CS231A: Computer Vision, From 3D Perception to 3D Reconstruction and beyond Homework #3

(Spring 2024) Due: **Friday, May 21**

Overview

This PSET will involve concepts from lectures 8, 10, 11, 12, and 13. It will involve 3D reconstruction, representation learning, and supervised/unsupervised monocular depth estimation. Although there are 4 problems so this PSET may look daunting, most of the problems require comparatively little work. Unlike prior PSETs, you have to use Colab for problems 2-4.

Submitting

Please put together a PDF with your answers for each problem, and submit it to the appropriate assignment on Gradescope. We recommend you to add these answers to the latex template files on our website, but you can also create a PDF in any other way you prefer.

The assignment will require you to complete the provided python files and create a PDF for written answers. The instructions are **bolded** for parts of the problem set you should respond to with written answers. To create your PDF for the written answers, we recommend you add your answers to the latex template files on our website, but you can also create a PDF in any other way you prefer. To implement your code, make sure you modify the provided ".py" files in the code folder.

For the written report, in the case of problems that just involve implementing code, you will only need to include the final output (where it is requested) and in some cases a brief description if requested in the problem. There will be an additional coding assignment on Gradescope that has an autograder that is there to help you double check your code. Make sure you use the provided ".py" files to write your Python code. Submit to both the PDF and code assignment, as we will be grading the PDF submissions and using the coding assignment to check your code if needed.

For submitting to the autograder, just create a zip file containing all the files in the 'code' folder. Makes sure to add your finished code to the '.py' files after your code works in colab.

1 Space Carving (45 points)

In this question, we will examine properties of projective transformations. We define a camera coordinate system, which is only rotated and translated from a world coordinate system (with a rigid transform).

You may optionally use Google Colab or the included Jupyter notebook to complete this part (just copy the code to p1.py at the end when uploading). If using Colab, you will nede to upload the homework folder to Google Drive, follow these steps to make sure you have the ability to work on this problem:

a. Click the wheel in the top right corner and select Settings.

b. Click on the Manage Apps tab.

c. At the top, select Connect more apps which should bring up a GSuite Marketplace window.

d. Search for Colab then click Add.

(a) (**Programming Question**) The first step in space carving is to generate the initial voxel grid that we will carve into. Complete the function `form_initial_voxels()`. [**5 points**]

(b) Now, the key step is to implement the carving for one camera. To carve, we need the camera frame and the silhouette associated with that camera. Then, we carve the silhouette from our voxel grid. Implement this carving process in `carve()`. **Briefly describe how you went about implementing this, and include the resulting image in your report. [10 code + 5 written points]**

(c) The last step in the pipeline is to carve out multiple views. **Submit the final output after all carvings have been completed, using `num_voxels` $\approx 6,000,000$. What are some flaws you notice in the reconstruction? [5 points]**

(d) Notice that the reconstruction is not really that exceptional. This is because a lot of space is wasted when we set the initial bounds of where we carve. Currently, we initialize the bounds of the voxel grid to be the locations of the cameras. However, we can do better than this by completing a quick carve on a much lower resolution voxel grid (we use `num_voxels` = 4000) to estimate how big the object is and retrieve tighter bounds. Complete the method `get_voxel_bounds()` and change the variable `estimate_better_bounds` in the `main()` function to `True`. **Include your new carving in the report, which should be more detailed. Are there any remaining issues with the reconstruction? Why do you think they are still there? [8 points code + 2 points written].**

(e) Finally, let's do a little experiment. Notice that in the first three steps, we used perfect silhouettes to carve our object. Look at the `estimate_silhouette()` function implemented and its output. Notice that this simple method does not produce a really accurate silhouette. However, when we run space carving on it, the result still looks decent!

   (i) **Why is this the case? [2 points]**

   (ii) **What happens if you reduce the number of views? [1 points]**

   (iii) **What if the estimated silhouettes weren't conservative, meaning that one or a few views had parts of the object missing? [2 points]**

2 Representation Learning (20 points)

In this problem, you'll implement a method for image representation learning for clasifying clothing items from the Fashion MNIST dataset.

Like in Q1, if you used colab, upload "p2/RepresentationLearning.ipynb" and the contents inside 'p3/code' to a location of your choosing on Drive. Then, navigate to this folder and open the file "RepresentationLearning.ipynb" with Colaboratory. This problem will be much easier to do in colab or as a jupyter notebook, so the entire problem is contained there, including the rest of the instructions. Note that there is no autograder for this problem, as you should be able to confirm whether your implementation works from the **images and plots**. Include the following in your writeup:

(a) Once you finish the section "Fashion MNIST Data Preparation", **include the 3 by 3 grid visualization of the Fashion MNIST data [3 point]**.

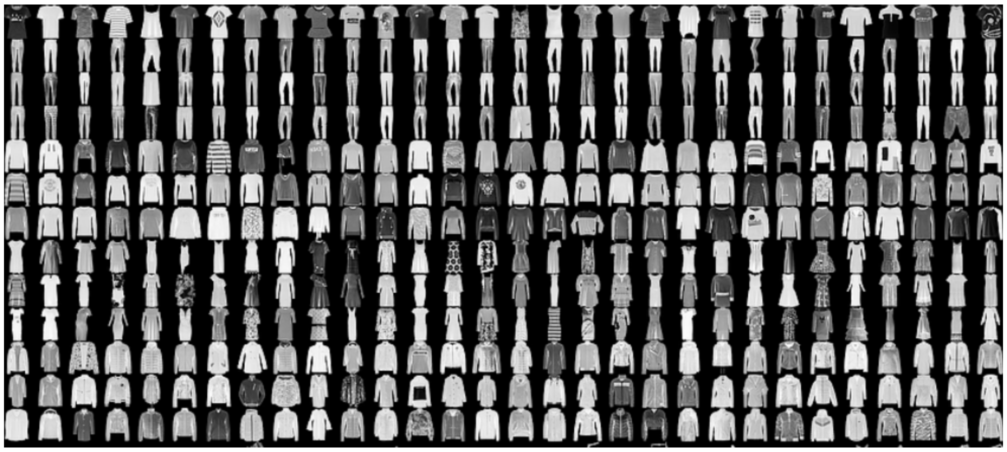

Figure 1: The Fashion MNIST dataset

(b) Once you finish the section "Training for Fashion MNIST Class Prediction", **include the two graphs of training progress over 10 epochs, as well as the test errors** [**5 points**].

(c) Once you finish the section "Representation Learning via Rotation Classification", **include the 3 by 3 grid visualization and training plot, as well as the test error** [**7 points**].

(d) Once you finish the section "Fine-Tuning for Fashion MNIST classification", **include all 3 sets of graphs from this section, as well as the test errors. Why do you think learning to un-rotate images works as a method of representation learning?** [**5 points**].

Fun fact: the method you just implemented if from the ICLR2018 paper "Unsupervised Representation Learning by Predicting Image Rotations".

3 Supervised Monocular Depth Estimation (15 points)

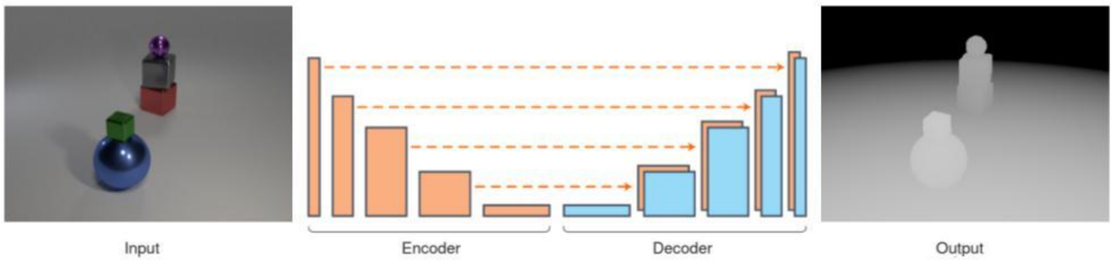

Figure 2: The task this problem attempts to

Now that you've had some experience with representation learning on a small dataset, we'll move on to the more complex task of training a larger model on the task of monocular depth estimation.

We will be implementing the approach taken in "High Quality Monocular Depth Estimation via Transfer Learning", which showed that taking a big model pre-trained on classifying objects from the ImageNet dataset helps a lot with training that model to perform monocular depth estimation on the NYU Depth v2 dataset. We created a smaller and simpler dataset that is a variation on the CLEVR dataset for this problem that we call CLEVR-D, so we'll actually not be using the pre-trained features, but otherwise the approach is the same. **Please download the dataset zip file at this Google Drive link and put it in the same folder as your code files (it is large, about 1GB, so you can try directly copying it to your Google Drive folder if that is easier).**

As with the previous problem, you'll be working with Colaboratory, so once again begin by uploading the contents of 'p3/code' to a location of your choosing on Google Drive. Then, open the file "MonocularDepthEstimation.ipynb" with Colaboratory. The rest of the instructions are provided in that document. Include the following in your writeup:

(a) Once you finish the section "Checking out the data", **include the grid visualization of the CLEVR-D data [5 points)]**. Make sure to return the a DataLoader with a DepthDatasetMemory object instead of just indexing into data.

(b) Once you finish the section "Training the model", **include a screenshot of your train and test losses from Tensorboard as well as the final outputs of the network [10 points]**.