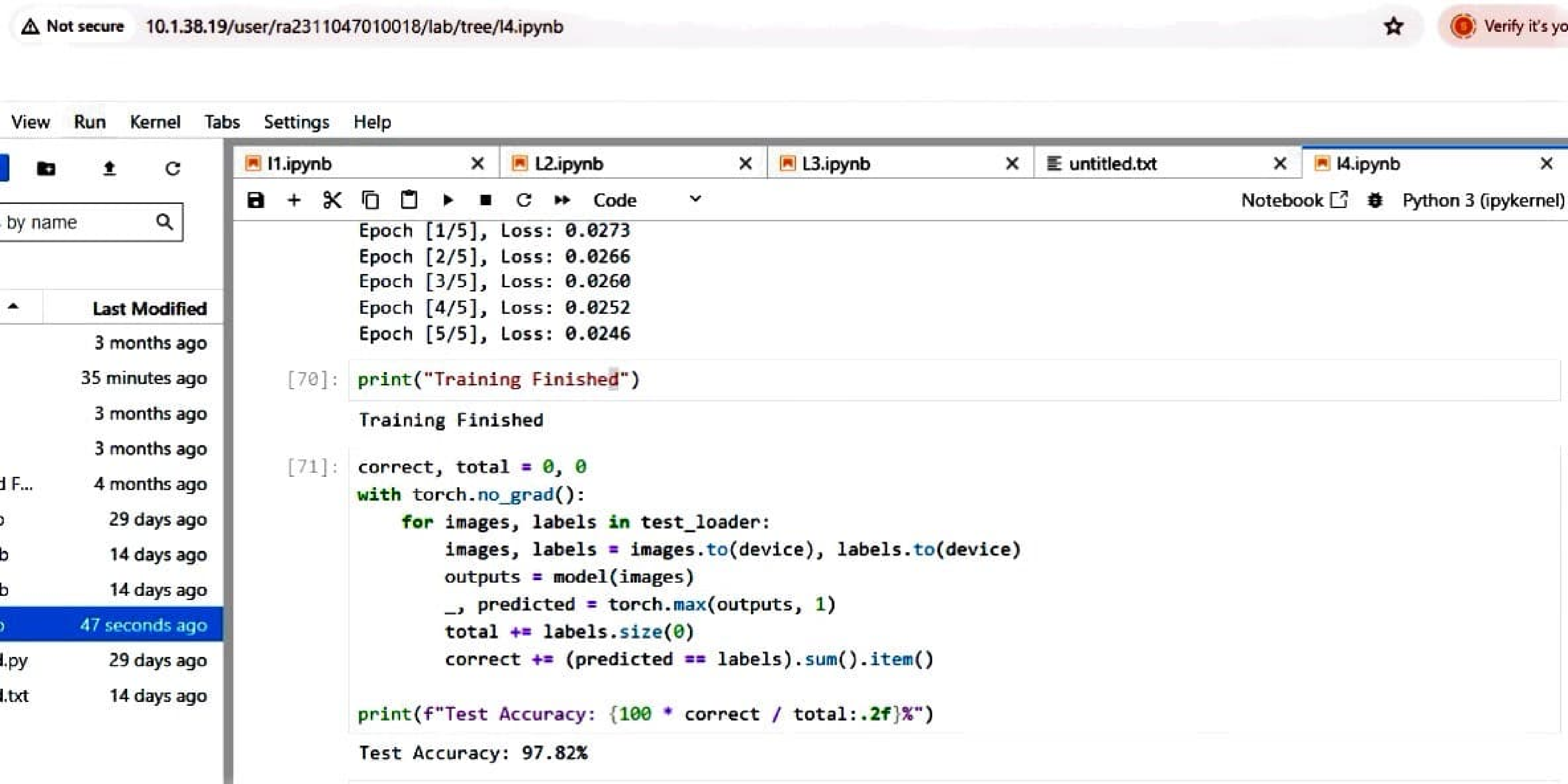
```
for epoch in range(epochs):
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

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

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{epochs}], Loss: {running_loss/len(train_loader):.4f}")
```



```
Epoch [1/5], Loss: 0.0273
Epoch [2/5], Loss: 0.0266
Epoch [3/5], Loss: 0.0260
Epoch [4/5], Loss: 0.0252
Epoch [5/5], Loss: 0.0246
```

```
[70]: print("Training Finished")
```

```
Training Finished
```

```
[71]: 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, 1)
              total += labels.size(0)
              correct += (predicted == labels).sum().item()

      print(f"Test Accuracy: {100 * correct / total:.2f}%")
```

```
Test Accuracy: 97.82%
```