Computer Vision Fundamentals - Assignment 5

Due: Mon 30 July 2018 at 7:00 pm (NO LATE SUBMISSIONS) Grade Scale: 100 points

THERE ARE NO LATE DAYS FOR THIS ASSIGNMENT

This assignment is to be implemented using MATLAB. It is recommended that you use the MATCONVNET library, which has a lot of useful functions built-in. However, you can also implement it using Torch, Caffe or TensorFlow. Your deliverable should be in the form of a single report document in which you describe what you did in each problem and why. Include all your relevant code segments as well as your explanations and results figures.

Zip all the code for each problem, named by the problem number, and upload them along with your report.

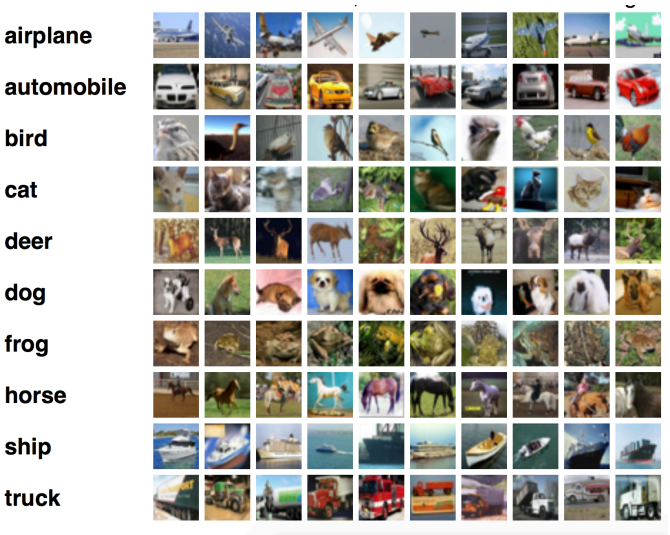
You can gain up to 20% additional points for implementing extra credit suggestions.

--

This assignment is adapted from Assignment 2 for Rob Fergus Computer Vision Course, available at http://cs.nyu.edu/~fergus/teaching/vision/assign2.pdf. The text below is mostly taken from this webpage with acknowledgement.

The goal of this assignment is to introduce you to Convolutional Neural Networks. We will explore two multi-class classification problems, initially using very simple single-layer fully connected neural network, and then building on towards multi-layered convolutional neural network for image classification. CNN approach is discussed in the lecture slides.

We will be using two 10-category datasets: hand-written digit recognition dataset by MNIST and scene-classification dataset by CIFAR-10. The first dataset is organized into 10 numeric digits categories – from 0 to 9 – with 60000 images for training and 10000 additional images for testing purpose. The second dataset is organized into 10 categories – Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship and Truck – and has 50000 training images and 10000 testing images.



Project material, including image dataset, starter code in MATLAB and MATCONVNET library is available at LMS. The documentation for MATCONVNET can be found at http://www.vlfeat.org/matconvnet/functions/ and a helpful tutorial is available at THIS link.

The starter code, starting from `main.m` contains more concrete guidance on the inputs, outputs, and suggested strategies for the 2 functions you will implement in this assignment:
`create_multiple_layer_nn_mnist.m, create_multiple_layer_nn_cifar10.m`.

The data for this assignment is available at this link: (Warning 807MB)
https://drive.google.com/file/d/0B6vylYfRx8fhQVd4bThUdVlFRms/view?usp=sharing

Problem 0. [10 points] Setup the starter code and the MATCONVNET Library. Run the starter code. This will compile the MATCONVNET library for your operating system, will load the dataset from disk and will create and train a single layer fully connected neural network.

Note that after the call of `@cnn_train`, MATCONVNET start training the network and displays the output on the MATLAB terminal. It also saves the weights at the end of each epoch in the destination/output folder, the path of which is provided as argument. If you re-run `main.m` the training process will resume from the saved state i.e. if you trained for 8 out of 10 total epochs in previous iterations, the network will load the weights from 8th epoch's saved state and will begin the 9th epoch. If you want to re-run from random initialization you have to delete the previously saved epoch states from destination/output directory.

(a): Add code to plot out the network weights as images (one for each output, of size 28 by 28) after the last epoch. Grab a screenshot of the figure and include it in your report.

(b): Give a breakdown of the parameters within the above model, and the overall number

Problem 1. [30 points] Train a Convolutional Neural Network on MNIST dataset

Start by creating a multi-layer CNN with the following architecture

Input → Convolution (5x5x20) → ReLU → Max Pooling (2x2, stride 2) → Convolution (5x5x50) → ReLU → Max Pooling (2x2, stride 2) → Fully Connected (size 500, with ReLU) → 10-category-perceptron-classifier

and train it on MNIST for 10 epochs (passes through the training data) using SOFTMAX LOSS. The default values for the learning rate, batch size and number of epochs are given at the top of the `create_multiple_layer_nn_mnist.m` file. Unless otherwise specified, you can use the default values throughout this assignment. Note the decrease in training loss and corresponding decrease in validation errors. Paste the output into your report.

(a): Add code to plot out the network weights as images (one for each output, of size 28 by 28) after the last epoch. Grab a screenshot of the figure and include it in your report.

(b): Reduce the number of training examples to just 5000. (Hint: You can copy the existing IMDB file and remove images from it. Do not change the testing data). Paste the output into your report and explain what is happening to the model.

(c): Experiment with different learning rates (for example from 1e0 to 1e-6) and note (and describe) the effect of different values of learning rates on training process in your report. (Attach screenshots of error and accuracy curves in your report)

(d): Give a breakdown of the weights within the above model, and the overall number of weights in your network.

(e): Compute the confusion matrix for your final trained network. Visualize a sample of the images on which the network is making errors and comment on them. By analyzing the confusion matrix, find out which digits are confused the most, and comment on why that is the case.

(f): [Extra Credit] Modify the above architecture and see how much you can improve the accuracy. 4 bonus points will be granted for each percentage point increase above 98%.

Problem 2. [60 points] Train a Convolutional Neural Network on CIFAR-10 dataset

Modify the network used in pervious question to take color images of size 32x32x3 as input and train it on CIFAR-10 dataset. (Hint: top1Err can never be above 1.00; if your top1Err is above 1.00 there is some bug in your implementation of network.)

(a): Add code to plot out the network weights as images after the last epoch. Grab a screenshot of the figure and include it in your report.

(b): Reduce the number of training examples to just 5000. (Hint: You can copy the existing IMDB file and remove images from it. Do not change the testing data). Paste the output into your report and explain what is happening to the model.

(c): Experiment with different learning rates (for example from 1e0 to 1e-6) and note (and describe) the effect of different values of learning rates on training process in your report. (Attach screenshots of error and accuracy curves in your report)

(d): Give a breakdown of the weights within the above model, and the overall number of weights in your network.

(e): Compute the confusion matrix for your final trained network. Visualize a sample of the images on which the network is making errors and comment on them. By analyzing the confusion matrix, find out which categories are confused the most, and comment on why that is the case.

(f): Check if the output of network changes by translating and rotating the input image. (Get an image, apply some translation/rotation to the image: fill the missing values with zeros, and get prediction on new image. Then compare the predictions of untransformed image vs transformed image)

(g): Modify your network to add additional layers as follows:
Input → Convolution (5x5x20) → ReLU → Convolution (5x5x20) → ReLU → Max Pooling (2x2, stride 2) → Convolution (5x5x50) → ReLU → Max Pooling (2x2, stride 2) → Fully Connected (size 500, with ReLU) → 10-category-perceptron-classifier

Compare the performance of this classifier with the first one. Repeats steps (c), (d) and (e) for this network and include in your report.

(f): [Extra Credit] Modify the above architecture and see if you can improve the accuracy. 1 bonus point will be granted for each percentage point increase in accuracy above 70%.

Problem 3. [Up to 10 bonus points] Train a Convolutional Neural Network using Caffe/TensorFlow/Torch

Although we suggest to implement the assignment in Matlab using MATCONVNET but if you implement it using any of the listed toolboxes, you can earn up to 10 bonus points. You have to plot graphs yourself and attach them in your report along with the code snippets.