# Intelligent Visual Computing [Spring 2023]



## Assignment 4: Neural Surface reconstruction from point clouds

## Overview

In this assignment you will implement a neural implicit surface reconstruction algorithm to approximate a surface represented by scattered point data. The problem statement is the same with assignment 1: given a set of points $\mathbf{P} = \{\mathbf{p_1}, \mathbf{p_2}, ..., \mathbf{p_n}\}$ in a point cloud, we will define an implicit function f(x,y,z) that represents a signed distance to the surface approximated by these points. The surface is extracted at f(x,y,z) = 0 using the marching cubes algorithm. In this assignment, you will implement a more advanced approach to implicit surface reconstruction based on a deep neural network (DeepSDF). Download the starter code and data below (same data as assignment 1).

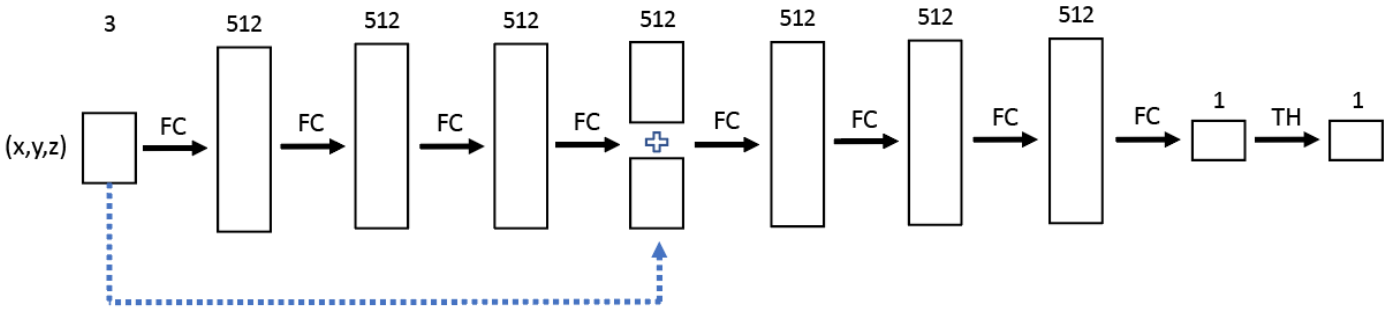
This assignment counts for **20 points** towards your final grade if you are taking the 574 section. **If you are taking the 674 section, multiply your total points by 0.75** (i.e., the assignment counts for 15 points).

## What You Need To Do

**[50%]** You will first implement the architecture and forward pass in "*model.py*" based on the DeepSDF paper (learning Continuous Signed Distance Functions for Shape Representation, CVPR 2019) for neural implicit surface reconstruction. More specifically, given a set of 3D point samples $\mathbf{P'}$ and their SDF values $\mathbf{S} = \{s_1, s_2, ..., s_n\}$, we train the parameters $\boldsymbol{\theta}$ of a multi-layer fully-connected neural network $f_\theta$ on ($\mathbf{P'}$, $\mathbf{S}$) to make $f_\theta$ an effective SDF approximator:

$s_i = f_\theta(\ \boldsymbol{p_i}'\ ), \quad \boldsymbol{p_i}' \in \boldsymbol{P'}, \quad s_i \in \boldsymbol{S}$

The network architecture is shown in the following figure. It takes as input the sample point coordinate $\boldsymbol{p_i}'$=(x,y,z) and passes it through 8 fully-connected layers (FC in figure). The size of the vector representaton in each hidden layer has size 512 as shown in the figure. Note that after the fourth FC layer, the layer outputs the hidden vector with 509-dim, which is concatenated with the original 3-dim point coordinate (x,y,z) to form the final 512-dim vector (see blue dashed line in the figure). In the first seven FC layers (i.e., except the last), a weight normalization layer is appended (nn.utils.weight_norm), then a PReLu non-linear activation layer with a common learnable slope for all channels and a dropout layer (dropout rate=0.1) are added. Finally, the predicted SDF value is obtained with the last FC layer which transforms the vector dimension from 512 to 1 and another tanh activation layer (denoted as TH in the figure).




**[25%]** Complete the code in "*train.py*" that implements the training procedure. The training is done by minimizing the sum over losses between the predicted and real SDF values of sample points under the clamped $L_1$ loss function:

$L(\ f_\theta(\ \boldsymbol{p_i}'\ ), s_i\ ) = |\ clamp(\ f_\theta(\ \boldsymbol{p_i}'\ ), \sigma) - clamp(s_i, \sigma)\ |$

where *clamp(t, σ)* clamps *t* value within the limits (-σ,σ ), where $\sigma=0.1$ in this implementation.

**[20%]** The sampling of training points and their SDF values should be impemented in "*utils.py*". The training 3D points are sampled around the given surface points of the point cloud along their normal directions: $\boldsymbol{p_i}' = \boldsymbol{p_i} + \varepsilon\ \boldsymbol{n_i}$ where ε is randomly sampled from a Gaussian distribution $N(0, 0.05^2)$. For each point $\boldsymbol{p_i}$, create 100 different samples $\{\boldsymbol{p_i}'\}$ for training and validation set.

The neural network weights are optimized by the AdamW optimizer, as specified in the starter code. The default learning rate is set to 0.0001, weight decay is set to 0.0001. The network is trained for 100 epochs with batch size=512. The loss for training and validation set are reported per epoch. The starter code saves the best model with the lowest loss on the validation set. Detailed training hyper-parameters can be found in the training code argument parser.

**[5%]** Show screenshots of the reconstructed bunny (500 and 1000 points) and sphere after training your best model for each of these objects in a PDF report.

# Submission

Please follow the **Submission** instructions in the course policy to upload your zip file to **Gradescope**. The zip file should contain all your Python code **plus a short PDF report including the results**.

python_starter_code.zip April 13 2023, 3:26 PM

## Submission status

| | |
|---|---|
| **Submission status** | This assignment does not require you to submit anything online |
| **Grading status** | Not graded |
| **Due date** | Tuesday, May 2, 2023, 5:00 PM |
| **Time remaining** | 2 days 2 hours |
| **Last modified** | - |
| **Submission comments** | Comments (0) |

◄ Thu Class Meeting

Jump to...

Paper Presentation Preferences (for 574 section) ►