Problem1

November 1, 2024

[1]:
```
#Library Imports
from torchvision import datasets
from torchvision.transforms import ToTensor
from torch.utils.data import DataLoader
import torch
from torch import nn
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
train_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=True)
test_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=False)

# Create DataLoaders for training and testing
train_loader = DataLoader(train_data, batch_size=1024, shuffle=True)
test_loader = DataLoader(test_data, batch_size=len(test_data), shuffle=False)
```

[3]:
```
# Define the MLP model with 2 hidden layers, both with 128 units
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.mlp = nn.Sequential(
            nn.Linear(16 * 16, 128),  # Input layer to first hidden layer
            nn.ReLU(),
            nn.Linear(128, 128),      # First hidden layer to second hidden␣
 ↪layer
            nn.ReLU(),
            nn.Linear(128, 128),      # Second hidden layer
            nn.ReLU(),
            nn.Linear(128, 10)        # Output layer for 10 classes
        )

    def forward(self, X):
        return self.mlp(self.flatten(X))
```

```
model = MLP().to('cuda')
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.5)
```

[4]:
```
%%time
epochs = 1000
CE = torch.zeros((epochs))
Training = torch.zeros((epochs))
Test = torch.zeros((epochs))
for epoch in range(epochs):
    cumulative_accuracy = 0
    cumulative_loss = 0
    for X, Y in train_loader:
        X, Y = X.to('cuda'), Y.to('cuda')
        out = model(X)
        loss = loss_fn(out, Y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        cumulative_loss += loss.item()
        cumulative_accuracy += (out.argmax(axis=1) == Y).sum().item()
    CE[epoch] = cumulative_loss / len(train_loader)
    Training[epoch] = cumulative_accuracy / len(train_data)
    with torch.no_grad():
        for Xt, Yt in test_loader:
            Xt, Yt = Xt.to('cuda'), Yt.to('cuda')
            test_out = model(Xt)
            test_accuracy_epoch = (test_out.argmax(axis=1) == Yt).sum().item() /
 ↪ len(test_data)
        Test[epoch] = test_accuracy_epoch
    if (epoch + 1) % 100 == 0:
        print(f"Epoch {epoch + 1}/{epochs} | Loss: {CE[epoch]:.4f} | Training␣
↪Accuracy: {Training[epoch]:.4f} | Test Accuracy: {Test[epoch]:.4f}")
```

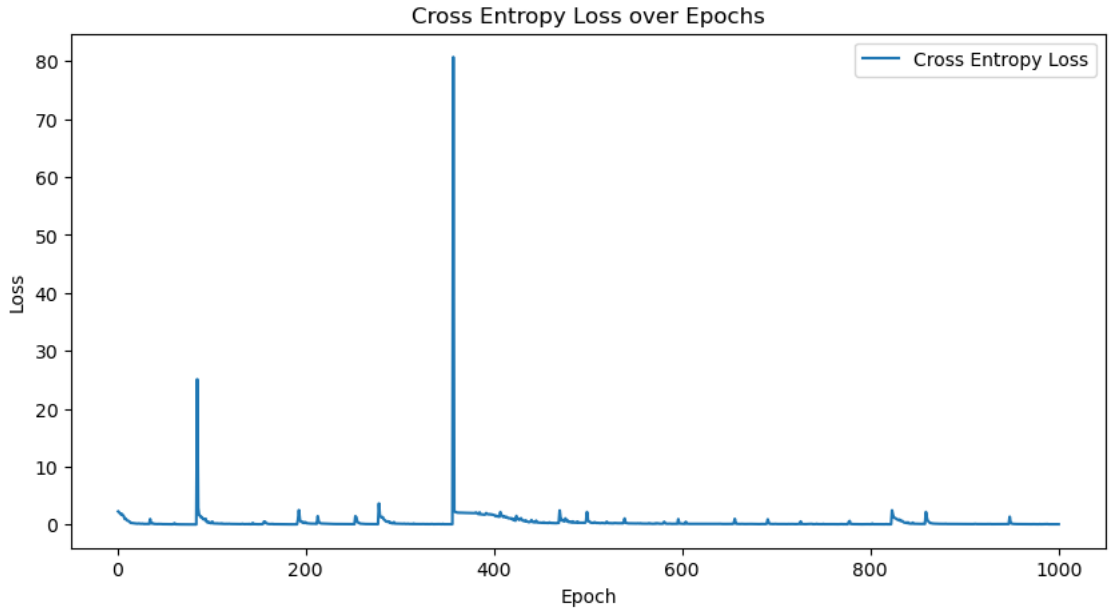
```
Epoch 100/1000 | Loss: 0.2390 | Training Accuracy: 0.9277 | Test Accuracy:
0.8416
Epoch 200/1000 | Loss: 0.4280 | Training Accuracy: 0.8624 | Test Accuracy:
0.8500
Epoch 300/1000 | Loss: 0.1539 | Training Accuracy: 0.9562 | Test Accuracy:
0.8974
Epoch 400/1000 | Loss: 1.6549 | Training Accuracy: 0.4063 | Test Accuracy:
0.4200
Epoch 500/1000 | Loss: 0.8089 | Training Accuracy: 0.7181 | Test Accuracy:
0.8052
Epoch 600/1000 | Loss: 0.1494 | Training Accuracy: 0.9557 | Test Accuracy:
0.8939
Epoch 700/1000 | Loss: 0.1030 | Training Accuracy: 0.9698 | Test Accuracy:
```

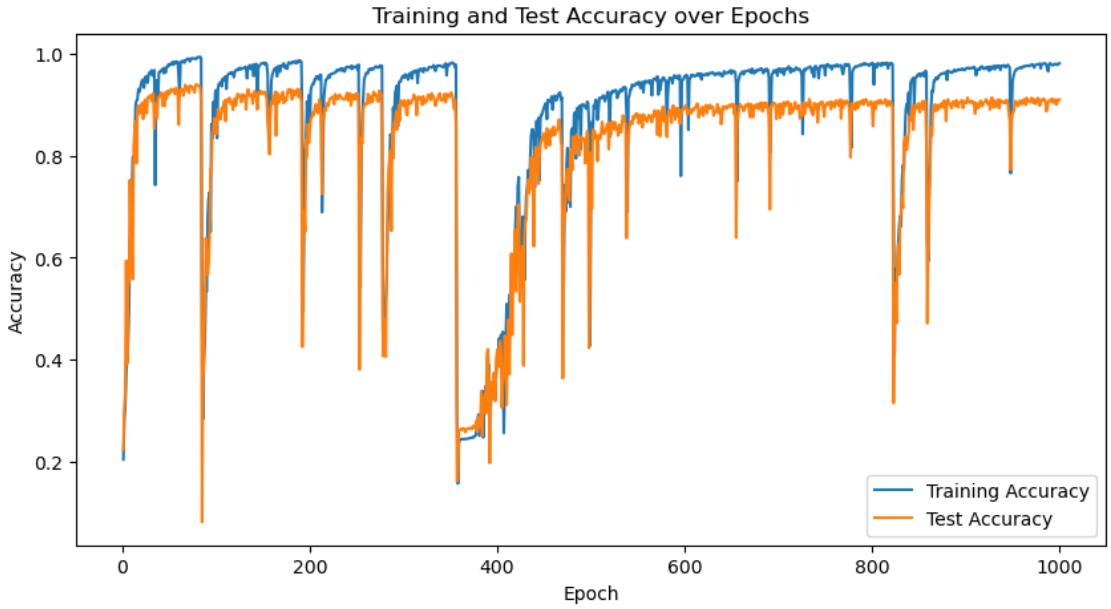
```
0.8939
Epoch 800/1000 | Loss: 0.0726 | Training Accuracy: 0.9783 | Test Accuracy:
0.8974
Epoch 900/1000 | Loss: 0.0965 | Training Accuracy: 0.9706 | Test Accuracy:
0.9023
Epoch 1000/1000 | Loss: 0.0694 | Training Accuracy: 0.9802 | Test Accuracy:
0.9093
CPU times: user 6min 2s, sys: 497 ms, total: 6min 3s
Wall time: 6min 3s
```

[5]:
```python
epochs_range = np.arange(1, epochs + 1)
CE_np = CE.cpu().numpy()
Training_np = Training.cpu().numpy()
Test_np = Test.cpu().numpy()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, CE_np, label="Cross Entropy Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Cross Entropy Loss over Epochs")
plt.legend()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, Training_np, label="Training Accuracy")
plt.plot(epochs_range, Test_np, label="Test Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training and Test Accuracy over Epochs")
plt.legend()
plt.show()
```


Cross Entropy Loss over Epochs
Cross Entropy Loss
Loss
Epoch
0
10
20
30
40
50
60
70
80
0
200
400
600
800
1000


Training and Test Accuracy over Epochs
Training Accuracy
Test Accuracy
Accuracy
Epoch
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

Problem2

November 1, 2024

[1]:
```
#Library Imports
from torchvision import datasets
from torchvision.transforms import ToTensor
from torch.utils.data import DataLoader
import torch
from torch import nn
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
train_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=True)
test_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=False)

# Create DataLoaders for training and testing
train_loader = DataLoader(train_data, batch_size=1024, shuffle=True)
test_loader = DataLoader(test_data, batch_size=len(test_data), shuffle=False)
```

[3]:
```
# Define the MLP model with 1 hidden layer of width 64, following the specified␣
 ↪structure
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.mlp = nn.Sequential(
            nn.Linear(16 * 16, 64),  # Input layer to hidden layer
            nn.ReLU(),
            nn.Linear(64, 64),       # Hidden layer
            nn.ReLU(),
            nn.Linear(64, 10)        # Output layer
        )

    def forward(self, X):
        return self.mlp(self.flatten(X))


model = MLP().to('cuda')
```

```
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.5)
```

[4]:
```
%%time
epochs = 1000
CE = torch.zeros((epochs))
Training = torch.zeros((epochs))
Test = torch.zeros((epochs))
for epoch in range(epochs):
    cumulative_accuracy = 0
    cumulative_loss = 0
    for X, Y in train_loader:
        X, Y = X.to('cuda'), Y.to('cuda')
        out = model(X)
        loss = loss_fn(out, Y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        cumulative_loss += loss.item()
        cumulative_accuracy += (out.argmax(axis=1) == Y).sum().item()
    CE[epoch] = cumulative_loss / len(train_loader)
    Training[epoch] = cumulative_accuracy / len(train_data)
    with torch.no_grad():
        for Xt, Yt in test_loader:
            Xt, Yt = Xt.to('cuda'), Yt.to('cuda')
            test_out = model(Xt)
            test_accuracy_epoch = (test_out.argmax(axis=1) == Yt).sum().item() /
 ↪ len(test_data)
        Test[epoch] = test_accuracy_epoch
    if (epoch + 1) % 100 == 0:
        print(f"Epoch {epoch + 1}/{epochs} | Loss: {CE[epoch]:.4f} | Training␣
↪Accuracy: {Training[epoch]:.4f} | Test Accuracy: {Test[epoch]:.4f}")
```

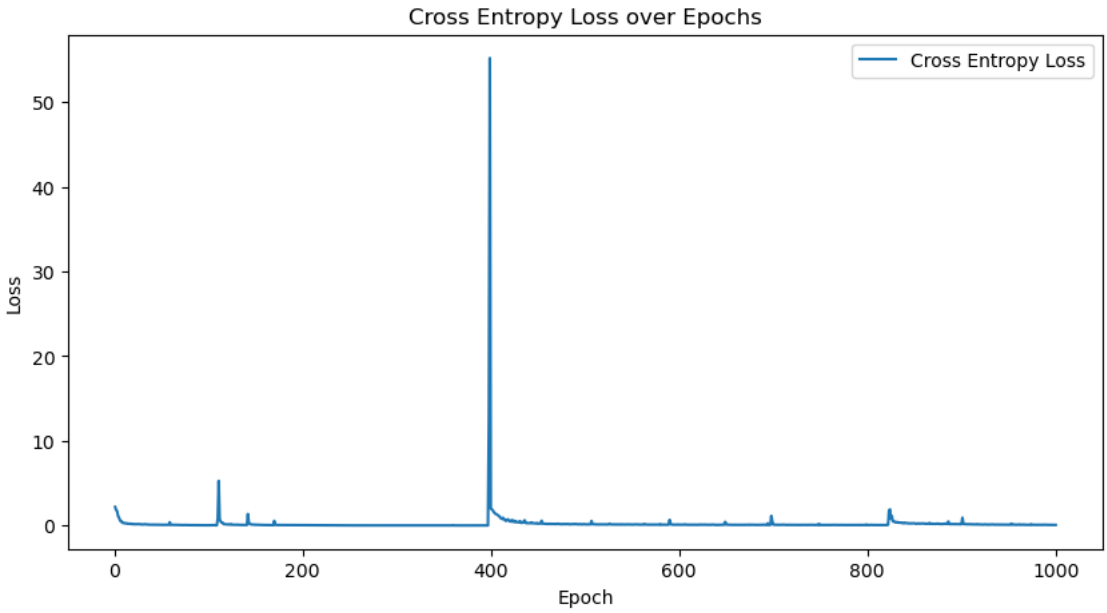
```
Epoch 100/1000 | Loss: 0.0295 | Training Accuracy: 0.9925 | Test Accuracy:
0.9402
Epoch 200/1000 | Loss: 0.0309 | Training Accuracy: 0.9914 | Test Accuracy:
0.9347
Epoch 300/1000 | Loss: 0.0082 | Training Accuracy: 0.9989 | Test Accuracy:
0.9387
Epoch 400/1000 | Loss: 2.0046 | Training Accuracy: 0.2369 | Test Accuracy:
0.2292
Epoch 500/1000 | Loss: 0.1458 | Training Accuracy: 0.9545 | Test Accuracy:
0.8904
Epoch 600/1000 | Loss: 0.0944 | Training Accuracy: 0.9686 | Test Accuracy:
0.9043
Epoch 700/1000 | Loss: 0.1129 | Training Accuracy: 0.9638 | Test Accuracy:
0.9033
Epoch 800/1000 | Loss: 0.0559 | Training Accuracy: 0.9813 | Test Accuracy:
```

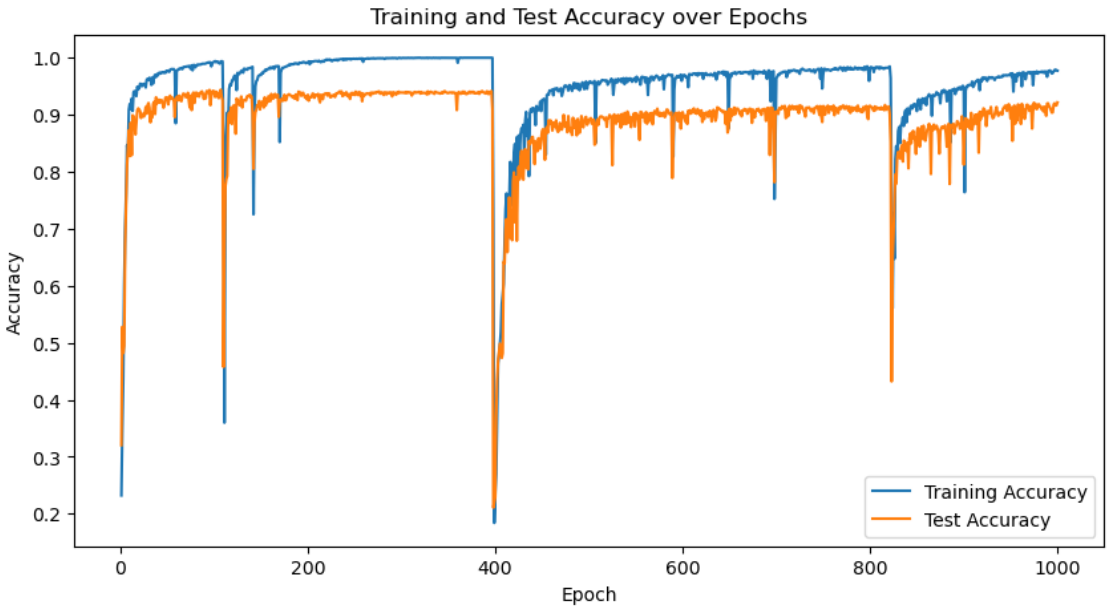
```
0.9153
Epoch 900/1000 | Loss: 0.1805 | Training Accuracy: 0.9468 | Test Accuracy:
0.8122
Epoch 1000/1000 | Loss: 0.0655 | Training Accuracy: 0.9770 | Test Accuracy:
0.9213
CPU times: user 6min 7s, sys: 1.13 s, total: 6min 8s
Wall time: 6min 7s
```

[5]:
```
epochs_range = np.arange(1, epochs + 1)
CE_np = CE.cpu().numpy()
Training_np = Training.cpu().numpy()
Test_np = Test.cpu().numpy()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, CE_np, label="Cross Entropy Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Cross Entropy Loss over Epochs")
plt.legend()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, Training_np, label="Training Accuracy")
plt.plot(epochs_range, Test_np, label="Test Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training and Test Accuracy over Epochs")
plt.legend()
plt.show()
```




Training and Test Accuracy over Epochs
Accuracy
Epoch
Training Accuracy
Test Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000

Problem3

November 1, 2024

[1]:
```
#Library Imports
from torchvision import datasets
from torchvision.transforms import ToTensor
from torch.utils.data import DataLoader
import torch
from torch import nn
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
train_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=True)
test_data = datasets.USPS(root='usps', download=True, transform=ToTensor(),␣
 ↪train=False)

# Create DataLoaders for training and testing
train_loader = DataLoader(train_data, batch_size=128, shuffle=True)
test_loader = DataLoader(test_data, batch_size=len(test_data), shuffle=False)
```

[3]:
```
# Define the MLP model with 2 hidden layers, both with 128 units
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.mlp = nn.Sequential(
            nn.Linear(16 * 16, 128),  # Input layer to first hidden layer
            nn.ReLU(),
            nn.Linear(128, 128),      # First hidden layer to second hidden␣
 ↪layer
            nn.ReLU(),
            nn.Linear(128, 128),      # Second hidden layer
            nn.ReLU(),
            nn.Linear(128, 10)        # Output layer for 10 classes
        )

    def forward(self, X):
        return self.mlp(self.flatten(X))
```

```
model = MLP().to('cuda')
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.5)
```

[4]:
```
%%time
epochs = 1000
CE = torch.zeros((epochs))
Training = torch.zeros((epochs))
Test = torch.zeros((epochs))
for epoch in range(epochs):
    cumulative_accuracy = 0
    cumulative_loss = 0
    for X, Y in train_loader:
        X, Y = X.to('cuda'), Y.to('cuda')
        out = model(X)
        loss = loss_fn(out, Y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        cumulative_loss += loss.item()
        cumulative_accuracy += (out.argmax(axis=1) == Y).sum().item()
    CE[epoch] = cumulative_loss / len(train_loader)
    Training[epoch] = cumulative_accuracy / len(train_data)
    with torch.no_grad():
        for Xt, Yt in test_loader:
            Xt, Yt = Xt.to('cuda'), Yt.to('cuda')
            test_out = model(Xt)
            test_accuracy_epoch = (test_out.argmax(axis=1) == Yt).sum().item() /
 ↪ len(test_data)
        Test[epoch] = test_accuracy_epoch
    if (epoch + 1) % 100 == 0:
        print(f"Epoch {epoch + 1}/{epochs} | Loss: {CE[epoch]:.4f} | Training␣
↪Accuracy: {Training[epoch]:.4f} | Test Accuracy: {Test[epoch]:.4f}")
```

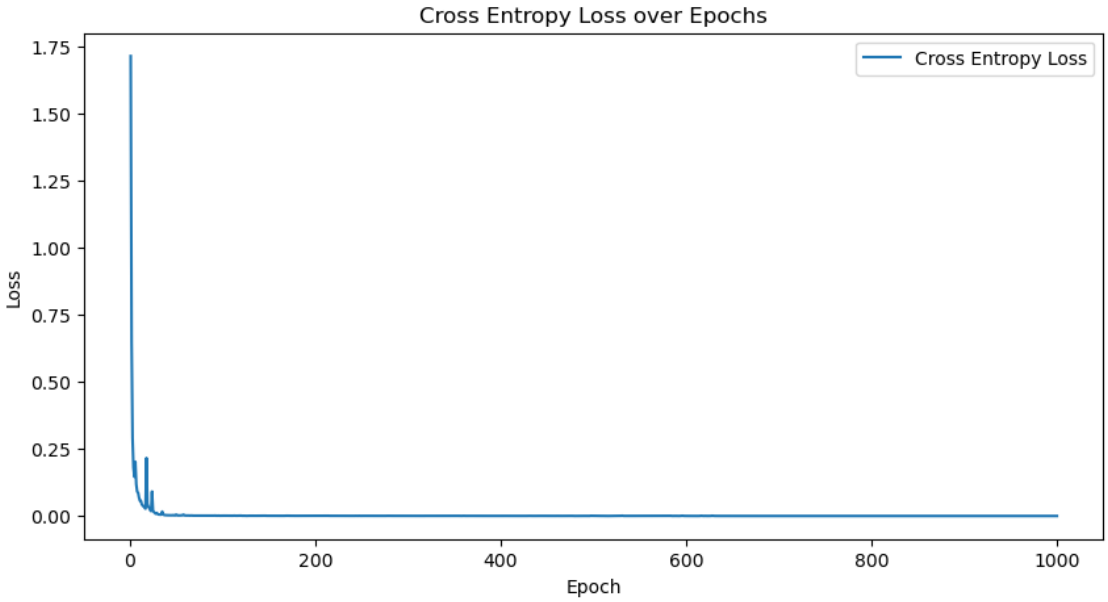
```
Epoch 100/1000 | Loss: 0.0010 | Training Accuracy: 0.9999 | Test Accuracy:
0.9447
Epoch 200/1000 | Loss: 0.0008 | Training Accuracy: 0.9999 | Test Accuracy:
0.9442
Epoch 300/1000 | Loss: 0.0005 | Training Accuracy: 0.9999 | Test Accuracy:
0.9427
Epoch 400/1000 | Loss: 0.0002 | Training Accuracy: 1.0000 | Test Accuracy:
0.9427
Epoch 500/1000 | Loss: 0.0003 | Training Accuracy: 0.9999 | Test Accuracy:
0.9427
Epoch 600/1000 | Loss: 0.0003 | Training Accuracy: 0.9999 | Test Accuracy:
0.9417
Epoch 700/1000 | Loss: 0.0000 | Training Accuracy: 1.0000 | Test Accuracy:
```

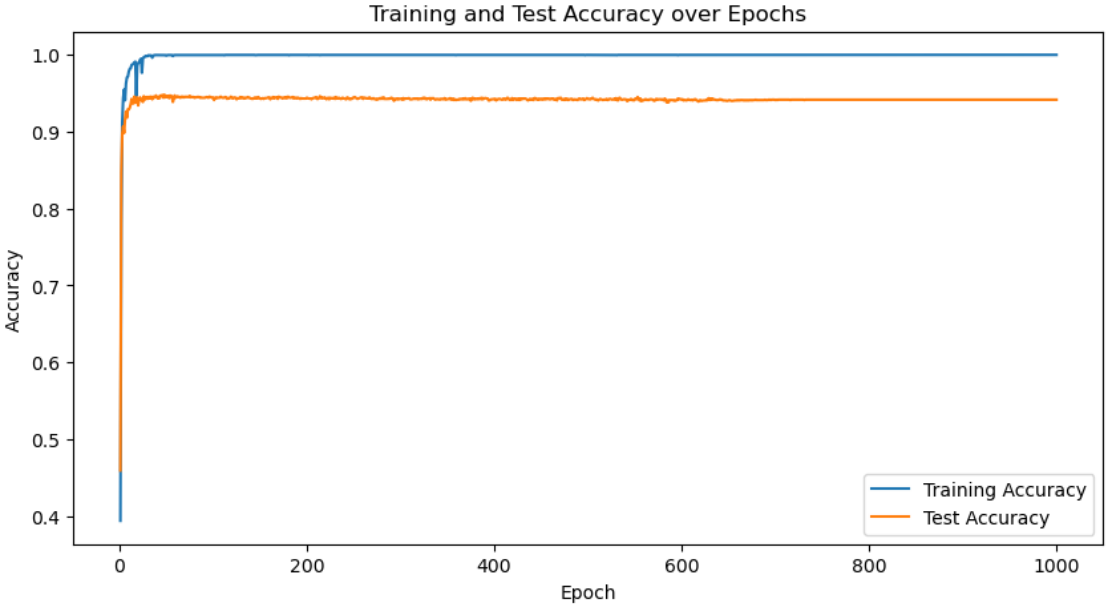
```
0.9412
Epoch 800/1000 | Loss: 0.0000 | Training Accuracy: 1.0000 | Test Accuracy:
0.9417
Epoch 900/1000 | Loss: 0.0000 | Training Accuracy: 1.0000 | Test Accuracy:
0.9417
Epoch 1000/1000 | Loss: 0.0000 | Training Accuracy: 1.0000 | Test Accuracy:
0.9417
CPU times: user 6min 51s, sys: 1.31 s, total: 6min 53s
Wall time: 6min 52s
```

[5]:
```
epochs_range = np.arange(1, epochs + 1)
CE_np = CE.cpu().numpy()
Training_np = Training.cpu().numpy()
Test_np = Test.cpu().numpy()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, CE_np, label="Cross Entropy Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Cross Entropy Loss over Epochs")
plt.legend()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(epochs_range, Training_np, label="Training Accuracy")
plt.plot(epochs_range, Test_np, label="Test Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("Training and Test Accuracy over Epochs")
plt.legend()
plt.show()
```


Cross Entropy Loss over Epochs
Cross Entropy Loss
Loss
Epoch
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000


Training and Test Accuracy over Epochs
Training Accuracy
Test Accuracy
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
200
400
600
800
1000