

Figure 6: Empirical studies of graph property confidence as a stronger indicator of effective meta-knowledge in GNNs design.

## A Methodology

In this section, we provide a detailed explanation of the methodology behind our DesiGNN framework, including Graph Understanding (Section 3.1), Knowledge Retrieval (Section 3.2), and GNN Model Suggestion and Refinement Modules (Section 3.3).

### A.1 Algorithm

The DesiGNN methodology is detailed in Algorithm 1. The algorithm is divided into three main phases: Graph Understanding, Knowledge Retrieval, and GNN Model Suggestion and Refinement. It also includes details on the Self-evaluation Mechanism, Empirical Filtering Mechanism, Adaptive Filtering Mechanism, Re-rank Mechanism, Controlled Exploration, Model Promotion Mechanism, Directional Exploitation, and Optimization.

### A.2 Graph Understanding Module

The Graph Understanding Module (Section 3.1) is designed to automatically analyze the graph properties to enhance task comprehension beyond mere user input and establish an empirical foundation for our meta-knowledge construction and unsupervised task similarity. The module initially prioritizes the most influential properties based on their *confidences* averaged across all benchmark datasets, which is illustrated in Figure 6b. The selected features are then used to generate textual descriptions of graph datasets, which are subsequently employed to perform adaptive elicitation, align knowledge, and finally identify similar benchmark datasets to retrieve data-aware model design knowledge.

#### A.2.1 Graph Properties Pool

We employ a comprehensive set of 16 graph properties by analyzing the graph learning literature [54, 47]. Illustrated in Table 1, the corresponding properties name from  $g_1$  to  $g_{16}$  are listed below (and are also in the initialized order based on their *confidences*  $\bar{I}(g_k)$ ): the average clustering coefficient, average betweenness centrality, density, average degree centrality, average closeness centrality, average degree, edge count, graph diameter, average shortest path length, assortativity, average eigenvector centrality, feature dimensionality, node count, node feature diversity, connected components, and label homophily. Together, they could capture the topology of graphs, which commonly motivates the specific design of GNNs in research. The underlined properties are measured on sampled subgraphs to reduce computational overhead and ensure scalability.

#### A.2.2 Statistical and Empirical Similarities

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 to infer graph similarity from two aspects: 1) the statistical distance ranking based on graph properties  $g_k$ , and 2) the empirical performance ranking of transferred top-performing GNNs from other graphs  $\{G^j \mid j \neq i\}$  to anchor graph  $G^i$ .

---

**Algorithm 1** DesiGNN Methodology

---

**Input:** Unseen Graph Dataset  $G^u$   
**Data:** Benchmark Datasets  $\{G^i\}_{i=1}^n$ , Benchmark Knowledge Base  $\mathcal{H}_G : \Theta \rightarrow P$ , Self-evaluation Bank  $BE$ , Task-aware LLMs  $\mathcal{LLM}_{GDC}, \mathcal{LLM}_{IMS}, \mathcal{LLM}_{KDR}$ , Empirical Filter  $\mathcal{F}(\cdot, \cdot)$ , #Properties  $N_f$ , Knowledge Pool size  $N_s$ , #Top models  $N_m$ , #Candidates  $N_c$ , stop criteria  $maxTrials$   
**Output:** Optimized GNN architecture  $\theta_u^*$  and model parameters  $\omega^*$

- 1: **Phase 1: Graph Understanding Module**
- 2: **if**  $\mathcal{F}(\cdot, \cdot)$  needs initialization or update **then**
- 3:   **Self-evaluation Mechanism**
- 4:   **for** each anchor dataset  $G^i$  and property  $g_k \in BE$  **do**
- 5:     **for** each other dataset  $G^j \in BE, G^j \neq G^i$  **do**
- 6:       Compute  $d_k^{ij} = |g_k^i - g_k^j|$  and retrieve  $p^{ij} \in BE$
- 7:     **end for**
- 8:      $\mathcal{SR}(i, k) = \text{argsort}(\{d_k^{ij} \mid j \neq i, G^j \in \mathbf{G}\}, \text{ASC})$
- 9:      $\mathcal{ER}(i) = \text{argsort}(\{p^{ij} \mid j \neq i, G^j \in \mathbf{G}\}, \text{DESC})$
- 10:     $I(G^i, g_k) := \text{KendallCorr}(\mathcal{SR}(i, k), \mathcal{ER}(i))$
- 11:    **end for**
- 12:     $\bar{I}(g_k) = \frac{1}{n} \sum_{i=1}^n I(G^i, g_k)$
- 13:    Initialize/Update  $\mathcal{F}(\cdot, \cdot) = \text{Top}_{N_f}(\{\bar{I}(g_k)\})$
- 14: **end if**
- 15: Generate unseen dataset's description with  $\mathcal{F}(G^u, \bar{I})$  and benchmark datasets' descriptions with  $\mathcal{F}(G^i, \bar{I})$
- 16: **Phase 2: Knowledge Retrieval Module**
- 17: Compute  $\mathcal{S}^u = \{\frac{1}{N_f} \sum_{k=1}^{N_f} w_k^u \cdot \frac{\bar{I}(g_k)}{1+d_k^{ui}}\}_{i=1}^n$ ;  
     $w_k^u$  is determined by  $\mathcal{LLM}_{GDC}(\mathcal{F}(G^u, \bar{I}), \mathcal{F}(\{G^i\}_{i=1}^n, \bar{I}))$
- 18: Retrieve  $\mathcal{K} = \bigcup_{G^i \in \text{Top-}N_s(\mathcal{S}^u)} \{(G^i, \{\theta_{im}^*\}_{m=1}^{N_m})\}$
- 19: **Phase 3: Initial Model Suggestion**
- 20: **for** each  $\mathcal{K}_i \in \mathcal{K}$  **do**
- 21:    $\theta_{ui} \leftarrow \mathcal{LLM}_{IMS}(\mathcal{F}(G^u, \bar{I}), \mathcal{K}_i)$
- 22:    $p^{ui} = \mathcal{H}(\theta_{ui}, \omega; G^u)$
- 23: **end for**
- 24:  $BE \leftarrow BE \cup \{(G^u, \{g_k^u\}, \{p^{ui}\})\}$
- 25: **Phase 4: Model Proposal Refinement**
- 26:  $R = \text{argsort}(\{p_{ui}\}, \text{DESC})$
- 27:  $\mathcal{K}_{1:N_s} = \{(G^{R[j]}, \{\theta_{R[j]m}^*\}_{m=1}^{N_m})\}_{j=1}^{N_s}$
- 28:  $\Theta_u^T \leftarrow (\theta_u^{R[1]}, p_u^{R[1]}); \theta_u^* = \theta_u^T[1]$
- 29: **for**  $t = N_s$  **to**  $maxTrials$  **do**
- 30:    $C^t = \{\theta_{ui}^{t'} \mid \text{Crossover}(\theta_u^*, \mathcal{K}_{2:N_s})\}_{i=1}^{N_c}$
- 31:    $\theta_u^{t'} = \text{argsort}(\{\mathcal{H}_{\mathcal{K}_1}(\theta_{ui}^{t'}) \mid \theta_{ui}^{t'} \in C^t\}, \text{DESC})[1]$
- 32:    $\theta_u^t \leftarrow \mathcal{LLM}_{MPR}(\theta_u^{t'}, \Theta^T, \mathcal{K}_1)$
- 33:    $p_u^t = \mathcal{H}(\theta_u^t, \omega; G^u)$
- 34:    $\Theta_u^T \leftarrow (\theta_u^t, p_u^t)$
- 35:   **if**  $p_u^t > p_u^*$  **then**
- 36:      $\theta_u^*, \omega^* = \theta_u^t, \omega$
- 37:      $p_u^* = p_u^t$
- 38:   **end if**
- 39: **end for**
- 40: **return**  $\theta_u^*, \omega^*$

---

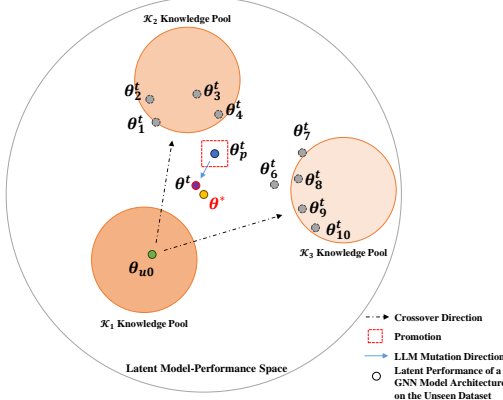


Figure 7: Conceptual illustration of controlled exploration and directional exploitation in GNN model suggestion and refinement.

Table 5: Statistics of the datasets.

Dataset	#Vtx	#Edges	Feat.	Cls	Metric
<b>Cora</b>	2,708	5,429	1,433	7	Acc.
<b>Citeseer</b>	3,327	4,732	3,703	6	Acc.
<b>PubMed</b>	19,717	44,338	500	3	Acc.
<b>CS</b>	18,333	81,894	6,805	15	Acc.
<b>Physics</b>	34,493	247,962	8,415	5	Acc.
<b>Photo</b>	7,487	119,043	745	8	Acc.
<b>Computers</b>	13,381	245,778	767	10	Acc.
<b>arxiv</b>	169,343	1,166,243	128	40	Acc.
<b>proteins</b>	132,534	39,561,252	8	112	ROC
<b>DBLP</b>	17,716	105,734	1,639	4	Acc.
<b>Flickr</b>	89,250	899,756	500	7	Acc.
<b>Actor</b>	7,600	30,019	932	5	Acc.

Specifically, given two graphs  $G^i$  and  $G^j \in \mathbf{G}$ , we first compute the statistical distance  $d_k^{ij} = |g_k^i - g_k^j|$  between  $G^i$  and  $G^j$  based on  $g_k$ , i.e., the reverse ranking of  $\{d_k^{ij}\}_{k=1}^{16}$  can assess the sectional topological similarities between graphs  $G^i$  and  $G^j$  after normalization. Then, to quantify empirical transferability, we let  $p^{ij}$  denote the performance of transferring the top-performing GNN patterns summarized by LLMs from another graph  $G^j$  to  $G^i$  (details in Section 3.3). Based on  $d_k^{ij}$  and  $p^{ij}$ , we define two similarity rankings of other graphs  $\{G^j \mid j \neq i, G^j \in \mathbf{G}\}$  to graph  $G^i$ :

$$\mathcal{SR}(i, k) = \text{argsort} \left( \{d_k^{ij} \mid j \neq i, G^j \in \mathbf{G}\}, \text{ASC} \right), \quad (7)$$

$$\mathcal{ER}(i) = \text{argsort} \left( \{p^{ij} \mid j \neq i, G^j \in \mathbf{G}\}, \text{DESC} \right) \quad (8)$$

where  $\mathcal{SR}(i, k)$  represents the statistical similarity ranking based on property  $g_k$ , and  $\mathcal{ER}(i)$  represents the empirical ground truth of knowledge transferability when meta-learning with LLMs.

### A.3 Knowledge Retrieval Module

As the knowledge source of our Knowledge Retrieval Module (Section 3.2), NAS-Bench-Graph [35] is a comprehensive benchmark designed to facilitate unified, reproducible, and efficient evaluations of graph neural architecture search (GraphNAS) methods. This benchmark encapsulates a well-defined search space and a rigorous evaluation protocol, which encompasses the training and testing of 26,206 unique graph neural network (GNN) architectures across nine diverse node classification datasets.

#### A.3.1 Benchmark Datasets

The benchmark utilizes nine graph datasets of varied sizes and types, including citation networks (Cora, Citeseer, PubMed) [37], co-authorship graphs (Coauthor CS and Coauthor Physics) [38], co-purchase networks (Amazon Computers and Amazon Photo) [38], and large-scale graphs like ogbn-arXiv [20] from the Open Graph Benchmark. These datasets are employed with fixed semi-supervised [35] splits to ensure consistent evaluation settings.

#### A.3.2 Search Space Design

The search space in NAS-Bench-Graph is constructed as a directed acyclic graph (DAG), representing various GNN architectures. It includes macro architectural choices constrained to 9 distinct patterns: [0,0,0,0], [0,0,0,1], [0,0,1,1], [0,0,1,2], [0,0,1,3], [0,1,1,1], [0,1,1,2], [0,1,2,2], and [0,1,2,3]. The operations include seven prominent GNN layer types—GCN [24], GAT [42], GraphSAGE [18], GIN [52], ChebNet [10], ARMA [2], and k-GNN [32]—plus Identity and fully connected layers for residual connections and non-graph structure use, respectively.

Table 6: Summary of hyperparameters used for each dataset

Dataset	#Pre-process	#Post-process	Dimension	Dropout	Optimizer	Learning Rate	Weight Decay	Epochs
Cora	0	1	256	0.7	SGD	0.1	0.0005	400
Citeseer	0	1	256	0.7	SGD	0.2	0.0005	400
PubMed	0	0	128	0.3	SGD	0.2	0.0005	500
CS	1	0	128	0.6	SGD	0.5	0.0005	400
Physics	1	1	256	0.4	SGD	0.01	0	200
Photo	1	0	128	0.7	Adam	0.0002	0.0005	500
Computers	1	1	64	0.1	Adam	0.005	0.0005	500
ogbn-arxiv	0	1	128	0.2	Adam	0.002	0	500
ogbn-proteins	1	1	256	0	Adam	0.01	0.0005	500
DBLP	1	1	256	0.5	SGD	0.1	0.0005	300
Flickr	1	1	128	0.5	Adam	0.001	0.0005	300
Actor	1	1	128	0.5	Adam	0.005	0.0005	400

Table 7: Performance Ranking of Initial Model Proposals.

Performance Rank (%)									
Avg.	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com	arX.	
<b>5.765</b>	5.899	3.324	4.926	11.036	0.053	3.144	7.353	10.383	

#### A.4 GNN Model Suggestion and Refinement

As introduced in Section 3.3 and illustrated in Figure 2, our model design refinement cycle continues iteratively, with each step methodically enhancing the model architecture based on accumulated knowledge and feedback. This process integrates user requirements, retrieved top-performing models from  $\mathcal{K}$ , and search space description (e.g., macro architectures and operations). The detailed procedure is as follows: (0) *Re-Ranking*: Initial proposals  $\{\theta_{ui}\}_{i=1}^{N_s}$  are re-ranked based on their validation performance, and the best-performing proposal  $\theta_{u1}$  serves as the refinement starting point. (1) *Controlled Exploration*: Configurations from other models in  $\mathcal{K}$  are crossovered with  $\theta_{u1}$  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  $\theta_u^t$  is validated and added to the optimization trajectory  $\Theta^T$ . The best-performing model is updated as  $\theta_u^*$ .

As conceptually illustrated in Figure 7, the search space is navigated in a controlled manner through simulated crossovers between  $\theta^*$  and configurations from  $\mathcal{K}_{2:N_s}$ , ensuring that exploration remains within high-potential range. Among these generated candidates, the one that demonstrates the highest potential based on its performance on the closest benchmark dataset is identified as the most promising for the unseen dataset. This candidate is then subjected to further exploitation by the LLM, guided by the most valuable directional insights derived from  $\mathcal{K}_1$ . Our empirical results support the short-run effectiveness of this local search strategy in quickly refining models. Our retrieval-then-verified promotion strategy not only ensures a reliable and efficient selection of new proposals by leveraging pre-existing, empirically validated knowledge but also requires only a single evaluation per optimization step. This sharply contrasts with existing traditional and LLM-based methods that necessitate accumulating knowledge from scratch and running multiple evaluations in parallel, thereby reducing computational overhead and accelerating the refinement process.

## B Experiments

### B.1 Experimental Settings

#### B.1.1 Task and Datasets.

We evaluate our DesiGNN framework on eleven diverse graph datasets, including eight out of nine benchmark datasets { Cora [37], Citeseer [37], PubMed [37], CS [38], Physics [38], Photo [38], Computer [38], ogbn-arXiv [20] } from NAS-Bench-Graph [35] and three additional datasets { DBLP [3], Flickr [56], Actor [34] }. All datasets are used to conduct node classification tasks, and

Table 8: Best model performance (accuracy) after 10 model validations on the benchmark datasets.

Type	Model	ACC (STD) %										
		Cora	Citeseer	PubMed	CS	Physics	Photo	Computer	arXiv	DBLP	Flickr	Actor
Auto.	GNAS	80.89(0.35)	67.45(1.40)	76.36(0.50)	88.94(0.93)	89.55(2.17)	87.76(3.37)	75.70(5.54)	69.84(1.46)	84.67(0.29)	54.69(1.02)	35.12(1.71)
	Auto-GNN	80.63(1.13)	68.45(0.72)	76.35(0.66)	88.46(2.52)	90.94(0.48)	90.08(1.63)	78.39(3.76)	67.78(1.91)	84.84(0.36)	51.27(2.01)	31.32(5.31)
Auto.	Random	80.79(0.60)	65.53(4.95)	75.79(0.79)	88.95(1.41)	90.68(0.27)	90.76(1.00)	76.95(3.00)	71.27(0.52)	82.68(5.89)	52.93(1.21)	33.33(3.35)
	EA	81.27(0.59)	67.29(1.37)	74.92(0.88)	88.08(1.61)	91.05(0.64)	89.63(1.26)	80.46(1.88)	70.40(0.50)	85.15(0.33)	54.38(0.87)	33.83(2.65)
	RL	80.28(0.77)	67.39(1.06)	75.39(1.34)	88.47(1.53)	90.70(0.90)	88.84(0.39)	77.48(3.90)	69.12(1.78)	84.46(0.23)	53.34(0.94)	33.61(2.43)
LLM	GPT4GNAS	80.33(0.39)	69.37(0.01)	76.40(0.20)	90.24(0.06)	91.06(0.61)	91.51(0.78)	82.88(1.25)	70.85(0.22)	85.45(0.16)	55.12(0.16)	36.14(0.54)
-based	GHGNAS	80.51(0.45)	69.30(0.22)	76.49(0.32)	<b>90.28(0.29)</b>	90.94(0.28)	91.64(0.47)	82.40(1.21)	70.88(0.38)	85.42(0.16)	55.00(0.18)	36.56(0.49)
<b>Our</b>	<b>DesiGNN</b>	<b>81.69(0.54)</b>	<b>70.58(0.46)</b>	<b>77.17(0.29)</b>	90.22(0.48)	<b>92.61(0.00)</b>	<b>92.25(0.21)</b>	<b>83.15(0.61)</b>	<b>71.92(0.15)</b>	<b>85.84(0.27)</b>	<b>55.20(0.09)</b>	<b>36.58(2.63)</b>

Table 9: Best model performance (accuracy) after 20 model validations on the benchmark datasets.

Type	Model	ACC (STD) %										
		Cora	Citeseer	PubMed	CS	Physics	Photo	Computer	arXiv	DBLP	Flickr	Actor
Auto.	GNAS	81.35(0.23)	69.49(0.65)	76.94(0.21)	90.12(0.27)	91.64(0.52)	91.47(0.19)	83.15(1.23)	71.49(0.43)	85.51(0.07)	55.22(0.21)	36.82(0.59)
	Auto-GNN	81.07(0.65)	69.78(0.52)	77.21(0.58)	90.36(0.35)	91.92(0.65)	91.52(1.16)	82.73(1.17)	71.50(0.30)	85.65(0.18)	54.99(0.18)	37.10(0.91)
Auto.	Random	80.97(0.45)	69.57(0.23)	76.69(0.15)	90.20(0.27)	91.80(0.31)	91.74(0.46)	83.21(0.61)	71.54(0.23)	85.37(0.13)	55.01(0.34)	37.31(0.58)
	EA	81.27(0.59)	67.82(0.93)	76.33(0.70)	89.53(1.03)	91.38(0.51)	91.67(0.49)	82.16(1.17)	71.48(0.27)	85.64(0.25)	55.01(0.23)	36.73(1.48)
	RL	80.53(0.48)	69.69(0.51)	<b>77.33(0.62)</b>	90.30(0.44)	91.48(0.51)	91.75(0.31)	82.46(0.99)	71.22(0.24)	85.44(0.16)	55.01(0.13)	37.52(0.31)
LLM	GPT4GNAS	80.73(0.15)	69.43(0.07)	76.88(0.41)	<b>90.44(0.06)</b>	91.84(0.52)	91.79(0.43)	<b>83.54(0.53)</b>	71.31(0.27)	85.57(0.14)	55.12(0.16)	36.66(0.73)
-based	GHGNAS	80.66(0.42)	69.44(0.06)	76.93(0.22)	90.42(0.06)	91.59(0.56)	92.03(0.39)	82.93(0.63)	71.44(0.31)	85.43(0.15)	55.06(0.11)	37.01(0.57)
<b>Our</b>	<b>DesiGNN</b>	<b>81.69(0.52)</b>	<b>70.87(0.09)</b>	77.27(0.41)	90.41(0.39)	<b>92.61(0.00)</b>	<b>92.29(0.05)</b>	83.43(0.77)	<b>71.98(0.16)</b>	<b>85.85(0.29)</b>	<b>55.32(0.14)</b>	<b>37.57(0.62)</b>

Table 10: Best model performance (accuracy) after 30 model validations on the benchmark datasets.

Type	Model	ACC (STD) %										
		Cora	Citeseer	PubMed	CS	Physics	Photo	Computer	arXiv	DBLP	Flickr	Actor
Auto.	GNAS	81.35(0.23)	69.67(0.47)	76.97(0.22)	90.18(0.31)	91.92(0.43)	91.61(0.23)	83.23(1.13)	71.54(0.36)	85.58(0.08)	55.23(0.20)	37.23(0.56)
	Auto-GNN	81.46(0.41)	69.78(0.52)	77.37(0.51)	90.50(0.20)	91.92(0.65)	91.97(0.40)	83.06(0.91)	71.61(0.35)	85.66(0.17)	55.20(0.15)	37.50(0.83)
Auto.	Random	81.17(0.44)	69.63(0.16)	77.25(0.44)	90.39(0.14)	91.81(0.31)	92.04(0.32)	83.62(0.63)	71.55(0.22)	85.48(0.15)	55.17(0.12)	37.39(0.59)
	EA	81.27(0.59)	68.05(0.70)	76.51(0.83)	89.70(0.80)	91.60(0.65)	91.78(0.60)	83.00(0.84)	71.62(0.26)	85.66(0.24)	55.09(0.21)	37.44(1.86)
	RL	80.91(0.24)	69.88(0.24)	77.33(0.62)	90.47(0.19)	91.83(0.38)	91.75(0.31)	82.48(0.96)	71.47(0.08)	85.52(0.23)	55.05(0.13)	37.52(0.31)
LLM	GPT4GNAS	81.31(0.24)	69.43(0.07)	76.90(0.41)	90.44(0.06)	92.12(0.21)	92.21(0.13)	83.96(0.83)	71.67(0.44)	85.57(0.14)	55.12(0.16)	36.70(0.70)
-based	GHGNAS	81.39(0.07)	69.64(0.38)	76.93(0.22)	90.42(0.06)	92.06(0.17)	<b>92.38(0.02)</b>	83.28(0.43)	71.84(0.03)	85.47(0.11)	55.06(0.11)	37.01(0.57)
<b>Our</b>	<b>DesiGNN</b>	<b>81.77(0.40)</b>	<b>71.00(0.09)</b>	<b>77.57(0.29)</b>	<b>90.51(0.42)</b>	<b>92.61(0.00)</b>	<b>92.38(0.06)</b>	<b>84.08(0.66)</b>	<b>72.02(0.18)</b>	<b>85.89(0.21)</b>	<b>55.44(0.06)</b>	<b>37.57(0.62)</b>

999 their statistics are summarized in Table 5. The reason for excluding ogbn-proteins [20] in evaluation  
1000 is that NAS-Bench-Graph only recorded partial model performance mapping due to explosion and  
1001 out-of-memory errors, making it hard to conduct benchmarking comparisons.

## 1002 B.1.2 Training Hyperparameters

1003 The training hyperparameters used in different datasets are summarized in Table 6. For the nine  
1004 benchmark datasets, we use the same hyperparameters as the NAS-Bench-Graph [35] benchmark  
1005 to ensure fair comparisons. These hyperparameters were selected based on random searches on 30  
1006 anchor GNN architectures. The hyperparameters for the three additional datasets are recommended  
1007 by LLMs following the same procedure from Graph Understanding (Section 3.1) to Initial Model  
1008 Suggestion (Section 3.3), except the retrieved model architecture knowledge (Section 3.2) is replaced  
1009 by the tuned hyperparameters of the corresponding benchmark datasets [35]. All the hyperparameters  
1010 are fixed across all experiments for fairness. All experiments are conducted on a single NVIDIA RTX  
1011 3080 GPU and any GPU that could run a conventional GNN training can work. Notably, DesiGNN’s  
1012 initial model recommendation can be delivered without GPU.

## 1013 B.1.3 Module Hyperparameters

1014 The hyperparameters used in the DesiGNN framework are summarized as follows: the best empirical  
1015 number of *confident* graph properties  $N_f = 8$  (without descriptive inputs), the size of the knowledge  
1016 pool  $N_s = 3$  to balance tolerance with computation cost (consider benchmark datasets with Top-  
1017 3 similarities to the unseen dataset), the number of top models  $N_m = 30$  (only top-performing  
1018 model design examples), the number of candidates  $N_c = 30$ , and the maximum number of trials  
1019  $maxTrials = 30$  for fast deployment scenario. These hyperparameters are selected based on the  
1020 ablation studies in Section C.2, C.3, and C.4, which are then fixed across all main experiments.

Table 11: The main ablation study on each component in GNN Model Suggestion and Refinement. \* is our complete setting.

Method	ACC (STD) %							
	Cora	Citeseer	PubMed	CS	Physics	Photo	Computers	arXiv
<b>Initial Proposal Performance</b>								
<b>Property Only*</b>	<b>80.31 (0.00)</b>	69.20 (0.16)	<b>76.60 (0.00)</b>	<b>89.64 (0.08)</b>	<b>92.10 (0.00)</b>	<b>91.19 (0.00)</b>	<b>82.20(0.00)</b>	<b>71.50 (0.00)</b>
<b>Descriptonal Only</b>	<b>80.31 (0.00)</b>	<b>69.26 (0.00)</b>	75.71 (0.00)	89.53 (0.00)	88.42 (0.00)	91.17 (0.13)	81.79 (1.21)	71.20 (0.34)
<b>Both</b>	<b>80.31 (0.00)</b>	<b>69.26 (0.00)</b>	75.71 (0.00)	89.55 (0.03)	91.45 (0.78)	91.13 (0.07)	<b>82.20(0.00)</b>	<b>71.50 (0.00)</b>
<b>w/o Knowledge</b>	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)
<b>Best Performance After 30 Validations</b>								
<b>All*</b>	<b>81.77 (0.40)</b>	<b>71.00 (0.09)</b>	<b>77.57 (0.29)</b>	<b>90.51 (0.42)</b>	92.61 (0.00)	<b>92.38 (0.06)</b>	<b>84.08 (0.66)</b>	<b>72.02 (0.18)</b>
<b>w/o Re-rank</b>	<b>81.77 (0.40)</b>	69.77 (0.31)	77.40 (0.24)	90.47 (0.00)	92.61 (0.00)	<b>92.38 (0.06)</b>	83.90 (0.14)	71.87 (0.12)
<b>w/o Promotion</b>	81.04 (0.42)	70.33 (0.00)	77.00 (0.30)	90.19 (0.47)	<b>92.71 (0.18)</b>	92.22 (0.16)	82.85 (0.01)	71.99 (0.01)
<b>w/o Exploration</b>	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)
<b>w/o Knowledge</b>	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)

## B.2 Main Results

### B.2.1 Performance Ranking of Initial Model Proposals

In this section, Table 7 presents the performance ranking of initial model proposals within the model space defined by NAS-Bench-Graph [35]. The average performance ranking of initial model proposals across all benchmark datasets is Top-5.77%, indicating that the initial model proposals suggested by our *DesiGNN-Init without any prior training* are highly competitive.

### B.2.2 Model Refinement and Short-run Efficiency

In this section, we present the complete results of the short-run experiments in Table 8 (after 10 model validations), Table 9 (after 20 model validations), and Table 10 (after 30 model validations). The performance trajectories of all automated baselines are presented in Figure 8 and Figure 9.

## C Ablation Studies

To delve deeper into the pivotal designs of *DesiGNN*, we conducted ablation and hyperparameter studies on three key modules, including Graph Understanding (Section 3.1), Knowledge Retrieval (Section 3.2), and GNN Model Suggestion and Refinement (Section 3.3).

### C.1 Main Ablation Study

We present the main ablation study on the Knowledge Retrieval and the GNN Model Suggestion and Refinement module in Table 11, quantifying the impact of the retrieved knowledge and each mechanism in our framework. The results from the initial model suggestion stage underscore the importance of designing a proper Graph Understanding method, demonstrating that relying solely on filtered graph properties in dataset descriptions outperforms any use of descriptive inputs. In the short-run performance evaluation 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.

### C.2 Ablation Studies on Graph Understanding

In this section, we study the combined effect of the number of selected properties  $N_f$  and the usage of descriptive inputs in understanding and comparing graph datasets, under varying sizes  $N_s$  of the Knowledge Pool, which influences the number of initial proposals suggested by LLMs (Section 3.3). The average rankings of different combinations across datasets in Table 12 illustrate that while the optimal  $N_f$  varies with different  $N_s$  values, integrating descriptive inputs often diminishes

Table 12: Average ranks of different combinations across datasets. A higher rank (with 1 being the highest) corresponds to better performance and lower variability in the initial model proposals.

Top- $N_s$	Desc.	Combined Rank						
		g=0	g=2	g=4	g=6	g=8	g=10	g=16
$N_s=1$	True	2.250	2.312	2.250	2.125	2.437	2.625	2.312
	False	2.562	2.125	2.562	<b>1.812</b>	2.187	2.250	2.125
$N_s=2$	True	2.500	2.187	2.625	1.937	1.687	2.312	2.750
	False	2.625	2.375	2.812	2.000	<b>1.500</b>	2.187	2.250
$N_s=3$	True	2.500	1.937	2.875	1.875	1.937	2.187	2.000
	False	2.125	1.687	2.000	1.750	1.500	<b>1.3750</b>	1.687

Table 13: The distribution of the best initial model among the Top-3 benchmark.

Scenario	Is Best Initial Model (%)		
	Best	Second	Third
<b>Descriptive Only</b>	30.0	50.0	20.0
$N_f = 6$	50.0	22.5	27.5
$N_f = 8$	<b>52.5</b>	<b>37.5</b>	<b>10.0</b>
$N_f = 10$	42.5	40.0	17.5

Table 14: Average performance rank of transferring the Top- $N_m$  model designs.

Top- $N_m$	Average Rank					
	$N_m=1$	$N_m=10$	$N_m=20$	$N_m=30$	$N_m=40$	$N_m=50$
	2.7077	2.8750	2.8438	<b>2.6875</b>	2.9231	2.8769

performance, particularly when  $N_f > 6$  and  $N_s = 3$ . This finding supports our notion of “artificial hallucinations,” where descriptive inputs can hinder the understanding of tasks when sufficient graph properties are provided. From another aspect, this result also validates our idea of analyzing graph properties to design tailored GNN models, which came from insights documented in the literature.

### C.3 Ablation Studies on GNN Model Suggestion

This study extends from Section 3.1 and quantifies the advantage of using the two  $N_s = 2$  or three  $N_s = 3$  most similar benchmarks instead of solely the most similar one  $s = 1$  (no tolerance). Table 13 shows the distribution of the best initial proposal stems from the first, second, or third most similar benchmark. For  $N_f = 8$  without descriptive inputs, approximately 53% of trials identify the optimal initial proposal from the most similar benchmark, with only 10% benefiting from the third. This data supports a balanced trade-off between performance and efficiency. Notably, this study also serves as the motivation for us to design the Re-rank Mechanism in Model Proposal Refinement (Appendix A.4).

Moreover, we explore the optimal number  $N_m$  of top-performing examples from each benchmark when establishing the Knowledge Pool and assess the impact of using poor examples. Table 14 shows that  $N_m = 30$  yields the best average performance across all dataset combinations (2160 records), with further analysis indicating that poor examples do not significantly affect performance outcomes (T-statistic =  $-0.1262$ , P-value =  $0.8998$ ). We thus adopt  $N_m = 30$  without bad examples for an appropriate token length of the LLM prompt.

### C.4 Ablation Studies on GNN Model Refinement

We investigate the impact of using the  $\mathcal{K}_1$  Knowledge Pool as context for LLMs and the Re-rank Mechanism that is based on the performance ranking of initial proposals. We also study the best number of candidates and the effect of adopting different  $N_s$  numbers of knowledge bases to guide the controlled search space exploration. Table 15 displays T-statistics for various configurations of Knowledge Pool usage and Re-rank Mechanism. For the row of w/o candidate, the True/False

Table 15: T-statistic of the ablation on the usage of knowledge-driven Exploration and Re-rank Mechanism with different candidate number and  $N_s$ .

Candidate Number=		10	20		30		
w/o Re-rank or w/o Exploration							
Ablation	$N_s$	True	False	True	False	True	False
w/o Exploration	2	<u>-0.280</u>	0.582	0.512	1.977	0.557	0.566
	3	0.863	0.072	0.055	<u>-0.164</u>	<u>-0.054</u>	<u>-0.358</u>
w/o Re-rank	2	0.647	1.383	<u>-0.334</u>	1.232	<u>-2.092</u>	<u>-0.852</u>
	3	<u>-0.116</u>	<u>-0.808</u>	<u>-0.593</u>	<u>-0.591</u>	<u>-0.751</u>	<u>-1.904</u>

Table 16: Average performance rank of the GNN Model Refinement hyperparameter combinations across benchmark.

Candidate Number=		10		20		30	
w/o Re-rank							
$N_s$	w/o Exploration	True	False	True	False	True	False
2	True	7.33	12	7.83	9	13.5	9.17
	False	7.16	14.5	11.7	17.33	16.17	11.83
3	True	11.7	11.17	11.83	9.5	15.33	7.33
	False	12.33	10.17	11.17	10.17	12.83	7

columns stand for w/o re-rank, and vice versa. While statistical significance is not reached (p-value > 0.05), certain trends emerge:

- When  $N_s = 3$  knowledge bases are used for a broader perspective, employing a Re-rank Mechanism consistently adds value. This pattern is intuitive as the Re-rank Mechanism will become more influential as the number of knowledge bases increases.
- Broader exploration with  $N_s = 3$  and a larger number of candidates (e.g., 30) is likely effective, as we also have a directional exploitation mechanism to refine the diversity of potentially immature solutions.

Given the complexity of these patterns, we further assess the performance ranking of GNN Model Refinement hyperparameter combinations across benchmarks to determine the most effective settings. Table 16 suggests that the best hyperparameter combination is  $N_s = 3$  and  $N_m = 30$  with both Re-rank Mechanism and knowledge-driven refinement strategy enabled.

## D Case Studies

### D.1 Case Studies: Lack of Prior Knowledge in LLMs

The first study verifies the *inherent* knowledge gap illustrated in Figure 1a: lack of prior knowledge in LLMs about the top-performing models of benchmark datasets within the NAS-Bench-Graph [35] model space (Section A.3). The first two columns in Table 20 are cases where the descriptive inputs or graph properties are directly sent to 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 (left panel) or key graph properties (middle panel). In fact, we have tested all the benchmark datasets in NAS-Bench-Graph [35] with descriptive inputs, and we found only two macro lists and three operation lists that LLM would recommend regardless of the datasets:

- **Architecture:** [0, 1, 2, 3] and [0, 0, 1, 3].
- **Operations:** ['gcn', 'gat', 'sage', 'skip'], ['gcn', 'gat', 'sage', 'gcn'], and ['gcn', 'gat', 'gin', 'sage'],

After providing the tailored knowledge retrieved from our framework to LLMs (right panel), it can be clearly seen that the model it recommends is specialized and has obvious performance improvements.

Table 17: Performance comparison of initial model suggestion with different LLMs as meta-controller.

Dataset	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com.	arX.
<b>Llama2</b>	80.26	68.90	75.83	<b>89.84</b>	90.75	<b>91.35</b>	80.27	70.95
<b>GPT-4</b>	<b>80.31</b>	<b>69.20</b>	<b>76.60</b>	89.64	<b>92.10</b>	91.19	<b>82.20</b>	<b>71.50</b>

Table 18: Performance comparison of model refinement with different LLMs as meta-controller.

Dataset	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com.	arX.
<b>Llama2</b>	80.96	70.38	77.40	90.19	91.47	92.09	83.69	71.63
<b>GPT-4</b>	<b>81.77</b>	<b>71.00</b>	<b>77.57</b>	<b>90.51</b>	<b>92.61</b>	<b>92.38</b>	<b>84.08</b>	<b>72.02</b>

Table 19: Performance comparison of different prompt designs on GPT4GNAS.

Dataset	Cora	Cit.	Pub.	CS	Phy.	Pho.	Com.	arX.
<b>Original</b>	81.31	69.43	76.90	<b>90.44</b>	92.12	<b>92.21</b>	<b>83.96</b>	71.67
<b>Our Prompt</b>	<b>81.38</b>	<b>70.06</b>	<b>77.03</b>	90.22	<b>92.16</b>	92.08	83.78	<b>71.72</b>

## 1106 D.2 Case Studies: Artificial Hallucination in LLMs

1107 The second case study in Table 21 and Figure 1b examines the phenomenon of “artificial halluci-  
 1108 nation”, or the *external* noise issue, in LLMs when comparing the similarities between the unseen  
 1109 dataset and the benchmark datasets, as detailed in Section 3.2. Figure 5 (leftmost) illustrates that the  
 1110 empirically most similar benchmark datasets to PubMed are CS, Physics, and Citeseer, in descending  
 1111 order of similarity. When employing our Graph Understanding method in Section 3.1, which lever-  
 1112 ages *confident* graph properties (right panel), the top three most similar datasets are Citeseer, CS,  
 1113 and Physics (align well with empirical ground truth). However, the current practice that relies solely  
 1114 on descriptive inputs (left panel) identifies Cora, Citeseer, and ogbn-arxiv as the top three, which  
 1115 does not align with the empirical ground truth. This discrepancy arises because LLMs overly rely on  
 1116 the shared characteristic of being citation graphs, assuming that citation datasets like PubMed, Cora,  
 1117 Citeseer, and ogbn-arxiv should have similar model design preferences. This case study demonstrates  
 1118 that relying solely on *external* descriptive inputs is insufficient to capture the similarities between  
 1119 datasets, which can overwhelm other pertinent information, leading to inaccurate task similarity  
 1120 assessment and incorrect knowledge retrieval.

## 1121 D.3 Case Studies: Different LLMs

1122 To study the effect of different LLMs as the meta-controller, we adopt an open-source LLM, Llama2-  
 1123 13b, to replace the GPT-4 meta-controller we originally employed in our paper. Table 17 shows the  
 1124 comparative results (averaged over five runs) on the initial model proposal, while Table 18 shows the  
 1125 comparative results (averaged over five runs) after 30 rounds of model validations. We can see that  
 1126 replacing GPT-4 with Llama2 results in a slight degradation of the performance of recommended  
 1127 models in both settings. This degradation is more consistent after the refinement of the model design.  
 1128 This suggests that GPT-4 is superior at proposing a reasonable next step from the trajectory and extra  
 1129 knowledge (optimization tasks), consistent with the existing study [53]. Notably, because the initial  
 1130 model proposal phrase relies more on the quality of our elicited knowledge, Llama2 can also achieve  
 1131 close performance to GPT-4.

## 1132 D.4 Case Studies: Prompt Design Impact

1133 We studied whether fine-tuning in prompt engineering has a positive or negative effect on our work.  
 1134 Since we did not adopt any prompt fine-tuning approach, we added an alternative experiment to  
 1135 compare whether there is any consistent difference in the performance of LLM-based baseline  
 1136 GPT4GNAS [43] between using our prompt design and its original design. This includes adding  
 1137 topological graph descriptions and replacing model space descriptions and general instructions. No  
 1138 knowledge-related design from our method is being transferred because it is our key novelty. After  
 1139 prompt replacement, GHGNAS [12] became the same as GPT4GNAS [43], so we only reported one  
 1140 of them. Table 19 below shows the comparative results (averaged over five runs) after 30 rounds

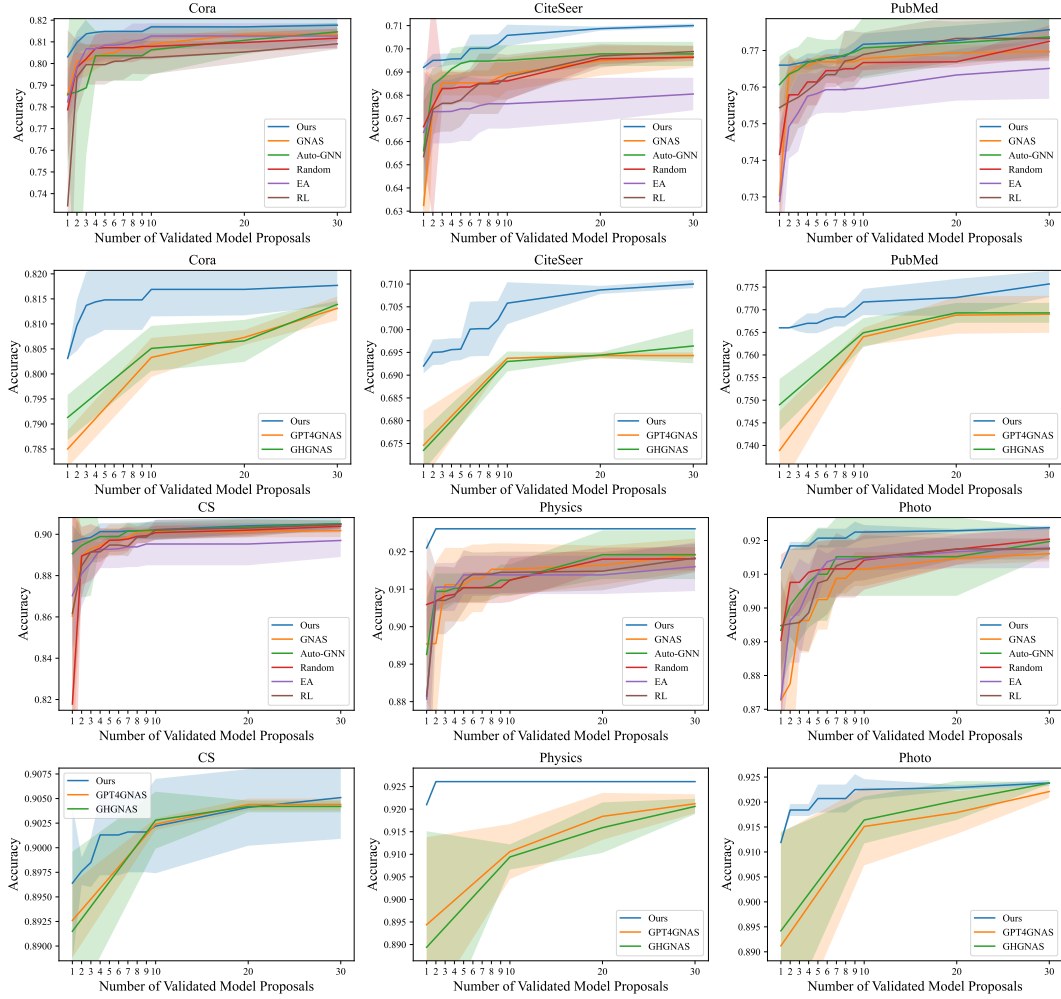


Figure 8: Short-run performance trajectories of DesiGNN compared to all the automated baselines after validating 1-30 proposals.

1141 of refinements. We can see that after replacing the prompt design of the baseline with our design,  
 1142 the impact on the quality of the recommended models is inconsistent. This study justifies that our  
 1143 method does not improve performance by fine-tuning the prompt; instead, the main source of our  
 1144 performance improvement arises from our tailored model design knowledge.

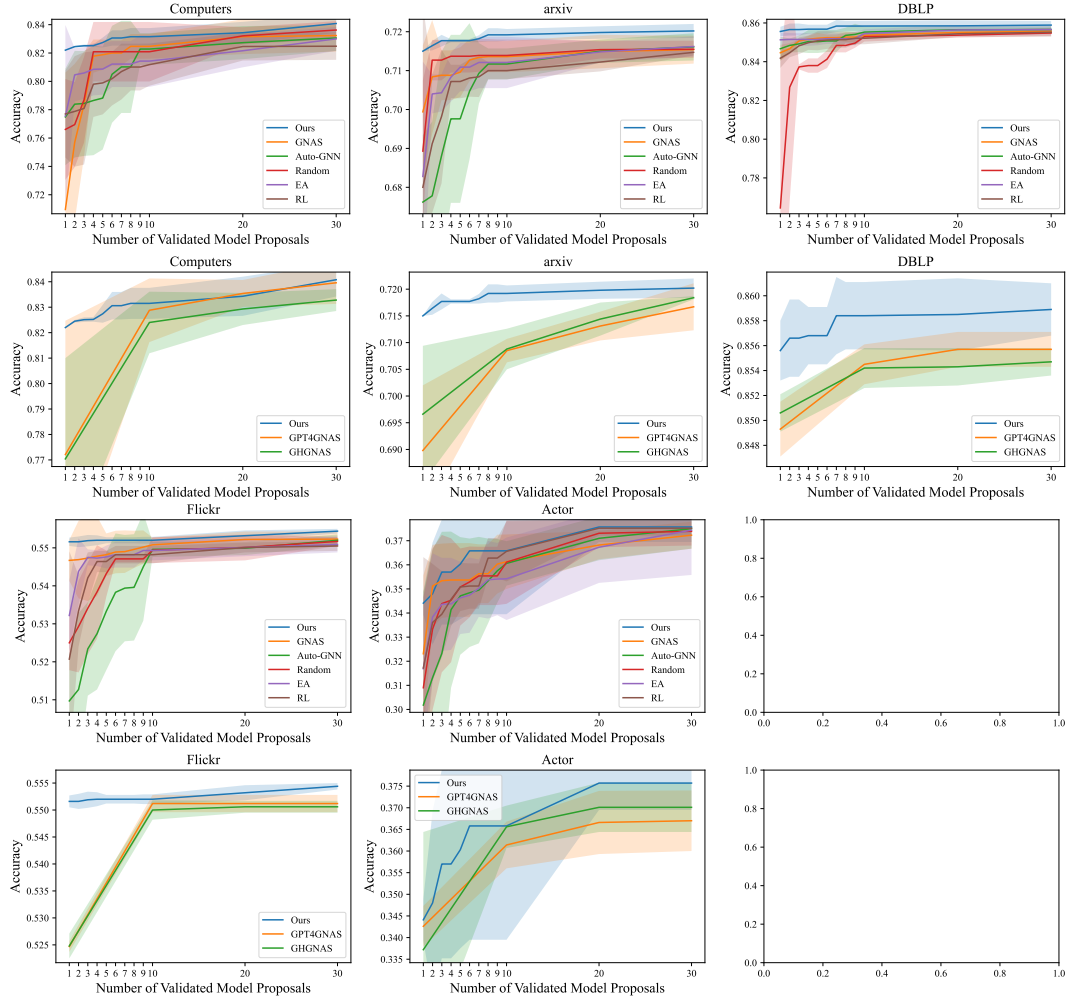


Figure 9: Short-run performance trajectories of DesiGNN compared to all the automated baselines after validating 1-30 proposals.

Table 20: Example LLM Response on Recommending Model for PubMed dataset

Section	Response (w/ Descriptive Only)	Response (w/ Properties Only)	Response (w/ Knowledge)
<b>Decision</b>	For the unseen dataset: (Architecture: [0, 0, 1, 3], Operations: ['gcn', 'gat', 'sage', 'gcn'])	For the unseen dataset: (Architecture: [0, 0, 1, 3], Operations: ['gcn', 'gat', 'sage', 'gin'])	For the unseen dataset: (Architecture: [0, 0, 0, 1], Operations: ['gcn', 'gat', 'arma', 'graph'])
<b>Reason</b>	<p><b>1. Architecture Choice ([0, 0, 1, 3]):</b> .....</p> <p><b>2. Operations Choice (['gcn', 'gat', 'sage', 'gcn']):</b></p> <p>- <b>First Operation - GCN ('gcn'):</b> The Graph Convolutional Network (GCN) is effective for aggregating neighborhood information, which is essential in a citation graph where the connectivity pattern (citations) plays a crucial role in defining the node (paper) features.</p> <p>- <b>Second Operation - GAT ('gat'):</b> The Graph Attention Network (GAT) introduces an attention mechanism, allowing the model to focus on the most relevant parts of the input graph structure. This is particularly useful in citation graphs for emphasizing influential papers.</p> <p>- <b>Third Operation - GraphSAGE ('sage'):</b> GraphSAGE is adept at sampling and aggregating features from a node's neighborhood, which can help in efficiently handling the local neighborhood structures and diversity in node degrees typical in citation graphs.</p> <p>- <b>Fourth Operation - GCN ('gcn'):</b> Another layer of GCN helps to further refine the feature representation by aggregating higher-order neighborhood information, which can be crucial for capturing the broader context in the graph.</p> <p>This combination and sequence of operations are designed to leverage both the local and global structural information in the graph, which is critical for a citation graph where both direct and indirect citations (connections) can influence the classification of a paper into topics. The use of both GCN and GAT helps in effectively capturing different levels of granularity in neighbor interactions, while GraphSAGE allows for efficient computation by focusing on a sampled subset of neighbors. The architecture's mix of parallel and sequential processing ensures a comprehensive feature transformation pathway, enhancing the model's ability to learn complex patterns in the data.</p>	<p><b>1. Architecture Choice ([0, 0, 1, 3]):</b> .....</p> <p><b>2. Operations Choice (['gcn', 'gat', 'sage', 'gin']):</b></p> <p>- <b>'gcn' (Graph Convolutional Network layer):</b> Effective for capturing neighborhood information through a simple yet powerful convolutional approach. Given the average degree and edge count, 'gcn' can efficiently aggregate local neighborhood features without being too computationally expensive.</p> <p>- <b>'gat' (Graph Attention Network layer):</b> Utilizes attention mechanisms to weigh the importance of nodes' neighbors. This is particularly useful in graphs where nodes vary significantly in their roles and influence, as indicated by the centrality measures.</p> <p>- <b>'sage' (GraphSAGE):</b> Samples and aggregates features from a node's neighborhood, which can handle the graph's density and scale effectively. This method is adaptive and can learn from different-sized neighborhoods, which is beneficial given the variability in node connectivity.</p> <p>- <b>'gin' (Graph Isomorphism Network layer):</b> Designed to capture more nuanced graph structural information, which can be crucial for understanding deeper patterns in the graph, especially given the non-trivial graph diameter and clustering co-efficient.</p> <p>This combination and sequence of operations are designed to first broadly and then selectively refine the feature representation of the nodes, considering both local and global structural properties of the graph. This should provide a robust model capable of handling the complexities and variabilities in the graph metrics provided.</p>	<p><b>1. Architecture Choice:</b> .....</p> <p><b>2. Operation Selection:</b> .....</p> <p>- <b>'arma' (Auto-Regressive Moving Average):</b> ARMA layers have been chosen in various top models, suggesting their utility in capturing more complex dependencies in graph data than standard convolutional layers.</p> <p>- <b>'graph' (k-GNN):</b> The inclusion of a k-GNN operation in the final step can extend the model's capability to capture higher-order interactions, which might be crucial for the specific characteristics of the unseen dataset.</p> <p><b>3. Combination of Operations:</b> The combination of GCN, GAT, and ARMA before a final GRAPH operation provides a robust sequence of transformations. Starting with GCN allows for a broad capture of neighborhood information, followed by the selective attention mechanism of GAT and the complex filtering of ARMA, culminating in a k-GNN layer that can capture higher-order dependencies. This sequence is designed to progressively refine and enhance the feature representations.</p> <p><b>4. General Observations from Top Models:</b> The selected operations and architecture are among the most recurrent in the top-performing models from similar datasets. This suggests that they are likely to be effective in handling the types of graph-structured data encountered in these benchmarks, making them a prudent choice for the unseen dataset.</p> <p>By synthesizing these observations, the recommended model design aims to balance complexity and depth of feature extraction and transformation, which is crucial for effectively processing graph-structured data. This design is expected to perform well on the unseen dataset by leveraging proven elements from similar successful models.</p>
<b>Perf.</b>	75.40	71.56	<b>76.60</b>

Table 21: Example LLM Response on Dataset Comparison for PubMed dataset

Benchmark	w/ Descriptive Inputs	w/ Confident Graph Properties
<b>Cora</b>	<b>Similarity Score:</b> 0.8 <b>Reason:</b> Both the unseen dataset and Cora are citation graphs with papers as vertices and citations as links. Both use bag-of-words features and have ground-truth topics as labels. The main differences are in the number of features and classes, with Cora having more features and classes.	<b>Similarity Score:</b> 0.1 <b>Reason:</b> The unseen dataset has a significantly higher graph diameter, average closeness centrality, and average betweenness centrality compared to Cora. Additionally, the general graph metrics such as edge count, average degree, and density are quite different, indicating a low similarity.
<b>Citeseer</b>	<b>Similarity Score:</b> 0.75 <b>Reason:</b> Similar to Cora, Citeseer is also a citation graph with similar characteristics to the unseen dataset. The differences are more pronounced in the number of features and classes, with Citeseer having significantly more features.	<b>Similarity Score:</b> 0.6 <b>Reason:</b> Citeseer has a closer match in terms of graph diameter and average closeness centrality with the unseen dataset. The average betweenness centrality and general graph metrics like edge count and average degree also show some similarities, suggesting a moderate level of task similarity.
<b>CS</b>	<b>Similarity Score:</b> 0.2 <b>Reason:</b> CS is a co-authorship graph, differing significantly in the type of vertices (authors vs. papers) and the nature of features (keywords vs. bag-of-words). The task and evaluation metric are similar, but the structural differences lower the similarity score.	<b>Similarity Score:</b> 0.3 <b>Reason:</b> CS shares a similar graph diameter with the unseen dataset but differs significantly in average closeness centrality, average betweenness centrality, and general graph metrics such as edge count and average degree, indicating a moderate to low similarity.
<b>Physics</b>	<b>Similarity Score:</b> 0.2 <b>Reason:</b> Similar to CS, Physics is a co-authorship graph with authors as vertices and co-author relationships as links, which is structurally different from the unseen dataset's citation graph nature.	<b>Similarity Score:</b> 0.2 <b>Reason:</b> Physics has a similar graph diameter but differs significantly in average closeness centrality, average betweenness centrality, and other general graph metrics like edge count and average degree, suggesting a low similarity.
<b>Photo</b>	<b>Similarity Score:</b> 0.3 <b>Reason:</b> Photo is a co-purchase graph, which is different in nature from a citation graph. However, both use bag-of-words features and have a similar number of features, slightly increasing the similarity score.	<b>Similarity Score:</b> 0.1 <b>Reason:</b> Photo, while having a similar graph diameter, differs significantly in average closeness centrality, average betweenness centrality, and general graph metrics such as edge count and average degree, indicating a low similarity.
<b>Computers</b>	<b>Similarity Score:</b> 0.3 <b>Reason:</b> Similar to Photo, Computers is also a co-purchase graph. The similarity is slightly higher due to a closer number of features and classes compared to the unseen dataset.	<b>Similarity Score:</b> 0.1 <b>Reason:</b> Computers has a similar graph diameter but differs significantly in average closeness centrality, average betweenness centrality, and general graph metrics like edge count and average degree, indicating a low similarity.
<b>ogbn-arxiv</b>	<b>Similarity Score:</b> 0.7 <b>Reason:</b> ogbn-arxiv is a citation graph similar to the unseen dataset, but it uses word embeddings for features instead of bag-of-words, and has a significantly different number of classes.	<b>Similarity Score:</b> 0.2 <b>Reason:</b> ogbn-arxiv shares a similar graph diameter but differs significantly in average closeness centrality, average betweenness centrality, and general graph metrics such as edge count and average degree, suggesting a low similarity.
<b>ogbn-proteins</b>	<b>Similarity Score:</b> -0.5 <b>Reason:</b> ogbn-proteins is structurally and fundamentally different from the unseen dataset, being a protein association graph with a multi-label binary classification task and ROC-AUC as the evaluation metric.	<b>Similarity Score:</b> 0.1 <b>Reason:</b> ogbn-proteins has a different graph diameter and significantly different average closeness centrality, average betweenness centrality, and general graph metrics such as edge count and average degree, indicating a very low similarity.