Homework 2

CSE 291 I00: Machine Learning for 3D Data (Winter 2018)

PLEASE READ:
Submit printouts of well commented codes on February 13th 2018. You are encouraged to discuss in groups (two people for each group), but please complete your programming and writeup individually. Indicate the name of your collaborator in your writeup.

1. **Multidimensional Scaling**:
   Recall the concept of Multidimensional scaling that we discussed in class. Given, a dissimilarity or distance matrix of n points in $\mathbb{R}^D$, the task is to find n points in $\mathbb{R}^d$, $d < D$, such that the new distance matrix M' is as close as possible to M.

   - Consider the MNIST dataset of handwritten digits. Sample 10,000 images (X) from the train dataset such that each class has equal number of images(1000). We now have 10000 data points in 784-dimensional space.
   - Construct the distance matrix D for all the pairs of points. D should be of dimension 10000×10000, where $D_{i,j}$ represents the Euclidean distance between the data points X_i and X_j.
   - Project these data points into 2-dimensional space, such that the new distance matrix M' closely approximates M, i.e find new points $(x_1, x_2, \ldots, x_n)$ such that

$$\min_{x_1,x_2,\ldots,x_n} \sum_{i<j} (||x_i - x_j||^2 - d_{i,j})^2$$

   - Solve this optimization problem using Tensorflow. Visualize the new embeddings learnt. Can you find any patterns? Please turn in your code, visualizations and a brief write up on what you infer from the embeddings. (25 marks)

2. **Farthest Point Sampling**:
   This part of the assignment will help you convert a mesh to point cloud. Farthest Point Sampling is a technique that samples points from a mesh. The key idea is to sample the next point from the least known area.

   - Using PyMesh (https://github.com/qnzhou/PyMesh) or PyntCloud(https://github.com/daavoo/pyntcloud) to load violin case obj (and teapot obj) file provided. Familiarize yourself with the package and read in the vertices and the faces.
   - Sample as many points randomly from the mesh (around 10,000 or more) and create a point set P. To sample points uniformly from one triangle, use the following steps(Refer to section 4.2 in `http://www.cs.princeton.edu/~funk/tog02.pdf`):
     - Calculate the area of all the triangles in the mesh. Normalize these areas to get weights of each triangle. Using this, find the number of points to be calculated in each triangle. The number of points in a triangle is given by the total number of points times the weight of a triangle.
     - Let A,B,C be the vertex of a triangle and r_1 and r_2 be the uniform random variables in [0,1]. The new point D is given by:

$$D = (1 - \sqrt{r_1})A + \sqrt{r_1}(1 - r_2)B + \sqrt{r_1 r_2}C$$

   - Compute the distance matrix, where $d_{i,j}$ stands for the distance between point i and point j in set P.
   - Create a point set S and add initial point randomly from P.
   - Compute the on-mesh distance between the points in S and the points in P.
   - Choose the farthest point and add it to set S.
   - Repeat until S contains 1000 points. Visualize the point cloud. Turn in your code and visualizations. (25 marks)

3. **Earth Mover's Distance**:
   For this part of the assignment, use the two point clouds generated in the previous question. Recall the Earth Mover's distance between 2 point clouds is given by:

$$\min_{\phi: S_1 \to S_2} \sum_{x \in S1} ||x - \phi(x)||^2$$

- Consider the two point clouds obtained in the previous task (violin case and teapot). Each point cloud is of dimension 500×3. Compute the bijective mapping using the Hungarian Algorithm and hence compute the Earth Mover's Distance between the two point clouds. Verify your answer by using the inbuilt scipy function (scipy.optimize.linear_sum_assignment()). Turn in a well commented code explaining the steps of the Hungarian Algorithm and also show that your code achieves the same solution as the scipy function.
- Use the tensorflow implementations of the EMD provided. Consider the set of circles (S) with their radii being continuous in some interval, say [1,10]. Sample 500 points from each circle (s). Find set of points(x) such that the EMD between x and the set of circles is minimized, i.e

$$\bar{x} = arg\min_{x} \mathbb{E}_{s\sim S}[d(x,s)]$$

  Turn in well commented code along with the visualization of the points learnt. (25 marks)

4. **Denoising Autoencoder**:
   In class, we saw the use of Encoder(CNN) to encode 2D image into shape embedding space and then decoding(fractionally strided convolutions). These type of architectures are common in many deep learning problems, especially in generative models (GANs) which we will discuss about in future lectures. This part of the assignment will help you in building a denoising autoencoder.

   - An Autoencoder is a useful technique to learn a compressed representation of the input data. As shown in Figure 1, the encoder module transforms the input(x) into some latent space of reduced dimension. The decoder then decodes (upsamples) from the latent space to obtain some reconstruction(x'). The network is trained to minimize this reconstruction loss.
   - Denoising autoencoder is an autoencoder where some noise is added to the input before the encoding phase. When decoding, however the reconstruction is compared to the original image.
   - Use the 60,000 images of the MNIST dataset for this part of the assignment. Add gaussian noise to the input data as a preprocessing step.
   - Construct an encoder that encodes each image to some latent space of 100 dimensions. Use convolutional neural networks to model the encoder. For the decoder, upsample using the fractionally strided convolutions

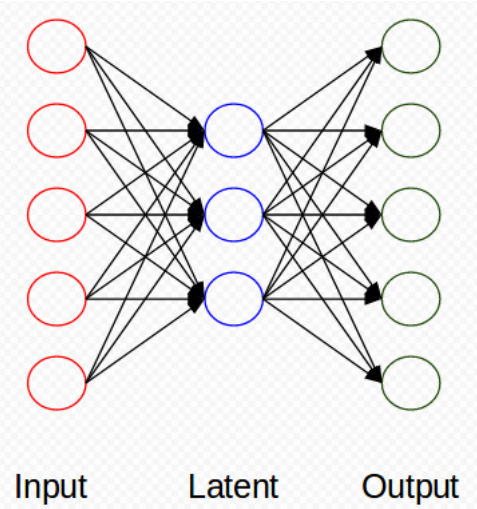

Figure 1: Autoencoder

(deconvolutions) to reconstruct the image. Minimize the reconstruction loss.

- You must submit the code(well commented) and also the visualizations of the original,noisy and reconstructed test images. (25 marks)