STAG: Smart Tools and Applications in Graphics (2025)
F. Maggioli, C. Romanengo, D. Cabiddu, and D. Giorgi (Editors)

Algorithms in Geometric Deep learning and 3D AI: Theoretical Survey

V. K. Katturu[1] P. K. Thind[2]
[1]Università di Pisa, Italy
[2]Università degli Studi di Roma "La Sapienza", Italy

Abstract

The study of shapes and geometric representations has long been central to Artificial Intelligence (AI). Early neural networks were limited to Euclidean domains such as images and sequences. The first extensions to non-Euclidean structures appeared in the 1990s and 2000s with recursive neural networks for hierarchical data and random walk–based graph methods. A major step forward came with spectral graph convolutional networks, which introduced convolution in the Fourier domain but faced scalability issues. Spatial methods later enabled more practical graph neural networks (GNNs). In parallel, 3D vision advanced with point cloud models such as PointNet and DGCNN, and mesh-based approaches like Geodesic CNN and MeshCNN, driving progress in classification, segmentation, and reconstruction. As algorithms in geometric deep learning and 3D AI expand, the field has grown both powerful and complex. This paper categorizes major algorithmic families, surveys key datasets across Euclidean and non-Euclidean domains, and highlights emerging advances and open research challenges.

CCS Concepts

• ***Computing methodologies*** → ***Feature selection; Neural networks;*** *Rule learning; Latent variable models; Mixture models;*

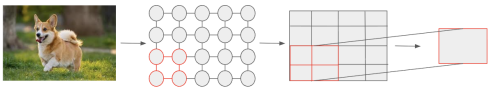
1. Introduction

Although the history of Geometric Deep Learning (GDL) is relatively short, its foundations are rooted in the long-standing progress of deep learning, particularly convolutional neural networks (CNNs). GDL research can be broadly divided into two major directions: methods for processing graph-structured data and those for manifold data.

Work on graph neural networks (GNNs) began as early as 2005, when [GMS05] first proposed a neural architecture capable of handling directed and undirected graphs. This was followed by defining GNNs as functions that map nodes and graphs into Euclidean space and introduced supervised learning algorithms for parameter estimation [SGT*09]. Subsequent advances extended convolutional operations to graphs, including spectral CNNs [BZSL14], diffusion-based approaches [AT16], and localized neighborhood extraction [NAK16]. Key milestones included the introduction of ChebNet [DBV17], the simplified Graph Convolutional Network (GCN) [KW17], and CayleyNet [LMBB19], which together laid the groundwork for modern spectral and spatial GNNs.

In parallel, research on manifold learning advanced techniques for analyzing 3D geometric data. Two early strategies dominated: voxel-based volumetric CNNs, which represent 3D objects as dense grids processed by 3D convolutions, and multi-view CNNs, which capture object geometry through multiple 2D projections. These approaches, while effective, suffered from resolution limitations and inefficiencies. A significant breakthrough came in 2015, when [MBBV15] proposed the Geodesic CNN (GCNN), extending convolutional operations directly to non-Euclidean manifolds [MBBV18]. This was followed by anisotropic CNNs [BMRB16], spectral-domain parameterizations such as SyncSpecCNN [YSGG16], and localized spectral CNNs (LSCNN) [BMM*15], all of which sought to generalize the convolution paradigm to intrinsic geometric domains.

Together, these developments established the foundations of Geometric Deep Learning as a unifying framework that extends deep learning beyond Euclidean domains [BBL*17], fueling its rapid expansion across scientific and engineering disciplines [CYHH20].



Euclidean Data

Figure 1: *Example of Euclidean data: an image represented on a regular grid, where convolution operations exploit the fixed neighborhood structure of pixels.*

While convolutional operations on images are intuitively defined through sliding windows over a regular Euclidean grid, extending this notion to non-Euclidean domains such as meshes or manifolds

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.2312/stag.20251326

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

is far less straightforward. In the image domain Figure 1, the convolutional kernel moves across a uniform lattice with well-defined local neighborhoods and translation invariance. However, on curved or irregular geometries—such as a sphere—there is no globally consistent notion of translation or fixed neighborhood structure. Consequently, defining an analogous moving window on such surfaces requires additional geometric considerations, such as local tangent parameterizations or intrinsic coordinate systems, to preserve consistency across the domain. This challenge motivates the development of GDL, which extends deep learning techniques to non-Euclidean domains.

The past decade has witnessed remarkable advances in GDL,landmark architectures such as Graph Convolutional Networks (GCNs), PointNet and PointNet++ [LB95], Special Euclidean group in 3D Equivariant Transformers (SE(3)-Transformers), and volumetric 3D CNNs have become foundational in extending deep learning to these settings.

Despite such progress, a critical gap remains between accuracy-driven research and efficiency-driven deployment. While state-of-the-art models achieve high performance across tasks like molecular property prediction [CYHH20], social network analysis [GZL*23], recommender systems, and 3D vision, their computational costs—latency, memory, and scalability—limit real-world applicability. For example, drug discovery pipelines require rapid molecular screening, autonomous driving needs real-time 3D perception, and social platforms demand large-scale, low-latency graph analytics. Existing GDL models are often too resource-intensive for such deployment, exposing a mismatch between research benchmarks and industry demands .

This theoretical paper is thus timely and motivated by the growing need for efficient, deployable GDL methods. While several surveys have discussed the theoretical background and algorithmic families of GDL, comparatively little focus has been placed on efficiency trade-offs. Our aim is to bridge this gap by offering a comprehensive perspective on both accuracy and practicality. In doing so, we highlight real-world case studies where algorithmic efficiency is as critical as predictive accuracy, demonstrating how design choices impact deployment feasibility.

This theoretical paper is structured to move from foundational concepts toward deployment-focused perspectives. Section 2 introduces the background and scope of Geometric Deep Learning, covering its main modalities—graphs, point clouds, meshes, and equivariant networks—alongside representative models such as GCN, PointNet, MeshCNN, and SE(3)-Transformer. It also defines the scope of our review, emphasizing three complementary perspectives on efficiency: algorithmic approaches, model compression, and systems-level optimizations. Section 3 then formalizes the metrics and benchmarks for efficiency, including latency, throughput, memory, floating-point operations (FLOPs), and energy, and reviews the most widely used datasets across graphs, molecules, 3D vision, and physics.

The subsequent sections analyze the methods and challenges in achieving efficient GDL. Section 4 surveys algorithmic efficiency techniques such as neighborhood sampling, sparse models, efficient attention, and approximate equivariance. Section 5 reviews model compression strategies including pruning, quantization, and knowledge distillation. Section 6 turns to hardware and systems-level optimizations, highlighting distributed frameworks, out-of-core processing, compiler accelerations, and specialized hardware. Section 7 presents industry case studies in domains such as autonomous driving, drug discovery, and Earth observation, where efficiency bottlenecks are critical. Section 8 discusses open challenges and future research directions, and Section 9 concludes by underscoring the need to treat efficiency as a first-class goal in GDL.

2. Background

GDL has emerged as a unifying framework for learning over data with non-Euclidean structure, such as graphs, point clouds, manifolds, and meshes. Unlike classical deep learning on grid-like data (images, audio, text), GDL methods operate on irregular domains where notions of locality, neighborhood, and symmetry must be explicitly defined. The goal of this section is to introduce the major families of GDL models, highlight canonical architectures, and frame the scope of this survey around efficiency-driven methods. We particularly emphasize three levels of optimization: architectural designs, model compression, and systems/hardware accelerations.

2.1. Spectral Convolution-based Graph Neural Networks

Before the development of spatial message-passing formulations, GNNs were initially defined in the spectral domain, inspired by the theory of graph signal processing. In this framework, convolution is formulated as a multiplication in the graph Fourier domain, enabling frequency-domain filtering of node features. Given a graph $G = (V, E)$ with a normalized graph Laplacian $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, let $L = U \Lambda U^{\top}$ denote its eigen decomposition, where U contains the eigenvectors (graph Fourier bases) and Λ is a diagonal matrix of eigenvalues. The spectral graph convolution between a signal x and a filter g_θ is then defined as

$$g_\theta * x = U\, g_\theta(\Lambda)\, U^{\top} x, \tag{1}$$

In 1 where $g_\theta(\Lambda)$ represents a learnable spectral filter parameterized over the eigenvalues [BZSL14]. While theoretically elegant, direct computation of the eigen decomposition is computationally prohibitive for large graphs. To address this, Chebyshev polynomial approximations were proposed to localize spectral filters in the vertex domain [DBV17], leading to scalable formulations such as the first-order approximation introduced in the GCN [KW17]. These spectral methods established the foundation for modern geometric deep learning by bridging graph signal processing and neural network architectures.

2.2. Graph-Based Models

Graphs are a fundamental data structure for representing complex relational systems in the real world. Many domains including social networks, protein-protein interaction networks, traffic systems, and knowledge graphs can be naturally modeled as graphs, while even regular Euclidean data such as images and videos can be interpreted as grid-structured graphs [CYHH20]. Let $G = (V, E)$ denote a graph with an adjacency matrix A and node features $X \in R^{n \times d}$.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

To enable message passing, self-loops are added to the adjacency matrix, $\tilde{A} = A + I$, with the corresponding degree matrix $\tilde{D}$, yielding a symmetrically normalized adjacency matrix $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. One of the most influential architectures built on this formulation is the GCN [KW17, DBV16, BZSL14], which generalizes convolution operations to non-Euclidean graph domains by aggregating and transforming neighborhood information through a layer-wise propagation rule defined as

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \tag{2}$$

In 2 where $H^{(l)}$ and $W^{(l)}$ denote the node representations and learnable parameters at layer l, and $\sigma(\cdot)$ is a non-linear activation function such as ReLU. Despite their effectiveness, GCNs face scalability challenges due to iterative neighborhood aggregation and non-linear transformations at each layer. To mitigate this, Linearized Graph Neural Networks (L-GNNs) were proposed to simplify the learning process by removing intermediate non-linearities. Among them, the Simplified Graph Convolution (SGC) [ZK21, GZL*23] pre-computes multiple propagation steps, collapsing the GCN layers into a single linear operation expressed as

$$H = \hat{A}^K XW, \tag{3}$$

In 3 where K denotes the number of propagation steps. This linear formulation reduces training to a regression problem on pre-smoothed features, substantially lowering computational and memory costs while maintaining competitive accuracy on large-scale graph benchmarks.

2.3. Graph Sample and Aggregate

Graph Sample and Aggregate (GraphSAGE) is an inductive graph neural network framework that learns node embeddings by sampling and aggregating features from local neighborhoods, enabling scalable representation learning on large graphs and generalization to unseen nodes or graphs. An inductive framework learns a transferable aggregation function that can infer representations for new nodes or graphs without retraining. Unlike transductive models such as GCNs, GraphSAGE does not require access to the full graph during training, making it well suited for dynamic graphs and applications such as recommendation systems and biological networks. By employing neighborhood sampling and aggregation functions (e.g., mean, LSTM, and pooling), GraphSAGE achieves scalable and flexible graph representation learning.It balances efficiency with expressive power while avoiding the high computational costs of full-graph convolution methods [HYL18] Instead of using the full adjacency matrix, GraphSAGE inductive samples a fixed-size neighborhood for each node v. The layer-wise update is:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \mathrm{CONCAT}\left(h_v^{(l)}, \mathrm{AGG}_{u\in\mathcal{N}(v)} h_u^{(l)}\right)\right), \tag{4}$$

In 4 where $\mathcal{N}(v)$ denotes the sampled neighbors of node v, and $\mathrm{AGG}(\cdot)$ is a permutation-invariant aggregator (mean, max-pooling, or an LSTM-based aggregator). By sampling only a fixed-size neighborhood (e.g., $|\mathcal{N}(v)| = 25$), GraphSAGE enables mini-batch training with computational complexity $\mathcal{O}(kd)$ per node, where k is the sample size.

2.4. Graph Attention Network

Graph Attention Networks (GATs) extend graph neural networks by incorporating an attention mechanism that learns the relative importance of neighboring nodes when aggregating information, allowing the model to focus on the most relevant parts of a graph's structure [VCC*18]. Unlike fixed normalization in GCNs, GATs dynamically compute attention coefficients, enabling better handling of heterogeneous graphs and alleviating issues of noisy neighbors. This flexibility has shown strong performance in tasks such as node classification, link prediction, and molecular property prediction, where capturing fine-grained local dependencies is crucial [LBH15]. GATs introduce attention coefficients that weigh neighbor contributions. The update rule for node v is:

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} \alpha_{vu}^{(l)} W^{(l)} h_u^{(l)}\right), \tag{5}$$

In 5 where the attention coefficient $\alpha_{vu}^{(l)}$ is computed as

$$\alpha_{vu}^{(l)} = \frac{\exp\left(\mathrm{LeakyReLU}\left(a^\top \left[Wh_v^{(l)} \,\|\, Wh_u^{(l)}\right]\right)\right)}{\sum_{k\in\mathcal{N}(v)} \exp\left(\mathrm{LeakyReLU}\left(a^\top \left[Wh_v^{(l)} \,\|\, Wh_k^{(l)}\right]\right)\right)}, \tag{6}$$

In 6 with a a learnable attention vector, $\|$ denoting concatenation, and $\alpha_{vu}^{(l)}$ the normalized attention score over neighbors. The complexity is $\mathcal{O}(|E|d)$ for single-head attention (or $\mathcal{O}(K|E|d)$ with K heads).

2.5. Point Cloud Models

Point cloud learning focuses on unordered sets of 3D points, commonly arising from LiDAR scans or 3D vision datasets. The seminal PointNet architecture [QYSG17], 201 processes each point independently with shared multilayer perceptrons (MLPs), followed by symmetric pooling (max/mean), guaranteeing permutation invariance. However, its global pooling scheme fails to capture local structures. To address locality, PointNet++ [QYSG17] [TQD*19] introduced hierarchical neighborhood grouping, recursively applying PointNet on progressively larger local regions. This design captures fine-grained details while maintaining efficiency through subsampling. PointNet processes each point independently with a shared-weight MLP and aggregates the features via a symmetric pooling function. The formulation is:

$$h_i = f_\theta(x_i), \qquad z = \gamma\left(\{h_i\}_{i=1}^N\right), \tag{7}$$

In 7 where $x_i \in R^3$ is the input point, f_θ is an MLP with shared weights applied to each point, and $\gamma(\cdot)$ is a symmetric pooling operator (e.g., max or average) to ensure permutation invariance across the set of N points. The complexity is $\mathcal{O}(Nd)$.

2.6. Point Transformer

Point Transformer is a neural network architecture designed for point cloud learning that leverages the self-attention mechanism to capture both local and global geometric relationships directly from unordered sets of points. Unlike PointNet or convolution-based methods, Point Transformer introduces a permutation-invariant attention layer that dynamically weighs point-to-point interactions

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

based on spatial positions and feature similarities, enabling it to model complex geometric structures effectively [ZK21]. By combining positional encoding with vector attention, it achieves state-of-the-art performance on 3D shape classification, part segmentation, and scene segmentation benchmarks while maintaining scalability to large point clouds. Point Transformer introduces neighborhood self-attention over point sets. For each point x_i, the update is:

$$y_i = \sum_{j \in \mathcal{N}(i)} \text{softmax}_j\big(q(x_i, x_j)\big) \cdot v(x_j), \tag{8}$$

with query and value functions defined as

$$q(x_i, x_j) = W_q x_i + W_k(x_j - x_i), \qquad v(x_j) = W_v x_j, \tag{9}$$

In 9 where W_q, W_k, W_v are learnable weight matrices and $\mathcal{N}(i)$ denotes the local k-nearest neighbors of x_i. By restricting attention to local neighborhoods, the computational complexity reduces from $\mathcal{O}(N^2)$ (full attention) to $\mathcal{O}(Nk)$.

2.7. Mesh Based Models

Mesh-based deep learning models extend geometric deep learning to data defined on 3D surfaces, where the irregularity of mesh structures poses unique challenges. MeshCNN [HHF*19] introduces edge-based convolution and pooling operations that directly operate on mesh connectivity, enabling tasks such as mesh classification and segmentation without remeshing or voxelization. By treating edges as fundamental units, MeshCNN captures both geometric and topological cues in a learnable way. MeshCNN performs convolutions directly on mesh edges rather than nodes or faces. For each edge e, the update rule is:

$$h_e^{(l+1)} = \sigma\Big(W^{(l)} \cdot \text{CONCAT}\Big(h_{e_1}^{(l)}, h_{e_2}^{(l)}, h_{e_3}^{(l)}, h_{e_4}^{(l)}\Big)\Big), \tag{10}$$

In 10 where $h_{e_1}^{(l)}, h_{e_2}^{(l)}, h_{e_3}^{(l)}, h_{e_4}^{(l)}$ are the features of the four edges adjacent to e in the mesh topology, and $W^{(l)}$ are learnable weights. The nonlinearity $\sigma(\cdot)$ is typically a ReLU activation.

By exploiting mesh connectivity, MeshCNN learns localized features invariant to mesh deformations, with complexity $\mathcal{O}(|E|d)$, where $|E|$ is the number of edges.

In contrast, DiffusionNet [SACO22] leverages intrinsic differential geometry by applying learned filters in the spectral domain using heat kernel diffusion, making it robust to mesh discretization and deformation. This intrinsic formulation allows DiffusionNet to generalize across meshes with varying connectivity while retaining shape awareness. Together, MeshCNN and DiffusionNet highlight complementary approaches: topology-aware edge processing and geometry-aware diffusion, pushing forward 3D shape analysis and geometric deep learning [HHF*19].

2.8. Equivariant Networks

Equivariant neural networks are designed to respect the symmetries of geometric data, such as rotations and translations, making them highly effective for molecular modeling, physics simulations, and 3D vision tasks. Early approaches to achieving equivariance, such as Tensor Field Networks and SE(3)-Transformers, relied on *spherical harmonics* to represent features as higher-order tensors and ensure rotational equivariance in three-dimensional space [TSK*18, FFA20]. While theoretically elegant, these formulations are computationally expensive due to the need for complex basis transformations. The Equivariant Graph Neural Network (EGNN) [SHW22] addresses this limitation by achieving Euclidean equivariance without using spherical harmonics. Instead, it updates node features and coordinates jointly in a way that preserves E(n) symmetries, enabling scalable learning on particle systems and molecular structures. SE(3)-Transformers [FFA20], in contrast, extend the Transformer architecture to 3D data by incorporating SE(3)-equivariant attention layers that maintain consistent predictions under rotations and translations—properties that are essential for tasks such as protein structure prediction and quantum chemistry. The `e3nn` library [GS22] provides a general-purpose framework for building group-equivariant layers based on spherical harmonics and tensor algebra, allowing researchers to construct custom equivariant architectures for diverse scientific domains.

Together, these models demonstrate how enforcing physical symmetries improves data efficiency, generalization, and interpretability in geometric deep learning. EGNNs extend GNNs to update both node features and coordinates in an equivariant manner. In this formulation, m_{ij} denotes the *message* passed from node j to node i, computed as a function of their features and relative distances, capturing both structural and geometric dependencies. The update rules are:

$$m_{ij} = \phi_m\Big(h_i, h_j, \|x_i - x_j\|^2\Big), \tag{11}$$

$$h_i' = \phi_h\left(h_i, \sum_{j \in \mathcal{N}(i)} m_{ij}\right), \tag{12}$$

$$x_i' = x_i + \sum_{j \in \mathcal{N}(i)} (x_i - x_j)\,\phi_x(m_{ij}), \tag{13}$$

In 11 where h_i are node features, $x_i \in R^3$ are node coordinates, and ϕ_m, ϕ_h, and ϕ_x are learnable multilayer perceptrons (MLPs) in 11.

This formulation preserves $E(3)$-equivariance, ensuring that predictions remain consistent under 3D rotations and translations, with computational complexity $\mathcal{O}(|E|d)$.

3. Metrics and Benchmarks for Efficiency

When assessing efficiency in geometric deep learning, several hardware and algorithmic metrics are commonly reported. Latency measures the time to process a single input or batch, reflecting suitability for real-time or interactive applications. Throughput quantifies how many samples per second a model can process, which is crucial for large-scale training and inference. Memory footprint captures the peak GPU/CPU memory required during execution, often the bottleneck in large graphs or high-resolution 3D data. FLOPs provide a hardware-agnostic measure of computational complexity, allowing comparisons across architectures. Finally, energy consumption is increasingly important in sustainable AI, reflecting both training cost and deployment feasibility. Because models often trade off accuracy for efficiency, evaluation naturally involves Pareto frontiers, where one compares architectures

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

that dominate others in both accuracy and speed versus those that sacrifice performance for efficiency gains.

3.1. Datasets

Efficiency must also be studied in the context of diverse benchmark datasets. In graph learning, large-scale collections such as OGB (Open Graph Benchmark), Reddit, Amazon product graphs, and citation networks (Cora, PubMed, Citeseer) provide canonical evaluation settings. For molecular learning, datasets like QM9 and MoleculeNet capture both small molecule properties and broader chemical benchmarks. In 3D vision, ModelNet40, ShapeNet, S3DIS (indoor scenes), and autonomous driving datasets like nuScenes and KITTI are widely used. In physics and simulation domains, PDE benchmarks such as Navier–Stokes solvers and CFD mesh datasets enable evaluation of equivariant and physics-informed networks. Each dataset stresses different bottlenecks—memory in large graphs, latency in real-time perception, throughput in molecular screening—making them essential for comprehensive efficiency analysis. [HFR*20] [HYL17] [WSK*15] [CFG*15] [GLU12] [ASZ*16]

4. Algorithmic Efficiency

Efficiency in geometric deep learning has been primarily driven by architectural innovations that reduce neighborhood explosion, limit computational redundancy, and approximate costly operations. Here we group representative methods into four broad categories, highlighting their impact on scalability, latency, and accuracy.

4.1. Sampling and Neighborhood Reduction

Graph neural networks often suffer from the "neighbor explosion" problem, where message passing over multiple layers leads to exponential growth in receptive fields and computational cost. Sampling-based approaches mitigate this challenge by limiting the number of neighbors involved in aggregation. GraphSAGE [HYL17] introduces fixed-size neighborhood sampling, enabling efficient stochastic training on large-scale graphs. PinSAGE [LFXP22] extends this idea to industrial recommender systems by combining random walks with importance-based neighbor sampling, scaling GNNs to graphs with billions of nodes. Cluster-GCN [CLS*19] partitions the graph into densely connected clusters and performs mini-batch training on these subgraphs, reducing cross-partition dependencies and improving memory locality. GraphSAINT [ZZS*20] introduces stochastic subgraph sampling with variance reduction techniques, further improving efficiency and convergence stability. These methods substantially reduce memory cost and latency, albeit at the expense of potential variance in accuracy due to approximation.

4.2. Sparse and Linearized Methods

A second family of approaches eliminates repeated message passing altogether by collapsing propagation steps. Simple Graph Convolutions (SGC) [WSK*15] precompute multi-step neighborhood diffusion and remove nonlinearities, reducing training to a linear classifier. SIGN generalizes this idea by precomputing multiple adjacency powers and concatenating them as features. PPRGo [BGP*20] leverages Personalized PageRank approximations to precompute sparse embeddings. These methods dramatically reduce latency, since training is decoupled from graph propagation, but can sacrifice expressivity compared to full GNNs.

4.3. Efficient Attention Mechanisms

Transformer-style models have recently been applied to graphs and 3D data but face quadratic costs in attention. To address this, Graphormer [YCL*21] encodes structural information into sparse attention, while SAN [DHI*21] and GPS combine local message passing with global sparse attention layers. In the point cloud domain, the Point Transformer [Z*20] restricts self-attention to local neighborhoods, reducing complexity from These approaches achieve strong accuracy, often outperforming GCN variants, but their scalability remains bounded by neighborhood search and memory demands.

4.4. Approximate Equivariance

Finally, efficiency is central in symmetry-aware GDL. EGNN [SHW21] provides linear-time E(n)-equivariant message passing using simple relative distances, offering scalability to larger graphs. By contrast, the SE(3)-Transformer [FFA20] enforces strict equivariance using spherical harmonics and tensor products, achieving higher precision but at significantly higher computational cost. Thus, equivariant models highlight a trade-off between speed (EGNN) and accuracy/expressivity (SE(3)-Transformer).

5. Model Compression Techniques

Model compression techniques aim to reduce the computational and memory footprint of geometric deep learning (GDL) models while preserving most of their predictive performance. This section reviews three dominant directions—pruning, quantization, and distillation—and highlights trade-offs across accuracy, latency, and deployment feasibility.

5.1. Pruning

Pruning removes redundant parameters, nodes, or structures to yield smaller and faster models. *Unstructured pruning* zeros out individual weights or edges, achieving high sparsity ratios but often resulting in irregular computation patterns that commodity hardware cannot efficiently exploit. In contrast, *structured pruning* removes entire channels, attention heads, or neighbor groups, leading to deployment-friendlier speedups since the resulting kernels remain SIMD- and BLAS-compatible. In the context of GNNs, structured pruning often targets nodes or edges with low centrality or marginal contribution to prediction [T*20], while in 3D CNNs and point cloud models pruning is typically applied to convolutional filters or backbone blocks [LFXP22]. Structured pruning generally provides more reliable latency gains than unstructured sparsity at moderate pruning levels, though overly aggressive pruning can destabilize message passing and lead to sharp drops in accuracy.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

Table 1: *Efficiency metrics commonly used in GDL evaluation.*

Metric	Definition / Measurement	Why it matters
Latency	Time to process one input (ms/sample).	Needed for real-time applications.
Throughput	Samples/sec processed.	Important for scaling and deployment efficiency.
Memory	Peak GPU/CPU usage.	Limits feasible graph/mesh resolution.
FLOPs	Floating-point operations per forward pass.	Hardware-independent cost estimate.
Energy	Power usage during training/inference.	Relevant for edge and sustainable AI.

Table 2: *Representative datasets for efficiency evaluation.*

Domain	Dataset	Task / Size	Efficiency Relevance
Graphs	OGB, Reddit, Amazon, Citation	Node classification, link prediction; up to millions of nodes	Stress-test memory and sampling.
Molecules	QM9, MoleculeNet	Property prediction; 100k–1M molecules	Demands high throughput.
3D Vision	ModelNet40, ShapeNet, S3DIS, nuScenes, KITTI	Classification, segmentation, detection	Tests latency in large 3D data.
Physics	Navier–Stokes, CFD meshes	PDE solving, spatio-temporal prediction	Bottlenecked by FLOPs and memory.

5.2. Quantization

Quantization compresses models by reducing the precision of weights and activations. Uniform INT8 quantization is the most common approach, widely supported by inference accelerators such as TensorRT, OpenVINO, and FBGEMM. Mixed-precision quantization tailors bit-widths across layers or nodes, which is particularly important in GNNs where node degree and aggregation variance make some activations more sensitive than others. Recent advances include:

- **A^{2}Q (Aggregation-Aware Quantization)** [F*23a]: adapts bit-widths to neighborhood aggregation statistics to mitigate quantization errors.
- **Degree-aware quantization** [Z*21]: scales precision with node degree distributions, allocating higher precision to high-degree nodes.
- **VQ-GNN (Vector-Quantized GNN)** [Z*22]: applies codebook-based vector quantization to compress graph embeddings and reduce storage overhead.

Equivariant models, such as SE(3)-Transformers, are generally more sensitive to quantization since tensor features encode geometric symmetries. In these settings, careful per-feature or per-channel scaling is often required to avoid degrading equivariance and downstream performance.

5.3. Distillation

Knowledge distillation transfers information from a large *teacher* model into a smaller *student* model, enabling compact architectures to approximate the performance of deeper or wider networks. In geometric deep learning, distillation is particularly effective since neighborhood aggregation patterns and relational structures can be preserved in compressed form. Representative approaches include:

- **TinyGNN** [Z*20]: a student–teacher pipeline designed for billion-scale graphs, achieving competitive accuracy with significantly reduced inference time.
- **PointDistiller** [L*23]: distillation for LiDAR-based 3D object detection, where compact students recover accuracy lost from backbone downsizing.
- Mesh and graph distillation variants (e.g., subgraph-level KD) have also been explored for structured data with promising results.

Distillation not only improves inference efficiency but also often enhances generalization, as the softened supervision signal from the teacher guides the student towards smoother decision boundaries.

5.4. Trade-offs

Compression methods must balance three axes:

1. **Compression ratio**: how much the model size and memory footprint are reduced.
2. **Accuracy retention**: the performance gap relative to the original model.
3. **Latency improvement**: the actual wall-clock speedup, which depends strongly on operator support in deployment backends (e.g., TensorRT, TVM, OpenVINO).

In practice, moderate compression (e.g., 2–4$\times$ smaller models) typically yields minimal accuracy drops ($<$2%), while extreme compression often degrades performance sharply. Structured pruning provides more consistent latency gains than unstructured sparsity, quantization benefits most from accelerator support, and distillation is widely applicable across domains.

6. Hardware & Systems-Level Optimizations

Beyond algorithmic and compression approaches, a major driver of efficiency lies in systems design. Figure 2 summarizes the end-to-end pipeline we discuss in this section, from data ingestion to deployment. As shown in Fig. 2, optimizations appear at four stages: (i) ingestion/streaming, (ii) partitioning and distributed training, (iii) compiler-level acceleration, and (iv) deployment on accelerators. We cover each in turn.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

Table 3: *Comparison of algorithmic efficiency strategies in GDL. Scalability refers to ability to handle large datasets; latency to per-sample inference time; accuracy trade-off indicates potential loss of predictive power.*

Category	Representative Algorithms	Scalability	Latency	Accuracy Trade-off
Sampling / Neighborhood Reduction	GraphSAGE, PinSAGE, Cluster-GCN, GraphSAINT	High (mini-batch friendly, billion-scale)	Low–Medium	May lose accuracy due to sampling variance
Sparse / Linearized	SGC, SIGN, PPRGo	Very High (precomputation avoids recursion)	Very Low	Reduced expressivity; weaker on heterophilic graphs
Efficient Attention	Graphormer, SAN, GPS, Point Transformer	Medium–High (restricted attention, local neighborhoods)	Medium (faster than full attention)	Strong accuracy, but memory still limits very large data
Approx. Equivariance	EGNN (fast), SE(3)-Transformer (precise)	Medium (EGNN linear in edges, SE(3)-T expensive)	EGNN: Low; SE(3)-T: High	EGNN: approximate equivariance; SE(3)-T: higher accuracy but slow

6.1. Distributed Training

Large-scale graph training quickly exceeds the capacity of a single device. Frameworks such as DistDGL [W*19] and PyG GraphBolt [F*23b] implement distributed neighbor sampling, partition-based training, and multi-GPU/multi-node execution. Partitioning strategies (e.g., METIS, random hashing) are crucial for minimizing inter-node communication. While these approaches enable training on billion-edge graphs, challenges remain in balancing workload skew and maintaining fast convergence.

6.2. Out-of-Core Graphs

Real-world graphs (e.g., social, e-commerce, financial) often exceed GPU memory. Out-of-core systems such as AliGraph [Z*19] and Amazon Neptune incorporate streaming, caching, and hierarchical storage to support training and inference without full graph loading. PyG GraphBolt has recently introduced GPU-accelerated out-of-core pipelines for mini-batch training. These methods improve memory efficiency but may introduce latency overhead due to host–device transfers.

6.3. Compiler Optimizations

Compiler stacks translate high-level GDL operations into hardware-efficient kernels. TVM [C*18], TensorRT, and OpenVINO provide graph-level operator fusion, kernel auto-tuning, and quantization-aware deployment. For GNNs, specialized passes handle irregular memory access patterns and sparse-dense operations, although compiler maturity is still lower than for CNNs/Transformers. Recent trends include automatic mixed precision and sparsity-aware code generation.

6.4. Specialized Hardware

Finally, deployment on specialized accelerators (e.g., NVIDIA Jetson, NPUs in smartphones, Intel VPUs such as Myriad X, FPGAs) enables real-time inference under strict energy and memory budgets. Hardware–algorithm co-design has shown promise for onboard GDL, such as in robotics and Earth observation, though general-purpose libraries remain scarce. The ecosystem is still fragmented, and portability across heterogeneous accelerators is a key open challenge.

Taken together, the components in Fig. 2 define a practical blueprint: ingest and cache data out-of-core, partition for distributed sampling, train across GPUs/hosts, then compile and quantize for the target device.

7. Case Studies in Industry

While geometric deep learning has shown strong accuracy in benchmarks, deployment in industry requires satisfying strict latency, memory, and energy budgets. Table 4 summarizes three representative domains where efficiency determines adoption.

Autonomous driving systems process high-resolution LiDAR point clouds and camera feeds at frame rates of 10–20Hz. Models such as PointNet++ and KPConv achieve strong segmentation accuracy, but their computational cost can exceed real-time budgets. Efficiency techniques include sparse convolutions, voxelization, and lightweight distillation. Recent edge deployments target $<$10ms per LiDAR frame on embedded GPUs (e.g., NVIDIA Xavier, Orin).

Molecular property prediction relies on graph neural networks (GNNs) to screen billions of candidate molecules. Here, throughput dominates over single-sample latency: an inference engine must process millions of graphs per second to be practical in virtual screening pipelines. Frameworks such as DistDGL and graph batching optimizations are commonly applied. Precision trade-offs (e.g., INT8 quantization) are acceptable as long as ranking accuracy is preserved.

Resource-constrained platforms such as satellites and drones require onboard inference to reduce downlink bandwidth. Geometric models are used for tasks such as land cover segmentation, wildfire mapping, and flood detection. Unlike data centers, onboard hardware imposes tight energy (Watt-level) and memory ($<$1GB) budgets. Compiler optimizations (OpenVINO, TensorRT) and pruning/quantization are essential. Latency requirements vary: sub-second for real-time monitoring, minutes for bulk EO processing.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

Table 4: *Representative case studies of efficiency requirements in industry applications of geometric deep learning.*

Domain	Data type	Efficiency bottleneck	Target constraint	Common solutions
Autonomous driving	LiDAR point clouds, RGB images	Real-time segmentation/detection	$<$10ms per frame; low power	Sparse convs, distillation, embedded GPUs (Xavier/Orin)
Drug discovery	Molecular graphs (billions)	Throughput on massive screening	Millions of graphs/sec; acceptable precision trade-offs	Graph batching, distributed inference, INT8 quantization
Earth observation	Multi-/hyper-spectral satellite data	Onboard memory & energy	Watt-level energy, $<$1GB RAM, sub-second to minutes latency	Pruning, quantization, OpenVINO/TensorRT deployment

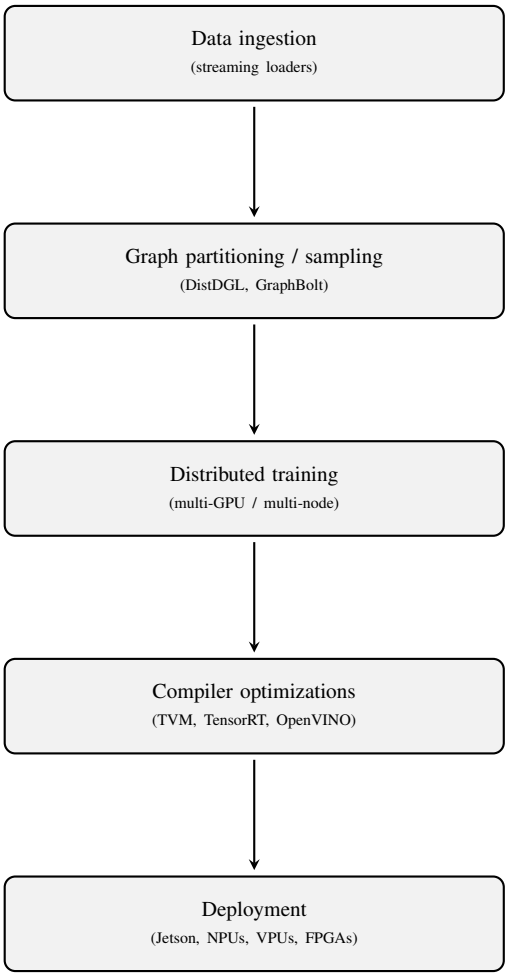


Figure 2: *Top–down schematic for hardware and systems-level optimization in geometric deep learning. Optimizations span ingestion, partitioning, distributed training, compiler acceleration, and deployment.*

8. Future Work and Open Problem

Despite rapid advances, efficiency in geometric deep learning remains far from solved. A persistent tension exists between accuracy and efficiency, often visualized through Pareto frontiers: models that dominate in both speed and accuracy are rare, while most architectures sacrifice one dimension to improve the other. Identifying principled ways to push this frontier outward — without ad-hoc trade-offs — is an open challenge.

Another barrier lies in dynamic and streaming graphs, such as evolving social networks or real-time sensor feeds, where existing sampling and batching methods break down due to continuous updates. Similarly, equivariant networks remain computationally demanding: while EGNN demonstrates efficient approximations, scaling strict SE(3) or higher-order equivariance to millions of nodes or long simulation horizons is still prohibitive. Bridging this gap calls for hybrid methods that approximate symmetry while retaining fidelity.

Finally, the field lacks standardized efficiency benchmarks. While datasets such as OGB and MoleculeNet test accuracy, there is no widely adopted suite for measuring latency, memory, or energy trade-offs across hardware and modalities. Looking forward, hardware–algorithm co-design presents an underexplored direction: aligning pruning, quantization, and sparse operations with emerging accelerators (NPUs, FPGAs, neuromorphic chips) could unlock orders-of-magnitude improvements. A research roadmap should therefore integrate three layers: (1) algorithmic advances that balance accuracy and speed, (2) benchmarks that holistically capture efficiency, and (3) co-design strategies that tightly couple models with hardware constraints.

9. Conclusion

This theoretical paper organized efficiency-oriented geometric deep learning into a taxonomy spanning algorithmic strategies, model compression, systems-level optimizations, and real-world case studies. We highlighted how sampling, linearization, efficient attention, and approximate equivariance tackle core scalability issues, and how compression and hardware optimizations extend these gains.

The central message is that efficiency must become a first-class citizen in GDL research. As applications move from benchmarks to billion-scale deployments, latency, memory, and energy are as critical as accuracy. The next generation of GDL models must therefore be accurate, scalable, and deployable — pushing the Pareto frontier outward through joint advances in algorithms, benchmarks, and hardware co-design.

References

[ASZ*16] ARMENI I., SENER O., ZAMIR A. R., JIANG H., BRILAKIS I., FISCHER M., SAVARESE S.: 3d semantic parsing of large-scale in-

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

door spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). 5

[AT16] ATWOOD J., TOWSLEY D.: Diffusion-convolutional neural networks, 2016. URL: https://arxiv.org/abs/1511.02136, arXiv:1511.02136. 1

[BBL*17] BRONSTEIN M. M., BRUNA J., LECUN Y., SZLAM A., VANDERGHEYNST P.: Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine 34*, 4 (July 2017), 18–42. URL: http://dx.doi.org/10.1109/MSP.2017.2693418, doi:10.1109/msp.2017.2693418. 1

[BGP*20] BOJCHEVSKI A., GASTEIGER J., PEROZZI B., KAPOOR A., BLAIS M., RÓZEMBERCZKI B., LUKASIK M., GÜNNEMANN S.: Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Aug. 2020), KDD '20, ACM, p. 2464–2473. URL: http://dx.doi.org/10.1145/3394486.3403296, doi:10.1145/3394486.3403296. 5

[BMM*15] BOSCAINI D., MASCI J., MELZI S., BRONSTEIN M. M., CASTELLANI U., VANDERGHEYNST P.: Learning Class-specific Descriptors for Deformable Shapes Using Localized Spectral Convolutional Networks. *Computer Graphics Forum* (2015). doi:10.1111/cgf.12693. 1

[BMRB16] BOSCAINI D., MASCI J., RODOLÀ E., BRONSTEIN M. M.: Learning shape correspondence with anisotropic convolutional neural networks, 2016. URL: https://arxiv.org/abs/1605.06437, arXiv:1605.06437. 1

[BZSL14] BRUNA J., ZAREMBA W., SZLAM A., LECUN Y.: Spectral networks and locally connected networks on graphs, 2014. URL: https://arxiv.org/abs/1312.6203, arXiv:1312.6203. 1, 2, 3

[C*18] CHEN T., ET AL.: Tvm: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2018). 7

[CFG*15] CHANG A. X., FUNKHOUSER T., GUIBAS L., HANRAHAN P., HUANG Q.-X., LI Z., SAVARESE S., SAVVA M., SONG S., SU H., ET AL.: Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015). 5

[CLS*19] CHIANG W.-L., LIU X., SI S., LI Y., BENGIO S., HSIEH C.-J.: Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 257–266. 5

[CYHH20] CAO W., YAN Z., HE Z., HE Z.: A comprehensive survey on geometric deep learning. *IEEE Access 8* (2020), 35929–35949. doi:10.1109/ACCESS.2020.2975067. 1, 2

[DBV16] DEFFERRARD M., BRESSON X., VANDERGHEYNST P.: Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2016), NIPS'16, Curran Associates Inc., p. 3844–3852. 3

[DBV17] DEFFERRARD M., BRESSON X., VANDERGHEYNST P.: Convolutional neural networks on graphs with fast localized spectral filtering, 2017. URL: https://arxiv.org/abs/1606.09375, arXiv:1606.09375. 1, 2

[DHI*21] DWIVEDI Y. K., HUGHES L., ISMAGILOVA E., AARTS G., COOMBS C., CRICK T., DUAN Y., DWIVEDI R., EDWARDS J., EIRUG A., GALANOS V., ILAVARASAN P. V., JANSSEN M., JONES P., KAR A. K., KIZGIN H., KRONEMANN B., LAL B., LUCINI B., MEDAGLIA R., LE MEUNIER-FITZHUGH K., LE MEUNIER-FITZHUGH L. C., MISRA S., MOGAJI E., SHARMA S. K., SINGH J. B., RAGHAVAN V., RAMAN R., RANA N. P., SAMOTHRAKIS S., SPENCER J., TAMILMANI K., TUBADJI A., WALTON P., WILLIAMS M. D.: Artificial intelligence (ai): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management 57* (2021), 101994. URL: https://www.sciencedirect.com/science/article/pii/S026840121930917X, doi:https://doi.org/10.1016/j.ijinfomgt.2019.08.002. 5

[F*23a] FANG X., ET AL.: A^2q: Aggregation-aware quantization for graph neural networks. In *International Conference on Learning Representations (ICLR)* (2023). 6

[F*23b] FEY M., ET AL.: Graphbolt: Efficient data loading for graph neural networks. *arXiv preprint arXiv:2302.10316* (2023). 7

[FFA20] FUCHS D., FUCHS L., ABRAMSON R.: *Peer-Assisted Learning Strategies (PALS): A Validated Classwide Program for Improving Reading and Mathematics Performance*. 03 2020, pp. 109–120. doi:10.1007/978-3-030-37285-9_6. 4, 5

[GLU12] GEIGER A., LENZ P., URTASUN R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012). 5

[GMS05] GORI M., MONFARDINI G., SCARSELLI F.: A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (2005), vol. 2, pp. 729–734 vol. 2. doi:10.1109/IJCNN.2005.1555942. 1

[GS22] GEIGER M., SMIDT T.: e3nn: Euclidean neural networks, 2022. URL: https://arxiv.org/abs/2207.09453, arXiv:2207.09453. 4

[GZL*23] GAO C., ZHENG Y., LI N., LI Y., QIN Y., PIAO J., QUAN Y., CHANG J., JIN D., HE X., LI Y.: A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst. 1*, 1 (Mar. 2023). URL: https://doi.org/10.1145/3568022, doi:10.1145/3568022. 2, 3

[HFR*20] HU W., FEY M., REN H., NAKATA M., DONG Y., LESKOVEC J.: Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020). 5

[HHF*19] HANOCKA R., HERTZ A., FISH N., GIRYES R., FLEISHMAN S., COHEN-OR D.: Meshcnn: a network with an edge. *ACM Transactions on Graphics 38*, 4 (July 2019), 1–12. URL: http://dx.doi.org/10.1145/3306346.3322959, doi:10.1145/3306346.3322959. 4

[HYL17] HAMILTON W., YING R., LESKOVEC J.: Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)* (2017). 5

[HYL18] HAMILTON W. L., YING R., LESKOVEC J.: Inductive representation learning on large graphs, 2018. URL: https://arxiv.org/abs/1706.02216, arXiv:1706.02216. 3

[KW17] KIPF T. N., WELLING M.: Semi-supervised classification with graph convolutional networks, 2017. URL: https://arxiv.org/abs/1609.02907, arXiv:1609.02907. 1, 2, 3

[L*23] LI B., ET AL.: Pointdistiller: Structured knowledge distillation towards efficient and compact 3d detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023). 6

[LB95] LECUN Y., BENGIO Y.: Convolutional networks for images, speech, and time-series. 2

[LBH15] LECUN Y., BENGIO Y., HINTON G.: Deep learning. *Nature 521* (05 2015), 436–44. doi:10.1038/nature14539. 3

[LFXP22] LIU Y., FAN T., XIANG S., PAN Y.: Channel pruning for point cloud neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022). 5

[LMBB19] LEVIE R., MONTI F., BRESSON X., BRONSTEIN M. M.: Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing 67*, 1 (2019), 97–109. doi:10.1109/TSP.2018.2879624. 1

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

[MBBV15] MASCI J., BOSCAINI D., BRONSTEIN M. M., VANDERGHEYNST P.: Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)* (2015), pp. 37–45. 1

[MBBV18] MASCI J., BOSCAINI D., BRONSTEIN M. M., VANDERGHEYNST P.: Geodesic convolutional neural networks on riemannian manifolds, 2018. URL: https://arxiv.org/abs/1501.06297, arXiv:1501.06297. 1

[NAK16] NIEPERT M., AHMED M., KUTZKOV K.: Learning convolutional neural networks for graphs, 2016. URL: https://arxiv.org/abs/1605.05273, arXiv:1605.05273. 1

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)* (2017). 3

[SACO22] SHARP N., ATTAIKI S., CRANE K., OVSJANIKOV M.: Diffusionnet: Discretization agnostic learning on surfaces, 2022. URL: https://arxiv.org/abs/2012.00888, arXiv:2012.00888. 4

[SGT*09] SCARSELLI F., GORI M., TSOI A. C., HAGENBUCHNER M., MONFARDINI G.: The graph neural network model. *IEEE Transactions on Neural Networks 20*, 1 (2009), 61–80. doi:10.1109/TNN.2008.2005605. 1

[SHW21] SATORRAS V. G., HOOGEBOOM E., WELLING M.: E(n) equivariant graph neural networks. In *International Conference on Machine Learning (ICML)* (2021). 5

[SHW22] SATORRAS V. G., HOOGEBOOM E., WELLING M.: E(n) equivariant graph neural networks, 2022. URL: https://arxiv.org/abs/2102.09844, arXiv:2102.09844. 4

[T*20] TANG J., ET AL.: Towards scale-free graph neural networks via node importance-based pruning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020). 5

[TQD*19] THOMAS H., QI C. R., DESCHAUD J.-E., MARCOTEGUI B., GOULETTE F., GUIBAS L. J.: Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). 3

[TSK*18] THOMAS N., SMIDT T., KEARNES S., YANG L., LI L., KOHLHOFF K., RILEY P.: Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)* (2018). 4

[VCC*18] VELIČKOVIĆ P., CUCURULL G., CASANOVA A., ROMERO A., LIÒ P., BENGIO Y.: Graph attention networks, 2018. URL: https://arxiv.org/abs/1710.10903, arXiv:1710.10903. 3

[W*19] WANG M., ET AL.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019). 7

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). 5

[YCL*21] YING C., CAI T., LUO S., ZHENG S., KE G., HE D., SHEN Y., LIU T.-Y.: Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)* (2021). 5

[YSGG16] YI L., SU H., GUO X., GUIBAS L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation, 2016. URL: https://arxiv.org/abs/1612.00606, arXiv:1612.00606. 1

[Z*19] ZHU R., ET AL.: Aligraph: A comprehensive graph neural network platform. In *Proceedings of the VLDB Endowment* (2019). 7

[Z*20] ZHANG H., ET AL.: Tinygnn: Learning efficient graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020). 5, 6

[Z*21] ZHOU J., ET AL.: Degree-aware quantization for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)* (2021). 6

[Z*22] ZHANG H., ET AL.: Vq-gnn: A vector-quantized approach for scalable graph representation learning. *arXiv preprint arXiv:2205.10875* (2022). 6

[ZK21] ZHU H., KONIUSZ P.: Graph convolutional network with generalized factorized bilinear aggregation, 2021. URL: https://arxiv.org/abs/2107.11666, arXiv:2107.11666. 3, 4

[ZZS*20] ZENG H., ZHOU H., SRIVASTAVA A., KANNAN R., PRASANNA V.: Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)* (2020). 5

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.