CSC 790 Homework 2: Implementing a 3-Layer Convolutional Neural Network

Objective:

Construct and train a 3-layer convolutional neural network using PyTorch to classify images from the CIFAR-10 dataset.

Requirements:

1. Setup: (10 points)

- Use the CIFAR-10 dataset available directly via ‘torchvision.datasets‘.
- Split the dataset into training and testing sets.

2. Model Architecture: (30 points)

First Layer:

- Convolutional Layer with 16 filters, kernel size of 3x3, stride of 1, and padding of 1.
- ReLU Activation Layer.
- MaxPooling Layer with kernel size of 2x2 and stride of 2.

Second Layer:

- Convolutional Layer with 32 filters, kernel size of 3x3, stride of 1, and padding of 1.
- ReLU Activation Layer.
- MaxPooling Layer with kernel size of 2x2 and stride of 2.

Third Layer:

- Convolutional Layer with 64 filters, kernel size of 3x3, stride of 1, and padding of 1.
- ReLU Activation Layer.
- MaxPooling Layer with kernel size of 2x2 and stride of 2.

Output Layer:

- Flatten the output from the last MaxPooling layer.
- Fully connected layer to map the flattened output to 10 classes (corresponding to CIFAR-10 classes).

3. Training: (20 points)

- Use Cross-Entropy Loss.
- Use the Adam optimizer.
- Train the model for at least 10 epochs.
- Print the training loss and accuracy after each epoch.

4. Testing: (10 points)

- Evaluate the model on the test set after training.
- Print the test accuracy.

5. Report: (30 points)

Submit a report including:

- The final architecture of the model.
- Training process documentation (include graphs of training loss and accuracy if possible).
- A discussion on how different layers affect the learning and performance of the network.
- Any challenges faced and how they were overcome.

6. Extra Points: (20 points) Train the model using three different learning rates: 0.1, 0.01, and 0.001. Include figures that record both training and testing loss in your report. Analyze how the learning rate impacts the training and testing accuracy.

Hints and Tips: - Make sure to shuffle the training data to avoid bias during training. - Use data loaders from ‘torch.utils.data.DataLoader‘ for batching.