AT3DCV – Challenge 04

In this challenge we compare different basic concepts in 3D deep learning. You are given a dataset containing a set of mesh files in eight different classes. Also, the dataset is broken in train and test set. Here is the general file information.

- *dataset* : containing raw dataset mesh files
- *model_objs* : to store trained pytorch models
- *processed* : the folder to save the processed dataset tensors
- *run.py* : main file to train and call routines, contains task# functions
- ***loader.py*** : contains the dataset files and utils
- *trainer.py* : contains train and validation functions
- ***models.py*** : contains torch neural network models to process the clouds

Task 1 - Point Clouds:

- Complete the mesh_parser() in loader.py with a normalization to normalize the cloud to unit ball [0,1]
- Complete the point_model class in models.py with MLP layers and symmetric aggregation functions for the classification problem. Here is a suggested main sequence inspired by PointNet:

MLP[3,n1]	MLP[n1,n2]	MLP[n2,1024]	Max pooling	FC [1024,n3]	FC[n3,8]

- Generate and save the converted data by running the ch_trainer.train(1). Then train the model with enough number of epochs, e.g. ch_trainer.train(100) and tune the parameters for better convergence.

Task 2- Voxel Grids:

- Similar to parse_to=='point', complete the mesh_parser() in loader.py to construct voxels of size 32x32x32 using open3d.geometry.VoxelGrid() and build a 3D tensor from the voxel grid
- Complete the voxel_model class in models.py with 3D convolutions for the classification problem. Here is a suggested main sequence like inspired by 3DShapeNet:

3dConv[1,n1], pooling	3dConv[n1,n2], pooling	3dConv[n2,n3], pooling	FC[n4,n5]	FC[n5,8]

- Generate and save the converted data by running the ch_trainer.train(1). Then train the model with enough number of epochs, e.g. ch_trainer.train(100) and tune the parameters for better convergence. Compare against Task 1, write your observations.

Task 3 - Spectral Embedding:

- Complete parse_to=='spectral' in loader.py to sample 1024 points from the mesh and find spectral embedding of the point cloud, use sklearn.manifold.SpectralEmbedding and construct complete graphs. Stack spectral features to positional features.

- Complete the spectral_model class in models.py similar to task 1.
- Generate and save the converted data by running the ch_trainer.train(1). Then train the model with enough number of epochs, e.g. ch_trainer.train(100) and tune the parameters for better convergence. Compare against previous tasks, write your observations.

Task 4 - Bonus:

Improve the performance by fusing modalities. (e.g. task 2 and 3, task 1 and 2)