

REFER THIS FOR DGCNN METRICS, PERFORMANCE AND VISUALISATION

[Dynamic Graph Convolutional Neural Network | pyg-point-cloud – Weights & Biases (wandb.ai)](https://wandb.ai/team-dubakur/pyg-point-cloud/reports/Dynamic-Graph-Convolutional-Neural-Network---Vmlldzo2MzkzODgx?accessToken=4cy51ubg3jmq263gjx6zxioqkvdwqtoh4zst5pb3y1vj1p2z6cipvlimvjulfkfo)

1. Install Required Packages

This section installs necessary packages using pip, including

**torch-geometric,**

**torchmetrics, and**

**Weights & Biases (wandb).**

It specifies the torch version and uses URLs to fetch specific versions of

**Torch-scatter,**

**torch-sparse, and**

**torch-cluster.**

2. Import Libraries

Here, the code imports required libraries and modules such as

**Torch,**

**wandb,**

**random,**

**numpy,**

**tqdm, and**

**various components from torch\_geometric** for handling geometric deep learning tasks.

3. Initialize Weights & Biases

This part initializes **Weights & Biases** by setting up a project name, run name, configurations, and initializes the wandb environment. It sets up configurations for various parameters related to the **DGCNN (Dynamic Graph CNN) model**, such as

**random jitter,**

**rotation intervals,**

**batch size,**

**number of workers, etc.**

4. Load ShapeNet Dataset using PyTorch Geometric

This segment involves loading the

**ShapeNet dataset using PyTorch Geometric,**

performing transformations like random jitter and rotation, splitting it into training and validation sets, and creating data loaders for

**training, validation, and visualization.**

5. Implementing the DGCNN Model using PyTorch Geometric

This section defines the architecture of the DGCNN model using

**Dynamic Edge Convolution layers and MLPs (Multi-Layer Perceptrons)**

for feature extraction and classification.

6. Training DGCNN and Logging Metrics on Weights & Biases

Here, the training loop for the DGCNN model is implemented. It consists of

**training steps,**

**validation steps,**

**visualization steps, and**

**checkpoint saving.**

It tracks and logs various metrics like

**loss,**

**Accuracy, and**

**IoU (Intersection over Union)**

during training and validation using Weights & Biases