# Distributed and Parallel Computing Lab

## CS461 Lab11

**Name: Dipean Dasgupta** **ID:202151188**

**Task: Performance Analysis of Various Methods on GPU**

**Image Classification with the MNIST Dataset using CUDA (GPU)**

**Platform: Google Colab; GPU: T4**

**Link to File:**
[https://colab.research.google.com/drive/1VJBsWtMTP4ADTcr3E4jRmrLWl5xYJykP?usp=sharing](https://colab.research.google.com/drive/1VJBsWtMTP4ADTcr3E4jRmrLWl5xYJykP?usp=sharing)

**Process:**

Checking for availability of CUDA (GPU)

```
[1] import torch
    import torch.nn as nn
    from torch.utils.data import Dataset, DataLoader
    from torch.optim import Adam
    # Visualization tools
    import torchvision
    import torchvision.transforms.v2 as transforms
    import torchvision.transforms.functional as F
    import matplotlib. pyplot as plt

[2] device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    torch.cuda.is_available()

True
```

CUDA is available.

```
train_set= torchvision.datasets.MNIST("./data/", train=True, download=True)
valid_set= torchvision.datasets.MNIST("./data/", train=False, download=True)

Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz
100%|| 9.91M/9.91M [00:00<00:00, 16.3MB/s]
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw
```

Downloading data from, the MNIST dataset under torchvision for classification.

```
[29] model= nn.Sequential(*layers)
     model

Sequential(
  (0): Flatten(start_dim=1, end_dim=-1)
  (1): Linear(in_features=784, out_features=512, bias=True)
  (2): ReLU()
  (3): Linear(in_features=512, out_features=512, bias=True)
  (4): ReLU()
  (5): Linear(in_features=512, out_features=10, bias=True)
)
```

Model Compilation

```
[32] model = torch.compile(model)

[33] loss_function= nn.CrossEntropyLoss()

[34] optimizer = Adam(model.parameters())

[35] train_N =len(train_loader.dataset)
     valid_N = len(valid_loader.dataset)
```

Training the model

```
def train():
    loss = 0
    accuracy = 0

    model.train()
    for x, y in train_loader:
        x, y = x.to(device), y.to(device)
        output = model(x)
        optimizer.zero_grad()
        batch_loss = loss_function(output, y)
        batch_loss.backward()
        optimizer.step()

        loss += batch_loss.item()
        accuracy += get_batch_accuracy(output, y, train_N)
    print('Train - Loss: {:.4f} Accuracy: {:.4f}'.format(loss, accuracy))
```

Defining function for training the model

```
def validate():
    loss = 0
    accuracy = 0

    model.eval()
    with torch.no_grad():
        for x, y in valid_loader:
            x, y = x.to(device), y.to(device)
            output = model(x)

            loss += loss_function(output, y).item()
            accuracy += get_batch_accuracy(output, y, valid_N)
    print('Valid - Loss: {:.4f} Accuracy: {:.4f}'.format(loss, accuracy))
```

Function for validating the model

```
W1119 11:34:37.627000 643 torch/_dynamo/convert_frame.py:1125]
Train - Loss: 375.3086 Accuracy: 0.9388
Valid - Loss: 29.8273 Accuracy: 0.9706
Epoch: 1
Train - Loss: 156.2106 Accuracy: 0.9745
Valid - Loss: 32.3706 Accuracy: 0.9691
Epoch: 2
Train - Loss: 112.1964 Accuracy: 0.9811
Valid - Loss: 30.6006 Accuracy: 0.9734
Epoch: 3
Train - Loss: 83.6986 Accuracy: 0.9857
Valid - Loss: 28.1743 Accuracy: 0.9757
Epoch: 4
Train - Loss: 65.6404 Accuracy: 0.9885
Valid - Loss: 23.5482 Accuracy: 0.9808
Epoch: 5
Train - Loss: 52.6170 Accuracy: 0.9906
Valid - Loss: 25.2658 Accuracy: 0.9801
Epoch: 6
Train - Loss: 52.3083 Accuracy: 0.9914
Valid - Loss: 30.5243 Accuracy: 0.9768
Epoch: 7
Train - Loss: 38.5636 Accuracy: 0.9933
Valid - Loss: 29.8365 Accuracy: 0.9828
Epoch: 8
Train - Loss: 40.1823 Accuracy: 0.9933
Valid - Loss: 28.8597 Accuracy: 0.9819
Epoch: 9
Train - Loss: 33.9273 Accuracy: 0.9941
Valid - Loss: 32.2365 Accuracy: 0.9812
```

10 epochs were run on which on the final we got accuracy of 99.41% in train and 98.12% in validatation.

Checking if the trained model Predicts correctly on test images:

```
[42] prediction.argmax(dim=1,keepdim=True)

tensor([[5]], device='cuda:0')

y_0

5
```

We see, the original test image has written 5 and the model has also predicted 5.

So, the training of the model is a success as it is predicting correctly.

The memory is then cleared after successful execution of the lab.

**----------END of ASSIGNMENT----------**