Assignment 3: Self-supervised and transfer learning on CIFAR10

Due date: Wednesday, November 6, 11:59:59 PM

This assignment aims to help you gain experience with PyTorch, learn how to use pre-trained models provided by the deep learning community, and adapt these models to new tasks and losses. You will use a simple self-supervised rotation prediction task to pre-train a feature representation on the CIFAR10 dataset without using class labels, and then fine-tune this representation for CIFAR10 classification.

Source: https://www.cs.toronto.edu/~kriz/cifar.html

You will use PyTorch to train a model on a self-supervised task, fine-tune a subset of the model's weights, and train a model in a fully supervised setting with different weight initializations. You will be using the CIFAR10 dataset, which is a dataset of small (32x32) images belonging to 10 different object classes. For self-supervised training, you will ignore the provided labels; however, you will use the class labels for fine-tuning and fully supervised training.

The model architecture you will use is ResNet18. We will use the PyTorch ResNet18 implementation, so you do **not** need to create it from scratch.

The self-supervised training task is image rotation prediction, as proposed by Gidaris et al. in 2018. For this task, all training images are randomly rotated by 0, 90, 180, or 270 degrees. The network is then trained to classify the rotation of each input image using

cross-entropy loss by treating each of the 4 possible rotations as a class. This task can be treated as pre-training, and the pre-trained weights can then be fine-tuned on the supervised CIFAR10 classification task.

The top-level notebook (`A3.ipynb`) will guide you through all the steps of training a ResNet for the rotation task and fine-tuning on the classification task. You will implement the data loader, training and fine-tuning steps in Pytorch based on the starter code. In detail, you will complete the following tasks:

1. Train a ResNet18 on the Rotation task. Based on the CIFAR10 dataloader, you will first generate the rotated images and labels for the rotation task. You will train a ResNet18 on the rotation task, report the test performance and store the model for the fine-tuning tasks. For the test performance, find the lowest test loss and report the corresponding accuracy. **The expected rotation prediction accuracy on the test set should be around 78%**.

2. Initializing from the Rotation model or from random weights, fine-tune only the weights of the final block of convolutional layers and linear layer on the supervised CIFAR10 classification task. Report the test results and compare the performance of these two models. **The expected test set accuracy for fine-tuning the specified layers of the pre-trained model should be around 60%.**

3. Initializing from the Rotation model or from random weights, train *the full network* on the supervised CIFAR10 classification task. Report the test results and compare the performance of these two models. **The expected test set accuracy for fine-tuning the whole pre-trained model should be around 80%.**

Extra Credit (13 points)

- In Figure 5(b) from the Gidaris et al. paper, the authors show a plot of CIFAR10 classification performance vs. number of training examples per category for a supervised CIFAR10 model vs. a RotNet model with the final layers fine-tuned on CIFAR10. The plot shows that pre-training on the Rotation task can be advantageous when only a small amount of labeled data is available. Using your RotNet fine-tuning code and supervised CIFAR10 training code from the main assignment, try to create a similar plot by performing supervised fine-tuning/training on only a subset of CIFAR10.
- Use a more advanced model than ResNet18 to try to get higher accuracy on the rotation prediction task, as well as for transfer to supervised CIFAR10 classification.

- If you have a good amount of compute at your disposal, try to train a rotation prediction model on the larger ImageNette dataset (still smaller than ImageNet, though).

Submission Instructions:

The assignment deliverables are as follows.

- Upload three files to CU:
    1. **sid_a3_output.pdf**: Your IPython notebook with output cells converted to **PDF format**
    2. **sid_a3.ipynb**: Your code
    3. **sid_a3_report.pdf**: Your assignment report **in PDF format**