Deep Learning / Fall 2022

Homework 2

Please upload your assignments on or before 5pm ET on Friday October 14, 2022.

- You are encouraged to discuss ideas with each other. But you **must acknowledge** your collaborator, and you **must compose your own** writeup and code independently.
- We **require** answers to theory questions to be written in LaTeX. (Figures can be hand-drawn, but any text or equations must be typeset.) Handwritten homework submissions will not be graded.
- We **require** answers to coding questions in the form of a Jupyter notebook. It is **important** to include brief, coherent explanations of both your code and your results to show us your understanding. Use the text block feature of Jupyter notebooks to include explanations.
- Upload both your theory and coding answers in the form of a **single PDF** on Gradescope.

---

1. **(3 points)** *Designing convolution filters by hand.* Consider an input 2D image and a 3×3 filter (say w) applied to this image. The goal is to guess good filters which implement each of the following elementary image processing operations.
   a. Write down the weights of w which acts as a *blurring* filter, i.e., the output is a blurry form in the input.
   b. Write down the weights of w which acts as a *sharpening* filter in the horizontal direction.
   c. Write down the weights of w which acts as a *sharpening* filter in the vertical direction.
   d. Write down the weights of w which act as a *sharpening* filter in a *diagonal* (bottom-left to top-right) direction.
2. **(3 points)** *Weight decay.* The use of ℓ_2 regularization for training multi-layer neural networks has a special name: *weight decay*. Assume an arbitrary dataset $\{(x_i, y_i)\}_{i=1}^n$ and a loss function $\mathcal{L}(w)$ where w is a vector that represents all the trainable weights (and biases).
   a. Write down the ℓ_2 regularized loss, using a weighting parameter λ for the regularizer.
   b. Derive the gradient descent update rules for this loss.
   c. Conclude that in each update, the weights are "shrunk" or "decayed" by a multiplicative factor before applying the descent update.
   d. What does increasing λ achieve algorithmically, and how should the learning rate be chosen to make the updates stable?
3. **(2 points)** *The IoU metric.* In class we defined the IoU metric (or the Jaccard similarity index) for comparing bounding boxes.
   a. Using elementary properties of sets, argue that the IoU metric between any two pair of bounding boxes is always a non-negative real number in $[0, 1]$.
   b. If we represent each bounding box as a function of the top-left and bottom-right coordinates (assume all coordinates are real numbers) then argue that the IoU metric is *non-differentiable* and hence cannot be directly optimized by gradient descent.
4. **(4 points)** *Training AlexNet.* Open the (incomplete) Jupyter notebook in Google Colab (or other Python IDE of your choice) and complete the missing items.

5. **(3 points)** *Object detection.* We will be finetuning a pretrained torchvision object detection model in this problem. It will be helpful to go through the excellent tutorial (with example code) here. You can reuse whatever code you like from this tutorial with *proper attribution*!

   a. The tutorial shows how to finetune a pretrained model (what they call option 1). Now, follow their approach to add a different backbone (option 2). You can use the same dataset (and code) as the tutorial.

   b. How do the performance of the two models compare on the training data after 10 epochs?

   c. Test both models on this image of the Beatles. Be careful to make sure the dimensions align and the scaling is similar. How do the two models compare in terms of performance on this image?