CSCI 145 Problem Set 7

October 7, 2025

Submission Instructions

Please upload *your* work by **11:59pm Monday October 13, 2025.**

- You are encouraged to discuss ideas and work with your classmates. However, you **must acknowledge** your collaborators at the top of each solution on which you collaborated with others and you **must write** your solutions independently.
- Your solutions to theory questions must be written legibly, or typeset in LaTeX or markdown. If you would like to use LaTeX, you can import the source of this document (available from the course webpage) to Overleaf.
- I recommend that you write your solutions to coding questions in a Jupyter notebook using Google Colab.
- You should submit your solutions as a **single PDF** via the assignment on Gradescope.

Grading: The point of the problem set is for *you* to learn. To this end, I hope to disincentivize the use of LLMs by **not** grading your work for correctness. Instead, you will grade your own work by comparing it to my solutions. This self-grade is due the Friday *after* the problem set is due, also on Gradescope.

Problem 1: CNNs and CAM

In this problem, we will explore convolutional neural networks (CNNs) and an interpetability tool called class activation maps (CAM). This tool from class may be helpful for visualizations.

Part A: Handmade Kernels

Write down one 3×3 kernel for blurring, one 3×3 kernel for detecting *horizontal* edges, and one 3×3 kernel for detecting *diagonal* edges. Confirm your kernels work as intended on example $2D$ matrices.

Part B: Pretrained CNN

Using `torch`, load a ResNet50 architecture (example code here). Preprocess an animal photo of your choice, pass it to the model, and inspect the predictions made by the model. In particular, print the five largest probabilities (apply softmax to the logits), and the corresponding human-readable labels.

Part C: Explaining Predictions via CAM

Before the final fully connected layer, we have an activation layer $\mathbf{A}$ with dimension [batch size, number of channels, height, and width]. We apply the an *adaptive* average pooling layer so that the height and width dimensions are compressed and the dimension becomes [batch size, number of channels, 1]. Finally, we apply a fully connected layer $\mathbf{W}$ with dimension number of classes by batch size to map to [batch size, number of classes].

The idea of a class activation map (CAM) is to use the activation $\mathbf{A}$ weighted by the *weights* of the class the model predicts i.e., $\mathbf{W}[i^*]$ where i^* is the index of the class with the largest logit. On your input image, extract $\mathbf{A}$ (there are several clever ways to do this), multiply each channel by the appropriate weight in $\mathbf{W}[i^*]$, and sum over channels. The result should be [batch size, height by width]. Upscale this matrix using `torch`'s `interpolate` function, and plot the result on top of the input image. What do you notice?