CS1674: Homework 9

Due: 4/15/2021, 11:59pm

This assignment is worth 50 points.

In this assignment, you will train deep networks to perform categorization. The assignment has four parts. In the first part, you will train a neural network from scratch. In the second and third, you will transfer layers from a pretrained network, then append and train one fully-connected (FC) layer. In the last part, you will describe your findings.

You will use the same dataset as for HW7. However, because we want to make sure we use square images (i.e. 227x227), you will need to create a separate folder with the same eight folders (categories) as those in HW7, but inside each folder, copy only the images with "resized" in the filename, resulting in 1200 images total in the top-level folder `scenes_lazebnik` (you can also download a new copy of the data with just the resized images from Canvas).

You will also need to install the Matlab Deep Learning (DL) Toolbox add-on. Go to Home --> Add-ons to do that.

For each problem, write your code in a separate script titled `part_X.m` where X is `i, ii` or `iii`. Also submit a separate file `answers.txt` where you describe and compare the performance of your different networks. Briefly hypothesize why you observe these trends, based on what we have discussed in class.

You will need to rely on the Matlab documentation to learn how to use the built-in neural network functions. Any existing functions are fair game-- of course, do not look for scripts that accomplish the entirety of what an assignment part asks. However, for this assignment, the goal IS to learn how to use the documentation, hence please do look up functions and examples. Some useful links are below. Please skim through all of them to get a sense of how the DL toolbox works.

1. deep learning with images
2. train network
3. training options
4. transfer learning
5. classify
6. convolution layer
7. max pooling layer
8. fully connected layer

Some of the functions you will need to call are `imageDatastore`, `splitEachLabel` (to load and split the dataset; we will not use `load_split_dataset` from before), `trainingOptions`, `trainNetwork`, `classify`. Doing the assignment will be very easy if you skim through the references above.

Unless otherwise specified, use a learning rate of 0.001, a maximum of 1 epochs, 100 images per class for training, and the rest (50) for testing.

Training each network should take about 1-5 min depending on your CPU.

Part I (15 pts):

In this part, you will train a neural network for the task of classifying the eight scenes from HW7, from scratch.

1. You need to specify a folder for the train set and the test set. Refer to the "train network" link for details on how to set up the data.
2. Create a network with three groups of layers (denoted A, B, C in the following). First, use an image input layer; this "layer" takes in the images at size 227x227x3. Then, construct the following layer group A: a convolution layer with 50 11x11 layers (same size as the filters in the first layer of AlexNet) with stride 4, followed by RELU and a max pooling layer of size 3x3 and stride 1. Then, layer group B: a convolution layer with 60 5x5 filters, RELU and max pooling of size 3x3 and stride 2. Then, layer group C: a fully-connected layer with size 8 (for 8 classes), followed by a softmax layer (which computes probabilities) and a classification layer. Check the links above for the corresponding functions and their inputs format.
3. You need to specify options for training the network. Use the "training options" link above. Specify the max number of epochs, the learning rate, and set the 'Plots' variable such that it shows training progress.
4. For simplicity, we will **not** use a validation set. Train the network and output performance on the test set after the last iteration. You can use the `classify` function and the `imdsTest.Labels` variable to get the ground-truth labels on the test set.
5. In your answers file, hypothesize why you see such high/low performance. Keep in mind what performance was in HW7.

Part II (15 pts):

In this part, you will transfer layers from an AlexNet network trained on the ImageNet dataset. Refer to the "transfer learning" link. Transfer all layers up to but excluding the FC6 layer (i.e., transfer the layers up to the one just before `'fc6'`). It will be helpful to see what layers you are transfering; try the net.Layers to get a list of the layers in AlexNet. Append a single fully-connected layer (of size 8), followed by softmax and classification. Then train and evaluate performance, and describe your observations.

Part III (10 pts):

In this part, you will also transfer layers, but additionally transfer FC6 and FC7, all the way up to (but excluding) FC8 (i.e., transfer the layers up to the one just before `'fc8'`). Now append a single fully-connected layer, as before. Train the network, evaluate performance on the test set, and describe your observations in the answers file.

Part IV (10 pts):

In a file `answers.txt`, list all accuracies for each setting described above. Then hypothesize the reasons for the relative performance of each method. For the first part, hypothesize reasons for the network's performance relative to the performance of the SVM classifier we developed in HW7. For later parts, hypothesize reasons for the performance of the network in that part, compared to the network in the previous part. Aim for about 3-5 sentences.

Extra experiments (No pts, just for fun):

This part is purely for just playing around with the network training. Don't submit this. It won't be graded. You also can simply ignore this section; I just wanted to use this HW as an opportunity to get some hands on network training experience beyond the above simple cases.

You may have noticed that the above training procedures (learning reate of 0.001, max epoch of 1) can probably be improved. In fact, they can be improved by better training procedures and hyperparameters. For your own interest, considering changing hyperparameters for your training (can use different hyperparamters for each Part):

- Max epoch: increase this to train for longer
- Initial learning rate: Instead of 0.001, conservatively trying smaller learning rates (e.g., 0.0001) often helps

Just by changing those two training hyperparameters, I was able to achieve ~42% accuracy for Part i, ~80% accuracy for Part ii, and ~92% accuracy for Part iii. This shows the importance of training for deep learning models: same architecture can have vastly different results. Note that there will be some randomness as you repeat the training (train/test split is random), so a small amount of increase or decrease (e.g., 1%) may be from this randomness. I suggest running it a few times to see if you get consistent improvements. Of course, there are other training options. Feel free to explore around. May be, see if you can beat my results.

Submission:

- `part_i.m`
- `part_ii.m`
- `part_iii.m`
- `answers.txt`

Acknowledgement: Adriana Kovashka.