MIT CSAIL

6.819/869 Advances in Computer Vision

Spring 2022

Problem Set 4

Posted: Tuesday, March 8, 2022 **Due:** Tuesday 23:59, March 15, 2022, ET

6.869 and 6.819 students are expected to finish all problems unless there is an additional instruction.We provide a python notebook with the code to be completed. You can run it locally or in Colab (upload it to Google Drive and select 'open in colab') to avoid setting up your own environment. Once you finish, run the cells and download the notebook to be submitted.

Please submit a .zip file named⟨yourkerberos⟩.zip containing 1) report named report.pdf including your answers to all required questions with images and/or plots showingyour results, and 2) the python notebook provided, with the cells run and the relevant source-code. If you include other source code files for a given exercise, please indicate it in the report.

Late Submission Policy: If your pset is submitted within 7 days (rounding up) ofthe original deadline, you will receive partial credit. Such submissions will be penalized by amultiplicative coefficient that linearly decreases from 1 to 0.5.

Problems 1, 2, and 4 assume no knowledge of backpropagation. Problem 3 is about backpropagation, which will be covered in the lecture on March 8.

In problems 2 and 4, you will be experimenting with neural networks using PyTorch. In order to run the experiments fast, you will need access to a GPU. We recommend using Colab to run the experiments, since it comes with GPU support. To enable it simply do:

Runtime - Change Runtime type - Hardware accelerator - GPU.

For those questions **include the generated images and relevant code in this report**.

Credit: part of this PSET is inspired by [1], [2], and [3].

Problem 1 *Forward Propagation (2pts)*

(a) (Convolution layer) Convolution layer is the most popular module in computer vision tasks. Consider your input x_{in} and output x_{out} are both 1-D signals with the same dimension N, and your kernel W has size k. Find the equation for its forward propagation through a convolution layer.

(b) (Pooling layer) **This question is required for 6.869 and optional for 6.819.** Pooling layer is a popular layer without trainable parameters. In this question, the pooling is a max pooling operator with stride 1. Consider your input x_{in} and output x_{out} are 1-D signals with the different size, find the equations or pseudo code for its forward propagation.

Problem 2 *Neural Network Inference (1pt)*

In this section, we will test a neural network toclassify an image between 1000 classes, defined in the Imagenet dataset. You can use the code to view which are those classes.

(a) Load the randomly initialized network. How many features are in the input of the last layer?

(b) Run the Corgi image through the network. What are the top predictions?

(c) Reload the network with pre-trained weights. Those weights correspond to training the network with the Imagenet dataset. What are the top predictions? What is the probability of the top prediction?

Problem 3 *Backpropagation (4pts)*

(a) (Convolution layer) Similar to in Problem 1(a), consider your input x_{in} and output x_{out} are both 1-D signals with the same dimension N, and your kernel W has size k. Find the equation for backward propagation.

(b) (Convolution layer) Consider the backpropagation process, with learning rate η, and the gradients from the last layer is $\frac{\partial C}{\partial x_{out}}$. Find the gradients of the input $\frac{\partial C}{\partial x_{in}}$, and the update rule for the kernel weights W^{i+1}.

(c) (Convolution layer) Discuss how you handle the boundaries and explain your choice.

(d) (Pooling layer) **This question is required for 6.869 and optional for 6.819.** Similar to in Problem 1(b), find the equations or pseudo code for backward propagation of the given input and output through a max pooling layer with stride 1.

(e) (Pooling layer) Discuss how you handle the boundaries and explain your choice.

Problem 4 *DeepDream and Adversarial Attacks (4pts)*

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) Modify the input image to maximize the log-probability of the class Tarantula (id 76). To do that, compute the gradient of the log-probability of Tarantula with respect to the input image[1]. Copy in the report the resulting image and the new predicted probabilities.

(b) Follow question (a) but instead maximize the log-probability of the class Tiger Cat (id 282). Copy in the report the resulting image and the new predicted probabilities.

(c) Explain why the perturbed image from question (b) looks more like the original input image compared to the perturbed image from question (a).

(d) **This question is required for 6.869 and optional for 6.819.** 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 call the output from an intermediate layers as the embedding or feature map of the network. For instance, each layer of network generates a different embedding of the input image. Instead of maximizing one single probability, modify a random image to minimize the $\ell 2$-distance between its feature and the feature of the Corgi image. Try this with 3 different layers. Do the resulted images look similar to the Corgi image? Include the modified image in the report.

(e) To avoid the adversarial examples, we need networks that are trained robustly to this kind of noise. Madry's group[2] at MIT has been working on robust neural networks and study their properties. Load the robustly trained network and repeat the previous experiment in question (b). What are the top predicted classes? Include the modified image in the report. How does it look now?

(f) **Bonus**. Try modifying the image into other classes. Include more examples of perturbations.

(g) **This question is required for 6.869 and optional for 6.819.** Repeat problem (4.d) with the robust neural network. Do the resulted images look more similar to the Corgi image this time? Include the modified image in the report.

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.

[1] You can use `torch.nn.functional.logsoftmax(output,1)[0, i]` to compute the log-probability of class i, where output is the output of the network (logits)

[2] `https://people.csail.mit.edu/madry/`