nature reviews electrical engineering

https://doi.org/10.1038/s44287-024-00076-z

Review article

Check for updates

Opportunities and challenges of graph neural networks in electrical engineering

Eli Chien[1], Mufei Li[1], Anthony Aportela[2], Kerr Ding[3], Shuyi Jia[3], Supriyo Maji[4], Zhongyuan Zhao[5], Javier Duarte[2], Victor Fung[3], Cong Hao[1], Yunan Luo[3], Olgica Milenkovic[6], David Pan[4], Santiago Segarra[5] & Pan Li[1]

Abstract

Graph neural networks (GNNs) are a class of deep learning algorithms that learn from graphs, networks and relational data. They have found applications throughout the sciences and made significant strides in electrical engineering. GNNs can learn from various electrical and electronic systems, such as electronic circuits, wireless networks and power systems, and assist in solving optimization or inference tasks where traditional approaches may be slow or inaccurate. Robust learning algorithms and efficient computational hardware developed and tailored for GNNs have amplified their relevance to electrical engineering. We have entered an era in which the studies of GNNs and electrical engineering are intertwined, opening to opportunities and challenges to researchers in both fields. This Review explores applications of GNNs that might generate notable impacts on electrical engineering. We discuss how GNNs are used to address electrical automatic design, as well as the modelling and management of wireless communication networks. Additionally, we delve into GNNs for high-energy physics, materials science and biology. Presenting the applications, data and computational challenges, the need for innovative algorithms and hardware solutions becomes clear.

Sections

[1]Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. [2]Physics, University of California San Diego, San Diego, CA, USA. [3]Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA. [4]Electrical and Computer Engineering, University of Texas Austin, Austin, TX, USA. [5]Electrical and Computer Engineering, Rice University, Houston, TX, USA. [6]Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Champaign, IL, USA. e-mail: panli@gatech.edu

Key points

- Graph neural networks (GNNs) hold immense potential for harnessing data power to effectively tackle a range of application challenges in electrical engineering.
- Research in information science and cutting-edge hardware within electrical engineering offer valuable insights into the practical implementation of GNNs, overcoming their limitations related to model reliability and computational efficiency.
- GNNs find extensive applications in various scientific domains, including physics, materials science and biology. Electrical engineering methodologies can enhance GNN performance in these fields and, potentially, lead to significant impacts in science.

Introduction

Since the beginning of the twenty-first century the field of electrical engineering has made great strides. For example, the development of robust and efficient telecommunication networks has revolutionized global communication, underpinned the Internet and enabled instantaneous, worldwide connectivity[1]. The field of information and signal processing has redefined our interaction with digital media and enhanced our capabilities in data analysis and interpretation[2]. Modern advances in circuit design have also led to miniaturization and paved the way for portable computing and the Internet of Things[3]. Electrical engineering has embarked on a new era — one of advanced communication systems, machine learning and neural networks — in which the need for ultra-efficient hardware and system design, coupled with ultra-fast, highly accurate information processing algorithms, has become paramount.

Meanwhile, artificial intelligence (AI) has undergone a transformation, demonstrating its extraordinary ability to distil knowledge from vast amounts of data and enhance human decision-making and predictive abilities across a wide range of applications. Representative examples include AlphaGo beating the best human players at the game of Go[4], and large language models performing language understanding and reasoning[5,6].

Among the various AI tools, graph neural networks (GNNs) have emerged as powerful techniques for processing irregular or non-Euclidean data, such as graphs or point clouds[7,8]. GNNs have also shown tremendous potential in numerous electrical engineering tasks such as forecasting and controlling the dynamics of electrical and electronic systems. Conventional methods based on assumptions, simulations or heuristics often fall short in terms of precision, efficiency and optimality when applied to real-world systems[9]. Luckily, many electrical engineering systems can be naturally represented as graphs or generate graph-structured data, opening the door for GNNs to leverage the power of data and provide improved forecasting and control strategies. For instance, in the domain of wireless communication, GNNs can encode the channel states between multiple user devices and access points, enabling improved understanding and optimization of communication processes[10–15]. Similarly, in power systems, GNNs can encode the structure and dynamics of power grids, facilitating tasks such as optimal power flow and prediction of power outages[16–18]. GNNs can also be used to predict latency and sizes of circuits without relying on time-consuming simulations[19–22] and provide the representations of circuit data to accelerate the iteration process in circuit design loops[20,23]. Besides the use of GNNs in strictly related electrical engineering applications, GNNs can also be used in other domains such as physics[24–26], materials science[27,28] and biology[29–31].

Despite the opportunities, GNNs face a generalization issue across the diverse scientific domains due to the gap between the less explored, yet promising, regimes and the well-explored ones in which most labelled data are found. Principled theories grounded in the studies of electrical engineering, including information theory and signal processing, offer promising directions to refine GNN architectures and their training algorithms to tackle this challenge. Moreover, the effective success of AI, and particularly that of GNNs, hinges on substantial computational power. Specifically, the processing of irregular data involves random memory access, often leading to a large memory footprint. This aspect highlights the need for dedicated hardware support, a requirement that electrical engineering is well positioned to address.

Here we present a selection of application examples to discuss the ability of GNNs in addressing challenges in electrical engineering and the potential of electrical engineering in overcoming GNN limitations across diverse domains. We start with a concise overview of basic GNN architectures. We then delve into representative GNN applications in electrical engineering, physics and biology to illustrate the opportunities and issues that arise at the intersection of GNNs with different disciplines. We conclude by offering an outlook on limitations, emerging possibilities and the potential social impact of GNN developments.

Graph neural networks

GNNs process data that can be represented as graphs. A graph G consists of a node set V and an edge set E. The structure of G can be denoted as an adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, where $A_{uv} = 1$ if there is an edge $(u, v) \in E$ and 0 otherwise. Here, we use a simple unweighted graph as an example. The processed graphs may also be associated with features, where each node $v \in V$ has a feature vector $X_v \in \mathbb{R}^F$, where F denotes a dimension. All feature vectors are collected in $X = [X_1^T, X_2^T, \ldots, X_{|V|}^T]^T$, where T is the transpose operation.

A GNN works as a function f that combines node features and graph structures in learning node representations: $H = f(A, X) = [H_1^T, H_2^T, \ldots, H_{|V|}^T]^T \in \mathbb{R}^{|V|\times F_{out}}$, where F_{out} denotes a dimension. Learned node representations can then be used for various tasks to be discussed later. A fundamental inductive bias for modelling graph data is that the order of nodes in the graph should not affect the representations associated with them, which is named permutation equivariance. Specifically, take a permutation matrix $P \in \mathbb{R}^{|V|\times|V|}$. A proper architecture of GNN f should satisfy $PH = Pf(A, X) = f(PAP^T, PX)$. The architectures of GNNs discussed here satisfy this property.

Depending on the design motivations and derivations, GNNs can be distinguished into two categories: message passing neural networks (MPNNs; sometimes called spatial GNNs)[32–35] and spectral GNNs[36–39] (see Box 1 and Supplementary information). When using GNNs in real-world scenarios, several limitations are noteworthy and deserve careful consideration by practitioners.

Limited expressive power

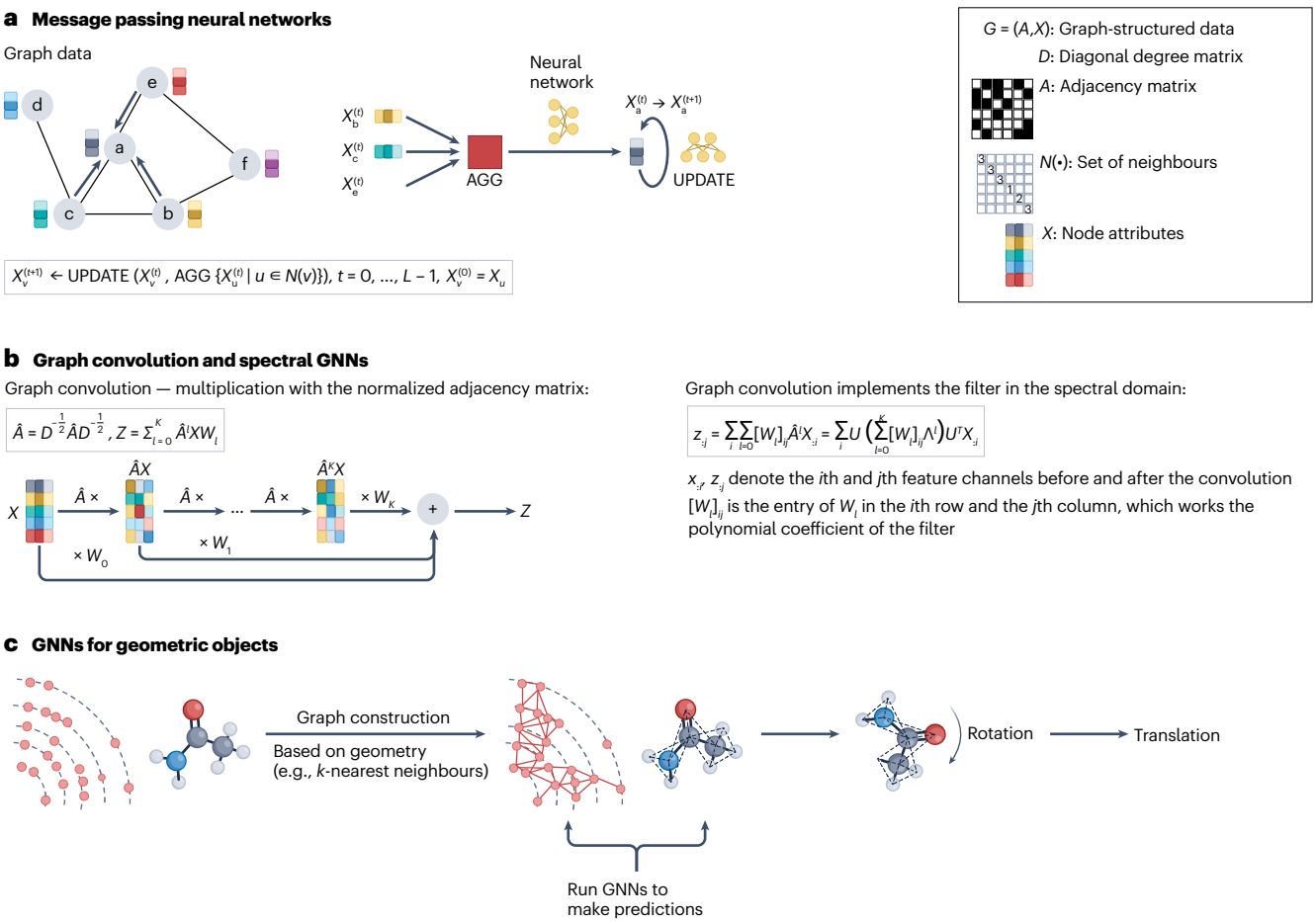
The expressive power of a model depicts its capability to approximate complex functions. Traditional feed-forward neural networks, for instance, are known for their capacity to approximate any continuous function within a compact space[40]. However, GNNs — including MPNNs

Box 1 | Overview of graph neural network architectures

As the name implies, message passing neural networks (MPNNs) follow a message passing procedure along the edges of the input graph to learn node representations (see the figure, panel **a**). In learning the representation of a node v, a single MPNN layer first aggregates messages, that is node representations $X_u^{(t)}$, from its neighbours $u \in N(v)$ and then combines the aggregated messages with v's own representation $X_v^{(t)}$ to obtain an update $X_v^{(t+1)}$. Here, t denotes the layer index. By stacking multiple MPNN layers sequentially (t=0,1,...,L−1) and treating the output representations of the previous graph neural network (GNN) layer as the input to the current GNN layer, the final node representations can be used as the ultimately learned representations $H = X^{(L)}$, which can be further used to make downstream predictions. An MPNN consists of two major operations: aggregating (AGG) neighbours' features; and combining aggregated features to update (UPDATE) one's own features.

A spectral GNN typically uses multiple-hop graph convolution (see the figure, panel **b**). The obtained Z will pass through a neural network to either make predictions or go through another round of graph convolution. The key idea of spectral GNNs lies in signal filtering. In classical signal processing, filtering refers to the amplification or attenuation of a signal at different frequencies. Spectral GNNs apply polynomial filters to the graph signals in the spectral domain. Suppose the graph signals have multiple channels. The filtering process to generate the output signal Z in the ith channel follows $Z_{:j} = \sum_{i=1}^{F} U(\sum_{l=0}^{K} [W_l]_{ij} \Lambda^l) U^T X_{:i}$, where $X_{:i}$ denotes the input signal X in the jth channel, and $[W_l]_{ij}$ is the ith-row, jth-column entry of a parameter matrix W_l, $0 \le l \le K$, which denotes a polynomial coefficient of the filter.

GNN architectures should keep the predictions unchanged when the input system is shifted according to some physical principles (see the figure, panel **c**). GNNs found extensive use in analysing irregular geometric objects, such as particle hits in Large Hadron Collider (LHC) detectors[24,25] or molecules with 3D structures[60]. These objects can often be conceptualized as point clouds, where each point v is associated with a coordinate vector in addition to other features. Graphs can be constructed to capture relationships among these points, such as k-nearest neighbour relations. GNNs can then be applied to the constructed graphs.



and spectral GNNs — lack this capacity when it comes to approximating continuous functions defined on graphs[41,42]. Practical methods to deal with this issue often rely on using higher-order tensors[42,43], or performing augmentation of node features or graph structure[44,45].

Over-smoothing

Many GNN architectures such as graph convolutional networks (GCNs)[46] adopt mean pooling to aggregate the features both of the neighbours and of the centre node. These GNNs suffer from an over-smoothing problem[47]. As the depth of the GNN increases, the acquired representations of nodes that are closely positioned on the graph become increasingly alike. This phenomenon hinders the predictive performance of GNNs in tasks where closely adjacent nodes on the graph are required to produce notably distinct predictions.

Over-squashing

The dimensions of node representations usually do not scale along with the model depth. This characteristic, when combined with the aggregation of neighbours' representations in MPNNs, can potentially lead to an over-squashing issue[48,49]. More specifically, node representations constrained by limited dimensions often struggle to maintain high-quality feature information from distant nodes within the graph. However, this over-squashing problem can be mitigated by rewiring edges to enhance connectivity across various graph sections[48].

GNN-enabled learning tasks

The application of GNNs across diverse domains can be broadly classified into three types of learning tasks: predictive, optimization and data generation tasks (Box 2).

Predictive tasks

Predictive tasks infer missing attributes or properties of entities within a graph. Depending on the number of nodes involved, predictive tasks can be classified into node-level, edge-level and graph-level predictions.

GNNs are extensively used to solve the predictive tasks across various applications. For instance, node-level predictions find use in detecting fault locations in power grids[50,51] and identifying noisy particle hits from collision events in Large Hadron Collider (LHC) experiments[52,53]. Edge-level predictions are utilized for predicting potential drug–disease associations[54,55], and for reconstructing particle tracks in the LHC[56–58]. In node or edge-level prediction tasks, the predictive labels can be determined by the characteristics and graph structure surrounding the pertinent nodes or edges. For instance, the likelihood of a power outage could be inferred from the weather signals captured by nearby sensors[17]. GNNs can effectively learn such relationships based on observed labels from different graph sections and extrapolate this understanding across the entire graph. Note that the permutation equivariant property of GNNs affords a valuable inductive bias for this generalization, thereby enhancing their predictive capability. Graph-level prediction tasks include predicting circuit performance[19–22,59], determining molecular properties such as electronic properties and solubility[27,60–62], and identifying the type of particle decaying process after collisions[63,64]. Traditional techniques to measure graph properties in these applications often necessitate the execution of time-consuming and expensive simulations, analyses or experiments[59]. GNNs may approximate the complex relationship between input and output, and bypass such a costly process via a direct estimation. Consequently, GNNs offer a notable improvement in efficiency in addressing prediction tasks.

Optimization tasks

Numerous electrical engineering applications rely on solving optimization problems. Many of these problems have intrinsic graph structures that may highly impact the optimization outcomes. Notable examples include resource allocation problems in wireless networks[10,12,13,65] and optimal power flow in power grids[16,18,66], as well as the low-density parity-check decoding problem where the relationship between variable nodes and check nodes form a graph[67,68].

GNNs can be used to accelerate the procedure of solving the optimization problems in these applications. A straightforward approach is to treat the problem as a predictive task, which could involve collecting a set of problem configurations $G_1, G_2, \ldots$ and the corresponding optimal solutions $\omega^\star(G_1), \omega^\star(G_2), \ldots$. A GNN can be trained to map directly from the problem configuration G_i to the corresponding optimal solution $\omega^\star(G_i)$. However, in practice, it has been observed that a neural network model trained in this manner tends to struggle with generalization or delivering high-accuracy solutions[69]. The conjectured reason behind this problem is that the optimal solution to an optimization problem is typically more numerically sensitive to variations in the problem configuration, compared with a standard predictive task. Consequently, a higher demand on the model's ability to generalize is placed on optimization tasks.

A more compelling approach is to harness the algorithmic structure inherent in some traditional methodologies, allowing neural networks to supplant the most time-consuming or uncertain aspects. An important approach that follows this idea is known as algorithm unrolling or unfolding[70]. This method prompts the unrolling of the iterative processes often found in traditional algorithms and lets GNNs replace some parts of the iterative procedure with a better data-driven operator. Suppose each iterative step can follow $\omega^{(t+1)} = g(\omega^{(t)}; G)$, where the selection of function g often depends on the problem structure and the system uncertainty ξ. Traditional approaches may yield a practically suboptimal g by assuming certain distributions of ξ, which results in slow convergence rates or imprecise solutions. In this context, GNNs can be used to learn from the data and replace the most uncertain and potentially suboptimal part of g. For instance, GNNs have been utilized to improve the unrolled belief propagation algorithm by learning from the data, which yields faster and more accurate low-density parity-check decoding[67,68]. GNNs for optimization can also be applied to link scheduling in wireless systems[71–73] or to optimize power flow in power systems[16,18,66].

Data generation tasks

Many applications in electrical engineering and scientific domains require practitioners to create objects with graph representations, such as circuit layouts designed to fulfil certain size and area constraints[74], or material designs with various mechanical and thermal properties[75]. GNNs for graph data generation have emerged as a promising method to solve these tasks.

Classical random graph models such as the Erdős–Rényi (ER) model[76] are not flexible enough to model the intricate distribution of real graphs. These models usually use simplistic parameterization of the graph distribution, which is often imprecise to describe the actual graph behaviour. By contrast, GNNs offer far more expressiveness and, thus, may consequently facilitate more accurate distribution modelling in practice. GNNs can be integrated with many generative AI frameworks for graph data generation such as variational autoencoder[9], generative adversarial network[77] and diffusion models[78,79]. In these frameworks, GNNs are generally trained to map simple distributions;

for example, from the standard Gaussian distribution to the complex distribution of graphs. Samples from the simple distribution can then be transformed to generated graph data via the trained GNNs. Recent advancements demonstrate the superior performance of GNN-based generative AI models in tasks such as conditional molecule data generation[80–82] and circuit design generation tasks[83–85].

Applications

The versatility of GNNs enable their application in different fields, from electrical engineering, in electronic design and wireless communication, to life science. These applications benefit from electrical engineering advancements, ensuring GNNs are both robust and computationally efficient.

Electronic design

The functionalities in electronic systems and the complexity of their design have experienced consistent growth in accordance with Moore's Law, driven by advancements across devices, circuits, systems and software. The design of electronic systems can be divided into two main groups: those encompassing analogue/radio-frequency/

Box 2 | Three categories of graph neural network-enabled learning tasks

Predictive tasks

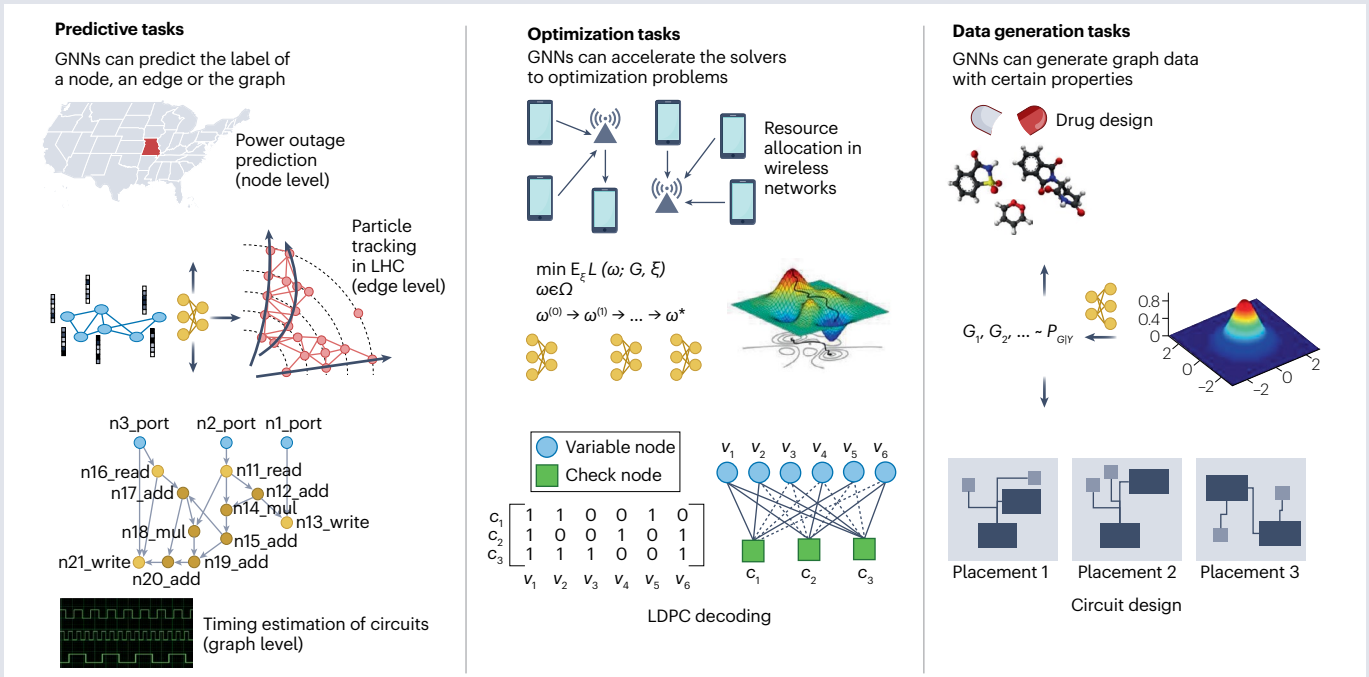
Node-level prediction is concerned with predicting a specific property, denoted as y_v, of a node $v \in V$ within the graph G (see the figure). This task is accomplished by training a model to map the node representation H_v output by the graph neural network (GNN) to the property y_v. Edge-level prediction focuses on predicting a specific property, denoted as y_{vu}, of a pair of nodes v, $u \in V$ within the graph G. This task is achieved by training a model to map a combination of the representations of the two relevant nodes, H_v and H_u, to the property y_{vu}. Commonly used combinations include the concatenation of H_v and H_u into a single vector, and the computation of the inner product or the cosine similarity between H_v and H_u. Graph-level prediction aims to predict a property, denoted as y_G, of the entire graph G. This task is often performed on datasets comprising multiple graphs. The prediction can be made by pooling all node representations into a graph-level representation, followed by learning a mapping from this representation to the label y_G. Pooling operations range from simple summation or averaging to more complex hierarchical methods.

Optimization tasks

The optimization problems can be generally denoted as $\min_{\omega} \mathbb{E}_{\xi} L(\omega; G, \xi)$, where ω denotes the optimization variables, G denotes the problem configuration represented as a graph, ξ denotes some randomness (such as signal noises within the system) and L represents the objective function (see the figure). The goal is to efficiently approximate the optimal solution $\omega^*(G)$ with a specified degree of accuracy.

Data generation tasks

Typically, the graph generation process begins with a collection of graph-structured data associated with specific properties, denoted as (G_i, y_i), where y_i represents the property of G_i (see the figure). The objective is to generate graphs G that possess properties complying with specific constraints $y \in \Omega$. Typically, the generation is also expected to be novel and diverse. The task essentially asks to model a distribution of the graphs conditioning on the properties of interest based on the graph dataset. LDPC, low-density parity check; LHC, Large Hadron Collider.



mixed-signal functions (with similar requirements to those used for analogue design); and those focused on digital functionalities. Over the course of evolution, chip design processes have integrated arrays of software tools operating across multiple levels of hierarchy. Referred to as electronic design automation (EDA) tools, these resources effectively serve to automate chip design, reducing the time to market while enhancing design quality[86].

Digital design. In the context of digital systems, designers primarily focus on the optimization of three performance metrics: power, speed and area. However, the number of components in a digital chip can be much higher, often leading to extensive runtime for full chip development. Notable progress has been made in the evolution of digital full-flow tools, encompassing logic synthesis and physical design. Consisting of four main tasks, floor planning, placement, routing and timing optimization, physical design takes most of the runtime. With the increasing demand for enhanced chip functionality, the scale of digital designs has grown to the point where running even non-optimal or heuristics algorithms for the original nondeterministic polynomial-time (NP)-complete problems poses notable challenges[87]. Machine learning algorithms are being increasingly used to address various problems in physical design. These algorithms effectively learn from existing layouts and design knowledge and improve the turnaround time and quality of solutions for unseen designs by providing early estimates of the critical parameters. Graph structures are natural representations of Boolean functions, circuits and layouts[88]. It is no surprise that, compared with other machine learning techniques, GNNs are a better fit for the problem.

As of June 2024, commercial digital EDA tools, placement, routing and timing optimization steps are executed one after the other. However, an optimal flow necessitates placement and a timing engine to work together. A fast and accurate pre-route timing estimation is therefore necessary for timing-driven placement as running routing and static timing analysis in iteration is expensive. A possible solution is the introduction of a timing engine-inspired GNN model that can predict arrival time and slack at timing end-points without static timing analysis (Fig. 1a). Experimental results on real-world open-source designs demonstrate that the proposed GNN framework has a strong correlation with labelled data while being more than three orders of magnitude faster than routing and static timing analysis-based approaches[22].

GNNs have also been used in other areas within the physical design. For example, if floor planning is posed as a reinforcement learning (RL) problem, an edge-based GCN architecture, capable of learning rich and transferable representations of the chip, can reduce the runtime of floor planning from months to hours while giving comparable results[83]. A GNN-assisted deep RL framework can also be trained to optimally tune placement parameters of a commercial tool in a completely unsupervised manner and without domain knowledge[89]. The trained RL agent achieves up to 11% and 2.5% wire-length improvements in just one iteration, that is 20× and 50× fewer iterations compared with a human engineer and a state-of-the-art tool, respectively. *k*-Means clustering can be used within the GNN-based framework to guide the placement tool to optimally cluster cells leading to 3.9% wire length, 2.8% power and 85.7% performance improvement compared with a commercial placer[90]. A framework that utilizes a long short-term memory network and a GNN is proposed to predict post-route total negative slack. Experiments show the prediction quality is within 5.2% of normalized root mean squared error in early design stages on two validation designs not included in the training[91]. A graph attention network (a variant of GNN)-based framework can be used to predict routing congestion. Experiments show that the framework improves prediction quality by 29%, and takes 19 s, compared with 10–60 min required by other methods on a circuit with 1.3 million cells[92].

Analogue design. In contrast to digital circuits, analogue circuits typically comprise fewer components. However, the number of performance metrics in analogue circuits is higher and navigating the trade-offs among the different parameters is complex. The typical analogue circuit design process starts from a specification and goes through topology selection, sizing and physical design steps. Although analogue physical design steps are similar to those of digital design, there are some unique requirements to this domain (such as matching, symmetry and variation tolerance) which make the process error-prone, non-optimal and often in need of expensive design iterations. Although considerable efforts have been invested towards automating analogue design, progress has been modest due to the knowledge-intensive nature and widely varying requirements typical of this domain[93]. Many tools have failed to encapsulate the intentions of expert designers and lacked the capability to achieve generalization as they have only been useful in handling specific classes of circuits[93]. The advent of AI has invigorated design automation efforts by enabling tools to emulate the methods used by experienced designers, effectively learning from them[94,95].

The problem of analogue circuit sizing across technology can be addressed, for example, by training an RL agent on one technology node and then directly applying the trained agent to search the same circuit under different technology nodes by virtue of similar design principles among different technologies[96] (Fig. 1b). By mapping the circuit to a graph, where transistors are represented as nodes and wires as edges, a GCN[46] is used to learn the circuit's topology representation. The optimization problem is then solved iteratively through the steps of action vector generation, denormalization, refinement, simulation and policy update. Experiments on transfer learning between five technology nodes and two circuit topologies demonstrated that RL with transfer learning can achieve a much higher figure of merit than methods without knowledge transfer. GNNs have also been explored in other areas in analogue circuit design, including topology generation[97], circuit sizing[21,97,98], parasitic prediction[21,99], performance-driven layout[100], automatic constraint generation[101] and analogue/radio-frequency design[98,102].

The use of GNNs in electronic system design is still in its early stages. The need for handling more complex circuit characteristics will necessitate the adoption of more complex graph structures, such as dynamic graphs or bipartite graphs[86]. The practicality of executing optimal algorithms might be limited by extensive runtime demands, but this problem is common to both analogue and digital chip design. The integration of GNN algorithms is essential to enhance efficiency by identifying promising starting points and informed guesses, thereby streamlining the optimization process in full chip development.

Process variation in analogue design is a crucial area that requires more attention from the machine learning community[103]. In digital design, marked performance improvement is possible if several physical design steps are combined into a single framework where timing, routing and placement all happen simultaneously.

Resource allocation in wireless systems

In a wireless network, devices talk with each other through a shared wireless medium. Similar to people at a cocktail party who need to

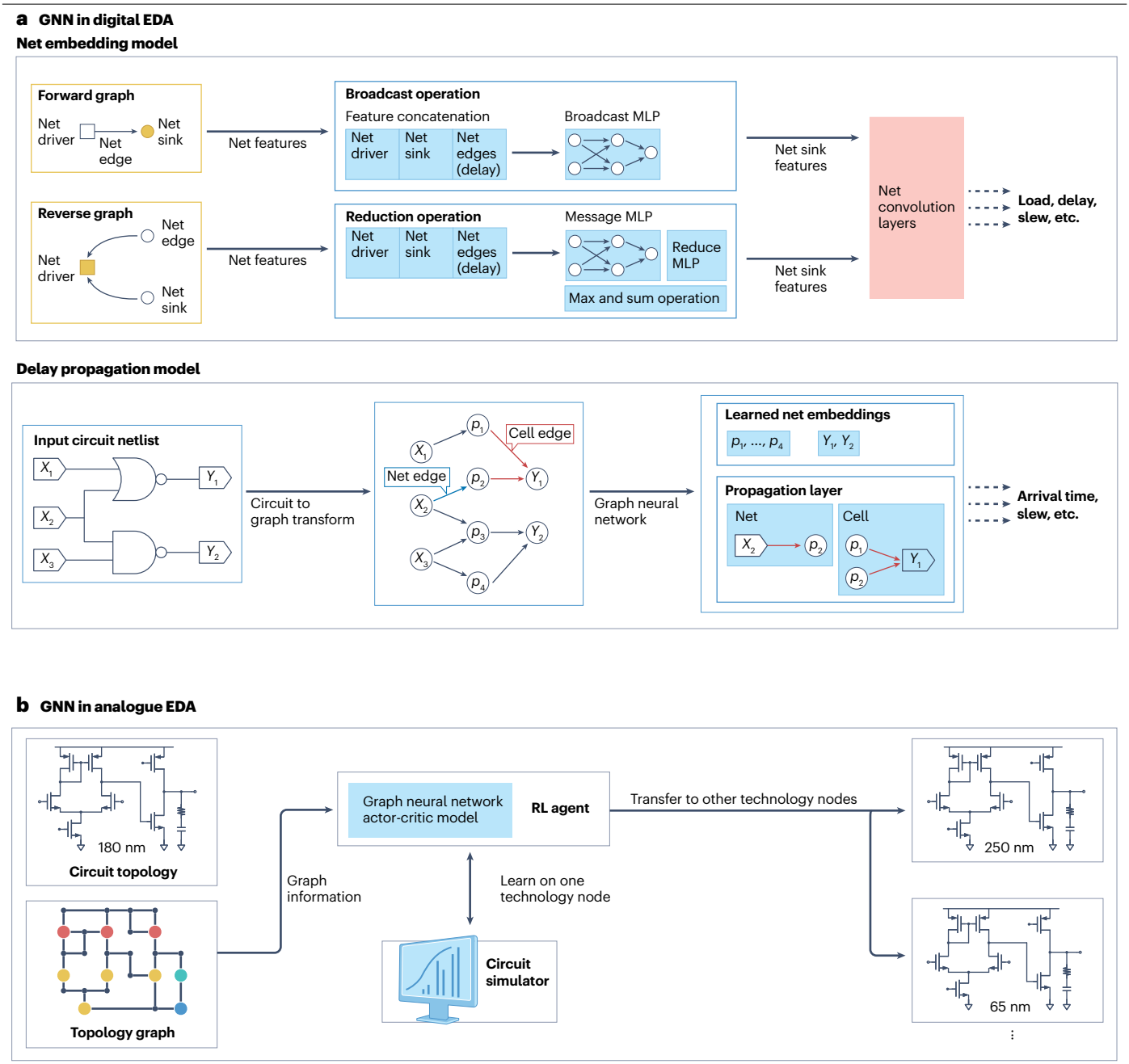


Fig. 1 | Application of graph neural networks in electronic design automation. a, Use of graph neural network (GNN) models in digital electronic design automation (EDA) for timing optimization. The logic circuit is mapped to graphs. GNN models are trained using these mapped graphs to predict net delays and cell delays. Delays are propagated through net propagation layers and cell propagation layers to estimate arrival time and slews at the timing end-points. To augment model training and enhance performance, net and cell delay predictions are used as auxiliary tasks. The GNN model, built on the concept of path delay calculations, reduces the required GNN receptive field and enhances model explainability and interpretability. The model consists of two stages. In net embedding, the first stage, the model captures post-route Elmore delay of nets from designs after placement. This stage utilizes a bi-directional graph consisting of net edges and reversed net edges. In the next stage, delay propagation, cell delays and computation of arrival time and slews are modelled. **b**, Use of a GNN for transistor sizing in analogue EDA. Here, the circuit is represented by a topology graph in which nodes are transistors and edges are wires. In a GNN-equipped reinforcement learning (RL) framework, the RL agent is trained on one circuit, and the trained model is used to optimize new circuits or the same circuit in different technology nodes. MLP, multilayer perceptron. Panel **b** is reprinted with permission from ref. 96, IEEE.

manage their volumes and orders of speaking to have meaningful conversations, resource allocation in wireless networks, such as transmit power control and link scheduling, is critical to the function and performance of wireless systems. Generally, these tasks are formulated as optimization problems on graphs, where the nodes of the graph capture devices in a wireless network, and the edges of the graph model the wireless channels between them.

In infrastructure-based wireless networks, such as cellular and Wi-Fi networks, resource allocation for wireless devices can be done by dedicated base stations or access points within a single hop. For example, mobile phones connect directly to their nearest base station without any intermediary relays. For multi-hop wireless networks, such as wireless ad hoc networks, wireless backhaul networks, device to device networks and certain vehicle to everything networks, distributed resource allocation is preferred as it is often infeasible to dedicate servers to resource allocation or collect the full network state at a server in a timely manner. GNNs are particularly well suited to solve these distributed resource allocation problems.

Power control for communication rate maximization. In wireless networks, raising the transmit power of one device can simultaneously increase the strength of the received signal at the intended receiver and the unwanted signal (interference) at other unintended receivers. Based on the wireless channels between all transmitters and receivers, the task of power control is to optimize the transmit power of each transmitter in a wireless network to ensure a reasonable signal-to-interference-and-noise ratio at each receiver. The signal-to-interference-and-noise ratio, in turn, determines the data rate of the corresponding link (link rate), using, for example, Shannon's capacity theorem. Common goals of power control include maximizing the sum rate of all links across the network and extending the lifetime of the network by avoiding draining the battery of certain devices.

To build a GNN for power control (Fig. 2a), the wireless network is typically modelled as an edge-weighted graph $(V,E,\mathbf{H})$, where a node $v \in V$ represents a transmitter–receiver pair and for all $v, u \in V$, a weighted edge $(v, u) \in E$ represents the channel state between the transmitter in v and the receiver in u (refs. 12–14,16,66,104). $\mathbf{H}$ denotes the channel state information matrix: the weight of a self-loop $\mathbf{H}_{vv}$ encodes the channel gain from a transmitter v to its intended receiver v. Whereas the weights of other edges, $\mathbf{H}_{vu}$ where $u \neq v$, represent the gains of interference channels between a transmitter v and other unintended receivers u. The problem of power control for sum-rate maximization can be formulated as to design a map $w(\cdot)$ that generates a power allocation $\mathbf{p}$ from a channel state information matrix $\mathbf{H}$, that is $\mathbf{p} = w(\mathbf{H})$, to maximize the rate of data transition. Finding the optimal power allocation $w^*(\cdot)$ is NP-hard[104]. Previous methods often adopted heuristics. GNNs can be used to learn a power control heuristic $\mathbf{p} = p(\mathbf{H};\boldsymbol{\Theta})$ with learnable parameters $\boldsymbol{\Theta}$ from data. The way to learn such GNNs is based on the principle of algorithmic unfolding[11–13,105]. Specifically, one can design a specific GNN with architecture inspired by the iterative algorithm weighted minimum mean square error (WMMSE)[106], denominated the unfolded WMMSE (UWMMSE)[12,105].

Link scheduling for network utility maximization. A pair of wireless transceivers that can directly talk to each other represent a link. In a wireless network with N transceivers, there could be at most N^2 links, but not all of them can be activated simultaneously. Link scheduling in wireless networks refers to the task of deciding which links in a wireless network should be activated at a given time slot (potentially along with other parameters such as transmit power or frequency).

Link scheduling (Fig. 2b) in multi-hop wireless networks under orthogonal multiple access is often defined on a conflict graph $(V,E,\mathbf{x})$, where a node $v \in V$ represents a link and the existence of an unweighted edge between two nodes, $(v_1,v_2) \in E$, indicates that the corresponding links cannot be activated simultaneously due to radio interference or sharing the same wireless device. A node feature $\mathbf{x}_v$ captures the utility of activating the corresponding link v.

The objective of link scheduling is to find a set of non-conflicting links with maximum total utility, which can be formulated as finding a maximum weighted independent set on the conflict graph[107]. As the maximum weighted independent set problem is known to be NP-hard[108], in practice it is solved approximately via fast heuristics, denoted as $\hat{\boldsymbol{y}} = h(V,E,\mathbf{x})$. Conventional heuristics include greedy search[109] and distributed local greedy search (LGS)[110]. GNNs can be used to augment link scheduling heuristics in a hybrid pipeline[71–73], where a GCN is followed by a conventional heuristic LGS. To be precise, we consider learnable topology-aware weights $\mathbf{z} = \Psi(V,E;\boldsymbol{\Theta})$ generated by the GCN $\Psi(\cdot;\boldsymbol{\Theta})$. These weights modify the utility of scheduling each corresponding link via the element-wise product $\mathbf{x} \odot \mathbf{z}$. The graph with modified utilities is then fed into a conventional heuristic $h(\cdot)$, which generates a discrete solution $\mathbf{y} = h(V,E,\mathbf{x} \odot \mathbf{z})$ guaranteed to be an independent set. Depending on the use cases, $h(\cdot)$ can be selected as different heuristics, for example, centralized rollout search (CRS) and distributed LGS are respectively used as the underlying heuristics of hybrid heuristics GCN-CRS and GCN-LGS by Zhao et al.[71]. Comparing approximation ratios of methods with regard to the optimal solutions generated by the Gurobi solver[111], the combination of GCNs with heuristics (denoted by the 'GCN-' prefix) outperforms the sole heuristic approaches and a message passing algorithm[112].

Other applications of GNNs in wireless systems. Other notable applications of GNNs in wireless systems include channel decoding in signal reception, packet routing and computational offloading. In channel decoding, the parity-check matrix can be represented as a factor graph, and processed with GNNs[67,68]. For packet routing in wireless networks, GNNs are able to close the optimality gaps of conventional heuristics for three different combinatorial optimization problems for routing[113] and can enhance the shortest-path biased backpressure routing[114]. GNNs can be used as efficient digital twins to predict the performance of routing, replacing computationally costly network simulators in quick evaluation of routing schemes[115,116]. In computational offloading[87], GNNs can be used to process both the computational context (subtask dependency relationship) and the edge network topology.

Experimetal particle physics

Since 2019, GNNs have been widely adopted in high-energy physics (HEP) across the energy, intensity and cosmic frontiers, thanks to their unparalleled ability to handle the irregularly structured, hierarchical and sparse data structures commonly found in the field[24–26]. Furthermore, there is substantial interest in applying GNN methods in real time, especially to select, filter and compress data using field programmable gate arrays (FPGAs) and application-specific integrated circuits, needed to deal with the extremely large data rates found in collider physics[117]. GNN applications in HEP can be categorized as node-level, edge-level or graph-level learning tasks[24–26]. Graph-structured data can be used for tracking charged particles, tagging jets, clustering calorimeter energy deposits and identifying signal events (Fig. 3).

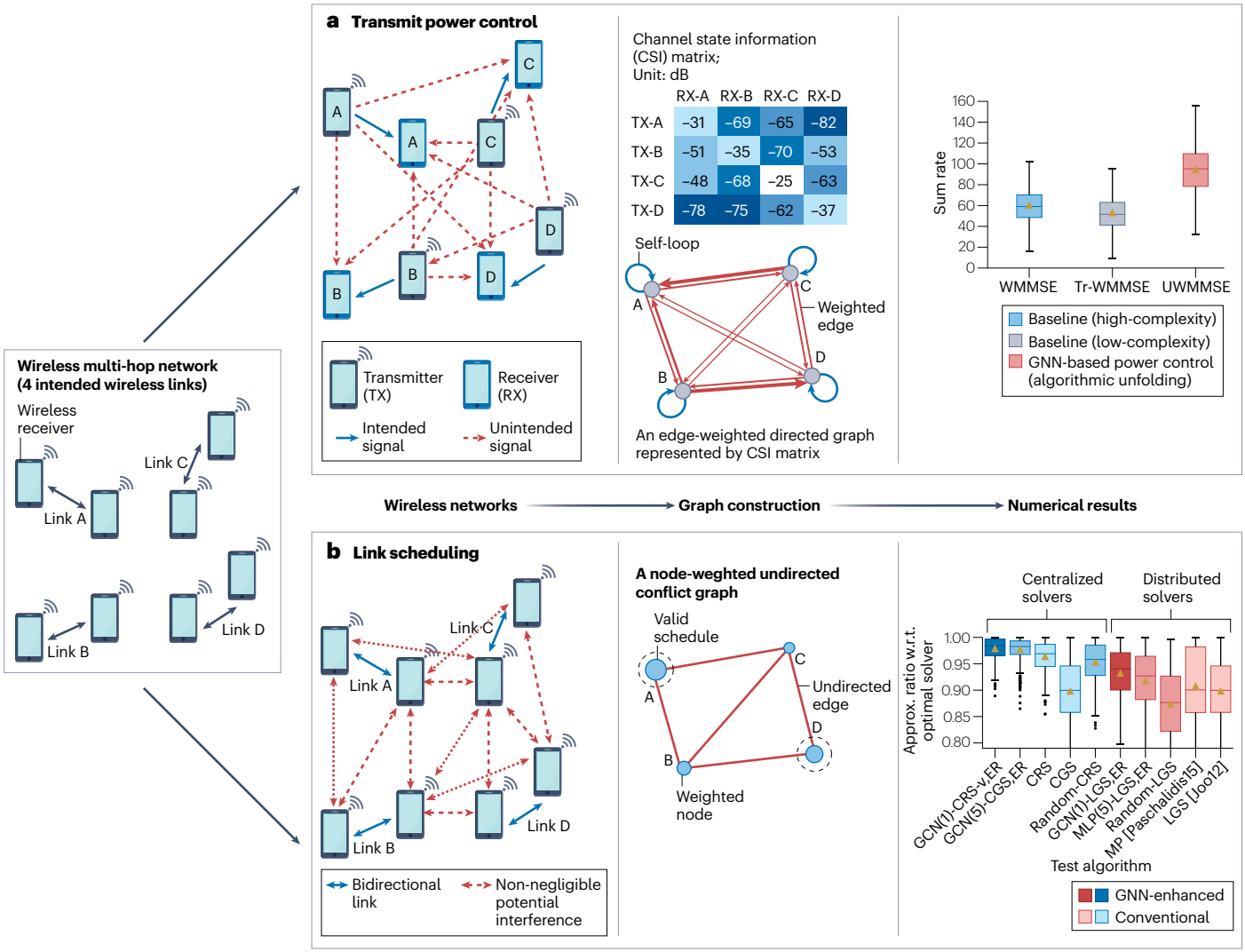


Fig. 2 | Graph modelling and performance evaluations of graph neural network-enhanced resource allocation in wireless multi-hop networks. **a**, Transmit power control under non-orthogonal multiple access. The generic transmit power control problem, its graph representation and the corresponding channel state information matrix (first two panels from the left). Sum rates achieved by an unfolded weighted minimum mean square error (UWMMSE) algorithm and two baseline methods in a numerical experiment with 10,000 instances of simulated multiple-input and multiple-output wireless networks with 20 pairs of 5-antenna transmitters and 3-antenna receivers, and Rayleigh fading channels under a low-noise regime[105] (right panel). The UWMMSE is composed of three unrolled weighted minimum mean square error (WMMSE) layers with each layer having two 2-layered graph convolutional networks (GCNs). The learnable components in UWMMSE enable the use of fewer layers than the iterations in WMMSE. The baseline WMMSE has 100 iterations, and the truncated WMMSE (Tr-WMMSE3) is just the classical WMMSE limited to 3 iterations (for comparison with the 3 layers in UWMMSE). Compared with WMMSE, UWMMSE increases the average sum rate by 61% with a per-instance execution time of only 1/24th that of WMMSE. **b**, Link scheduling under orthogonal multiple access. The generic link scheduling problem and its corresponding node-weighted undirected conflict graph (first two panels from the left). Comparison of the performance of GCN-enhanced maximum weighted independent set heuristics compared with the baselines (right panel). The approximation ratios of the tested heuristics with regard to the optimal solutions are generated by the Gurobi solver[111]. A set of 500 random graphs generated from the Erdős–Rényi (ER) model[76] are considered, with 100–300 nodes and uniformly distributed node weight $\mathbf{x}_v \in U(0,1)$. Consistently, the combination of GCNs with heuristics (denoted by the 'GCN-' prefix) outperform the sole heuristic approaches and a message passing algorithm[112]. In link scheduling[71], such a higher approximation ratio can be translated to higher throughput in the network. CGS, centralized greedy search; CRS, centralized rollout search; GNN, graph neural network; LGS, local greedy search; MLP, multilayer perceptron; MP, message passing.

Particle-flow reconstruction. Particle-flow reconstruction is an essential application of GNNs for node-level prediction in HEP[118]. This application comprehensively reconstructs each individual particle (charged and neutral hadrons, photons, electrons and muons) within an event using combined information from all sub-detectors. The task calls for an intricate understanding of the responses of various detectors (silicon trackers, electromagnetic and Hadron calorimeters, and muon detectors) to different particle types, providing a precise global

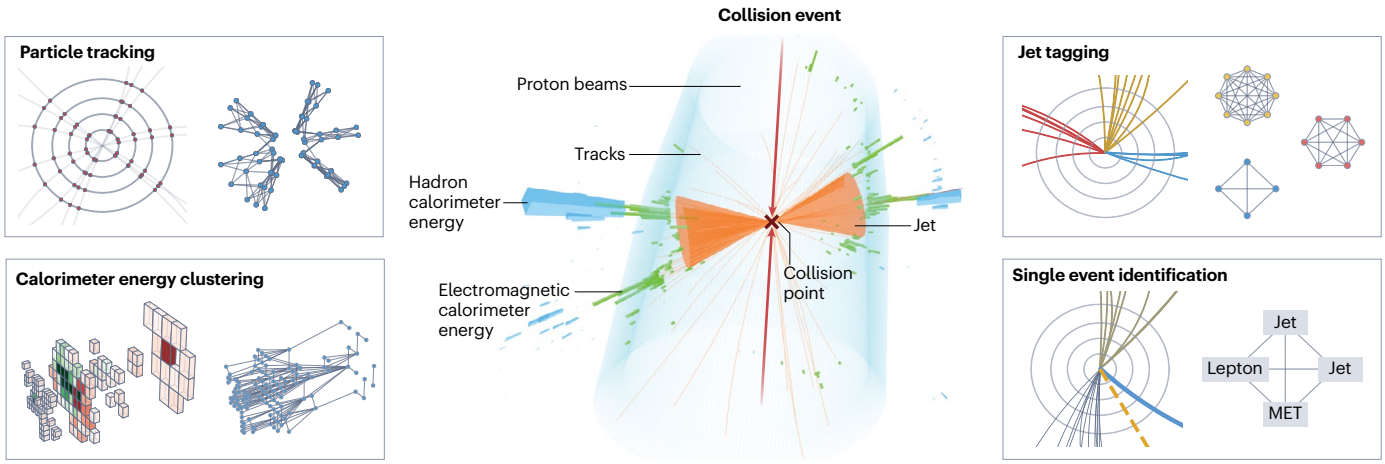


Fig. 3 | Visual representation of a collision event at the Large Hadron Collider and several graph neural network tasks. Proton beams (red arrows) cross at a collision point (brown cross). Outgoing particles make tracks (curved orange lines), energy deposits in the electromagnetic calorimeter (green boxes) and energy deposits in the Hadron calorimeter (blue boxes). The orange cone represents a cluster of tracks and energy deposits reconstructed as a jet. Four examples of constructing graph-structured data are shown for the tasks of charged particle tracking (top left), jet tagging (top right), calorimeter energy clustering (bottom left) and signal event identification (bottom right). MET, missing transverse momentum. The figure is adapted from ref. 24, CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/).

description of the final-state particles in an event. Traditional particle-flow reconstruction algorithms rely on a complex set of rules based on physicist-engineered features and are sensitive to detector imperfections and changing conditions. GNNs have emerged as a promising tool for modelling the complex underlying relationships between different detector measurements, embedded as nodes in a graph. GNNs allow simultaneous classification of particle types and regression of particle properties, leading to improved event reconstruction accuracy and improved physics measurements[119]. As reconstructed particles are often related to a collection of input detector measurements, specialized set-to-set loss functions, such as object condensation[120], and hypergraph networks[121] have been developed. A key advantage of GNN approaches is improved computational scalability in both training and inference[122]. Similar approaches can also be applied to identify particles originating from additional simultaneous collisions during particle beam crossing, known as pileup[123]. Recasting pileup identification as a node classification task, in which nodes represent particles and edges represent pairwise relations, GNNs are used to understand the complex structure of data to distinguish signal particles from pileup particles[53].

Charged particle tracking. In HEP data analysis it is crucial to estimate as accurately as possible the kinematics (such as the position, direction and momentum of the particles at their production points) of the particles produced in a collision event. Tracking devices, placed close to the beam collision area and immersed in a strong magnetic field, can provide high-precision position measurements from which the trajectories of charged particles can be determined. This task, known as charged particle tracking[124], is challenging because of the volume of data, noise and complexity of interactions with the detectors.

Traditional track reconstruction methods, based on combinatorial Kalman filters[125], are computationally intensive. Reformulated as an edge classification task, with tracker hits as nodes and edges indicating prospective track 'doublets' (pairs of hits belonging to a track), GNNs can exploit the relational structure of data, enabling efficient and accurate track reconstruction[56–58]. It is possible to deploy compressed and quantized GNNs with hundreds of nodes and thousands of edges with latency in the order of microseconds[126,127] with the HLS4ML Python package[128,129]. Nuclear physics experiments have also explored GNN tracking for trigger applications[130].

Jet tagging. Jets are sprays of particles produced through the strong force in high-energy collisions. Jets are complex manifestations of the originating quarks and gluons, but they contain structure and information that help us determine their origin. The task of jet tagging is to infer, on a statistical basis, the origin of a jet based on its measured characteristics. Thus, jet tagging can provide crucial information for various searches and measurements involving the decay products of top quarks, W and Z bosons, Higgs bosons and, potentially, new, undiscovered particles. GNNs address this as a graph classification task, where a graph represents a jet and nodes and edges represent particles and their geometric relations, respectively, capturing the underlying complex structural information of the particles within a jet and delivering improved performance[63,131,132]. The JEDI-net architecture[131] has been implemented as a low-latency FPGA implementation[133] for real-time applications. Because GNN jet tagging algorithms run on multiple jets per event, and potentially billions of simulated and real events, approaches to leverage coprocessors, such as graphic processing units (GPUs) and FPGAs, as a service to accelerate these algorithms have been pursued[134,135]. It is also possible to encode symmetries, such as Lorentz symmetry, or other physics-inspired inductive biases in GNN models[136,137].

Anomaly detection is a related task, which can be viewed as an unsupervised way to detect novel jets in a collision event. Rather than applying the standard supervised learning paradigm, the goal of anomaly detection is to train an algorithm to learn the distribution or characteristics of the most common backgrounds, and thus look for outliers that are different along some dimensions. Typically, (variational) autoencoders are used[138]. GNN-based autoencoders,

in which individual reconstructed particles are the nodes, have been applied to identify both anomalous events and jets[138,139]. Models equivariant to Lorentz transformations have also been explored[140]. Finally, there is notable interest in deploying such GNN anomaly detection algorithms in the real-time event selection system at the LHC[141].

Materials science

The versatility and flexibility of GNNs for irregular data have made them well suited for applications covering the breadth of the materials space in the materials sciences. The rapid introduction and adoption of GNNs in the materials sciences have been motivated by several critical challenges. Traditionally, evaluating materials for their potentially useful properties is a resource-consuming and time-consuming process often involving complex synthetic procedures and characterization techniques, or computationally expensive simulations with ab initio modelling (such as density functional theory). At the same time, building viable machine learning models to predict these properties often requires extensive feature engineering and domain-specific knowledge for a given material and property, which were rarely transferable across the materials space. In this domain, GNNs have been able to make substantial breakthroughs as broadly applicable models for materials and have been successfully used in property prediction and screening, molecular dynamics simulations and the inverse design of new materials[28,142] (Fig. 4a).

Material property prediction. Fundamentally, the structure of a material at the atomic level can be represented as a materials graph or crystal graph with atoms as nodes and with edges encoding spatial information between these atoms, such as their distances and angles. Structure–property relationships of materials can then be learned as either node-level tasks for atomic properties (for example, charge, magnetic moments, local electronic states) or as graph-level properties (for example, formation energies, band gaps, bulk moduli and a myriad of other functional properties). GNNs can be used to predict the properties of glassy materials, such as particle propensity, to understand the long-term dynamics of the system[143] or the band gaps of complex disordered crystals, aided using a multi-fidelity transfer learning approach[144]. GNNs have also successfully predicted the synthesizability of a material[145], which is a key property for screening which materials could be experimentally realized in the laboratory.

The architectures of GNNs have evolved over the years to better leverage the geometric information of atomic structures and their atomic local environments and improve the prediction accuracy of material properties (Fig. 4b). SchNet[60] and CGCNN[62] are the earliest examples of message passing GNN models for periodic systems encompassing distance-based edges encoding the material structure and node updates. Building on top of this message passing framework for materials, a succession of novel model architectures have been introduced with better geometric learning capability. For instance,

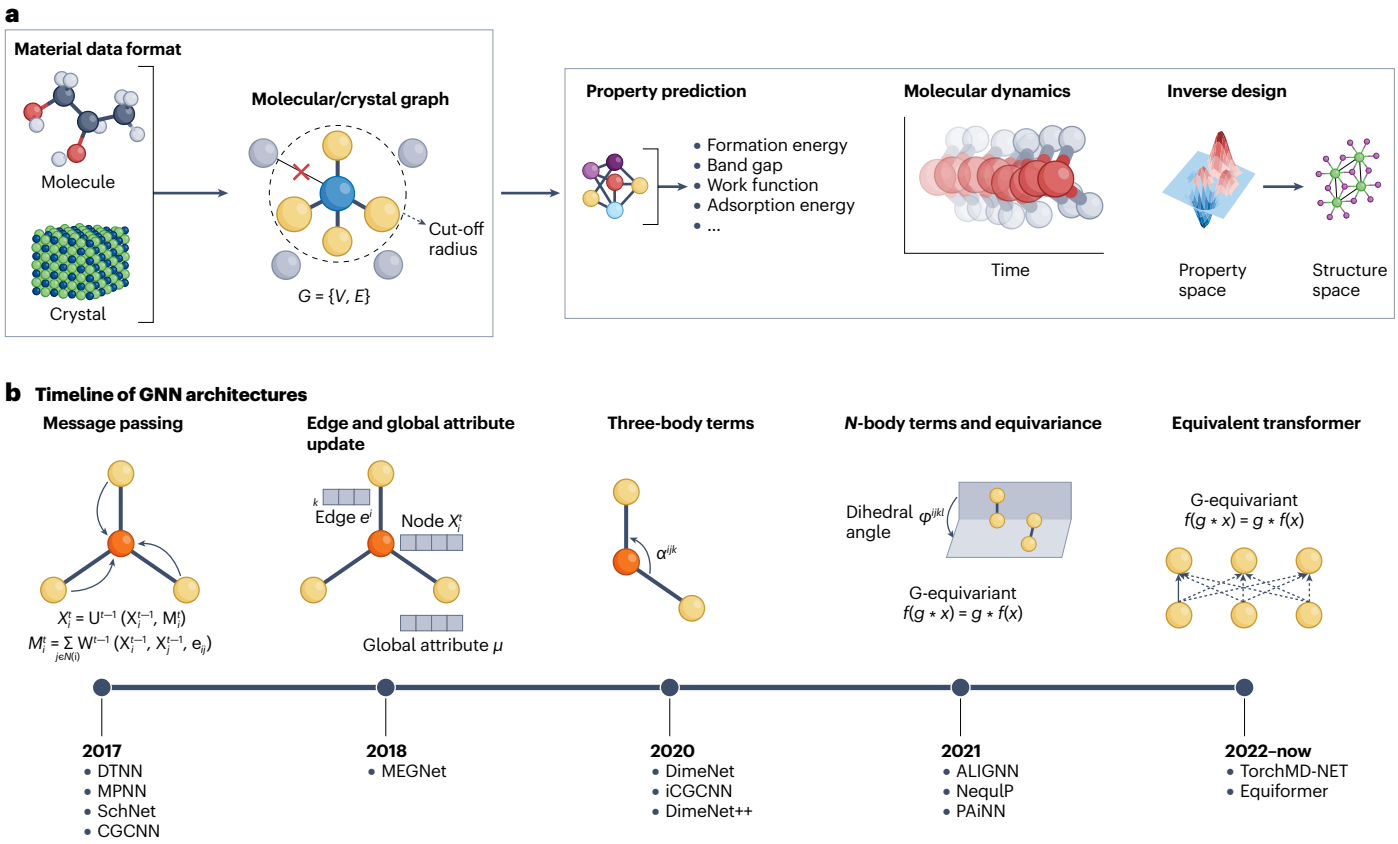


Fig. 4 | Molecular and crystal graphs and a chronological overview of graph neural networks used in material design. a, Molecules and crystals undergo an initial transformation into molecular and crystal graphs, which then serve as the input data for graph neural networks (GNNs). Main applications of GNNs in molecular and materials chemistry include property prediction, molecular dynamics and inverse design. **b**, Chronological overview of recent advancements in GNN architecture, featuring highlights of some notable added features and models. MPNN, message passing neural network.

MEGNet[146] incorporates additional features such as residual connections and edge and global state updates. Subsequently, it became evident that the existing models overlooked the spatial orientations between atoms, a piece of pivotal directional information for structural fidelity. This limitation motivated the development of models such as DimeNet[147], ALIGNN[148] and M3GNet (ref. 149), which include information about the angles between atoms. DimeNet incorporates angular embeddings directly in the message passing process whereas ALIGNN creates an additional line graph wherein nodes represent interatomic bonds and edges represent the bond angles. These models use scalar features such as interatomic distances and angles that are invariant to translation and rotation. Alongside such scalar features, other GNN models, such as NequIP[150] and PaiNN[151], encompass equivariant operations which preserve transformations. Beginning in 2022, equivariant transformers such as TorchMD-NET[152] and Equiformer[153] have emerged as a new class of models that combine the strengths of transformers and equivariant features.

Accelerating molecular dynamics simulation. A subset of the property prediction problem is the prediction of the total potential energy and atomic forces of the system, which can be used to perform geometry optimization and molecular dynamics simulations. Molecular dynamics serves as a powerful tool to understand the dynamical processes and evolution of materials over time at the atomic level. This process traditionally requires expensive quantum mechanical evaluation of the atomic forces for every time step. GNNs utilizing angular information and equivariance-preserving message passing provide an inexpensive method for obtaining forces for simulations, achieving near ab initio performance with accuracies of <0.1 eV $Å^{-1}$. Although other machine learning approaches have also been used to predict forces, such as structural descriptor-based methods[154], GNNs possess several appealing advantages. For example, the ability to provide compact embeddings of local atomic environments for systems or datasets containing many unique elements, which structural descriptors struggle to do[155]. This characteristic has led to the rise of 'universal potentials' which can be used for systems of arbitrary composition across the periodic table, rather than a narrow subset of elements. Successful demonstrations of universal potentials using GNNs such as M3GNet (ref. 149) and ALIGNN[156] have been performed, with these models showing broad applicability to materials with compositions across the periodic table. This approach could be used to greatly accelerate materials discovery by not only predicting stable materials from existing experimental databases of crystal structures but also performing optimization on new enumerated structures with no known structures[149].

Data-driven inverse design. Beyond using GNNs as a tool for high-throughput screening of materials and molecular dynamics, recent efforts have leveraged these architectures with generative models to generate entirely new materials in a data-driven manner, conditional on some desired property. GNNs provide a promising avenue for the data-driven inverse design of materials, a long-standing challenge in the field[157]. Using G-SchNet[158], for example, molecules can be constructed one atom at a time by relying on the GNN to predict the probability of the next atomic position. In another approach, CDVAE[159], crystal structures are generated via a diffusion process, where a GNN is trained to denoise a noisy structure. Using CDVAE, new materials that exhibit unique 2D structures[160] or superconducting properties[161] were discovered. A shared theme to these generative approaches for materials structures is the utilization of the GNNs to provide effective low-dimensional embeddings of the structures to be used in various decoder architectures.

Current challenges and opportunities. GNNs are an effective class of machine learning models for materials science for various different tasks. At the same time, there is still much room for improvement in these models. Known challenges of GNNs, such as over-smoothing and limitations of global pooling for graph-level predictions, could hamper the effective prediction of material properties. Moreover, as highlighted by the Open Catalyst competition[162], the prediction of properties of noisy or imprecise input structures is an ongoing grand challenge. Most critically, GNNs are also well known to be data-hungry models. This aspect presents a problem for materials science where datasets are generally many orders of magnitudes smaller than in other domains such as computer vision. Apart from purpose-built large-scale materials databases, most materials problems require machine learning models to work well with, at most, hundreds of labelled data samples. Consequently, new developments to GNN model architectures and methods for improving data efficiency and out-of-distribution performance such as model pre-training are needed to enable the widespread application of GNNs in this domain.

Biology

Networks (or graphs) are ubiquitous across all scales of biological systems (Fig. 5a). In the molecular world (0.1–10 nm), biomolecules such as proteins and small-molecule drugs are formed by atoms and bonds, which can be represented as graphs. Numerous protein functions are jointly carried out by protein complexes or pathways (10–100 nm), which are sub-graphs in the global protein–protein interaction networks. At the genomic level, gene regulatory networks extracted from transcriptomic data describe the inhibition, activation and co-expression relationships and genetic interactions among genes. Many biological processes (100 nm) are connected and contribute to various phenotypes at the cellular level (10 µm), or in tissues (1 cm) and complex organisms (10 cm) of humans (1 m). Popular databases organized such knowledge of biological processes in graph-like forms, such as Gene Ontology (GO)[163]. At the population level (1 m–10^3 km), individuals are also linked in a network through common genotypes, phenotypes, geolocations and other factors to study complex diseases.

The above biological entities and relations possess a higher degree of irregularity and heterogeneity characterized by the underlying complex topology. GNNs are well suited to model those types of biological data as the relational inductive bias encoded in GNNs naturally describes the graph-structured information in biological networks.

Protein structures and functions. GNNs provide natural ways to model protein structures, where amino acids or heavy atoms are represented as graph nodes and spatial relations (proximity or distance) are encoded as edges. AlphaFold[164], the state-of-the-art protein structure predictor, is arguably the most notable example of GNN application in biology. The key idea of AlphaFold is formulating protein structure prediction as a graph inference problem, where the edges to be inferred reflect the spatial proximity of residues in the 3D space (Fig. 5b). The remarkable prediction accuracy of AlphaFold, as demonstrated by comprehensive evaluations[164] and in the CASP14 challenge of protein structure prediction[165], has suggested the importance of domain-guided design principles for GNNs in real-world applications in biology.

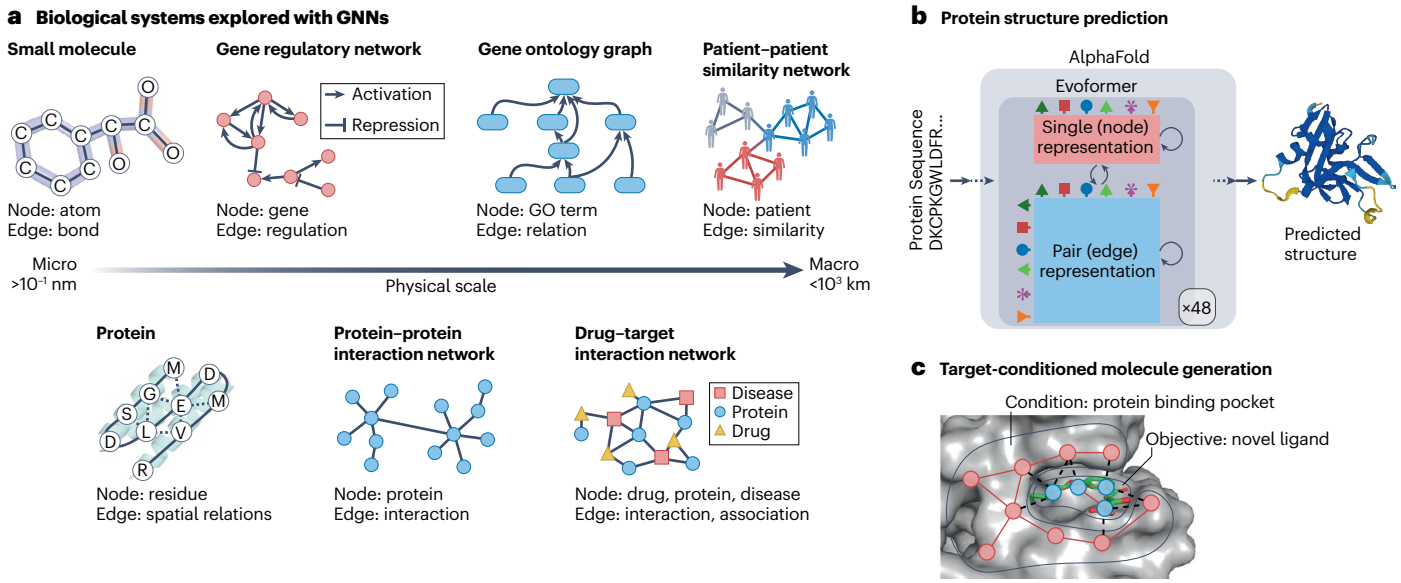


Fig. 5 | The ubiquity of networks (or graphs) in biological systems. **a**, Representative biological systems that can be naturally explored with graphical neural networks (GNNs) from nanoscale to macroscale: small molecules, proteins, gene regulatory networks, protein–protein interaction networks, Gene Ontology (GO) graph, heterogeneous drug–target interaction networks and patient–patient similarity networks. Nodes and edges are defined for every biological system. **b**, AlphaFold[29] is the state-of-the-art protein structure predictor. AlphaFold takes protein sequences as input and implements a message passing-based module called Evoformer to iteratively refine single (node) and pairwise (edge) representations of residues. It thus predicts the 3D coordinates for the input protein from its sequence, that is, p(structure|sequence) **c**, Target-conditioned molecule generation refers to the task of designing small-molecule drugs that have binding affinity with the target protein binding pocket. Conditioned on the graph built from the target, GNN-based generative models have been widely applied to generate novel ligands with desired target binding affinity[183–187].

In addition to protein structure prediction, GNNs are also widely used in structure-based protein sequence design (inverse folding) problems[30], where the goal is to design a protein sequence that can fold into a desired structure, that is, p(sequence|structure). Here, methods such as ProteinMPNN[30] used GNNs as the structure encoder to learn representations of the 3D structure input and guide the prediction of sequences. Invariant geometric features (such as distance, direction, dihedral angles and orientation frames) are often derived from the protein structure and used as features for nodes and edges to learn information-rich protein representations[166]. Furthermore, as a structure encoder, the representations learned by GNNs have substantially benefited a wide range of applications, including protein function annotation[167], fitness prediction[168], stability prediction[169] and many intermolecular predictions, such as protein–protein interaction[170], protein–ligand binding[171] and the binding pocket or interacting surface[172].

Apart from predictive models, generative models for proteins have recently been extensively studied by the community which focus on generating protein sequences or structures with a desired function (also known as protein design). For example, GNNs are used as structure encoders and decoders in other generative machine learning frameworks, such as diffusion models[173], to design novel protein structures with a specified property[174] or generate binding structure[175].

Small-molecule drug discovery. The chemical structure of a drug is naturally described as a graph, with atoms as nodes and chemical bonds as edges. For this reason, GNNs can be used to model small-molecule drugs for various applications in drug discovery, including property prediction[31,32,60], drug target identification[60], lead optimization[60] and others[176]. The applications of GNNs in small-molecule drug discovery can be categorized into prediction and generation tasks. In prediction tasks, GNNs are applied to learn the molecule to property relationship, such as the predictions of toxicity, efficacy, binding activity to targets and side effects[177–179]. General GNN architectures have been specialized to incorporate the domain knowledge of small molecules, such as including chemical information as node or edge features[31] and decomposing the molecular graph into semantic substructures known as junction trees[82]. The recent progress in geometric learning has also led to new GNN architectures, known as equivariant GNNs[180], for 3D molecular structure modelling, in which the non-linear transformations are specifically designed to respect the physical symmetry (such as rotation and translation) in the 3D space. High-quality benchmark datasets have been used to evaluate GNNs for molecular property prediction[61,181].

As with proteins, the generative modelling of molecules has also gained interest, and GNNs have been coupled with generative machine learning frameworks, such as variational autoencoder[82] and diffusion models[173], to generate an ensemble of molecular conformations[78] and molecules with desired functions[182]. A representative generation task is the target-conditioned molecule generation, where the goal is to design a small-molecule drug that can dock into the binding pocket of a given target protein (Fig. 5c). Recent studies paired equivariant GNN architectures with diffusion models to build models for molecule generation[183]. Conditioned on the protein binding pocket structures, equivariant GNNs are used to predict the atom types and coordinates of a fit molecule. The prediction process often involves multiple iterations of message passing to update the representation of the nodes and/or of

the edges and refine the predicted atom types and coordinates. Domain insights are also incorporated to guide molecule generation. For example, inspired by the traditions in the drug design process, some generative models[183] chose to decompose molecules into scaffolds and arms and generate them separately with decomposed priors in consideration of the different roles of ligand atoms. Enabled by GNNs and other generative machine learning frameworks, target-conditioned molecule generation has shown promising results for generating molecules that are chemically valid and novel with improved properties[183–187]. Nevertheless, many interesting questions and challenges remain, such as ensuring the synthesizability of generated molecules, given commercially available molecules as building blocks, and simultaneously optimizing multiple properties in the generation process.

Other applications of GNNs in biology. In addition to the applications of GNNs in modelling proteins and small molecules, research efforts have explored the potential of GNNs in other levels of biological systems to address fundamental questions in biology and healthcare. Some examples include single-cell RNA data analyses — GNN-based models have been developed to learn low-dimensional embeddings that capture the cell–cell relationships characterized by gene expression and transcriptional regulation information in single-cell RNA data, which inform downstream tasks such as data imputation, cell clustering and regulatory network inference[188,189]; network medicine — GNNs have increasingly become a key instrument in network medicine, aiding in the development of predictive models and therapeutic strategies[190,191] and the prediction of drug–disease associations[55], drug–drug and drug–target interactions[179,192], which are problems crucial for important applications such as identifying biomarkers of disease and repurposing drugs[54,193]; and healthcare and population health — beyond the molecular and cellular scales, GNNs also have notable impacts on a larger scale in healthcare and population health, such as processing and analysing biomedical imaging[194,195] and electronic health records[196,197].

Software and hardware

With the advancement of GNN algorithms and their applications, computational challenges, encompassing both training and inference, have rapidly emerged as limiting factors. For instance, the challenge of handling extra-large graph sizes affects memory and bandwidth. Scalable and distributed training poses difficulties for current training platforms and strategies. The demand for high-throughput and/or low-latency inference pushes the boundaries of existing hardware.

Despite these challenges, recent innovations in software and hardware have stimulated the development of new GNN architectures and their deployment mechanisms. On standard GPU platforms, leading frameworks such as PyTorch Geometric (PyG)[198] and Deep Graph Library (DGL)[199] excel in optimizing data throughput. They leverage GPU acceleration, sophisticated CUDA kernels and efficient batching methods.

Furthermore, certain applications, especially in domains such as HEP, require exceptional inference speeds combined with energy efficiency. For such requirements, custom hardware designs using FPGAs or application-specific integrated circuits often outperform GPUs and CPUs by marked margins. The HLS4ML package[129], for instance, is specifically designed to enhance machine learning inference on FPGAs and is currently integrating GNN operations from PyG[198]. Simultaneously, FlowGNN[200] offers a specialized FPGA-based solution, optimized for a diverse set of GNN models, emphasizing graph structures and computational sparsity.

Open challenges and outlook

Although GNNs show remarkable promise in various electrical engineering and scientific applications, there are opportunities for further enhancement to fully realize their potential. These areas of improvement also pave the way for compelling future research trajectories in GNNs.

Data scarcity

Similar to most deep learning methods, abundant high-quality data are an essential factor in GNN performance. Data scarcity is a shared problem between electrical engineering and the scientific domains and hampers GNNs from unleashing their full potential in real-world problems. Considering, for example, using GNNs for EDA, to achieve accurate circuit property prediction, one must first gather a substantial volume of labelled data, often through time-consuming simulation or synthesis[201]. Yet a challenge arises as there is no established infrastructure to accelerate such data collection. Moreover, there is no standard graph format that raw EDA data can be seamlessly transformed into[103]. The varied representations, such as directed acylic graphs (DAGs)[202] or directed hypergraphs[90], complicate benchmarking processes. A similar problem has been shown with data acquisition in wireless networks[203], where real-time communication data are often extracted in various time granularities depending on the capturing device. The development of standardized data collection infrastructures is then necessary to facilitate standard data collection.

Data scarcity issues also hinder GNNs from having a greater impact on scientific applications. In chemistry and biology domains, data labelling from the wet laboratory is expensive in both time and resources[204]. Fortunately, abundant data are sometimes available from well-studied areas, but these may have different distributions compared with less explored yet scientifically intriguing regions. Therefore, transfer learning of GNNs and enhancing out-of-distribution generalization of GNNs have emerged as viable strategies to address the data scarcity issue[205]. Notably, with such a goal, the field of electrical engineering offers insights: researchers with a rich history in information theory have presented multiple tools that could help understand the generalization capability of GNNs and foster the development of more adaptable GNNs. A case in point is the information bottleneck method[206], integrated into GNN model training in 2022 (ref. 207). The model captures only the essential information from data features, enhancing its prediction accuracy, out-of-distribution generalizability and model interpretation.

Real-time computation

Real-time inference boasting minimal latency is indispensable for numerous practical GNN applications. For instance, in HEP, GNNs must process particle hit data within a strict time frame matching that of real-time LHC experiments. Given that the detector harvests collision data every 25 ns, any GNN latency surpassing this limit risks overflowing memory buffers, leading to data loss[208]. Similarly, tasks such as power allocation and link scheduling in wireless networks demand instantaneous processing. A case in point is the ultra-reliable and low-latency communications in 5G, which mandates sub-1 ms latency while sustaining a downlink speed of 20 Gbps[209]. Therefore, the need to expedite GNNs to meet these real-time requisites cannot be overstated.

Electrical engineering techniques, especially in hardware acceleration, emerge as highly promising avenues to address the challenges posed by GNNs. Computing hardware can be properly tailored to deal with the irregular nature of graph data and message passing mechanisms innate to GNNs. These tailored hardware solutions, whether

through FPGAs or application-specific integrated circuits, promise speed and energy enhancements over conventional GPUs and CPUs. Worthwhile exploration directions encompass process-in-memory and emerging memory-based techniques[210,211], exploiting graph structures and sparsity[200,212] and conducting algorithm and hardware co-design.

The challenge of graph data privacy

The rising utilization of GNNs in areas with sensitive graph data, such as healthcare data[213] and device data in a fragile network system, highlights the imperative to address the relevant privacy of graph data. When applying GNNs to this type of data, it is crucial to ensure sensitive data details remain confidential. However, the inherent interdependence of graph data exacerbates the privacy challenge: the leakage risk intensifies because one data point's information might be intertwined with another's. Recognizing this complexity, there has been a surge in efforts to develop privacy-preserving GNN models[214,215]. These models strive to ensure their parameters remain insensitive to specific nodes or edges in the training graph. Machine unlearning, another frontier in privacy preservation, emphasizes the rights of data providers to retract their data from the dataset used for training a model[216,217]. This field explores efficient ways to modify trained models when users revoke their data, with the goal of approximating a model trained without the rescinded data. Whereas effort has been made in creating private GNN models and pursuing GNN unlearning[218], many current methods either exhibit heightened privacy vulnerability or are built upon oversimplified architectures. There remains a notable path ahead.

Privacy concerning graph data typically hinges on the indistinguishability of algorithmic (say GNN) outputs when the input graphs vary by a single node or edge. Such privacy assurance is contingent on the comparison of the distributions of such outputs[219].

Conclusions

GNNs, with their intrinsic ability to handle sparse, irregular and relational data, have emerged as a potent instrument for the intricate graph data found in diverse applications.

Traditional algorithms in electronic design, wireless communication and power systems often involve time-consuming optimization steps, which can be swapped out for the faster methods built upon GNNs. This change offers electrical engineering researchers a chance to boost their work with the efficiency of GNNs. There is also room for electrical engineering experts to make GNNs even better. They can tackle current GNN challenges by designing dedicated hardware for quick, on-the-spot applications or by designing better model architectures or more principled training strategies for areas where there are not many labels but need GNNs to be out-of-distribution generalizable.

As we find ourselves balancing between classical electrical engineering and the new world of machine learning, GNNs, backed by careful research and teamwork, have the potential to bring fresh, exciting changes to the field.

Published online: 05 August 2024

References

1. Tanenbaum, A. S. *Computer Networks* (Pearson Education India, 2003).
2. Shannon, C. E. *Claude Elwood Shannon: Collected Papers* (IEEE, 1993).
3. Akpakwu, G. A., Silva, B. J., Hancke, G. P. & AbuMahfouz, A. M. A survey on 5G networks for the Internet of Things: communication technologies and challenges. *IEEE Access* **6**, 3619–3647 (2017).
4. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
5. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (eds Burstein, J. et al.) 4171–4186 (ACL, 2019).
6. OpenAI et al. Gpt-4 technical report. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2303.08774 (2023).
7. Gori, M., Monfardini, G. & Scarselli, F. A new model for learning in graph domains. In *Proc. 2005 IEEE International Joint Conference on Neural Networks* Vol. 2 729–734 (IEEE, 2005).
8. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **20**, 61–80 (2008).
9. Kipf, T. N. & Welling, M. Variational graph autoencoders. In *NIPS Workshop on Bayesian Deep Learning* (NIPS, 2016).
10. Shen, Y., Shi, Y., Zhang, J. & Letaief, K. B. A graph neural network approach for scalable wireless power control. In *2019 IEEE Globecom Workshops* 1–6 (IEEE, 2019).
11. Chowdhury, A., Verma, G., Rao, C., Swami, A. & Segarra, S. ML-aided power allocation for Tactical MIMO. In *2021 IEEE Military Communications Conference* 273–278 (IEEE, 2021).
12. Chowdhury, A., Verma, G., Rao, C., Swami, A. & Segarra, S. Unfolding WMMSE using graph neural networks for efficient power allocation. *IEEE Trans. Wirel. Commun.* **20**, 6004–6017 (2021).
13. Li, B., Verma, G. & Segarra, S. Graph-based algorithm unfolding for energy-aware power allocation in wireless networks. *IEEE Trans. Wirel. Commun.* **22**, 1359–1373 (2022).
This paper discusses the use of the algorithm unrolling framework to address the power allocation problem in the application of GNNs in wireless networks.
14. Wang, Z., Eisen, M. & Ribeiro, A. Learning decentralized wireless resource allocations with graph neural networks. *IEEE Trans. Signal. Process.* **70**, 1850–1863 (2022).
15. Shen, Y., Zhang, J., Song, S. H. & Letaief, K. B. Graph neural networks for wireless communications: from theory to practice. *IEEE Trans. Wirel. Commun.* **22**, 3554–3569 (2023).
16. Owerko, D., Gama, F. & Ribeiro, A. Optimal power flow using graph neural networks. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing* 5930–5934 (IEEE, 2020).
17. Owerko, D., Gama, F. & Ribeiro, A. Predicting power outages using graph neural networks. In *IEEE Global Conference on Signal and Information Processing* 743–747 (IEEE, 2018).
18. Donon, B. et al. Neural networks for power flow: graph neural solver. *Electr. Power Syst. Res.* **189**, 106547 (2020).
19. Ustun, E., Deng, C., Pal, D., Li, Z. & Zhang, Z. Accurate operation delay prediction for FPGA HLS using graph neural networks. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* 1–9 (IEEE, 2020).
20. Xie, Z. et al. Preplacement net length and timing estimation by customized graph neural network. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **41**, 4667–4680 (2022).
21. Liu, M. et al. Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* 1372–1377 (IEEE, 2021).
22. Guo, Z. et al. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proc. 59th ACM/IEEE Design Automation Conference* 1207–1212 (ACM, 2022).
23. Yang, Z. et al. Versatile multi-stage graph neural network for circuit representation. In *Proc. 36th International Conference on Neural Information Processing Systems* 20313–20324 (Curran Associates Inc., 2022).
24. Shlomi, J., Battaglia, P. & Vlimant, J.-R. Graph neural networks in particle physics. *Mach. Learn. Sci. Technol.* **2**, 021001 (2020).
A timing engine inspired graph neural network model for pre-routing slack prediction.
25. Duarte, J. & Vlimant, J.-R. Graph neural networks for particle tracking and reconstruction. In *Artificial Intelligence for High Energy Physics* 387 (World Scientific, 2022).
26. DeZoort, G., Battaglia, P. W., Biscarat, C. & Vlimant, J.-R. Graph neural networks at the Large Hadron Collider. *Nat. Rev. Phys.* **5**, 281 (2023).
27. Fung, V., Zhang, J., Juarez, E. & Sumpter, B. G. Benchmarking graph neural networks for materials chemistry. *npj Comput. Mater.* **7**, 84 (2021).
28. Reiser, P. et al. Graph neural networks for materials science and chemistry. *Commun. Mater.* **3**, 93 (2022).
29. Baek, M. et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science* **373**, 871–876 (2021).
30. Dauparas, J. et al. Robust deep learning–based protein sequence design using ProteinMPNN. *Science* **378**, 49–56 (2022).
31. Stokes, J. M. et al. A deep learning approach to antibiotic discovery. *Cell* **180**, 688–702 (2020).
32. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. In *Proc. 31st International Conference on Neural Informaton Processing Systems* 1025–1035 (Curran Associates Inc., 2017).
33. Veličković, P. et al. Graph attention networks. In *International Conference on Learning Representations* (ICLR, 2018).
34. Battaglia, P. W. et al. Relational inductive biases, deep learning, and graph networks. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1806.01261 (2018).
35. Defferrard, M., Bresson, X. & Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. 30th International Conference on Neural Information Processing Systems* 3844–3852 (Curran Associates Inc., 2016).
36. Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. & Vandergheynst, P. Geometric deep learning: going beyond Euclidean data. *IEEE Signal. Process. Mag.* **34**, 18–42 (2017).
37. Chien, E., Peng, J., Li, P. & Milenkovic, O. Adaptive universal Generalized PageRank graph neural network. In *International Conference on Learning Representations* (ICLR, 2021).

38. Wang, X. & Zhang, M. How powerful are spectral graph neural networks. In *Proc. 39th International Conference on Machine Learning* 23341–23362 (ICML, 2022).
39. Corso, G., Cavalleri, L., Beaini, D., Liò, P. & Veličković, P. Principal neighbourhood aggregation for graph nets. In *Proc. 34th International Conference on Neural Information Processing Systems* 13260–13271 (Curran Associates Inc., 2020).
40. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**, 359–366 (1989).
This fundamental work presents the architecture of GNNs, showing how it can be represented and implemented in a message passing process.
41. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations* (ICLR, 2019).
42. Morris, C. et al. Weisfeiler and Leman go neural: higher order graph neural networks. In *Proc. 33rd AAAI Conference on Artificial Intelligence* 4602–4609 (AAAI, 2019).
43. Maron, H., Ben-Hamu, H., Shamir, N. & Lipman, Y. Invariant and equivariant graph networks. In *International Conference on Learning Representations* (ICLR, 2019).
44. Li, P., Wang, Y., Wang, H. & Leskovec, J. Distance encoding: design provably more powerful neural networks for graph representation learning. In *Proc. 34th International Conference on Neural Information Processing Systems* 4465–4478 (Curran Associates Inc., 2020).
45. Bouritsas, G., Frasca, F., Zafeiriou, S. & Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 657–668 (2022).
46. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations* (ICLR, 2017).
47. Chen, D. et al. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proc. AAAI Conference on Artificial Intelligence* 3438–3445 (AAAI, 2020).
48. Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X. & Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations* (ICLR, 2022).
49. Alon, U. & Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations* (ICLR, 2021).
50. Chen, K., Hu, J., Zhang, Y., Yu, Z. & He, J. Fault location in power distribution systems via deep graph convolutional networks. *IEEE J. Sel. Areas Commun.* **38**, 119–131 (2019).
51. de Freitas, J. T. & Coelho, F. G. F. Fault localization method for power distribution systems based on gated graph neural networks. *Electr. Eng.* **103**, 2259–2266 (2021).
52. Arjona Martínez, J., Cerri, O., Pierini, M., Spiropulu, M. & Vlimant, J.-R. Pileup mitigation at the Large Hadron Collider with graph neural networks. *Eur. Phys. J. Plus* **134**, 333 (2019).
53. Li, T. et al. Semi-supervised graph neural networks for pileup noise removal. *Eur. Phys. J. C.* **83**, 99 (2023).
54. Luo, Y. et al. A network integration approach for drug–target interaction prediction and computational drug repositioning from heterogeneous information. *Nat. Commun.* **8**, 573 (2017).
55. Yu, Z., Huang, F., Zhao, X., Xiao, W. & Zhang, W. Predicting drug–disease associations through layer attention graph convolutional network. *Brief. Bioinform.* **22**, bbaa243 (2021).
56. Farrell, S. et al. Novel deep learning methods for track reconstruction. In *International Workshop Connecting The Dots* (2018).
57. Ju, X. et al. Performance of a geometric deep learning pipeline for HL-LHC particle tracking. *Eur. Phys. J. C* **81**, 876 (2021).
58. DeZoort, G. et al. Charged particle tracking via edgeclassifying interaction networks. *Comput. Softw. Big Sci.* **5**, 26 (2021).
59. Wu, N., Yang, H., Xie, Y., Li, P. & Hao, C. High-level synthesis performance prediction using GNNs: Benchmarking, modeling, and advancing. In *Proc. 59th ACM/IEEE Design Automation Conference* 49–54 (ACM, 2022).
60. Schütt, K. et al. SchNet: a continuous-filter convolutional neural network for modeling quantum interactions. In *Proc. 31st International Conference on Neural Information Processing Systems* 992–1002 (Curran Associates Inc., 2017).
61. Wu, Z. et al. MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
62. Xie, T. & Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.* **120**, 145301 (2018).
63. Qu, H. & Gouskos, L. ParticleNet: jet tagging via particle clouds. *Phys. Rev. D* **101**, 056019 (2020).
64. Guo, J., Li, J., Li, T. & Zhang, R. Boosted Higgs Boson jet reconstruction via a graph neural network. *Phys. Rev. D* **103**, 116025 (2021).
65. Eisen, M. & Ribeiro, A. Optimal wireless resource allocation with random edge graph neural networks. *IEEE Trans. Signal. Process.* **68**, 2977–2991 (2020).
66. Owerko, D., Gama, F. & Ribeiro, A. Unsupervised optimal power flow using graph neural networks. In *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 6885–6889 (IEEE, 2024).
67. Nachmani, E. & Wolf, L. Hyper-graph-network decoders for block codes. In *Proc. 33rd International Conference on Neural Information Processing Systems* 2329–2339 (Curran Associates Inc., 2019).
68. Cammerer, S., Hoydis, J., Aoudia, F. A. & Keller, A. Graph neural networks for channel decoding. In *2022 IEEE Globecom Workshops* 486–491 (IEEE, 2022).
69. Chen, T. et al. Learning to optimize: a primer and a benchmark. *J. Mach. Learn. Res.* **23**, 8562–8620 (2022).
70. Monga, V., Li, Y. & Eldar, Y. C. Algorithm unrolling: interpretable, efficient deep learning for signal and image processing. *IEEE Signal. Process. Mag.* **38**, 18–44 (2021).
71. Zhao, Z., Verma, G., Rao, C., Swami, A. & Segarra, S. Link scheduling using graph neural networks. *IEEE Trans. Wirel. Commun.* **22**, 3997–4012 (2022).
72. Zhao, Z., Verma, G., Swami, A. & Segarra, S. Delay-oriented distributed scheduling using graph neural networks. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing* 8902–8906 (IEEE, 2022).
73. Zhao, Z., Verma, G., Rao, C., Swami, A. & Segarra, S. Distributed scheduling using graph neural networks. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing* 4720–4724 (IEEE, 2021).
74. Kahng, A. B., Lienig, J., Markov, I. L. & Hu, J. *VLSI Physical Design: From Graph Partitioning to Timing Closure* 312 (Springer, 2011).
75. Callister Jr, W. D. & Rethwisch, D. G. *Fundamentals of Materials Science and Engineering: An Integrated Approach* (Wiley, 2020).
76. Erdős, P. & Rényi, A. On random graphs I. *Publ. Math. Debr.* **6**, 290–297 (1959).
77. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. Adversarial autoencoders. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1511.05644 (2015).
78. Xu, M. et al. Geodiff: a geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations* (ICLR, 2022).
79. Vignac, C. et al. Digress: discrete denoising diffusion for graph generation. In *International Conference on Learning Representations* (ICLR, 2023).
80. Mercado, R. et al. Graph networks for molecular design. *Mach. Learn. Sci. Technol.* **2**, 025023 (2021).
81. Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R. & Jensen, K. F. Generative models for molecular discovery: recent advances and challenges. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **12**, e1608 (2022).
82. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *Proc. 35th International Conference on Machine Learning* 2323–2332 (ICML, 2018).
83. Mirhoseini, A. et al. A graph placement methodology for fast chip design. *Nature* **594**, 207–212 (2021).
84. Cheng, R. et al. The policy-gradient placement and generative routing neural networks for chip design. In *Proc. 36th International Conference on Neural Information Processing Systems* 26350–26362 (Curran Associates Inc., 2022).
85. Chen, T., Zhang, G. L., Yu, B., Li, B. & Schlichtmann, U. Machine learning in advanced IC design: a methodological survey. *IEEE Des. Test* **40**, 17–33 (2022).
This review covers the integration of deep learning tools with conventional optimization algorithm frameworks to enhance the resolution of signal and image processing tasks through data-driven approaches.
86. Sánchez, D., Servadei, L., Kiprit, G. N., Wille, R. & Ecker, W. A comprehensive survey on electronic design automation and graph neural networks: theory and applications. *ACM Trans. Des. Autom. Electron. Syst.* **28**, 1–27 (2023).
87. Zhang, J. et al. Fine-grained service offloading in B5G/6G collaborative edge computing based on graph neural networks. In *IEEE International Conference on Communications* 5226–5231 (IEEE, 2022).
88. Ma, Y., He, Z., Li, W., Zhang, L. & Yu, B. Understanding graphs in EDA: from shallow to deep learning. In *Proc. 2020 International Symposium on Physical Design* 119–126 (ACM, 2020).
89. Agnesina, A., Chang, K. & Lim, S. K. VLSI placement parameter optimization using deep reinforcement learning. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (IEEE, 2020).
90. Lu, Y.-C., Pentapati, S. & Lim, S. K. The law of attraction: Affinity-aware placement optimization using graph neural networks. In *Proc. 2021 International Symposium on Physical Design* 7–14 (ACM, 2021).
91. Lu, Y.-C., Siddhartha, N., Khandelwal, V. & Lim, S. K. Doomed run prediction in physical design by exploiting sequential flow and graph learning. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* 1–9 (IEEE, 2021).
92. Kirby, R., Godil, S., Roy, R. & Catanzaro, B. CongestionNet: routing congestion prediction using deep graph neural networks. In *27th International Conference on Very Large Scale Integration (VLSI-SoC)* 217–222 (IEEE, 2019).
93. Maji, S., Budak, A. F., Poddar, S. & Pan, D. Z. Toward end-to-end analog design automation with ML and data-driven approaches. In *Proc. 29th Asia and South Pacific Design Automation Conference* 657–664 (IEEE, 2024).
94. Zhu, K., Chen, H., Liu, M. & Pan, D. Z. Tutorial and perspectives on MAGICAL: a silicon-proven opensource analog IC layout system. *IEEE Trans. Circuits Syst. II: Express Br.* **70**, 715–720 (2023).
95. Kunal, K. et al. ALIGN: Open-source analog layout automation from the ground up. In *Proc. 56th Annual Design Automation Conference 2019* 1–4 (ACM, 2019).
96. Wang, H. et al. GCN-RL circuit designer: transferable transistor sizing with graph neural networks and reinforcement learning. In *57th ACM/EDAC/IEEE Design Automation Conference* 1–6 (IEEE, 2020).
97. Dong, Z. et al. CktGNN: circuit graph neural network for electronic design automation. In *International Conference on Learning Representations* (ICLR, 2023).
98. Zhang, G., He, H. & Katabi, D. Circuit-GNN: graph neural networks for distributed circuit design. In *Proc. 36th International Conference on Machine Learning* 7364–7373 (ICML, 2019).
99. Ren, H., Kokai, G. F., Turner, W. J. & Ku, T.-S. ParaGraph: layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)* 1–6 (IEEE, 2020).

100. Li, Y. et al. A customized graph neural network model for guiding analog IC placement. In *Proc. 39th International Conference on Computer-Aided Design* 1–9 (ACM, 2020). **This groundbreaking work discusses the application of GNNS and RL to EDA, solving the global placement problem in chip design and outperforming the state-of-the-art method for this task**.
101. Chen, H. et al. Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)* 1243–1248 (IEEE, 2021).
102. Cao, W., Benosman, M., Zhang, X. & Ma, R. Domain knowledge-infused deep learning for automated analog/radio-frequency circuit parameter optimization. In *59th ACM/IEEE Design Automation Conference* 1015–1020 (ACM, 2022).
103. Shi, W. et al. RobustAnalog: fast variation-aware analog circuit design via multi-task RL. In *Proc. 2022 ACM/IEEE Workshop on Machine Learning for CAD* 35–41 (ACM, 2022).
104. Luo, Z.-Q. & Zhang, S. Dynamic spectrum management: complexity and duality. *IEEE J. Sel. Top. Signal. Process.* **2**, 57–73 (2008).
105. Chowdhury, A., Verma, G., Swami, A. & Segarra, S. Deep graph unfolding for beamforming in MU-MIMO interference networks. *IEEE Trans. Wirel. Commun.* **23**, 4889–4903 (2023).
106. Shi, Q., Razaviyayn, M., Luo, Z.-Q. & He, C. An iteratively weighted MMSE approach to distributed sumutility maximization for a MIMO interfering broadcast channel. *IEEE Trans. Signal. Process.* **59**, 4331–4340 (2011).
107. Tassiulas, L. & Ephremides, L. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Autom. Control.* **37**, 1936–1948 (1992).
108. Joo, C., Sharma, G., Shroff, N. B. & Mazumdar, R. R. On the complexity of scheduling in wireless networks. *Eurasip J. Wirel. Commun. Netw.* **2010**, 418934 (2010).
109. Dimakis, A. & Walrand, J. Sufficient conditions for stability of longest-queue-first scheduling: second-order properties using fluid limits. *Adv. Appl. Probab.* **38**, 505–521 (2006).
110. Joo, C. & Shroff, N. B. Local greedy approximation for scheduling in multihop wireless networks. *IEEE Trans. Mob. Comput.* **11**, 414–426 (2012).
111. Gurobi Optimization. Gurobi optimizer reference manual. *Gurobi* https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.0/refman.pdf (2020).
112. Paschalidis, I. C., Huang, F. & Lai, W. A message-passing algorithm for wireless network scheduling. *IEEE/ACM Trans. Netw.* **23**, 1528–1541 (2015).
113. Zhao, Z., Swami, A. & Segarra, S. Graph-based deterministic policy gradient for repetitive combinatorial optimization problems. In *International Conference on Learning Representations* (ICLR, 2023).
114. Zhao, Z., Radojicic, B., Verma, G., Swami, A. & Segarra, S. Delay-aware backpressure routing using graph neural networks. In *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 4720–4724 (IEEE, 2023).
115. Rusek, K., Suárez-Varela, J., Almasan, P., Barlet-Ros, P. & Cabellos-Aparicio, A. RouteNet: leveraging graph neural networks for network modeling and optimization in SDN. *IEEE J. Sel. Areas Commun.* **38**, 2260–2270 (2020).
116. Li, B. et al. Learnable digital twin for efficient wireless network evaluation. In *2023 IEEE Military Communications Conference (MILCOM)* 661–666 (IEEE, 2023).
117. Deiana, A. M. et al. Applications and techniques for fast machine learning in science. *Front. Big Data* **5**, 787421 (2022).
118. Sirunyan, A. M. et al. Particle-flow reconstruction and global event description with the CMS detector. *J. Instrum.* **12**, P10003 (2017).
119. Pata, J., Duarte, J., Vlimant, J.-R., Pierini, M. & Spiropulu, M. MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *Eur. Phys. J. C* **81**, 381 (2021).
120. Kieseler, J. Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph and image data. *Eur. Phys. J. C* **80**, 886 (2020).
121. Di Bello, F. A. et al. Reconstructing particles in jets using set transformer and hypergraph prediction networks. *Eur. Phys. J. C* **83**, 596 (2023).
122. Pata, J. et al. Scalable neural network models and terascale datasets for particle-flow reconstruction. Preprint at *arXiv* https://doi.org/10.21203/rs.3.rs-3466159/v1 (2023).
123. Sirunyan, A. M. et al. Pileup mitigation at CMS in 13 TeV data. *J. Instrum.* **15**, P09018 (2020).
124. Strandlie, A. & Frühwirth, R. Track and vertex reconstruction: from classical to adaptive methods. *Rev. Mod. Phys.* **82**, 1419 (2010).
125. Chatrchyan, S. et al. Description and performance of track and primary-vertex reconstruction with the CMS tracker. *J. Instrum.* **9**, P10009 (2014).
126. Elabd, A. et al. Graph neural networks for charged particle tracking on FPGAs. *Front. Big Data* **5**, 828666 (2022).
127. Huang, S.-Y. et al. Low latency edge classification GNN for particle trajectory tracking on FPGAs. In *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)* 294–298 (IEEE, 2023).
128. Duarte, J. et al. Fast inference of deep neural networks in FPGAs for particle physics. *J. Instrum.* **13**, P07027 (2018).
129. FastML Team. fastmachinelearning/hls4ml. *Github* https://github.com/fastmachinelearning/hls4ml (2023).
130. Xuan, T. et al. Trigger detection for the sPHENIX experiment via bipartite graph networks with set transformer. In *Machine Learning and Knowledge Discovery in Databases* 51–67 (Springer, 2023).
131. Moreno, E. A. et al. JEDI-net: a jet identification algorithm based on interaction networks. *Eur. Phys. J. C* **80**, 58 (2020).
132. Mikuni, V., Nachman, B. & Shih, D. Online-compatible unsupervised non-resonant anomaly detection. *Phys. Rev. D* **105**, 055006 (2022).
133. Que, Z. et al. LL-GNN: Low latency graph neural networks on FPGAs for high energy physics. In *ACM Transactions on Embedded Computing Systems* 1–28 (ACM, 2024). **This extensive review discusses the integration of powerful machine learning methods into a real-time experimental data processing loop to accelerate the scientific discovery**.
134. Duarte, J. et al. FPGA-accelerated machine learning inference as a service for particle physics computing. *Comput. Softw. Big Sci.* **3**, 13 (2019).
135. Krupa, J. et al. GPU coprocessors as a service for deep learning inference in high energy physics. *Mach. Learn. Sci. Technol.* **2**, 035005 (2021).
136. Bogatskiy, A. et al. Lorentz group equivariant neural network for particle physics. In *Proc. 37th International Conference on Machine Learning* 992–1002 (ICML, 2020).
137. Gong, S. et al. An efficient Lorentz equivariant graph neural network for jet tagging. *J. High Energy Phys.* **7**, 030 (2022).
138. Tsan, S. et al. Particle graph autoencoders and differentiable, learned energy mover's distance. In *Advances in Neural Information Processing Systems* (NIPS, 2021).
139. Atkinson, O., Bhardwaj, A., Englert, C., Ngairangbam, V. S. & Spannowsky, M. Anomaly detection with convolutional graph neural networks. *J. High Energy Phys.* **8**, 080 (2021).
140. Hao, Z., Kansal, R., Duarte, J. & Chernyavskaya, N. Lorentz group equivariant autoencoders. *Eur. Phys. J. C* **83**, 485 (2023).
141. Govorkova, E. et al. Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider. *Nat. Mach. Intell.* **4**, 154–161 (2022).
142. Gong, W. & Yan, Q. Graph-based deep learning frameworks for molecules and solid-state materials. *Comput. Mater. Sci.* **195**, 110332 (2021).
143. Bapst, V. et al. Unveiling the predictive power of static structure in glassy systems. *Nat. Phys.* **16**, 448–454 (2020). **This fundamental work demonstrates the potential of FPGA-implemented deep learning models for achieving ultra-high inference efficiency in particle physics**.
144. Chen, C., Zuo, Y., Ye, W., Li, X. & Ong, S. P. Learning properties of ordered and disordered materials from multi-fidelity data. *Nat. Comput. Sci.* **1**, 46–53 (2021).
145. Jang, J., Gu, G. H., Noh, J., Kim, J. & Jung, Y. Structure-based synthesizability prediction of crystals using partially supervised learning. *J. Am. Chem. Soc.* **142**, 18836–18843 (2020).
146. Chen, C., Ye, W., Zuo, Y., Zheng, C. & Ong, S. P. Graph networks as a universal machine learning framework for molecules and crystals. *Chem. Mater.* **31**, 3564–3572 (2019).
147. Gasteiger, J., Groß, J. & Günnemann, S. Directional message passing for molecular graphs. In *International Conference on Learning Representations* (ICLR, 2020).
148. Choudhary, K. & DeCost, B. Atomistic line graph neural network for improved materials property predictions. *npj Comput. Mater.* **7**, 185 (2021).
149. Chen, C. & Ong, S. P. A universal graph deep learning interatomic potential for the periodic table. *Nat. Comput. Sci.* **2**, 718–728 (2022).
150. Batzner, S. et al. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat. Commun.* **13**, 2453 (2022).
151. Schütt, K., Unke, O. & Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *Proc. 38th International Conference on Machine Learning* 9377–9388 (ICML, 2021).
152. Thölke, P. & De Fabritiis, G. TorchMD-NET: Equivariant transformers for neural network based molecular potentials. In *International Conferenc on Learning Representations* (ICLR, 2022).
153. Liao, Y.-L. & Smidt, T. Equiformer: Equivariant graph attention transformer for 3D atomistic graphs. In *International Conference on Learning Representations* (ICLR, 2023).
154. Unke, O. T. et al. Machine learning force fields. *Chem. Rev.* **121**, 10142–10186 (2021).
155. Musil, F. et al. Physics-inspired structural representations for molecules and materials. *Chem. Rev.* **121**, 9759–9815 (2021).
156. Choudhary, K. et al. Unified graph neural network force-field for the periodic table: solid state applications. *Digit. Discov.* **2**, 346–355 (2023).
157. Zunger, A. Inverse design in search of materials with target functionalities. *Nat. Rev. Chem.* **2**, 0121 (2018).
158. Gebauer, N., Gastegger, M. & Schütt, K. Symmetry adapted generation of 3D point sets for the targeted discovery of molecules. In *Proc. 33rd International Conference on Neural Information Processing Systems* 7566–7578 (Curran Associates Inc., 2019).
159. Xie, T., Fu, X., Ganea, O.-E., Barzilay, R. & Jaakkola, T. Crystal diffusion variational autoencoder for periodic material generation. In *International Conference on Learning Representations* (ICLR, 2022).
160. Lyngby, P. & Thygesen, K. S. Data-driven discovery of 2D materials by deep generative models. *npj Comput. Mater.* **8**, 232 (2022).
161. Wines, D., Xie, T. & Choudhary, K. Inverse design of next-generation superconductors using data-driven deep generative models. *J. Phys. Chem. Lett.* **14**, 6630–6638 (2023).
162. Chanussot, L. et al. Open Catalyst 2020 (OC20) dataset and community challenges. *ACS Catal.* **11**, 6059–6072 (2021).
163. Gene Ontology Consortium. The Gene Ontology resource: 20 years and still going strong. *Nucleic Acids Res.* **47**, D330–D338 (2019).
164. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
165. Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K. & Moult, J. Critical assessment of methods of protein structure prediction (CASP) — Round XIV. *Proteins: Struct. Funct. Genet.* **89**, 1607–1617 (2021).
166. Ingraham, J., Garg, V., Barzilay, R. & Jaakkola, T. Generative models for graph-based protein design. In *Proc. 33rd International Conference on Neural Information Processing Systems* 15820–15831 (Curran Associates Inc., 2019).

167. Luo, J. & Luo, Y. Contrastive learning of protein representations with graph neural networks for structural and functional annotations. *Pac. Symp. Biocomput.* **2023**, 109–120 (2023).
168. Gelman, S., Fahlberg, S. A., Heinzelman, P., Romero, P. A. & Gitter, A. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. *Proc. Natl Acad. Sci. USA* **118**, e2104878118 (2021).
169. Chen, T. et al. HotProtein: A novel framework for protein thermostability prediction and editing. In *International Conference on Learning Representations* (ICLR, 2022).
170. Gao, Z. et al. Hierarchical graph learning for protein–protein interaction. *Nat. Commun.* **14**, 1093 (2023).
171. Lu, W. et al. TANKbind: Trigonometry-aware neural networks for drug–protein binding structure prediction. In *Proc. 36th International Conference on Neural Information Processing Systems* 7236–7249 (Curran Associates Inc., 2022).
172. Gainza, P. et al. De novo design of protein interactions with learned surface fingerprints. *Nature* **617**, 176–184 (2023).
173. Ho, J., Jain, A. & Abbeel, P. Denoising diffusion probabilistic models. In *Proc. 34th International Conference on Neural Information Processing Systems* 6840–6851 (Curran Associates Inc., 2020).
174. Watson, J. L. et al. De novo design of protein structure and function with rfdiffusion. *Nature* 1–3 (2023).
175. Stärk, H., Ganea, O., Pattanaik, L., Barzilay, R. & Jaakkola, T. EquiBind: Geometric deep learning for drug binding structure prediction. In *Proc. 39th International Conference on Machine Learning* 20503–20521 (ICML, 2022).
176. Qian, W. W. et al. Metabolic activity organizes olfactory representations. *eLife* 12 (2023).
177. Morselli Gysi, D. et al. Network medicine framework for identifying drug-repurposing opportunities for COVID-19. *Proc. Natl Acad. Sci. USA* **118**, e2025581118 (2021).
178. Li, S. et al. MONN: a multi-objective neural network for predicting compound–protein interactions and affinities. *Cell Syst* **10**, 308–322 (2020).
179. Zitnik, M., Agrawal, M. & Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* **34**, i457–i466 (2018).
180. Satorras, V. G., Hoogeboom, E. & Welling, M. E(*n*) equivariant graph neural networks. In *Proc. 38th International Conference on Machine Learning* 9323–9332 (ICML, 2021).
181. Townshend, R. J. et al. ATOM3D: Tasks on molecules in three dimensions. In *35th Conference on Neural Information Processing Systems* (NIPS, 2021).
182. Hoogeboom, E., Satorras, V. G., Vignac, C. & Welling, M. Equivariant diffusion for molecule generation in 3D. In *Proc. 39th International Conference on Machine Learning* 8867–8887 (ICML, 2022).
183. Guan, J. et al. DecompDiff: Diffusion models with decomposed priors for structure-based drug design. In *Proc. 40th International Conference on Machine Learning* 11827–11846 (ICML, 2023).
This article discusses the application of graph learning models to biology, presenting unprecedentedly high accuracy in predicting protein structures.
184. Luo, S., Guan, J., Ma, J. & Peng, J. A 3D generative model for structure-based drug design. In *Proc. 35th International Conference on Neural Information Processing Systems* 6229–6239 (Curran Associates Inc., 2021).
185. Liu, M., Luo, Y., Uchino, K., Maruhashi, K. & Ji, S. Generating 3D molecules for target protein binding. In *Proc. 39th International Conference on Machine Learning* 13912–13924 (ICML, 2022).
186. Peng, X. et al. Pocket2Mol: Efficient molecular sampling based on 3D protein pockets. In *Proc. 39th International Conference on Machine Learning* 17644–17655 (ICML, 2022).
187. Guan, J. et al. 3D equivariant diffusion for target-aware molecule generation and affinity prediction. In *International Conference on Learning Representations* (ICLR, 2023).
188. Wang, J. et al. SCGNN is a novel graph neural network framework for single-cell RNA-seq analyses. *Nat. Commun.* **12**, 1882 (2021).
189. Li, H. et al. Inferring transcription factor regulatory networks from single-cell ATAC-seq data based on graph neural networks. *Nat. Mach. Intell.* **4**, 389–400 (2022).
190. Cheng, F. et al. Network-based approach to prediction and population-based validation of in silico drug repurposing. *Nat. Commun.* **9**, 2691 (2018).
191. Cheng, F., Kovács, I. A. & Barabási, A.-L. Network-based prediction of drug combinations. *Nat. Commun.* **10**, 1197 (2019).
192. Jin, W. et al. Deep learning identifies synergistic drug combinations for treating COVID-19. *Proc. Natl Acad. Sci. USA* **118**, e2105070118 (2021).
193. Ge, Y. et al. An integrative drug repositioning framework discovered a potential therapeutic agent targeting COVID19. *Signal. Transduct. Target. Ther.* **6**, 165 (2021).
194. Zhou, Y. et al. CGC-Net: cell graph convolutional network for grading of colorectal cancer histology images. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* 388–398 (IEEE, 2019).
195. Wu, Z. et al. Graph deep learning for the characterization of tumour microenvironments from spatial protein profiles in tissue specimens. *Nat. Biomed. Eng.* **6**, 1435–1448 (2022).
196. Liu, Z., Li, X., Peng, H., He, L. & Philip, S. Y. Heterogeneous similarity graph neural network on electronic health records. In *2020 IEEE International Conference on Big Data* 1196–1205 (IEEE, 2020).
197. Choi, E. et al. Learning the graphical structure of electronic health records with graph convolutional transformer. In *Proc. 34th AAAI Conference on Artificial Intelligence* 606–613 (AAAI, 2020).
198. Fey, M. & Lenssen, J. E. Fast graph representation learning with pytorch geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds* (ICLR, 2019).
199. Wang, M. et al. Deep graph library: a graph-centric, highly-performant package for graph neural networks. Preprint at *arXiv* https://doi.org/10.48550/arXiv.1909.01315 (2019).
200. Sarkar, R., Abi-Karam, S., He, Y., Sathidevi, L. & Hao, C. FlowGNN: A dataflow architecture for real-time workload-agnostic graph neural network inference. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* 1099–1112 (IEEE, 2023).
201. Huang, G. et al. Machine learning for electronic design automation: a survey. *ACM Trans. Des. Autom. Electron. Syst.* **26**, 1–46 (2021).
202. He, Z., Wang, Z., Bail, C., Yang, H. & Yu, B. Graph learning-based arithmetic block identification. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)* 1–8 (IEEE, 2021).
203. He, S. et al. An overview on the application of graph neural networks in wireless networks. *IEEE Open. J. Commun. Soc.* **2**, 2547–2565 (2021).
204. Zitnik, M., Sosič, R. & Leskovec, J. Prioritizing network communities. *Nat. Commun.* **9**, 2544 (2018).
205. Hu, W. et al. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations* (ICLR, 2020).
206. Tishby, N., Pereira, F. C. & Bialek, W. The information bottleneck method. In *Proc. 37th Annual Allerton Conference on Communication, Control and Computing* 368–377 (1999).
207. Miao, S., Liu, M. & Li, P. Interpretable and generalizable graph learning via stochastic attention mechanism. In *Proc. 39th International Conference on Machine Learning* 15524–15543 (ICML, 2022).
208. Iiyama, Y. et al. Distance-weighted graph neural networks on FPGAs for real-time particle reconstruction in high energy physics. *Front. Big Data* **3**, 598927 (2021).
209. Wu, H. & Wang, H. Decoding latency of LDPC codes in 5G NR. In *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)* 1–5 (IEEE, 2019).
210. Wang, Z. et al. GNN-PIM: A processing-in-memory architecture for graph neural networks. In *Conference on Advanced Computer Architecture* 73–86 (Springer, 2020).
211. Huang, Y. et al. Accelerating graph convolutional networks using crossbar-based processing-in-memory architectures. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* 1029–1042 (IEEE, 2022).
212. Liang, S. et al. EnGN: a high-throughput and energy efficient accelerator for large graph neural networks. *IEEE Trans. Comput.* **70**, 1511–1525 (2020).
213. Choi, E., Bahadori, M. T., Song, L., Stewart, W. F. & Sun, J. GRAM: Graph-based attention model for healthcare representation learning. In *Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 787–795 (ACM, 2017).
214. Sajadmanesh, S., Shamsabadi, A. S., Bellet, A. & Gatica Perez, D. GAP: Differentially private graph neural networks with aggregation perturbation. In *Proc. 32nd USENIX Conference on Security Symposium* 3223–3240 (USENIX Association, 2023).
215. Chien, E. et al. Differentially private decoupled graph convolutions for multigranular topology protection. In *Proc. 37th International Conference on Neural Information Processing Systems* 45381–45401 (Curran Associates Inc., 2023).
216. Cao, Y. & Yang, J. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy* 463–480 (IEEE, 2015).
217. Chien, E., Wang, H. P., Chen, Z. & Li, P. Langevin unlearning. In *Privacy Regulation and Protection in Machine Learning Workshop* (ICLR, 2024).
218. Chien, E., Pan, C. & Milenkovic, O. Efficient model updates for approximate unlearning of graph-structured data. In *International Conference on Learning Representations* (ICLR, 2023).
219. Mironov, I. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)* 263–275 (IEEE, 2017).

Author contributions

E.C., M.L., A.A., K.D., S.J., S.M., Z.Z., J.D., V.F., Y.L., D.P., S.S. and P.L. researched data for the article. All authors contributed to the discussion of the content. E.C., M.L., A.A., K.D., S.J., S.M., Z.Z., C.H., O.M. and P.L. wrote the article. E.C., M.L., A.A., K.D., S.J., S.M., Z.Z., J.D., V.F., Y.L., D.P., S.S. and P.L. reviewed and edited the article.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44287-024-00076-z.

Peer review information *Nature Reviews Electrical Engineering* thanks the anonymous reviewers for their contribution to the peer review of this work.