Assignment- Neural Networks & PyTorch

Objective

In this assignment, you will:

- Learn core neural network concepts by implementing them from scratch
- Use PyTorch to build, train, and evaluate a neural network
- Work with real-world datasets like MNIST or Fashion-MNIST
- Gain hands-on experience with custom training loops, optimizers, and loss functions

---

Task 1: Setup & Dataset

- Install PyTorch and necessary libraries
- Load **Fashion-MNIST** dataset using `torchvision.datasets`
- Normalize the data (zero mean, unit variance)
- Use `DataLoader` to create train and test batches

Deliverables:

- Code to load and normalize dataset
- Output showing shape of one batch from `DataLoader`

Task 2: Define a Custom Neural Network

Create a feedforward neural network `MyNeuralNet` using `nn.Module` with the following architecture:

- **Input Layer**: 784 units (28×28 image flattened)
- **Hidden Layer 1**: 128 neurons with ReLU
- **Hidden Layer 2**: 64 neurons with ReLU
- **Output Layer**: 10 neurons (for 10 classes)

Use `nn.CrossEntropyLoss` for classification.

Deliverables:

- Code for `MyNeuralNet` class
- Implementation of `forward()` function

Task 3: Training Loop from Scratch

Implement a custom training loop using:

- `Adam` optimizer (`lr = 0.001`)
- `nn.CrossEntropyLoss` for computing loss
- Manual loop using `zero_grad()`, `loss.backward()`, and `optimizer.step()`

Train for 5–10 epochs and report:

- Training loss at each epoch
- Accuracy on train and test set after training

Task 4: Evaluation and Visualization

- Plot **loss vs epochs**
- Compute and print:
  - Test accuracy
  - Confusion matrix
  - 10 misclassified images with predicted vs actual labels

Deliverables:

- Confusion matrix (use `sklearn.metrics` if needed)
- Matplotlib plots of misclassified images

Task 5: Experiments

Choose **any two** of the following experiments:

- A. Add dropout between layers and compare performance
- B. Try a different optimizer like SGD and observe convergence
- C. Add batch normalization
- D. Use weight initialization (e.g., Xavier)
- E. Build a 1D CNN version for the same dataset
- F. Compare your custom model to `torchvision.models.resnet18` fine-tuned on Fashion-MNIST

Deliverables:

- Accuracy and loss comparison
- Observations on why performance changed

Bonus Challenge

Implement your own version of the forward and backward pass **without using `nn.Module` or autograd**, using only tensors and `.backward()` manually.