ELEC5698/CMSI5634 Deep Learning Applications Spring 2020

Project Assignment #2

Part 1: Baseline Implementation of the workflow for CIFAR-10 image classification

1.1 Data Preparation:

a. Load the CIFAR-10 data set. Find how many images are in training and testing set, respectively. Also, find what type of labeling is the data set using.
b. Show the first 5 images in the training set, display them in one row, with the corresponding class label below each image.
c. Show the first 5 images in the testing set, display them in a second row, with the corresponding class label below each image.
d. Normalize all image pixels to the range of 0-1.

1.2 Model Selection:

a. Build a CNN model with the following structure:
    i. Three convolutional layers with 16, 32, and 64 3x3 filters and Relu activations. Each output feature map should have the same dimension as the input to this layer. Each Conv layer is followed by a 2x2 Maxpooling layer with stride of 2.
    ii. A Fully Connected layers with 1024 neurons.
b. Before training the above model, test your model with the first image in the testing set, print out the predicted probability for each class, and find the cross entropy loss of the untrained model for this testing sample. Verify this loss is approximately log(C).

1.3 Model Training:

a. Use default hyperparameter values, train the model for 20 epochs. validate after each epoch.
b. Plot the training loss and validation loss curves in one figure, and training accuracy and validation accuracy curves in another figure.

1.4 Model Evaluation:

a. Use testing data set to evaluate the testing accuracy of the trained model.

1.5 Prediction:

a. Generate the class label for the first image in the testing set using the trained model. Display the predicted probability for each class, and calculate the cross-entropy loss of the trained model for this testing sample. Compare the results with those of the untrained model in 1.2.b.

Part 2: Additional functionalities.

2.1 Accuracy analysis

Generate a confusion table for the model testing. The rows of the table is the ground truth of labels, while the column is the model output labels. So the N_{ij} in the table indicate the number of testing images with ground truth label i, but being classified as label j by the trained model.

2.2 Model visualization

a. Display the 16 trained filters in the first conv layer as images.
b. Display the 16 feature maps as images from the first conv layer generated for the first testing image.

2.3 Overfitting observation

a. Use a small set of training images (100) to run the same implementation as in Part 1. Plot the training and validation loss curves in a single figure. Find out the number of epochs it takes to overfitting.
b. Add drop out layers as regularization method to reduce overfitting.

Part 3: Model and training fine-tuning

3.1 Modify the model architecture parameters, including
   a. Number of different layers
   b. Number of filters/outputs in each layer

3.2 Modify the training hyperparameters including (but not limited to)
   a. Learning rate
   b. Batch size
   c. Number of epochs

Document the training/testing performances for all your experiments.

Part 4: Transfer Learning

4.1 Save your best performed model in Part 3. Then reload the saved module and continue to train for 10 epochs. Show both your training accuracy and validation accuracy curves

4.2 Save your newly trained model in 4.1, then reload it. Fix the weights in all the layers before the classification layer(s), and continue to train the classification layer(s) only for another 10 epochs, show both your training and validation curves.

4.3 Use the same base model (without the classification layers) in 4.2, and add a new classifier layer, and train the new classifier layer from scratch. Show both your training and validation curves.

4.4 Compare the testing accuracy for the three models above.