Edge Graph Intelligence: Reciprocally Empowering Edge Networks With Graph Intelligence

Liekang Zeng, *Member, IEEE*, Shengyuan Ye, *Graduate Student Member, IEEE*, Xu Chen, *Senior Member, IEEE*, Xiaoxi Zhang, *Member, IEEE*, Ju Ren, *Senior Member, IEEE*, Jian Tang, *Fellow, IEEE*, Yang Yang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Recent years have witnessed a thriving growth of computing facilities connected at the network edge, cultivating edge networks as a fundamental infrastructure for supporting miscellaneous intelligent services. Meanwhile, Artificial Intelligence (AI) frontiers have extrapolated to the graph domain and promoted Graph Intelligence (GI). Given the inherent relation between graphs and networks, the interdiscipline of graph learning and edge networks, i.e., Edge GI or EGI, has revealed a novel interplay between them – GI aids in optimizing edge networks, while edge networks facilitate GI model deployment. Driven by this delicate closed-loop, EGI is recognized as a promising solution to fully unleash the potential of edge computing power and is garnering growing attention. Nevertheless, research on EGI remains nascent, and there is a soaring demand within both the communications and AI communities for a dedicated venue to share recent advancements. To this end, this paper promotes the concept of EGI, explores its scope and core principles, and conducts a comprehensive survey concerning recent research efforts on this emerging field. Specifically, this paper introduces and discusses: 1) fundamentals of edge computing and graph learning, 2) emerging techniques centering on the closed loop between graph intelligence and edge networks, and 3) open challenges and research opportunities of future EGI. By bridging the gap across communication, networking, and graph learning areas, we believe that this survey can garner increased attention, foster meaningful discussions, and inspire further research ideas in EGI.**

***Index Terms*—Edge computing, edge intelligence, graph learning, artificial intelligence, wireless communication.**

Received 13 June 2024; revised 25 October 2024 and 2 January 2025; accepted 2 January 2025. Date of publication 9 January 2025; date of current version 19 December 2025. This work was supported in part by the Guangdong S&T Programme under Grant 2024B0101020004; in part by the Guangzhou Basic and Applied Basic Research Program under Grant 2024A04J6367, Grant 2023B1515120058, Grant 2024A1515010161, and Grant 2023A1515012982; in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X355; in part by NSFC under Grant 62472460; and in part by the Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province. *(Corresponding authors: Xu Chen; Yang Yang.)*

Liekang Zeng was with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China. He is now with the IoT Thrust and the Research Center for Digital World with Intelligent Things, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China (e-mail: zenglk3@gmail.com).

Shengyuan Ye, Xu Chen, and Xiaoxi Zhang are with the School of Computer Science and Engineering and the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, Guangzhou 510275, China (e-mail: yeshy8@mail2.sysu.edu.cn; chenxu35@mail.sysu.edu.cn; zhangxx89@mail.sysu.edu.cn).

Ju Ren is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: renju@tsinghua.edu.cn).

Jian Tang is with the Midea Group, Shanghai 201702, China (e-mail: tangjian22@midea.com).

Yang Yang is with the IoT Thrust and the Research Center for Digital World with Intelligent Things, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China, also with Peng Cheng Laboratory, Shenzhen 518055, China, and also with Terminus Group, Beijing 100027, China (e-mail: yyiot@hkust-gz.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/COMST.2025.3527561

I. Introduction

EDGE networks are swiftly proliferating. By assembling progressively spreading computing facilities at the network edge, edge networks have hosted ever-increasing amounts of data, storage, and computing resources. They have become a fundamental infrastructure supporting miscellaneous applications like smart industrial manufacturing [1], [2], streaming video analytics [3], [4], and Internet of Robotics and Vehicles [5], [6], etc. As a complementary symmetry of the centralized core network, edge networks locate at the end of the Internet and encompass users in their physical vicinity, allowing for user-centric services with reduced response latency, improved resource efficiency, and enhanced privacy and security. Benefited from these unique architectural superiorities, edge networks have been a vital experimentation arena for advanced communication techniques. They are practically favorable for emerging intelligence services with delay-sensitive, resource-demanding, and privacy-preserved requirements, and have been widely recognized as a promising prospect for bridging the last mile between Artificial Intelligence (AI) and human beings [7], [8].

Meanwhile, AI is also rapidly booming. To fully unleash the potential of big data in diverse forms, recent AI advances have extrapolated representation learning over massive data from Euclidean structure to graph topology, pushing Deep Learning (DL) frontiers to a new stream of models named Graph Neural Network (GNN) [9], [10]. Different from traditional DNN (e.g., CNN, RNN) that typically applies 1D/2D convolutions, GNN introduces graph embedding techniques to digest information from graph relations [11], [12]. Specifically, it applies neighbor aggregation on an input graph iteratively and captures hierarchical patterns through neural network operators from subgraphs of varying sizes. This enables GNNs to abstract and learn the properties of specific vertices, links, or the entire graph, and thus generalize to unobserved graphs. Leveraging such powerful expressiveness, learning with GNN, i.e., Graph Learning (GL), has exhibited superior

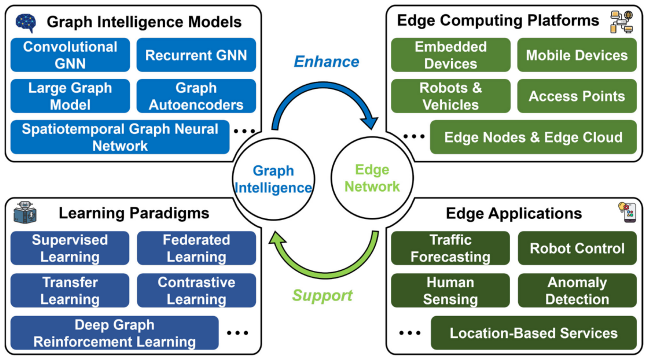


Fig. 1. Illustration of the interplay between GI and edge networks, where GI can be applied as a data-driven tool to optimize edge networks, and conversely, edge networks perform as digital infrastructure to support GI deployment.

graph analysis performance and empowers various graph-related tasks from node classification and link prediction to graph isomorphism and categorization [13], [14].

Given the remarkable success of Graph Intelligence (GI) and edge networks in their respective fields, the inherent connection between graphs and networks impels them to a confluence. As illustrated in Fig. 1, GI provides a vast zoo of empirical learning models (e.g., convolutional and recurrent GNNs, graph autoencoders) as well as various learning paradigms like Transfer Learning (TL) and Reinforcement Learning (RL), allowing advanced learning ability from graph data. Symmetrically, edge networks generally comprise of a rich set of platforms including mobile devices, robots, vehicles, and edge nodes, which host miscellaneous graph-based applications such as traffic forecasting and network resource management. Their bidirectional interaction, where GI enhances and optimizes edge networks and edge networks support and enable GI computation, draws a closed loop with mutual empowerment and nurtures a reciprocal interplay of their integration, namely "**Edge Graph Intelligence**" or "**EGI**" for brevity.

While the term EGI is fresh to come, research and practices have begun early. Since the development of GCN in 2015 [15], GI has increasingly gained popularity in the AI community and ignited a wave of building GNNs over various real-world graphs. Meanwhile, edge networks and edge computing are also rapidly evolving and actively embracing AI since 2019, giving rise to the concept of edge AI or edge intelligence [8], [16], [17]. Currently, the interplay of EGI has attracted growing attention from both the industry and academia and propelled a plethora of innovative optimizations, techniques, and applications at the network edge, e.g., traffic flow forecasting [18], [19], location-based recommendation [20], [21], and vehicle trajectory prediction [22], [23]. As a substantial extension of edge AI, EGI sheds light on its fundamental questions - how deeply edge networks and AI techniques can be fused and how much potential their fusion can shine - and demonstrates its powerful capability through plentiful realistic applications.

In this paper, we discuss in-depth how GI and edge networks are reciprocal to each other, and conduct a comprehensive and concrete survey of the recent research efforts on EGI.

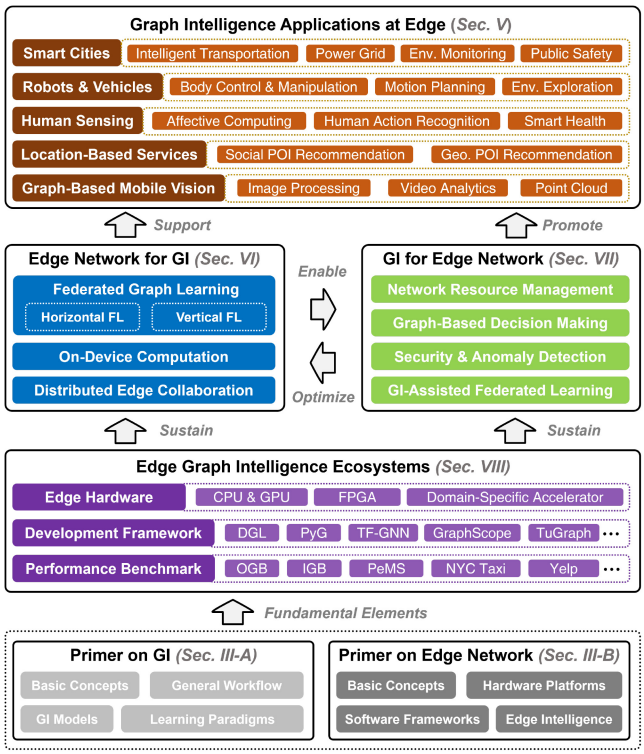


Fig. 2. Outline and conceptual relationships of EGI aspects discussed in this survey: Based on fundamental elements of GI (Section III-A) and edge networks (Section III-B), EGI ecosystems (Section VIII) sustain all stakeholders in the closed loop of edge networks and graph intelligence. Edge Network for GI (Section VI) reviews techniques for supporting edge computation of GI models and GI for Edge Network (Section VII) discusses GI-based optimizations on edge networks. Both of them serve as support for a rich set of EGI applications (Section V).

In particular, centering around the inherently interconnected nature of graphs and networks, this paper reveals the bilateral interplay, for the first time, between GI and edge networks, and provides a concise rating in accordance with their mutually beneficial interactions. In light of the rating, our survey identifies the four primary enablers essential for EGI, as illustrated in Fig. 2:

- Edge applications of GI Models (Section V): Typical application scenarios and use cases for applying GI in edge networks;
- Edge Networks for GI (Section VI): Paradigms of GI model computation, including model training and inference, for GI over edge networks;
- GI for Edge Networks (Section VII): Practical GI-based methods for optimizing edge networks concerning their specific functionalities;
- EGI ecosystems (Section VIII): full-stack infrastructural support for high-performance EGI computation in terms of hardware, software, and benchmarks.

In general, these key enablers can be well accommodated in the closed loop, i.e., "edge for GI" and "GI for edge" as described in Fig. 1, In the "edge for GI" course, edge networks provide physical platforms and software stacks to graph intelligence, serving as infrastructure to support GI models training and inference processes. More specifically, GI models' intensive training workload can be resolved

TABLE I
LIST OF MAIN ABBREVIATIONS

Abbr.	Definition	Abbr.	Definition	Abbr.	Definition
AI	Artificial Intelligence	FPGA	Field Programmable Gate Array	LLM	Large Language Model
AR	Augmented Realtiy	GAE	Graph Autoencoder	LSH	Locality-Sensitive Hash
CL	Contrastive Learning	GAN	Generative Adversarial Network	LSTM	Long Short Term Memory networks
CDN	Content Delivery Network	GAT	Graph Attention Network	ML	Machine Learning
CNN	Convolutional Neural Network	GCN	Graph Convolutional Network	MLP	Multilayer Perceptron
CPS	Cyber-Physical Systems	GFL	GI-Assisted Federated Learning	MOT	Multi Object Tracking
DGRL	Deep Graph Reinforcement Learning	GFM	Graph Foundation Model	NAS	Neural architecture search
D2D	Device-to-Device	GI	Graph Intelligence	NPU	Neural Processing Unit
DAG	Directed Acyclic Graph	GL	Graph Learning	QoS	Quality of Service
DL	Deep Learning	GNN	Graph Neural Networks	POI	Point-of-Interest
DNN	Deep Neural Network	HAR	Human Action Recognition	RecGNN	Recurrent Graph Neural Network
DP	Differential Privacy	HE	Homomorphic Encryption	RL	Reinforcement Learning
DQN	Deep Q-Network	HLS	High-Level Synthesis	RNN	Recurrent Neural Network
DRL	Deep Reinforcement Learning	IID	Independent and IDentically	SDN	Software-Defined Networks
DSA	Domain-Specific Accelerators	IoT	Internet of Things	SLO	Service Level Objective
DT	Digital Twin	IoV	Internet of Vehicles	SpMM	Sparse-dense Matrix Multiplication
EGI	Edge Graph Intelligence	ITS	Intelligent Transportation Systems	STGNN	Spatio-Temporal Graph Neural Network
EHR	Electronic Health Record	KD	Knowledge Distillation	TL	Transfer Learning
EI	Edge Intelligence	KG	Knowledge Graph	VAE	Variational Autoencoder
FER	Facial Expression Recognition	LAN	Local-Area Network	VNF	Virtual Network Function
FGL	Federated Graph Learning	LBS	Location-Based Service	VR	Virtual Reality
FL	Federated Learning	LGM	Large Graph Model	WAN	Wide-Area Network

by means of pools of edge resources (e.g., federated edge learning), and edge inference techniques are developed for deploying and accelerating GI models under resource constraints and SLO requirements. Alternatively, in the "GI for edge" course, GI models with these inference solutions can thereafter be efficiently executed upon edge platforms, which enables miscellaneous graph-based applications and optimizes various aspects of edge networks. Besides reviewing these key enablers, our survey provides fundamental and friendly premiers of GI and edge networks that assume no prior knowledge of GI or edge computing. We also discuss various open challenges and research directions toward future EGI, encouraging both AI and communications communities to advance EGI for a broader range of people.

In summary, this paper makes the following key contributions.

- A brief tutorial on the primer of EGI key components that explains basic and necessary concepts about GI and edge networks.
- We submit an in-depth discussion on EGI advantages as well as an EGI rating taxonomy that can be used as a criterion to identify the advancement of EGI systems.
- We conduct a comprehensive survey on various aspects of EGI, including edge applications of GI, edge networks for GI, GI for edge networks, and EGI ecosystems.
- We discuss open challenges and research directions of EGI, acted as a reference for future EGI innovations.

The rest of this paper is organized as follows: First, Sections III-A and III-B briefly review the primers of graph intelligence and edge computing networks, respectively. Next, the subsequent sections introduce research efforts with respect to the four enablers: GI applications at Edge (Section V), edge networks for GI (Section VI), GI for edge networks (Section VII), and EGI ecosystems (Section VIII). Finally, Section IX discusses open challenges and future research opportunities of EGI and Section X concludes. Table I lists the main abbreviations used in this survey.

II. RELATED SURVEY

While EGI intersects edge networks and GI, its investigation is still narrowly limited to unilateral dimensions.

On the GI side, a volume of literature has reviewed the general landscape of GL, providing insights on model taxonomy [10], capability boundaries [13], and practical tutorials [14]. As flourishing attention is attracted, publications relevant to GI topics explode. Following this trend, some researchers begin to deliver reviews concentrated on specific types of models, such as GAE [24], [25], STGNN [26], [27] and DGRL [28], [29]. Surveys on certain scenarios are also released. For instance, some researchers review the use of GI in traffic domain [30], [31], [32], discussing how traffic networks can be constructed as graphs and how traffic patterns can be resolved with GI; several surveys target power grids, where the electrical network are naturally graphs [33], [34], [35]. Nevertheless, these investigations either center on graph learning landscapes with limited discussion about their roles in edge networks [9], [10], [13], or focused particularly on applying GI techniques on some specific edge scenarios, yet ignoring the big picture of general edge networks spectrum. Some recent literature [36], [37], [38] also reviews the progress of GI in the context of IoT and wireless networks, aiming at exploiting GI for optimizing communication channels in IoT services. These works, however, mainly focus on GI applications in their discussed scopes and lack a systematic taxonomy on how GI can be computed in hierarchical edge networks, which is one of the fundamental pillars in EGI's ecosystem.

TABLE II
LIST OF MAIN NOTATIONS IN GRAPH REPRESENTATION LEARNING

Notation	Definition
$\mathcal{G}$	The graph input to the GNN model.
$\mathcal{V}$, V, v	The input graph $\mathcal{G}$ includes a set $\mathcal{V}$ of vertices, where its size is V and v is an arbitrary vertex in $\mathcal{V}$.
$\mathcal{E}$, E, e	The input graph $\mathcal{G}$' includes a set $\mathcal{E}$ of links, where its size is E and e is an arbitrary link in $\mathcal{E}$.
$\mathcal{N}_v^{(k)}$	The set of vertex v's k-hop neighbors in $\mathcal{G}$.
$\mathcal{L}$, L, l	The GNN model's layers set $\mathcal{L}$ is of size L, where l is an arbitrary layer in $\mathcal{L}$.
$h_v^{(l)}$, $h_e^{(l)}$	The representation vector of vertex v and link e of layer l, respectively. When $l = 0$ and $l = L$, they represent the cases of input and output representation vectors of vertex v (link e), respectively.
$\varphi_V^{(l)}$, $\varphi_E^{(l)}$	Aggregation function of vertices and links of layer l.
$\phi_V^{(l)}$, $\phi_E^{(l)}$	Update function of vertices and links of layer l.
$\theta^{(l)}$, $\vartheta^{(l)}$	Sample function and pooling function of layer l.
ψ	Readout function.
y	Global output vector.

On the edge AI side, surveys on the broader edge computing topics have been carried out for years, pertaining to visions and challenges [7], systems and tools [39], communication and IoT perspectives [40], [41], [42], etc. With the popularization of AI in edge networks, many researchers turned their attention to the potential and future directions of edge AI. For instance, Zhou et al. [8] discuss the motivation and advantages of edge intelligence along with a grading mechanism for edge intelligence by assessing the training and inference on edge platforms. Wang et al. [43] submit a comprehensive taxonomy of edge AI and analyze a set of open challenges toward future edge AI development. Although these surveys have extensively investigated edge intelligence systems, a majority of them [8], [43], [44], [45], [46] center on general AI computation or are dedicated to traditional DL workloads such as CNN or RNN. GI models, which possess distinct capabilities and unique computing characteristics, are much less understood in the edge AI context.

In summary, while the AI and edge computing communities have pushed their investigation to their respective frontiers, a thorough review of EGI, the combination of both lines, is still absent and desires actions.

III. PRIMER ON EDGE GRAPH INTELLIGENCE

Before diving into various aspects of EGI, we briefly review the basic concepts and relevant techniques of GI and edge networks, respectively.

A. Graph Intelligence

As one of the key flywheel actuating the loop within EGI, graph representation learning is devoted to the algorithm side and contributes enhanced ability in graph data processing. Before diving into EGI, this section introduces graph representation learning with respect to its basic concepts, general workflow, and representative models and learning paradigms. For a more comprehensive treatment of GI, concentrated reviews [9], [10], [13], [14] on GL are highly recommended. Table II lists the main notations used in this section.

1) Basic Concepts–Graphs: Graphs are a way to organize data, and with graphs, one can succinctly characterize relationships across scattered data points. The input of GI models, i.e., GNNs, are graphs, which typically contain two types of data. One is the adjacency matrix or adjacency list, which interprets the graph topology, and the other is the feature vectors that describe vertices and edges' actual properties. Formally, an input graph is denoted as $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, with vertices and links collected in $\mathcal{V}$ and $\mathcal{E}$, respectively.

Vertices and Links: Vertices or nodes in a graph can be items, objects, or entities, and are not necessarily homogeneous when constructing a graph. For instance, a location-based knowledge graph can represent its vertices as human users, IoT devices, scenic spots, and any other entities of various types within a specific district. Links are another essential component in graphs that characterizes the relationships between these items, objects, or entities. Note that to avoid misunderstanding, we exclusively use "links" to indicate the connection between vertices in an input graph while leaving "edge" for edge networks. A link can be defined with respect to the two (not necessarily unique) vertices associated with it. For $\mathcal{V} \in \mathcal{G}$ and $\mathcal{E} \in \mathcal{G}$, we denote their size, i.e., the number of vertices and links, as V and E, respectively, and use v and e to index arbitrary vertex and link in them.

Neighbors: Neighbors are ego-networks centering on specific vertices within a graph. For a vertex v, its neighbors cover the vertices directly connected to v and their adjoining links. Note that a vertex's neighbors can be iteratively expanded by considering the neighbors of its neighbors. Formally, given $\mathcal{N}_v^{(k)}$ as vertex v's k-hop neighbors, we have $\mathcal{N}_v^{(k+1)} = \{\mathcal{N}_u^{(1)} | \forall u \in \mathcal{N}_v^{(k)}\}$, where $\mathcal{N}_v^{(1)}$ indicates v's one-hop direct neighbors.

Representation Vectors, Features, and Embeddings: Representation vectors are the numerical vectors associated with vertices and links, and are also referred to as encodings, representations, latent vectors, or high-level feature vectors depending on the context. In this section, we respectively denoted representation vectors by $h_v^{(l)}$ and $h_e^{(l)}$ at the l-th GNN layer. Upon input to the model, the initial representation vectors $h_v^{(0)}$ and $h_e^{(0)}$ are exactly the features attached to vertices and links, which quantify physical properties in specific applications. Extending the above knowledge graph example, the features of a vertex may include the users' age and food preferences, and for a scenic spot, it can be location and popularity. After processing through model layers, the exported representation vectors are embeddings, a form of compressed feature representations of vertices, links, neighbors, or graphs. Embeddings can be viewed as the mappings of original data in latent spaces, which effectively reserve the semantics implicated in the input graph while can be used by downstream models for specific tasks (e.g., vertex classification and link prediction).

Model Output: The final output of GI models depends on the way of processing embeddings, i.e., the readout function. In general, the outcome of GI models can be grouped into

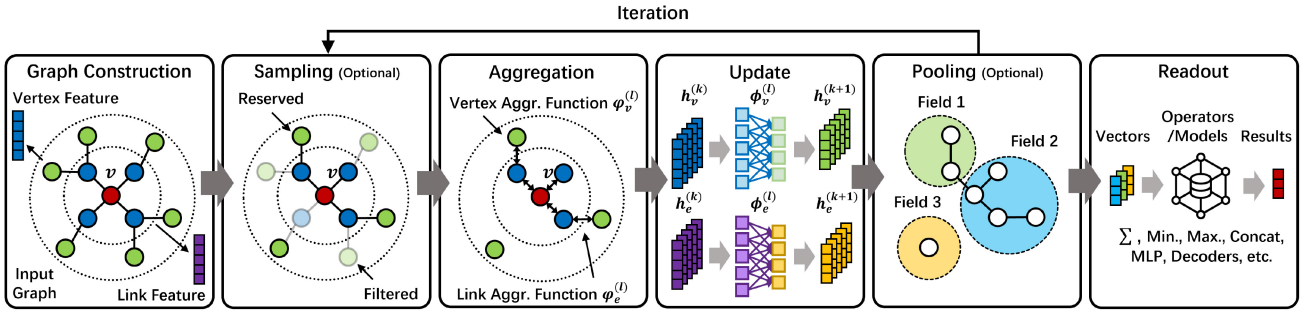


Fig. 3. General workflow of GI models. Given an input graph with feature vectors, a GI model iteratively performs sampling, aggregation, update, and pooling through consecutive model layers. The obtained embeddings will be finally converted to results in an expected form through a readout function.

three types: 1) Vertex-level output, where the result is vertex-wise predictions (e.g., classes, scores) for some dedicated vertices. 2) Link-level output, where the result is link-wise predictions for some dedicated links. 3) Graph-level output, where the results are the prediction of the whole graph (e.g., the operational status of a power grid).

2) General Workflow: A GI model is an algorithm that essentially leverages graph topology to abstract and learn the relationships between vertices and links. It takes an attributed graph as input, and output embeddings or predictions in an application-admitted format. Among versatile GI models, the GNN series is the state-of-the-art genre and is prevalent in various types of edge applications, thus we illustrate the general workflow of GI model based on GNN models. Fig. 3 depicts the general workflow of them.

Preprocessing: The first step serves as an initialization to prepare data in a format aligned with the targeted GI model's requirement. This can be, for example, reorganizing the adjacency matrix in a dense format or the compressed sparse row format and dropout some irrelevant elements from the feature vectors. Since the GI model is often known ahead of runtime, graph preprocessing is usually done offline.

Sampling: With the preprocessed input graph, the GI model dives into iterations. The number of iterations required is exactly the number of layers the GI model possesses. Within each iteration, it first applies sampling on the input graph $\mathcal{G}$ to reduce the computational complexity of subsequent steps. Assuming the sampling function as $\theta^{(l)}$, this step can be formally written in:

$$\mathcal{G}' = \theta^{(l)}(\mathcal{G}). \tag{1}$$

The result of sampling is a sampled graph $\mathcal{G}'$, where $\mathcal{G}' = \langle \mathcal{V}', \mathcal{E}' \rangle$. Note that this is an optional step and inference processes typically deactivate this step for prediction accuracy.

Aggregation: Upon the sampled graph $\mathcal{G}'$, the GI model performs neighbor aggregation where each vertex/link pulls feature vectors from its neighbors. Taking vertices aggregation as example, let $\varphi_V^{(l)}$ be the aggregation function of vertices, we have

$$a_v^{(l)} = \varphi_V^{(l)}\left(h_v^{(l)}, h_u^{(l)}\right), \ \forall v \in \mathcal{V}', \ \forall u \in \mathcal{N}(v), \tag{2}$$

where $\{h_u^{(l)} | u \in \mathcal{N}(v)\}$ collects v's neighboring vertices' representation vectors and $a_v^{(l)}$ is the aggregation result.

Update: The GI model then passes the aggregation $h_v^{(l)}$ through a neural network operator $\phi_V^{(l)}$ to update v's representation vector:

$$h_v^{(l+1)} = \phi_V^{(l)}\left(h_v^{(l)}\right), \ \forall v \in \mathcal{G}', \tag{3}$$

This operator is usually learnable Multi-Layer Perceptron (MLP) and non-linear activations like the sigmoid function. Note that Eq. (2) and Eq. (3) are merely described for vertices for simplicity. Repeating the same procedure to all vertices and links yields the complete representation vectors for the whole graph.

Pooling: Pooling is also an optional step that aims at reducing the original graph to a smaller graph for lower computational complexity. It directly operates the sampled graph with updated representation vectors, pooling fields of graphs:

$$\bar{\mathcal{G}} = \vartheta^{(l)}(\mathcal{G}'). \tag{4}$$

Readout: The above sampling, aggregation, update, and pooling steps will iterate until all model layers are processed, and thereafter generate embeddings of all vertices and links. To attain the desired results demanded by applications, these embeddings are obliged to the final readout step, which applies a pre-defined operator or model to transform embeddings to a global output, as explained in Section III-A1. Given a readout function ψ, the global output vector can be obtained by:

$$y = \psi\left(h_v^{(L)}, h_e^{(L)} | v, e \in \mathcal{G}\right). \tag{5}$$

In summary, executing a GI model can be regarded as processing a collection of operators and neural networks iteratively over a graph, where each iteration, i.e., each model layer, comprises weights that specify the computation of vertices' and links' feature vectors. These weights are learnable via model training, i.e., through the means of backpropagation algorithms such as gradient-based optimization. Specifically, during model training, a GI model with L layers undergoes a forward pass: first transforms input graph through model layers to graph embeddings, and next converts the obtained embeddings into desired results. With the exported result and known labels, the model computes the pre-defined loss function, where the gradient is then backpropagated across the layers, updating the shared weights. This process is carried out iteratively with multiple samples, which are often in batches

until an expected accuracy is attained. For model inference, it directly goes through a forward pass and generates the predictions.

3) Graph Learning Models: There are multifarious GI model variants developed for multifarious applications with multifarious ability requirements. For brevity, here we enumerate several representatives that are commonly adopted in edge scenarios.

Recurrent Graph Neural Network (RecGNN) is a pioneering architecture that builds the conceptual foundation in the field of graph representation learning [9]. They are primely proposed to learn node representations using recurrent neural architectures, integrating a recurrent hidden state and graph signal processing (GSP) to exploit the spatial structural information inherent in graph processes. As a genre of GI models dating back to the "pre-deep-learning" era, RecGNNs have inspired numerous subsequent research such as convolutional variants.

Convolutional Graph Neural Network (ConvGNN) extend convolution operations from grid data to graph data, aggregating the features of vertices and links with their neighbors to generate representations [10], [14]. Contrary to RecGNNs, successive graph convolutional layers are stacked in ConvGNNs to extract hierarchical patterns from subgraphs [15]. Among GNNs, ConvGNNs serve as a foundational component in constructing various advanced GI models [47].

Graph Attention Network (GAT) inherit spatial ConvGNNs by incorporating the attention mechanism into the aggregation functions, which effectively improves the capacity as well as the expressiveness of GI models [48]. The rationale behind this combination is to differentiate the contribution of vertices' neighbors in a learning manner [49]. Based on this, many types of attention mechanisms are derived such as self-attention, gating attention, and semantic-level attention.

Graph Autoencoder (GAE) are unsupervised frameworks that encode vertices, links, or graphs into a latent vector space and reconstruct graph data by decoding their encoded information [13]. GAEs are particularly useful for graph generation tasks, where a GAE model employs graph convolutional layers to compute embeddings for vertices and rebuild the graph adjacency matrix via decoders. Variational GAE goes beyond traditional GAE by transforming vertex embeddings as distribution, where each node's embedding is represented by a mean and variance, allowing the model to capture uncertainties by sampling from these distributions [50].

Spatio-Temporal Graph Neural Network (STGNN) analyze dynamic graphs from both the spatial and temporal dimensions [26], [51]. Each vertex and link in these graphs attaches a feature vector that describes their behaviors within a time window, and STGNNs aim to learn their patterns and predict their changes in the incoming time slots. To extract information from spatial-temporal dependencies, many sequential decision-making approaches are applied, e.g., Long Short-Term Memory (LSTM).

Graph Transformer and Graph Foundation Model Graph transformers explore embracing transformer architecture to the graph domain in pursuit of improved graph modeling ability [52], [53], [54]. Typically, graph transformers incorporate GNN with transformers in dual ways: 1) design tailored positional embedding modules and graph-specific attention matrices, and 2) exploit GNNs as an auxiliary module by combining GNNs into transformer architectures [55]. Following this line, scaling small transformers to huge foundation models leads to Graph Foundation Models (GFMs) [56], [56] and Large Graph Models (LGMs) [57]. Motivated by the success of Large Language Models (LLMs), many researchers believe that utilizing pretraining techniques to resolve massive graph data can breed a comprehensive GI model, which possesses advanced abilities such as in-context graph understanding and versatile graph reasoning [58]. Nevertheless, GFMs and LGMs are still in a very early stage of their development and demand for further exploration.

4) Learning Paradigms: Given a rich zoo of GI models, distinct learning paradigms empower them with distinct abilities for edge applications. We provide several example learning paradigms as follows.

Supervised Learning is one of the most fundamental training paradigms in ML, which requires labeled data to guide the optimization of models [59], [60]. For GNNs, supervised learning enables to capture both local and global graph structures and the latent information of vertices and links. It iteratively optimizes model parameters by minimizing the difference between the model's predictions and the labels, where the performance of the model is evaluated on a separate test set of labeled objects. Supervised GL has found success in various domains such as social network analysis, intelligent transportation, and recommendation systems. However, in many real-world scenarios where labels are unavailable or expensive, supervised learning is constrained and other semi-supervised or unsupervised learning paradigms are introduced.

Transfer Learning (TL) for GI models builds upon the assumption that the learned representations and knowledge from a source graph can be effectively transferred and applied to a targeted graph [61], [62]. Based on this, it exploits knowledge distillation methods to extract the knowledge from one graph data to improve the learning performance on another different but related graph data. By transferring knowledge across graphs, the target graph benefits from the learned representations, enabling improved prediction performance for the targeted graph, even with limited labeled data.

Contrastive Learning (CL) for GI models is a self-supervised learning paradigm that learns meaningful representations by contrasting positive and negative samples, where positive samples are pairs of vertices, links, or subgraphs that are similar or related, and negative ones are the contrary [63], [64], [65]. With these samples, contrastive learning maximizes the similarity between positive samples and minimizes the similarity between negative samples. Various techniques have been applied to enhance contrastive learning on graphs, such as graph augmentation and sample generation with random walks.

Variational Learning extends the idea of variational inference to GNN, introducing a probabilistic framework to learn node or graph-level representations. This technique is mainly

applied in Variational GAEs [50], where each node's embedding is treated as a latent variable sampled from a learned distribution. The variational framework enables the model to capture uncertainties in graph data by learning distributions over node embeddings rather than fixed values. This allows for uncertainty quantification by modeling distributions over node embeddings, improving robustness to noisy data, and preventing overfitting through regularization. This leads to more expressive representations and enables generative tasks like predicting missing links or generating new graph structures [66].

Federated Graph Learning (FGL) FGL applies Federated Learning (FL) on graphs, which allows multiple clients to collaboratively train a GI model without sharing their local data [67], [68], [69]. In particular, each client trains the GI model using its local graph data and shares only the model updates, i.e., gradients or weights, with a central coordinator, and the coordinator aggregates the updates and sends a merged model update back to each client. Graph data are typically distributed across clients, with respect to structural segments, i.e., subgraphs, or feature segments. More details on FGL are discussed in Section VI-A.

Deep Graph Reinforcement Learning (DGRL) DGRL combines GI models with Deep Reinforcement Learning (DRL) techniques for interacting with graph environments [28], [29]. Typically, in DGRL, the GI model is responsible for processing the graph data, extracting features, and capturing the relationships between vertices and links. The DRL component then uses the embeddings computed by the GI model to learn a policy and make decisions, often by taking actions like vertex selection, link insertion/removal, or graph modification. Benefiting from the superior capability of representing graphs, DGRL has become a powerful tool for graph-based sequential decision-making tasks.

B. Edge Networks and Edge computing

1) Basic Concepts: Edge networks have emerged as a pivotal architecture, facilitating the transition from centralized cloud-based core networks to the much more decentralized edge [70], [71], [72]. Fig. 4 illustrates the edge network as a cohesive architecture that integrates the *Edge* and *End* layers, encompassing devices naturally situated closer to end users. Besides traditional network-style topology, edge networks can exhibit versatile organizations such as peer-to-peer connection (e.g., client-server mode) and star-like interactions (e.g., one-to-many subscription).

Specifically, in contrast to the centralized core networks, several salient features distinguish edge networks: (1) *Distributed*: Edge networks are characterized by their geographically dispersed resources, such as edge servers and mobile devices, in stark contrast to the centralized nature of cloud-based data centers. (2) *Heterogeneity*: Edge networks encompass a diverse array of edge devices and servers, each varying in computational capacity, network bandwidth, and hardware architecture, embodying an extremely heterogeneous computing environment. (3) *Resource Constraints*: Unlike cloud-based data centers equipped with high-performance

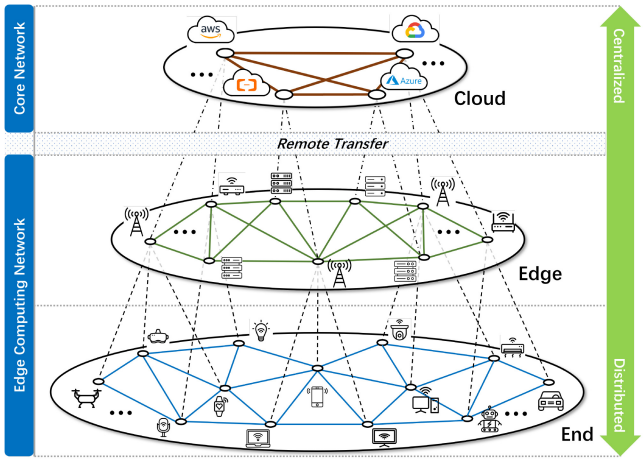


Fig. 4. Architecture overview of cloud-edge-end hierarchy, where distributed edge devices within edge networks serve as infrastructure for graph-intelligent applications and their networked data can be analyzed by graph representation learning models.

accelerators and dedicated ultra-speed links, devices within edge networks often operate under resource limitations, facing constraints in aspects like computational throughput, device-to-device communication bandwidth, and memory capacity. (4) *Dynamic Resources*: The proximity of edge network devices to end users inherently results in greater dynamism in resource availability. Their mobility across various networking domains and multitasking with multiple applications simultaneously contribute to frequent fluctuations in both network and computational resources.

Emerging from the advanced architecture of edge networks, edge computing stands as a new computing paradigm in contrast to cloud computing [8], [43]. Edge computing represents a shift in computational paradigm that brings data processing exponentially closer to the point of data collection and consumption, enabling low-latency and high-bandwidth communication essential for real-time applications [73], [74]. With lower reliance on the core network, edge computing not only alleviates both the stress of centralized cloud and backhaul bandwidth usage but also enhances privacy preservation through localized data processing.

2) Components of Edge Networks: Locating at the periphery of the Internet, edge networks cover a wide spectrum of diverse and heterogeneous platforms. From fragmentation to integration, we enumerate example components in edge networks in five levels as in Fig. 5.

Sensors are devices that capture real-time data from the physical environment, such as image, temperature, motion, and gas composition. This type of device is the most scattered IoT and is thus located at the bottom level in Fig. 5, where several example sensors as shown.

Embedded and Mobile Devices are advanced systems that integrate multiple sensors and MCUs into a single, often portable, framework. Unlike standalone sensors or MCUs that primarily capture or process data, these devices offer comprehensive functionality, including data processing, analysis, and user interaction. Embedded devices are specialized computing units designed for specific functions within larger systems.

Fig. 5. The spectrum of edge networks can be classified into five categories: sensors and micro control units, embedded and mobile devices, robotics and Vehicles, edge servers, and edge cloud.

Robots and Vehicles usually excel in functionality, autonomy, and complexity. While embedded devices are integral to these systems, robotics and vehicles incorporate them into more complex frameworks, designed for higher autonomy and minimal human intervention.

Edge Servers act as micro data centers to deliver services in a way similar to using cloud servers but with a key difference: they are situated closer to the data source. This proximity not only ensures adherence to data localization laws but also significantly reduces data transfer latency.

Edge Cloud services extend cloud computing's convenience to the network's edge, with providers like AWS and Google offering solutions such as AWS Edge Service [75] and Google Distributed Cloud Edge [76]. Edge clouds are usually hosted by micro-data centers comprising edge servers that store, analyze, and process data faster than is possible using a connection to a cloud data center.

3) Software Frameworks for Edge Networks: Software frameworks for edge computing across edge networks are often cross-platform, cross-protocol, language-agnostic, and resource-efficient. EdgeX Foundry [77] is an open-source edge platform that lets users create IoT gateway functionality from edge devices, which acts as a dual transformation engine sending and receiving data to and from cloud and edge applications. Eclipse Kura [78] has taken a similar approach to EdgeX Foundry to provide a platform for developing IoT gateways. Apache Edgent [79] is an open-source stream processing framework for edge computing, enabling developers to process sensor-collected streaming data in real time on edge devices. RedisEdge [80] is a purpose-built, multi-model database for the demanding conditions at the IoT edge, which can ingest millions of writes per second with less than 1ms latency and a very small footprint (less than 5MB). TensorFlow Lite [81] and MNN [82] is an open-source lightweight framework that enables on-device machine learning by helping developers run their AI models on embedded and mobile devices to achieve edge intelligence.

Some software frameworks offer a simulation environment to model and simulate edge network infrastructures and services, providing timely, repeatable, and controllable methods for evaluating the performance of new edge applications and policies prior to actual development. EdgeCloudSim [83] provides a simulation environment based on CloudSim [84] but adds considerable functionality so that it can be efficiently used for edge network scenarios. IFogSim2 [85] is a toolkit for modeling and simulation of resource management techniques in IoT, edge, and fog computing environments. YAFS [86] is a Python-based simulator tool for architectures like Fog Computing ecosystems, facilitating analyses related to resource placement, deployment costs, and network design.

4) Edge Intelligence: AI stands at the forefront of modern technological advancements, enabling computer systems to perform tasks that typically require human intelligence. This encompasses a wide range of activities including learning, decision making, and problem solving. In recent years, deep learning (DL), a branch of machine learning, has become synonymous with AI's progress. Characterized by its use of multi-layered neural networks, DL enables the processing of complex data, driving innovations in areas such as image recognition, natural language processing, and automated decision-making. The marriage of edge computing and AI catalyzes the birth of *edge intelligence* [8], [43].

In examining the advancements in the field of AI, particularly in the realms of DL model training and inference, it becomes clear that edge intelligence holds distinct advantages over traditional cloud intelligence: (1) *Model Training*: In the current AI landscape, there has been a significant shift in data sources from centralized cloud data centers to increasingly ubiquitous edge devices, such as mobile and IoT devices. These methods require the collection of massive amounts of data in the cloud, exerting immense pressure on core networks due to high data traffic. Additionally, this centralized approach strains data center resources, both in terms of storage and data processing capabilities. Edge intelligence offers a strategic solution to this challenge. By facilitating the processing and training of deep learning models directly at the network edge, closer to where the data is generated [87], [88], [89]. (2) *Model Inference*: Upon completion of model training, these models are deployed to provide inference services to end users. Routing user requests to cloud-based systems for inference poses significant privacy concerns, as it could lead to the leakage of sensitive information, such as personal identifiers and location data. Moreover, relying on cloud-based inference services, which depend on the stability of core networks, may introduce significant propagation delays. This aspect is particularly critical for applications requiring real-time response, such as autonomous driving and medical-surgical robots. Edge intelligence, by processing these inference requests locally, not only enhances data privacy but also ensures low-latency,

efficient service delivery, crucial for such time-sensitive applications [90], [91]. EGI can be regarded as a particular genre of edge intelligence concerning GI and is yet in its seed time for exploration and exploitation. In the following sections, we will discuss EGI in detail from different aspects.

IV. Edge Graph Intelligence as a New Edge AI Paradigm

Stemed from edge AI, EGI dives into the fusion of edge networks and GI techniques, serving as a brand-new avenue for the evolution of AI deployment. In what follows, we discuss the benefits of EGI and provide a rating taxonomy of EGI in six levels.

A. Reciprocal Benefits of Edge Graph Intelligence

EGI provides reciprocal benefits in the following aspects.

- *Edge networks for GI.* With the rapid proliferation of mobile and IoT devices, data generated at edge networks have skyrocketed in both quantity and modality (e.g., physical signals, digital audio, and visual content). As predicted by IDC [92], the billions of IoT devices in edge networks are expected to generate over 90 Zettabytes of data in 2025. This naturally provides rich data nurseries and abundant application scenarios for modifying, training, and fine-tuning GI models with real-world data, thus boosting GI models toward higher-degree intelligence. Besides, edge networks as user-nearby infrastructures can effectively enhance GI model computing performance, e.g., edge networks assisted with cloud as the back-end can largely reduce the computing latency, thus boosting the GI performance.
- *GI for edge networks.* Given the rich relational data collected at edge networks, GI enables modern graph analysis for understanding, diagnosing, and optimizing edge networks, which steers the enhancement of network performance such as robustness and Quality of Service (QoS). Applying GI to edge networks thus unlocks their extended capability in securing edge networks from anomalies, developing new graph-based applications, and intelligently serving graph-related tasks.
- *Reciprocal technology advancement.* The combination of GI and edge computing not only carries out mutual empowerment to improve each other (as mentioned above) but also enhances the advancement of both research and technology. For instance, EGI catalyzes a number of new edge applications (Section V) and incubates an ecosystem including GNN-oriented hardware accelerators and programming frameworks (Section VIII), which actually expedites edge AI popularization. Additionally, and perhaps more notably, new technologies and system models for edge AI are being nourished and flourishing, extending beyond the traditional scopes and standards of these two.

B. Rating of Edge Graph Intelligence

Given the reciprocal benefits of EGI, we advocate that EGI should not be restricted to merely applying GI on edge data or running GI on edge platforms. Instead, GI and edge networks are blending a confluence and EGI should be treated as a whole to reflect the inherent interplay between GI and edge networks. This indicates that their bilateral empowerment desires a comprehensive exploration such that the degree of EGI can be identified and measured. Specifically, according to the fusion of GI and edge networks, we may rate EGI into six levels from their respective perspectives, as shown in Fig. 6.

- *Level 0:* Given the graph data implicated in the edge network infrastructure, analytical models in Level 0 are unaware of the graph structures. The edge computing systems also process data in a graph-agnostic way. In other words, neither the model side nor the infrastructure side explicitly tackles "graphs", and thus they are categorized to the initial level. For instance, visual data mining with traditional ML models and conventional neural networks like CNN [93], [94] and RNN [95], [96] can be categorized in Level 0 since their data are not formulated in graphs and their data processing pipeline has no specific relations with graphs.
- *Level 1:* Data collected from edge networks are modeled in graphs but are not exploited in analytics. Systems at Level 1 push one step further over Level 0 by endowing graph semantic to edge data through graph modeling with collected edge data, but process data with general computing methods (instead of graph-related methods). As an example, while wireless sensory data can be generally abstracted in graphs [36], [37], Level-1 systems treat them in a vertex-wise manner, e.g., separately analyzing individual sensors' data arrivals, instead of processing as a whole graph.
- *Level 2:* Edge data in graph formats are processed with traditional graph computing algorithms such as PageRank [97] and single-source shortest path algorithms [98]. In this respect, edge data are modeled in graphs and processed with traditional graph algorithms rather than ML methods. Systems at Level 2 outperform Level 1 by enabling graph-oriented computing capability though they do not introduce AI techniques in graph processing.
- *Level 3:* Edge networks serve GI model inference with graph data, where the models may be trained on the cloud. Compared with lower levels, systems at Level 3 not only allow graph formulation and computing of edge data but also initiate AI to edge networks and embrace GI models such as GCN [15] and GraphSAGE [47]. In other words, edge systems that compute GNN inference such as GNN-based traffic forecasting systems [18], [99] and DGRL decision systems [100], [101] are located at Level 3.
- *Level 4:* Edge networks perform GI model training with graph data. Example includes SUGAR [102] and HGNAS [103]. The key difference between Level 4 and Level 3 lies in the ability to learn edge-native GI models, e.g., fine-tuning model parameters with edge data. Otherwise stated, systems at Level 4 can customize their GI models in-situ, thereby enabling tailored ability for edge AI services.
- *Level 5:* Interactional EGI, where GI and edge networks can dynamically adapt their configurations during the

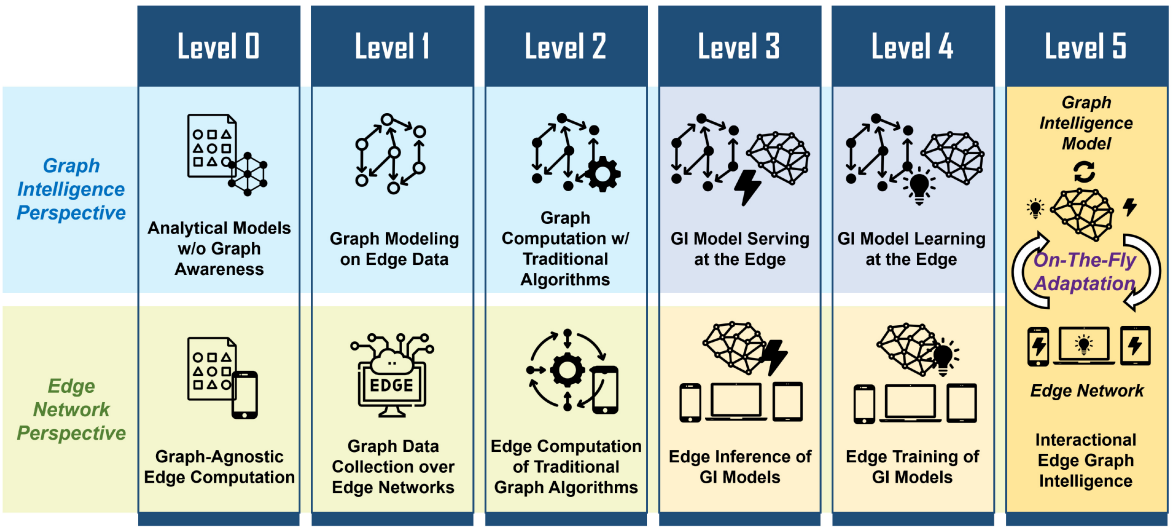


Fig. 6. Rating of EGI from the perspectives of both graph intelligence and edge network. In particular, EGI in Level 5 coalesces both lines and renders interactional EGI that can dynamically adapt the GI model for optimal model performance and the edge network for optimal runtime performance on the fly.

runtime for optimal EGI performance. Systems at Level 5 outperform all other levels because they can adjust GI and edge networks on the fly, whereas lower levels are all static settings. Both perspectives of GI and edge networks reach a convergence since they are in complete harmony.

The EGI rating can be mainly divided into three intervals. The first interval covers from Level 0 to Level 2, where EGI is less related to AI and even processes non-graph data. The second interval comprises Level 3 and Level 4, where EGI incorporates GI models by either inference or training on edge networks. The third interval is exactly Level 5, which stands at the highest level because its GI and edge networks have profoundly blended as integration and can adapt to diverse scenarios on the fly. As EGI systems are located at higher levels, their fusions of GI and edge networks go deeper. As a result, the intelligence resources of GI and infrastructural resources of edge networks are progressively exploited for better EGI performance. Nonetheless, this may also come at the cost of additional development effort and system overhead. This conflict implies that there is no "silver bullet" in all cases. Instead, the panacea of EGI in practice should align with user demand, anticipating a joint consideration of specific application scenarios as well as available resources budgets. In this survey, given the focus on advanced GNN in edge environments, the reviewed systems mostly lie in Level 3 and beyond.

V. Edge Applications of Graph Intelligence

Graphs are highly abstract data structures and can be used to represent a spectrum of data generated in edge networks. These graph data drive miscellaneous applications at the last mile of the Internet, as exemplified in Fig. 7, and catalyze graph learning as a promising principle for edge intelligence. Following the insights discussed in Section IV, we first provide a panorama of representative EGI applications before showing how edge networks and GI interact with each other in subsequent sections. For clarity, we categorize them into five groups, i.e., smart cities, robots and vehicles, human sensing and analysis, location-based recommendation, and graph-based mobile vision. Note that edge applications of GI models mostly only compute GI model inference at the edge and these related EGI systems therefore lie at Level 3 in the rating taxonomy.

A. Smart Cities

Managing large cities is one of the key problems for modernization, especially under the stresses of continuously booming urbanization and the rapidly growing population. To enable resource-efficient management, AI has been widely recognized as an advantageous way to assist, promoting the concept of "smart cities" [106], [107]. Given the ubiquity of wireless sensor networks and graph data in cities, GI shines among AI techniques and enables numerous smart applications. Note that many use cases may not be fully deployed on edge networks but involve the cloud as a backend, our review focuses on the processing of graph data collected at the edge and demonstrates how these applications run. In the following, we discuss representatives concerning transportation, energy, environment, and public safety.

1) Intelligent Transportation Systems: The transportation system spans geographical areas and acts like the blood vessels of a city. To keep it smooth and unimpeded, GI models are applied for forecasting its states, predicting the supply and demand of users, as well as coordinating its resources.

Traffic state forecasting involves forecasting traffic amount, traffic flow, and travel time between two locations, etc. Typically, it takes data from roadside sensors and organizes them in a graph along with the structure of the road net, as visualized in Fig. 7(a), where the feature vector of each vertex is the collected data of each sensor. GI models can thereafter be utilized to learn the semantics of traffic networks and export predictions. For example, Li et al. [18] introduce the Diffusion Convolutional Recurrent neural network to model

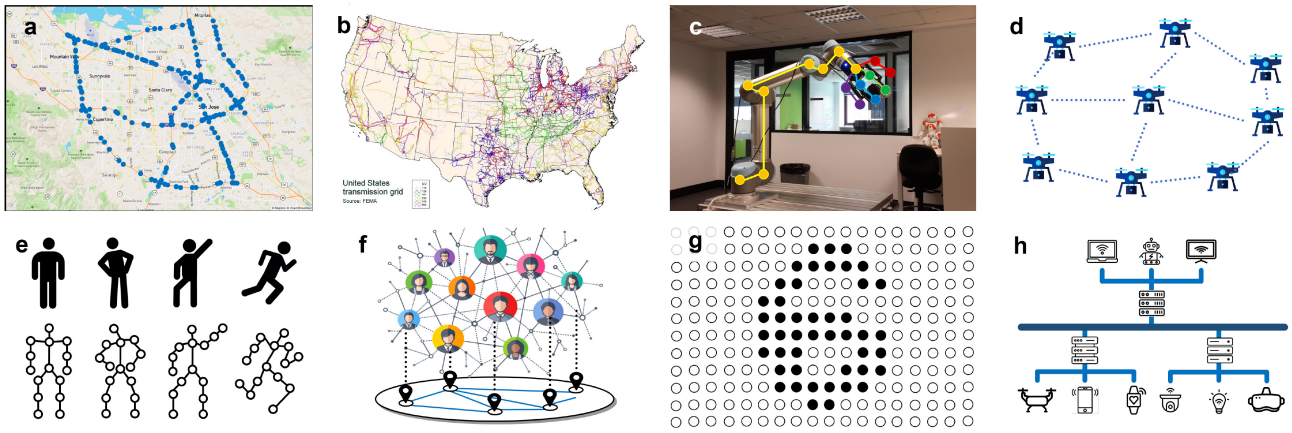


Fig. 7. Edge networks host a broad spectrum of platforms including wireless sensors, robots, vehicles, gateways, and cameras. These facilities spawn miscellaneous smart GI applications with their data organized in graphs. (a) A traffic sensory network is a graph with sensors as vertices and links aligned with the road net. (b) A power grid is naturally a graph that connects electrical facilities (image adapted from [104]). (c) A robotic arm can be abstracted in a graph with joints as vertices (image adapted from [105]). (d) A drone swarm comprises vehicles and their communication relationships shape a graph. (e) Human skeletons can be represented in graphs. (f) Location-based social networks are constructed upon users' fellowship. (g) A graph representation of an 18×12-pixels image of the digit "6", where pixels are vertices and their direct adjacency are links. (h) An edge network hierarchy is a tree-like graph.

traffic flow as a diffusion process on a directed graph, capturing both spatial dependencies through bidirectional random walks and temporal dynamics using an encoder-decoder architecture. Recognizing the critical role of spatial and temporal dependencies in traffic flow, Yu et al. [99] present Spatio-Temporal Graph Convolutional Networks, a novel approach for time series prediction in traffic analysis. Instead of applying regular convolutional and recurrent units, STGCN utilizes a fully convolutional structure on graphs, enhancing training efficiency and reducing parameter count.

Supply and demand predictions are a type of trip prediction task though they are often viewed as an ensemble, namely the origin-destination demand prediction. This can be relevant in many ride-hailing services (e.g., Uber [108], DiDi [109], and Lyft [110]), where accurately predicting users' demand is crucial for dispatch orders concerning efficiency and profit. Building on the importance of capturing dynamic dependencies in traffic flow, Lu et al. [111] developed a dual attentive GNN that can effectively predict the distribution of metro passenger flow considering the spatial and temporal influences. Further expanding on the application of attention mechanisms in GNNs, Makhdomi and Gillani [112] introduce a GAT method for passenger origin-destination flow prediction that leverages diverse linear and non-linear dependencies among requests originating from distinct locations, capturing both the repetition pattern and contextual data of each location. Their superior performance demonstrates the capability to capture inherent connections between regions or origin-destination pairs, and some of them [111], [113] have been deployed in the wild.

Traffic resource management is to detect or schedule dedicated components in transportation systems. As an example, Wang et al. [114] and Yu and Hu [115] both introduce GI-based approaches aimed at addressing critical issues in intelligent transportation systems. These frameworks capture complex dependencies within their respective applications, showcasing the versatility of GNNs in this field. However, their focuses differ significantly. Wang et al. [114] specifically design their GI model for traffic anomaly analysis in IoT-based intelligent transportation systems, aiming to identify and analyze traffic anomalies, while Yu and Hu [115] focus on the operational problem of relocating and repositioning idle vehicles, a crucial aspect of traffic resource management.

2) Power Grid: As basic facilities for smart cities, power grids deliver electricity from power plants to end users through transmission and distribution lines, substations, and equipment. To serve as many people in the city as possible, modern power grids have evolved to a colossal scale, becoming an extremely complex system that is vulnerable to attacks and system failure. Given the graph nature of power grids, GI models are employed to ensure their reliability and stability, i.e., for failure detection and energy harvesting prediction.

Failure detection aims at localizing potential or yet-occurring failures within the power grid. Liao et al. [116] utilize the adjacency matrix to represent the similarity between unknown and labeled samples, and propose graph convolutional layers for identifying complex nonlinear relationships between dissolved gas and fault type. Building on the use of GCNs in power system diagnostics, Liu et al. [117] further present a technique that employs GCNs for swiftly identifying key failure points in power systems. This method simplifies and speeds up the process of detecting critical cascading failures, making it highly effective for complex power networks. Extending GI to the realm of cybersecurity in power grids, Boyaci et al. [118] introduce a deep learning model utilizing Chebyshev GCNs for detecting cyberattacks, specifically false data injection attacks, in large-scale AC power grids.

Energy harvesting prediction is for power grids with vulnerable energy input, e.g., solar energy and wind energy. Accurately predicting their future behavior is of great importance for power systems with high penetration of RESs for stable operation and economic consideration. Karimi et al. [119] focus on improving photovoltaic power forecasting by leveraging spatial and temporal coherence

among power plants. They propose an STGNN to harness the relationship between power plants, observing that plants in a region experience similar environmental conditions and can thus inform each other's power forecasts. Khodayar et al. [25] introduce a convolutional GAE for capturing continuous probability densities on graph nodes. By integrating spectral graph convolutions and variational Bayesian inference, convolutional GAE generates samples from these densities. This approach is applied to probabilistic solar irradiance prediction, using an undirected graph model of solar radiation measurement sites for future irradiance estimation. In their subsequent research, Khodayar et al. [120] further tackle the challenge of forecasting behind-the-meter load and rooftop photovoltaic generation in power systems using a spatio-temporal GAE and graph dictionary learning optimization.

3) Environment Monitoring: Environmental science assumes a crucial role in comprehending the dynamics of natural systems and their complex interactions with human activities, particularly for smart cities. DL, as a powerful tool, has emerged to support environmental science in smart cities, enabling the analysis of vast amounts of data to model and predict environmental phenomena such as air quality concentration. In this context, GI techniques have found extensive applications in the study of wireless sensor networks for environmental monitoring in the realm of smart cities, with respect to environmental assessment and prediction as well as meteorological monitoring and forecasting.

Environmental assessment and prediction aims at extracting environmental information from sensory data collected across smart cities. Chen et al. [121] introduce the group-aware GNN (GAGNN) for nationwide air quality forecasting, in order to understand latent dependencies among geographically distant cities. GAGNN constructs a hierarchical model with a city graph and city group graph, combined with a differentiable grouping network to discover and encode these complex inter-city relationships. In addition to the use of GCNs, both Gao and Li [122] and Mao et al. [123] incorporate LSTM to integrate spatio-temporal information for more accurate predictions. For example, Mao et al. [123] develop a graph convolutional temporal sliding LSTM model to predict various air pollutants. They use GCNs to model spatial dependencies and a temporal sliding LSTM strategy to capture dynamic changes over time, allowing for a broader application beyond just PM2.5 prediction.

Meteorological monitoring and forecasting leverage GI models to analyze meteorological data for understanding current weather conditions and predicting future weather patterns. Remote automatic weather stations can collect geo-distributed meteorological data for temperature prediction. The targeted sensory data are usually in a spatio-temporal form, covering atmospheric parameters such as temperature, humidity, pressure, wind speed and direction, precipitation, and other relevant variables. As an example, Lin et al. [124] present the Conditional Local Convolution Recurrent Network for spatio-temporal forecasting in weather prediction, overcoming challenges of high nonlinearity and complex spatial patterns. It introduces a GCN that captures local spatial patterns through conditional local convolutions and incorporates these into an RNN architecture to model temporal dynamics. Addressing the challenge of accurately forecasting frost events, which significantly impact agriculture, Lira et al. [125] describe a GNN with spatio-temporal attention architecture. The model, adept at handling the localized nature of frost influenced by various environmental factors, maps weather station data onto a graph structure, optimizing an adjacency matrix during training to capture complex environmental interactions. Similarly focusing on the spatial relationships among weather stations, Stanczyk and Mehrkanoon [126] introduce GI models for wind speed prediction, focusing on handling data from multiple weather stations.

4) Public Safety: Ensuring residents' physical safety and security is of primary importance in building smart cities. Towards that, GI techniques are employed to analyze graph data collected from city-wide edge networks and to find patterns related to residents' health. We next discuss them in three respects, namely natural disaster prevention, crime warning, and public health.

Natural disaster prevention utilizes GI to aid people in mitigating the impact of unexpected extreme hazards, including earthquakes, wildfires, floods, and hurricanes. Li et al. [27] presents a batch normalization GCN for early earthquake detection, which integrates a CNN with a GNN. The model employs batch normalization to reduce training epochs and stabilize activation value distributions, enhancing training efficiency and prediction accuracy for key earthquake parameters like magnitude, depth, and location. This approach of integrating CNNs with GCNs for efficient feature extraction and spatial relationship modeling is not limited to seismology. Similarly, Jin et al. [127] tackle the challenge of predicting urban fire dynamics, which is crucial for urban safety and emergency response. They propose a model that combines CNNs with GCNs, where CNNs extract pixel-level latent representations from fire situation awareness images and GCNs manage the spatial relationships between different urban areas. By integrating the strengths of CNNs and GCNs, [127] effectively illustrates how this combined approach can address complex spatio-temporal phenomena, highlighting its versatility and potential across different domains.

Urban crime is another critical factor imperiling residents' properties and lives. To raise the attention of people potentially in criminal danger, researchers develop GI models to predict crime frequency in dedicated regions. Xia et al. [128] present the Spatial-Temporal Sequential Hypergraph network (ST-SHN), aimed at improving crime prediction by addressing the dynamic nature of criminal patterns in both spatial and temporal domains, and the evolving dependencies between different crime types. ST-SHN employs a graph-structured message-passing architecture combined with hypergraph learning to manage the spatial-temporal dynamics and global context of crime. In parallel, Sun et al. [129] present AGI-STAN, a framework that improves crime prediction by discarding reliance on domain-specific knowledge and predefined graphs. It uses adaptive graph learning to autonomously discern interdependencies among urban communities and integrates a time-aware self-attention mechanism to model temporally varying crime incidents. Toward the same problem, these two

methods take different optimization paths: ST-SHN focuses on global context through hypergraph learning, while AGI-STAN discovers community interdependencies and processes temporal information adaptively.

Epidemic outbreaks, notably the recent novel coronavirus, pose significant threats to global public health systems. Accurate epidemic prediction is crucial, as it plays a vital role in safeguarding public health by informing and guiding proactive strategies to mitigate the impact of such health crises. In this context, GI has emerged as a powerful tool for modeling the complex interactions inherent in epidemic data. Both Kapoor et al. [130] and Keicher et al. [131] leverage the expressiveness power of GNNs to address different facets of COVID-19 prediction, underscoring the versatility and efficacy of graph-based modeling in this domain. While both studies employ GNNs to harness the complexity of COVID-19 data, they differ in their specific applications and data integration strategies. Kapoor et al. focus on macro-level predictions of case numbers across regions, emphasizing the epidemiological aspect of the disease spread. In contrast, Keicher et al. concentrate on micro-level patient outcomes within a clinical setting, emphasizing the individual variability in disease progression and response to treatment.

B. Intelligent Robots and Vehicles

Robots and vehicles are typical facilities deployed in edge networks and have undertaken multifaceted services for human beings [132]. Graphs are also widely used to represent their behaviors, thereby promoting GI as a promising tool to empower more autonomous machine intelligence. Regarding the objects that are represented in graphs, we discuss GI for robots and vehicles in three respects, i.e., body control and manipulation, motion prediction and planning, and environment exploration.

1) Body Control and Manipulation: Controlling the body of a robot is an initial step to steer its work, which requires an accurate modeling of it. However, body modeling is non-trivial, since it usually interacts with objects, environments, and human beings. For instance, when directing a robotic arm to grasp a compliant object, e.g., a soft bread, it very likely suffers a non-linear deformation, depending on the force and torques made by the arm. To address that, the robots are formulated in graphs with the spatial and kinetic relationships between keypoints, as depicted in Fig. 7(c), and GI models are applied to abstract them. Almeida et al. [133] address the complex task of modeling soft robots using a GNN to simulate a non-rigid kinematic chain, like a robotic soft hand.

With a model of the robot, we may envisage to command it to firmly grasp or manipulate some objects. This requires, beyond body modeling, to characterize the interaction between robots, objects, and environments. In this context, Li et al. [134] make a significant contribution by introducing dynamic particle interaction networks (DPINets), a differentiable, particle-based simulator. This innovative approach leverages dynamic interaction graphs to adeptly handle the manipulation of a variety of materials, including fluids, deformables, and rigid bodies. Building upon the foundation laid by the DPINets' focus on material interaction, Xie et al. [135] present a deep imitation learning framework for bimanual robotic manipulation in continuous state-action spaces, focusing on the challenge of generalizing manipulation skills to objects in varying locations. Further extending the application of graph-based learning in robotics, Liang et al. [136] introduce a self-supervised learning method, target-oriented Deep Q-Network (DQN), which employs visual affordance graphs for a complex object grasping task, guiding robot actions using environmental cues in both simulated and real-world scenarios. These three works collectively exemplify the power of graph-based modeling in advancing robotic manipulation capabilities

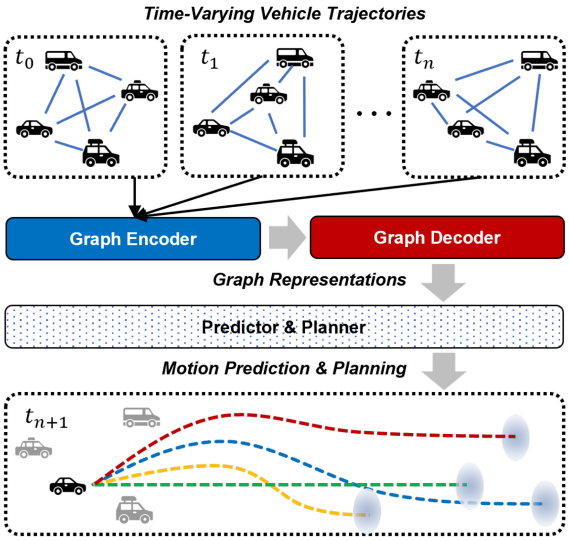


Fig. 8. An illustration of vehicle trajectory planning via GI models, where the historical trajectories of vehicles are modeled as a time-varying graph.

2) Motion Prediction and Planning: In many circumstances, vehicles are not launched individually but collaboratively, which requires global planning of vehicles to attain mutual goals [137]. For these cases, GI is also profitable with vehicles modeled in graphs. For instance, Fig. 7(d) shows a drone swarm committed to collision-freely flying from a starting point to a targeted location, where they can be viewed as a graph with links indicating their communication paths. Another illustration is shown in Fig. 8, where the movement of vehicles is modeled in a time-varying graph and fed into a GAE for learning graph representations. Through a predictor and a planner, the EGI system can export the prediction and planning of the vehicle's next motion. Li et al. [138] tackle the challenge of decentralized multi-robot path planning by introducing a model that synthesizes local communication and decision-making policies for robots in constrained workspaces. The model combines a CNN for extracting features from local observations and a GNN for efficient communication among robots. A similar structure is exploited by [139], which presents the dynamic motion planning model based on graph neural networks and historical information to enhance path-finding in decentralized multi-robot systems.

3) Environment Exploration: Contrary to motion planning where destinations are known, multi-robot exploration tasks

demand exploring an unknown environment a key aspect of automation applications like cleaning, searching, and rescue. For these tasks, scheduling robot swarms to efficiently cover the underexplored field while avoiding conflicts is vital [140], [141].

To achieve that, researchers define the spatial relationships between robots and regions of the environment and intend to apply GI to regulate this complex process. Luo et al. [142] were among the pioneers in this domain, introducing a graph-based DRL approach for autonomous exploration in unknown territories. By translating the exploration task into a graph domain through a hierarchical map segmentation, they paved the way for utilizing a GCN to effectively assign exploration targets to agents. This seminal work highlighted the potential of GNNs in enhancing multi-agent exploration efficiency. Building upon this, Gosrich et al. [143] advanced the application of GNNs by developing a decentralized control policy for multi-robot sensor coverage. Their innovative approach allowed robots with limited sensing capabilities to efficiently detect events in regions of varying importance. In parallel, Zhang et al. [144] have taken the application of GNNs a step further with the introduction of Hierarchical-Hops Graph Neural Networks (H2GNN) for multi-robot coarse-to-fine exploration. Their approach stands out by enabling selective integration of key environmental information through a multi-head attention mechanism, which discerns the importance of information from different hops.

Together, these studies provide a comprehensive view of how GNNs can be leveraged to address a spectrum of challenges in multi-agent robotics, setting the stage for future advancements in the field.

C. Human Sensing and Analysis

Human sensing is one of the core applications in many edge services, which use edge devices to collect and process data related to human activities and behaviors. Various platforms are involved, including sensors, smartphones, wearables, cameras, etc., to monitor various aspects of human behavior such as movement, gestures, vital signs, facial expressions, and environmental interactions. GI models are employed in this context for those data that can be formulated in graphs. In general, these human sensing applications are mainly recognized in three types: affective computing with facial graphs, action recognition with human skeletons, and smart health with human-centric sensor networks.

1) Affective Computing: Facial expressions contribute to more than 55% of the information perceived by individuals during verbal communication [145], [146], which plays a crucial role in conveying significant information related to emotional states and reactions within human interactions. To enable user-centric services deployed at the edge, the community has developed various intelligent affective computing methods to understand users' facial affects. Notably, some researchers establish graph representations for faces, where facial landmarks are marked as vertices and are connected in facial shapes, and introduce GI models to reason users' states.

In a harmonious convergence, Xie et al. [147], Liu et al. [148], and Zhao et al. [149] have each harnessed GNNs to decode the complex language of human emotions manifested through facial movements. Specifically, Xie et al. [147] delve into the micro-level of facial expressions, integrating action units with emotion category labels within a graph framework, thereby enhancing the precision of recognition tasks that are critical in various real-world applications. Building upon this graph-centric perspective, Liu et al. [148] propose a semantic graph-based dual-stream network that addresses a key limitation of conventional CNN methods by incorporating semantic information into the recognition process. Their model not only learns from physical appearance but also understands the deeper meaning behind facial movements. Similarly, Zhao et al. [149] contribute to the field by developing a geometry-aware FER framework, which combines the structural analysis capabilities of a GNN with the feature extraction prowess of a CNN. This dual-pronged strategy allows for a detailed examination of both the geometric and appearance-based aspects of facial expressions, leading to a robust recognition system that is sensitive to the minute details of human emotions.

2) Action Recognition: The rapid advancement of Human Action Recognition (HAR) and tracking techniques has emerged as a crucial catalyst for diverse edge applications, e.g., for security surveillance and robot imitation. The rationale behind using GI models for action recognition is to model human bodies into skeleton-based graphs, as exemplified in Fig. 7(e). Specifically, body joints are regarded as vertices with each attached a feature vector indicating its 3D/2D coordinates and confidence scores, while they are connected along with the human skeletons.

In the realm of graph-based models for skeleton dynamics, a spectrum of approaches has been explored, categorized by their focus on spatial or spatio-temporal characteristics. Wang et al. [150] present Graph-PCNN, a framework that refines keypoint localization through a two-stage process, enhancing the accuracy of human pose estimation. This model-agnostic framework underscores the importance of accurate initial localization, which is further improved through a graph pose refinement module. Building upon the spatial foundation, Peng et al. [151] take a leap into the spatio-temporal domain by employing NAS to evolve an adaptive GCN for HAR. Their approach integrates dynamic graph modules and multiple-hop connections, which not only bolsters the network's capacity to capture the nuances of motion in skeleton data but also aligns with the spatial-temporal emphasis of Graph-PCNN, albeit in a different context. Similarly, Jin et al. [152] address the complexity of multi-person pose estimation by proposing a hierarchical graph grouping method. This work, like Graph-PCNN, leverages the power of graph structures but diverges by reformulating the human part grouping into a graph clustering task, offering a fresh perspective on the integration of graph theory in pose estimation.

3) Smart Health: Smart health, also known as digital health or eHealth, has been a widely-concerned field in edge networks. GI techniques are widely adopted in the healthcare domain and their applications are mainly about Electronic

Health Records (EHRs) analysis and health condition analysis. In this paper, we particularly focus on the latter since it relates to edge networks.

Health condition analysis integrates GI with sensing techniques to enable healthcare monitoring and intervention. The synergy of these technologies is exemplified in Dong et al.'s work [153], where they introduce a GI model that harnesses mobile sensing data to detect early signs of influenza-like symptoms by encapsulating the dynamics of state transitions and internal dependencies within human behaviors through graph representation. Building upon this foundation, the concept of semi-supervised learning is elegantly integrated in the work [154], where they propose a Graph Instance Transformer (GIT). GIT not only combines multi-instance learning with contrastive self-supervised learning but also demonstrates the adaptability of GNNs in predicting early signs of mental health disorders. Similarly, Jia et al. [155] contribute to the field by introducing GraphSleepNet, a GI framework specifically tailored for automatic sleep stage classification. This framework adeptly tackles the challenge of utilizing brain spatial features and the transition information among sleep stages, showcasing the versatility of GNNs in addressing complex health-related problems.

D. Location-Based Services

Social relationships in edge networks usually appear in a location-based manner, i.e., with social entities distributed geographically. Provided with the nature graph structure of social networks as shown in Fig. 7(f), GI models bring advanced learning ability for mining location-based social information and exploring Point-of-Interest (POI), raising significant attention from the community. In the context of location-based social networks, we discuss their applications for social POI recommendation and geographical POI recommendation, respectively.

1) Social POI Recommendation: Social POI recommendation utilizes users' activities on social media platforms to provide recommendations for friends they may know or products they may like to purchase. In a pioneering approach, Kefalas et al. [156] enhance the traditional recommender system by introducing a hybrid tripartite graph that intertwines user, location, and session networks. This structure is analyzed using an advanced random walk with a restart algorithm, which captures the fluid nature of user preferences over time and space. Building upon this foundation, Salamat et al. [157] introduce HeteroGraphRec, a sophisticated recommender system that considers social networks a heterogeneous graph. Continuing this thread of innovation, Wang et al. [158] tackle the specific challenge of session-based recommendations by proposing an STGCN model. This model predicts a user's immediate interests by considering the sequence of their recent anonymous behaviors, thereby providing a more nuanced and responsive recommendation strategy.

2) Geographical POI Recommendation: The objective of geographical POI recommendation is to exploit users' behaviors to predict their potentially interested locations. Both Luo et al. [159] and Yang et al. [160] have introduced models that ingeniously incorporate spatio-temporal dynamics to predict user preferences for future locations. A common thread between these works is their reliance on attention mechanisms to dissect the complex interplay of spatial and temporal factors. Luo et al.'s STAN model [159] introduces self-attention to directly model the spatial and temporal aspects of user check-ins, excelling at capturing the significance of non-adjacent and non-consecutive interactions. In a parallel yet distinctive approach, Yang et al.'s STAM model enhances GNNs by incorporating scaled dot-product and multi-head attention mechanisms, which enriches the learning of neighbor embeddings. The transition from STAN to STAM illustrates the field's maturation, moving from specific spatio-temporal interactions to a more integrated and comprehensive understanding of user preferences in recommendation systems.

E. Graph-Based Mobile Vision

Computer vision tasks accept inputs in the forms of images, videos, and point clouds, where many of them are represented by graphs. For instance, Fig. 7(g) shows an image of the digit "6" in 18×12 pixels, whereby viewing pixels as vertices and their direct adjacency as links, it can be recognized as a graph. Another example is the skeleton-based graphs in many images of human actions, as discussed in Section V-C2 and depicted in Fig. 7(e). GI models are applied to these graphs to assist various traditional computer vision tasks like image classification, object detection, point cloud analysis, etc.

1) Image Processing: In the evolving landscape of image processing, Graph Inference (GI) models have emerged as a powerful paradigm, particularly for tasks that require an understanding of the relationships and context within visual data.

At the forefront of this advancement is the work by Liu et al. [162], who introduced the Structure Inference Network (SIN). This pioneering approach in object detection treats objects as nodes and their interactions as links within a graph, providing a more holistic and contextual understanding of the visual scene. Expanding on this foundation, Chen et al. [163] delve into the domain of multi-label image recognition. They propose a GCN model that constructs a directed graph of object labels, capturing the intricate dependencies and co-occurrences among them. This model not only builds upon the graph-based reasoning initiated by Liu et al. [162]. but also extends the concept to the challenges of recognizing multiple objects within an image. Following this thread, Lu et al. [164] first apply the GCN to image semantic segmentation with their Graph-FCN, an approach that redefines the task as a graph node classification problem, successfully integrating local location information often overlooked by standard deep learning methods.

2) Video Analytics: Video analytics with GI models are mainly for HAR and Multi-Object Tracking (MOT). While HAR has been thoroughly explored in Section V-C2, this section delves into the latter, focusing on the innovative approaches that have emerged.

Brasó and L. Leal-Taixé [165] set a message-passing network that not only integrates learning within the MOT paradigm but also crucially extends it to the data association

phase (which is traditionally a bottleneck). Simultaneously, Liu et al. [166] introduce the graph similarity model for MOT, which integrates a unique graph representation of object features and relationships, coupled with a graph matching module, to enhance robustness against occlusion and similar appearances in tracking scenarios. Both of them have independently demonstrated the potential of GI models to transform MOT by focusing on different yet complementary aspects of the problem, i.e., global reasoning and robust feature representation.

3) Point Cloud Processing: Different from images in 2D representations, point clouds are organized in 3D structure and provide richer semantics in describing scenes and objects. For instance, many autonomous vehicles use LiDAR to scan the environment and export point cloud data for perception, as in Fig. 9. Yet the 3D data are still accommodated in a graph representation, and thereby GI models are leveraged for efficient processing.

Wang et al. [167] develop EdgeConv, which enhances the representation power of point clouds by recovering topological information. This method uses dynamic graph construction to capture local geometric features, proving its suitability for high-level tasks in computer graphics and vision. Building on this idea of dynamic feature extraction, Shi and Rajkumar [161] propose Point-GNN, a GI model for object detection from LiDAR point cloud, which incorporates an auto-registration mechanism to reduce translation variance and a box merging and scoring operator for accurate detection over multiple vertices.

Extending these advancements, Lin et al. [168] develop 3D-GCN to address the challenges of processing and summarizing information in unstructured point clouds for 3D vision applications. Their approach is particularly effective when dealing with data variations such as shift and scale changes, aligning with the principles seen in EdgeConv. Furthermore, Zhang et al. [169] present the linked dynamic GNN, which tackles the challenge of processing sparse, unstructured point clouds by linking hierarchical features from dynamic graphs. This method optimizes the network's architecture for more effective point cloud classification and segmentation, showing how linking hierarchical features can avoid the vanishing gradient problem and improve overall performance.

Lessons Learned (Section V)
EGI has catalyzed a broad range of applications beyond traditional edge AI scenarios. These applications typically follow a general working procedure: the system collects data from components of edge networks and formulates them in graphs. The graph data are ingested into a GI model for graph embedding, along with a subsequent model, e.g., FCN and LSTM, for downstream tasks. The key to building a well-performed EGI application is carefully modeling the graph data and properly selecting the GI model.

VI. Edge Networks for Graph Intelligence

Given the applications introduced in Section V, this section discusses how edge networks serve GI execution in their multi-tier hierarchy. In the following, we survey various computing systems in different architectures according to their reliance on the cloud. Note that in each architecture, we review both model training and model inference systems, and their EGI ratings thus diverge.

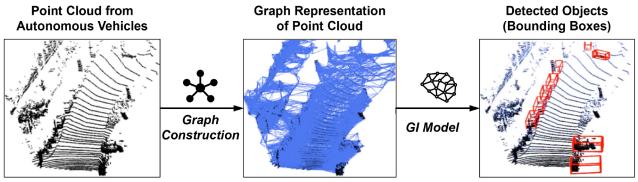


Fig. 9. Example point cloud processing with GI models. The point cloud data captured by autonomous vehicles can be reorganized in a graph format and processed with GI models for high-precision object detection. Image adapted from [161].

A. Federated Graph Learning

Edge-cloud synergy extends cloud resources as remote computing assists in optimizing the performance and efficiency of user-centric computing and data processing. Specifically, it enables certain computing tasks reserved at the edge for immediate processing, while employing the cloud for offloading more resource-intensive tasks with long-term storage and complex data analysis. This hybrid computing paradigm combines the differentiated capabilities of edge and cloud computing, thereby admitting flexible service computing for various applications [170].

For edge computation of GI models, typical workloads in the form of edge-cloud synergy are FL of GI models, i.e., Federated Graph Learning (FGL). FGL is the Level-4 EGI system in the rating taxonomy but is distinct from GI-assisted FL (GFL) as discussed in Section VII-D. Although they both combine GL and FL, the targeted model to be federatively trained in FGL is exactly GI models but GFL allows the targeted model to be any model not only GI models. Readers particularly interested in the combination of GI and FL may find dedicated reviews [171], [172] for more information. With respect to the segmentation of the data space, FGL can be divided into horizontal ones (graph topology space) and vertical ones (feature data space).

1) Horizontal Federated Graph Learning: Horizontal Federated GNNs pertain to the scenario in which multiple clients participate in a federated learning setting, sharing identical node feature spaces while employing distinct node IDs [68]. Each client possesses a minimum of one graph or a collection of graphs, and vertices are distributed across diverse clients with interconnections. Concerning the explicity of links between subgraphs on clients, horizontal FGL works can be categorized into two classes, namely with explicit cross-client links and with implicit cross-client links.

The first setting considers FGL with explicit links across subgraphs on FL clients, as in Fig. 10. The conventional approach in this setting entails training local GI models within individual clients to initially acquire the local representations or node embeddings of the respective graphs. Subsequently, a Federated Learning (FL) algorithm is applied to facilitate

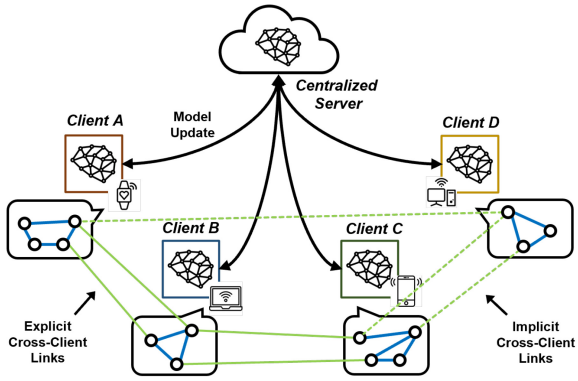


Fig. 10. Federated Graph Learning (FGL) leverages FL mechanisms to collaboratively train GI models with distributed clients. Specifically, horizontal FGL considers data splitting in the graph topology dimension, where the relationship across client data may be explicit or implicit to the centralized server.

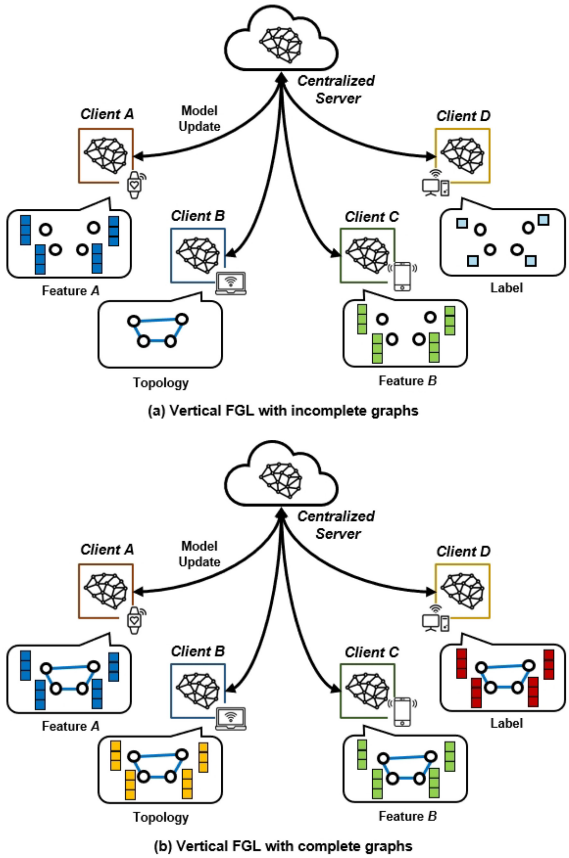


Fig. 11. Vertical FGL considers data split in the feature space with (a) incomplete graphs or (b) complete graphs.

the aggregation process, and the FL server, which is usually a cloud server, accumulates the model parameters or gradients received from the clients and performs FL aggregation. The resultant updated parameters are then returned to the clients for subsequent rounds of training. The introduction of FGL within this framework is contingent upon the specific research problems they address. For Non-IID graph data issues, many approaches intend to enhance the adaptation performance of local models or acquire a robust global model through model adjustment like model interpolation [173], [174] and meta learning [175].

The second setting assumes that certain links connecting vertices across different clients are implicit and absent, illustrated in Fig. 10. For the case where the implicit links connect vertices with dissimilar IDs, a prevalent strategy involves rectifying the local graphs by reconstructing the missing links. This ensures the local graph's completeness, thereby facilitating high-quality graph representation. Furthermore, the inclusion of links connecting clients aids in mitigating the challenges associated with non-IID data. FASTGNN [176] introduces a straightforward link generator in the FL server, which is responsible for reconstructing absent links between clients by introducing Gaussian randomly generated links. These reconstructed edges are broadcast to all clients during FL training, prompting them to update their local graphs. In a similar vein, the link generator in FedGL [67] produces a global pseudo graph with vertex embeddings provided by clients. Subsequently, this global pseudo-graph is disseminated to all clients, enabling them to amend their local graphs to facilitate GI model training. Based on the simple yet effective idea of link generation, follow-up works further develop more complex techniques for better performance [177], [178], consider privacy issues with security means [179], [180], [181], and explore to improve communication efficiency [182], [183].

For the case where links are implicit in different clients, Knowledge Graph (KG) techniques are involved to complete information between these vertices. Once the local graphs have been rectified, the application of an FL algorithm facilitates the training of GI models in a similar manner as observed in scenarios with explicit links. FKGE [184] adapts a GAN-based module [185] to facilitate the translation of aligned entity and relation embeddings across paired KGs, where the refined embeddings will be broadcast as long as the paired KGs are improved. FedE [186] establishes an entity table on the server that catalogs unique entities from multiple clients. This table employs the FedAvg aggregation to process the aligned entity embeddings, and once processed, the updated embeddings are distributed back to the clients. Upon FedE, FedR [187] and FedEC [188] further improve the capability of privacy preservation and non-IID data resilience.

2) Vertical Federated Graph Learning: Vertical FGL assumes that clients possess nodes characterized by fully overlapping node IDs, albeit with distinct feature spaces. Through the utilization of FL, clients collaborate in training a global Graph Neural Network (GNN) model using features sourced from multiple clients. Within vertical FGL, there exist two distinct cases.

In the first case, graph topology, vertex features, and vertex labels are distributed across different clients. Consequently, clients may not be able to obtain the full graph data with respect to both vertex features and graph topology, as shown in Fig. 11(a). To orchestrate these clients in such semi-blind situations, SGCNN [189] computes a similarity matrix based on the dynamic time-warping algorithm to sketch the graph's topology without knowing the underlying structure. One-hot encoding is employed to describe vertices' features under

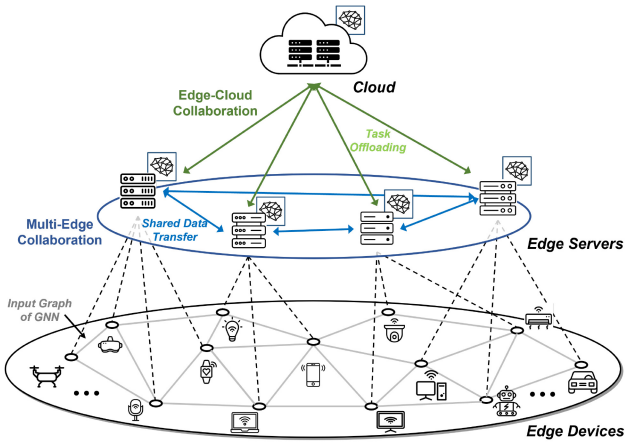


Fig. 12. The graph data collected from edge networks can be processed via distributed edge collaboration, i.e., edge-cloud collaboration or multi-edge collaboration. The former bridges the cloud and the participants from edge networks through GI model computation offloading, while the latter usually assembles multiple edge platforms as a whole to run GI services in a cooperative manner.

privacy requirements, which will be transmitted to the server to train the global GI model for GI-based tasks. FedSGC [190] considers a two-client decentralized setting with vertices' features and graph topology being respectively possessed. In this scenario, FedSGC proposes an additively Homomorphic Encryption (HE) method to ensure privacy preservation during FGL.

The second case entails the ownership of solely disparate vertex feature spaces by clients, while the graph topology remains accessible to all clients (Fig. 11(b)). For the case with only two clients and a central server, FedVGCN employs HE to encode intermediate data transferred across clients, where the server generates encryption key pairs and performs global FL aggregation. FMLST [191] assumes a shared global spatio-temporal pattern across clients and applies an MLP to fuse local patterns and global patterns for personalizing client models. Graph-Fraudster [192] investigates adversarial attacks on this vertical FGL scenario and shows differential privacy mechanisms and top-k mechanisms as two viable defense strategies against such attacks.

B. Distributed Edge Collaboration

Besides completely offloading to the cloud or executing in-situ, distributed edge collaboration, as a balanced compromise between them, introduces distributed training or inference of GI models within an edge network or between the edge-cloud continuum as in Fig. 12. This compromise offers several advantages, such as reduced remote transfer without cloud communication, (conditional) privacy preservation, and resource-augmented scalability [193], [194], [195], [196]. According to the dependency of cloud servers, distributed edge collaboration typically exhibits in two forms, i.e., edge-cloud collaboration and multi-edge collaboration.

1) Edge-Cloud Collaboration: Edge-cloud collaboration differs from traditional integrated cloud offloading by splitting computation workload between edge platforms and cloud servers and thus can fortify the duty of the edge side and alleviate the reliance on the remote cloud [16], [197], [198]. This allows edge facilities to be a key processor that blocks sensitive raw data from the cloud and eliminates redundant data transmission in the backhaul. Following this technical path, Branchy-GNN [199] concentrates on GI-based point cloud processing and applies edge-cloud collaboration with a learning-based source-channel coding scheme for bandwidth-efficient intermediate data transfer. It also borrows the idea of joint split learning and early exiting to reduce the computational cost on edge devices [16], [200]. Pyramid proposes a hierarchical architecture for edge-cloud synergetic GNNs that trains a local, small model on edge servers with its collected data and a global model on a cloud server to capture inter-region spatio-temporal relationships. During the runtime, Pyramid [201] allows GI-based predictions with either the local model or the global model in an on-demand fashion and achieves lower inference latency against baselines. Branchy-GNN is a Level-3 EGI system since it only tackles GNN inference at the edge, while Pyramid rates Level-4 because of its joint training and inference designs.

2) Multi-Edge Collaboration: Multi-edge collaboration [202], [203], [204] provisions all data flow within the edge networks, where edge devices individually perform local computation on the local data they possess and collectively train or infer GI models by sharing information and merging the results. On the one hand, compared to the cloud-assisted solutions, multi-edge collaboration fully reserves data without leaking them to the remote datacenter, thus avoiding users' privacy concerns from cloud providers [90], [205], [206]. On the other hand, collaborating multiple edge devices augments the available resources for targeted GI tasks beyond a single edge device, which therefore enables superior performance against on-device computation [87], [207]. Besides, privacy requirements are also guaranteed if the participating edge devices are mutually trusted, which can be relevant for many edge scenarios, e.g., in a smart home with edge devices owned by the same user and in a manufacturing pipeline with cameras and sensors possessed by the same company [170], [208].

Under this paradigm, Fograph [209], [210] proposes a distributed GNN inference system that leverages multiple edge servers in proximity to IoT data sources for real-time GI model serving. To address resource heterogeneity and dynamics among edge servers, Fograph designs a load-aware inference execution planner as well as an agile workload scheduler for maximum parallelization. It also introduces a degree-aware compression mechanism to minimize data transmission overhead. Following this principle, GLAD [211] studies the cost optimization problem for distributed GNN serving in multi-tier edge networks. By formulating cost factors in the whole lifecycle of edge-enabled GI services, it separately considers static graphs and dynamics graphs and derives a holistic graph layout scheduling solution with theoretical performance guarantees. Both Fograph and GLAD are Level-4 systems for their functionality of GNN inference at the edge.

C. On-Device Computation

On-device computation for GI models refers to performing GI model computations directly on edge devices rather than relying on cloud or remote servers. This entails executing the training or inference tasks locally on edge devices such as smartphones, laptops, or tablets, without requiring a constant Internet connection or external server support. This paradigm is particularly useful for privacy-sensitive applications because it ensures users' data to be fully preserved in situ. Nonetheless, given the intensive workload of computing GI models, on-device edge computation of them is non-trivial, and researchers have developed several routines to tackle that by designing resource-friendly procedures and models or compression for compact GI models.

1) Resource-Efficient Procedure: Without modifying GI models, resource-efficient procedure design indicates the endeavors that develop GI processing flows in minimal resource usage, e.g., memory footprint. For instance, centering on the out-of-memory issues where GI models' sizes are too large to fit into edge devices' memory, Zhou et al. [212] (Level-3 rating) design a feature decomposition approach that decomposes the dimension of feature vectors and performs aggregation respectively, reducing up to 5× memory footprint against generic GI frameworks. Building on the theme of memory efficiency, SUGAR [102] (Level-4 rating) targets resource-efficient training on resource-constrained edge devices through a subgraph-level training scheme that first partitions the initial graph into a set of disjoint subgraphs and then performs local training for individual subgraphs. Complementing these approaches, RAIN [213] (Level-3 rating) explores the opportunity of conservatively processing similar inference batches based on local sensitive hash and reusing repeated vertices' data among successive batches to reduce redundant data loading.

There are also some works that utilize channel pruning methods to reduce input feature dimensions for GI model inference acceleration. Zhou et al. [214] set the stage by employing a LASSO regression-based pruning technique, which identifies the most influential feature channels, thus ensuring that the pruning process is both effective and minimally disruptive to the model's accuracy. Yik et al. [215] extend this line of inquiry by conducting a meticulous analysis of the trade-offs inherent in channel pruning. They develop algorithms that not only preserve the model's predictive prowess but also enhance computational efficiency, aligning seamlessly with the work [214] by further honing in on the most valuable features.

2) Resource-Efficient Model: Designing resource-efficient GI models usually involves simplifying operators in GI semantics to improve computational efficiency in both GI model training and inference. Recent advances in this routine have explored eliminating redundant operators like linear aggregation and non-linear activation. For instance, SGC [216] removes the non-linear ReLU operators in conventional GI models to reduce the model complexity and reserves only the final Softmax function for probabilistic output generation, which significantly accelerates the model training yet maintains comparable accuracy in many generic tasks like text and graph classification. Other examples include LightGCN [217] and UltraGCN [218] for location-based social recommendation tasks, where the former drops the feature transformation and non-linear activation and the latter modifies the message-pass process into a simplified approximate embedding method. As a result, both of them effectively decrease the computational load against baselines and attain remarkable speedup.

To automate the resource-efficient model design for edge platforms, some researchers [103], [219], [220] introduce NAS techniques to the GI domain and present hardware-aware NAS frameworks tailored to GNNs, aiming to strike a balance between model accuracy and efficiency corresponding to the characteristics of targeted devices. Zhou et al. [103] present HGNAS, a hardware-aware graph neural architecture search (GNAS) framework that optimizes GNN designs for edge devices by balancing accuracy and efficiency using a hardware performance predictor. Similarly, PaSca [219] constructs scalable GNNs through a novel scalable GNAS paradigm, enabling systematic exploration of a vast design space to optimize GNNs for both accuracy and efficiency via multi-objective optimization. Building upon these concepts, MaGNAS [220] is a mapping-aware GNAS framework that efficiently processes vision-based GNN workloads on heterogeneous multi-processor platforms by identifying optimal GNN architectures and mappings for maximized resource efficiency and performance trade-offs. Together, these works highlight the growing trend towards hardware-aware and scalable GNAS frameworks that address both the accuracy and efficiency requirements crucial for deploying GNNs on edge devices. This line of work represents Level-4 EGI systems since they enable both edge training and inference.

3) Model Compression: Compressing a complete model into a smaller one that appropriately fits the capability of the targeted edge device has been a popular deployment means for many edge intelligence services. The smaller model, which usually with fewer parameters, allows much less computing cost and thus significant acceleration for on-device computation. To obtain such a lightweight model for EGI, researchers typically employ three ways, i.e., model quantization, knowledge distillation, and neural architecture search.

Model quantization intends to quantize GI model parameters from an origin, larger bitwidth to a targeted, smaller bitwidth, and thus reduce the total size of the model. In this context, SGQuant [221] proposes a multi-granularity method that can quantize feature vectors at the component level, topology level, and layer level to meet diverse data precision demands. It also designs an automatic bitwidth-selecting algorithm to attain proper configuration among different quantization granularities. Building upon the quantization paradigm, Degree-Quant [222] pioneers a quantization-aware training scheme specifically crafted for GI models. This scheme is designed to facilitate low-precision inference on devices with constrained resources, without compromising the model's predictive capabilities. VQ-GNN [223] takes a

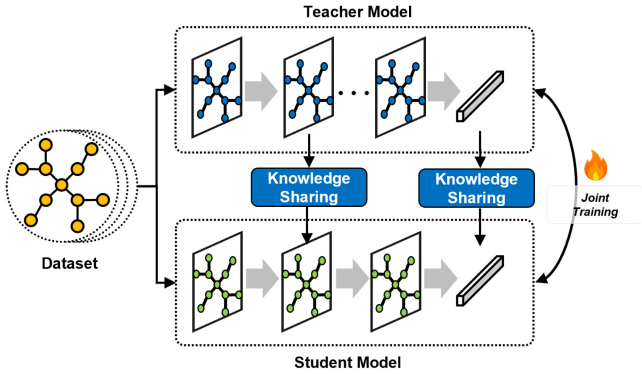


Fig. 13. In general, KD for GNN follows a similar procedure of KD for traditional DL models, where we share the knowledge of the teacher model with the student model to improve its accuracy.

quantum leap in this domain by introducing a unified framework that applies vector quantization (VQ) for dimensionality reduction and learns a small number of quantized reference vectors of global node representations, which viably avoids the "neighbor explosion" problem and enables faster mini-batch training and inference of GI models. Following this quantization principle, some researchers go one step further with GI model binarization, i.e., binarizing parameters in GI models for aggressive execution speedup [224].

Knowledge Distillation (KD) aims at extracting knowledge from a complex teacher model and transferring it into a compact student model while maintaining comparable performance [225], [226], as depicted in Fig. 13 While KD is originally applied in the computer vision domain [227], [228], [229], Yang et al. [230] first introduce KD for GI models along with a local structure preserving method for teacher-student similarity measurement and experimentally show its effectiveness in GI-based 3D object detection tasks. Along with this line, Yang et al. [231] combine parameterized label propagation and feature-based MLP in the KD's student model, allowing better utilization of the knowledge from both the teacher model and prior basement. TinyGNN [232] focuses the KD principle on accelerating model inference and designs a peer-aware module and a neighbor distillation strategy for preserving model accuracy. They all focus on the edge inference aspect and are Level-3 EGI systems in the rating taxonomy.

Lessons Learned (Section VI)
Edge networks as a physical infrastructure can support GI in various architectures, including FGL with edge-cloud synergy, collaborative computing with distributed edge platforms, and in-situ computing on individual edge devices. The architecture used for EGI services depends on the demand of SLO (e.g., latency and privacy requirements) and the availability of the cloud. The lower reliance on the cloud the system has, the more stringent the constraint of computing resources, and thus the more attention is paid to resource-efficient optimizations.

VII. Graph Intelligence for Edge Networks

Besides directly using GI models to specific edge applications discussed in Section V, GI techniques are also employed for optimizing edge networks. This can be relevant since many components in edge networks, e.g., network hierarchy as illustrated in Fig. 7(e), can be naturally recognized in graphs. Towards that, we discuss GI-based optimization on edge networks with respect to their resource management, graph-based decision-making, security issues, and GI-assisted FL in the following. Using GI for optimizing edge not only involves GI model inference at the edge but also training tailored GI at targeted edge platforms, thus their EGI systems are at Level 4 in the rating taxonomy.

A. Network Resource Management

Given the graph structure of massively connected edge networks, GI models are deemed to be a useful tool to achieve that and are leveraged for optimizing network resource allocation and network function orchestration.

1) Resource Allocation: Resource allocation is critical for ensuring networking efficiency and robustness, and has been extensively studied using traditional optimization means. Instead of applying complicated formulation and optimization, GI-based methods model the whole network as a graph and schedule an allocation strategy in a holistic manner [233], [234], [235].

Power control in wireless networks seeks the optimal power allocation to transmitters to minimize power consumption while maintaining signal quality [236], [237]. Shen et al. [238] introduce an interference GCN to solve the power control problem in wireless networks, designed to model K-user interference channels as complete graphs and incorporate wireless channel data as graph features for scalability. Their approach is complemented by Eisen et al. [239], [240], who employed spectral-based GCNs for power allocation in Device-to-Device (D2D) wireless networks, highlighting the effectiveness of GCNs in managing interference. Salaün et al. [241] study cell-free MIMO networks and formulate a graph of the dominant dependence relationship between access points and user equipment, using GNN for scheduling the power control. Building on these ideas, Nikoloska and Simeone [242] utilized REGNN to tackle power control challenges in decentralized wireless networks, leveraging the adaptability of GNNs to changing network topologies. Similarly, Yang et al. [243] propose UWGNN, which enhances GNNs by unrolling modules in the weighted minimum mean-square error algorithm into GI frameworks, demonstrating a method to integrate knowledge more effectively. Together, these studies underscore the versatility of GNNs in addressing power control problems across various wireless network settings, from centralized to decentralized architectures, and from static to dynamic topologies.

Link and channel scheduling is another key issue in communication networks as it drastically impacts the overall performance of the system (throughput, delay, etc.) [237]. Some researchers apply GNN or combine GNN with other models for channel tracking and throughput optimization in

MIMO networks, where the channel states and links are formulated in graphs [241], [244], [245], [246]. Yang et al. [244] intend to represent the obtained channel data in graphs by associating channel correlations as links' weights, based on which a GNN model is trained for channel tracking. D2D networks are also studied in a similar vein, e.g., GCN-GAN [247] integrates the strengths of GCN, LSTM, and GAN to capture the dynamics, topology, and evolutionary patterns of weighted dynamic networks for temporal link prediction. These methods collectively leverage graph-based models to enhance link scheduling and prediction, demonstrating the versatility and effectiveness of graph neural networks in addressing various aspects of network performance optimization.

Beamforming techniques are regarded as promising solutions in multi-user, multi-antenna communication systems [248], and also play a crucial role in wireless resource management. Chen et al. [249] present an innovative unsupervised GNN approach for beamforming design in D2D wireless networks. Their method significantly reduces the problem dimension by transforming beamforming into primal power and dual variables, showcasing superior performance with fewer samples. Further extending the application of GNNs, Kim et al. [250] develop a bipartite graph neural network framework for multi-antenna beamforming optimization in multi-user MISO systems. This approach partitions the beamforming optimization into suboperations for individual antennas and users, allowing for scalable and efficient computations. Both studies leverage the flexibility and efficiency of GNNs to handle complex beamforming tasks, demonstrating the potential of graph-based methods in optimizing wireless communication systems.

2) Network Function Orchestration: Virtual Network Function (VNF) virtualizes the functions that traditionally run on dedicated hardware devices such as routers and firewalls and has become a key enabler in software-defined networks (SDNs) [251]. Researchers employ GI models on VNF for orchestrating various tasks including adaptive Service Function Chain (SFC) and network slicing. SFC, crucial for identifying efficient paths in network servers for processing requested VNFs, faces challenges in maintaining high QoS in complex scenarios.

Heo et al. [252] introduce a GAE for dynamic network topologies, where the encoder represents the network topology and the decoder evaluates nodes for processing VNFs, adapting to topology changes. Heo et al.'s subsequent work [253] overcomes the limitation of earlier approaches restricted to fixed topologies, by employing RL for GNN-based model training across various unlabeled network topologies. Pioneering studies on network slicing [254], [255] employ Digital Twin (DT) technology to digitally replicate slicing-enabled networks, creating detailed virtual models that mirror the physical network's dynamic characteristics. These models are then used to develop GNNs that learn and optimize network behaviors directly from these non-Euclidean graph representations, thereby addressing challenges in resource allocation and QoS management in highly dynamic network environments. Besides the above aspects, we refer readers interested more about GNN for communications to related discussions [30], [256].

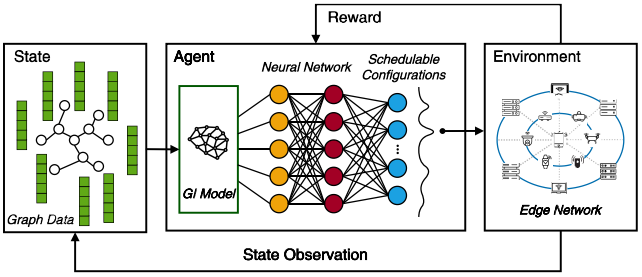


Fig. 14. In a general GI-incorporated DRL workflow, an agent observes states in a graph format and, through the GI model, abstracts relational semantics from states. Next, the agent passes the encoded data to a neural network operator to decide a scheduling action and receives a reward from the targeted environment (i.e., edge networks).

B. Graph-Based Decision Making

Given the powerful ability to abstract graphs, GI models can be integrated with DRL techniques to enable decision-making on graphs. Their confluence leads to DGRL, which is extensively utilized for scheduling decisions on edge networks, e.g., offloading, routing, and wireless network control. In general, the DGRL framework can be illustrated in Fig. 14, where an agent observes the state, schedules configurations, and feeds the agent (neural network) back with a reward after each action. The GI model is incorporated as a graph information encoder to abstract relational semantics from the state.

1) Offloading and Routing: Offloading and routing are classic techniques in edge networks that allow workload migration across networking entities and some researchers [100], [101], [257] use graphs to abstract tasks to be scheduled for offloading decisions. Specifically, ACED [100] introduces an actor-critic mechanism for DAGs-based computation offloading decisions considering task dependencies and wireless interference. Swaminathan et al. [101] enhance a routing algorithm for SDN using a GNN functioning with a deep Q-Network. Trained to forecast efficient routing paths, this GI model minimizes packet delivery delays through simultaneous training and inference, enabling continuous learning and real-time adaptation.

2) Wireless-Enabled Control Systems: DRL techniques have been frequently used in attaining efficient control of communication resources in wireless networks. However, trivially applying DRL does not scale well with the network size, and GI models are thus employed to overcome this challenge. For example, Nakashima et al. [258] design a DGRL method for scalable, model-free resource allocation in large-scale wireless control systems. It capitalizes on the ability of GNNs to apply spatial-temporal convolutions to graph-structured data, harnessing the natural graph form of wireless networks for efficient policy design. Some of its follow-up works [259], [260] extend this approach by incorporating long-term constraints and demonstrate the permutation invariance and transferability

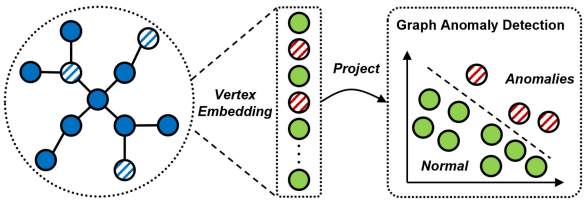


Fig. 15. GI models can abstract vertices's features into embeddings, which can be used for detecting anomalies by classifying in a latent space.

of graph-based reinforcement learning policies in wireless control systems.

C. Security and Anomaly Detection

Detecting anomalies in a graph is an important task in the analysis of complex systems, especially for ubiquitous edge networks. The anomalies, in the context of graphs, are patterns that fail to conform to normal patterns expected of the graph's attributes or structures. In this respect, the community has developed GI models for anomaly detection tasks and applied them to optimize edge networks. Fig. 15 illustrates its rationale, where vertices (with features) can be transformed into embeddings, which are used for anomaly identification in a latent space.

1) Static Network: For edge networks that are static within a period, e.g., a smart home network in a short slot, we call it a static network. Extensive research has been conducted focusing on the detection of anomalies within static graphs, encompassing various categories including vertex, link, and graph-level anomalies [261]. AnomMAN [262] incorporates a graph auto-encoder module to leverage the low-pass filtering characteristic of graph convolution, typically a disadvantage for anomaly detection, transforming it into a beneficial feature by utilizing reconstruction errors to effectively identify anomalies. EFraudCom [263] proposes a fraud detection system employing GI to identify anomalous edges in e-commerce platforms. The system's core innovation lies in modeling the distributions of normal and fraudulent behaviors separately, enhancing robustness against evolving fraud patterns. GLocalKD [264] and OCGTL [265] both tackle graph-level anomaly detection. The former introduces a method using random distillation of graph and node representations for deep anomaly detection, and the latter employs self-supervised and transformation learning to enhance existing deep one-class approaches [266].

2) Dynamic Network: Dynamic edge networks take temporality into consideration where their topology and attributes may change over time [267]. Recent studies, such as those by Deng and Hooi [268] and Zhang et al. [269], have delved into the intricacies of multivariate time series anomaly detection within cyber-physical systems. Both of them underscore the significance of understanding the complex interplay between sensors over time. Deng and Hooi [268] propose an attention-based graph deviation network to learn the complex relationships between sensors in multivariate time series, not only accurately detecting anomalies but also providing an explainable model that identifies and clarifies deviations from normal sensor interactions. Similarly, Zhang et al. [269] focus on sensor dependence relationships in cyber-physical systems, utilizing a VAE for feature extraction and a GNN with stochastic learning to capture inter-sensor dependencies. Building upon these methodologies, FuSAGNet [270] combines Sparse Autoencoder with GNN for anomaly detection in high-dimensional time series, which optimizes both reconstruction and forecasting while explicitly modeling the relationships within multivariate time series.

D. GI-Assisted Federated Learning

In certain FL scenarios, clients exhibit relational dependencies, a structural representation of which can be established through a graph. For instance, traffic sensors in physical proximity often result appear in similar traffic patterns. In this case, a cross-client graph can be established by identifying them as respective vertices. The presence of this cross-client graph prompts the utilization of GI algorithms to enhance the FL training process, resulting in the paradigm of GI-assisted FL (GFL), as shown in Fig. 16. As we have discussed in Section VI-A, GFL and FGL are different mechanisms, where the former applies GI techniques to optimize FL on arbitrary ML models (which do not necessarily be GI models) and the latter applies FL to jointly train GI models. In other words, GI models are optimization tools in GFL, whereas they are the targeted models to be trained in FGL. Due to such distinct functionality, we categorize GFL into the GI-based optimization for edge networks and discuss it in this subsection, while FGL will be introduced in the context of edge computation of GI models in Section VI-A.

GFL works upon the premise that clients closely connected in the graph likely share similar data distributions. Based on that, GNNs can utilize the graph structure inherent in the FL system to tackle the non-IID data heterogeneity among clients. With respect to the utilization of centralized servers, we discuss GFL in the centralized setting and the decentralized setting, respectively.

1) Centralized Federated Learning: Under the centralized setting for GFL, a centralized server is employed to coordinate FL among clients, as depicted in Fig. 16(a). The GI model, which is applied for optimizing the FL procedure based on cross-client graphs, can be executed either on the server or the clients.

For the case of server-side GI execution, a GI model undergoes training on the server using graph relationships between clients, operating under the assumption that proximate clients exhibit similarities in their local models or feature embeddings. Initially, the server initiates the collection of parameters from clients, following a standard FL protocol. These uploaded parameters collected from client models are construed as node features within the context of the cross-client graph, using which the server trains a GI model to facilitate the aggregation process in FL. Note that the specification of the cross-client graph may either be predefined or dynamically extracted through the application of a self-attention module during the training phase [271]. The resultant updated parameters are then transmitted back to the respective clients.

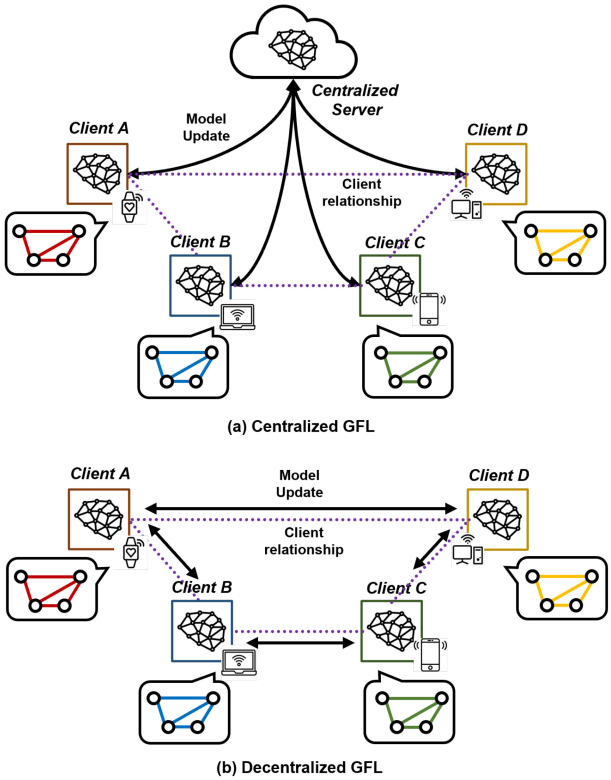


Fig. 16. GI-assisted FL employs GI models as a tool to orchestrate FL clients with relationships and can be categorized into (a) centralized settings and (b) decentralized settings.

In this process, researchers have proposed different routines to tackle different issues in training the coordinated GI model. For instance, some researchers focus on the convergence speed of GI models under non-IID circumstances, designing bi-level optimization to train both the client models and the server-side GI model simultaneously, e.g., using unsupervised contrastive learning [272] or regularization-aided supervised learning methods [273]. Others ponder the specific functionality of the GI models and develop training strategies in various aspects [274], [275].

For the case of client-side GI execution, each client is assumed to possess a global cross-client graph and trains not only its local model but a GI model to obtain global knowledge from other clients. Building different graphs within the clients can be employed to address various problem-solving objectives like data heterogeneity and model heterogeneity. For instance, FedCG [276] establishes a fully connected graph by leveraging the similarity among clients' model weights or pattern features. Each client undertakes the training of a GNN using the cross-client graph and derives a global embedding. Subsequently, the local model embedding and the global embedding are amalgamated by applying a trainable weight, which enhances the capacity of the model to incorporate both local and global information, addressing data heterogeneity.

2) Decentralized Federated Learning: Contrary to the centralized setting, decentralized GFL allows clients to communicate with their neighbors but does not have a central server to coordinate them (Fig. 16(b)), making FL model aggregation a key challenge for global model convergence. To address that, existing literature usually path two ways, i.e., weight summation across clients and graph regularization.

Weight summation methods update clients' local models by aggregating their neighbors' local model parameters following the global cross-client graph topology. Under this principle, DSGT [277] employs decentralized stochastic gradient tracking to achieve faster convergence, and PSO-GFML [278] selectively exchanging local model parameters with the servers to improve communication efficiency. The weights of the cross-client graph's links are determined based on the similarity between unlabeled graph embeddings [279] or hidden parameters [280].

Graph regularization integrates graph Laplacian regularization into the objective function to transform model parameters from neighboring clients [281] and is particularly beneficial to multi-task GFL. In dFedU [282], a pre-defined fully connected cross-client graph is given ahead where each client is assigned a single task. When receiving local updated models from neighboring clients, each client conducts model updating with graph regularization. He et al. [283] alternatively posit that each client runs multiple tasks and initializes a cross-client task relationship graph with task classifier parameters, where decentralized periodic averaging SGD is employed to optimize the objective function.

Lessons Learned (Section VII)

GI can serve as an optimization tool for enhancing different functionalities of edge networks, including network resource management, graph-based decision-making, security and anomaly detection, and GFL. To well harness GI in these cases, there are dual knobs: on the one hand, the system needs to concisely identify the optimization constraints and objectives from graph perspectives, where one or more graph(s) should be built to represent the structure of the targeted problem; on the other hand, the graph may associate appropriate (encoded) properties that supply sufficient information and convert optimization objectives into learning objectives, such that GI models can use these graph semantics to induce expected output.

VIII. Edge Graph Intelligence Ecosystems

After reviewing the interaction between GI and edge networks in Sections VI and VII, we discuss the EGI ecosystem inflated from this loop. Fig. 17 shows the landscape of the EGI ecosystem from levels of hardware, software, benchmark, and applications.

A. Edge Hardware for EGI

Edge networks cover a wide spectrum of edge platforms from embedded and mobile devices to vehicles and edge servers, as discussed in Section III-B2. At the core of these edge facilities, different processors deliver different capabilities and require different optimizations for GI model computation. In general, they can be discussed in three types,

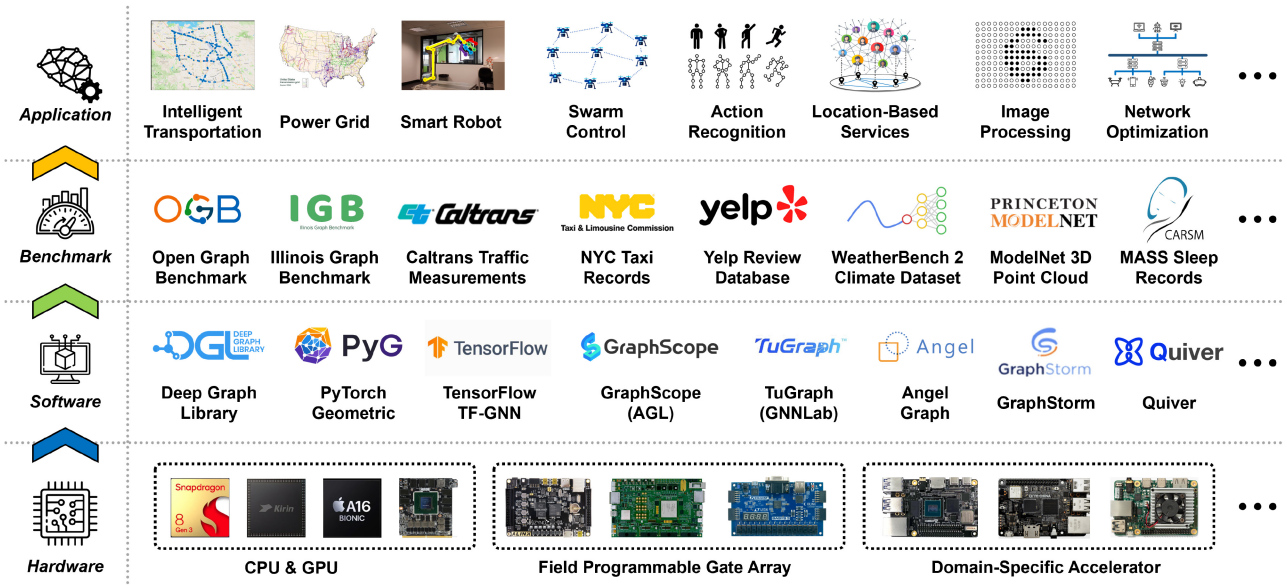


Fig. 17. The EGI ecosystem includes all stakeholders related to EGI from hardware, software, benchmark, and applications. The hardware involves edge hardware for sustaining EGI computation, covering CPU, GPU, FPGA, and DSA. The software indicates development frameworks for programming and deploying EGI models such as DGL and PyG. The benchmarks provide datasets and unified toolkits for evaluating dedicated EGI models and services. The applications show a rich set of scenarios where EGI actively works.

mobile CPUs and GPUs, Field Programmable Gate Array (FPGA), and Domain-Specific Accelerators (DSAs).

1) CPU and GPU: A majority of edge facilities, especially wearable and mobile devices, are equipped only with conventional CPUs and GPUs such as Qualcomm SnapDragon series and Huawei Kirin series. Given the intensive workload of GI-driven applications, efficient execution of them requires meticulously exploiting the capability of low-power processors. To better understand the characteristics of GI models on conventional platforms, Huang et al. [284] examine in detail representative GI systems and reveal five major gaps in optimizing GI computation performance, i.e., poor data locality, severe workload imbalance, redundant memory access, large memory footprint, and inefficiency on variant feature lengths. Based on these observations, Huang et al. [284] propose a set of tailored optimizations to bridge the gaps and achieve 1.37×-15.5× performance improvement on various GI models. Towards the same objective, Zhang et al. [285] study the GI execution performance on commercial platforms from the perspective of computational graph and summarize three issues that existing systems suffer, namely redundant neural operator computation, inconsistent thread mapping, and excessive intermediate data. To tackle these challenges, Zhang et al. [285] introduce three corresponding designs, i.e., propagation-postponed operator reorganization, unified thread mapping for fusion, and intermediate data recomputation, achieving notable speedup in lower memory IO and footprint.

These empirical studies motivate literature [286], [287], [288], [289], [290] in efficiently leveraging the potential of conventional CPUs and GPUs. For instance, Adiletta et al. [291] study a detailed characterization of GI models on Intel's Programmable Integrated Unified Memory Architecture (PIUMA) for scalable training. Graphite [287] addresses GI model training on CPUs by eliminating the memory limitation through software-hardware co-design including layer fusion, feature compression, and a vertex processing reordering algorithm for improving temporal locality, and implements a high-performance GI computation system on Intel CPUs. Some works explore optimizing matrix computation in GI tasks from the kernel level. FeatGraph [292] provides efficient implementations of Sampled Dense-Dense Matrix Multiplication (SDDMM) and Sparse-dense Matrix Multiplication (SpMM) for both CPUs and GPUs, and FusedMM [286] further develops a general solution for different GI programming backends. There are also many comprehensive systems developed to tackle the various aspects of GI training and inference for distributed training on CPU or GPU clusters (e.g., Dorylus [289], FlexGraph [290]). However, most of them rely on the computation infrastructure at datacenter levels and are thus out of the scope of this survey. Interested readers may refer to recent reviews [293], [294] on large-scale distributed GI systems.

2) Field-Programmable Gate Array: FPGA is a type of integrated circuit with a differentiated ability to be (re)programmed to desired functionality requirements after manufacturing. This advanced ability allows FPGA to be used in applications with requirements on flexibility, speed, and parallelism, and thus are embedded and employed in edge platforms for GI workload processing. HP-GNN [295] generates high-throughput GNN training implementations on a given CPU-FPGA platform by designing an automatic algorithm that maps GI training algorithms, GI models, and hardware specifics to the targeted CPU-FPGA platform. GenGNN [296] develops a generic GI model acceleration framework using High-Level Synthesis (HLS) techniques, aiming to render GI inference in low latency and support various GI models in flexible extensibility. It embraces an optimized message-passing structure to accommodate the majority of popular GI

models and provides a rich library of model-specific components. GraphAGILE [297] designs a hardware module named adaptive computation kernel for efficient execution of GI matrix operations such as sparse-dense matrix multiplication and sampled dense-dense matrix multiplication, and proposes a set of compiler optimizations to reduce inference latency. Graph-OPU [298] proposes an FPGA-based overlay processor for accelerating GI execution, which introduces microarchitecture for fully-pipelined GI processing and customizes the instruction sets for various GI model adoptions.

Besides optimizing FPGA solutions for general GI model execution, some researchers further study how to utilize FGPA-integrated platforms for specific applications efficiently. For example, Heintz and Razavimaleki [299] develop an FPGA implementation for charged particle tracking based on GI models, incorporating OpenCL and hls4ml in an interaction network architecture. Zhang et al. [300] consider synthetic aperture radar-assisted remote-sensing image recognition tasks and design an FPGA acceleration solution with customized data path and memory organization for various GI model kernels. To further speed up GI inference, it exploits FPGA's high bandwidth memory for loading data and storing intermediate results, as well as a splitting kernel technique to improve the on-chip routability and frequency. Huang et al. [301] specify an FPGA-based parallel particle tracking system with memory-aware data allocation and buffer arrays and collision-aware scheduling on parallel events.

3) Domain-Specific Accelerator: Domain-Specific Accelerators (DSAs) are important computing components and becoming more pervasive in many edge facilities such as dedicated edge servers and some mobile devices [302], [303], [304], [305]. By optimizing specifically for a narrow range of computations, DSAs are tailored to meet the needs of targeted algorithms in dedicated domains and enable orders of magnitude improvements in performance or cost compared to general-purpose processors. Motivated by the achieved success of DSAs in CNNs, the community has made plentiful efforts in developing DSAs for GI workload. Towards that, EnGN [306] presents a unified architecture inspired by CNN accelerators, where GI computations are viewed in a matrix perspective and matrix multiplication optimizations are applied for improved runtime performance. Auten et al. [307] propose a modular DSA architecture for convolutional GNNs, where each tile as a basic unit is composed of an aggregation module, a DNN accelerator module, a DNN queue, and a graph processing element.

Nonetheless, GNNs are semantically different from traditional DNNs (c.f. Section III-A), bringing new computing characteristics desired for tailored designs. To better understand such computing characteristics, Yan et al. [308] and Lin et al. [309] respectively conduct detailed profiling of general GNN workload and distributed GNN training workload on GPUs, providing useful insight on GNN-oriented DSA design. Based on that, HyGCN [310] introduces a hybrid architecture to tame the alternating phases of GI computation, which is composed of separate dedicated engines for the aggregate and update functions as well as a control scheme to pipeline the execution of both functions. GRIP [311] leverages the programming abstraction of GReTA [312] to develop a general DSA for GNNs, which organizes a GI model inference into four steps, i.e., gather, reduce, transform, and activate. Regarding this abstraction, GRIP specializes in the processing units for links and vertices separately and designs a control module to coordinate data movement between units and buffers. AWB-GCN [313] raises an aggressive adaptation to the structural sparsity of GI models under the motivation that for power-law graphs, GI computation can be polarly dense or sparse and suffers from workload imbalance. To address this imbalance, AWB-GCN [313] develops a custom matrix multiplication engine with autotuning workload balancing techniques, and GNNIE [314] advocates a flexible MAC architecture that splits features into blocks with load (re)distribution and graph-specific caching to bypass the high costs of random DRAM.

B. Development Frameworks

GI models' unique computing workflow introduces different programming patterns from traditional DL models (e.g., CNNs and RNNs), which requires dedicated frameworks for implementation. Towards that, the community has developed various programming frameworks compatible with edge computing platforms. Among them, PyTorch Geometric (PyG) [315] and Deep Graph Library (DGL) [316] are the two most popular ones and are widely used in various applications. PyG is a geometric deep learning extension library built upon PyTorch that provides flexible interfaces for programming GI models on both CPUs and GPUs. It adopts a message-passing programming abstraction for building GI models, which is programmer-friendly to express various GI variants. DGL is another general framework specialized for deep learning models on graphs. It also leverages the message-passing primitives in a user-configurable way and introduces a set of parallelization strategies for high-performance GNN execution. Both PyG and DGL can be used in edge devices and edge servers as long as they enable the PyTorch library in their execution environments. In addition, DGL also supports TensorFlow and MXNet, and edge platforms that run these two frameworks also allow model execution with DGL. Besides PyG and DGL, the industry has also developed many frameworks upon their customized demand, such as AGL [317], BGL [318], TF-GNN [319], and Angel-Graph, etc. Nonetheless, most of them are dedicated to cloud-level resources without consideration of the deployment on edge networks and thus require tailored adjustments when building edge GI services.

C. Open Datasets

To assess how well edge GI systems perform and learn their behaviors upon deployment, versatile datasets are collected from diverse edge scenarios, serving as performance benchmarks. Table III lists some representative datasets in several application domains discussed in Section V.

The first dataset series lies on smart cities, including data collected from intelligent transportation systems, meteorological stations, power grids, and governments. Their graphs are mostly derived from the geographical network relations in

TABLE III
SELECTED OPEN DATASETS FOR EGI APPLICATIONS

Domain	Dataset	Graph	Description	Link
Smart Cities	PeMS	Vertex: Traffic sensor Link: Road nets	Traffic sensory data collected by CalTrans Performance Measurement System in California.	[321]
	METR-LA	Vertex: Traffic sensor Link: Road nets	Traffic sensory data of loop detectors in the highway of Los Angeles.	[322]
	NYC Taxi	Vertex: Taxi Link: Road nets	Origin-Destination demand dataset with taxi trip records capturing pick-up and drop-off locations, trip distances, payment types, etc.	[323]
	Chicago Bike	Vertex: Bike Link: Road nets	Origin-Destination demand dataset with bike trip records capturing trip start/end time, station, and rider type.	[324]
	Beijing Air Quality	Vertex: Region Link: Geographical adjacency	Air quality data of distributed monitoring sites from Beijing Municipal Environmental Monitoring Center.	[325]
	WeatherBench2	Vertex: Region Link: Geographical adjacency	Comprehensive meteorological data for mid-range weather forecasting, including temperature, humidity, wind, and cloud cover, etc.	[326]
	PowerGraph	Vertex: Energy station Link: Power Grid	Electrical power grid dataset that models cascading failures in power grids.	[327]
	COVID-19	Vertex: Region Link: Geographical adjacency	Data repository of 2019 Novel Coronavirus visual dashboard, which collects global COVID-19 infectious data.	[328]
Intelligent Robots & Vehicles	US-101	Vertex: Vehicle Link: Vehicle Connection	Vehicle trajectory data on southbound US 101 that provides the precise locations of vehicles every one-tenth of a second.	[329]
	Stanford Drone	Vertex: Drone Link: Drone Connection	Flight records of navigated drones that collects images and videos of various types of agents in a realistic outdoor environment	[330]
	Kinetics	Vertex: Entity Link: Entity relationship	Large-scale human action video dataset collected from YouTube, covering human-object interactions and human-human interactions.	[331]
	nuScenes	Vertex: Vehicles or pedestrians Link: Entity relationship	A large-scale, multimodal dataset for autonomous driving, containing annotated sensor data from LiDAR, radar, cameras, and GPS across diverse driving scenarios in urban environments.	[332]
	CORSMAL	Vertex: Entity Link: Entity relationship	A multimodal dataset focused on robotic perception for manipulating containers, including audio, visual, and depth data to estimate container properties and contents in real-time interactions.	[333]
	RoboNet	Vertex: Entity Link: Entity relationship	Collection of robotic manipulation data across diverse objects, environments, and robotypes, designed to enable learning generalizable manipulation policies.	[334]
Human Sensing	NTU RGB+D	Vertex: Body joint Link: Human skeleton	Large-scale dataset for human action recognition that contains RGB videos, depth map sequences, and 3D skeletal data. data.	[335]
	MobiAct	Vertex: Body joint Link: Human skeleton	Human action recognition dataset with acclerometer, gyroscope, and orientation sensory data	[336]
	TST V2	Vertex: Body joint Link: Human skeleton	Fall detection dataset with depth frames, skeleton joints, acceleration sensory data	[337]
	MASS-SS3	Vertex: Monitoring channel Link: Channel connection	Polysomnography recordings collected in a lab-based environment for sleep quality prediction.	[338]
	AffectNet	Vertex: Action unit Link: FACS-based link	Large-scale facial expression analysis dataset collected from the Internet.	[339]
Locations -Based Social Recommendation	Gowalla	Vertex: User Link: Friendship	Location-based social networks where users share their locations by checking-in.	[340]
	Foursquare	Vertex: User and Hotel Link: Visiting history	Location-based digital footprints from Foursquare for personalized location recommendation and search.	[341]
	SIoT	Vertex: IoT device Link: Social connection	Network of IoT devices with social connections with identity information including device type, brand, and ownership.	[342]
	Yelp	Vertex: User comment Link: Comment history	Users' review comments of POIs from Yelp with location, purchasing, and social information.	[343]
Mobile Vision	MNIST	Vertex: Pixel Link: Pixel adjacency	Handwritten digits dataset, where pixels are viewed in grid-structure graphs.	[344]
	DAVIS	Vertex: Entity Link: Entity relationship	Video object segmentation dataset that consists of 50 videos in total with pixel-wise annotations for every frame.	[345]
	YouTubeVIS	Vertex: Entity Link: Entity relationship	Video instance segmentation dataset that contains video clips from YouTube with segmentation masks, instance identity labels, etc.	[346]
	KITTI	Vertex: Point Link: Point adjacency	Point cloud dataset of various kinds of objects for 3D object detection.	[347]
	ModelNet40	Vertex: Point Link: Point adjacency	Synthetic object point clouds with CAD-generated meshes in various categories.	[348]

cities, e.g., road nets, power networks, and regional adjacencies. The second domain is intelligent robotics and vehicles, taking datasets from various robot manipulation scenarios such as UAV swarms and vehicle trajectories The third category is for human sensing applications, including human action recognition, facial affective analysis, and sleep quality prediction.

Specifically, for the sleep quality dataset, researchers construct sleep stage graphs with electronic channel records and employ GI models for prediction. The fourth domain is location-based social recommendation, where social networks are naturally graphs and the location information of users puts its analysis into the context of edge networks. The last group focuses on mobile vision, where the datasets are also widely utilized in many computer vision models. Readers are encouraged to exact their interested datasets from the links to learn more about their sources and applications.

Lessons Learned (Section VIII)
The promising fusion of EGI facilitates the EGI ecosystem in a full-stack manner. Specifically, the ecosystem provides an abundant set of facilities covering hardware, software, and testing benchmarks, acting as a stage for developing and deploying EGI services. Developers are suggested to develop their EGI services agilely with well-established open-sourced toolkits.

IX. Open Challenges and Future Research Opportunities

The confluence of GI and edge networks is still in its infancy stage and many of the considered parts of the EGI landscape are yet under exploration. In this section, we articulate key open challenges and future research directions for EGI.

A. New Promising Applications and Optimizations

The integration of GI and edge networks opens up new possibilities for low-latency, context-aware analysis, and decision-making, holding great potential for miscellaneous promising applications. Given the ubiquitous nature of graph data, there are still many areas to be explored for service providers, network operators, and end users with optimization opportunities as well as business revenues.

1) New EGI Applications: Besides applications discussed in Section V, there are still many burgeoning edge scenarios well matching EGI's capability. For instance, in digital twin networks, the mapping between physical objects and digital twins can be regarded as graphs and EGI may be applied for task offloading, power allocation, and network slicing [254], [348], [349]. Satellite-aerial-ground integrated networks are also one of the emerging edge networks [350], [351], [352], where high-altitude platform stations and ground stations constitute vertices with their communication channels as links. EGI thus can also be used for them.

2) B5G and 6G Protocols Design: Communication in edge networks, particularly for cross-device services, often involves complex interactions that can be naturally represented as graphs. The networking topology, defined by communication nodes, forms the vertices, while relevant data such as channel capacity, queuing delays, and link states can be modeled as graph properties. This representation opens up opportunities to apply GI for developing advanced communication protocols, such as those for B5G and 6G networks, enabling smarter, faster, and more energy-efficient data transmission. For instance, GNNs can be utilized to dynamically optimize routing strategies and detect intermittent anomalies within edge networks [30], [353]. Furthermore, communication channel states can be incorporated as link attributes in graph models, allowing GI methods to enhance network management and resource allocation [241], [244], [245], [246].

3) Personalized Services: End users nowadays are immersively surrounded by a vast amount of IoT devices like mobile smartphones, wearable kits, and smart home assets. For individual users, we can weave a user-centric network [354] by collecting all user-related data from these surrounding devices and connecting them in a logic graph, on which EGI techniques can be applied to process, understand, and learn. In other words, EGI can serve as a tool to summarize data points scattered around individual users and be leveraged as a core component to build everyone-centric customized services.

B. Edge-Centric Graph Learning Models and Systems

Many existing GI models are originally designed to improve accuracy on specific tasks. In real deployment, however, a set of SLOs beyond accuracy are required by service providers such as responsive latency, system energy, and data privacy, and implementing GI models at edge faces unique challenges. Note that the graphs in edge networks may not be as huge as social networks on Twitter [355]. However, it does not necessarily mean that training and deploying GNN on edge networks is technically trivial.

1) Resource Constraints to GI Models: As we discussed in Section VI, edge devices typically have limited but heterogeneous computational capabilities compared to cloud servers, possessing platforms from MCUs to mobile smartphones to edge servers. Edge-centric GI models need to be lightweight, have acceptable memory footprints, and minimize computational complexity while still delivering accurate results, especially for EGI-based services with real-time or near real-time requirements. This directs innovations in both model level and system level, e.g., lightweight algorithms like computation deduplication [356] and memory-efficient on-device systems [102]. Besides, fine-grained characteristics of GI models also expect precious profiling for refined resource utilization.

2) System Dynamics: There are two types of system dynamics, namely resource dynamics and graph dynamics. On the one hand, edge devices frequently operate in dynamic and unpredictable resource conditions, including intermittent connectivity, dynamic computing capabilities, or fluctuated communication bandwidths [357], [358]. A stable, satisfactory EGI system is obliged to be robust to these variations and capable of adapting to different network conditions to ensure reliable and timely analysis, especially for those services that process graph data across multiple edge devices. On the other hand, many real-world graph data can be dynamic in both properties and topologies, requiring both algorithmic and system flexibility to adapt to the input variation. Some

work [210], [211] exploit online incremental adjustment for minor input variation, where drastic changes like graph reconstitution are still under exploration.

3) Scaling to Large Graphs: Many real-world networks represented as graphs can be massive and evolve in size, but how to scale EGI systems to large and heavy-attributed graphs poses significant challenges on both the algorithm and system sides. From the algorithm perspective, large graph models borrow wisdom from LLM and incorporate pre-training methods to embrace graphs on giant scales, but also bring overload stress to edge networks. Distributed edge collaboration (cf. Section VI-B), which aggregates multiple edge devices as a resource pool for model computation, can be a potential solution but still requires tailored designs with parallel processing, graph partitioning, and incremental learning, etc [359]. From the system perspective, orchestrating a tremendous volume of graph data through edge network channels tackles the communication bottleneck as well as the storage capacity of edge devices. Data compression like graph pruning and feature quantization can be helpful, but how to balance the tradeoff between model accuracy and execution latency requires careful design.

C. Emerging Learning Paradigms

Data generated at the network edge are massive but also usually in varying quality. To improve the applicability and usability of EGI models, emerging learning paradigms should be utilized to tackle different data cases. While some have been adopted (e.g., Federated Learning, Reinforcement Learning, etc.), there are still many that can be exploited and we discuss three below.

1) Large Graph Model: LGMs borrow the wisdom of LLMs to the graph domain that simultaneously scales both training data and models. This scaling mechanism envisions LGMs to be capable of capturing complex relationships and patterns within large-scale graphs, which are common representations of data in various domains such as social networks, biological networks, and transportation systems [56], [58]. Nonetheless, applying LGM in edge networks severely suffers from the constrained computing capability and network capacity. Edge-cloud collaboration, which marries the local data processing ability of edge networks and the powerful computing power of the cloud, can be a potential solution for deployment. Given the promising ability of large models, LGMs desire more effort in the context of EGI. On the other hand, EGI may also explore the translation between graphs and natural language to exploit the capability of LLMs, e.g., by semantically converting a graph into a paragraph of sentences for LLM input and generation [57].

2) Learning With Small Data: Edge devices can collect a large volume of data but only with a small portion of labels, which demands EGI to be capable few-shot learners. Towards that, few-shot learning techniques can be promising, which aims at training GL models for accurate predictions on unseen classes or categories with limited labeled data. For example, in some EGI-based object detection tasks [163], GI models can learn to recognize novel objects with a few labeled examples by leveraging information from similar objects and their relationships in a graph representation. Another potential path is transfer learning, which borrows knowledge learned from a source domain or task to improve performance in a target domain or task. An instance of transfer graph learning is anomaly detection in networked systems [263], [264], where GI can learn patterns of fraud by pre-training on a large graph and transfer this knowledge to identify fraudulent activities in a target domain with limited labeled data. TL can also be used for STGNN streams, e.g., for traffic prediction in smart cities [360].

3) Continual Learning: Continual Learning (CL) for GI is to incrementally retrain GNN models to adapt to new data or tasks without forgetting previous knowledge. It also adjusts the learned latent distribution like Transfer Learning but emphasizes learning patterns from the additional incoming data while preserving previously learned representations. For EGI services, CL enables incremental service refinement for evolved graphs, e.g., in temporal graph data analysis tasks, without training GI models from scratch. Examples include predicting traffic conditions in a road network and running the status of power grids, where continual learning of GI models allows the model to adapt over time and incorporate new temporal information while retaining the knowledge learned from past time steps.

D. Native Edge Support for Graph Intelligence

Edge networks serve as the infrastructural support of GI model computation, for which an EGI-native pipeline is crucial for efficient and effective EGI services provisioning. Despite its importance, research on EGI-native edge support is yet limited in the context, and it confronts challenges in the full lifespan of GI model computation, i.e., from data collection to data processing to system development.

1) Graph Data Collection: EGI applications can not be realized without pervasive graph data. For many applications, however, these graph data are generated and distributed in different places, e.g., traffic sensory data scattered in road networks and perception information gathered by vehicles in robotic swarms. How to collect these (geographically) distributed data over the edge networks efficiently, i.e., in a high-quality, low-latency, and privacy-preserved manner, thereby becomes an unavoidable problem in running GI-based services. In certain situations, graph generation techniques [361] may be a useful supplement for graph data requirements.

2) Edge-Cloud Collaboration: Edge networks render computing power in versatile forms, e.g., by a single edge device, a pool of collaborated ones, and even a synergetic edge-cloud continuum. Provisioning these versatile resources efficiently for EGI applications not only requires flexible resource management but also judicious workload scheduling that well aligns with computation and communication. In particular, edge-cloud collaboration provides a combined mechanism to embrace both data locality at the edge and resource richness in the cloud, which allows GI to be more efficient and sustainable and attracts growing attention from the community. Possible

means towards this objective include designing GI-oriented communication protocols, GI-specific computation scheduling mechanisms, and a holistic edge network cost optimization methodology.

3) System Deployment: Given the broad use cases of EGI, their development yet lacks a comprehensive framework with full-stack toolkits, particularly for distributed edge collaboration and edge-cloud synergy. This requires addressing the complexity of distributed systems, including edge device coordination, data synchronization, and fault tolerance, and also tackling long-term interoperability and testing issues. Existing virtualization techniques such as container and function computing can be further explored for friendly EGI development and deployment. The communication aspects, such as timely data transmission of (potentially distributed) graph data and graph modeling of network channels, also desire tailored consideration.

4) Supporting Large Models: As an edge-oriented methodology, EGI can support large models in the following angles. First, EGI provides an effective way to abstract edge data in the form of graphs, acting as a graph data processing module for large model deployment at the edge. For instance, we may use EGI to transform graph data into prompts acceptable by large language models, allowing language intelligence on edge graphs. Second, with EGI models, edge data can be abstracted into graph embeddings. These embeddings can be used as a knowledge base for multi-modal foundation models aiming at specific edge scenarios, i.e., enabling retrieval-augmented generation. Third, EGI can also be operated as an optimization tool to orchestrate resource allocation for large model computation. To deploy resource-hungry large models on edge platforms, we may utilize many devices in edge networks, where EGI can be a scheduler, as discussed in Section VII-A, to schedule efficient resource allocation for large model acceleration.

E. Explainability, Security, and Privacy

The open nature of EGI requires trustworthiness across different entities in the ecosystem to make EGI happen. Nonetheless, in many scenarios, edge services are practical only if they are reliable. For EGI, this relates to explainability, security, and privacy.

1) Explainability: Explainability makes GI models predictable and understandable, which is crucial for some action-critical applications such as crime warnings and autonomous vehicle control. However, taming EGI models' behaviors is challenging, which usually involves specific knowledge in their application scenarios. Some literature has explored explaining GI models' behaviors in general with mathematical tools like Weisfeiler-Lehman graph isomorphism test [362], [363], [364], but those GI models dedicated designed for edge scenarios still lack investigation.

2) Security: Security holds a dual significance in the context of EGI services. From a model perspective, GI models must be both secure and robust, as they are particularly vulnerable to attacks when malicious vertices are present in the graph input. The risk is further amplified when input graph data is aggregated from multiple entities, increasing the likelihood of fraudulent activity, and underscoring the importance of developing and implementing secure GI models. From a system perspective, edge networks are also susceptible to malicious devices, which can disrupt the learning process or compromise data integrity. This calls for the establishment of verifiable trust mechanisms among edge networks, alongside the development of secure tools to defend against and recover from such attacks.

3) Privacy: Edge networks typically operate in close proximity to individuals, collecting data from edge devices that may include personal and sensitive information, such as geographic location, health and activity records, and electricity consumption patterns [358]. In light of privacy protection regulations, such as the European Union's General Data Protection Regulation (GDPR), the sharing and processing of this data across edge service providers poses significant risks of privacy leakage and unforeseen legal liabilities. Consequently, this raises critical research questions regarding not only secure but privacy-preserving computation in EGI models, where cryptographic techniques such as differential privacy and homomorphic encryption can be utilized to mitigate these risks.

X. Conclusion

Recent advances in ML have extrapolated representation learning techniques to the graph domain, cultivating GI as a powerful tool to learn from graphs. Meanwhile, edge computing networks are thriving with the rapid proliferation of edge facilities and becoming a fundamental infrastructure of miscellaneous user-centric intelligent services. These game-changing trends push GI and edge networks to a close-looped confluence, where edge networks serve as infrastructural support to GI-based tasks and GI conversely serves as a key enabler to build and optimize edge services. Their complementary interaction raises a promising paradigm, i.e., Edge GI or EGI for brevity, to model, abstract, and analyze the ubiquitous graph data in edge networks, flourishing versatile graph-based applications at the network edge.

In this paper, we advocate EGI as a brand-new principle in the context of edge intelligence by revealing its motivational benefits, conceptual scope, and rating principle. Based on that, we conduct a comprehensive survey of recent research efforts in this emerging field from multiple dimensions. We first provide primers on GI and edge networks and summarize miscellaneous edge applications built with GI models. Next, from the perspectives of "edge for GI" and "GI for edge", we thoroughly review related concepts, techniques, and systems on computing GI models on edge networks and applying GI for optimizing edge networks, respectively. We finally present the overview of EGI ecosystems, discuss future directions, and conclude. We hope that this survey can garner increased attention, foster meaningful discussions, and inspire future research in EGI.

REFERENCES

[1] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[2] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7806–7819, Nov. 2021.

[3] R. Xu, S. Razavi, and R. Zheng, "Edge video analytics: A survey on applications, systems and enabling techniques," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 2951–2982, 4th Quart., 2023.

[4] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[5] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022.

[6] B. Ji et al., "Survey on the Internet of Vehicles: Network architectures and applications," *IEEE Commun. Stand. Mag.*, vol. 4, no. 1, pp. 34–41, Mar. 2020.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, May 2021.

[10] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, no. 1, pp. 57–81, 2020.

[11] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing graph neural networks: A survey from algorithms to accelerators," *ACM Comput. Surveys*, vol. 54, no. 9, pp. 1–38, 2021.

[12] M. Besta and T. Hoefler, "Parallel and distributed graph neural networks: An in-depth concurrency analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 2584–2606, May 2024.

[13] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[14] I. R. Ward, J. Joyner, C. Lickfold, Y. Guo, and M. Bennamoun, "A practical tutorial on graph neural networks," *ACM Comput. Surveys*, vol. 54, no. 10s, pp. 1–35, 2022.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Rep. (ICLR)*, Apr. 2017, pp. 1–9.

[16] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[17] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge Ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep./Oct. 2019.

[18] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. 6th Int. Conf. Learn. Rep. (ICLR)*, 2018, p. 8.

[19] X. Wang et al., "Traffic flow prediction via spatial–temporal graph neural network," in *Proc. Web Conf.*, 2020, pp. 1082–1092.

[20] T. Zhong, S. Zhang, F. Zhou, K. Zhang, G. Trajcevski, and J. Wu, "Hybrid graph convolutional networks with multi-head attention for location recommendation," *World Wide Web*, vol. 23, pp. 3125–3151, Jun. 2020.

[21] B. Chang, G. Jang, S. Kim, and J. Kang, "Learning graph-based geographical latent representation for point-of-interest recommendation," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, 2020, pp. 135–144.

[22] H. Jeon, D. Kum, and J. Choi, "SCALE-net: Scalable vehicle trajectory prediction network under random number of interacting vehicles via edge-enhanced graph convolutional neural network," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2095–2102.

[23] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "AST-GNN: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction," *Neurocomputing*, vol. 445, pp. 298–308, Jul. 2021.

[24] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6519–6528.

[25] M. Khodayar, S. Mohammadi, M. E. Khodayar, J. Wang, and G. Liu, "Convolutional graph autoencoder: A generative deep neural network for probabilistic spatio-temporal solar irradiance forecasting," *IEEE Trans. Sustain. Energy*, vol. 11, no. 2, pp. 571–583, Apr. 2020.

[26] G. Jin, Y. Liang, Y. Fang, Z. Shao, J. Huang, and J. Zhang, "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 10, pp. 5388–5408, Oct. 2024.

[27] Y. Li, D. Yu, Z. Liu, M. Zhang, X. Gong, and L. Zhao, "Graph neural network for spatiotemporal data: Methods and applications," 2023, *arXiv:2306.00012*.

[28] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep with graph neural networks: A comprehensive review of algorithms and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 15051–15071, Nov. 2024.

[29] M. Nie, D. Chen, and D. Wang, "Reinforcement learning on graphs: A survey," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1065–1082, Aug. 2023.

[30] W. Jiang, "Graph-based deep learning for communication networks: A survey," *Comput. Commun.*, vol. 185, pp. 40–54, Mar. 2022.

[31] K.-H. N. Bui, J. Cho, and H. Yi, "Spatial–temporal graph neural network for traffic forecasting: An overview and open research issues," *Appl. Intell.*, vol. 52, no. 3, pp. 2763–2774, 2022.

[32] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, "Graph neural networks for intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8846–8885, Aug. 2023.

[33] W. Liao, B. Bak-Jensen, J. R. Pillai, Y. Wang, and Y. Wang, "A review of graph neural networks and their applications in power systems," *J. Mod. Power Syst. Clean Energy*, vol. 10, no. 2, pp. 345–360, 2021.

[34] M. Abdelmalak, V. Venkataramanan, and R. Macwan, "A survey of cyber-physical power system modeling methods for future energy systems," *IEEE Access*, vol. 10, pp. 99875–99896, 2022.

[35] B. Huang and J. Wang, "Applications of physics-informed neural networks in power systems—A review," *IEEE Trans. Power Syst.*, vol. 38, no. 1, pp. 572–588, Jan. 2023.

[36] S. He et al., "An overview on the application of graph neural networks in wireless networks," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 2547–2565, 2021.

[37] G. Dong et al., "Graph neural networks in IoT: A survey," *ACM Trans. Sensor Netw.*, vol. 19, no. 2, pp. 1–50, 2023.

[38] S. K. Moorthy and J. Jagannath, "Survey of graph neural network for Internet of Things and NextG networks," 2024, *arXiv:2405.17309*.

[39] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proc. IEEE*, vol. 107, no. 8, pp. 1537–1562, Jun. 2019.

[40] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[41] L. Kong et al., "Edge-computing-driven Internet of Things: A survey," *ACM Comput. Surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[42] W. Yu et al., "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017.

[43] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[44] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surveys*, vol. 54, no. 8, pp. 1–37, 2021.

[45] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, May 2022.

[46] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Comput. Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[47] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[48] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Rep. (ICLR)*, May 2018, p. 12.

[49] C. Sun et al., "Attention-based graph neural networks: A survey," *Artif. Intell. Rev.*, vol. 56, no. S2, pp. 2263–2310, 2023.

[50] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop Bayesian Deep Learn.*, 2016, p. 12.
[51] Z. A. Sahili and M. Awad, "Spatio-temporal graph neural networks: A survey," 2023, *arXiv:2301.10569*.
[52] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 4–24.
[53] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proc. Web Conf.*, 2020, pp. 2704–2710.
[54] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 14501–14515.
[55] E. Min et al., "Transformer for graphs: An overview from architecture perspective," 2022, *arXiv:2202.08455*.
[56] J. Liu et al., "Towards graph foundation models: A survey and beyond," 2023, *arXiv:2310.11829*.
[57] Z. Zhang, H. Li, Z. Zhang, Y. Qin, X. Wang, and W. Zhu, "Graph meets LLMS: Towards large graph models," in *Proc. NeurIPS Workshop*, 2023, pp. 2–9.
[58] L. Xia, B. Kao, and C. Huang, "OpenGraph: Towards open graph foundation models," 2024, *arXiv:2403.01121*.
[59] Y. Chong, Y. Ding, Q. Yan, and S. Pan, "Graph-based semi-supervised learning: A review," *Neurocomputing*, vol. 408, pp. 216–230, Sep. 2020.
[60] Z. Song, X. Yang, Z. Xu, and I. King, "Graph-based semi-supervised learning: A comprehensive review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8174–8194, Nov. 2023.
[61] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, "Transfer learning of graph neural networks with ego-graph information maximization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1766–1779.
[62] X. Han, Z. Huang, B. An, and J. Bai, "Adaptive transfer learning on graph neural networks," in *Proc. 27th ACM SIGKDD Conf. Knowl. Disc. Data Min.*, 2021, pp. 565–574.
[63] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 5812–5823.
[64] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, 2021, pp. 2069–2080.
[65] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An empirical study of graph contrastive learning," in *Proc. Neural Inf. Process. Syst. Track Datasets Benchmarks*, vol. 1, 2021, p. 8.
[66] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Rep. (ICLR)*, Apr. 2014, p. 67.
[67] C. Chen, Z. Xu, W. Hu, Z. Zheng, and J. Zhang, "FedGL: Federated graph learning framework with global self-supervision," *Inf. Sci.*, vol. 657, Feb. 2024, Art. no. 119976.
[68] F. Chen, P. Li, T. Miyazaki, and C. Wu, "Fedgraph: Federated graph learning with intelligent sampling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1775–1786, Aug. 2022.
[69] Z. Wang et al., "FederatedScope-GNN: Towards a unified, comprehensive and efficient package for federated graph learning," in *Proc. 28th ACM SIGKDD Conf. Knowl. Disc. Data Min.*, 2022, pp. 4110–4120.
[70] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
[71] M. Pooyandeh and I. Sohn, "Edge network optimization based on AI techniques: A survey," *Electronics*, vol. 10, no. 22, p. 2830, 2021.
[72] Y. Yang, "Multi-tier computing networks for intelligent IoT," *Nat. Electron.*, vol. 2, no. 1, pp. 4–5, 2019.
[73] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, and X. Wang, "Large-scale measurements and optimizations on latency in edge clouds," *IEEE Trans. Cloud Comput.*, vol. 12, no. 4, pp. 1218–1231, Oct.–Dec. 2024.
[74] X. Wang, Y. Zhao, C. Qiu, Q. Hu, and V. C. M. Leung, "Socialized learning: A survey of the paradigm shift for edge intelligence in networked systems," *IEEE Commun. Surveys Tuts.*, early access, Oct. 17, 2024, doi: 10.1109/COMST.2024.3482978.
[75] Amazon. "AWS for the edge." Accessed: Jan. 31, 2024. [Online]. Available: https://aws.amazon.com/edge/
[76] Google. "Google distributed cloud edge." Accessed: Jan. 31, 2024. [Online]. Available: https://cloud.google.com/distributed-cloud-edge
[77] Linux Foundation. "EdgeX foundry: An open framework for IoT edge computing." Accessed: Jan. 31, 2024. [Online]. Available: https://www.edgexfoundry.org/
[78] Eclipse Foundation. "Eclipse Kura: Open source IoT edge framework." Accessed: Jan. 31, 2024. [Online]. Available: https://www.eclipse.org/kura/
[79] The Apache Software Foundation. "Apache edgent." Accessed: Jan. 31, 2024. [Online]. Available: https://incubator.apache.org/projects/edgent.html
[80] Redis. "RedisEdge: A dedicated IoT database for edge computing." Accessed: Jan. 31, 2024. [Online]. Available: https://redis.com/blog/redisedge-iot-database-for-edge-computing/
[81] TensorFlow. "On-device training with tensorflow lite." Accessed: Jan. 31, 2024. [Online]. Available: https://www.tensorflow.org/lite/examples/o_device_training/overview
[82] X. Jiang et al., "MNN: A universal and efficient inference engine," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 1–13.
[83] EdgeCloudSim Project Team. "EdgeCloudSim: An environment for performance evaluation of edge computing systems." Accessed: Jan. 31, 2024. [Online]. Available: https://github.com/CagataySonmez/EdgeCloudSim
[84] Cloudslab. "CloudSim: A framework for modeling and simulation of cloud computing infrastructures and services." Accessed: Jan. 31, 2024. [Online]. Available: https://github.com/Cloudslab/cloudsim
[85] Cloudslab. "iFogSim: Modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments." Accessed: Jan. 31, 2024. [Online]. Available: https://github.com/Cloudslab/iFogSim
[86] YAFS Project Team. "YAFS: Yet another fog simulator." Accessed: Jan. 31, 2024. [Online]. Available: https://github.com/acsicuib/YAFS
[87] S. Ye, L. Zeng, X. Chu, G. Xing, and X. Chen, "Asteroid: Resource-efficient hybrid pipeline parallelism for collaborative DNN training on heterogeneous edge devices," in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, 2024, pp. 1–15.
[88] D. Cai, Y. Wu, S. Wang, F. X. Lin, and M. Xu, "Efficient federated learning for modern NLP," in *Proc. 29th Annu. Int. Conf. Mobile Comput. Netw.*, 2023, pp. 1–16.
[89] D. Xu et al., "Mandheling: Mixed-precision on-device DNN training with DSP offloading," in *Proc. 28th Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 214–227.
[90] S. Ye et al., "Galaxy: A resource-efficient collaborative edge AI system for *in-situ* transformer inference," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2024, pp. 1001–1010.
[91] F. Jia et al., "CODL: Efficient CPU-GPU co-execution for deep learning inference on mobile devices," in *Proc. 20th Annu. Int. Conf. Mobile Syst. Appl. Services*, 2022, pp. 209–221.
[92] D. Reinsel, J. Gantz, and J. Rydning, *The Digitization of the World From Edge to Core*, Int. Data Corporat., Framingham, MA, USA, 2018.
[93] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[94] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[95] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
[96] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, 2019.
[97] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, nos. 1–7, pp. 107–117, 1998.
[98] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[99] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. IJCAI*, 2018, pp. 3634–3640.
[100] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, "Multitask offloading strategy optimization based on directed acyclic graphs for edge computing," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9367–9378, Jun. 2022.
[101] A. Swaminathan, M. Chaba, D. K. Sharma, and U. Ghosh, "GraphNET: Graph neural networks for routing optimization in software defined networks," *Comput. Commun.*, vol. 178, pp. 169–182, Oct. 2021.
[102] Z. Xue, Y. Yang, and R. Marculescu, "SUGAR: Efficient subgraph-level training via resource-aware graph partitioning," *IEEE Trans. Comput.*, vol. 72, no. 11, pp. 3167–3177, Jun. 2023.

[103] A. Zhou et al., "Hardware-aware graph neural network automated design for edge computing platforms," in *Proc. IEEE 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.

[104] Rolypolyman@Wikipedia. "United states power grid." Accessed: Dec. 4, 2023. [Online]. Available: https://en.wikipedia.org/wiki/File:UnitedStatesPowerGrid.jpg

[105] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid, "Structure aware SLAM using quadrics and planes," in *Proc. 14th Asian Conf. Comput. Vis. (ACCV)*, 2019, pp. 410–426.

[106] L. G. Anthopoulos, "Understanding the smart city domain: A literature review," in *Transforming City Governments for Successful Smart Cities*. Cham, Switzerland: Springer, 2015, pp. 9–21. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-03167-5_2

[107] T. Shelton, M. Zook, and A. Wiig, "The actually existing smart city," *Cambridge J. Regions Econ. Soc.*, vol. 8, no. 1, pp. 13–25, 2015.

[108] Uber Technologies, Inc. "Uber company information." 2024. [Online]. Available: https://www.uber.com

[109] DiDi Chuxing. "DiDi company information." 2024. [Online]. Available: https://www.didiglobal.com

[110] Lyft, Inc.. "Lyft company information." 2024. [Online]. Available: https://www.lyft.com

[111] Y. Lu, H. Ding, S. Ji, N. Sze, and Z. He, "Dual attentive graph neural network for metro passenger flow prediction," *Neural Comput. Appl.*, vol. 33, pp. 13417–13431, Aug. 2021.

[112] A. A. Makhdomi and I. A. Gillani, "GNN-based passenger request prediction," *Transp. Lett.*, vol. 16, no. 10, pp. 1237–1251, 2024.

[113] Y. Wang et al., "Gallat: A spatiotemporal graph attention network for passenger demand prediction," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, 2021, pp. 2129–2134.

[114] Y. Wang et al., "Contrastive GNN-based traffic anomaly analysis against imbalanced dataset in IoT-based ITS," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2022, pp. 3557–3562.

[115] Z. Yu and M. Hu, "Deep with graph representation for vehicle repositioning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13094–13107, Aug. 2022.

[116] W. Liao, D. Yang, Y. Wang, and X. Ren, "Fault diagnosis of power transformers using graph convolutional network," *CSEE J. Power Energy Syst.*, vol. 7, no. 2, pp. 241–249, 2020.

[117] Y. Liu, N. Zhang, D. Wu, A. Botterud, R. Yao, and C. Kang, "Searching for critical power system cascading failures with graph convolutional network," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 3, pp. 1304–1313, Sep. 2021.

[118] O. Boyaci, M. R. Narimani, K. Davis, and E. Serpedin, "Cyberattack detection in large-scale smart grids using Chebyshev graph convolutional networks," in *Proc. 9th Int. Conf. Elect. Electron. Eng. (ICEEE)*, 2022, pp. 217–221.

[119] A. M. Karimi, Y. Wu, M. Koyuturk, and R. H. French, "Spatiotemporal graph neural network for performance prediction of photovoltaic power systems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 15323–15330.

[120] M. Khodayar, G. Liu, J. Wang, O. Kaynak, and M. E. Khodayar, "Spatiotemporal behind-the-meter load and PV power forecasting via deep graph dictionary learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4713–4727, Oct. 2021.

[121] L. Chen, J. Xu, B. Wu, and J. Huang, "Group-aware graph neural network for nationwide city air quality forecasting," *ACM Trans. Knowl. Disc. Data*, vol. 18, no. 3, pp. 1–20, 2023.

[122] X. Gao and W. Li, "A graph-based LSTM model for PM2.5 forecasting," *Atmosph. Pollution Res.*, vol. 12, no. 9, 2021, Art. no. 101150.

[123] W. Mao, L. Jiao, W. Wang, J. Wang, X. Tong, and S. Zhao, "A hybrid integrated deep learning model for predicting various air pollutants," *GIScience Remote Sens.*, vol. 58, no. 8, pp. 1395–1412, 2021.

[124] H. Lin, Z. Gao, Y. Xu, L. Wu, L. Li, and S. Z. Li, "Conditional local convolution for spatio-temporal meteorological forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 7470–7478.

[125] H. Lira, L. Martí, and N. Sanchez-Pi, "Frost forecasting model using graph neural networks with spatio-temporal attention," in *Proc. AI Model. Oceans Climate Change Workshop ICLR*, 2021, p. 12.

[126] T. Stanczyk and S. Mehrkanoon, "Deep graph convolutional networks for wind speed prediction," in *Proc. 29th Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn.*, 2021, pp. 147–152.

[127] G. Jin, C. Zhu, X. Chen, H. Sha, X. Hu, and J. Huang, "UFSP-Net: A neural network with spatio-temporal information fusion for urban fire situation prediction," *IOP Conf. Mater. Sci. Eng.*, vol. 853, no. 1, 2020, Art. no. 12050.

[128] L. Xia et al., "Spatial–temporal sequential hypergraph network for crime prediction with dynamic multiplex relation learning," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, 2021, pp. 1631–1637.

[129] M. Sun, P. Zhou, H. Tian, Y. Liao, and H. Xie, "Spatial–temporal attention network for crime prediction with adaptive graph learning," in *Proc. Int. Conf. Artif. Neural Netw.*, 2022, pp. 656–669.

[130] A. Kapoor et al., "Examining COVID-19 forecasting using spatio-temporal graph neural networks," in *Proc. Int. Workshop Min. Learn. Graphs*, 2020, pp. 1–6.

[131] M. Keicher et al., "Multimodal graph attention network for COVID-19 outcome prediction," *Sci. Rep.*, vol. 13, no. 1, 2023, Art. no. 19539.

[132] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multi-robot simultaneous localization and mapping," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[133] J. D. Almeida, P. Schydlo, A. Dehban, and J. Santos-Victor, "Sensorimotor Graph: Action-conditioned graph neural network for learning robotic soft hand dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 9569–9576.

[134] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *Proc. Int. Conf. Learn. Rep.*, 2018, p. 9.

[135] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, "Deep imitation learning for bimanual robotic manipulation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 2327–2337.

[136] H. Liang, X. Lou, Y. Yang, and C. Choi, "Learning visual affordances with target-orientated deep Q-network to grasp objects by harnessing environmental fixtures," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 2562–2568.

[137] H. Chen, L. Zeng, X. Zhang, and X. Chen, "AdaDrone: Quality of navigation based neural adaptive scheduling for edge-assisted drones," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2022, pp. 548–558.

[138] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 11785–11792.

[139] J. Li, Z. Su, and Y. Qiu, "Dynamic motion planning model for multirobot using graph neural network and historical information," *Adv. Intell. Syst.*, vol. 5, no. 8, 2023, Art. no. 2300036.

[140] L. Zeng, H. Chen, D. Feng, X. Zhang, and X. Chen, "A3D: Adaptive, accurate and autonomous navigation for edge-assisted drones," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 713–728, Feb. 2024.

[141] R. Möller, A. Furnari, S. Battiato, A. Härmä, and G. M. Farinella, "A survey on human-aware robot navigation," *Robot. Auton. Syst.*, vol. 145, Mar. 2021, Art. no. 103837.

[142] T. Luo, B. Subagdja, D. Wang, and A.-H. Tan, "Multi-agent collaborative exploration through graph-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Agents (ICA)*, 2019, pp. 2–7.

[143] W. Gosrich et al., "Coverage control in multi-robot systems via graph neural networks," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2022, pp. 8787–8793.

[144] H. Zhang, J. Cheng, L. Zhang, Y. Li, and W. Zhang, "H2GNN: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3435–3442, Apr. 2022.

[145] A. Mehrabian et al., *Silent Messages*. Belmont, CA, USA: Wadsworth, 1971.

[146] A. Mehrabian and J. A. Russell, *An Approach to Environmental Psychology*. Cambridge, MA, USA: MIT Press, 1974.

[147] H.-X. Xie, L. Lo, H.-H. Shuai, and W.-H. Cheng, "Au-assisted graph attention convolutional network for micro-expression recognition," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 2871–2880.

[148] Y. Liu, X. Zhang, J. Zhou, and L. Fu, "SG-DSN: A semantic graph-based dual-stream network for facial expression recognition," *Neurocomputing*, vol. 462, pp. 320–330, Oct. 2021.

[149] R. Zhao, T. Liu, Z. Huang, D. P.-K. Lun, and K. K. Lam, "Geometry-aware facial expression recognition via attentive graph convolutional networks," *IEEE Trans. Affect. Comput.*, vol. 14, no. 2, pp. 1159–1174, Apr./Jun. 2023.

[150] J. Wang, X. Long, Y. Gao, E. Ding, and S. Wen, "Graph-PCNN: Two stage human pose estimation with graph pose refinement," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 492–508.

[151] W. Peng, X. Hong, H. Chen, and G. Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 2669–2676.

[152] S. Jin et al., "Differentiable hierarchical graph grouping for multi-person pose estimation," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 718–734.

[153] G. Dong, L. Cai, D. Datta, S. Kumar, L. E. Barnes, and M. Boukhechba, "Influenza-like symptom recognition using mobile sensing and graph neural networks," in *Proc. Conf. Health Inference Learn.*, 2021, pp. 291–300.

[154] G. Dong, M. Tang, L. Cai, L. E. Barnes, and M. Boukhechba, "Semi-supervised graph instance transformer for mental health inference," in *Proc. 20th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2021, pp. 1221–1228.

[155] Z. Jia et al., "GraphSleepNet: Adaptive spatial–temporal graph convolutional networks for sleep stage classification," in *Proc. IJCAI*, 2020, pp. 1324–1330.

[156] P. Kefalas, P. Symeonidis, and Y. Manolopoulos, "Recommendations based on a heterogeneous spatio-temporal social network," *World Wide Web*, vol. 21, no. 1, pp. 345–371, 2018.

[157] A. Salamat, X. Luo, and A. Jafari, "HeteroGraphRec: A heterogeneous graph-based neural networks for social recommendations," *Knowl. Based Syst.*, vol. 217, Apr. 2021, Art. no. 106817.

[158] H. Wang, Y. Zeng, J. Chen, Z. Zhao, and H. Chen, "A spatiotemporal graph neural network for session-based recommendation," *Exp. Syst. Appl.*, vol. 202, Sep. 2022, Art. no. 117114.

[159] Y. Luo, Q. Liu, and Z. Liu, "STAN: Spatio-temporal attention network for next location recommendation," in *Proc. Web Conf.*, 2021, pp. 2177–2185.

[160] Z. Yang, M. Ding, B. Xu, H. Yang, and J. Tang, "STAM: A spatiotemporal aggregation method for graph neural network-based recommendation," in *Proc. ACM Web Conf.*, 2022, pp. 3217–3228.

[161] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1711–1719.

[162] Y. Liu, R. Wang, S. Shan, and X. Chen, "Structure inference net: Object detection using scene-level context and instance-level relationships," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6985–6994.

[163] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5177–5186.

[164] Y. Lu, Y. Chen, D. Zhao, and J. Chen, "Graph-FCN for image semantic segmentation," in *Proc. Int. Symp. Neural Netw.*, 2019, pp. 97–105.

[165] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6247–6257.

[166] Q. Liu, Q. Chu, B. Liu, and N. Yu, "GSM: Graph similarity model for multi-object tracking," in *Proc. IJCAI*, 2020, pp. 530–536.

[167] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[168] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1800–1809.

[169] K. Zhang et al., "Linked dynamic graph CNN: Learning through point cloud by linking hierarchical features," in *Proc. 27th Int. Conf. Mechatron. Mach. Vis. Pract. (M2VIP)*, 2021, pp. 7–12.

[170] S. Ye, L. Zeng, Q. Wu, K. Luo, Q. Fang, and X. Chen, "Eco-FL: Adaptive federated learning with efficient edge collaborative pipeline training," in *Proc. 51st Int. Conf. Parallel Process.*, 2022, pp. 1–11.

[171] X. Fu, B. Zhang, Y. Dong, C. Chen, and J. Li, "Federated graph machine learning: A survey of concepts, techniques, and applications," *ACM SIGKDD Explor. Newslett.*, vol. 24, no. 2, pp. 32–47, 2022.

[172] R. Liu, P. Xing, Z. Deng, A. Li, C. Guan, and H. Yu, "Federated graph neural networks: Overview, techniques, and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 8, 2024, doi: 10.1109/TNNLS.2024.3360429.

[173] L. Zheng, J. Zhou, C. Chen, B. Wu, L. Wang, and B. Zhang, "ASFGNN: Automated separated-federated graph neural network," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 3, pp. 1692–1704, 2021.

[174] T. Zhang, C. Mai, Y. Chang, C. Chen, L. Shu, and Z. Zheng, "Fedego: Privacy-preserving personalized federated graph learning with ego-graphs," *ACM Trans. Knowl. Disc. Data*, vol. 18, no. 2, pp. 1–27, 2023.

[175] B. Wang, A. Li, M. Pang, H. Li, and Y. Chen, "Graphfl: A federated learning framework for semi-supervised node classification on graphs," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2022, pp. 498–507.

[176] C. Zhang, S. Zhang, J. James, and S. Yu, "FASTGNN: A topological information protected federated learning approach for traffic speed forecasting," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8464–8474, Dec. 2021.

[177] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 6671–6682.

[178] L. Peng, N. Wang, N. Dvornek, X. Zhu, and X. Li, "FedNI: Federated graph learning with network inpainting for population-based disease prediction," *IEEE Trans. Med. Imag.*, vol. 42, no. 7, pp. 2032–2043, Jul. 2023.

[179] Y. Qiu, C. Huang, J. Wang, Z. Huang, and J. Xiao, "A privacy-preserving subgraph-level federated graph neural network via differential privacy," in *Proc. Int. Conf. Knowl. Sci. Eng. Manag.*, 2022, pp. 165–177.

[180] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nat. Commun.*, vol. 13, no. 1, p. 3091, 2022.

[181] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "FEDGNN: Federated graph neural network for privacy-preserving recommendation," 2021, *arXiv:2102.04925*.

[182] B. Du and C. Wu, "Federated graph learning with periodic neighbor sampling," in *Proc. IEEE/ACM 30th Int. Symp. Qual. Service (IWQoS)*, 2022, pp. 1–10.

[183] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, "FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 36, 2024, p. 12.

[184] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, "Differentially private federated knowledge graphs embedding," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 1416–1425.

[185] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *Proc. Int. Conf. Learn. Rep.*, 2018, p. 9.

[186] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "FEDE: Embedding knowledge graphs in federated setting," in *Proc. 10th Int. Joint Conf. Knowl. Graphs*, 2021, pp. 80–88.

[187] K. Zhang, Y. Wang, H. Wang, L. Huang, C. Yang, and L. Sun, "Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation," in *Proc. ACL Workshop Feder. Learn. Nat. Lang. Process.*, 2022, p. 22.

[188] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "Federated knowledge graph completion via embedding-contrastive learning," *Knowl.-Based Syst.*, vol. 252, Sep. 2022, Art. no. 109459.

[189] G. Mei, Z. Guo, S. Liu, and L. Pan, "SGNN: A graph neural network based federated learning approach by hiding structure," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2019, pp. 2560–2568.

[190] T.-H. Cheung, W. Dai, and S. Li, "FEDSGC: Federated simple graph convolution for node classification," in *Proc. IJCAI Workshops*, 2021, pp. 1–9.

[191] W. Li and S. Wang, "Federated meta-learning for spatial–temporal prediction," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10355–10374, 2022.

[192] J. Chen, G. Huang, H. Zheng, S. Yu, W. Jiang, and C. Cui, "Graph-fraudster: Adversarial attacks on graph neural network-based vertical federated learning," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 2, pp. 492–506, Apr. 2023.

[193] W.-Q. Ren et al., "A survey on collaborative DNN inference for edge intelligence," *Mach. Intell. Res.*, vol. 20, no. 3, pp. 370–395, 2023.

[194] C. Zhou, J. Gao, M. Li, N. Cheng, X. Shen, and W. Zhuang, "Digital twin-based 3-D map management for edge-assisted device pose tracking in mobile AR," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17812–17826, May 2024.

[195] S. Hu, M. Li, J. Gao, C. Zhou, and X. S. Shen, "Adaptive device-edge collaboration on DNN inference in AIoT: A digital twin-assisted approach," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12893–12908, Apr. 2024.

[196] K. Qu, W. Zhuang, W. Wu, M. Li, X. Shen, and X. Li, "Stochastic cumulative DNN inference with RL-aided adaptive IoT device-edge collaboration," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 18000–18015, Oct. 2023.

[197] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the Industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 96–103, Sep./Oct. 2019.

[198] S. Ye et al., "Jupiter: Fast and resource-efficient collaborative inference of generative LLMs on edge devices," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2025, pp. 1–9.

[199] J. Shao, H. Zhang, Y. Mao, and J. Zhang, "Branchy-GNN: A device-edge co-inference framework for efficient point cloud processing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021, pp. 8488–8492.

[200] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic progressive inference of neural networks over device and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–15.

[201] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, "Pyramid: Enabling hierarchical neural networks with edge computing," in *Proc. ACM Web Conf.*, 2022, pp. 1860–1870.

[202] L. Zeng, S. Ye, X. Chen, and Y. Yang, "Implementation of big AI models for wireless networks with collaborative edge computing," *IEEE Wireless Commun.*, vol. 31, no. 3, pp. 50–58, Jun. 2024.

[203] S. Yang, L. Jiao, R. Yahyapour, and J. Cao, "Online orchestration of collaborative caching for multi-bitrate videos in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4207–4220, Dec. 2022.

[204] M. Zhang, J. Cao, L. Yang, L. Zhang, Y. Sahni, and S. Jiang, "ENTs: An edge-native task scheduling system for collaborative edge computing," in *Proc. IEEE/ACM 7th Symp. Edge Comput. (SEC)*, 2022, pp. 149–161.

[205] W. Xu, H. Wang, Z. Lu, C. Hua, N. Cheng, and S. Guo, "Mobile collaborative learning over opportunistic Internet of Vehicles," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 3187–3199, Apr. 2024.

[206] H. Li, X. Li, Q. Fan, Q. He, X. Wang, and V. C. Leung, "Distributed DNN inference with fine-grained model partitioning in mobile edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9060–9074, Oct. 2024.

[207] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.

[208] P. Dong, J. Ge, X. Wang, and S. Guo, "Collaborative edge computing for Social Internet of Things: Applications, solutions, and challenges," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 1, pp. 291–301, Feb. 2022.

[209] L. Zeng, P. Huang, K. Luo, X. Zhang, Z. Zhou, and X. Chen, "FOGRAPH: Enabling real-time deep graph inference with fog computing," in *Proc. ACM Web Conf.*, 2022, pp. 1774–1784.

[210] L. Zeng, X. Chen, P. Huang, K. Luo, X. Zhang, and Z. Zhou, "Serving graph neural networks with distributed fog servers for smart IoT services," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 550–565, Feb. 2024.

[211] L. Zeng, C. Yang, P. Huang, Z. Zhou, S. Yu, and X. Chen, "GNN at the edge: Cost-efficient graph neural network processing over distributed edge servers," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 3, pp. 720–739, Mar. 2023.

[212] A. Zhou et al., "Brief industry paper: Optimizing memory efficiency of graph neural networks on edge computing platforms," in *Proc. IEEE 27th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2021, pp. 445–448.

[213] T. Liu, P. Li, Z. Su, and M. Dong, "Efficient inference of graph neural networks using local sensitive hash," *IEEE Trans. Sustain. Comput.*, vol. 9, no. 3, pp. 548–558, May/Jun. 2024.

[214] H. Zhou, A. Srivastava, H. Zeng, R. Kannan, and V. Prasanna, "Accelerating large scale real-time GNN inference using channel pruning," *Proc. VLDB Endowment*, vol. 14, no. 9, pp. 1597–1605, 2021.

[215] J. Yik, S. R. Kuppannagari, H. Zeng, and V. K. Prasanna, "Input feature pruning for accelerating GNN inference on heterogeneous platforms," in *Proc. IEEE 29th Int. Conf. High Perform. Comput. Data Anal. (HiPC)*, 2022, pp. 282–291.

[216] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[217] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[218] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: Ultra simplification of graph convolutional networks for recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 1253–1262.

[219] W. Zhang et al., "PASCA: A graph neural architecture search system under the scalable paradigm," in *Proc. ACM Web Conf.*, 2022, pp. 1817–1828.

[220] M. Odema, H. Bouzidi, H. Ouarnoughi, S. Niar, and M. A. A. Faruque, "MaGNAS: A mapping-aware graph neural architecture search framework for heterogeneous MPSoC deployment," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 5s, pp. 1–26, 2023.

[221] B. Feng, Y. Wang, X. Li, S. Yang, X. Peng, and Y. Ding, "SGQuant: Squeezing the last bit on graph neural networks with specialized quantization," in *Proc. IEEE 32nd Int. Conf. Tools Artif. Intell. (ICTAI)*, 2020, pp. 1044–1052.

[222] S. A. Tailor, J. Fernández-Marqués, and N. D. Lane, "Degree-quant: Quantization-aware training for graph neural networks," in *Proc. 9th Int. Conf. Learn. Rep. (ICLR)*, 2021, p. 8.

[223] M. Ding et al., "VQ-GNN: A universal framework to scale up graph neural networks using vector quantization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 6733–6746.

[224] J. Wang, Y. Wang, Z. Yang, L. Yang, and Y. Guo, "Bi-GCN: Binary graph convolutional network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1561–1570.

[225] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[226] J. Liu, T. Zheng, G. Zhang, and Q. Hao, "Graph-based knowledge distillation: A survey and experimental evaluation," 2023, *arXiv:2302.14643*.

[227] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4794–4802.

[228] H. Chen, L. Zeng, S. Yu, and X. Chen, "Knowledge distillation for mobile edge computation offloading," *ZTE Commun.*, vol. 18, no. 2, pp. 40–48, 2020.

[229] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021.

[230] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, "Distilling knowledge from graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7074–7083.

[231] C. Yang, J. Liu, and C. Shi, "Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework," in *Proc. Web Conf.*, 2021, pp. 1227–1237.

[232] B. Yan, C. Wang, G. Guo, and Y. Lou, "TinyGNN: Learning efficient graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2020, pp. 1848–1856.

[233] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 101–115, Jan. 2021.

[234] M. Ma, C. Gong, L. Zeng, Y. Yang, and L. Wu, "FlocOff: Data heterogeneity resilient federated learning with communication-efficient edge offloading," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 11, pp. 3262–3277, Nov. 2024.

[235] K. Huang, C. Qiu, C. Hou, X. Li, and X. Wang, "HyperJet: Joint communication and computation scheduling for hypergraph tasks in distributed edge computing," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2025, pp. 1–9.

[236] M. Chiang, P. Hande, T. Lan, and C.-W. Tan, "Power control in wireless cellular networks," *Found. Trends Netw.*, vol. 2, no. 4, pp. 381–533, 2008.

[237] R. L. Cruz and A. V. Santhanam, "Optimal routing, link scheduling and power control in multihop wireless networks," in *Proc. IEEE 32nd Annu. Joint Conf. Comput. Commun. Soc. (IEEE INFOCOM)*, vol. 1, 2003, pp. 702–711.

[238] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "A graph neural network approach for scalable wireless power control," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6.

[239] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 2977–2991, 2020.

[240] M. Eisen and A. Ribeiro, "Transferable policies for large scale wireless networks with graph neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2020, pp. 5040–5044.

[241] L. Salaün, H. Yang, S. Mishra, and C. S. Chen, "A GNN approach for cell-free massive MIMO," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2022, pp. 3053–3058.

[242] I. Nikoloska and O. Simeone, "Fast power control adaptation via meta-learning for random edge graph neural networks," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2021, pp. 146–150.

[243] H. Yang, N. Cheng, R. Sun, W. Quan, R. Chai, and K. Aldubaikhy, "Knowledge-driven resource allocation for wireless networks: A WMMSE unrolled graph neural network approach," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 18902–18916, May 2024.

[244] Y. Yang, S. Zhang, F. Gao, J. Ma, and O. A. Dobre, "Graph neural network-based channel tracking for massive MIMO networks," *IEEE Commun. Lett.*, vol. 24, no. 8, pp. 1747–1751, Aug. 2020.

[245] H. He, X. Yu, J. Zhang, S. Song, and K. B. Letaief, "Message passing meets graph neural networks: A new paradigm for massive MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 4709–4723, May 2024.

[246] V. Ranasinghe, N. Rajatheva, and M. Latva-Aho, "Graph neural network based access point selection for cell-free massive MIMO systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2021, pp. 01–06.

[247] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 388–396.

[248] E. Björnson, M. Bengtsson, and B. Ottersten, "Optimal multiuser transmit beamforming: A difficult problem with a simple solution structure [lecture notes]," *IEEE Signal Process. Mag.*, vol. 31, no. 4, pp. 142–148, Jul. 2014.

[249] T. Chen, M. You, G. Zheng, and S. Lambotharan, "Graph neural network based beamforming in D2D wireless networks," in *Proc. 25th Int. ITG Workshop Smart Antennas (WSA)*, 2021, pp. 1–5.

[250] J. Kim, H. Lee, S.-E. Hong, and S.-H. Park, "A bipartite graph neural network approach for scalable beamforming optimization," *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 333–347, Jan. 2023.

[251] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 1st Quart., 2021.

[252] D. Heo, S. Lange, H.-G. Kim, and H. Choi, "Graph neural network based service function chaining for automatic network control," in *Proc. 21st Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2020, pp. 7–12.

[253] D. Heo, D. Lee, H.-G. Kim, S. Park, and H. Choi, "Reinforcement learning of graph neural networks for service function chaining in computer network management," in *Proc. 23rd Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2022, pp. 1–6.

[254] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1367–1376, Feb. 2022.

[255] F. Naeem, G. Kaddoum, and M. Tariq, "Digital twin-empowered network slicing in B5G networks: Experience-driven approach," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–5.

[256] J. Suarez-Varela, P. Almasan, M. Ferriol-Galmes, K. Rusek, F. Geyer, and X. Cheng, "Graph neural networks for communication networks: Context, use cases and opportunities," *IEEE Netw.*, vol. 37, no. 3, pp. 146–153, May/Jun. 2022.

[257] M. Ma, C. Gong, L. Zeng, and Y. Yang, "MOGR: Multi-task offloading via graph representation in heterogeneous computing network," in *Proc. IEEE Int. Conf. Commun.*, 2024, pp. 1–6.

[258] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep reinforcement learning-based channel allocation for wireless lans with graph convolutional networks," *IEEE Access*, vol. 8, pp. 31823–31834, 2020.

[259] A. Chowdhury, G. Verma, C. Rao, A. Swami, and S. Segarra, "Unfolding WMMSE using graph neural networks for efficient power allocation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6004–6017, Sep. 2021.

[260] S. Szott, K. Kosek-Szott, P. Gawlowicz, J. T. Gómez, B. Bellalta, and A. Zubow, "Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 3, pp. 1843–1893, 3rd Quart., 2022.

[261] H. Kim, B. S. Lee, W.-Y. Shin, and S. Lim, "Graph anomaly detection with graph neural networks: Current status and challenges," *IEEE Access*, vol. 10, pp. 111820–111829, 2022.

[262] L.-H. Chen et al., "AnomMAN: Detect anomalies on multi-view attributed networks," *Inf. Sci.*, vol. 628, pp. 1–21, May 2023.

[263] G. Zhang et al., "eFraudCom: An e-commerce fraud detection system via competitive graph neural networks," *ACM Trans. Inf. Syst.*, vol. 40, no. 3, pp. 1–29, 2022.

[264] R. Ma, G. Pang, L. Chen, and A. van den Hengel, "Deep graph-level anomaly detection by GLOCAL knowledge distillation," in *Proc. 15th ACM Int. Conf. Web Search Data Min.*, 2022, pp. 704–714.

[265] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2196–2203.

[266] L. Ruff et al., "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.

[267] H. Chen and H. Eldardiry, "Graph time-series modeling in deep learning: A survey," *ACM Trans. Knowl. Disc. Data*, vol. 18, no. 5, pp. 1–35, 2024.

[268] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 4027–4035.

[269] W. Zhang, C. Zhang, and F. Tsung, "GRELEN: Multivariate time series anomaly detection from the perspective of graph relational learning," in *Proc. 31st Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 7, 2022, pp. 2390–2397.

[270] S. Han and S. S. Woo, "Learning sparse latent graph representations for anomaly detection in multivariate time series," in *Proc. 28th ACM SIGKDD Conf. Knowl. Disc. Data Min.*, 2022, pp. 2977–2986.

[271] Z. Wu, X. Wu, and Y. Long, "Multi-level federated graph learning and self-attention based personalized Wi-Fi indoor fingerprint localization," *IEEE Commun. Lett.*, vol. 26, no. 8, pp. 1794–1798, Aug. 2022.

[272] P. Xing, S. Lu, L. Wu, and H. Yu, "Big-fed: Bilevel optimization enhanced graph-aided federated learning," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 903–914, Dec. 2024.

[273] F. Chen, G. Long, Z. Wu, T. Zhou, and J. Jiang, "Personalized federated learning with a graph," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, 2022, pp. 2575–2582.

[274] B. Li, A. Swami, and S. Segarra, "Power allocation for wireless federated learning using graph neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2022, pp. 5243–5247.

[275] H. Lee, A. L. Bertozzi, J. Kovačević, and Y. Chi, "Privacy-preserving federated multi-task linear regression: A one-shot linear mixing approach inspired by graph regularization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2022, pp. 5947–5951.

[276] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2749–2758.

[277] S. Lu, Y. Zhang, and Y. Wang, "Decentralized federated learning for electronic health records," in *Proc. IEEE 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, 2020, pp. 1–5.

[278] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," in *Proc. IEEE 56th Annu. Conf. Inf. Sci. Syst. (CISS)*, 2022, pp. 101–106.

[279] Y. Tao, Y. Li, and Z. Wu, "SemiGraphFL: Semi-supervised graph federated learning for graph classification," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2022, pp. 474–487.

[280] X. Yuan, J. Chen, J. Yang, N. Zhang, T. Yang, and T. Han, "FedSTN: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8738–8748, Aug. 2023.

[281] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[282] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, "A new look and convergence rate of federated multitask learning with Laplacian regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 6, pp. 8075–8085, Jun. 2024.

[283] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Serverless multi-task federated learning for graph neural networks," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 6865–6873.

[284] K. Huang, J. Zhai, Z. Zheng, Y. Yi, and X. Shen, "Understanding and bridging the gaps in current GNN performance optimizations," in *Proc. 26th ACM SIGPLAN Symp. Principle Pract. Parallel Program.*, 2021, pp. 119–132.

[285] H. Zhang et al., "Understanding GNN computational graph: A coordinated computation, IO, and memory perspective," *Proc. Mach. Learn. Syst.*, vol. 4, 2022, pp. 467–484.

[286] M. K. Rahman, M. H. Sujon, and A. Azad, "FuseDMM: A unified SDDMM-SPMM kernel for graph embedding and graph neural networks," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2021, pp. 256–266.

[287] Z. Gong, H. Ji, Y. Yao, C. W. Fletcher, C. J. Hughes, and J. Torrellas, "GraphITE: Optimizing graph neural networks on CPUs through cooperative software-hardware techniques," in *Proc. 49th Annu. Int. Symp. Comput. Architect.*, 2022, pp. 916–931.

[288] M. Vasimuddin et al., "DistGNN: Scalable distributed training for large-scale graph neural networks," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–14.

[289] J. Thorpe et al., "DORYLUS: Affordable, scalable, and accurate *GNN* training with distributed *CPU* servers and serverless threads," in *Proc. 15th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2021, pp. 495–514.

[290] L. Wang et al., "FlexGraph: A flexible and efficient distributed framework for GNN training," in *Proc. 16th Eur. Conf. Comput. Syst.*, 2021, pp. 67–82.

[291] M. J. Adiletta et al., "Characterizing the scalability of graph convolutional networks on Intel PIUMA," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2023, pp. 168–177.

[292] Y. Hu et al., "FeatGraph: A flexible and efficient backend for graph neural network systems," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC)*, 2020, pp. 1–13.

[293] Y. Shao et al., "Distributed graph neural network training: A survey," *ACM Comput. Surveys*, vol. 56, no. 8, pp. 1–39, 2024.

[294] H. Lin, M. Yan, X. Ye, D. Fan, S. Pan, and W. Chen, "A comprehensive survey on distributed training of graph neural networks," *Proc. IEEE*, vol. 111, no. 12, pp. 1572–1606, Dec. 2023.

[295] Y.-C. Lin, B. Zhang, and V. Prasanna, "HP-GNN: Generating high throughput GNN training implementation on CPU-FPGA heterogeneous platform," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2022, pp. 123–133.

[296] S. Abi-Karam, Y. He, R. Sarkar, L. Sathidevi, Z. Qiao, and C. Hao, "GENGNN: A generic FPGA framework for graph neural network acceleration," 2022, *arXiv:2201.08475*.

[297] B. Zhang, H. Zeng, and V. Prasanna, "GraphAGILE: An FPGA-based overlay accelerator for low-latency GNN inference," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 9, pp. 2580–2597, Sep. 2023.

[298] R. Chen, H. Zhang, S. Li, E. Tang, J. Yu, and K. Wang, "Graph-OPU: A highly integrated FPGA-based overlay processor for graph neural networks," in *Proc. 33rd Int. Conf. Field Program. Logic Appl. (FPL)*, 2023, pp. 228–234.

[299] A. Heintz and V. Razavimaleki, "Accelerated charged particle tracking with graph neural networks on FPGAs," in *Proc. 3rd Workshop Mach. Learn. Phys. Sci. (NeurIPS)*, 2020, pp. 1–9.

[300] B. Zhang, R. Kannan, V. Prasanna, and C. Busart, "Accelerating GNN-based SAR automatic target recognition on HBM-enabled FPGA," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, 2023, pp. 1–7.

[301] S.-Y. Huang et al., "Low latency edge classification GNN for particle trajectory tracking on FPGAs," in *Proc. IEEE 33rd Int. Conf. Field Program. Logic Appl. (FPL)*, 2023, pp. 294–298.

[302] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong, and C. Young, "Ultra power-efficient CNN domain specific accelerator with 9.3 tops/watt for mobile and embedded applications," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 1677–1685.

[303] Google. "Tensor processing unit." Accessed: Jan. 25, 2024. [Online]. Available: https://cloud.google.com/tpu/

[304] Qualcomm. "Mobile AI solutions." Accessed: Jan. 25, 2024. [Online]. Available: https://www.qualcomm.com/products/mobile/snapdragon/smartphones/mobile-ai

[305] Apple. "Deploying transformers on the apple neural engine." Accessed: Jan. 25, 2024. [Online]. Available: https://machinelearning.apple.com/research/neural-engine-transformers

[306] S. Liang, Y. Wang, C. Liu, L. He, H. Li, and D. Xu, "ENGN: A high-throughput and energy-efficient accelerator for large graph neural networks," *IEEE Trans. Comput.*, vol. 70, no. 9, pp. 1511–1525, Sep. 2021.

[307] A. Auten, M. Tomei, and R. Kumar, "Hardware acceleration of graph neural networks," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–6.

[308] M. Yan, Z. Chen, L. Deng, X. Ye, Z. Zhang, and D. Fan, "Characterizing and understanding GCNs on GPU," *IEEE Comput. Archit. Lett.*, vol. 19, no. 1, pp. 22–25, Jan.–Jun. 2020.

[309] H. Lin, M. Yan, X. Yang, M. Zou, W. Li, and X. Ye, "Characterizing and understanding distributed GNN training on GPUs," *IEEE Comput. Archit. Lett.*, vol. 21, no. 1, pp. 21–24, Jan.–Jun. 2022.

[310] M. Yan et al., "HYGCN: A GCN accelerator with hybrid architecture," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2020, pp. 15–29.

[311] K. Kiningham, P. Levis, and C. Ré, "GRIP: A graph neural network accelerator architecture," *IEEE Trans. Comput.*, vol. 72, no. 4, pp. 914–925, Apr. 2023.

[312] K. Kiningham, P. Levis, and C. Re, "GReTA: Hardware optimized graph processing for GNNs," in *Proc. Workshop Resource Constrained Mach. Learn. (ReCoML)*, 2020, p. 10.

[313] T. Geng et al., "AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitect. (MICRO)*, 2020, pp. 922–936.

[314] S. Mondal, S. D. Manasi, K. Kunal, and S. S. Sapatnekar, "GNNIE: GNN inference engine with load-balancing and graph-specific caching," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 565–570.

[315] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Rep. Learn. Graphs Manifolds*, 2019, p. 12.

[316] M. Wang et al., "Deep graph library: A graph-centric, highly-performance package for graph neural networks," 2019, *arXiv:1909.01315*.

[317] D. Zhang et al., "AGL: A scalable system for industrial-purpose graph machine learning," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3125–3137, 2020.

[318] T. Liu et al., "*BGL* : GPU-efficient *GNN* training by Optimizing graph data *i/O* and preprocessing," in *Proc. 20th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2023, pp. 103–118.

[319] O. Ferludin et al., "TF-GNN: Graph neural networks in tensorflow," 2022, *arXiv:2207.03522*.

[320] Caltrans Performance Measurement System. "PeMS dataset." Accessed: Jul. 31, 2023. [Online]. Available: http://pems.dot.ca.gov/

[321] Metro. "METR-LA dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.kaggle.com/datasets/annnnguyen/metr-la-dataset

[322] NYC Taxi & Limousine Commission. "NYC taxi dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[323] Divvy. "Chicago bike trip dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://divvybikes.com/system-data

[324] Beijing Municipal Environmental Monitoring Center. "Beijing multi-site air quality dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data

[325] Google. "WeatherBench2 dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://github.com/google-research/weatherbench2

[326] A. Varbella, K. Amara, B. Gjorgiev1, and S. Giovanni. "PowerGraph dataset." Accessed: Mar. 1, 2024. [Online]. Available: https://figshare.com/articles/dataset/PowerGraph/22820534

[327] JHU CSSE. "COVID-19 dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://github.com/CSSEGISandData/COVID-19

[328] Federal Highway Administration Research and Technology. "US-101 Dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.fhwa.dot.gov/publications/research/operations/07030/

[329] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. "Stanford drone dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://cvgl.stanford.edu/projects/uav_data/

[330] Kinetics Project Team. "Kinetics dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://github.com/cvdfoundation/kinetics-dataset

[331] nuScenes Team. "nuScenes dataset." Accessed: Aug. 20, 2024. [Online]. Available: https://www.nuscenes.org/

[332] CORSMAL Project Team. "CORSMAL dataset." Accessed: Aug. 20, 2024. [Online]. Available: https://corsmal.eecs.qmul.ac.uk/

[333] RoboNet Project Team. "RoboNet dataset." Accessed: Aug. 20, 2024. [Online]. Available: https://www.robonet.wiki/

[334] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot. "NTU RGB+D dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://rose1.ntu.edu.sg/dataset/actionRecognition/

[335] BMI Lab. "MobiAct dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://bmi.hmu.gr/the-mobifall-and-mobiact-datasets-2/

[336] E. Cippitelli, E. Gambi, S. Gasparrini, and S. Spinsante. "TST fall detection datasets V2 dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://ieee-dataport.org/documents/tst-fall-detection-dataset-v2

[337] CARSM. "Montreal archive of sleep studies dataset." Accessed: Jul. 31, 2023. [Online]. Available: http://ceams-carsm.ca/mass/

[338] A. Mollahosseini, B. Hasani, and M. H. Mahoor. "AffectNet dataset." Accessed: Jul. 31, 2023. [Online]. Available: http://mohammadmahoor.com/affectnet/

[339] E. Cho, S. A. Myers, and J. Leskovec. "Gowalla dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://snap.stanford.edu/data/loc-gowalla.html
[340] Foursquare Dataset Team. "Foursquare dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[341] C. Marche, L. Atzori, V. Pilloni, and M. Nitti. "Social IoT dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.social-iot.org/
[342] Yelp. "Yelp open dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.yelp.com/dataset
[343] Y. LeCun, C. Cortes, and C. Burges. "MNIST dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://www.tensorflow.org/datasets/catalog/mnist
[344] DAVIS Dataset Team. "DAVIS dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://davischallenge.org/
[345] YouTube-VOS Project Team. "YouTube video instance segmentation dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://youtube-vos.org/dataset/vis/
[346] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "The KITTI vision benchmark suite." Accessed: Jul. 31, 2023. [Online]. Available: https://www.cvlibs.net/datasets/kitti/
[347] Z. Wu et al. "ModelNet Dataset." Accessed: Jul. 31, 2023. [Online]. Available: https://modelnet.cs.princeton.edu/
[348] Z. Yao, S. Xia, Y. Li, and G. Wu, "Cooperative task offloading and service caching for digital twin edge networks: A graph attention multi-agent approach," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3401–3413, Nov. 2023.
[349] H. Zhang, X. Ma, X. Liu, L. Li, and K. Sun, "GNN-based power allocation and user association in digital twin network for the terahertz band," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3111–3121, Oct. 2023.
[350] Y. Kawamoto, A. Matsushita, S. Verma, N. Kato, K. Kaneko, and A. Sata, "HAPS-based interference suppression through null broadening with directivity control in space–air–ground integrated networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16098–16107, Dec. 2023.
[351] Y. Kawamoto, M. Takahashi, S. Verma, N. Kato, H. Tsuji, and A. Miura, "Traffic prediction-based dynamic resource control strategy in HAPS-mounted MEC-assisted satellite communication systems," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13824–13836, Apr. 2024.
[352] M. Ariyoshi, J. Funada, E. L. T. D. Gabory, S. Asai, T. K. Rodrigues, and Y. Kawamoto, "Challenges and machine learning solutions for optical communications in space–air–ground integrated networks for 6G," *IEEE Wireless Commun.*, vol. 31, no. 6, pp. 21–28, Dec. 2024.
[353] Y. Yang, J. Wu, T. Chen, C. Peng, J. Wang, and J. Deng, "Task-oriented 6G native-AI network architecture," *IEEE Netw.*, vol. 38, no. 1, pp. 219–227, Jan. 2024.
[354] Y. Yang, M. Ma, H. Wu, Q. Yu, X. You, and J. Wu, "6G network AI architecture for everyone-centric customized services," *IEEE Netw.*, vol. 37, no. 5, pp. 71–80, Sep. 2022.
[355] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.
[356] J. Wang, J. Liang, J. Liang, and K. Yao, "GUIDE: Training deep graph neural networks via guided dropout over edges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4465–4477, Apr. 2024.
[357] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 199–213.
[358] T. K. Rodrigues, S. Verma, Y. Kawamoto, N. Kato, M. M. Fouda, and M. Ismail, "Smart handover with predicted user behavior using convolutional neural networks for WiGig systems," *IEEE Netw.*, vol. 38, no. 4, pp. 190–196, Jul. 2024.
[359] M. Ma, C. Gong, L. Zeng, and Y. Yang, "Multi-tier multi-node scheduling of LLM for collaborative AI computing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2025, pp. 1–9.
[360] Z. Yao, S. Xia, Y. Li, G. Wu, and L. Zuo, "Transfer learning with spatial–temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8592–8605, Aug. 2023.
[361] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5370–5390, May 2023.
[362] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Rep.*, 2018, pp. 1–9.
[363] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, "Provably powerful graph networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 2153–2164.
[364] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.

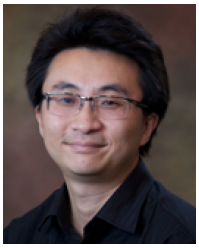
Liekang Zeng (Member, IEEE) received the B.E. and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China. His current research interests include edge intelligence, mobile computing, and machine learning systems.

Shengyuan Ye (Graduate Student Member, IEEE) received the B.E. degree from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, in 2021, where he is currently pursuing the Ph.D. degree. His current research interests include mobile edge computing, resource-efficient mobile AI systems, and distributed machine learning systems.

Xu Chen (Senior Member, IEEE) received the Ph.D. degree in information engineering from the Chinese University of Hong Kong in 2012. He is a Full Professor with Sun Yat-sen University, Guangzhou, China, and the Vice Director of the National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He was a Postdoctoral Research Associate with Arizona State University, Tempe, USA, from 2012 to 2014, and a Humboldt Scholar Fellow with the Institute of Computer Science, University of Goettingen, Germany, from 2014 to 2016. He was a recipient of the Prestigious Humboldt Research Fellowship awarded by the Alexander von Humboldt Foundation of Germany, the 2014 Hong Kong Young Scientist Runner-Up Award, the 2017 IEEE Communication Society Asia–Pacific Outstanding Young Researcher Award, the 2017 IEEE ComSoc Young Professional Best Paper Award, the Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics, the Best Paper Runner-Up Award of 2014 IEEE International Conference on Computer Communications, and the Best Paper Award of 2017 IEEE International Conference on Communications. He is currently an Area Editor of IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, and an Associate Editor of the IEEE TRANSACTIONS WIRELESS COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, and IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Series on Network Softwarization and Enablers.

Xiaoxi Zhang (Member, IEEE) received the B.E. degree in electronics and information engineering from the Huazhong University of Science and Technology in 2013, and the Ph.D. degree in computer science from The University of Hong Kong in 2017. She was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University from 2017 to 2020. She is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University. She is broadly interested in optimization, algorithm design, and system implementation for networked systems, including cloud and edge computing networks, distributed machine learning systems, and vehicular networks. She was a recipient of the Young Outstanding Award of the Guangdong Province, the Best Student Paper Award of IEEE MSN 2024, the Best Paper Award of IEEE BigCom 2024, and the Best Paper Nominee of IEEE/ACM IWQoS 2023. She is currently an Area Editor of *Computer Networks* (Elsevier), an Associate Editor of IEEE NETWORKING LETTERS, and a Guest Editor of Symmetry (MDPI) in Optimization Theory and Its Applications.

Ju Ren (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Central South University, Changsha, China, in 2009, 2012, and 2016, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include Internet of Things, edge computing, edge intelligence, and security and privacy. He received many Best Paper Awards from IEEE Flagship Conferences, including the IEEE ICC'19 and the IEEE HPCC'19, the IEEE TCSC Early Career Researcher Award in 2019, and the IEEE ComSoc Asia–Pacific Best Young Researcher Award in 2021. He was recognized as a Highly Cited Researcher by Clarivate from 2020 to 2022. He currently serves as an Associate Editor for many journals, including IEEE TRANSACTIONS ON CLOUD COMPUTING and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He also served as the General Co-Chair for IEEE BigDataSE'20, the TPC Co-Chair for IEEE BigDataSE'19, the Publicity Co-Chair for IEEE ICDCS'22, the Poster Co-Chair for IEEE MASS'18, and the Symposium Co-Chair for IEEE/CIC ICCC'23 and 19, I-SPAN'18, and IEEE VTC'17 Fall.



Jian Tang (Fellow, IEEE) received the Ph.D. degree from Arizona State University. He is currently with Midea Group, China. His research interests include big data and machine learning. He received the NSF CAREER Award, the 2016 Best Vehicular Electronics Paper Award from the IEEE Vehicular Technology Society, and the Best Paper Awards from IEEE ICC'14 and IEEE Globecom'15. He is an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON MOBILE COMPUTING.

Yang Yang (Fellow, IEEE) is a Professor with the IoT Thrust, the Director of the Research Center for the Digital World with Intelligent Things, and the Associate Vice President for Teaching and Learning with the Hong Kong University of Science and Technology (Guangzhou), China. He is also an Adjunct Professor with the Department of Broadband Communication, Peng Cheng Laboratory, and a Chief Scientist of IoT with Terminus Group, China. Before joining HKUST (GZ), he has held faculty positions with the Chinese University of Hong Kong, Brunel University, U.K., University College London, U.K., CAS-SIMIT, and ShanghaiTech University, China. His research interests include multitier computing networks, 5G/6G systems, AIoT technologies, intelligent services and applications, and advanced wireless testbeds. He has published more than 300 papers and filed more than 120 technical patents in these research areas.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, AI for networks, and vehicular networks. He received the "West Lake Friendship Award" from Zhejiang Province in 2023, the President's Excellence in Research from the University of Waterloo in 2022, the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award from the IEEE Vehicular Technology Society in 2018, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society (ComSoc), and the Technical Recognition Award from Wireless Communications Technical Committee in 2019 and AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award from the University of Waterloo in 2006 and the Premier's Research Excellence Award from the Province of Ontario, Canada, in 2003. He served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, an International Fellow of the Engineering Academy of Japan, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society. He is the Past President of the IEEE Communications Society. He was the Vice President for Technical and Educational Activities, the Vice President for Publications, a Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, and a member of IEEE Fellow Selection Committee of the ComSoc.