ECES 681Computer Vision Homework 3

Somayeh Keshavarz

This assignment is due on **Aug 7, at 11:59 PM EST**.

Submission:

You need to submit:

- A PDF (hw3_writeup.pdf) containing answers to all questions, along with plots and visualization.
- Python Code (hw3_code.ipynb) including all functions.
- Ensure that the **Jupyter Notebook** is self-contained, functional, and includes all outputs.
- **For this homework, you can use any deep learning libraries such as PyTorch, Keras or TensorFlow.**

Problem 1

Implement a Two-Layer Network.

In this problem, you need to implement a two-layer network.

This network has two FC layers with one ReLU activation layer: input -> FC layer -> ReLU layer -> FC layer -> scores. Write two_layer_net.ipynb function.

Then you need to train your implemented two-layer network on the CIFAR-10 dataset for image classification. This dataset consists of 32 × 32 RGB images of 10 different categories. It provides 50,000 training images and 10,000 test images. Figure 1 shows a few example images from the dataset.

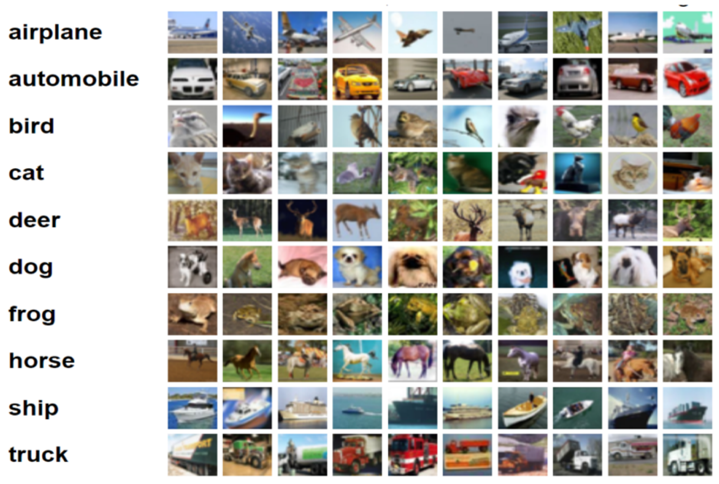

Figure 1: Example images for the CIFAR-10 dataset.

First, You need to download and unpack the CIFAR10 dataset. After initializing the two-layer network and the optimizer using stochastic gradient descent, implement a training loop for training.

The loss should be the sum of two terms:
• A data loss term, which is the softmax loss between the model's predicted scores and the ground-truth image labels.
• A regularization loss term, which penalizes the L2 norm of the weight matrices of all the fully-connected layers of the model. You should not apply L2 regularization to the biases.

After your implementation, it is time to train the network. Print out training losses and train and val set accuracy as it trains. After training concludes, also make a plot of the training losses as well as the training and validation-set accuracy of the model during training. You may see a plot that looks like this:

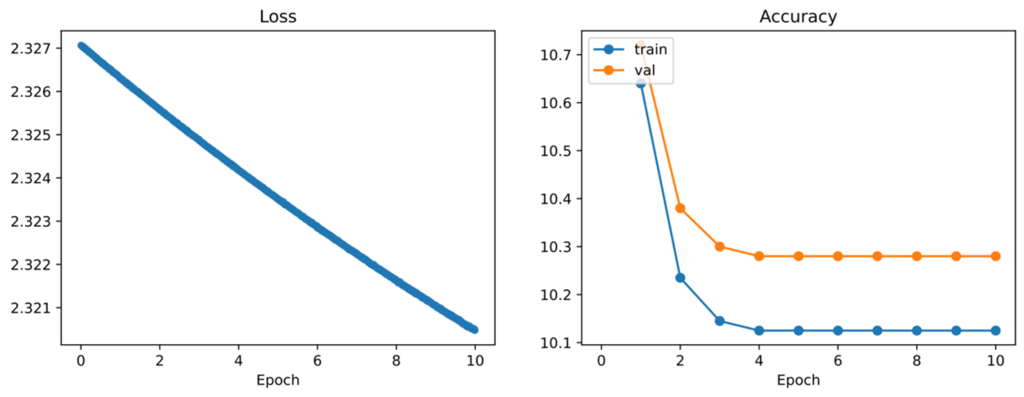

Figure 2: Trainng loss and train-validation accuracy.

Unfortunately, it seems that your model is not training very effectively – the training loss has not decreased much from its initial value of 2.3, and the training and validation accuracies are very close to 10% which is what we would expect from a model that randomly guesses a category label for each input.

You will need to tune the hyperparameters of your model in order to improve it. Try changing the hyperparameters of the model. You can consider changing any of the following hyperparameters:

• num_train: The number of images to use for training
• hidden_dim: The width of the hidden layer of the model
• batch_size: The number of examples to use in each minibatch during SGD
• num_epochs: How long to train. An epoch is a single pass through the training set.
• learning_rate: The learning rate to use for SGD
• reg: The strength of the L2 regularization term

You should tune the hyperparameters and train a model that achieves at least 40% on the validation set. In your homework submission, include the loss / accuracy plot for your best model. After tuning your model, run your best model exactly once on the test set.
Your model should not take an excessive amount of time to train. For reference, our hyperparameter settings achieve 42% accuracy on the validation set in less than 1 minute of training on a desktop with an Intel i9 CPU.

Problem 2

Network (CNN) with a Fully Connected Layer

In this assignment, you will design, implement, and train a two-layer Convolutional Neural Network (CNN)followed by one fully connected (dense) layer at the end. Your goal is to improve the model's performance using techniques learned in class.
Instructions:

1. **Build the CNN Architecture**
    - The network should consist of **two convolutional layers** followed by at least **one fully connected layer** at the end.
    - You may use **ReLU activation** for non-linearity and **max pooling** to reduce spatial dimensions.
    - Feel free to experiment with **kernel sizes, strides, and padding** to improve performance.
2. **Enhance Model Performance**
    - Implement any **optimization techniques** learned in class, such as:
        - **Batch Normalization** (for stable and faster training)
        - **Dropout** (to prevent overfitting)
        - **Data Augmentation** (to increase dataset diversity)
    - You may also experiment with **different learning rates, optimizers, or regularization techniques** to achieve higher accuracy.
3. **Train Your Model**
    - Use the **same dataset** as in Problem 1.
    - Follow all the **training steps and preprocessing** mentioned in Problem 1.
    - Ensure proper **train-validation splitting** to evaluate the model correctly.
4. **Evaluate Your Model**
    - Report **training and validation accuracy/loss** using appropriate plots.
    - Report **test accuracy/loss** using using the best setting.
    - Compare your results with those from Problem 1.
    - Discuss how the addition of CNN layers and techniques like dropout or batch

normalization affected performance.