

# Graph Neural Network-Based Power, Performance, and Area Prediction for Large-Scale Integrated Circuit Design Using GPU-Accelerated Learning

Emin Butte

Department of Computer Science, University of New Hampshire, Durham, NH, USA.  
emin.butte726@unh.edu

Maartje Heribert van Laon

Department of Computer Science and Engineering, University of Nevada, Reno, Reno, NV, USA.  
maartjeheribertvanlaon@unr.edu

Urbano Alamo

School of Computing, Clemson University, Clemson, SC, USA.  
alamourbano@clemson.edu

## Abstract

The increasing scale and heterogeneity of modern integrated circuits have made early prediction of power, performance, and area a central challenge in electronic design automation. Conventional analytical and simulation-based methods often struggle to balance accuracy, runtime, and scalability when applied to billion-transistor systems, especially during early architectural exploration and physical design. This paper examines graph neural network-based prediction of power, performance, and area for large-scale integrated circuit design using GPU-accelerated learning. It argues that circuit graphs provide a natural abstraction for capturing structural dependencies among cells, nets, macros, timing paths, and physical regions, while GPU acceleration enables the training and inference of large heterogeneous graph models at practical speed. The discussion focuses on system-level architecture, deployment trade-offs, data governance, robustness, fairness, and sustainability. It considers how heterogeneous graph learning can integrate logical, physical, and electrical features without reducing design intent to isolated local metrics. The paper further analyzes infrastructure requirements for distributed training, memory hierarchy constraints, graph partitioning, and workload scheduling. It also discusses the socio-technical implications of relying on learned predictors for critical design decisions, including explainability, accountability, reproducibility, and access equity across design teams and regions. Through cross-domain comparisons with molecular graph learning, combinatorial optimization, and large-scale language model training, the paper identifies structural lessons for scalable circuit intelligence. It concludes that graph neural networks with GPU-accelerated learning can become a foundational layer for predictive integrated circuit design, but only if technical performance is matched by governance mechanisms, robust validation, and sustainable compute practices.

## Keywords

graph neural networks; integrated circuit design; power performance and area; GPU acceleration; electronic design automation; heterogeneous graphs; socio-technical infrastructure; design governance.

## 1. Introduction

Large-scale integrated circuit design has become a problem of structural complexity rather than merely transistor count. Modern systems-on-chip combine digital logic, analog blocks, memory arrays, accelerators, interconnect fabrics, and thermal constraints within a single design envelope. Predicting power, performance, and area early in the design cycle is therefore not a narrow estimation task but a system-level negotiation among architecture, packaging, manufacturing, software workload, and verification. Traditional analytical models remain valuable, yet they often depend on assumptions that degrade as designs become more heterogeneous and as physical effects become more coupled. Machine learning has emerged as a complementary strategy, particularly where repeated design exploration creates abundant historical data and where simulation is too expensive for rapid iteration [1]. Within this landscape, graph neural networks are especially promising because circuits are inherently relational systems.

The central premise of this paper is that graph neural network-based prediction can support power, performance, and area estimation for large-scale integrated circuit design when it is coupled with GPU-accelerated learning infrastructure. A circuit graph can represent cells, nets, pins, macros, clock domains, power domains, and physical regions as nodes and edges, allowing message passing to capture local and global dependencies. Such a representation is not merely a technical convenience. It reflects the relational nature of timing, congestion, switching activity, thermal coupling, and voltage drop. Unlike image-based or sequence-based models, graph learning can preserve connectivity semantics while allowing heterogeneous node and edge types to carry different electrical and physical meanings [2]. GPU acceleration further changes the feasibility boundary by enabling training on large graph batches, mixed-precision computation, and distributed model parallelism.

However, the movement from proof-of-concept prediction to production-grade design infrastructure raises questions that exceed model accuracy. Design teams must ask who owns the training data, how models are validated across technology nodes, how uncertainty is communicated to engineers, and how compute-intensive learning affects organizational and environmental sustainability. The paper therefore treats graph neural network-based prediction as a socio-technical system. It examines architecture, deployment, governance, robustness, fairness, and policy implications alongside algorithmic performance. The discussion is organized around the idea that prediction models for power, performance, and area are not neutral calculators. They encode design priorities, historical biases, and infrastructure asymmetries. Their value depends not only on predictive error but also on whether they can be audited, reproduced, and integrated into human decision-making [7].

## 2. Background and Related Work

Graph neural networks originated as a way to learn representations over relational data through iterative message passing and state aggregation [3]. Early graph neural network models demonstrated that structural dependencies can be learned without flattening graphs into fixed-size vectors. Subsequent work introduced convolutional and attention-based variants that improved scalability and expressiveness. Graph convolutional networks provided a localized spectral-inspired approach that aggregates information from neighboring nodes [4],

while graph attention networks allowed models to weight neighbor importance dynamically [5]. Inductive representation learning further enabled generalization to unseen nodes and subgraphs, which is essential when a trained circuit model must be applied to new blocks or design revisions [6]. Survey work has since consolidated these advances and clarified trade-offs among depth, receptive field, oversmoothing, and computational cost [2]. Deep learning on graphs has also expanded into heterogeneous and dynamic settings that are directly relevant to integrated circuit design [7].

Electronic design automation has adopted machine learning for tasks such as placement, routing, timing prediction, congestion estimation, and design rule checking. A broad survey of machine learning in electronic design automation shows that the field has moved from isolated regression models toward integrated predictive flows that inform optimization and decision-making [8]. Placement has received particular attention because it combines combinatorial search with physical constraints. A graph placement methodology demonstrated that reinforcement learning and graph representation can produce floorplans and placements that generalize across blocks, although the approach also highlighted the importance of reward design and validation [9]. Open-source infrastructure such as OpenROAD has made such research more reproducible and has enabled the community to examine how learned predictors interact with traditional physical design engines [10]. Routing congestion prediction has likewise benefited from learned models that estimate localized pressure before detailed routing, reducing expensive iterations [11]. Convolutional and graph-based routability predictors have shown that spatial and topological features can be combined to anticipate congestion hotspots [12].

The required GPU-oriented acceleration study in this paper's reference set examines heterogeneous circuit graph neural network training on GPUs [13]. Its relevance lies in the recognition that circuit graphs are not generic homogeneous graphs. They contain multiple node types, edge types, feature distributions, and irregular memory access patterns. Efficient training therefore requires careful treatment of graph sampling, feature gathering, kernel fusion, and communication. This aligns with broader work on message passing for structured scientific domains, where graph models must respect domain-specific semantics [14]. Introductory and survey treatments of graph neural networks also emphasize that model design should follow the structure of the problem rather than forcing all data into uniform tensors [15]. The present paper builds on these foundations by positioning GPU-accelerated heterogeneous graph learning as an infrastructural capability for power, performance, and area prediction.

### **3. Graph Representation of Large-Scale Integrated Circuits**

A large-scale integrated circuit is naturally represented as a heterogeneous graph. Standard cells, macros, memory instances, input-output pads, and power switches can be modeled as nodes, while nets, timing arcs, physical adjacencies, and power delivery connections can be modeled as edges. Each node may carry features such as drive strength, capacitance, leakage, area, voltage threshold, and physical coordinates. Each edge may carry resistance, capacitance, delay, fanout, or coupling information. This representation supports message passing across multiple abstraction levels, allowing a model to learn how local changes propagate to global power, performance, and area outcomes. The graph is not a static artifact. It evolves through synthesis, placement, clock tree synthesis, routing, and signoff, so a predictive system must either condition on the design stage or maintain stage-specific models.

The value of graph representation lies in its ability to preserve relational context that tabular models lose. A cell's timing behavior depends not only on its own attributes but also on its fanin, fanout, load, and neighboring physical congestion. A macro's power profile depends on activity patterns, supply network impedance, and thermal proximity. These relationships are difficult to encode fully in flat feature vectors. Graph neural networks can learn higher-order interactions through layers of message passing, where node embeddings accumulate information from increasingly distant neighborhoods. For power prediction, this can capture switching activity propagation and clock network effects. For performance, it can approximate critical path behavior and timing slack distribution. For area, it can estimate resource pressure and the likelihood that placement or routing will require additional cells, buffers, or spacing.

Heterogeneity is both an opportunity and a burden. Opportunity arises because different node and edge types can be embedded with distinct update functions and attention mechanisms. Burden arises because heterogeneous graphs increase memory consumption, kernel complexity, and the risk of overfitting to one design style. Circuit graphs also exhibit scale-free and small-world characteristics, with long-range connections and highly connected clock or power networks. A naive full-graph training approach may exceed GPU memory, while aggressive sampling may discard globally relevant paths. A production system therefore needs a hybrid strategy that combines local message passing with global summaries, such as critical path descriptors, power domain aggregates, and congestion maps. This hybrid design reflects a broader principle: learned prediction should augment, not erase, the structured knowledge embedded in existing design tools.

#### **4. GPU-Accelerated Learning Architecture**

GPU acceleration is essential for training large heterogeneous circuit graph models. Circuit graphs can contain millions of nodes and edges, and training may require many epochs across multiple technology nodes, design families, and operating conditions. Graphics processing units provide high throughput for dense matrix operations, but graph learning introduces irregular memory access, variable neighbor counts, and dynamic batching. An effective architecture must therefore combine hardware-aware graph sampling, feature caching, fused message passing, and efficient scatter-gather operations. GPU memory hierarchy matters because node features, edge features, gradients, and optimizer states compete for limited high-bandwidth memory. Techniques such as mixed-precision training, gradient checkpointing, and parameter sharding can extend the feasible model size, but they also introduce numerical and reproducibility concerns. The required acceleration study [13] is relevant here because it directly addresses the training acceleration of heterogeneous circuit graph neural networks on GPUs.

At the system level, GPU-accelerated learning for integrated circuit prediction should be understood as a pipeline rather than a single model. Data ingestion converts design databases into graph tensors, feature extraction normalizes electrical and physical attributes, graph partitioning divides workloads across devices, and training orchestrates forward and backward passes. Inference may occur in interactive design sessions, where latency requirements are strict, or in overnight optimization loops, where throughput dominates. These modes require different deployment strategies. Interactive inference may use model pruning, quantization, or cached embeddings, while batch training may use distributed data parallelism or model parallelism. Graph neural network libraries provide useful abstractions, but their performance depends on kernel-level optimization and on how well the graph

structure matches the hardware topology [16]. Frameworks such as PyTorch Geometric and PyTorch have simplified implementation, but production systems still require careful profiling and tuning [17]. Large-scale training techniques from language modeling, including efficient parallelism and memory optimization, offer transferable lessons even though circuit graphs differ from token sequences [18]. Memory optimization methods such as zero redundancy optimizer strategies can reduce duplication and enable larger effective batch sizes [19]. General deep learning frameworks provide the software foundation for these efforts [20].

The deployment architecture also has organizational consequences. If training requires a large GPU cluster, then access to predictive capability becomes concentrated among well-resourced design teams, foundries, and cloud providers. This concentration can affect negotiation power, tool interoperability, and the ability of smaller firms or academic groups to participate in advanced design research. A sustainable architecture should therefore include tiered model sizes, federated or privacy-preserving training options, and shared benchmarks that allow smaller organizations to benefit from pretrained representations without exposing proprietary designs. The infrastructure is not merely a computational substrate. It shapes who can design, who can validate, and who can contest automated predictions.

## **5. Prediction of Power, Performance, and Area**

Power, performance, and area prediction is a multi-objective problem with competing constraints. Reducing area can increase congestion and worsen timing. Increasing performance through higher voltage or larger drivers can raise power and thermal stress. Reducing power through voltage scaling can degrade timing margins. A graph neural network can model these trade-offs by learning joint representations across design features and label types. One approach is multi-task learning, where a shared graph encoder feeds separate prediction heads for dynamic power, leakage power, timing slack, frequency, and area. Another approach is conditional prediction, where the model receives target constraints such as clock frequency or power budget and estimates achievable area or performance risk. Both approaches require careful normalization because power, performance, and area operate on different scales and have different sensitivities to technology node, temperature, and process variation.

Performance prediction is especially dependent on path-level structure. Timing analysis depends on critical paths, clock domains, and false paths, which are not fully captured by local neighborhood aggregation. A graph model may need explicit path features, hierarchical embeddings, or attention mechanisms that can focus on long-range dependencies. Power prediction similarly depends on switching activity, which may be workload-dependent and not fully encoded in the circuit graph alone. Area prediction depends on placement and routing feasibility, which changes as the physical design progresses. A credible prediction system must therefore state its operating assumptions, report uncertainty, and indicate when a prediction should be treated as advisory rather than authoritative. Calibration is crucial, because an overconfident model can mislead engineers into premature convergence or unnecessary optimization.

The integration of prediction with optimization is another system-level concern. A predictor can be used as a surrogate objective, as a screening mechanism, or as a diagnostic aid. Surrogate optimization may accelerate design space exploration, but it risks optimizing the model rather than the physical system. Screening can reduce expensive simulations, but false negatives may eliminate promising candidates. Diagnostic use may be safest because it informs human judgment without replacing signoff. Mature deployment likely requires a

portfolio approach in which graph neural networks handle fast estimation, analytical models provide interpretable baselines, and detailed simulation validates high-stakes decisions. This layered strategy mirrors practices in other safety-critical domains, where learned components are bounded by conventional checks and formal verification.

## **6. Robustness, Fairness, and Governance**

Robustness is a first-order requirement for learned prediction in integrated circuit design. A model trained on one technology node, foundry process, or design family may fail when applied to another. Distribution shift can arise from new standard cell libraries, different clocking strategies, changes in packaging, or novel workloads. Robustness testing should include cross-node validation, adversarial perturbations of graph structure, missing feature scenarios, and sensitivity analysis under process variation. Graph neural networks are particularly vulnerable to structural perturbations because small changes in connectivity can alter message passing significantly. A single missing edge or mislabeled macro can change embeddings across a broad neighborhood. Robust training may require noise injection, edge dropout, ensemble methods, and uncertainty-aware architectures.

Fairness in this context is not primarily about protected demographic groups, but about equitable access to design capability and fair treatment of design styles, vendors, and regions. If models are trained predominantly on data from a few large foundries or high-volume product categories, they may perform poorly for low-volume, specialized, or emerging designs. This can create a feedback loop in which only mainstream designs benefit from automation, while niche designs face higher costs and longer schedules. Governance should therefore address data provenance, representation, model auditing, and dispute resolution. Design teams need to know what data trained a model, what assumptions were made, and how errors are handled. When a prediction influences tapeout decisions, accountability cannot rest solely on an opaque model. Human review, documented rationale, and traceable versioning are necessary.

Policy implications extend beyond individual firms. Governments increasingly treat semiconductors as strategic infrastructure, which means predictive design tools can become part of industrial policy, export control, and national security discussions. Open benchmarks and shared research infrastructure can reduce duplication and improve reproducibility, but they must also protect proprietary design data. Privacy-preserving learning, synthetic data generation, and secure enclaves may offer middle paths, yet each introduces trade-offs in accuracy and cost. The governance challenge is to encourage innovation while preventing lock-in, opaque decision-making, and unsustainable concentration of computational resources.

## **7. Infrastructure, Deployment, and Sustainability**

The infrastructure for GPU-accelerated graph learning includes compute clusters, high-bandwidth storage, design data management systems, model registries, and integration layers with commercial and open-source electronic design automation tools. Training requires reliable access to large graph datasets, which may be stored as design databases, netlists, parasitic extraction files, and timing reports. Converting these heterogeneous sources into consistent graph tensors is a major engineering effort. Inference requires low-latency serving, versioned models, and fallback mechanisms when predictions are unavailable or outside the validated domain. Deployment may occur on premises, in a private cloud, or in a hybrid architecture, depending on data sensitivity and cost. Each choice affects latency, security, and operational complexity.

Sustainability is an increasingly important dimension. Training large graph models consumes energy, and repeated retraining across technology nodes multiplies that cost. If prediction models are used to reduce expensive simulation and physical prototyping, they may produce net energy savings. If they are used indiscriminately or retrained without need, they may add overhead. Sustainable practice includes model reuse, transfer learning, distillation, and scheduled training during low-carbon energy periods. It also includes measuring the embodied carbon of hardware and the operational carbon of data centers. A design organization should evaluate the carbon return on compute investment, not merely the speedup of a single training run.

Cross-domain comparisons are instructive. Molecular graph learning has shown that graph neural networks can learn chemical properties from structure, but it also revealed the importance of inductive biases and data quality [14], [15]. Combinatorial optimization research has demonstrated that learned models can guide search but often require hybrid integration with exact methods. Large-scale language model training has pushed distributed GPU infrastructure to maturity, offering lessons in checkpointing, communication overlap, and fault tolerance [18], [19]. Integrated circuit prediction differs because its graphs are engineered, hierarchical, and constrained by physics. Yet the infrastructural lessons are similar: scalability depends on data pipelines, hardware utilization, and organizational discipline as much as on model architecture.

## **8. Case Illustrations and Cross-Domain Comparisons**

A representative case is early-stage placement and routing exploration. A design team may have a synthesized netlist and rough floorplan but no final physical implementation. A graph neural network can predict congestion, timing risk, and power hotspots before detailed routing. If the prediction identifies a region likely to exceed routing capacity, engineers can adjust floorplanning, macro placement, or pin assignment earlier, when changes are cheaper. The required acceleration study [13] supports this scenario by addressing how heterogeneous circuit graphs can be trained efficiently on GPUs. However, the case also shows why governance matters. If the predictor is trained on previous designs that used different routing rules, its recommendations may be biased toward obsolete practices. The team must therefore validate against current technology and route the final design through conventional signoff.

Another case is multi-die and chiplet integration. Chiplets introduce new graph structures because connectivity spans package substrates, interposers, and separate dies. Power, performance, and area prediction must account for inter-die latency, thermal coupling, and yield trade-offs. A graph model can represent dies, bumps, traces, and power delivery networks as heterogeneous nodes and edges. This extends the circuit graph into a package-level graph, increasing scale and heterogeneity. GPU acceleration becomes even more important because package-level graphs may combine detailed intra-die structures with coarse inter-die links. The model must handle multiple resolutions without losing critical bottlenecks.

Cross-domain comparison with social network analysis highlights both promise and risk. Social graphs exhibit community structure and influence propagation, which resemble power and timing propagation in circuits. Yet social prediction often tolerates uncertainty, while circuit signoff does not. Cross-domain comparison with supply chain networks is also relevant because both involve dependency graphs, bottlenecks, and cascading failures. A learned predictor for chip design can borrow ideas from supply chain resilience, such as stress testing, scenario analysis, and redundancy metrics. These comparisons are not mere analogies.

They suggest that graph neural networks should be evaluated not only by average error but also by behavior under rare, high-impact events.

## **9. Challenges and Future Directions**

Several challenges remain. First, data scarcity and heterogeneity limit generalization. Proprietary designs are not easily shareable, and even within a firm, design databases may be inconsistently labeled. Federated learning, transfer learning, and synthetic graph generation are promising but immature. Second, model explainability is difficult for deep graph networks. Engineers need to know why a prediction was made, especially when it contradicts intuition. Attention visualization, counterfactual analysis, and surrogate rule extraction can help, but they do not fully solve the problem. Third, evaluation must move beyond mean absolute error toward decision-relevant metrics. A model that predicts average power well but misses worst-case timing is dangerous. Metrics should include false negative rates for critical violations, calibration under distribution shift, and robustness to graph perturbations.

Fourth, integration with existing design flows is a major barrier. Electronic design automation tools are complex, proprietary, and highly optimized. Adding a learned predictor requires stable interfaces, version control, and cross-team coordination. Fifth, sustainability and cost require ongoing attention. Larger models are not always better, and smaller task-specific models may be more practical for repeated inference. Sixth, policy and standardization are needed for data sharing, model auditing, and liability. If a learned predictor contributes to a failed tapeout, responsibility must be distributed among model developers, data providers, tool vendors, and design engineers. Clear standards can reduce uncertainty and build trust.

Future research should explore hierarchical graph transformers that combine local message passing with global attention over critical paths and power domains. It should investigate uncertainty quantification that is calibrated to signoff risk. It should develop GPU-aware graph partitioning and communication schemes that scale to package-level and multi-die graphs. It should also examine human-in-the-loop workflows where engineers can interrogate, override, and retrain models. The most valuable systems will not be those that replace designers but those that make complex trade-offs more visible, testable, and governable.

## **10. Conclusion**

Graph neural network-based prediction of power, performance, and area for large-scale integrated circuit design is a promising response to the growing complexity of semiconductor systems. Circuit graphs preserve connectivity, hierarchy, heterogeneity, and physical context, while GPU-accelerated learning makes large-scale training and inference practical. The technical opportunity is substantial, but the system-level challenge is equally important. Prediction models must be robust across technology nodes and design styles, integrated with conventional verification, governed transparently, and deployed in ways that are equitable and sustainable. They must handle multi-objective trade-offs without hiding uncertainty or shifting accountability away from human engineers. Cross-domain lessons from molecular graph learning, combinatorial optimization, and large-scale GPU training suggest that scalability depends on both algorithmic innovation and infrastructural discipline. If these conditions are met, graph neural networks can become a foundational layer for predictive integrated circuit design, not as autonomous authorities but as accountable instruments that help design teams navigate complexity with greater foresight.

## **References**

1. Kahng, A. B. (2018). Machine learning applications in physical design: Recent results and directions. In *Proceedings of the 2018 International Symposium on Physical Design* (pp. 68–73). Association for Computing Machinery.
2. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24.
3. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1), 61–80.
4. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.
5. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*.
6. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 1024–1034.
7. Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1), 249–270.
8. Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., Wu, J., Xu, Y., Zhang, H., Zhong, K., & others. (2021). Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems*, 26(5), 1–46.
9. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Tuncer, E., Vahid, A., Yang, J., & Dean, J. (2021). A graph placement methodology for fast chip design. *Nature*, 594(7862), 207–212.
10. Chan, T. B., Gupta, P., Kahng, A. B., & Menon, S. (2020). OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain. In *Proceedings of the Government Microcircuit Applications and Critical Technology Conference*.
11. Alawieh, M. B., Li, W., Lin, Y., Singhal, N., Iyer, M., & Pan, D. Z. (2020). High-definition routing congestion prediction for large-scale FPGAs. In *Proceedings of the 25th Asia and South Pacific Design Automation Conference* (pp. 26–31). IEEE.
12. Xie, Z., Huang, R., Fang, X., Xu, Y., Wang, Y., & others. (2018). RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design* (pp. 1–8). IEEE.
13. Luo, Yuebo, et al. "Dr-circuitgnn: Training acceleration of heterogeneous circuit graph neural network on gpus." *Proceedings of the 39th ACM International Conference on Supercomputing*. 2025.
14. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 1263–1272). PMLR.

15. Sanchez-Lengeling, B., Reif, E., Platt, B., & Wiltchko, A. B. (2021). A gentle introduction to graph neural networks. *Distill*. <https://doi.org/10.23915/distill.00033>
16. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., Huang, Z., Guo, Q., Zhang, H., Lin, H., Zhao, J., Li, J., Smola, A. J., & Zhang, Z. (2019). Deep graph library: A graph-centric, highly performant package for graph neural networks. *arXiv*. <https://arxiv.org/abs/1909.01315>
17. Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.
18. Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V. A., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A., & Zaharia, M. (2021). Efficient large-scale language model training on GPU clusters using Megatron-LM. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1–15). ACM.
19. Rajbhandari, S., Rasley, J., Ruwase, O., & He, Y. (2020). ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1–16). IEEE.
20. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8024–8035.