Homework IV

Due date: 11/5/2024 at 11:59pm

Submission Guidelines: 1. Put all the documents into one folder, name that folder as firstnamelastname_UIN and compress it into a .zip file.

2. For coding problems, your submission should include **code files** (either .py or .ipynb), and also a **pdf file** (in one file together with other non-coding questions) to report the results that are required in the question. **You don't have to submit the trained model and the dataset which might be potentially large.**

Problem 1: Deep Residual Networks for CIFAR-10 Image Classification(60 points)

In this assignment, we are using a classic convolutional neural networks(ResNet) to perform a 10-class classification task on CIFAR-10 dataset with PyTorch. In this classification task, models will take a 32×32 image with RGB channels as inputs and classify the image into one of ten pre-defined classes. The "ResNet" folder provides the starting code. In this assignment, you need to use a GPU.

(a) (10 points) Download the CIFAR-10 dataset and complete the code for the "load_data" function in "DataReader.py". For the dataset, you can download any version. But make sure you write corresponding code in "DataReader.py" to read it. Please do not use torchvision.dataset and torchvision.transforms in part(a) and (b).

(b) (10 points) Implement data augmentation. To complete "ImageUtils.py", you will implement 3 augmentation processes(random crop; random flip; normalization) for a single image using numpy. Note: for normalization, please do a per-channel normalization, i.e. for each channel, subtract its mean and divide it by its standard deviation.

(c) (10 points) Complete "Model.py". In this part, you're asked to implement training pipeline and testing pipeline.

(d) (15 points) The "NetWork.py" contains standard ResNet18 model. For more detail of ResNet please refer to the reading materials(e.g. Figure5 Left in [1]; Figure4(a) in [2]). You are asked to compare (i) ResNet18 with (ii) ResNet18 without any residual connection and (iii) ResNet18 without any batchnorm.
Hyper-parameters: number of epochs: 100; learning rate: tune from {0.003, 0.01, 0.03}, lr scheduler: for every 10 epochs decreases 1.5 times; optimizer: Adam; batchsize: 128; others: by default.

(e) (15 points) For all the three models, train with the best hyperparameters chosen from (d), plot the training loss and testing loss curves (for test loss, compute every 10 epochs), report your final testing accuracy.

Reading Materials:

(1) Deep Residual Learning for Image Recognition (`https://arxiv.org/abs/1512.03385`)

(2) Identity Mappings in Deep Residual Networks (`https://arxiv.org/abs/1603.05027`)

(3) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (`https://arxiv.org/abs/1502.03167`)

Problem 2: Optimizing AUC on Imbalanced MedMNIST Dataset with LibAUC(40 points)

The Receiver Operating Characteristic (ROC) curve is a graphical representation used to assess the performance of a binary classification model. It plots the true positive rate against the false positive rate across different decision thresholds. A model with better performance will have a curve closer to the top-left corner, indicating high true positive rates with low false positives. The Area Under the ROC Curve (AUROC) is a single scalar value summarizing this performance, where a higher AUC indicates better model discrimination.
The Precision-Recall Curve (PRC) is also essential in machine learning classification task, especially for imbalanced datasets or when dealing with rare events. Unlike accuracy or the ROC curve, which can give overly optimistic views of performance when one class dominates, PRC focuses specifically on the performance for the positive class, making it more informative in certain scenarios.
Recall that

$$TPR = \frac{\text{TruePositive}}{\text{TruePositive+FalseNegative}}, FPR = \frac{\text{FalsePositive}}{\text{TrueNegative+FalsePositive}}$$

and

$$Precision = \frac{\text{TruePositive}}{\text{TruePositive+FalsePositive}}, Recall = \frac{\text{TruePositive}}{\text{TruePositive+FalseNegative}}$$

In this task, you are asked to maximize AUROC and AUPRC (area under precision-recall curve) on PneumoniaMNIST (one of the MedMNIST datasets) using LibAUC library.

(a) (10 points) You are asked to (i) load the PneumoniaMNIST dataset from medmnist library (ii) perform data augmentation (totensor+randomcrop+ randomflip+resize for train set and totensor+resize for test set, refer to codebase).

(b) (20 points) Train ResNet18 models with (i)APLoss and SOAP optimizer with 'Adam' mode. (Note: AP means Average Precision, which is an estimator of AUPRC) (ii) AUCMLoss and PESG optimizer with 'Adam' mode. (iii) CE loss and Adam optimizer.
Hyperparameters: total epochs: 15; learning rate: tune from {5e-4, 1e-3, 2e-3} ({0.02, 0.05, 0.1} for PESG) ; lr scheduler: decay 10 times at epoch 9; batchsize: 64; weight decay: 0; gamma for APLoss: tune from {0.2, 0.5, 0.8}; others: refer to the code base

(c) (10 points) Plot the curves for AUPRC and AUROC on both the train set and the test set during training of all models with tuned hyperparameter(i.e. there're in total 12 curves). Discuss your observations of the curves.

Note: 1. For code base, please refer to the 2nd and the 3rd tutorials in `https://github.com/Optimization-AI/LibAUC/tree/1.4.0/examples`
2. The codebase is based on an older version so you may need to make slight modifications to work with the newer version of LibAUC.
3. In this problem you can import ResNet18 Model from LibAUC as in code base.
4. Replace the first convolution layer of ResNet18(model.conv1) with a convolution layer of only 1 input channel(original ResNet is designed for RGB image input)
5. To install MedMNIST, run 'pip install medmnist'.
6. There is a slight API mismatch between MedMNIST and LibAUC, one easy way is to modify the return of function '__getitem__' in class 'MedMNIST2D' in dataset.py of MedMNIST to be "return index, img, target"