CS364/AM792 : Computer Vision 15 October 2019

Assignment 6: Deep Learning

Please prepare a single, condensed and neatly edited document for submission. For each problem include a short description of what you did, results, and a brief discussion/interpretation. It is not necessary to include any code in your document, unless a snippet helps to explain your method. Upload a zip-file containing your document and code to SUNLearn (under "Assignment 6") **before 17:00 on 29 October**. Also hand in a printed copy of the document at my office, before 17:00 on 30 October.

You are free to use any programming language. I recommend Matlab or Python. Collaboration is restricted to the exchange of a few ideas, and no form of plagiarism will be tolerated. All code, results, and write-up that you submit must be your own work.

---

1. We return to the dataset of Assignment 5 problem 2, and will now attempt to train convolutional neural networks to solve the 9-class scene classification problem. We begin by training a CNN **from scratch**.

(a) The network will ask for images of a fixed resolution, so first resize[1] all the images to be 227×227. Use a deep learning package[2] to set up a network with three convolutional stages (each consisting of a convolution layer, batch normalization, ReLU, then max pooling), followed by a fully connected layer with 9 outputs, and finally a softmax layer. Have a look at tried-and-tested network architectures for inspiration on hyperparameter choices for each layer. Pick a learning rate and batch size[3], and train the model using stochastic gradient descent with momentum, for about 20 epochs. You should get a test accuracy in the region of 60% (which is somewhat disappointing).

(b) The training set is very small, and the network quickly overfits. Here are two possible remedies:

Augment the training set. If we left-right flip (mirror) an image in this dataset, its class doesn't change. We can do this for all the images in the training set, to effectively double the number of training samples. Even though the samples in this new training set are not truly independent, this approach can lessen the risk of overfitting quite a bit.

Introduce regularization. An effective form of regularization in neural networks is called dropout. Include in your report a short description of what dropout does and why it can be useful to prevent overfitting (cite all your sources).

Implement both of these remedies[4] and train the network again, from scratch. It takes a bit longer, but you should now reach a test accuracy of about 70% to 75%.

---

2. Instead of training a CNN from scratch, let us attempt to **fine-tune** an existing network that was trained on a much larger set of images for a different task. We will use AlexNet for this purpose; a deep CNN trained on ImageNet to perform 1,000-class image classification, and can be found in (or downloaded for) most deep learning packages. *Note: if you cannot find a straightforward implementation of AlexNet, feel free to use another network pre-trained on ImageNet, like one of the VGG variants.*

Load the pre-trained network into your environment[5], remove the last fully connected layer with 1,000 outputs, and replace it with a new fully connected layer having 9 outputs. Use the pre-trained weights, together with random weights for the new layer, as initialization for training. The hope is that those pre-trained weights are a much better initialization than the random weights used when training from scratch. With similar training parameters (learning rate, batch size, and number of epochs), and without any training set augmentation, you should now get a test accuracy close to 95%. Not bad at all!

Hand in: 29 October 2019

[1] use an existing function for resizing; you can choose any resolution, but 227×227 happens to be required in problem 2

[2] try the Deep Learning Toolbox in Matlab, or Keras for Python; with this small training set, GPU-acceleration is not necessary

[3] as a start you may consider a learning rate of 0.001 and batch size of 64, but feel free to experiment with these values

[4] as a start you may try to add a dropout layer with $p = 0.5$ immediately before the final convolutional layer

[5] AlexNet requires 227×227 *colour* images as input, so first convert all the greyscale images to equivalent colour images