EDGE SOUP: CONTINUOUS-SCALE POINT CLOUD LEARNING WITH A SINGLE GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

Point-cloud learning is dominated by hierarchical architectures that explicitly discretize geometric information into a fixed set of scales. We introduce *Edge Soup*, a graph construction method that instead models scale as a continuous property of nodes within a single graph. By assigning points a scale coordinate and performing a k-nearest-neighbor query in an augmented space, we generate a homogeneous "soup" of edges that has connectivity ranging from local to long-range. The resulting graph reproduces structural characteristics of hierarchical representations while requiring only a single graph-construction step. A simple sequential Graph Neural Network (GNN) operating on this graph achieves competitive performance on challenging point-cloud segmentation benchmarks without the need for separate graphs or cross-scale connections like pooling and unpooling. Such a network exhibits increased robustness to density variation, noise, and sampling artifacts, maing it particularly effective on large-scale LiDAR datasets. Because Edge Soup replaces expensive hierarchy construction with a single graph building step, it substantially reduces data preparation overhead and lowers end-to-end inference latency on large point clouds.

1 INTRODUCTION

Capturing geometric structure across multiple scales is a fundamental objective in point-cloud learning. Fine-scale information is needed to recognize local surface details, while larger-scale context is often necessary to reason about object shape, semantic categories, and scene layout. As a result, most successful point-cloud architectures rely on hierarchical multiscale representations that repeatedly downsample the point cloud and process it at progressively coarser resolutions (e.g., PointNet++ Qi et al. (2017) and the recent PTv3 Wu et al. (2024)).

Hierarchical representations, while effective, come at a cost. They require explicitly constructing multiple graphs, along with dedicated downsampling, upsampling, and cross-scale communication mechanisms. More fundamentally, they impose a prior that geometric information is organized into a small set of discrete, hand-designed scales. This raises a natural question: can multiscale point-cloud learning be achieved without an explicit hierarchy and without discretizing scale?

In this paper, we introduce *Edge Soup*, a graph for point clouds that models scale as a continuous property of nodes within a single graph. Rather than constructing separate graphs at different resolutions, we assign each point a scale value ranging from coarse to fine and generate connectivity that respects these assignments. The resulting graph is a homogeneous "soup" of edges of varying length that naturally includes connections across a continuum of scales.

Constructing an Edge Soup is computationally simple. Each point is assigned one additional coordinate derived from its scale value, lifting the point cloud into a higher-dimensional space. A single k-nearest-neighbor (kNN) query in this augmented space then produces edges that connect points with a similar scale value, creating longer connections between coarse nodes and shorter connections between fine nodes (Fig. 1). This yields multiscale connectivity without explicitly constructing a hierarchy. Unlike conventional multiscale architectures, the resulting graph does not distinguish between convolution and cross-scale connections like pooling and unpooling; all communication flows through a single homogeneous graph structure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Despite its simplicity, Edge Soup reproduces key structural characteristics of hierarchical graph representations, including comparable path lengths, clustering behavior, and edge-length distributions. A simple sequential graph neural network operating on the resulting graph achieves performance competitive with established hierarchical architectures while exhibiting improved robustness to noise, density variation, and sampling artifacts. Because Edge Soup replaces the construction of multiple graphs with a single graph-building step, it also substantially reduces graph construction costs and overall inference latency.

We evaluate Edge Soup along multiple dimensions. First, we analyze the structural properties of the resulting graphs and show that they closely resemble hierarchical graph representations despite being constructed as a single graph. Second, we apply Edge Soup to a variety of classification and segmentation benchmarks, demonstrating competitive accuracy across diverse point-cloud domains. Third, we evaluate robustness under a range of input corruptions and observe consistent improvements over comparable hierarchical architectures, particularly under variations in sampling density and noise. Finally, we measure computational performance and find that the simplified graph construction translates into substantially faster graph generation and reduced end-to-end inference latency.

Our contributions are as follows:

- We introduce Edge Soup, a simple method for constructing a graph from a point cloud such that nodes are distributed across a continuum of scales, with short and long edges.
- We propose an analytical procedure for controlling the scale of nodes such that the resulting Edge Soup graph has properties similar to those of hierarchical representations.
- We demonstrate that a sequential GNN operating on Edge Soup graphs can achieve competitive benchmark performance, with high robustness against sampling artifacts such as noise points and density variation.
- We show that Edge Soup substantially reduces graph-construction cost and overall inference latency compared to hierarchical alternatives.

2 Related Work

Local graph construction Most graph-based point cloud learning methods construct neighborhoods from spatial proximity. Since point clouds lack a natural notion of connectivity, unlike meshes or sparse voxel grids (Choy et al., 2019; Graham et al., 2018), the graph itself becomes an important design choice. A common approach is radius-based connectivity, where points are connected to all neighbors within a fixed spatial radius. This strategy was popularized by PointNet and PointNet++ (Qi et al., 2016; 2017) and remains widely used in subsequent architectures such as PointNeXT (Qian et al., 2022). Another common choice is k-nearest-neighbor (kNN) connectivity, where each node is connected to a fixed number of nearby points. While kNN does not directly control edge lengths, the fixed neighborhood size enables highly optimized implementations and is therefore employed in efficient architectures such as DeLA (Yang et al., 2024). Recent transformer-based methods often approximate neighborhood queries through spatial sorting. By ordering points according to Hilbert or Morton curves, nearby points in space tend to remain nearby in the resulting sequence. This enables the construction of very large neighborhoods at low cost and is now used in the Point Transformer family (Zhao et al., 2021; Wu et al., 2022; 2024). Despite their differences, these methods all produce graphs where nodes are connected only to others nearby in space.

Hierarchical multiscale representations To overcome the limitations of purely local connectivity, many successful point-cloud architectures employ hierarchical multiscale representations. These methods repeatedly resample the point cloud and apply local convolutions at progressively coarser resolutions. In many cases, this takes the form of a U-Net architecture, where information is aggregated along a hierarchy through downsampling and upsampling operations (Qi et al., 2017; Qian et al., 2022; Yang et al., 2024). Other architectures, such as PointHR (Qiu et al., 2023) and GMNN Campolattaro et al. (2026), maintain parallel graphs at multiple resolutions and enable communication between scales throughout the network. Our work aims to achieve the same effect without the need to explicitly construct such hierarchies or define the connections between their levels.

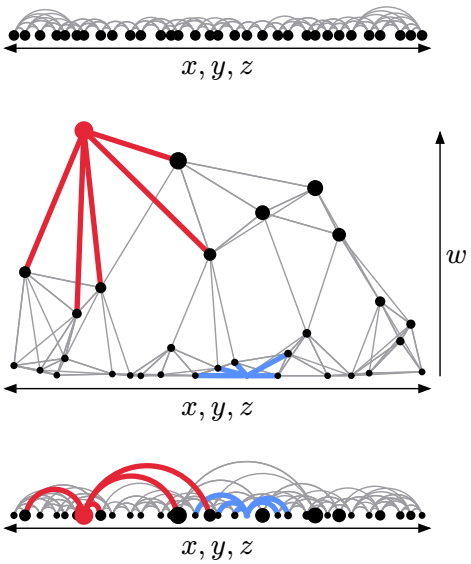


Figure 1: Edge Soup construction in 1D. (Top) Simple kNN. (Middle) Nodes stratified along dimension w. (Bottom) Resulting Edge Soup. The coarsest node and its edges are highlighted in red, the finest in blue.

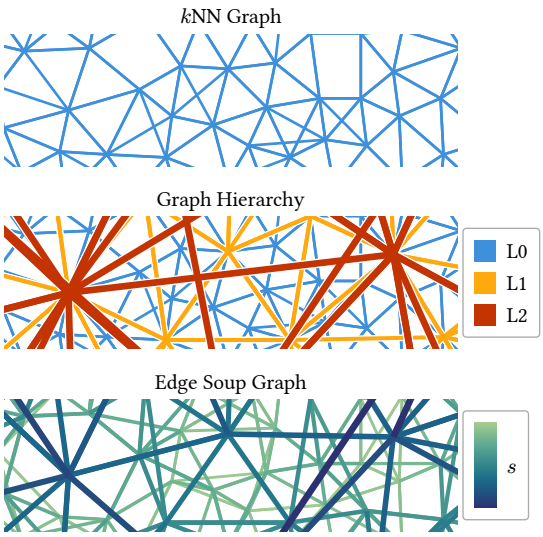


Figure 2: Edge Soup (bottom) vs alternatives. A simple kNN graph (top) contains only short edges. A multiscale approach (middle) uses different graphs on multiple scales. A soup graph (bottom) contains edges on a continuum of scales.

Alternative mechanisms for long-range communication Several methods extend the receptive field of network layers without explicitly constructing multiscale hierarchies. Dilated graph convolutions (Li et al., 2019; Qian et al., 2021) increase communication range by sparsifying neighborhoods, allowing information to propagate over multiple scales while maintaining the original graph resolution. Unlike these methods, Edge Soup organizes communication across scales at the graph level rather than the convolution level and naturally supports a continuous range of scales. Other approaches modify graph connectivity itself. DGCNN (Wang et al., 2019) dynamically recomputes neighborhoods in feature space per layer, enabling long-range interactions between semantically similar points. While effective, this requires repeated high-dimensional nearest-neighbor searches during training and inference. DiffusionNet (Sharp et al., 2022) propagates information through spectral diffusion operators, decoupling communication from graph connectivity but requiring costly additional preprocessing.

Robustness to sampling artifacts Real-world point clouds frequently exhibit density variation, missing observations, and sensor noise. Prior work has primarily addressed robustness through stronger training augmentations, including PointMixup (Chen et al., 2020), PointCutMix (Zhang et al., 2022), and PointWOLF (Kim et al., 2021). Other methods adapt their representation to account for non-uniform sampling densities. For example, PolarNet and Cylinder3D warp the underlying coordinate system to produce more favorable neighborhood structures for LiDAR data (Zhang et al., 2020; Zhu et al., 2020). In contrast, we find that multiscale connectivity within a single graph yields substantial robustness improvements without requiring specialized augmentations or task-specific coordinate systems.

Efficient point-cloud learning Other works have focused on improving computational efficiency. Some approaches accelerate convolutional operators (Tang et al., 2020; Yang et al., 2024), while others reduce the cost of hierarchy construction through faster downsampling (Thomas et al., 2019; Hu et al., 2020; Wang et al., 2017; Xu et al., 2020) or more efficient neighborhood queries (Wu et al., 2024). Our approach is complementary to these efforts. Rather than optimizing the construction of a hierarchy, Edge Soup removes the hierarchy entirely, replacing it with a single graph construction step.

3 EDGE SOUP GRAPH

Edge Soup is a graph construction method designed to enable continuous multiscale point-cloud learning without an explicit hierarchy. Rather than constructing separate graphs at different resolutions, Edge Soup treats scale as a property of individual nodes within a single graph. In this section, we describe the graph construction procedure, analyze the resulting graph structure, and discuss computational costs in comparison to alternatives.

3.1 GRAPH CONSTRUCTION

The central idea of Edge Soup is to treat scale as a property of individual nodes rather than creating separate levels of a hierarchy. Each point is stochastically assigned a scale $s \in \mathbb{R}$ that influences its connectivity. Points associated with coarser scales are encouraged to form longer connections, while finer points primarily connect over short distances. The resulting graph, therefore, contains both short- and long-range interactions and supports communication across the continuum of scales.

To construct the graph efficiently, each point $\mathbf{p}_i = (x_i, y_i, z_i)$ is lifted into a four-dimensional space by appending a scale-dependent coordinate w_{s_i}, resulting in the augmented point $\tilde{\mathbf{p}}_i = (x_i, y_i, z_i, w_{s_i})$, where s_i denotes the assigned scale of point i. A standard k-nearest-neighbor (kNN) query in this augmented space then generates the graph edges. By choosing s and w appropriately, nodes preferentially connect to other nodes of similar scale while still permitting communication with adjacent scales. This yields multiscale connectivity through a single graph-construction step. Figure 1 illustrates the process for a one-dimensional example.

The scale assignment is defined as a monotonically increasing function s_n over point indices $n \in [0, \dots, N)$. Indices are assigned using a random order or by following the sampling sequence of Furthest-Point Sampling (Eldar et al., 1997) or Inverse-Density Importance Sampling (Hu et al., 2020). Lower values of s_n correspond to coarser nodes, while higher values correspond to finer nodes. Thus, s should be understood as an assignment of scales from coarse to fine rather than a physical scale parameter. To mimic the population growth often used in hierarchical representations, where each successive level contains a fixed multiple of the points on the previous level, we distribute scales logarithmically. Specifically, the number of points assigned to scales in the interval $[s, s+1)$ grows proportionally to Q^s, matching the growth pattern of a conventional hierarchy. The scale function is therefore defined as

$$s_n = \max\left[0,\ \log_Q\left(\frac{n(Q-1)}{N_0} + 1\right) - 1\right], \tag{1}$$

where N_0 denotes the number of nodes on the coarsest scale and Q controls the population growth across scales. The resulting scale assignments are illustrated in Fig. 3. The scale increases by one whenever an additional $Q^s N_0$ points have been assigned, creating a continuum of scales whose population closely resembles that of a multilevel hierarchy.

To determine the additional coordinate w_s, we seek to emulate another important property of hierarchical representations: information is typically exchanged between neighboring levels, while direct communication across many levels is rare. The relevant quantity is the expected connection radius of nodes on scale s, denoted by ρ_s. Intuitively, if the separation between neighboring scales is comparable to the connection radius of nodes operating at that scale, then nodes can communicate across adjacent scales while avoiding direct connections across many scales. This motivates our choice of condition $w_s - w_{s+1} = \rho_s$.

Figure 3: Node scale s as a function of index.

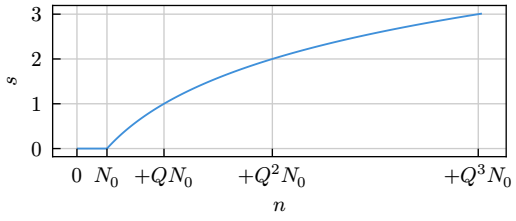
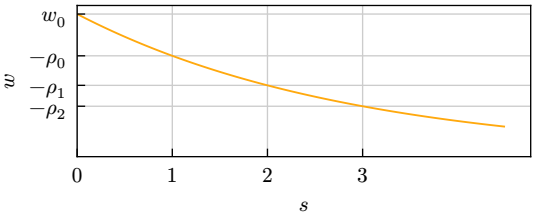


Figure 4: w-coordinate as a function of scale.



The derivation of ρ_s is provided in Section A.1. Solving this relationship yields the weight function

$$w_s = \frac{Q^{-\frac{s+1}{2}}}{Q^{\frac{1}{2}} - 1}\rho_0 \tag{2}$$

whose behavior is illustrated in Fig. 4. The precise derivation is given in Section A.2. The resulting construction produces a single graph with connectivity spanning the continuum of scales and connections between nodes of different scales.

3.2 Soup Graph Properties

If Edge Soup is to serve as a continuous-scale alternative to hierarchical multiscale representations, it should exhibit similar structural properties. In particular, we would like to preserve three characteristics that make hierarchical methods effective: efficient long-range communication, strongly localized neighborhoods, and a broad distribution of edge-lengths. We therefore compare Edge Soup graphs to conventional kNN graphs and hierarchical graph representations derived from U-Net architectures (depicted in Fig. 2). To enable a direct comparison with the latter, a four-level hierarchy is collapsed into a single graph by concatenating its nodes and adjacency matrices.

Long-range communication Long-range edges in multiscale architectures enable information to propagate between distant regions of the domain, which is critical for incorporating non-local context in tasks such as segmentation. Edge Soup graphs are expected to provide similar capabilities through their longer edges. This property can be quantified via the characteristic path length, a common measure from network theory defined as the average number of edges in the shortest path between every pair of vertices (Watts and Strogatz, 1998). In Table 1, path lengths are measured on graphs constructed from ShapeNet-Part (Chang et al., 2015) point clouds, which are assumed to form a single connected component, ensuring finite distances. The kNN graph represents a worst-case baseline, as it lacks long-range edges entirely, while a random graph with k edges per node represents an upper bound. The Edge Soup graph exhibits mean path lengths comparable to those of the merged hierarchical graph. For a sequential GNN operating on a soup graph, the average path length directly relates to the number of message-passing layers required for information to propagate between arbitrary points.

Local connectivity In addition to global communication, strong local connectivity is essential for effective feature aggregation. This property is quantified using the clustering coefficient, another concept from network theory that measures the proportion of shared neighbors between connected nodes (Watts and Strogatz, 1998). As shown in Table 1, the kNN graph provides an upper bound for this metric, since spatial proximity naturally induces highly clustered neighborhoods. In contrast, random graphs achieve short path lengths at the expense of extremely low clustering. The soup graph again closely matches the hierarchical graph, balancing local coherence with global connectivity.

Table 1: Graph Statistics on ShapeNet

Graph	Path Length ↓	Clustering ↑
kNN	**11.0 ± 5.7**	**0.33**
Random	**2.8 ± 0.5**	**0.01**
U-Net Hierarchy	7.3 ± 2.5	0.28
Soup (Ours)	7.2 ± 2.9	0.30

Edge-length distribution To further analyze graph structure, the distribution of edge lengths is examined across different graph constructions. In Fig. 5, the relative frequency of edge lengths is shown for hierarchical (U-Net), soup, and kNN graphs on ShapeNet-Part and S3DIS point clouds. The kNN graph exhibits a narrow distribution centered around a characteristic local scale, while both hierarchical and Edge Soup constructions extend this distribution to include longer edges. For S3DIS, where U-Net levels are typically defined via voxelization for efficiency, the hierarchical graph shows distinct peaks corresponding to the individual levels. In contrast, Edge Soup produces a smooth distribution of edge lengths that spans a continuum of scales, reproducing multiscale connectivity without explicitly discretizing scale.

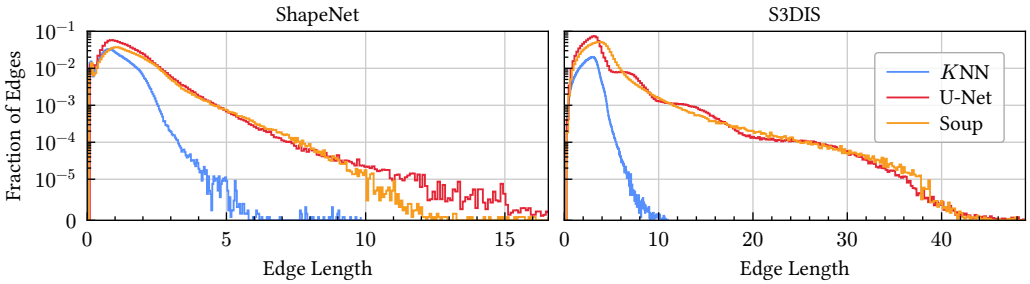


Figure 5: Histograms showing the distribution of edge lengths for different graphs types on one batch of ShapeNet and S3DIS.

3.3 Construction Speed

A practical advantage of Edge Soup is that multiscale connectivity is obtained through a single graph-construction step. Building a hierarchy with L levels requires L separate 3D kNN queries on progressively smaller point sets, along with $L - 1$ additional queries to establish parent-child relationships between levels. In contrast, an Edge Soup graph is constructed using a single kNN query in an augmented four-dimensional space. For efficient tree-based nearest-neighbor implementations, increasing the dimensionality from three to four incurs only a small overhead, while the overall complexity remains $O(N \log N)$. As shown in Section 4.3, this simplification translates into practical speedups, with Edge Soup graphs being constructed substantially faster than comparable hierarchical representations.

4 Experiments

We begin by describing the GNN model we apply on Edge Soup graphs, and then evaluate the combined system along several axes. First, we measure semantic segmentation performance on large-scale outdoor LiDAR datasets characterized by severe density variation and irregular sampling. Second, we evaluate computational efficiency in comparison to an efficient U-Net and transformer model. Third, we analyze the contribution of the Edge Soup graph and continuous scale through ablation studies. Finally, we assess robustness under a range of synthetic corruptions outside the training distribution. We do not expect accuracy gains on regularly-sampled data, as Edge Soup graphs feature similar connectivity to U-Nets (see Section 3.2). To ensure Edge Soup maintains baseline performance, we provide additional results on object classification (ModelNet40, ScanObjectNN), part segmentation (ShapeNet-Part), and indoor scene segmentation (S3DIS) in Appendix B.

4.1 Experimental Setup

To isolate the contribution of the graph construction method, we employ a simple sequential graph neural network operating directly on the Edge Soup graph. Because all convolutions operate on the full graph, memory requirements are higher than in a hierarchy where later layers process progressively smaller point sets. We therefore adopt the highly optimized DeLA convolution design (Yang et al., 2024), retaining its training procedures wherever applicable. On the largest datasets, we use gradient accumulation, gradient-checkpointing, and randomly drop some points from each batch to further reduce memory requirements during training. Further details about configuration choices are available in Appendix B.

The network must also operate across a wide range of edge lengths. Rather than enforcing scale invariance, we allow the model to learn scale-dependent behavior by decomposing each edge vector into a normalized direction and magnitude $p'_{ij} = (\frac{p_{ij}}{|p_{ij}|}, |p_{ij}|)$. Only the initial spatial encoding layer is modified; subsequent convolutions remain identical to those used by DeLA.

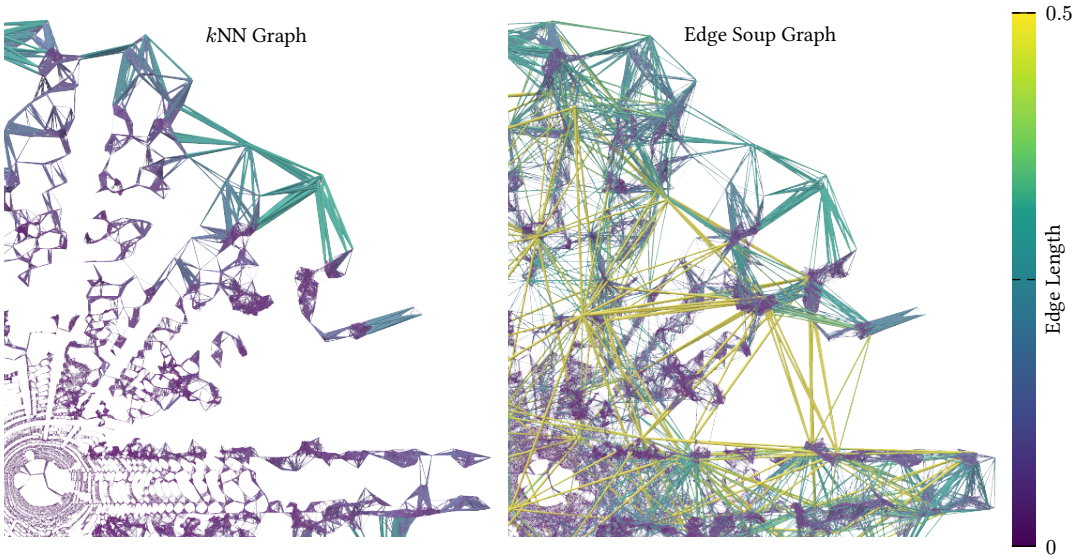


Figure 6: Top-down view of KITTI data. Input point clouds exhibit strong density variation and anisotropic sampling. A kNN graph has edge lengths determined only by local sampling density and thereby inherits these artifacts, whereas Edge Soup produces connectivity spanning a broader range of scales.

Table 2: Segmentation mIoU on Outdoor Scene Segmentation Tasks

Method	Params.	KITTI val.	nuScenes val.	Waymo val.
MinkowskiNet Choy et al. (2019)	21.7M	63.8	63.8	65.9
SPVNAS Tang et al. (2020)	10.8M	64.7	77.4	-
Cylinder3D Zhu et al. (2020)	55.9M	64.3	76.1	-
SphereFormer Lai et al. (2023)	32.3M	67.8	78.4	69.9
PTv2 Wu et al. (2022)	12.8M	70.3	80.2	70.6
PTv3 Wu et al. (2024)	46.2M	68.3	**80.4**	**71.3**
Soup (Ours)	**629k**	**70.6**	79.3	69.3

4.2 Outdoor LiDAR Segmentation

Outdoor LiDAR datasets are particularly relevant to the motivation behind Edge Soup. Unlike synthetic point clouds or uniformly sampled scenes, these datasets exhibit extreme variation in sampling density, strong anisotropy, and substantial occlusion effects. Figure 6 illustrates a representative example from KITTI (Geiger et al., 2012). Points near the sensor are densely sampled while distant regions become increasingly sparse. Such characteristics complicate the construction of effective multiscale hierarchies, and so many otherwise competitive methods like DeLA cannot be directly applied to this sort of data. We evaluate on KITTI, nuScenes (Fong et al., 2021), and Waymo (Sun et al., 2020), comparing against state-of-the-art point-based approaches without pretraining or cross-training. Results are shown in Table 2.

Despite using a fraction of the parameters of competing methods ($1-5\%$), Edge Soup achieves competitive performance across all three datasets. The strongest results are observed on KITTI, where Edge Soup achieves state-of-the-art validation performance among the methods considered. We hypothesize that the continuous-scale connectivity provided by Edge Soup is particularly beneficial in settings with severe density variation and irregular sampling.

Table 3: Mean Inference Latency (ms)
Total (Construction + Forward Pass)

	nuScenes		S3DIS		ShapeNet	
PTv3	112	(79 + 33)	1389	(1309 + 80)		-
DeLA		-	137	(124 + 13)	21	(10 + 11)
Soup (Ours)	**34**	(27 + 7)	**96**	(83 + 13)	**13**	(6.5 + 6.7)

4.3 SPEED

A major practical motivation behind Edge Soup is the elimination of explicit graph hierarchies. We therefore measure end-to-end inference latency, including both graph construction and network evaluation. Disk-IO and transfers between devices constitute less than 1% of runtime and are not included. Results are shown in Table 3. For comparison, we evaluate against our reproduction of DeLA, whose convolution operator forms the basis of our implementation, and Point Transformer V3, the current state-of-the-art transformer architecture. DeLA has no configuration on outdoor segmentation tasks like nuScenes, and PTv3 has no configuration on smaller tasks like ShapeNet. For DeLA and Soup, graph and hierarchy construction is performed on the CPU using an optimized kNN implementation (nanoflann Blanco and Rai (2014)) and fast voxelization and Furthest Point Sampling algorithms where relevant. Inference is performed on an NVIDIA RTX 4090 GPU.

Edge Soup substantially reduces graph-construction cost in comparison to DeLA while maintaining a simple sequential architecture, translating to a meaningful reduction in overall inference time on scenes of all sizes. We also include PTv3 for context. This model does not require kNN queries to find neighborhoods, but it uses repeated voxel downsampling to produce a multiscale hierarchy. It uses especially dense sampling on S3DIS to improve accuracy and a less optimized voxelization algorithm, resulting in high construction costs for very high overall latency.

4.4 GRAPH ABLATION

To isolate the contribution of the graph itself, we train identical networks using either a conventional kNN graph or an Edge Soup graph. Results are presented in Table 4 as mean and standard deviation over three runs on each dataset.

Table 4: Graph vs. Accuracy (mIoU)

Graph	nuScenes	S3DIS	ShapeNet
kNN	76.1 ± 0.4	70.3 ± 0.2	85.4 ± 0.2
Soup$^{(\text{quantized})}$	77.7 ± 0.3	70.4 ± 0.4	**85.7 ± 0.2**
Soup (Ours)	**78.0 ± 0.4**	**71.4 ± 0.4**	**85.7 ± 0.1**

Replacing the local kNN graph with Edge Soup consistently improves segmentation accuracy across all datasets. The effect is most pronounced on nuScenes, suggesting that access to multiscale connectivity becomes increasingly important as scene size and variability grow. Since all experiments use the same network architecture and training procedure, the improvements can be attributed directly to the choice of graph.

To evaluate the importance of continuous scale, we compare Edge Soup against a quantized variant, denoted Soup$^{(\text{quantized})}$, in which node scales are rounded to the nearest integer. This produces a small number of discrete scale levels while leaving the graph construction and network architecture otherwise unchanged. The quantized variant performs consistently at or below the continuous formulation. This suggests that the accuracy advantage of Edge Soup depends on both the presence of multiple scales and the continuous nature of those scales. On ShapeNet-Part, where point clouds are smaller and more uniformly sampled, both variants perform similarly.

Table 5: Overall Accuracy on ModelNet40 and various corruptions

Model	Mean↑	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate	(Clean)
PointNet	65.8	88.1	**79.7**	87.6	77.8	12.1	56.2	59.1	90.7
PointNet++	75.1	91.8	62.8	84.1	62.7	81.9	72.7	69.8	93.0
DGCNN	76.4	90.6	68.4	75.2	79.3	70.5	72.5	78.5	92.6
SimpleView	75.7	91.8	77.4	69.2	71.9	71.0	76.8	71.7	**93.9**
CurveNet	77.9	91.8	77.1	82.4	78.8	60.3	72.5	**82.6**	93.8
RPC	79.5	92.1	71.8	**87.8**	83.5	72.6	72.2	76.8	93.0
DeLA	76.4	92.1	62.5	80.6	86.6	66.1	73.5	73.6	93.5
Soup (Ours)	**83.4**	**92.9**	62.2	86.5	**88.2**	**91.3**	**87.9**	74.3	93.7

4.5 Robustness

To evaluate robustness against sampling artifacts outside the training distribution, we measure ModelNet40 accuracy under the corruption benchmark of Ren et al. (2022). The benchmark includes geometric perturbations such as scaling, rotation, and jitter, as well as sampling-related corruptions including point removal and the addition of noise points. For context, we include their reported numbers for the most robust models an our reproduction of DeLA. Results are shown in Table 5.

Edge Soup exhibits particularly strong robustness to additive and subtractive corruptions, both local (L) and global (G). There is no special advantage on changes to the underlying geometry like rotation and jitter. It is plausible that this behavior arises from the continuous-scale structure of the graph: In a hierarchical architecture, convolutions at each level are each trained on a controlled range of edge lengths. In contrast, convolutions over an Edge Soup graph are exposed to neighborhoods spanning the continuum of scales, potentially improving generalization when sampling density or graph structure differs from those observed during training.

5 Discussion and Conclusion

We introduced *Edge Soup*, a continuous-scale graph construction method for point-cloud learning. Unlike conventional multiscale architectures, which rely on explicit hierarchies of discrete resolutions, Edge Soup treats scale as a property of nodes within a single graph, yielding connectivity spanning a continuum of scales. The resulting graphs reproduce key structural properties of hierarchical representations, and enable a simple sequential GNN to achieve competitive performance across classification and segmentation tasks. Beyond accuracy, Edge Soup provides two practical advantages: improved robustness to sampling artifacts like density variation and noise, and a simple graph construction procedure that requires only a single kNN query and results in lower end-to-end inference latency on large point clouds.

The primary limitation of the current approach is memory efficiency. Whereas a U-Net processes progressively smaller point sets at deeper layers, every convolution in our sequential architecture operates on the full graph. Future work could address this with scale-aware operators could selectively process subsets of nodes and recreate the behavior of a U-Net or more elaborate gating mechanisms that make the architecture itself learnable.

Edge Soup should be viewed as an alternative to graphs based on neighborhood connectivity rather than a particular network architecture. The same construction could be combined with alternative convolutions or even attention-based models. For example, architectures such as PointTransformerV3 could potentially incorporate continuous-scale connectivity by extending their spatial-ordering schemes from three to four dimensions. Our proposed procedure produces graphs with properties of hierarchical representations, but alternative approaches may produce graphs with properties more suitable for learning.

More generally, our results suggest that explicit multiscale hierarchies are not the only effective way to organize point-cloud computation. Edge Soup offers an alternative approach to multiscale point-cloud representations that is both conceptually simple and computationally efficient. We hope this perspective motivates further exploration of continuous-scale representations for point-cloud learning and related 3D perception tasks.

AI USE STATEMENT

The original manuscript was drafted and edited by the authors. We used AI-assisted tools for word- and sentence-level language-related tasks, including grammar checking, proofreading, and polishing clarity and readability. The authors reviewed and approved all AI-assisted edits and remain fully responsible for the content, claims, analyses, and conclusions presented in this submission.

REFERENCES

Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1534–1543. doi:10.1109/CVPR.2016.170

Jose Luis Blanco and Pranjal Kumar Rai. 2014. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. https://github.com/jlblancoc/nanoflann

Jackson Campolattaro, Ruben Wiersma, and Klaus Hildebrandt. 2026. Geometric multigrid neural networks. *Comput. Aided Geom. Des.* 128 (2026), 102571. doi:10.1016/J.CAGD.2026.102571

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *ArXiv* abs/1512.03012 (2015). https://api.semanticscholar.org/CorpusID:2554264

Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees G. M. Snoek. 2020. PointMixup: Augmentation for Point Clouds. In *European Conference on Computer Vision (ECCV)*. Springer-Verlag, Berlin, Heidelberg, 330–345. doi:10.1007/978-3-030-58580-8_20

Chris Choy, JunYoung Gwak, and Silvio Savarese. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. 3070–3079. doi:10.1109/CVPR.2019.00319

Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. 1997. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing* 6, 9 (1997), 1305–1315. doi:10.1109/83.623193

Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. 2021. Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. *arXiv preprint arXiv:2109.03805* (2021).

Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 2018. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9224–9232.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. 2020. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020).

Sihyeon Kim, Sanghyeok Lee, Dasol Hwang, Jaewon Lee, Seong Jae Hwang, and Hyunwoo J. Kim. 2021. Point Cloud Augmentation With Weighted Local Transformations. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 548–557.

Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. 2023. Spherical Transformer for LiDAR-based 3D Recognition. In *CVPR*.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go as Deep as CNNs?. In *The IEEE International Conference on Computer Vision (ICCV)*.

Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. *arXiv preprint arXiv:2202.07123* (2022).

Jinyoung Park, Sanghyeok Lee, Sihyeon Kim, Yunyang Xiong, and Hyunwoo J. Kim. 2023. Self-Positioning Point-Based Transformer for Point Cloud Understanding. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 21814–21823. doi:10.1109/CVPR52729.2023.02089

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv preprint arXiv:1612.00593* (2016).

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5105–5114.

Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. 2021. PU-GCN: Point Cloud Upsampling Using Graph Convolutional Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11683–11692.

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. In *Advances*.

Haibo Qiu, Baosheng Yu, Yixin Chen, and Dacheng Tao. 2023. PointHR: Exploring High-Resolution Architectures for 3D Point Cloud Segmentation. *arXiv preprint arXiv:2310.07743* (2023).

Jiawei Ren, Liang Pan, and Ziwei Liu. 2022. Benchmarking and Analyzing Point Cloud Classification under Corruptions. In *International Conference on Machine Learning (ICML)*.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Trans. Graph.* 41, 3 (March 2022). doi:10.1145/3507905

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. 2020. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. 2020. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *European Conference on Computer Vision*.

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. *Proceedings of the IEEE International Conference on Computer Vision* (2019).

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. 2019. Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In *International Conference on Computer Vision (ICCV)*.

Changshuo Wang, Meiqing Wu, Siew-Kei Lam, Xin Ning, Shangshu Yu, Ruiping Wang, Weijun Li, and Thambipillai Srikanthan. 2024. GPSFormer: A Global Perception and Local Structure Fitting-Based Transformer for Point Cloud Understanding. In *European Conference on Computer Vision*. Springer-Verlag, Berlin, Heidelberg, 75–92. doi:10.1007/978-3-031-73242-3_5

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-Based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4 (July 2017). doi:10.1145/3072959.3073608

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* 38, 5 (Oct. 2019). doi:10.1145/3326362

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (June 1998), 440–442. doi:10.1038/30918

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024. Point Transformer V3: Simpler, Faster, Stronger. In *CVPR*.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. 2022. Point transformer V2: Grouped Vector Attention and Partition-based Pooling. In *NeurIPS*.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. 2020. Grid-GCN for Fast and Scalable Point Cloud Learning. In *CVPR*.

Weikang Yang, Xinghao Lu, Binjie Chen, Chenlu Lin, Xueye Bao, Weiquan Liu, Yu Zang, Junyu Xu, and Cheng Wang. 2024. DeLA: An extremely faster network with decoupled local aggregation for large scale point cloud learning. *International Journal of Applied Earth Observation and Geoinformation* 135 (2024), 104255. doi:10.1016/j.jag.2024.104255

Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujin Chen, Yanmei Meng, and Danfeng Wu. 2022. Pointcutmix: Regularization strategy for point cloud classification. *Neurocomputing* (2022).

Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. 2020. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. 2021. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16259–16268.

Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. 2020. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. *arXiv preprint arXiv:2011.10033* (2020).

APPENDIX

Section A derives the Edge Soup graph-construction equations and discusses parameter selection. Section B reports additional benchmark results omitted from the main paper due to space constraints, along with the configurations used on all benchmarks.

A GRAPH CONSTRUCTION DERIVATION

This section derives the scale-dependent node weights used in Edge Soup graph construction. We first estimate the characteristic edge length associated with each scale and then derive a weight function that reproduces the communication structure of a hierarchical multiscale representation.

A.1 EXPECTED EDGE LENGTHS

If we assume the nodes are distributed according to a Poisson point process, then the spherical contact distribution function can be used to estimate edge lengths. The expected connection radius ρ for a kNN graph is given by the expected distance R_k to the kth nearest neighbor,

$$\rho = \mathbb{E}[R_k] = \frac{\Gamma(k + \frac{1}{d})}{\Gamma(k)} (\lambda V_d)^{-\frac{1}{d}}, \qquad V_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(1 + \frac{d}{2})}, \tag{3}$$

where d is the intrinsic dimensionality of the point cloud, V_d is the volume of the unit ball in dimension d, and λ is the point density.

Let λ_0 denote the density of the coarsest scale. Since density is proportional to point count, the density at scale s is

$$\lambda_s = \lambda_0 \frac{N_s}{N_0} = \lambda_0 Q^s. \tag{4}$$

Substituting into the expression above yields the expected connection radius for nodes on scale s:

$$\rho_s = \frac{\Gamma(k + \frac{1}{d})}{\Gamma(k)} (\lambda_0 Q^s V_d)^{-\frac{1}{d}} = Q^{-\frac{s}{d}} \rho_0, \tag{5}$$

where ρ_0 denotes the expected connection radius for the coarsest scale.

A.2 SCALE-DEPENDENT NODE WEIGHTS

To reproduce the communication pattern of a hierarchical representation, nodes on neighboring scales should be allowed to connect, while direct communication across many scales should be discouraged. Although other choices are possible, we adopt a simple construction in which adjacent scales are separated by approximately one characteristic connection radius

$$w_s - w_{s+1} = \rho_s. \tag{6}$$

Solving this recurrence yields

$$w_s = \frac{\rho_{s+1}}{Q^{\frac{1}{d}} - 1} = \frac{Q^{-\frac{s+1}{d}} \rho_0}{Q^{\frac{1}{d}} - 1}. \tag{7}$$

This construction produces a smooth stratification that promotes interactions between neighboring scales while discouraging distant ones.

A.3 PARAMETER SELECTION

The Edge Soup formulation depends on four parameters: d, k, N_0, and Q. The first two have natural defaults. We set $d = 2$ for point clouds sampled from surfaces and use $k \approx 20$, consistent with PointNet++-style convolutions. The remaining parameters, N_0 and Q, determine the distribution of scales and therefore require task-specific consideration.

Table 6: Model and Soup hyperparameters

Task	Model		Soup		
	Width	Blocks	Coarse Nodes	Q	Ordering (train/test)
Classification	192	4	$N_0 = k$	4.0	Random/FPS
Part Segmentation	192	6	$N_0 = k$	2.0	Random/FPS
Indoor Scene Segmentation	160	6	$\lambda_0 = 0.5$	3.0	Random
Outdoor Scene Segmentation	96	4	$\lambda_0 = 0.01$	2.0	Inv. Density

The parameter N_0 controls the density of the coarsest scale. On relatively small point clouds it can be set close to k, ensuring that coarse nodes remain well-connected. For point clouds with highly variable size, such as S3DIS or KITTI, it is often more convenient to specify a target density λ_0 and compute

$$N_0 = N\frac{\lambda_0}{\lambda}, \tag{8}$$

where λ is the density of the original point cloud.

The parameter Q controls the relative prominence of coarse and fine scales. Small values ($Q \approx 2$) produce a smooth distribution of scales, while larger values ($Q \approx 8$) allocate many more nodes to the finest scales.

For a point cloud with N nodes, the maximum scale is approximately

$$S = \log_Q \left(\frac{N(Q-1)}{N_0} \right) - 1. \tag{9}$$

The maximum scale determines how many convolutional layers are required for information to travel between the finest and coarsest nodes of the graph.

B Configurations and Additional Benchmark Results

This section reports configuration details and benchmark results. The advantages of Edge Soup connectivity are apparent on tasks with extreme density variation, such as the outdoor LiDAR segmentation. Nevertheless, Edge Soup remains competitive on conventional classification, part-segmentation, and indoor-scene benchmarks, sometimes approaching the performance of a comparable U-Net (DeLA Yang et al. (2024)) despite using fewer parameters and a simpler architecture.

An overview of the hyperparameters used for different tasks is shown in Table 6. These experiments use the architecture described in Section 4 and follow the training procedures of DeLA where applicable.

B.1 Object Classification

On small classification tasks, DeLA uses progressive pooling on points chosen by Furthest Point Sampling (FPS), before globally max-pooling. Our model applies 4 DeLA blocks in series to the Soup graph. Each DeLA block contains 4 Local Aggregations, so the overall model consists of 1 PointNet++-style spatial encoder convolution and 16 inexpensive aggregation steps, interspersed with point-wise MLPs. This produces features on all nodes, but because these tasks are so small, global pooling is viable, and the same head can be used as in DeLA. During training, point ordering is random, and at test time we use precomputed Furthest Point Sampling (FPS) so that coarse points are well-distributed. No point dropout is applied for these tasks, as this would have been similar to the global dropout corruption in Table 5. Results on ModelNet40 (Wu et al., 2015) and ScanObjectNN (Uy et al., 2019) are shown in Table 7. Edge Soup achieves accuracy comparable to DeLA while using fewer parameters.

B.2 Part Segmentation

On ShapeNet-Part Chang et al. (2015), DeLA uses a four-level U-Net structure. We increase the depth of the sequential model to 6 DeLA blocks (for a total of 1 spatial encoder and 24 aggregations).

Table 7: Classification Results

Method	Params.	ModelNet40		ScanObjectNN	
		OA	mAcc	OA	mAcc
PointNet++ Qi et al. (2017)	1.5 M	91.9	-	77.9	75.4
PointNeXT Qian et al. (2022)	**1.4 M**	93.2 ± 0.1	90.8 ± 0.2	87.7 ± 0.4	85.8 ± 0.6
DGCNN Wang et al. (2019)	5.3 M	92.9	90.2	-	-
KPConv Thomas et al. (2019)	14.3 M	92.9	-	-	-
PTv3 Wu et al. (2024)	-	-	-	86.4	83.9
PointHR Qiu et al. (2023)	-	93.9	-	-	-
PointMLP Ma et al. (2022)	13.2 M	94.1	91.3	85.4 ± 1.3	83.9 ± 1.5
GPSFormer Wang et al. (2024)	2.4 M	**94.2**	**92.2**	**95.4**	**93.8**
DeLA Yang et al. (2024)	5.3 M	94.0	**92.2**	90.4	89.3
Soup (Ours)	4.0 M	93.6	91.7	90.7	89.8

Table 8: Part Segmentation results on ShapeNet-Part.

Method	Params.	Ins. IoU	Cat. IoU
DGCNN Wang et al. (2019)	1.3 M	85.2	82.3
PointNet++ (Qi et al., 2017)	**1.0 M**	85.1	81.9
PointNeXT-S (C=64) (Qian et al., 2022)	3.7 M	86.9	84.8
PointNeXT-S (C=160) (Qian et al., 2022)	22.5 M	87.0	85.2
PTv1 (Zhao et al., 2021)	7.8 M	86.6	83.7
SPoTr (Park et al., 2023)	1.7 M	87.2	85.4
PointHR (Qiu et al., 2023)	7.4 M	87.2	-
DeLA (Yang et al., 2024)	7.5 M	**87.5**	**86.0**
Soup (Ours)	3.7 M	87.4	85.8

The part-segmentation head of DeLA requires no special adaptation. We find that we can use point-dropout to reduce training costs without loss of accuracy; in 50% of training batches up to 90% of nodes are dropped at random; this is not done for the experiment in Section 4.4. We use the same 10-way voting scheme as DeLA at test time, with a random starting point chosen for Furthest Point Sampling in each vote, to ensure that the same point is not coarse for all votes.

Results on ShapeNet-Part are shown in Table 8. Edge Soup achieves performance comparable to recent hierarchical architectures while using substantially fewer parameters than many competing approaches.

B.3 Indoor Scene Segmentation

We evaluate indoor scene segmentation on S3DIS (Armeni et al., 2016). This dataset is substantially larger than ShapeNet-Part, with individual rooms containing hundreds of thousands of points and a small number of scenes containing several million points. Most methods operate on a voxelized representation of the original point cloud; we follow the same strategy as DeLA. Rather than applying point dropout after voxelization, we randomly vary the voxel size between 0.04 m and 0.2 m; this is not done for the experiment in Section 4.4.

DeLA employs a four-level U-Net hierarchy based on voxel sub-sampling, whereas we construct Edge Soup graphs using randomized ordering. The model is similar to that used for ShapeNet-Part, with the head adapted to match DeLA on this task and channel count reduced to manage memory requirements. Gradient accumulation is used to match the effective batch size of DeLA.

Results are shown in Table 9. While accuracy is high overall, the Soup model does not match DeLA on this task. In contrast to the gains observed on datasets with strong variation in point density and geometric scale, S3DIS consists of densely sampled indoor scenes with comparatively regular structure after voxelization. This suggests that the advantages of Edge Soup are most pronounced when scale variation is severe, whereas conventional hierarchical discretizations remain highly effective in more uniformly sampled settings. As our primary focus is on less regularly sampled data,

Table 9: Indoor Scene Segmentation results on S3DIS Area5.

Method	Params.	mIoU
PointNet++ Qi et al. (2017)	**1.0 M**	53.5
PointNeXT Qian et al. (2022)	41.6 M	71.1
DGCNN Wang et al. (2019)	-	56.1
KPConv Thomas et al. (2019)	14.9 M	67.1
PTv1 Zhao et al. (2021)	4.9 M	70.4
PTv2 Wu et al. (2022)	12.8 M	71.6
PTv3 Wu et al. (2024)	124.8 M	73.4
PointHR Qiu et al. (2023)	-	73.2
DeLA Yang et al. (2024)	7.0 M	**74.1**
Soup (Ours)	2.5 M	73.0

we began from the settings used by DeLA and performed only limited tuning on this task; further improvement may be possible.

B.4 Outdoor Scene Segmentation

DeLA does not have configurations for KITTI, nuScenes, or Waymo, so we use the hyperparameters from DeLA's configuration on S3DIS and the loss, data preprocessing, and regularization from PointTransformerV3 Wu et al. (2024). To manage memory use during training, we reduce model depth and width, use gradient accumulation, and incorporate gradient checkpointing. Instead of dropout we vary the fine voxel size between 0.05 and 0.25; this is not done for the experiment in Section 4.4.

On these tasks we find that it is beneficial to use Inverse Density Sampling to choose the ordering used to assign weights. The effect of this sampling scheme is to place fewer coarser nodes in denser areas of the scan and fewer fine nodes in sparse areas, ensuring that the coarsest scale has similar edge lengths everywhere. This approach increases construction costs because it requires an additional kNN query (over a subset of nodes) to estimate local density as well as an importance sampling step. We justify this cost with Table 10, which shows that the sampling scheme improves accuracy substantially and reliably over random ordering.

Table 10: Soup Ordering vs. Accuracy (%mIoU)

Ordering	nuScenes Val.
Random	77.1 ± 0.5
Inv. Density	**77.6 ± 0.3**

For our best results on KITTI and nuScenes, we increased the number of epochs from 50 to 100; Waymo was only trained for 25 epochs. Results on these tasks are presented in Table 2.