Assignment 5
Due May 30, 2018 at 11.59 PM via ilearn Assignment-5 Submission folder

Independent Reading. ImageNet Classification with Deep Convolutional Neural Networks (papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf)

Problem [10 pts] In this assignment, you will be training a Convolutional Neural Network (CNN) from scratch. Recall, in Assignment-4, you used pre-trained weights of ResNet50 for feature extraction and trained just a logistic regression model, which is essentially the final layer of a CNN used for classification. In this assignment, you will be using the CIFAR-10 dataset, which contains 32×32 images divided into 10 categories. The training set contains $50,000$ images and the test set contains $10,000$ images. The dataset is already prepared for you and available on iLearn as well as the UCR EE server under /data/ece243/cifar10. The training images are in cifar10/train_images and the test images are in cifar10/test_images. The files cifar10_train_labels.mat and cifar10_test_labels.mat contains the labels for the train and test set respectively in the same order as in the images in the folder.

The starter code for this problem is `train.py`. Before starting to fill up this code, please execute the generate_traintestlist.py code after mentioning the path at the top of the code to the cifar10 dataset in the system you are working. This will generate two files namely train.list and test.list where each line will be of the following format -

<path/to/image/> <label>

After this, you need to fill in the following.

[5 pts] Fill in function named `inference` in `cnn_model.py`. The CNN model should be written in this function. The function takes as input a tensor with dimension $(B, 32, 32, 3)$, where B is the batch size. The output should be a matrix of dimension $(B, 10)$ representing the confidence (before applying the softmax function) for the 10 categories. Your CNN should have 4 convolutional layers with ReLU, Dropout, Batch Normalization and Max pooling operations in each (if required), followed by a classifier layer at the end. You can use the function `_variable_with_weight_decay` in `cnn_model.py` to initialize weights of the convolutional layers. You are allowed to choose stride, weight decay, dropout rate, filter kernel size, number of output feature maps for each layer and other hyper-parameters. FYI: A 4 layer network (written properly) should be able to obtain at least an accuracy of 65% on the test set.

[0.5 pts] Fill in loss function with cross-entropy loss. Please note that the code to add the regularizer loss is also completed just after ce_loss.

[1 pts] Fill in the function `read_data` to read images and the labels for the positions mentioned in readpos variable. You may add any data augmentation techniques to obtain better performance.

[1 pts] Fill in to obtain accuracy of current batch of data.

[1.5 pts] Fill in to obtain test accuracy over the entire test set and append it to the test_accuracy variable.

[1 pts] Plot training loss, train accuracy and test accuracy from the saved variables. Can you infer based on the plots, whether the model is overfitted, under-fitted or perfectly fitted ?