MIT CSAIL

6.8300/1 Advances in Computer Vision

Spring 2024

Problem Set 3

Posted: Thursday, Feb 29, 2024 **Due:** Tuesday 23:59, March 12, 2024

We provide a Python notebook with the code to be completed. You can run it locally or on Google Colab. To use Colab, upload it to Google Drive and double-click the notebook (or right-click and select Open with Google Colaboratory), which will allow you to complete the problems without setting up your own environment. Once you have finished, make sure all the cells are run before downloading the notebook, and also copy the code sections that you have completed into your report (e.g., as screenshots or in some other legible manner).

Submission Instructions: Please submit two files on Gradescope. (1) your pdf report and (2) your Python notebook with all cells run. Different submission portals this time.

Attention: **We can't give credit when code is missing from `kerb.pdf`.** Maybe set a reminder for yourself to check that you have answered all parts of all questions, with all code included.

Late Submission Policy: If your Problem Set is submitted within 7 days (rounding up) of the original deadline, you will receive partial credit. Such submissions will be penalized by a multiplicative coefficient that linearly decreases from 1 to 0.5, stepwise by day.

Problems 1, 2, and 4 assume no knowledge of backpropagation. Problem 3 is about backpropagation, which will be covered in class after this assignment is released.

In problems 2 and 4, you will be experimenting with neural networks using PyTorch. To speed up the execution of the experiments, we recommend using GPU acceleration. Colab comes with free GPU support. On Colab, select GPU as your runtime type as follows:
Runtime → **Change runtime type** → **Hardware accelerator** → **GPU** → **Save**.

Credit: Parts of this Problem Set are inspired by [1], [2], and [3].

Problem 1 *Forward Propagation* (6.8301: 10 points, 6.8300: 20 points)

(a) (Convolutional layer) The convolutional layer is the most popular module in computer vision tasks. Assume your input x_{in} and output x_{out} are both 1D signals of size N, and your kernel W has size k, where k is an odd number. Assume the stride size is 1. Find the equation for the forward propagation. You can omit the bias term of the convolutional layer.

(b) Imagine a convolutional filter which detects "cat." In image A, there is one cat in the bottom left; in image B, many cats. Which model layer would distinguish between these images better, global max pooling or a global average pooling? Write "global max pooling" or "global average pooling."

(c) (Pooling layer) [**6.8300 Only**] The pooling layer is a popular layer without trainable parameters. In this question, consider a max pooling operator. Assume your input x_{in} and output x_{out} are both 1D signals of size N and your kernel has size k, where k is an odd number. Assume the stride size is 1. Find the equation for the forward propagation.

Problem 2 *Neural Network Inference* (10 points)

In this section, we will ask a neural network to classify an image into one of the 1000 object classes from the ImageNet dataset.

(a) In the provided Python notebook, load the randomly initialized network and answer the following question: How many features are in the input of the last layer?

(b) Run the Corgi image through the network. Include the generated plot in your report and answer the following question: What are the top-5 predictions?

(c) Reload the network with pre-trained weights. These weights originate from a network trained on the ImageNet dataset. Run the Corgi image through the network. Then complete `TODO1` and rerun the network. Include the two generated plots in your report (one displays the top-5 predicted logits, the other displays the top-5 predicted probabilities), and answer the following question: What are the top-5 predictions now? Please also include the code you completed in your report (i.e., the code for function `output2prob`).

Problem 3 *Backpropagation* (6.8301: 30 points, 6.8300: 40 points)

(a) (Convolutional layer) As in Problem 1a, assume your input x_{in} and output x_{out} are both 1D signals of size N, and your kernel W has size k, where k is an odd number. Assume the stride size is 1. Find the equation for the backward propagation. You can omit the bias term of the convolutional layer.

(b) (Convolutional layer) Consider the backpropagation process with learning rate η. Assume you are given the gradients from the last layer $\frac{\partial C}{\partial x_{out}}$. Find the gradients of the input $\frac{\partial C}{\partial x_{in}}$ (your answer should be of the form $\frac{\partial C}{\partial x_{out}} * \ldots$), and find the update rule for the kernel weights W^{i+1}.

(c) (Convolutional layer) Discuss how you handle the boundaries (i.e., the padding) and explain your choice. Note your choice must justify both x_{in} and x_{out} having the same size.

(d) (Pooling layer) [**6.8300 only**] Now consider the max pooling operator. As in Problem 1b, assume your input x_{in} and output x_{out} are both 1D signals of size N and your kernel has size k, where k is an odd number. Assume the stride size is 1. Find the equation for the backward propagation.

(e) (Pooling layer) [**6.8300 only**] Discuss how you handle the boundaries (i.e., the padding) and explain your choice. Note your choice must justify both x_{in} and x_{out} having the same size.

Problem 4 *DeepDream and Adversarial Attacks* (6.8301: 30 points, 6.8300: 40 points)

When training neural networks, we optimize the model parameters to maximize a certain objective. In some cases, we may be interested in optimizing the input of the network for a fixed set of parameters. In this section we will study how this technique can be used to generate images that can help us interpret the network, or fool it by making imperceptible changes to the input.

Neural networks are generally differentiable with respect to their input. Therefore, we can compute the gradient of the objective with respect to the loss and use it to update the input through gradient descent.

We will start by using this technique to obtain the input that maximizes the activations of the pre-trained neural network.

(a) In this part, you will modify the input image to maximize the log-probability of the class Tarantula (id 76) by computing the gradient of the log-probability of Tarantula with respect to the input image. Complete `TODO2` in the provided Python notebook. In your report, include the original image, attacked image, and the plot displaying the top-5 predicted probabilities. Please also include your code in your report.

(b) Now maximize the log-probability of the class Tiger Cat (id 282) instead. In your report, include the original image, attacked image, and the plot displaying the top-5 predicted probabilities.

(c) Can you tell a visual difference between the attacked images and the original image? Briefly discuss why such attacks could cause security problems.

(d) [**6.8300 only**] In (a) and (b), adversarial examples are generated by maximizing the log-probability of different classes. Let's try a different objective here. Typically, we refer to the output from an intermediate layer as an "embedding" or "feature map" of the network. Each layer of the network generates a different embedding of the input image. Instead of maximizing one single probability, instead modify a random image to minimize the $\ell 2$-distance between its feature maps and the feature maps of the Corgi image. Try this with three different layers. To do all this, complete `TODO3` and `TODO4` in the notebook. Include the original and modified images for the three layers in your report (in total there should be six images), include the code for `TODO4` in your report, and answer the following question: Do the resulting images look similar to the Corgi image?

(e) To avoid the adversarial examples, we need networks that are trained robustly to this kind of noise. In this problem, we will load the robustly trained network and repeat the previous experiment in question (b). Run the relevant code in the notebook (you don't have to write code for this problem). In your report, include the original image, attacked image, and the plot displaying the top-5 predicted probabilities for the class Tiger Cat (id 282) and two additional classes of your choice (by changing the class id in the code), and answer the following question: Do the attacked images now look noticeably different from the original image?

(f) **[6.8300 only]** Repeat problem 4d with the robust neural network by completing `TODO5` and `TODO6` in the notebook and running the relevant code for three different layers. Include the original and modified images for the three layers in your report (in total there should be six images), include the code for `TODO6` in your report, and answer the following question: Do the resulting images now look more similar to the Corgi image?

References

[1] S. Santurkar, D. Tsipras, B. Tran, A. Ilyas, L. Engstrom, and A. Madry, "Image synthesis with a single (robust) classifier," *Advances in neural information processing systems*, 2019.

[2] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry, "Adversarial robustness as a prior for learned representations," *arXiv preprint arXiv:1906.00945*, 2019.

[3] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. https://distill.pub/2017/feature-visualization.