Homework #1: 3D Basics; Auto-encoder; Variational Auto-encoder [50 points]
Due Date: Wednesday, 19 April 2023

Homework policies

Collaboration in solving the problems is encouraged in this class—you have a lot to learn from your fellow students. However, in order to make grading the homeworks a meaningful way to measure your effort and your understanding of the material, we allow collaboration groups of at most three students. A single write-up is sufficient for a collaboration group.

Please be sure to respect the honor code in all assignments: the work you present must be your own, or obtained jointly with your team partner. Other than that, it is not permitted to give or receive help from others in doing the assignments. Please be sure to reference all sources and resources used in obtaining your solution, as you would when publishing a scientific paper.

Assignments in this class must be submitted in digital form through Gradescope. Solution write-ups must be properly typeset in LaTeX or Microsoft Word and the PDF uploaded to Gradescope. No hand-written assignments will be accepted, or scans/photos of such. For programming problems, in your write-up please include (i) the relevant code snippet, and (ii) a link where the full source code can be downloaded (preferably as a zip file, e.g., Google Drive).

It is very important in this course that every homework be turned in on time. We recognize that occasionally there are circumstances beyond your control that prevent an assignment from being completed on time. You will be allowed two classes of grace during the quarter. This means that you can either hand in two assignments each late by one class, or one assignment late by two classes. Any further assignments handed in late will be penalized by 20% for each class that they are late and by 100% after that, unless special arrangements have been made previously with the instructor or the CA due to extenuating circumstances (e.g., serious illness). However, no assignment may be handed in later than Wednesday, 7 June 2023.

Goals of This Assignment

This assignment is a short one-week warmup assignment to help you:

- earn basic skills for processing and visualizing 3D data;
- get started with training an auto-encoder using 3D point cloud shapes as input;
- adapt the auto-encoder to a variational version that can be used for 3D shape generation.

Instructions

This assignment contains three programming problems that require access to a machine with decent CPUs and one GPU to train the neural networks on big scale 3D data. Please use

Google CoLab (check https://www.youtube.com/watch?v=IZUz4pRYlus for instructions) or the Cloud Credits (check https://github.com/cs231n/gcloud for instructions; ask the TAs for coupons) to get access to a GPU. All necessary files for this assignment can be found in the folder `code`. Please follow the instructions in the `README.md`, provided in the `code` folder, to install all necessary dependencies for running the provided codes. Although this is a short warmup assignment, we recommend to start as early as possible and definitely not wait until the last days to begin, as the network training will take some time.

Problem 1. 3D Basics: Data Processing and Visualization [20 points]

Unlike 2D images, when it comes to representing 3D data, we have several available representations such as 3D voxel-grids, point clouds, meshes, implicit fields, etc.), each of them has its own merits and drawbacks. In the first problem, you will learn how to process and visualize 3D data that are represented either as point clouds or voxel-grids. In your final report, please also include your implementations in the `prob1.ipynb`.

(a) (4 points). For this question, we want you to load a ShapeNet [1] chair, represented using three different ways – the original 3D CAD model mesh (`chair_example.obj`), the sampled point cloud shape (`chair_example.ply`), and the processed voxelized shape (`chair_example.binvox`). You can find all these files in the `data` folder. Starting from the `prob1.ipynb`, use the provided code for loading the different 3D representations (see the `load_obj`, `load_ply`, and `load_binvox` functions in `utils.py`) and load the 3D data formats. In your final report, please show the data structure for each of three data files: what matrices are there? what is the shape for each matrix? what does each dimension of the matrix denote? what information does each matrix contain? For the volumetric data, what percentage of voxels are occupied?

(b) (4 points). We have provided code utilities to help you visualize each of the three data formats in the `prob1.ipynb`. You can also use external tools such as Meshlab (https://www.meshlab.net/) and Drububu (https://drububu.com/miscellaneous/voxelizer/) to visualize the data. Obviously, as the data is in 3D, you can rotate the shapes to view them from different angles. For this task, report one image showing the default view using the provided visualization code, a second image from a different view, and an image using a third-party tool (e.g. Meshlab) for the data visualization, for each of the three data formats.

(c) (4 points). Now, we will try to generate your own point cloud and voxelized shapes from the input 3D mesh `chair_example2.obj`. For sampling point cloud shapes, we use the utility function `trimesh.sample.sample_surface_even` provided by trimesh (https://github.com/mikedh/trimesh), which returns samples which are approximately evenly spaced. You can use any online shape voxelizer (e.g. Drububu) for generating your voxelized shapes in different resolutions. For this question, please submit two figures of the generated point cloud shapes with 1000 and 10000 points (note that since the trimesh function does not return exactly the specified numbers of points, generate

as close as you can); submit two figures for the volumetric shapes of resolutions 16^3 and 32^3 and write a few sentences to explain your findings

(d) (4 points). 3D data are more easily manipulatable than images. For example, we can translate the shapes in space, rotate them, or scale them to make them bigger or smaller. For this question, you need to implement two functions: the first should take as input a mesh and rotate it and the second should take as input a mesh and scale it. For this task you have to use the `chair_example.obj` and `chair_example.ply` and submit one figure that rotates the shape along the up-axis by 45 degree clock-wise if you view from the top and one figure that doubles the scales of the shape along the other two axes (not the up-axis); please also show a reference image without performing any operation and make sure that all the figures share the same viewing angle.

(e) (4 points). Given two shapes, we may be interested in computing their difference quantitatively in order to use it for several downstream applications, such as shape retrieval. For the case of point clouds and voxel-grids, this can be easily done. In particular, to measure the distance between two binary voxel-grids (i.e. 1 for occupied, 0 for empty), it suffices to compute what percentage of voxels has different values between the two. To measure the geometric distance between two point clouds, we can use the Chamfer distance, as introduced in the class lecture. Formally, given two point clouds $O_1 = \{p_1, p_2, \cdots, p_n \| p_i \in \mathbb{R}^3\}$ and $O_2 = \{q_1, q_2, \cdots, q_m \| q_i \in \mathbb{R}^3\}$, we define the Chamfer distance as

$$CD(O_1, O_2) := \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\min_{j=1}^{m}\left\|p_i - q_j\right\|_2 + \frac{1}{m}\sum_{j=1}^{m}\min_{i=1}^{n}\left\|p_i - q_j\right\|_2\right). \tag{1}$$

For this task, compute and report the distances between two shapes (`chair_example.obj` and `chair_example2.obj`), after you have converted them to voxel-grid and point clouds (please sample 2000 points per point cloud and use resolution 32^3 for the volumetric grids).

Problem 2. Auto-Encoders for 3D Shapes [14 points]

For the second problem, we will train an auto-encoder (AE) for reconstructing 3D shapes and use the point cloud representation, as it is easier to implement and faster to train. The data for training the auto-encoder can be downloaded from http://download.cs.stanford.edu/orion/cs348n/chair_dataset.zip.

(a) (10 points). Implement a vanilla PointNet [2] (without the two T-Nets) as the encoder and a simple Multilayer Perceptron (MLP) decoder, by adding the missing parts in the `model.py`. Once you have implemented PointNet, you can train your network using `train_ae.py`. By default, this script outputs the training and validation curves, as well as the reconstruction performance evaluations using the Chamfer-distance metric. Finally, it also outputs result visualizations over a batch of validation data. You need to check the curves and the dumped result visualization for tuning your network. For this problem,

please provide the code you wrote (`model.py` and any other files you edited) and also visualize the training and validation curves, as well as the reconstruction performance numbers on both data splits. Notice that there is no single answer to this question and you will get a full grade if the submitted code, the curves, and the CD distances look reasonable.

(b) (4 points). In this problem, we want to test how well the network is actually doing w.r.t. reconstructing the 3D point cloud shapes. In your report, please show three good and three bad reconstruction results, by showing the ground-truth and your prediction images from the **validation set**. Please also make sure to discuss your findings, e.g. on what shapes your network performs well and on which it fails? What are the failure patterns? What is the potential reason for the failures?

Problem 3. Variational Auto-Encoders for 3D Shape Generation [16 points]

Although auto-encoders are not generative models, we can easily modify the auto-encoder by adding a KL-divergence regularization over shape latent codes to make it a generative model, which is typically referred to as a variational auto-encoder (VAE). The data for training the variational auto-encoder are the same that we used to train the auto-encoder.

(a) (8 points). Finish the KL-divergence implementation in the `Sampler` class in `model.py` and train the VAE network until convergence using the provided `train_vae.py` script. In your report, please provide the code you wrote for computing the the KL-Divergence, submit the training and validation curves, as well as the reconstruction performance numbers on both data splits. Notice that there is no standard answer to this question and you will get a full grade if the submitted code, the curves, and the CD distances look reasonable.

(b) (4 points). Let us first check how well the network performs for shape reconstruction and how does the results compare to the AE results. In this problem, pick three shapes and show the corresponding ground-truth shapes from **the validation set**, your reconstruction results when you use the autoencoder that we implemented in Problem 2, and the reconstruction results using the VAE. Please write a few sentences discussing your results, e.g., how does the variational auto-encoder performs to comparing to the auto-encoder results?

(c) (4 points). Using the VAE that we just trained, we now want to use it to generate novel shapes. The idea is that you can remove the encoder and only use the decoder that take as input random Gaussian noises, as the latent codes, and generates new 3D point cloud shapes. We have provided the code for the generation in `randgen.py`, so you only have to show 3 randomly generated 3D point cloud shapes that are good and 3 that are bad. Please also make sure to explain your findings.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.