Proficient Graph Neural Network Design by Accumulating Knowledge on Large Language Models

Jialiang Wang
jwangic@connect.ust.hk
Hong Kong University of Science and Technology
Hong Kong SAR, China

Hanmo Liu[*]
hliubm@connect.ust.hk
Hong Kong University of Science and Technology
Hong Kong SAR, China

Shimin Di
sdiaa@connect.ust.hk
Southeast University
Nanjing, China

Zhili Wang
zwangeo@connect.ust.hk
HoHuawei Hong Kong Research Center (HKRC)
Hong Kong SAR, China

Jiachuan Wang[†]
wangjc@slis.tsukuba.ac.jp
University of Tsukuba
Tsukuba, Japan

Lei Chen[‡]
leichen@cse.ust.hk
Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Xiaofang Zhou
zxf@cse.ust.hk
Hong Kong University of Science and Technology
Hong Kong SAR, China

Abstract

High-level automation is increasingly critical in AI, driven by rapid advances in large language models (LLMs) and AI agents. However, LLMs, despite their general reasoning power, struggle significantly in specialized, data-sensitive tasks such as designing Graph Neural Networks (GNNs). This difficulty arises from (1) the *inherent* knowledge gaps in modeling the intricate, varying relationships between graph properties and suitable architectures and (2) the *external* noise from misleading descriptive inputs, often resulting in generic or even misleading model suggestions. Achieving *proficiency* in designing data-aware models—defined as the meta-level capability to systematically accumulate, interpret, and apply data-specific design knowledge—remains challenging for existing automated approaches, due to their inefficient construction and application of meta-knowledge. To achieve meta-level *proficiency*, we propose DesiGNN, a knowledge-centered framework that systematically converts past model design experience into structured, fine-grained knowledge priors well-suited for meta-learning with LLMs. To account for the *inherent* variability and *external* noise, DesiGNN aligns empirical property filtering from extensive benchmarks with adaptive elicitation of literature insights via LLMs. By constructing a solid meta-knowledge between unseen graph understanding and known effective architecture patterns, DesiGNN can deliver top-5.77% initial model proposals for unseen datasets within seconds and achieve consistently superior performance with minimal search cost compared to baselines.

CCS Concepts

• **Computing methodologies → Search methodologies**; **Neural networks**; **Knowledge representation and reasoning**.

Keywords

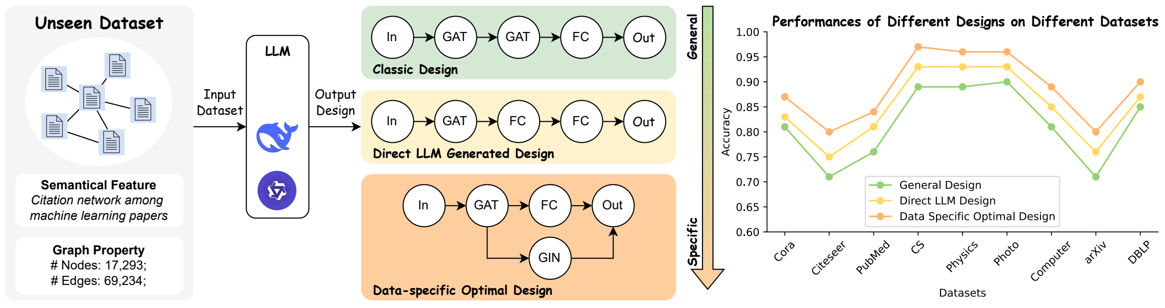
Neural Architecture Search; Graph Neural Networks; Large Language Models

ACM Reference Format:
Jialiang Wang, Hanmo Liu, Shimin Di, Zhili Wang, Jiachuan Wang, Lei Chen, and Xiaofang Zhou. 2026. Proficient Graph Neural Network Design by Accumulating Knowledge on Large Language Models. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining (WSDM '26), February 22–26, 2026, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3773966.3777982

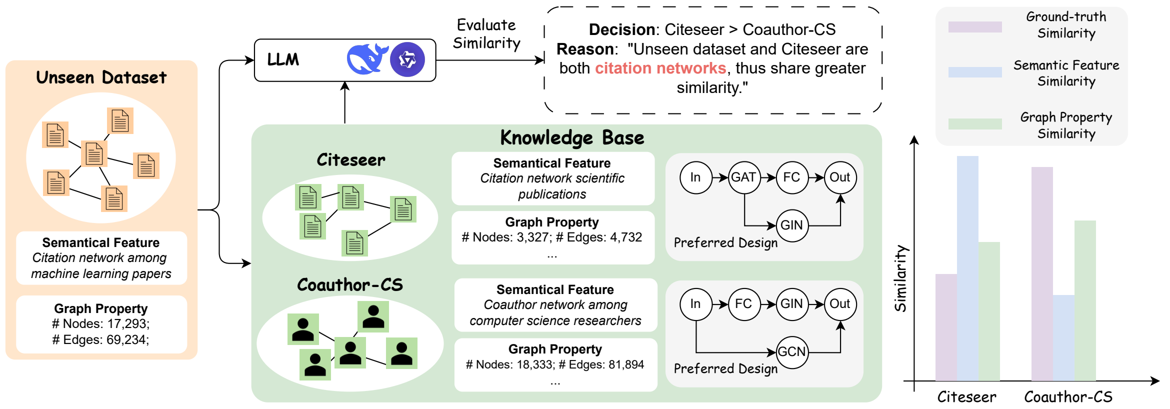
1 Introduction

Automation has become a defining trend in artificial intelligence, reshaping how complex tasks are executed at scale without human intervention or the requirement of deep expertise. From code generation [7, 9, 68] and agentic workflow orchestration [33, 37, 58, 65, 73] to the multi-task generalization of foundation models [32], this paradigm shift is transforming how expertise is embedded and operationalized in AI systems. Among various frontiers, the automated design of machine learning models stands as one of the most fundamental and impactful challenges in this automation landscape. In the past, automated machine learning (AutoML) [49, 52, 54, 71] has over-emphasized model search specific to independent data and tasks from scratch, which does not conform to the cross-domain and cross-task automation trends. The recent breakthroughs in large language models (LLMs) [4, 11, 46, 63] and AI agents [27, 62, 72] with

[*]Also with Hong Kong University of Science and Technology (Guangzhou).
[†]Corresponding author.
[‡]Also with Hong Kong University of Science and Technology.



(a) Inherent Knowledge Gap: LLMs propose overly generic GNNs when provided with purely semantic and property descriptions, unable to induce data-specific configurations.


Unseen Dataset
Semantical Feature
Citation network among machine learning papers
Graph Property
# Nodes: 17,293;
# Edges: 69,234;
LLM
Evaluate Similarity
Decision: Citeseer > Coauthor-CS
Reason: "Unseen dataset and Citeseer are both citation networks, thus share greater similarity."
Knowledge Base
Citeseer
Semantical Feature
Citation network scientific publications
Graph Property
# Nodes: 3,327; # Edges: 4,732
...
Preferred Design
Coauthor-CS
Semantical Feature
Coauthor network among computer science researchers
Graph Property
# Nodes: 18,333; # Edges: 81,894
...
Preferred Design
Ground-truth Similarity
Semantic Feature Similarity
Graph Property Similarity
Similarity
Citeseer
Coauthor-CS

(b) External Noise: LLMs misinterpret task similarity by weighing more on the citation networks category, causing "artificial hallucinations" and ineffective knowledge transfers.

Figure 1: Case studies illustrating *inherent* and *external* limitations of existing LLM-based methods.

strong academic backgrounds and contextual reasoning have further fueled expectations—*could they autonomously generate tailored models for unseen tasks, thereby democratizing machine learning?*

Inspired by LLMs' general reasoning and generation capabilities, recent studies [45] have leveraged LLMs to replace human involvement beyond traditional abilities of AutoML, such as task understanding [14, 48] and problem formulation [53]. Despite democratizing the complex process of model design, realizing highly data-aware automated model design, especially in data-sensitive domains like graph learning, remains elusive for LLMs. Unlike tasks in vision or language domains, where LLMs excel in proposing generic solutions that could dominate all tasks and data properties, designing Graph Neural Networks (GNNs) [10, 20, 26, 47, 57, 67] for specific graph properties and graph learning tasks requires a nuanced understanding of how data topology influences model architecture [51, 60]. For example, the homophily nature behind GCN's aggregation has catastrophic effects when propagating information over heterophily graphs, where two nodes with different labels are more likely to be linked. Unfortunately, current LLM-based methods often rely entirely on LLMs' *inherent* knowledge and reasoning, lacking fine-grained design knowledge of the relationships between graph-specific properties and effective GNN configurations, and are also sensitive to *external* input noise. As observed in Figure 1a, the LLM agents adopted in recent methods [14, 48, 53] tend to suggest architectures that are generally well-performing yet overly generic, *inherently* falling short in tailoring designs to unseen graph structures. Worse still, as shown in Figure 1b, they are vulnerable to "artificial hallucinations"—being easily misled by *external*, irrelevant user descriptions (e.g., being a citation network) or blindly enumerated data statistics that do not accurately reflect the data-aware challenges in model design, resulting in misleading knowledge association and ineffective designs.

Achieving *proficiency* in designing data-aware models thus demands far more than general LLM-based model design methods, which call for *a meta-level capability to systematically accumulate, interpret, and apply fine-grained, data-specific design knowledge*. Humans, akin to training a new Ph.D. student, face substantial hurdles in accumulating this complex knowledge, often undergoing a lengthy learning process to design tailored models *proficiently*. Meta-based AutoML methods represent initial efforts toward this goal computationally, seeking transferable insights across tasks via learned task embeddings [1, 28, 60] or surrogate predictors [55, 56]. These methods, however, often rely on simple, predefined meta-features that may fail to generalize robustly to complex data diversity, thus bending the current automation trend. Extending this concept to graphs, AutoTransfer [5] has made the first promising strides by creating task-model repositories and embedding-based knowledge transfers to improve search efficiency. Nonetheless, it remains limited by predefined task embeddings and the rigid aggregation of prior tasks into random search, insufficiently capturing the fine-grained, graph-aware meta-knowledge documented in the vast graph learning literature [51, 60].

To address these limitations, we propose DesiGNN, a knowledge-centered and proficiency-oriented framework for the automated design of data-aware GNNs for unseen tasks. DesiGNN systematically converts past model design experiences into structured, fine-grained knowledge priors well suited to meta-learning with LLMs, thereby novelly bridging the long-standing gap between unseen graph understanding and known effective architecture patterns in graph application research. To address the observed *external* noise from descriptive inputs and the *inherent* variability in graph properties, DesiGNN constructs an empirically validated meta-level understanding of graph-GNN-performance from benchmarks and adaptively aligns key properties with graph-specific insights elicited from the graph learning literature, using LLMs as knowledge aligners to enhance the constructed meta-knowledge. This hybridization of empirical knowledge and LLM-guided reasoning empowers DesiGNN to deliver immediate, data-aware model proposals without any training, efficiently refine these architectures based on accumulated insights, and ultimately deliver superior performance with minimal computational overhead. Our contributions are as follows:

- We empirically identify two key limitations in current LLM-based GNN design approaches: the *external* noise from descriptive inputs and the *inherent* gap in fine-grained design knowledge. Motivated by these insights, we propose a knowledge-centered framework that enhances meta-level *proficiency* in designing tailored, data-aware GNNs for unseen tasks by systematically accumulating, interpreting, and applying fine-grained meta-knowledge.
- To account for the *inherent* variability of graph properties on GNN performance, DesiGNN aligns empirical property filtering from extensive benchmarks with adaptive elicitation of literature insights via LLMs, building a solid meta-knowledge between graph topology and high-performing GNN architectures. This enables DesiGNN to *proficiently* deliver immediate, data-aware GNN recommendations without any training on the unseen data.
- Extensive experiments across 3 out-of-distribution datasets and 8 benchmarks demonstrate that DesiGNN delivers initial GNN proposals ranking in the top 5.77% of all possible architectures within seconds. Moreover, its knowledge-driven refinement strategy rapidly optimizes these initial suggestions, consistently outperforming automated methods in both performance and efficiency.

2 Related Work

2.1 Automated GNNs

Automated GNNs (AutoGNNs) [17, 49, 71] aim to automatically find an optimal GNN architecture θ for an unseen graph G^u using a controller π parameterized by α, where the architecture θ with network weight ω can achieve best performance $\mathcal{H}(\theta, \omega; G^u)$ on G^u. In AutoGNNs, the search spaces are categorized into intra-layer [16] and inter-layer [22] designs, with strategies including reinforcement learning [16], evolutionary algorithms [43], differentiable search [50], and search-based model fine-tuning [70]. Similar strategies exist for the knowledge graph [12, 13, 23, 44]. However, these approaches lack *proficiency* and demand substantial computational resources [36], as they optimize configurations θ without leveraging prior knowledge and require extensive sampling from $\pi_\alpha(\theta)$. Recently, meta-learning [6, 15] methods have sought to overcome these limitations by transferring insights from past tasks through learned task embeddings [1, 5] or surrogate predictors [30, 55, 56]. While these methods mark an initial step toward knowledge-driven AutoGNNs, they rely on coarse, predefined meta-features that are often incapable of capturing generalizable relationships between diverse data properties and effective model configurations.

NAS Benchmark for Designing GNNs. NAS-Bench-Graph [39] offers a standard benchmark for GNN architecture search, comprising nine datasets and 26,206 unique architectures with associated performance records. Similarly, many other benchmarks have been created by pioneers across different domains [8]. While these benchmarks could provide valuable insights into model-performance relationships, existing works underutilize their potential for knowledge-driven design because they lack a data dimension, which is the foundation of knowledge transfer. Therefore, only a few methods [5, 31, 60] attempted to design feasible models for unseen tasks using such model-performance knowledge. Notably, they rely primarily on semi-supervised or supervised techniques to fill in missing data associations, i.e., require direct training on unseen data.

2.2 LLMs for Designing GNNs and Limitations

Inspired by LLMs' ability to solve optimization tasks [59], recent studies explore their use to automate model design [19, 24, 61, 66, 68, 69], focusing on managing task formulation, architecture generation, and optimization. Auto2Graph [53] democratizes the use of AutoGNNs by translating descriptive user inputs (e.g., Cora [41] is a citation network) into configuration specifications. GPT4GNAS [48] and GHGNAS [14] iteratively refine model designs through crafted prompts, focusing on different graph geneity. Despite democratizing the usage, they suffer from: (1) Over-reliance on *external* descriptive user inputs that result in poor alignment between graph properties and architecture designs; (2) Raw LLMs *inherently* lack deep expertise in fine-grained graph-GNN-performance relationships, limiting their ability to suggest data-aware models; (3) Refining model designs solely based on feedback from testing models, lacking actionable, knowledge-driven search strategies.

3 Proficient Graph Neural Network Design

As discussed in Section 1, current LLM-based methods lack *proficiency* in designing data-aware GNNs, with *inherent* knowledge gaps and *external* noise as major barriers. To formalize these challenges, we adopt a Bayesian perspective to represent the automated GNNs design as solving the posterior belief over candidate architecture θ_i when facing an unseen graph G^u:

$$\mathbb{P}(\theta_i|G^u) = \frac{\mathbb{P}(G^u|\theta_i)\,\mathbb{P}(\theta_i)}{\mathbb{P}(G^u)}, \tag{1}$$

where $\mathbb{P}(\theta_i)$ is derived from a knowledge pool $\mathcal{K}$ accumulated from prior tasks $\{G^i\}$. However, *it remains a non-trivial problem to solve pseudo-likelihood* $\mathbb{P}(G^u|\theta_i)$ *without formally testing the model on unseen data*—a common *proficiency* bottleneck in existing AutoML that leads to exhaustive search. This *proficiency* denotes the capability to systematically accumulate, interpret, and apply fine-grained, data-aware meta-knowledge relating graph properties to effective GNN architectures. In our formulation, such capability can be conceptually represented as solving for the posterior belief $\mathbb{P}(\theta_i|G^u)$, with minimal or even no testing of candidate models $\{\theta_i\}$ on G^u.

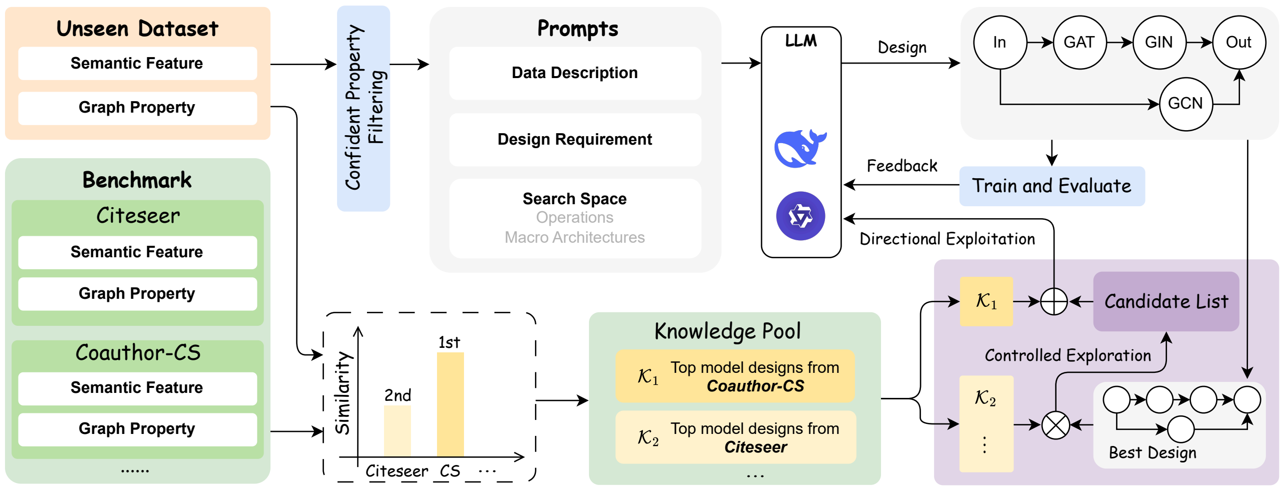


Figure 2: The illustration of our knowledge-centered DesiGNN pipeline for designing GNNs. The orange module is Graph understanding, the green module is Knowledge Retrieval, and the purple module is Model Suggestion and Refinement.

To enhance such meta-level *proficiency* without directly testing the unseen data, DesiGNN explicitly integrates empirically validated graph-property knowledge with adaptive LLM alignment to construct a trustworthy meta-level understanding $\mathcal{M}$ between graphs $\mathbf{G}$, GNNs $\mathbf{\Theta}$, and performances $\mathbf{P}$: $\mathcal{M} : (\mathbf{G}, \mathbf{\Theta}) \rightsquigarrow \mathbf{P}$ based on crucial insights from graph literature [51, 60]. Figure 2 illustrates our Bayesian-inspired pipeline: Graph Understanding, Knowledge Retrieval, and Model Suggestion and Refinement, each component systematically addressing *external* noise in descriptive data inputs and *inherent* meta-knowledge gaps, and how to strategically apply the constructed meta-knowledge in designing data-aware models.

3.1 Connect Graph Topology to GNN Design

Performance-oriented modeling of discrete graph data is essential to quickly capture data priors from massive benchmarks to construct trustworthy meta-knowledge, a key challenge for LLMs due to their inability to naturally interpret graph-structured data [18]. As discussed in Section 2.2, existing LLM-based methods rely heavily on user-provided task descriptions, which fail to capture key graph topologies (e.g., heterophily) that often motivate specific GNN designs in the vast graph learning literature [51, 60]. Our studies in Figures 1b, 3b, and Section 4.3 demonstrate that semantic or purely descriptive inputs (e.g., a citation graph) frequently mislead LLMs, leading to "artificial hallucination" and ineffective solutions. Such descriptive inputs introduce significant *external* noise, undermining the reliability of generic LLM suggestions when connecting graphs to model performance. Thus, it is necessary to establish an empirical foundation that captures fine-grained graph-GNN-performance mappings, enabling the construction of trustworthy meta-knowledge for data-aware model design.

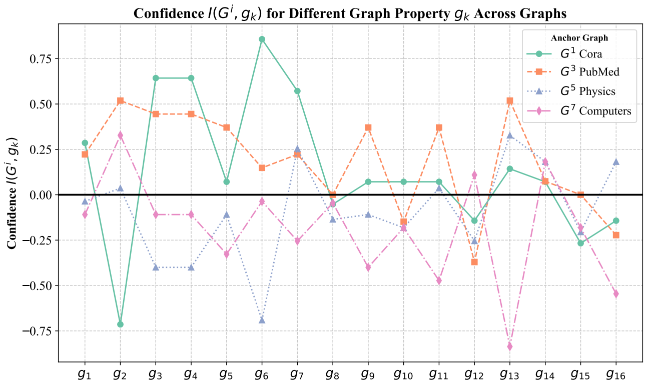
To empirically study the missing meta-level understanding about graph-GNN-performance, we start with 16 graph properties $\{g_k\}_{k=1}^{16}$ from the literature: clustering, betweenness, density, degree centrality, closeness, degree, edge count, graph diameter, shortest path, assortativity, eigenvector, dimensionality, node count, diversity, connected component, and label homophily. We infer graph similarity from two aspects: 1) the statistical distance ranking $\mathcal{SR}(i, k)$ based on graph properties g_k, and 2) the empirical performance ranking $\mathcal{ER}(i)$ of transferred top-performing GNNs from other graphs $\{G^j \mid j \neq i\}$ to anchor graph G^i. To assess how reliably specific graph properties predict successful GNN design knowledge transfers, we introduce the concept of graph property *confidence* for g_k on G^i through Kendall's τ Coefficient [25]:

$$I(G^i, g_k) := \text{KendallCorr}\big(\mathcal{SR}(i, k), \mathcal{ER}(i)\big), \tag{2}$$

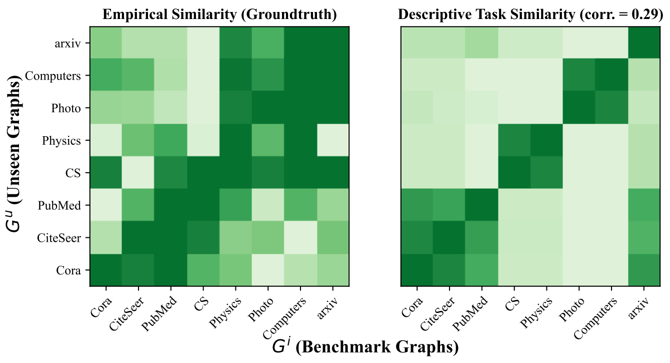
where $I(G^i, g_k)$ quantifies the correlation between the topological distance measured by g_k and the empirical performance differences in transferring different top-performing GNNs from $\{G^j \mid j \neq i\}$ to G^i. A higher $I(G^i, g_k)$ indicates that g_k is a stronger indicator of effective model design transfers, thus playing a key role in meta-learning with LLMs to design data-aware models for unseen tasks.

Variability in Property Confidence. To validate the reliability of graph properties for GNN performance, we assess 16 *confidence* scores $\{I(G^i, g_k)\}_{k=1}^{16}$ for each anchor graph G^i against the other 8 graphs in the benchmark [39]. Figure 3a shows that *confidence* $I(G^i, g_k)$ fluctuates widely across diverse graphs. e.g., while $I(\text{Physics}, g_{16}{=}\text{homophily})$ is positive, the same property exhibits negative correlations for other graphs. *This inherent variability highlights that specific graph properties are strong indicators of transferability for certain graphs but unreliable for others.* Notably, this directly exposes why generic LLM suggestions fail to capture fine-grained insights—LLMs cannot implicitly discern when particular properties matter without explicit, empirically validated priors.

Empirical Filtering as Trustworthy Foundation. Recognizing those empirical insights on *external* noise and *inherent* property variability, we first use empirical filtering to preliminarily identify graph properties that are reliably influential by averaging their predictive *confidences* $I(G^i, g_k)$ across benchmark graphs as $\bar{I}(g_k) = \frac{1}{n}\sum_{i=1}^{n} I(G^i, g_k)$. This process ensures that our property set built from the literature is also empirically grounded, reducing susceptibility to *external* noise and *inherent* property variability. Then, based on the empirical study in Figure 4, we develop a filter



(a) Property *confidence* varies across graphs.



(b) Descriptive inputs induce invalid task associations that deviate from empirical truth. DesiGNN exhibits a better correlation of 0.52.

Figure 3: Empirical studies of two barriers: *inherent* property variability and *external* noise.

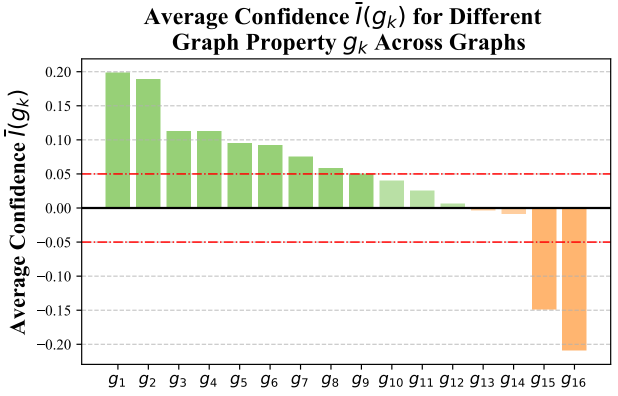


Figure 4: Average *confidence* of graph properties.

$\mathcal{F}(G^i, \bar{I})$ to correlate the discrete graph modeling with transferred model performance, selecting the Top-N_f properties with the highest $\bar{I}(g_k)$. This leads to the alternative formulation of Equation 1:

$$\mathbb{P}(\theta_i|\mathcal{F}(G^u)) = \frac{\mathbb{P}(\mathcal{F}(G^u)|\theta_i)\,\mathbb{P}(\theta_i)}{\mathbb{P}(\mathcal{F}(G^u))}. \tag{3}$$

This mechanism mitigates the typical LLM failure of "artificial hallucinations"—often triggered by unprofessional graph modeling—at negligible offline computational cost. To accumulate new knowledge, we design a self-evaluation bank *BE* that stores measured graph properties and similarity rankings for new graphs on the fly, enabling $\mathcal{F}$ to scalably evolve over time by integrating new empirical evidence. Grounded in empirical transferability, this mechanism serves as a trustworthy foundation for capturing graph-to-graph (G^i to G^j) meta-priors that underpin the correlations between graphs and high-performing GNN models, yet it still *calls for an effective property variability handler.*

3.2 Knowledge Retrieval Module

Proficiency in GNN design largely depends on capturing meta-knowledge in the form of the posterior belief for unseen tasks. However, even after empirically filtering and establishing a general correlation between graph properties and model performance, solving pseudo-likelihood $\mathbb{P}(\mathcal{F}(G^u)|\theta_i)$ remains a resource-intensive accumulation of testing various θ_i on G^u. To overcome it, we novelly propose to leverage the known likelihood $\mathbb{P}(\mathcal{F}(G^i)|\theta_i)$ on benchmark datasets to form a practical proxy for $\mathbb{P}(\mathcal{F}(G^u)|\theta_i)$ on the unseen. Eliciting such fine-grained model design knowledge paves the way for an efficient approximation of the posterior $\mathbb{P}(\theta_i|\mathcal{F}(G^u))$ with accumulated prior knowledge. Finding such a proxy for the unseen task among the massive benchmark knowledge requires an unsupervised task similarity formulation that properly assesses the likelihood of successful model design knowledge transfer from G^i to G^u, which *has to reflect the performance outcomes* $\mathcal{H}(\cdot,\cdot;G^u)$.

Definition 3.1 (Task Similarity for Model Design Knowledge Transfer). Given an unseen graph G^u and two benchmark graphs G^i and G^j, each associated with N_m top-performing models $\{\theta^*_m\}_{m=1}^{N_m} \in \Theta$, an effective task similarity $\mathcal{S}(\cdot,\cdot)$ for transferring model design knowledge satisfies:

$$\mathcal{S}(G^u, G^i) \geq \mathcal{S}(G^u, G^j) \\ \Rightarrow \;\mathbb{P}\Big[\mathcal{H}(\{\theta^*_{im}\},\cdot,G^u) \geq \mathcal{H}(\{\theta^*_{jm}\},\cdot,G^u)\Big] \geq \delta. \tag{4}$$

where $\mathcal{H}(\{\theta^*_{im}\},\cdot,G^u)$ denotes the performance of the transferred top-performing pattern $\{\theta^*_{im}\}$ from benchmark G^i on the unseen G^u, and δ is a confidence threshold. This ensures that higher task similarity between G^u and G^i implies a larger likelihood of high-performing transfer from G^i to G^u.

To design *unsupervised* task similarity between G^u and G^i, we first apply the filter $\mathcal{F}(\cdot)$ on G^u and G^i to establish an empirical foundation based on *performance-oriented confidence* $\bar{I}(g_k)$. However, as discussed in Section 3.1, graph properties *inherently* exhibit varying levels of *confidence* across graphs, making it challenging to transfer knowledge reliably with standalone statistical properties. To account for *inherent* variability, we further introduce the adaptive importance w^u_k for each property g_k specifically tailored to the unseen graph G^u and define our unsupervised task similarity as:

$$\mathcal{S}(G^u, G^i) = \frac{1}{N_f}\sum_{k=1}^{N_f} w^u_k \cdot \frac{\bar{I}(g_k)}{1 + \hat{d^{ui}_k}}, \tag{5}$$

where N_f is the number of influential graph properties selected via empirical filtering, and $\bar{I}(g_k)/(1+\hat{d^{ui}_k})$ measures the *confident* performance similarity of property g_k between graphs G^u and G^i.

Adaptive Knowledge Elicitation and Alignment. While $\bar{I}(g_k)$ is grounded in the observations from benchmarks, their adaptation to unseen tasks requires finding the adaptive importance of property w_k^u, which cannot be observed without testing on G^u. Notably, such adaptive importance of property for a given graph commonly motivates specific GNN designs in the graph learning literature [51, 60], e.g., GraphSAGE [20] handles high-degree graphs. Therefore, we novelly propose to leverage LLMs—pre-trained on a vast corpus of graph learning literature—as meta-controllers to adaptively elicit w_k^u for G^u by aligning task descriptions, filtered graph properties, and high-performing architecture patterns in textual space, followed by contextual reasoning [34]. This process ensures that the data-specific weights w_k^u reflect interpretable insights from the literature on graph properties, GNNs, and performance, aligning with general empirical evidence to construct trustworthy meta-knowledge that empowers LLMs to quickly design effective, data-aware models for unseen tasks.

Knowledge Pool Construction. Using the computed task similarity set $\mathcal{S}^u$ for G^u, we rank the benchmark graphs and select the Top-N_s most similar sources. The knowledge pool $\mathcal{K}$ for G^u is then constructed by aggregating the top-performing models from them:

$$\mathcal{K} = \bigcup_{G^i \in \text{Top-}N_s(\mathcal{S}^u)} \{(G^i, \{\theta_{im}^*\}_{m=1}^{N_m})\}, \tag{6}$$

The top-performing models $\{\theta_i^*\}$ act as a truncated prior $\mathbb{P}(\theta_i^*)$, summarizing accumulated design knowledge from benchmark G^i, while task similarity $\mathcal{S}(G^u, G^i)$ serves as a gating module for solving pseudo-likelihood $\mathbb{P}(\mathcal{F}(G^u)|\theta_i^*)$, indicating the effectiveness of θ_i^* on unseen graph G^u.

3.3 Model Suggestion and Refinement

Overall, our formulation of Knowledge Pool $\mathcal{K}$ approximates the posterior $\mathbb{P}(\theta_i^*|\mathcal{F}(G^u))$, prioritizing models that align with data-aware evidence. This Bayesian-inspired process ensures solid use of constructed meta-knowledge for designing models in both one-shot and a few attempts.

Initial Model Suggestion. By supplying contextual meta-knowledge of high similarities to the unseen graph G^u from $\mathcal{K}$, we leverage LLM as a surrogate meta-controller to sample N_s initial proposals $\{\theta_{ui}\}_{i=1}^{N_s}$ based on the approximated posterior, where $\mathcal{K}_i$ represents the specific knowledge from the i-th similar benchmark graph in $\mathcal{K}$: $\theta_{ui} \leftarrow \mathcal{LLM}_{IMS}(\mathcal{F}(G^u, \bar{I}), \mathcal{K}_i)$.. Compared to existing AutoGNNs, our method *avoids any model training* on unseen data, delivering promising designs by combining fine-grained literature and empirical insights for immediate model deployment.

Knowledge-Driven Refinement. To further optimize initial model proposals, we design a structured, knowledge-driven refinement strategy inspired by expert-like exploration-exploitation trade-offs, as illustrated in Figure 2. Our local search strategy iteratively improves models using the constructed meta-knowledge of high similarities to the unseen from $\mathcal{K}$. *(0) Re-Ranking:* Initial proposals $\{\theta_{ui}\}_{i=1}^{N_s}$ are re-ranked based on their validation performance, and the best-performing proposal θ_{u1} serves as the refinement starting point. *(1) Controlled Exploration:* Configurations from other models in $\mathcal{K}$ are crossovered with θ_{u1} to generate N_c new candidates. *(2) Model Promotion Mechanism:* Candidates are ranked based on their retrieved performances on the benchmark dataset of $\mathcal{K}_1$, with the most promising candidate $\theta_u^{t'}$ advanced for further refinement at iteration t. *(3) Directional Exploitation:* The LLM meta-controller $\mathcal{LLM}_{KDR}$ mutates $\theta_u^{t'}$ using user requirements, task descriptions, search space details, and previous training logs. *(4) Evaluation and Update:* Each refined candidate θ_u^t is validated and added to the optimization trajectory Θ^T. The best-performing model is updated as θ_u^*. This refinement process balances exploration (testing diverse configurations) and exploitation (focusing on high-potential candidates), leveraging prior knowledge in the textual space to efficiently optimize model designs.

4 Experiments

4.1 Experimental Settings

Task and Datasets. We conduct node classification studies using 8 benchmark datasets from NAS-Bench-Graph [39]: Cora, Citeseer, PubMed [41], CS, Physics, Photo, Computer [42], and ogbn-arXiv [21]. We include 3 extra datasets for out-of-distribution (e.g., heterophily) generalization: DBLP [3], Flickr [64], and Actor [38].

Baselines. We compare DesiGNN with 4 categories of 17 baselines: *(1) Manually Designed GNNs:* 7 GNNs [2, 10, 20, 26, 35, 47, 57] showcasing human experts' efforts in designing tailored models for specific types of graphs; *(2) Classical Automated Approaches:* 3 typical automated search strategies [29, 40, 74], plus 2 advanced AutoGNN systems [17, 71], all operating within the same search space as DesiGNN for fairness; *(3) Meta-based Approaches:* 3 meta-based, semi-supervised methods [5, 39, 60] that recommend models from the most similar benchmark graphs or based on meta-knowledge after testing anchor models on the unseen task; *(4) LLM-Based Approaches:* 2 recent SOTA LLM-Based methods [14, 48] that leverage LLMs for iterative architecture search.

Evaluation Metrics. To ensure reliability, we report the average accuracy and standard deviation over 10 runs. For automated methods, we also analyze the best-so-far accuracy after validating 1-30 model proposals to measure short-run efficiency. To assess efficiency under varying LLM API traffic, we set the basic time unit as the time to validate a model proposal on the unseen dataset.

Implementation Details. Our code[1] is implemented in PyTorch and LangChain (GPT-4). For clarity, `Desi-Init` refers to initial model proposals, while `DesiGNN` denotes the whole pipeline, including model refinement. We use NAS-Bench-Graph [39] to accumulate knowledge and provide a unified search space. To prevent data leakage, when a benchmark dataset is treated as unseen, it is completely excluded from the knowledge retrieval process and anonymized to ensure fair evaluation. We have validated that LLMs, given only dataset descriptions, lack prior knowledge of effective architectures in the NAS-Bench-Graph search space.

4.2 Main Results

Initial Model Suggestions. First, we use `Desi-Init` in Table 1 to refer to the initial model suggestion part in Section 3.3, i.e., the first GNN suggested by LLM within seconds (avg. 0.07 sec of DesiGNN system time cost + avg. 10.77 sec each for LLM API communication,

[1]**Full code, data, prompts:** https://github.com/jilwang84/DesiGNN.

Table 1: Initial performance comparison (accuracy) of GNNs designed by different methods. $N_s = 3$ for `Desi-Init`. Best overall results are in bold, and best among each category are underlined.

Type	Model	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com.	arX.	DBL.	Fli.	Act.
Manual GNNs	GCN	80.97(0.39)	69.90(1.26)	77.46(0.61)	88.65(0.57)	90.85(1.20)	89.44(0.48)	83.16(0.55)	71.08(0.16)	84.25(0.25)	54.27(0.14)	24.78(1.79)
	GAT	80.83(0.47)	70.70(0.71)	75.93(0.26)	88.72(0.73)	89.47(1.14)	89.93(1.75)	81.35(1.26)	71.24(0.10)	84.98(0.15)	51.85(0.26)	26.91(1.09)
	SAGE	79.47(0.31)	66.13(0.90)	75.50(1.14)	87.81(0.18)	91.43(0.29)	88.29(1.03)	81.46(0.73)	70.78(0.17)	85.41(0.06)	52.84(0.19)	30.13(0.70)
	GIN	79.77(0.38)	63.30(1.26)	76.74(0.86)	81.08(3.09)	86.67(0.86)	87.37(1.01)	73.95(0.16)	61.33(0.70)	82.82(0.82)	51.08(0.08)	27.15(0.38)
	ChebNet	79.40(0.57)	67.03(1.02)	75.13(0.49)	89.50(0.36)	89.75(0.87)	86.65(0.77)	79.10(2.26)	70.87(0.10)	84.84(0.21)	53.75(0.16)	30.46(0.77)
	ARMA	78.33(0.69)	66.20(0.75)	75.00(0.51)	89.87(0.35)	88.88(1.09)	86.55(3.35)	78.47(0.57)	70.87(0.17)	84.31(0.30)	54.23(0.04)	31.29(0.43)
	k-GNN	78.06(0.47)	30.97(3.56)	75.38(0.97)	83.81(0.58)	88.98(0.54)	86.45(0.21)	76.31(1.34)	63.18(0.38)	83.59(0.07)	51.18(0.33)	32.74(0.68)
Auto. NNI	Random	77.87(2.41)	66.64(1.32)	74.16(1.68)	81.78(9.41)	90.59(0.94)	89.04(2.55)	76.61(3.56)	68.93(1.82)	76.45(7.53)	52.50(0.72)	30.90(3.75)
	EA	78.23(1.04)	66.40(2.63)	72.88(2.11)	87.03(2.64)	88.07(2.41)	87.30(1.38)	77.56(6.42)	68.28(2.95)	85.13(0.38)	53.22(0.92)	31.69(2.95)
	RL	73.44(8.11)	65.35(2.40)	75.44(1.24)	86.17(5.09)	88.15(4.24)	89.48(1.35)	77.70(3.07)	68.00(4.71)	84.18(0.50)	52.07(2.84)	31.72(4.60)
Auto. AutoGL	GNAS	78.55(1.20)	63.25(5.87)	73.04(1.64)	86.04(7.88)	89.54(1.52)	87.27(2.96)	70.96(9.66)	69.94(1.71)	84.46(0.41)	54.67(0.54)	32.31(3.25)
	Auto-GNN	78.58(2.18)	65.60(2.69)	76.07(0.77)	89.06(0.42)	89.26(1.51)	89.34(1.75)	77.49(3.41)	67.62(1.72)	84.67(0.62)	50.97(1.21)	30.18(4.67)
Meta -based	Kendall	67.73	69.20	71.80	88.56	91.56	88.90	76.85	71.49	-	-	-
	Overlap	79.36	67.30	71.80	88.56	89.95	90.37	76.85	71.68	-	-	-
	AutoTrans.	80.24(0.72)	68.19(1.22)	76.04(0.86)	89.19(0.77)	91.19(1.04)	90.93(0.37)	81.77(1.25)	71.37(0.39)	-	-	-
LLM -based	GPT4GNAS	78.50(0.37)	67.46(0.76)	73.89(0.86)	89.26(0.38)	89.44(1.94)	89.12(2.26)	77.21(5.26)	68.98(1.22)	84.93(0.22)	52.47(0.10)	34.26(0.47)
	GHGNAS	79.13(0.45)	67.35(0.44)	74.90(0.57)	89.15(0.81)	88.94(2.57)	89.42(1.99)	77.04(3.96)	69.66(1.28)	85.06(0.15)	52.48(0.23)	33.72(2.72)
Ours	`Desi-Init`	80.31(0.00)	69.20(0.16)	76.60(0.00)	89.64(0.08)	92.10(0.00)	91.19(0.00)	82.20(0.00)	71.50(0.00)	85.56(0.24)	55.16(0.11)	34.41(0.48)
	`DesiGNN`	**81.77(0.40)**	**71.00(0.09)**	**77.57(0.29)**	**90.51(0.42)**	**92.61(0.00)**	**92.38(0.06)**	**84.08(0.66)**	**72.02(0.18)**	**85.89(0.21)**	**55.44(0.06)**	**37.57(0.62)**

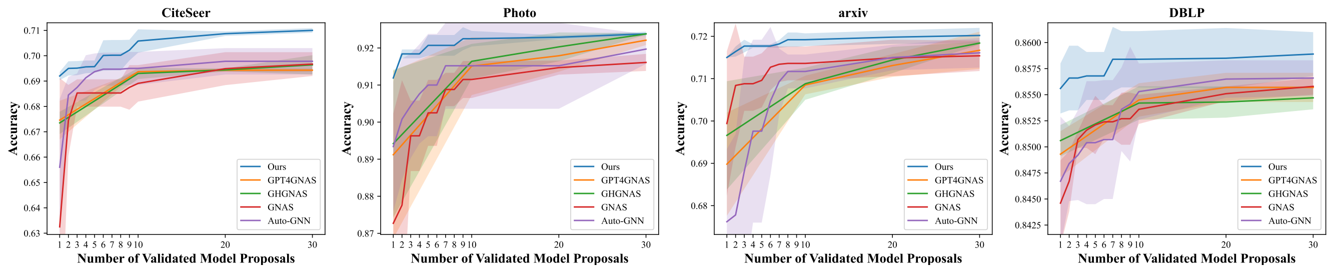


Figure 5: Short-run performance of `DesiGNN` compared to stronger automated baselines.

which may vary among users and API providers). While human-designed GNNs like GCN and GAT excel on popular datasets (e.g., Cora, Citeseer), they fail to generalize to less common heterophily datasets such as Flickr and Actor. In contrast, `Desi-Init` surpasses these overly specialized models, demonstrating robustness across diverse graphs. Notably, `Desi-Init` outperforms semi-supervised meta-based baselines, validating our contributions of establishing *proficient* "graph-GNN-performance" meta-knowledge through unsupervised task similarity assessment and graph understanding with adaptive properties. Furthermore, compared to automated baselines, `Desi-Init` achieves superior performance across all 11 datasets, underscoring the strength of our knowledge-driven approach over conventional automated methods (which lack domain-specific insights) and LLM-based methods (which rely solely on commonplace knowledge). Overall, DesiGNN's initial model proposals rank in the top 5.77% of all possible architectures.

Model Refinement and Short-run Efficiency. Building on the initial proposals, `DesiGNN` can rapidly refine models to outperform all manually designed and automated GNNs across 11 datasets, demonstrating its efficient use of meta-knowledge to exploit the high potential range. Figure 5 highlights `DesiGNN`'s short-run efficiency, achieving significant performance improvements with fewer

Table 2: Computational resources (in #proposal validations) needed by baselines to reach `DesiGNN`'s 10-validation performance. * contains failed cases within 100 validations.

	#Model Proposal Validations									
Model	**Cora**	**Cit.**	**Pub.**	**CS**	**Phy.**	**Pho.**	**Com.**	**arX.**	**DBL.**	**Fli.**
GNAS	40.3^*	48.3^*	21.5^*	38.8	∞	66^*	34.8	55^*	69.4	77.2^*
Auto-.	21.8^*	∞	11.6	18.4	43^*	29.3^*	17^*	59.3^*	76.2^*	∞
Rand.	21.7^*	∞	45	18.8	89^*	33^*	22	87^*	64.7	∞
EA	24.7^*	∞	48.8^*	40.8^*	∞	23^*	51.3^*	72.8^*	∞	46.3
RL	82^*	∞	11.8	14.8	∞	84.5^*	48.2	37^*	38.8	99^*
G4G.	30-40	∞	40-50	10	∞	30-40	10-20	40-50	40-50	∞
GHG.	20-30	∞	40-50	10	∞	20-30	20-30	30-40	∞	∞

model proposal validations than automated baselines. Unlike methods that require extensive validation to identify feasible models, `DesiGNN` emulates human-like refinement strategies, reducing computational costs and accelerating optimization. Specifically, Table 2 shows that other automated methods require significantly more model validations to match `DesiGNN`'s performance after just 10 validations. These results confirm that DesiGNN effectively addresses

Table 3: The main ablation study on each component in GNN Model Suggestion and Refinement. * is our complete setting.

	ACC (STD) %							
Method	**Cora**	**Citeseer**	**PubMed**	**CS**	**Physics**	**Photo**	**Computers**	**arXiv**
Initial Proposal Performance								
Property Only*	**80.31 (0.00)**	69.20 (0.16)	**76.60 (0.00)**	**89.64 (0.08)**	**92.10 (0.00)**	**91.19 (0.00)**	**82.20(0.00)**	**71.50 (0.00)**
Descriptional Only	**80.31 (0.00)**	**69.26 (0.00)**	75.71 (0.00)	89.53 (0.00)	88.42 (0.00)	91.17 (0.13)	81.79 (1.21)	71.20 (0.34)
Both	**80.31 (0.00)**	**69.26 (0.00)**	75.71 (0.00)	89.55 (0.03)	91.45 (0.78)	91.13 (0.07)	**82.20(0.00)**	**71.50 (0.00)**
w/o Knowledge	79.30 (0.00)	55.29 (0.00)	71.56 (0.00)	81.94 (0.00)	91.45 (0.00)	86.61 (0.00)	69.32 (0.00)	70.68 (0.00)
Best Performance After 30 Validations								
All*	**81.77 (0.40)**	**71.00 (0.09)**	**77.57 (0.29)**	**90.51 (0.42)**	92.61 (0.00)	**92.38 (0.06)**	**84.08 (0.66)**	**72.02 (0.18)**
w/o Re-rank	**81.77 (0.40)**	69.77 (0.31)	77.40 (0.24)	90.47 (0.00)	92.61 (0.00)	**92.38 (0.06)**	83.90 (0.14)	71.87 (0.12)
w/o Promotion	81.04 (0.42)	70.33 (0.00)	77.00 (0.30)	90.19 (0.47)	**92.71 (0.18)**	92.22 (0.16)	82.85 (0.01)	71.99 (0.01)
w/o Exploration	81.61 (0.10)	70.34 (0.64)	77.40 (0.14)	90.24 (0.33)	92.61 (0.00)	91.87 (0.18)	83.74 (0.23)	71.99 (0.08)
w/o Knowledge	80.90 (0.00)	69.93 (0.00)	76.72 (0.00)	90.24 (0.00)	91.82 (0.00)	92.21 (0.00)	83.63 (0.00)	71.71 (0.00)

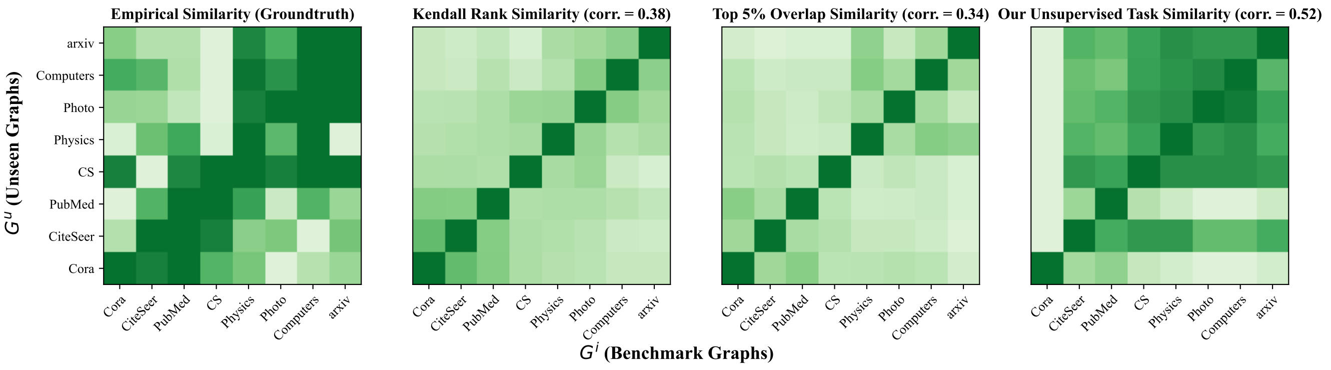


Figure 6: Task similarities computed by empirical performance (target), two meta-based baselines, and ours. Kendall's τ correlation coefficient with the empirical ground truth is shown in parentheses.

Table 4: Hit rate δ^* across unseen datasets.

Top-s Benchs	N_s=1	N_s=2	N_s=3
Kendall	0	0.25	0.625
Overlap	0	0.375	0.625
Property Similar. (w/o LLMs)	**0.375**	0.5	0.625
Description Only (with LLMs)	0.262	0.5	0.575
Property Only (with LLMs)	**0.375**	**0.6**	**0.725**
Both (with LLMs)	0.275	0.5	0.6

computational inefficiencies by leveraging accumulated knowledge to enable immediate feedback and rapid optimization, delivering consistently superior performance under limited budgets.

4.3 Effective Knowledge via Task Similarity

To assess whether our unsupervised task similarity in Equation 5 reflects the probability of successful knowledge transfer defined in Definition 3.1, we quantify the *hit rate*, defined as the tighter probability δ^* that the empirically most relevant benchmarks (i.e., those maximizing $\mathcal{H}(\{\theta^*_{im}\}, \cdot, G^u)$) are included in the Top-N_s benchmarks retrieved by $\mathcal{S}$ (N_s is the tolerance). A higher hit rate directly supports the probabilistic claim in Definition 3.1, where $\mathcal{S}$ correlates with the likelihood of transferring high-performing models from G^i to G^u. As shown in Table 4, our unsupervised task similarity based on graph properties achieves the highest hit rate across all tolerance settings, outperforming methods that rely on Kendall or Top 5% similarities, as well as purely statistical property similarity (without LLMs to conduct adaptive knowledge elicitation and alignment). This demonstrates that our *performance-oriented* $\mathcal{S}$, derived from key graph properties and adaptive weighting by LLMs, effectively captures the meta-knowledge required for data-aware model design. Besides, we observed a negative effect when incorporating descriptive inputs into similarity assessment, suggesting that some semantic properties (e.g., being a citation graph) are less relevant to designing the model architecture (see the case study in Figure 1b). We further visualize the pairwise similarities between graphs in Figures 3b and 6, showing that our unsupervised task similarity closely aligns with empirical performance similarity (the highest Kendall's τ correlation). This further validates the claim that $\mathcal{S}$ approximates a pseudo-likelihood $\mathbb{P}(\mathcal{F}(G^u)|\theta^*_i)$ well, reinforcing its role in supporting the Bayesian-inspired framework in Section 3.2.

4.4 Main Ablation Study

We present the main ablation study on the Knowledge Retrieval and the GNN Model Suggestion and Refinement module in Table 3. The results from the initial model suggestion stage underscore the

Table 5: Different LLMs for model suggestion and refinement.

Dataset	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com.	arX.
Initial Model Suggestion								
Llama2	80.26	68.90	75.83	**89.84**	90.75	**91.35**	80.27	70.95
GPT-4	**80.31**	**69.20**	**76.60**	89.64	**92.10**	91.19	**82.20**	**71.50**
Model Refinement								
Llama2	80.96	70.38	77.40	90.19	91.47	92.09	83.69	71.63
GPT-4	**81.77**	**71.00**	**77.57**	**90.51**	**92.61**	**92.38**	**84.08**	**72.02**

importance of designing an effective Graph Understanding method, demonstrating that relying solely on filtered graph properties in dataset descriptions outperforms using descriptional inputs. In the short run, after 30 model validations, the Re-rank mechanism significantly improves the knowledge-driven model refinement process when the original order of the Knowledge Pool $\mathcal{K}$ is incorrect. Additionally, the Model Promotion mechanism, which simulates the strategy of human experts refining a model based on accumulated knowledge, plays a crucial role in enhancing the efficacy of model refinement. Lastly, the Directional Exploration mechanism is empirically beneficial, as it leverages the most relevant model design knowledge and the in-context learning ability of LLMs to further refine the best candidate model promoted.

4.5 Case Studies

Different LLMs. To study the effect of different LLMs as meta-controllers, we use an open-source Llama2-13b to replace the GPT-4 meta-controller. Table 5 shows the comparative results on the initial model proposal and refinement. We observed that replacing GPT-4 with Llama2 results in a slight degradation of the performance of recommended models in both settings, and is more consistent in model refinement. This suggests that GPT-4 is superior at proposing a reasonable next step from the trajectory and extra knowledge, consistent with the existing study [59]. Notably, because the initial model proposal phrase relies more on the quality of our elicited knowledge, Llama2 can also achieve close performance to GPT-4.

Lack of Prior Knowledge in LLMs. The *inherent* knowledge gap illustrated in Figure 1a refers to the lack of prior knowledge in LLMs about the top-performing models of benchmark datasets within the NAS-Bench-Graph [39] model space. We analyze LLM responses in which the descriptive inputs or graph properties were directly sent to the LLMs for model suggestions. The results demonstrate that LLMs are only able to suggest commonplace layer connections and operations based on user-provided descriptive inputs or key graph properties. We have tested all benchmark datasets in NAS-Bench-Graph and found that only two macro lists and three operation lists would be recommended by LLMs regardless of the datasets: (1) **Architecture:** [0, 1, 2, 3] and [0, 0, 1, 3]; (2) **Operations:** ['gcn', 'gat', 'sage', 'skip'], ['gcn', 'gat', 'sage', 'gcn'], and ['gcn', 'gat', 'gin', 'sage']. After providing tailored knowledge derived from our framework, the LLM-recommended model becomes specialized and shows significant performance improvements.

Artificial Hallucination in LLMs. In Figure 1b, we examine the phenomenon of "artificial hallucination," or the *external* noise issue, in LLMs when comparing the similarities between unseen and benchmark datasets (Section 3.2). Figure 6 (leftmost) illustrates that the empirically most similar benchmark datasets to PubMed are CS, Physics, and Citeseer, in descending order of similarity. When employing our Graph Understanding method in Section 3.1, which leverages *confident* graph properties, the top three most similar datasets are Citeseer, CS, and Physics (align with the empirical ground truth). However, the current practice that relies solely on descriptive inputs identifies Cora, Citeseer, and ogbn-arxiv as the top three, which does not align with the empirical ground truth. This discrepancy arises because LLMs overly rely on the shared characteristic of being citation graphs, assuming that citation datasets like PubMed, Cora, Citeseer, and ogbn-arxiv should have similar model design preferences. This reveals that relying solely on *external* descriptive inputs is insufficient to capture the similarities between datasets, which can overwhelm other pertinent information, leading to inaccurate task association and knowledge retrieval.

5 Conclusion

We present DesiGNN, a knowledge-centered framework that demonstrates deep expertise in *proficiently* designing data-aware GNNs and eliminates *external* noise by systematically converting accumulated model design experience into fine-grained meta-knowledge well suited to meta-learning with LLMs. Leveraging a performance-oriented task similarity grounded in empirically validated graph properties and adaptive knowledge elicitation via LLMs, DesiGNN bridges the gap between literature insights and practical, data-specific model deployment. DesiGNN integrates graph understanding, effective knowledge retrieval, and strategic model suggestion and refinement under a Bayesian perspective, enabling tailored GNN designs for diverse graphs in a single shot or with a few attempts. Extensive experiments demonstrate that DesiGNN expedites the design process by delivering top-tier GNN designs within seconds and achieving superior performance at minimal cost.

Acknowledgments

Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, Guangdong-Hong Kong Technology Innovation Joint Funding Scheme Project No. 2024A0505040012, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Key Areas Special Project of Guangdong Provincial Universities 2024ZDZX1006, Guangdong Province Science and Technology Plan Project 2023A0505030011, Guangzhou municipality big data intelligence key lab, 2023A03J0012, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab, 2023 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund, from Shui On Xintiandi and the InnoSpace GBA, and HKUST(GZ) - CMCC(Guangzhou Branch) Metaverse Joint Innovation Lab under Grant No. P00659. Jiachuan Wang's work is supported in part by JST CREST (JPMJCR22M2). Shimin Di's work is supported by National Science Foundation of China (NSFC) under Grant No. 62506075.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6430–6439.

[2] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3496–3507.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Kaidi Cao, Jiaxuan You, Jiaju Liu, and Jure Leskovec. 2023. AutoTransfer: AutoML with Knowledge Transfer - An Application to Graph Neural Networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/forum?id=y81ppNf_vg

[6] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2021. Meta-learning via language model in-context tuning. *arXiv preprint arXiv:2110.07814* (2021).

[7] Liying Cheng, Xingxuan Li, and Lidong Bing. 2023. Is gpt-4 a good data analyst? *arXiv preprint arXiv:2305.15038* (2023).

[8] Krishna Teja Chitty-Venkata, Murali Emani, Venkatram Vishwanath, and Arun K Somani. 2023. Neural architecture search benchmarks: Insights and survey. *IEEE Access* 11 (2023), 25217–25236.

[9] Hyunjun Choi, Jay Moran, Nicholas Matsumoto, Miguel E Hernandez, and Jason H Moore. 2023. Aliro: an automated machine learning tool leveraging large language models. *Bioinformatics* 39, 10 (2023), btad606.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Shimin Di and Lei Chen. 2023. Message function search for knowledge graph embedding. In *Proceedings of the ACM Web Conference 2023*. 2633–2644.

[13] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to sparsify tensor decomposition for n-ary relational data. In *Proceedings of the web conference 2021*. 4043–4054.

[14] Haoyuan Dong, Yang Gao, Haishuai Wang, Hong Yang, and Peng Zhang. 2023. Heterogeneous graph neural architecture search with gpt-4. *arXiv preprint arXiv:2312.08680* (2023).

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[16] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[17] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2021. Graph neural architecture search. In *International joint conference on artificial intelligence*. International Joint Conference on Artificial Intelligence.

[18] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

[19] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. *arXiv preprint arXiv:2402.17453* (2024).

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[21] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[22] Zhao Huan, YAO Quanming, and TU Weiwei. 2021. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 552–563.

[23] Roshni G Iyer, Yunsheng Bai, Wei Wang, and Yizhou Sun. 2022. Dual-geometric space embedding model for two-view knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 676–686.

[24] Ganesh Jawahar, Muhammad Abdul-Mageed, Laks VS Lakshmanan, and Dujian Ding. 2023. LLM Performance Predictors are good initializers for Architecture Search. *arXiv preprint arXiv:2310.16712* (2023).

[25] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1-2 (1938), 81–93.

[26] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[27] Naveen Krishnan. 2025. AI Agents: Evolution, Architecture, and Real-World Applications. *arXiv preprint arXiv:2503.12687* (2025).

[28] Cat P Le, Mohammadreza Soltani, Juncheng Dong, and Vahid Tarokh. 2022. Fisher task distance and its application in neural architecture search. *IEEE Access* 10 (2022), 47235–47249.

[29] Liam Li and Ameet Talwalkar. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*. PMLR, 367–377.

[30] YuFei Li, Jia Wu, and TianJin Deng. 2023. Meta-GNAS: Meta-reinforcement learning for graph neural architecture search. *Engineering Applications of Artificial Intelligence* 123 (2023), 106300.

[31] Hanmo Liu, Shimin Di, Jialiang Wang, Zhili Wang, Jiachuan Wang, Xiaofang Zhou, and Lei Chen. [n. d.]. Structuring Benchmark into Knowledge Graphs to Assist Large Language Models in Retrieving and Designing Models. In *The Thirteenth International Conference on Learning Representations*.

[32] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. 2025. Graph Foundation Models: Concepts, Opportunities and Challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[33] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170* (2023).

[34] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837* (2022).

[35] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[36] Babatounde Moctard Oloulade, Jianliang Gao, Jiamin Chen, Tengfei Lyu, and Raeed Al-Sabri. 2021. Graph neural architecture search: A survey. *Tsinghua Science and Technology* 27, 4 (2021), 692–708.

[37] Chaoqian Ouyang, Ling Yue, Shimin Di, Libin Zheng, Linan Yue, Shaowu Pan, Jian Yin, and Min-Ling Zhang. 2025. Code2MCP: Transforming Code Repositories into MCP Services. *arXiv preprint arXiv:2509.05941* (2025).

[38] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[39] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2022. Nas-bench-graph: Benchmarking graph neural architecture search. *Advances in neural information processing systems* 35 (2022), 54–69.

[40] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[41] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[42] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[43] Min Shi, Yufei Tang, Xingquan Zhu, Yu Huang, David Wilson, Yuan Zhuang, and Jianxun Liu. 2022. Genetic-gnn: Evolutionary architecture search for graph neural networks. *Knowledge-based systems* 247 (2022), 108752.

[44] DI Shimin, YAO Quanming, CHEN Lei, et al. 2021. Efficient relation-aware scoring function search for knowledge graph embedding. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1104–1115.

[45] Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, et al. 2023. Automl in the age of large language models: Current challenges, future opportunities and risks. *arXiv preprint arXiv:2306.08107* (2023).

[46] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[47] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[48] Haishuai Wang, Yang Gao, Xin Zheng, Peng Zhang, Hongyang Chen, and Jiajun Bu. 2023. Graph neural architecture search with gpt-4. *arXiv preprint arXiv:2310.01436* (2023).

[49] Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Automated graph machine learning: Approaches, libraries and directions. *arXiv preprint arXiv:2201.01288* (2022).

[50] Zhili Wang, Shimin Di, and Lei Chen. 2021. Autogel: An automated graph neural network with explicit link information. *Advances in Neural Information Processing Systems* 34 (2021), 24509–24522.

[51] Zhili Wang, Shimin Di, and Lei Chen. 2023. A Message Passing Neural Network Space for Better Capturing Data-dependent Receptive Fields. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2489–2501.

[52] Zhenyi Wang, Huan Zhao, and Chuan Shi. 2022. Profiling the design space for graph neural networks based collaborative filtering. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1109–1119.

[53] Lanning Wei, Zhiqiang He, Huan Zhao, and Quanming Yao. 2023. Unleashing the power of graph learning through llm-based autonomous agents. *arXiv preprint arXiv:2309.04565* (2023).

[54] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling architecture search for graph classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2091–2100.

[55] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. 2020. Neural predictor for neural architecture search. In *European conference on computer vision*. Springer, 660–676.

[56] Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. 2021. How powerful are performance predictors in neural architecture search? *Advances in Neural Information Processing Systems* 34 (2021), 28454–28469.

[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[58] Shengxiang Xu, Jiayi Zhang, Shimin Di, Yuyu Luo, Liang Yao, Hanmo Liu, Jia Zhu, Fan Liu, and Min-Ling Zhang. 2025. RobustFlow: Towards Robust Agentic Workflow Generation. *arXiv preprint arXiv:2509.21834* (2025).

[59] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409* (2023).

[60] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 17009–17021.

[61] Caiyang Yu, Xianggen Liu, Wentao Feng, Chenwei Tang, and Jiancheng Lv. 2023. GPT-NAS: Evolutionary neural architecture search with the generative pre-trained model. *arXiv preprint arXiv:2305.05351* (2023).

[62] Ling Yue, Shimin Di, and Shaowu Pan. 2025. Autonomous scientific discovery through hierarchical ai scientist systems. *Preprints, July* (2025).

[63] Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, et al. 2025. Don't Overthink It: A Survey of Efficient R1-style Large Reasoning Models. *arXiv preprint arXiv:2508.02120* (2025).

[64] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).

[65] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762* (2024).

[66] Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. 2023. Mlcopilot: Unleashing the power of large language models in solving machine learning tasks. *arXiv preprint arXiv:2304.14979* (2023).

[67] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).

[68] Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automl-gpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499* (2023).

[69] Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. 2023. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970* (2023).

[70] WANG Zhili, DI Shimin, CHEN Lei, and ZHOU Xiaofang. 2024. Search to fine-tune pre-trained graph neural networks for graph-level tasks. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2805–2819.

[71] Kaixiong Zhou, Xiao Huang, Qingquan Song, Rui Chen, and Xia Hu. 2022. Auto-gnn: Neural architecture search of graph neural networks. *Frontiers in big Data* 5 (2022), 1029307.

[72] Yizhang Zhu, Liangwei Wang, Chenyu Yang, Xiaotian Lin, Boyan Li, Wei Zhou, Xinyu Liu, Zhangyang Peng, Tianqi Luo, Yu Li, et al. 2025. A Survey of Data Agents: Emerging Paradigm or Overstated Hype? *arXiv preprint arXiv:2510.23587* (2025).

[73] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

[74] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

Ethical Considerations

Our work contributes to the growing movement toward high-level automation in artificial intelligence by enabling the proficient, data-aware design of Graph Neural Networks (GNNs) through structured knowledge reuse and language model alignment. As AI systems become more complex and specialized, the ability to automate model design across unseen tasks without human expertise offers transformative potential for a wide range of real-world applications. DesiGNN lowers the barrier to applying advanced graph-based machine learning, making it more accessible to scientists, engineers, and practitioners across domains such as biology, social networks, and recommendation systems. By replacing ad hoc manual tuning with systematic and interpretable design strategies, our framework may promote reproducibility, consistency, and scalability in the deployment of machine learning systems. More broadly, this work aligns with the vision of LLM-powered agents acting as autonomous scientific assistants—able to rapidly translate raw data into effective models—thereby accelerating innovation in scientific discovery, infrastructure optimization, and decision-making. We hope our contribution supports the responsible expansion of AI automation by enabling efficient and expert-level design processes at scale.

However, such high-level automation also warrants caution. While DesiGNN significantly reduces computational cost and expertise requirements, its efficacy may be affected by the quality and representativeness of the knowledge source, as well as the reliability of LLMs' inherent contextual reasoning capabilities. Additionally, while our method operates at a level removed from direct applications, any downstream deployment of GNNs—e.g., in social network analysis or recommender systems—can have far-reaching societal implications, such as privacy leakage or algorithmic bias, if not handled responsibly.