EE5178: Modern Computer Vision

Programming Assignment 1

Instructions

- You are required to use Python and PyTorch for this assignment.
- Read the problem statement thoroughly to understand the complete procedure.
- Comment your code generously and include appropriate titles for all figures.
- Post any doubts on Moodle discussion threads. This will also benefit your peers.
- Submit your codes (the actual notebook with all the cells executed and outputs displayed, along with a PDF version) and your assignment report in a **single zip file** titled `PA1_RollNumber.zip` via the submission link provided on Moodle.

Preliminaries

The objective of this assignment is to experiment with Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) for image classification. Please use PyTorch, an open-source library for deep learning, to implement your models. You may use any other necessary libraries.

Dataset: You will implement an image classifier using the popular CIFAR-10 dataset. This dataset contains 60,000 32 $\times$ 32 color images across 10 different classes, with 6,000 images per class. The classes include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

Note: PyTorch provides the CIFAR-10 dataset in the torchvision.datasets module, which can be used directly for loading the dataset.

---

1 MLP

In this section, you will implement a simple MLP baseline for image classification on the CIFAR-10 dataset.

1.1 Architecture and Training

- The input to the network is a 32 $\times$ 32 $\times$ 3 image, flattened into a 3072-dimensional vector.
- The network consists of three hidden layers: h_1 (500 units), h_2 (250 units), and h_3 (100 units). Use ReLU activation for all hidden layers.
- The output layer consists of 10 units, one for each class. Apply the softmax function to obtain class probabilities.
- Use cross-entropy loss between the predicted output $\hat{y}_i$ and the true one-hot encoded label y_i.
- Train the model using the SGD optimizer.

Deliverables:

1. Plot training error, validation error, and prediction accuracy over training iterations.
2. Report the average test accuracy on the 10,000 test images.
3. Display randomly selected test images, showing both the true and predicted class labels.
4. Report the confusion matrix (a 10 $\times$ 10 matrix where rows represent true labels and columns represent predicted labels).
5. Experiment with batch normalization. Analyze its effect on test accuracy and training time.

2 CNN

In this section, you will implement the VGG11 architecture from scratch for image classification.

2.1 Experimental Settings

- Since the dataset contains 10 classes, modify (or add an extra layer to) the last layer of VGG11 accordingly.
- Experiment with different learning rates (e.g., 0.001, 0.0001) and batch sizes. Report the best-performing hyperparameters.
- Use the CrossEntropy loss function and the SGD optimizer.
- You may experiment with modifying the fully connected layers (e.g., changing the number of neurons or layers).
- Optionally, remove the last max-pooling layer before the fully connected layers.

Deliverables:

1. Report training settings (e.g., learning rate, number of training epochs) for all configurations tested.
2. Discuss any bottlenecks or challenges faced during training and testing.
3. Provide training loss and accuracy plots.
4. Report the test dataset accuracy.
5. Display visual results for five test images per class.
6. Test the trained model on five randomly downloaded images (or images captured from a smartphone/camera) and analyze the results.
