GNN-CB: A Graph Neural Network Competition Benchmark for Human and LLM Evaluation *

Anonymous submission

Abstract

Large language models (LLMs) have demonstrated strong performance on coding and reasoning benchmarks; however, their ability to solve graph-structured machine learning problems remains largely unexplored. In particular, no benchmark currently evaluates whether LLMs can autonomously solve end-to-end Graph Neural Network (GNN) coding tasks under realistic competition settings. To address this gap, this paper introduces **GNN-CB**, the first competition-based benchmark for evaluating both humans and LLMs on GNN coding tasks. GNN-CB consists of 18 curated competitions spanning node-, edge-, and graph-level prediction across diverse graph categories, domains, and difficulty tiers. All submissions are evaluated through a unified automated pipeline with hidden test sets and standardized scoring. Human participants solve tasks under controlled competition constraints, while LLMs are evaluated using a frozen zero-shot prompting protocol based on a plan-then-code paradigm with bounded execute-and-repair loops. The benchmark additionally supports both non-agent and autonomous agent-based evaluation under identical conditions. The experimental results show that LLMs rarely match Human Top performance and exhibit larger gaps and less stable performance across competitions. No single model dominates: a few competitions are won by LLMs, yet humans still hold the top score on most tasks. We release GNN-CB as a living benchmark with automated evaluation infrastructure, dynamic leaderboards, and reproducible execution pipelines. Beyond benchmarking, GNN-CB also serves as an educational corpus for progressive learning of GNNs. The benchmark and evaluation framework are publicly available at here.

*To preserve review anonymity, the repository and leaderboard are temporarily disabled and will be released with full code and activated links upon acceptance.

1 Introduction

Large language models (LLMs) have recently demonstrated strong capabilities in coding and reasoning tasks, including program synthesis, competitive programming, software engineering, and autonomous code repair (Chen et al., 2021; Li et al., 2022; Jimenez et al., 2023; Zhuo et al., 2024). However, existing evaluations mainly focus on general-purpose programming while largely overlooking graph-structured machine learning workflows.

Graph Neural Networks (GNNs) are widely used for learning over graph-structured data in domains such as molecular modeling, recommender systems, social networks, neuroimaging, and knowledge graphs (Gilmer et al., 2017; Hamilton et al., 2017; Li et al., 2018; Ying et al., 2018; Li et al., 2021; Dauparas et al., 2022; Schlichtkrull et al., 2018). Unlike conventional coding tasks, GNN problems require understanding graph topology, designing relational learning pipelines, selecting suitable message-passing architectures, and optimizing the training under task-specific constraints. These factors make them substantially more challenging than isolated algorithmic coding tasks.

Despite the growing importance of graph machine learning, no existing benchmark evaluates whether LLMs can autonomously solve realistic end-to-end GNN coding tasks. Benchmarks such as HumanEval (Chen et al., 2021), CodeContests (Li et al., 2022), SWE-bench (Jimenez et al., 2023), and BigCodeBench (Zhuo et al., 2024) focus on algorithmic reasoning or software engineering workflows without graph-learning components, while graph benchmarks such as OGB (Hu et al., 2020) and LRGB (Dwivedi et al., 2022) are designed to evaluate the predictive performance of GNN models rather than their implementation.

To address this gap, we introduce **GNN-CB**, the first competition-based benchmark designed to evaluate both humans and LLMs on GNN coding

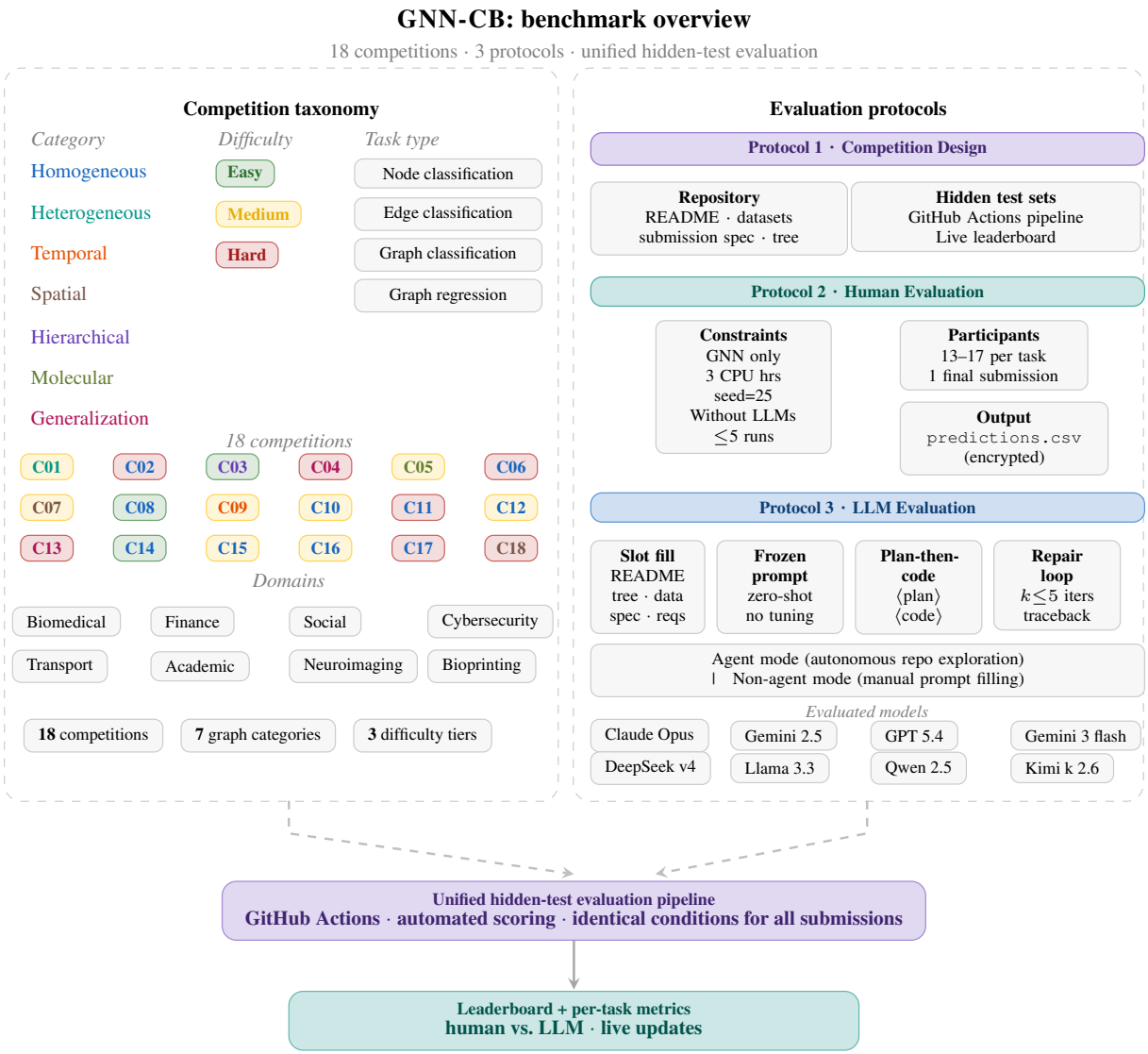


Figure 1: Overview of GNN-CB.

tasks. As illustrated in Figure 1, GNN-CB contains 18 *human-curated* competitions spanning node-, edge-, and graph-level prediction tasks across multiple graph categories, domains, and difficulty tiers. Each competition provides standardized repositories, datasets, hidden test sets, and automated GitHub-based evaluation pipelines.

To ensure fair and reproducible assessment, we define three complementary protocols: (i) competition design, (ii) human evaluation, and (iii) LLM evaluation. Human participants solve tasks under controlled competition constraints with fixed computational budgets and limited submissions. LLMs are evaluated using a frozen zero-shot prompting protocol based on a structured plan-then-code paradigm with bounded execute-and-repair loops. The benchmark additionally supports both non-agent and autonomous agent-based evaluation settings while maintaining identical evaluation conditions.

Our contributions are summarized as follows. (i) **GNN-CB** is introduced as the first competition-based benchmark for evaluating both humans and LLMs on GNN coding tasks. (ii) The benchmark contains 18 human-curated competitions spanning diverse graph categories, application domains, and difficulty tiers under unified hidden-test evaluation protocols. (iii) A controlled LLM evaluation framework is proposed based on frozen prompting, plan-then-code generation, and bounded execute-and-repair loops, supporting both agent-based and non-agent settings under identical constraints. (iv) Evaluation of human participants and LLMs enables systematic analysis of performance variability, task difficulty, and implementation challenges in realistic GNN coding environments. (v) GNN-CB is

released as a living benchmark with automated evaluation infrastructure, reproducible pipelines, and dynamic leaderboards to support future research on the capabilities and limitations of LLMs as autonomous GNN engineers, as well as serving as an educational corpus for progressive learning of GNNs.

2 Related Work

Recent benchmarks have significantly advanced evaluation of LLM-based code generation and reasoning. HumanEval (Chen et al., 2021) and CodeContests (Li et al., 2022) introduced standardized evaluation for functional correctness and competitive programming. Later works extended evaluation across programming languages and stricter testing settings, including HumanEval-X (Peng et al., 2024), MultiPL-E (Cassano et al., 2023), and EvalPlus (Liu et al., 2023). More realistic software engineering benchmarks such as DS-1000 (Lai et al., 2023), SWE-bench (Jimenez et al., 2023), BigCodeBench (Zhuo et al., 2024), and LiveCodeBench (Jain et al., 2024) evaluate repository-level reasoning, data science workflows, and instruction-following capabilities.

Graph machine learning benchmarks have standardized evaluation for graph neural networks (GNNs). The Open Graph Benchmark (OGB) (Hu et al., 2020) provides datasets and evaluation protocols for node-, edge-, and graph-level prediction tasks, while Benchmarking GNNs (Dwivedi et al., 2023) and LRGB (Dwivedi et al., 2022) focus on structural generalization and long-range graph dependencies. However, these benchmarks do not assess whether LLMs can perform at a human level on GNN competitions.

GNN-CB bridges these two research directions by introducing a competition-based benchmark for GNN coding tasks, supporting unified evaluation of both human participants and LLMs under controlled and reproducible conditions.

3 Benchmark Construction

GNN-CB is a competition-based benchmark for evaluating GNN coding tasks under standardized settings. Unlike traditional code-generation benchmarks centered on isolated functions or short algorithmic problems, GNN-CB targets repository-level graph learning workflows where participants or LLMs must build complete pipelines spanning data processing, graph construction, architecture design, optimization, and hidden-test prediction generation.

The benchmark construction consists of three protocols : (i) competition design protocol, (ii) human evaluation protocol, and (iii) LLM evaluation protocol.

3.1 Competition Design Protocol

The competition design protocol organizes benchmark tasks across diverse graph learning settings. The benchmark contains 18 independent competitions spanning node classification, edge prediction, graph classification, and graph regression tasks (Fig 1). Competitions are organized along two axes: graph category and difficulty tier. The benchmark covers seven graph categories, including homogeneous, heterogeneous, temporal, spatial, hierarchical, molecular, and generalization graphs. Competitions are additionally grouped into Easy, Medium and Hard difficulty tiers based on implementation complexity, architectural requirements, and observed human performance. Table 1 summarizes all benchmark competitions, including graph categories, dataset scales, feature dimensionalities, application domains, and evaluation metrics. Detailed dataset specifications and repository descriptions are provided in Appendix A.

Each competition is associated with a dedicated repository that specifies the task, submission format, evaluation metrics, and leaderboard configuration. An automated evaluation infrastructure is provided for submission validation, scoring, leaderboard updates, and execution logging, ensuring that all submissions are processed through a unified evaluation workflow across competitions.

All submissions are evaluated exclusively on hidden test sets that remain inaccessible throughout the competition lifecycle. Evaluation is performed through automated GitHub-based pipelines that validate submission formats, execute scoring scripts, and update leaderboards without manual intervention.

The infrastructure additionally supports secure submission mechanisms to prevent information leakage. Depending on repository configuration, participants either submit predictions through controlled external platforms or upload encrypted prediction files to designated repository directories, where they are automatically decrypted only during evaluation prior to scoring.

Table 1: Overview of all 18 GNN Coding Benchmark competitions. **Cat.** = Graph Category; **Diff.** = Difficulty; **N** = typical number of nodes; **E** = typical number of edges; **Feat.** = node feature dimensionality.

#	Competition	Task Description	Cat.	Diff.	N	E	Feat.	Domain	Metric	Dataset
C01	PROVEN-GNN	Graph classification on source code representations	Hetero	Med	260K	1.7M	527	Cybersecurity	Macro-F1	DiverseVul
C02	CGCC	Graph classification of city street-network layouts	Homo	Hard	9.396K	13.533K	3	Urban	Macro-F1	OpenStreetMap
C03	Real Or Fake!	Binary graph classification for fake news detection	Hier	Easy	314K	308K	1078	Social	Binary F1	GossipCop
C04	GLIMPS-GNN	Inductive cfRNA → preeclampsia classification	Gen	Hard	320	3200	6650	Biomedical	F1	GEO
C05	GNN_BACE	Molecular graph classification for BACE-1 inhibition	Mol	Med	32	34	8	Medical	Macro-F1	BACE (MoleculeNet)
C06	DiaGraph	Binary node classification for diabetes prediction	Homo	Hard	96K	1.2M	12	Medical	Macro-F1	Kaggle Diabetes
C07	GRAPE	Binary graph classification for diabetic retinopathy	Spatial	Med	252	846	4	Medical	Macro-F1	DRIVE/STARE/HRF
C08	Liar Nodes	Node Classification under Adversarial Feature Corruption for Cancer Single-Cell Analysis	Homo	Easy	3.7K	27K	1.7K	Biomedical	Accuracy	CancerSCEM
C09	GraphFMD	Temporal inductive node classification (Bitcoin)	Temp	Med	25.5K	—	165	Finance	Macro-F1	Elliptic
C10	NetLinkArena	Link prediction in sparse citation graphs	Homo	Med	3.3K	9.1K	2.7K	Academic	AUC	Citation
C11	PARK-GNN	Node classification for Parkinson's detection	Homo	Hard	195	∼1K	22	Medical	Macro-F1	UCI Parkinson's
C12	GNN-CoRA	Node classification on citation network	Homo	Med	2.7K	5.4K	1433	Academic	Accuracy	Cora
C13	GTA	Graph classification on ideal and perturbed conditions	Gen	Hard	∼18	∼ 40	7	Bioinformatics	Macro-F1	MUTAG
C14	BrainAge	Graph regression for age prediction from brain connectome	Homo	Easy	29.3K	478.8K	2	Neuroimaging	MAE	OASIS-3
C15	OctoNode Cup	Node classification on GitHub social network	Homo	Med	37.7K	289K	42	Social	Macro-F1	GitHub Social
C16	Graph4ASD	Binary classification of brain connectivity graphs	Homo	Med	127.4K	12676.3K	1	Neuroimaging	Macro-F1	ABIDE
C17	Mix2Print	Learning material interactions for 3D bioprinting	Homo	Hard	423	740	442	Bioprinting	Normalized-MAE	CECT
C18	THInC	Inductive GNN for cell type classification	Spatial	Hard	51K	258K	4	Biomedical	Macro-F1	NuCLS

3.2 Human Evaluation Protocol

The human evaluation protocol measures participant performance under standardized computational and submission constraints. Participants solve each competition independently using only GNN models, without external solutions, and submit a single final prediction file evaluated through the same automated hidden-test pipeline used for LLMs. LLM usage during the human track is restricted to low-level utility support without exposure to task-specific benchmark information. More detailed protocol and participant information are provided in Appendix B. Participants represent 18 universities across 12 countries (Figure 6, Appendix B), ensuring geographic diversity in benchmark design and human performance evaluation.

3.3 LLM Evaluation Protocol

GNN-CB evaluates LLMs under a controlled and reproducible protocol designed to measure repository-grounded graph learning ability rather than prompt-engineering performance. All models are evaluated using a frozen prompt template shared across all competitions, without task-specific tuning or adaptive prompting (Zhuo et al., 2024). As illustrated in Figure 2, repository information is converted into a standardized prompt through slot filling (Jain et al., 2024), after which the model generates a structured reasoning plan and an executable code following a plan-then-code paradigm (Jiang et al., 2024). The generated solution is executed locally in a controlled environment and optionally refined through bounded execute-and-repair iterations (Yang et al., 2024). Final predictions are evaluated on hidden test sets using the same automated infrastructure used for human participants, enabling direct and fair comparison under identical conditions. The protocol consists of four components: (i) frozen prompt instantiation, (ii) plan-then-code generation, (iii) execution and repair, and (iv) evaluation settings.

3.3.1 Frozen Prompt Instantiation

All models were evaluated using a standardized prompt template with limited, controlled task-specific context added to ensure executable outputs. The template contains predefined slots populated with repository-derived information such as

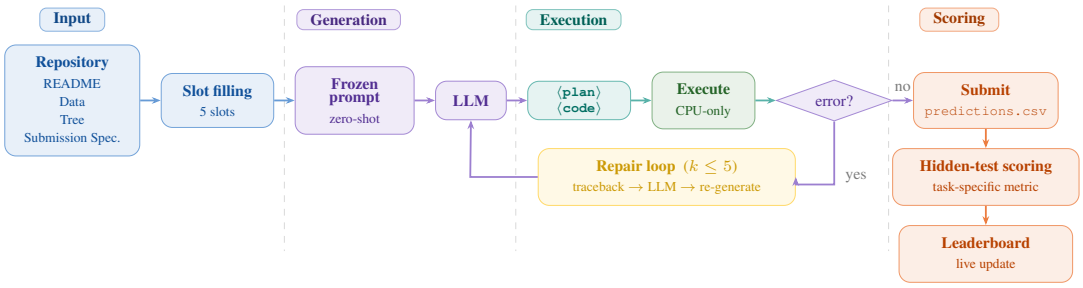


Figure 2: LLM evaluation pipeline.

the README, repository tree, dataset summary, submission specification, and dependency requirements.

Repository components are inserted into structured slots without manual modification (Jain et al., 2024). The instantiated prompt requires the model to produce a structured `<plan>` followed by executable `<code>`, providing a standardized interface across all competitions. Figure 8 in Appendix C illustrates the slot-filling construction, and the full template is provided in Listing 1.

3.3.2 Plan-Then-Code Generation

Given the instantiated prompt, the model generates: (i) a high-level reasoning plan and (ii) an executable code. This plan-then-code strategy (Jiang et al., 2024) encourages explicit reasoning before implementation and reduces common failures such as invalid pipelines, hallucinated APIs, or malformed submissions. The generated plan typically specifies feature engineering, GNN architecture design, optimization procedures, validation strategy, and submission generation. The accompanying code implements the complete pipeline as a standalone executable solution. A representative plan-then-code output is shown in Figure 9 in Appendix C.

3.3.3 Execution and Repair Loop

Generated code is executed locally in a controlled environment under CPU-only constraints. If execution fails due to syntax errors, runtime exceptions, dependency issues, or invalid outputs, an execute-and-repair loop (Yang et al., 2024) is triggered in which the execution traceback is appended to the conversation context and returned to the model for refinement. Repair attempts are bounded ($k \leq 5$) to maintain a controlled evaluation budget.

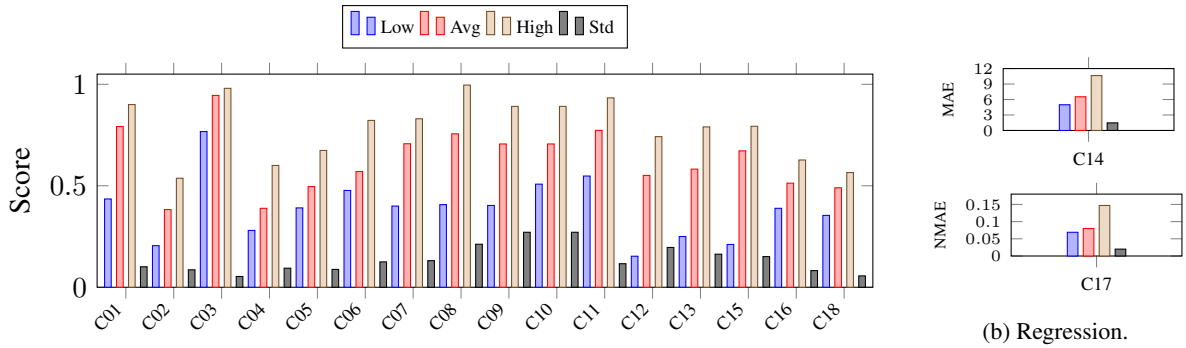
3.3.4 Agent-Based and Non-Agent Evaluation

GNN-CB supports two complementary execution modes: *non-agent* and *agent-based*. In the *non-agent* mode, a lightweight orchestration script instantiates the frozen prompt with repository-derived slots, dispatches the API request, parses the response into `<plan>` and `<code>` blocks, executes the resulting `solution.py` under the CPU-only environment, and feeds runtime tracebacks back into the prompt for a bounded number of repair iterations; a simplified version of this control loop is reproduced in Listing 2 of Appendix C. In the *agent-based* mode, an autonomous coding system inspects repository contents, generates an executable solution, runs it, and iteratively repairs failures without human intervention, using the same execute-and-repair budget. Both modes consume identical repository information, enforce the same execution constraints, and submit through the same automated hidden-test pipeline, so that observed performance differences reflect the model's reasoning and implementation ability rather than prompt-engineering effort or orchestration design.

3.3.5 LLM Implementations

To ensure deterministic, reproducible outputs, all models are decoded with `temperature = 0`, applied where the provider's API exposes this setting. API-only (non-agent) runs use a per-call output budget of `max_output_tokens = 10K`, while agent-mode runs follow each agent's native session budget, which may consume substantially more tokens across multiple internal tool calls and extended-thinking traces. A full inventory of evaluated models, their API identifiers, and source links is provided in Table 4 (Appendix C).

In the API-only mode, models are accessed through provider SDKs or OpenAI-compatible endpoints, including Together AI serverless inference, and receive the frozen prompt through a lightweight orchestration script that handles slot filling, code extraction, execution, and bounded repair iterations. Models evaluated in this mode include Gemini-3 Flash, Gemini 2.5 Pro, Llama-3.3 70B, Claude Opus 4.6, Qwen2.5-Coder 32B, Kimi



(a) Classification competitions (F1 / Accuracy / AUC).

(b) Regression.

Figure 3: Per-task human-performance statistics across all 18 GNN-CB competitions: minimum, average, maximum, and standard deviation of participant scores. Panel (a) covers the 16 classification competitions on a common 0–1 axis (F1 / Accuracy / AUC); panel (b) covers the two regression competitions (C14 BrainAge, MAE; C17 Mix2Print, NMAE) on independent axes due to different metric scales. Higher is better in (a); lower is better in (b).

k2.6, and DeepSeek v4 Pro.

In the agent-based mode, models operate as autonomous coding agents that directly inspect repository contents, generate solutions, execute code, and iteratively repair failures without human intervention. Models evaluated in this mode include Claude Opus 4.7 and GPT-5.4 Codex.

All generated solutions are executed locally under CPU-only constraints inside isolated environments. The framework automatically parses model outputs into `<plan>` and `<code>` sections, executes the generated `solution.py`, performs bounded repair iterations ($k \leq 5$), validates prediction files, and logs execution artifacts including runtime traces and generated submissions. This design enables reproducible evaluation across multiple model families without requiring local GPU deployment.

4 Results and Analysis

We evaluate GNN-CB along two complementary tracks: a human-participant track (Section 3.2) in which expert volunteers solve each competition under standardized reproducibility constraints, and an LLM track (Section 3.3) in which nine leading proprietary and open-source models are evaluated under a frozen plan-then-code prompting protocol with bounded execute-and-repair iterations. Both tracks submit through the same automated hidden-test pipeline and are scored with task-specific metrics: macro-F_1, accuracy and, AUC (Area Under Curve) for classification tasks, and (normalized) mean absolute error (NMAE/MAE) for regression tasks. The first subsection presents per-task descriptive statistics over the human cohort (mean, standard deviation, best, and worst submission); the second reports each LLM's per-competition score together with its gap to the human top, and quantifies each model's cross-task variability to assess consistency alongside raw performance.

Table 2: Descriptive statistics of human performance across competitions. Difficulty level (Diff), number of participants (P), average score (AVG), and standard deviation (STD) summarize overall human performance.High and Low columns report the maximum and minimum human scores observed, respectively, Metric indicates the evaluation criterion used for tasks.

#	Competition	Diff.	P	AVG	STD	High	Low	Metric
C01	PROVEN-GNN	Med	17	0.792	0.101	0.900	0.435	F1
C02	CGCC	Hard	17	0.383	0.086	0.537	0.205	F1
C03	Real Or Fake!	Easy	15	0.945	0.053	0.980	0.767	F1
C04	GLIMPS-GNN	Hard	16	0.389	0.094	0.600	0.28	F1
C05	GNN_BACE	Med	14	0.496	0.088	0.674	0.391	F1
C06	DiaGraph	Hard	15	0.570	0.125	0.822	0.477	F1
C07	GRAPE	Med	16	0.707	0.131	0.830	0.4	F1
C08	Liar Nodes	Easy	17	0.756	0.212	0.996	0.407	Acc.
C09	GraphFMD	Med	17	0.706	0.271	0.891	0.403	F1
C10	NetLinkArena	Med	17	0.706	0.271	0.891	0.508	AUC
C11	PARK-GNN	Hard	14	0.773	0.116	0.933	0.548	F1
C12	GNN-CoRA	Med	14	0.551	0.196	0.742	0.153	Acc.
C13	GTA	Hard	17	0.5816	0.1629	0.79	0.25	F1
C14	BrainAge	Easy	14	6.531	1.474	10.638	4.984	MAE
C15	OctoNode Cup	Med	15	0.672	0.151	0.793	0.211	F1
C16	Graph4ASD	Med	14	0.513	0.082	0.627	0.389	F1
C17	Mix2Print	Hard	13	0.08	0.02	0.147	0.069	NMAE
C18	THInC	Hard	14	0.490	0.056	0.565	0.354	F1

4.1 Human Performance Results

Table 2 and Figure 3 together summarize the performance of human participants across all 18 competitions. Overall, human performance exhibits substantial variability across tasks, reflecting differences in task difficulty, graph structure, and domain complexity. Easy classification tasks (e.g., C03) achieve high scores with low variance, indicating

Table 3: **LLM Benchmark Leaderboard on GNN Competitions.** Scores and gaps to Human Top performance across 18 GNN competitions for 9 LLMs are reported. Gap is defined as score − human top ; sign-corrected for regression tasks. Blue subscripts ↓ indicate the absolute drop in score below the human top. Green subscripts ↑ indicate the LLM matches or exceeds the human top. For classification tasks, cells are tinted red (below human, gap $\geq 5\%$), light red (below human, gap $< 5\%$), yellow (tied) and green (above human). For regression tasks, cells are colored as red (below human) and green (above human). Underlining denotes best result within each model category; **boldface** highlights best overall performance per competition. The bottom row, *Avg. human–LLM gap (%)*, reports the mean drop of the nine LLMs below the human top for each competition.

Models	Easy			Medium									Hard					
	C03	C08	C14	C01	C05	C07	C09	C10	C11	C12	C15	C16	C02	C04	C06	C13	C17	C18
	Real_Or_Fake	*Liar Nodes*	*Brain-Age*	*PROVEN*	*GNN_BACE*	*GRAPE*	*Graph FMD*	*NetLinkArena*	*PARK-GNN*	*GNN_CoRA*	*OctoNode*	*Graph4ASD*	*CGCC*	*GLIMPS-GNN*	*DiaGraph*	*GTA*	*Mix2Print*	*THInC*
	val F1	*acc.*	*MAE*	*mF1*	*F1*	*mF1*	*val F1*	*AUC*	*val F1*	*acc.*	*Mac-F1*	*Mac-F1*	*OOF F1*	*F1*	*Mac-F1*	*F1 (P.)*	*NMAE*	*Mac-F1*
Proprietary Reasoning LLMs (Frontier)																		
Gemini 2.5 Pro (Google DeepMind, 2025)	93.6 $_{\downarrow 4.5}$	92.7 $_{\downarrow 6.9}$	5.524 $_{\downarrow 0.540}$	75.5 $_{\downarrow 14.5}$	**61.9** $_{\downarrow 5.5}$	**83.0** $_{0}$	83.6 $_{\uparrow 3.0}$	**87.5** $_{\downarrow 1.6}$	71.2 $_{\downarrow 22.1}$	64.8 $_{\downarrow 9.4}$	54.1 $_{\downarrow 25.2}$	48.8 $_{\downarrow 13.9}$	40.7 $_{\downarrow 13.0}$	29.0 $_{\downarrow 31.0}$	48.9 $_{\downarrow 33.4}$	65.7 $_{\downarrow 14.2}$	0.093 $_{\downarrow 0.025}$	36.0 $_{\downarrow 20.6}$
Claude Opus 4.7 (Anthropic, 2026b)	**97.6** $_{\downarrow 0.5}$	**99.8** $_{\uparrow 0.2}$	**5.504** $_{\downarrow 0.520}$	78.1 $_{\downarrow 11.9}$	55.4 $_{\downarrow 12.0}$	83.0 $_{0}$	76.2 $_{\downarrow 4.4}$	69.6 $_{\downarrow 19.5}$	90.2 $_{\downarrow 3.0}$	55.1 $_{\downarrow 19.1}$	74.6 $_{\downarrow 4.8}$	60.8 $_{\downarrow 1.9}$	32.6 $_{\downarrow 21.0}$	44.0 $_{\downarrow 16.0}$	76.9 $_{\downarrow 5.3}$	79.1 $_{\downarrow 0.8}$	0.069 $_{\downarrow 0.000}$	57.2 $_{\downarrow 0.6}$
GPT-5.4 Codex (OpenAI, 2026)	70.7 $_{\downarrow 27.3}$	88.9 $_{\downarrow 10.6}$	5.847 $_{\downarrow 0.863}$	50.2 $_{\downarrow 39.8}$	53.4 $_{\downarrow 14.0}$	44.4 $_{\downarrow 38.5}$	77.6 $_{\downarrow 3.1}$	83.9 $_{\downarrow 5.2}$	84.6 $_{\downarrow 8.7}$	74.0 $_{\downarrow 0.2}$	71.8 $_{\downarrow 7.5}$	45.7 $_{\downarrow 17.0}$	36.2 $_{\downarrow 17.5}$	**52.0** $_{\downarrow 8.0}$	72.5 $_{\downarrow 9.7}$	**83.5** $_{\uparrow 3.6}$	0.074 $_{\downarrow 0.005}$	48.6 $_{\downarrow 7.9}$
Open-source Reasoning LLMs																		
DeepSeek-v4 Pro (DeepSeek-AI, 2026)	95.2 $_{\downarrow 2.8}$	73.0 $_{\downarrow 26.6}$	5.512 $_{\downarrow 0.528}$	79.4 $_{\downarrow 10.6}$	53.5 $_{\downarrow 13.9}$	83.0 $_{0}$	78.1 $_{\downarrow 2.5}$	66.5 $_{\downarrow 22.7}$	42.6 $_{\downarrow 50.6}$	70.0 $_{\downarrow 4.2}$	54.2 $_{\downarrow 25.1}$	32.9 $_{\downarrow 29.8}$	15.6 $_{\downarrow 38.1}$	30.0 $_{\downarrow 30.0}$	74.4 $_{\downarrow 7.9}$	39.7 $_{\downarrow 40.2}$	0.067 $_{\downarrow 0.002}$	43.1 $_{\downarrow 13.4}$
Proprietary Chat LLMs																		
Gemini-3 Flash (Google DeepMind, 2026)	97.6 $_{\downarrow 0.5}$	98.8 $_{\downarrow 0.7}$	5.710 $_{\downarrow 0.726}$	81.1 $_{\downarrow 8.9}$	53.2 $_{\downarrow 14.2}$	76.2 $_{\downarrow 6.8}$	78.0 $_{\downarrow 2.7}$	68.3 $_{\downarrow 20.8}$	**93.3** $_{0}$	69.7 $_{\downarrow 4.5}$	**74.9** $_{\downarrow 4.4}$	51.5 $_{\downarrow 11.2}$	33.5 $_{\downarrow 20.2}$	38.0 $_{\downarrow 22.0}$	49.5 $_{\downarrow 32.8}$	59.5 $_{\downarrow 20.4}$	0.069 $_{\downarrow 0.000}$	47.9 $_{\downarrow 8.6}$
Claude Opus 4.6 (Anthropic, 2026a)	95.4 $_{\downarrow 2.6}$	98.3 $_{\downarrow 1.3}$	5.668 $_{\downarrow 0.684}$	**85.4** $_{\downarrow 4.6}$	45.2 $_{\downarrow 22.2}$	83.0 $_{0}$	81.8 $_{\uparrow 1.2}$	78.1 $_{\downarrow 11.0}$	52.9 $_{\downarrow 40.3}$	**81.6** $_{\uparrow 7.4}$	65.5 $_{\downarrow 13.9}$	55.4 $_{\downarrow 7.3}$	40.7 $_{\downarrow 12.9}$	28.0 $_{\downarrow 32.0}$	**78.3** $_{\downarrow 3.9}$	69.3 $_{\downarrow 10.6}$	0.072 $_{\downarrow 0.003}$	54.2 $_{\downarrow 2.3}$
Open-source Chat LLMs																		
Llama-3.3 70B (Meta AI, 2024)	79.4 $_{\downarrow 18.6}$	66.2 $_{\downarrow 33.4}$	7.449 $_{\downarrow 2.465}$	72.7 $_{\downarrow 17.3}$	39.1 $_{\downarrow 28.3}$	40.0 $_{\downarrow 43.0}$	72.8 $_{\downarrow 7.8}$	50.0 $_{\downarrow 39.1}$	51.1 $_{\downarrow 42.2}$	65.9 $_{\downarrow 8.3}$	42.9 $_{\downarrow 36.5}$	39.4 $_{\downarrow 23.3}$	**46.7** $_{\downarrow 7.0}$	38.0 $_{\downarrow 22.0}$	32.9 $_{\downarrow 49.4}$	63.8 $_{\downarrow 16.1}$	0.070 $_{\downarrow 0.001}$	37.6 $_{\downarrow 18.9}$
Qwen2.5-Coder 32B (Alibaba Cloud, 2024)	96.6 $_{\downarrow 1.5}$	66.6 $_{\downarrow 33.0}$	9.502 $_{\downarrow 4.518}$	77.1 $_{\downarrow 12.9}$	39.1 $_{\downarrow 28.3}$	40.0 $_{\downarrow 43.0}$	69.8 $_{\downarrow 10.8}$	54.8 $_{\downarrow 34.3}$	42.6 $_{\downarrow 50.6}$	80.0 $_{\uparrow 5.8}$	52.5 $_{\downarrow 26.8}$	44.0 $_{\downarrow 18.6}$	19.6 $_{\downarrow 34.1}$	31.0 $_{\downarrow 29.0}$	50.2 $_{\downarrow 32.0}$	25.5 $_{\downarrow 54.4}$	0.117 $_{\downarrow 0.048}$	38.4 $_{\downarrow 18.1}$
Kimi-k2.6 (Moonshot AI, 2026)	96.4 $_{\downarrow 1.6}$	42.9 $_{\downarrow 56.7}$	6.139 $_{\downarrow 1.155}$	48.4 $_{\downarrow 41.6}$	52.1 $_{\downarrow 15.3}$	83.0 $_{0}$	**95.9** $_{\uparrow 15.3}$	71.4 $_{\downarrow 17.8}$	88.0 $_{\downarrow 5.3}$	70.5 $_{\downarrow 3.7}$	70.4 $_{\downarrow 9.0}$	**64.0** $_{\uparrow 1.3}$	43.0 $_{\downarrow 10.7}$	48.0 $_{\downarrow 12.0}$	49.3 $_{\downarrow 33.0}$	79.1 $_{\downarrow 0.8}$	**0.064** $_{\downarrow 0.005}$	**57.2** $_{\downarrow 0.7}$
Human reference																		
Human Top (%)	*98.0*	*99.6*	*4.98*	*90.0*	*67.4*	*83.0*	*80.6*	*89.1*	*93.3*	*74.2*	*79.3*	*62.7*	*53.7*	*60.0*	*82.2*	*79.9*	*0.069*	*56.5*
Avg. human–LLM gap (%)	↓ 6.6	↓ 18.8	↓ 1.333	↓ 18.0	↓ 17.1	↓ 14.6	↓ 1.3	↓ 19.1	↓ 24.8	↓ 4.0	↓ 17.0	↓ 13.5	↓ 19.4	↓ 22.4	↓ 23.0	↓ 17.1	↓ 0.008	↓ 9.8

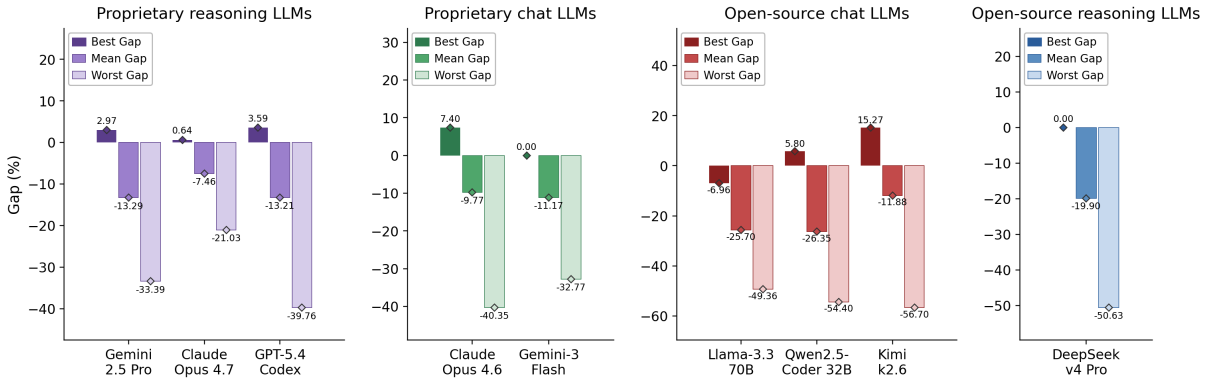


Figure 4: Best, Mean, and Worst Gap performance for leading LLMs on the classification tasks of the GNN-CB benchmark, grouped by model family. **Best Gap** denotes the best performance margin across the 16 classification competitions compared to the human top performance. **Mean Gap** is the average performance difference relative to the human top performance across all classification competitions. **Worst Gap** represents the most negative performance difference relative to the human top performance.

that standard GNN pipelines are implemented reliably when task specifications are clear. In contrast, harder competitions (e.g., C02, C04, C18) show lower performance and larger gaps between best and worst submissions, reflecting higher sensitivity to architectural and optimization choices. Several medium-difficulty tasks (e.g., C09, C10) also exhibit notable variability, suggesting that engineering and training decisions remain critical even for relatively standard formulations. The two regression tasks (C14 and C17) display similar behaviour, with larger relative performance gaps and less stable convergence compared to most classification settings. These observations suggest that human participants perform consistently on conventional GNN problems but encounter increasing difficulty as tasks require more advanced graph reasoning, domain-specific adaptation, and complex optimization strategies.

4.2 LLM Evaluation Results

Table 3 reports the per-competition performance of the nine evaluated LLMs on all 18 GNN-CB tasks

under a fixed plan-then-code prompting protocol with a bounded execute-and-repair loop. The most important finding is that LLMs consistently fall short of Human Top performance across the benchmark. Concretely, across all 9 evaluated models and 18 competition tasks (162 model–task pairs in total), LLMs surpass or ties the Human Top score in only 19 cases. In other words, in more than 88% of cases, no LLM reaches the performance level achieved by the best human competitor. As shown in Figure 13, Human Top averages are significantly higher than LLM averages across all competitions. Among models achieving Human Top-level performance, no model family stands out, and the Human-Top level performance counts are roughly proportional to the number of models in each family. In addition, the average LLM performance exceeds the average human performance in only 6 of 16 competitions (37.5%), as shown in Figure 14, indicating their limited ability across competitions.

To further assess LLM performance fairly, we analyze the gap to Human Top performance across 16 classification competitions. Open-source models exhibit larger average gaps across tasks, indicating lower reliability and higher cross-task variability. This trend is observed for both reasoning and chat models. DeepSeek-v4 Pro underperforms proprietary reasoning counterparts in mean gap, while open-source chat models (Llama-3.3 70B, Qwen2.5-Coder 32B) show larger gaps than proprietary chat models (Claude Opus 4.6, Gemini-3 Flash). An exception is Kimi-k2.6, which approaches the consistency of proprietary chat models. Within the proprietary category, reasoning and chat models achieve comparable performance, with no consistent advantage for either group, as also shown in Figure 4. These results suggest that reasoning-oriented LLMs do not consistently outperform chat-oriented models on GNN-based tasks. In addition, LLMs exhibit larger performance gaps on Hard competitions compared to Easy and Medium competitions, indicating that human expertise remains particularly valuable for challenging GNN tasks. Further results are presented in Appendix C.5.

5 Discussion

The findings in Table 3 and Figure 4 reveal the core limitations of LLMs as autonomous GNN engineers. Although they do not match the Human Top, the primary limitation lies is the cross-task inconsistency: all evaluated models exhibit large variance across the 18 competitions, indicating limited transferability of GNN construction and reasoning capabilities across heterogeneous problem settings.

This inconsistency persists across model families. Proprietary models demonstrate better average gaps than their open-source counterparts, yet no meaningful difference emerges between reasoning and chat LLMs within either tier—suggesting that chain-of-thought reasoning capabilities do not confer a systematic advantage on GNN-specific tasks. In addition, performance gaps widen on Hard tier competitions, which shows the inability of LLMs dealing with complex GNN-based tasks.

The absence of reliable transfer of GNN design and optimization strategies across task types suggests that ensembling or task-aware model routing may be more effective than scaling a single model. Furthermore, competition-style evaluation provides a more discriminative signal of real-world GNN coding ability than isolated benchmarks, and underscores the risks of deploying LLM-generated GNN solutions without expert oversight.

6 Conclusion

We introduced GNN-CB, the first *competition-based* benchmark for evaluating both humans and LLMs on GNN coding tasks. The benchmark spans 18 human-designed and curated competitions across diverse graph categories, application domains, and difficulty tiers, and is evaluated through a unified hidden-test pipeline that enforces identical conditions on human and LLM submissions. To enable fair and reproducible assessment, we proposed three complementary protocols covering competition design, controlled human evaluation, and frozen-prompt LLM evaluation with bounded execute-and-repair iterations, supporting both API-only and autonomous agent-based execution. Our experiments show that current LLMs do not match expert human performance on most GNN tasks across all difficulty levels. Their main limitation is the inability to maintain robust and consistent performance across heterogeneous graph learning problems. By releasing the benchmark and dynamic leaderboards publicly, GNN-CB provides a foundation for evaluating LLM capabilities in GNN coding, enables continuous comparison with future submissions, and serves as a progressive educational corpus for GNN coding.

Limitations

Due to API and compute-budget constraints, our LLM evaluation is limited to nine leading proprietary and open-source models across two execution modes (API-only and agent-based), and our human evaluation involves 13–17 expert volunteers per competition. The model and task selection was guided by a balance between inclusiveness and feasibility, aiming to cover diverse access types (proprietary vs. open-source), and model families (Claude, Gemini, GPT, Llama, Qwen, Kimi, DeepSeek) while maintaining reproducibility and accessibility for research groups with limited compute resources. We believe this setup provides a meaningful and representative evaluation of contemporary LLM capabilities on end-to-end GNN coding tasks, without requiring the extensive infrastructure typically available only to large industrial laboratories.

While GNN-CB covers a broad range of node-, edge-, and graph-level problems, certain emerging settings such as graph generation, quantum graph modelling, and self-supervised or reinforcement-learning paradigms are not included in the current release. The use of task-specific evaluation metrics (macro-F_1, accuracy, AUC, MAE, and NMAE) reflects the heterogeneity of real-world graph problems and limits direct cross-task comparison. To mitigate this, we report per-task gap-to-human-top scores and separate classification and regression results in our visualizations, particularly since the benchmark contains only two regression competitions. Finally, because LLM capabilities evolve rapidly, the reported results represent model snapshots available as of May 2026, although the qualitative patterns we observe—difficulty stratification, the capability-versus-consistency gap, and family-level differences—are expected to remain informative as models continue to advance. These limitations can be viewed as opportunities for future extensions of GNN-CB toward broader graph-learning paradigms, larger model panels, and continuously updated evaluations.

Future Directions for Extended LLM Evaluation

The present study establishes GNN-CB as a zero-shot, frozen-prompt benchmark, and its infrastructure enables several deeper analyses that we did not pursue here due to computational and budgetary constraints. On the modelling side, future work can measure how performance scales beyond the current zero-shot protocol through few-shot prompting, retrieval-augmented generation, and task-specific fine-tuning, and can characterise stochastic variability via repeated sampling, confidence intervals, and pass@k-style metrics rather than the single greedy run reported here. Because every submission is executed and validated, the benchmark also supports a fine-grained error-mode taxonomy—incorrect graph construction, invalid message-passing architectures, preprocessing data leakage, mishandled class imbalance, and malformed submissions—that would reveal *where* and *why* LLMs break down beyond aggregate scores.

On the evaluation side, the logged execution and repair traces enable a detailed token-economy and cost–latency analysis, particularly contrasting the per-call budget of API-only runs with the multi-step session budget of agent-based runs, alongside an analysis of agentic depth (tool-call counts, repair-loop depth, and the contribution of extended chain-of-thought). Rigour can be further strengthened through paired significance testing, bootstrap confidence intervals, and rank-correlation analysis of the human–LLM ordering. Finally, as a living benchmark with automated infrastructure and dynamic leaderboards, GNN-CB can be extended to broader paradigms (graph generation, self-supervised, temporal, and reinforcement-learning formulations) and used longitudinally to track whether the capability–consistency gap narrows as new models are released.

Ethics Statement

Participant Consent: All human participants provided informed consent for the publication of their performance results. Participants were expert volunteers who co-designed the benchmark; no monetary compensation was provided. Only aggregate counts and per-task scores are reported (Table 2); no personally identifying information about participants is released.

Personally Identifying Information and Sensitive Content: All datasets are used in their publicly released forms. Sensitive biomedical datasets such as OASIS-3 (LaMontagne et al., 2019) and ABIDE (Craddock et al., 2013) follow their original de-identification protocols and data-use agreements; only aggregate per-task scores are reported. The GitHub Social Network dataset (Rozemberczki et al., 2019) contains public GitHub user-

names that are already openly available; the usernames are not the target of prediction and no additional de-anonymization is performed. The Elliptic dataset (Weber et al., 2019) uses pseudo-anonymous transaction identifiers with no link to real identities. None of the datasets contain free-form user-generated text in their released versions, so no offensive-content screening was required.

Licenses: All datasets are used under their original public licenses (MoleculeNet, Cora, OGB, OASIS-3 under data-use agreement, Elliptic, GossipCop, GitHub Social Network, etc.). LLMs are accessed under the providers' API terms of service. GNN-CB itself, including the benchmark repositories, evaluation pipeline, and leaderboard infrastructure, will be released under the MIT license.

Potential Risks: GNN-CB evaluates LLM-generated GNN code that, if deployed in safety-critical domains such as medical diagnosis, financial-fraud detection, or neuroimaging without expert verification, could amplify model failures or propagate dataset biases. We release the benchmark for controlled research and educational use and discourage direct deployment of LLM-generated solutions without expert review.

Use of AI Assistants. AI assistants were used solely for editorial and coding support during paper preparation, including grammar correction, LaTeX formatting, table and figure layout, and minor stylistic polishing. AI assistants were not used to generate experimental results, write benchmark code, or produce scientific claims. All scientific content—research questions, benchmark design, dataset curation, experimental design, analyses, and conclusions—was conceived and verified by the human authors. The AI assistants used for paper preparation are distinct from the LLMs evaluated in the benchmark.

References

Alibaba Cloud. 2024. Qwen2.5-Coder-32B-Instruct. https://huggingface.co/Qwen/Qwen2.5-Coder-32B-Instruct. Accessed: 2026-05-10.

Mohamed Amgad, Lamees A Atteya, Hagar Hussein, Kareem Hosny Mohammed, Ehab Hafiz, Maha AT Elsebaie, Ahmed M Alhusseiny, Mohamed Atef Al-Moslemany, Abdelmagid M Elmatboly, Philip A Pappalardo, and 1 others. 2022. Nucls: A scalable crowdsourcing approach and dataset for nucleus classification and segmentation in breast cancer. *GigaScience*, 11:giac037.

Anthropic. 2026a. Claude opus 4.6. https://www.anthropic.com/claude/opus. Accessed: 2026-05-10.

Anthropic. 2026b. Claude opus 4.7. https://www.anthropic.com/claude/opus. Accessed: 2026-05-10.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, and 1 others. 2023. Multipl-e: A scalable and polyglot approach to benchmarking neural code generation. *IEEE Transactions on Software Engineering*, 49(7):3675–3691.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yizheng Chen, Zhoujie Ding, Lamya Alowain, Xinyun Chen, and David Wagner. 2023. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In *Proceedings of the 26th international symposium on research in attacks, intrusions and defenses*, pages 654–668.

Cameron Craddock, Yassine Benhajali, Carlton Chu, Francois Chouinard, Alan Evans, András Jakab, Budhachandra Singh Khundrakpam, John David Lewis, Qingyang Li, Michael Milham, and 1 others. 2013. The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7(27):5.

J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, and 3 others. 2022. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56.

DeepSeek-AI. 2026. DeepSeek-V4 Pro: a reasoning-optimised open-weights llm. https://huggingface.co/deepseek-ai. Accessed: 2026-05-10.

Rahul S Desikan, Florent Ségonne, Bruce Fischl, Brian T Quinn, Bradford C Dickerson, Deborah Blacker, Randy L Buckner, Anders M Dale, R Paul Maguire, Bradley T Hyman, and 1 others. 2006. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *Neuroimage*, 31(3):968–980.

Yingtong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. 2021. User preference-aware fake news detection. In *Proceedings of the 44th international*

ACM SIGIR conference on research and development in information retrieval, pages 2051–2055.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2022. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340.

Bruce Fischl. 2012. Freesurfer. *Neuroimage*, 62(2):774–781.

Justin Gilmer, Samuel S. Schütt, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272. PMLR.

Google DeepMind. 2025. Gemini 2.5 pro. https://deepmind.google/technologies/gemini/pro/. Accessed: 2026-05-10.

Google DeepMind. 2026. Gemini 3 flash. https://deepmind.google/technologies/gemini/flash/. Accessed: 2026-05-10.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

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*, 33:22118–22133.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Xue Jiang, Yihong Dong, Lecheng Wang, Zheng Fang, Qiwei Shang, Ge Li, Zhi Jin, and Wenpin Jiao. 2024. Self-planning code generation with large language models. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–30.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? In *The twelfth international conference on learning representations*.

Csaba Kerepesi, Balint Szalkai, Balazs Varga, and Vince Grolmusz. 2017. The braingraph.org database of high resolution structural connectomes and the brain graph tools. *Cognitive Neurodynamics*, 11:483–486.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Pamela J LaMontagne, Tammie LS Benzinger, John C Morris, Sarah Keefe, Russ Hornbeck, Chengjie Xiong, Elizabeth Grant, Jason Hassenstab, Krista Moulder, Andrei G Vlassenko, and 1 others. 2019. Oasis-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer disease. *medrxiv*, pages 2019–12.

Xiaoxiao Li, Yuan Zhou, Nicha Dvornek, Muhan Zhang, Siyuan Gao, Juntang Zhuang, Dustin Scheinost, Lawrence H. Staib, Pamela Ventola, and James S. Duncan. 2021. Braingnn: Interpretable brain graph neural network for fmri analysis. *Medical Image Analysis*, 74:102233.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Max Little. 2007. Parkinsons. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59C74.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572.

Meta AI. 2024. Llama 3.3 70b instruct. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct. Accessed: 2026-05-10.

Moonshot AI. 2026. Kimi K2.6: a long-context open-weights chat llm. https://kimi.moonshot.cn/. Accessed: 2026-05-10.

OpenAI. 2026. GPT-5.4 Codex. https://openai.com/index/gpt-5/. Accessed: 2026-05-10.

Qiwei Peng, Yekun Chai, and Xuhong Li. 2024. Humaneval-xl: A multilingual code generation benchmark for cross-lingual natural language generalization. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pages 8383–8394.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale attributed node embedding. *Preprint*, arXiv:1909.13021.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607. Springer International Publishing.

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

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188.

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 974–983.

Jingyao Zeng, Yadong Zhang, Yunfei Shang, Jialin Mai, Shuo Shi, Mingming Lu, Congfan Bu, Zhewen Zhang, Zaichao Zhang, Yang Li, and 1 others. 2022. Cancerscem: a database of single-cell expression map across various human cancers. *Nucleic acids research*, 50(D1):D1147–D1155.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*.

A Competition Specifications

This appendix provides detailed descriptions of the competitions included in GNN-CB, each following a standardized structure (task definition, dataset statistics, evaluation metric, and submission protocol) and a unified evaluation pipeline based on hidden test sets, secure submission handling, and automated GitHub Actions scoring to ensure reproducibility and fairness.

The benchmark spans a diverse set of graph problem categories, including homogeneous, heterogeneous, temporal, spatial, hierarchical, molecular, and generalization graphs. To fomally define, Homogeneous : Graph with uniform node and edge types, Heterogeneous : Graph with multiple node and edge type, Temporal : Graph changes over time, Spatial: Graph is embedded in a spatial domain where nodes and edges reflect physical or geometric proximity, Hierarchical: Graph with multilevel structure among nodes, Molecular: Graph represents molecular structures, where nodes are atoms and edges are chemical bonds, Generalization: Test graph is drawn from distributions different from the training graph.

We acknowledge that some categories may overlap; however, we assign each competition to its most distinct category. For example, a molecular graph is also homogeneous, but it is labeled as Molecular to distinguish it from other types.

Across all competitions, a unified submission and evaluation protocol is enforced to ensure fairness, reproducibility, and consistent leaderboard construction. Each participant is allowed a single final submission per competition version, and only the latest valid submission is considered for evaluation. Submissions that are not properly encrypted, fail decryption, or do not conform to the required format are automatically rejected and excluded from scoring. Leaderboard ranking follows a Kaggle-style tie-breaking rule, where identical scores are assigned the same rank, and no additional ordering is enforced between tied entries.

C01: PROVEN-GNN

PROVEN-GNN (**PRO**gram **V**ulnerability **E**xamination **N**etwork using **G**raph **N**eural **N**etworks) is a graph classification task on heterogeneous graph representations of code functions. Each source code function is represented as a graph $G_k = (V_k, E_k)$, where nodes denote code fragments (methods, variable declarations, etc.) and edges represent the relationships between them (AST, CFG, etc.). The objective is to detect vulnerable code functions based on these representations.

The dataset consists of 3109 functions from the

DiverseVul dataset (Chen et al., 2023), with vulnerable functions representing 29% of the data. The graph representations were generated using the tool *Joern*. It allows generating code property graphs that capture syntactic properties, control flows, and data dependencies. Each node is represented with 512 features, while edge features are represented using a one-hot encoding of the edge type (12 features).

Participants receive training and test graph files containing node embeddings, edge indices, and edge features. The labels (1 for vulnerable and 0 for non-vulnerable) are provided only for the training graphs. Participants are then required to generate predictions for the test graphs and submit them in an encrypted form using the provided public key. An automated pipeline then validates, decrypts, scores, and updates the leaderboard, evaluating each submission using the macro F1-score.

C02:CGCC

The **CGCC** (**C**ity **G**raph **C**lassification **C**hallenge) is a graph-level classification task defined over real-world urban street networks. Each sample is a city represented as a homogeneous graph $G_k = (V_k, E_k)$, wherenodes denote road intersections and edges denote the road segments connecting them. Given a collection of such graphs $\mathcal{G} = \{G_k\}$, the objective is to learn a mapping $f(G_k) \rightarrow y_k$ that assigns each city to one of three layout classes: organic (0), grid (1), or hybrid (2).

The dataset comprises street networks for 120 cities distributed across North America, South America, Europe, Asia, Africa, Oceania, the Middle East, and Central Asia, extracted from OpenStreetMap via OSMnx within a 500 m buffer around each city centre and restricted to driveable roads. The class distribution reflects naturally occurring urban patterns and is moderately imbalanced, with 37 grid, 31 organic, and 52 hybrid cities. Each graph retains node coordinates, edge geometries and lengths, and graph-level attributes such as the city name and layout type; for the evaluation split, the layout label is withheld. The corpus is partitioned 70/30 into training and test sets with stratification by class.

Participants are provided with serialized city graphs and a label table that associates each training graph with its target class. No engineered urban-form descriptors (e.g., circuity, intersection density, orientation entropy) are provided. Participants are required to generate a prediction file mapping each test city to its predicted class, encrypt it locally using the provided public key, and submit encrpted file through a pull-request-based workflow. An automated pipeline then validates, decrypts, and scores the submissions, updating the leaderboard. Each submission is evaluated using the macro F1-score.

C03: Real Or Fake!

The **Real or Fake** challenge is a graph classification task based on Twitter news propagation data from the *GossipCop* dataset (Shu et al., 2020; Dou et al., 2021). The objective is to train a Graph Neural Network (GNN) that predicts whether a news article is real or fake by leveraging both node features and the underlying propagation structure. Given a collection of hierarchical graphs $\mathcal{G} = \{G_k\}$, where each graph $G_k = (V_k, E_k)$ represents the spread of a single news article, the task is to predict the label y_k for each graph :$f(G_k) \rightarrow y_k$ where $y_k \in \{0, 1\}$, with 1 indicating verified news and 0 indicating fake news.

In this setting, nodes correspond to either the root news article or users who interacted with it, while edges capture interactions such as retweets. Each node is associated with rich features, including textual embeddings (e.g., BERT and spaCy) and user profile attributes. The dataset is split into 3,826 training graphs, 546 validation graphs, and 1,092 test graphs.

Participants are provided with the graph structure, node features, and data splits distributed across multiple files. They are then expected to generate predictions for the test graphs, encrypt them locally using the provided public key and and submit through a pull-request. An automated pipeline subsequently validates, decrypts, and evaluates the submissions, updating the leaderboard accordingly. Each submission is assessed using the accuracy score.

C04: GLIMPS-GNN

The GLIMPS-GNN (**G**raph-based **L**iquid-biopsy **I**nductive **M**odeling for **P**reeclamp**S**ia) challenge is an inductive node-classification task on a biological graph. Given $G = (V, E)$ with feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$, the goal is to learn a mapping $f(v_i) \rightarrow y_i$ where each node $v_i \in \mathcal{V}$ is assigned a label $y_i \in \{0, 1\}$ corresponding to Control (0) and Preeclampsia (1). The training subgraph comprises cell-free RNA (cfRNA) from maternal plasma, while the inductive test set contains unseen

placental RNA-seq samples. Models must therefore generalize across a substantial domain shift between liquid-biopsy and tissue transcriptomics, making this competition fall under the genereliza-tion category.

Data are sourced from GEO: maternal plasma cfRNA (GSE192902) and placental RNA-seq (GSE234729). Nodes are individual samples; edges encode similarity and ancestry. The graph contains 320 nodes and 3200 edges. Node features are harmonized gene-expression profiles. The training set has 209 cfRNA samples; the test set has 111 placenta samples with moderately imbalanced classes.

Participants are provided with the graph structure, node features, and data splits distributed across multiple files. They are then expected to generate predictions for the test graphs and encrypt them locally using the provided public key. After submission, an automated GitHub Actions pipeline subsequently validates, decrypts, and evaluates the submissions, updating the leaderboard accordingly. Each submission is assessed using the macro-averaged F1-score.

C05: GNN_BACE

GNN_BACE (BACE Inhibition Prediction) is a graph-level binary classification task whose primary objective is to predict whether a given molecule acts as an inhibitor of the β-secretase 1 (BACE-1) enzyme, a critical target in Alzheimer's disease research. Formally, the objective is to learn a mapping $f(G_k) \to y_k$ for each molecule represented as molecular graphs $G_k = (V_k, E_k)$. Molecules are represented as undirected graphs $G_k = (V_k, E_k)$, where nodes correspond to atoms and edges correspond to chemical bonds. Each node is described by an 8-dimensional one-hot atom-type feature vector, while edges include simple bond-level attributes. Label is 1 for active inhibitors and inactive compounds as 0.

Participants are provided only with atom features and connectivity information, without any pre-computed molecular descriptors (e.g., MolLogP, Polar Surface Area). The dataset for this challenge is derived from the BACE dataset in the widely used MoleculeNet benchmark suite (Wu et al., 2018). Participants are then required to generate predictions for the test graphs and encrypt them locally using the provided public key. After submitting encrypted file using pull-request, an automated GitHub Actions pipeline subsequently validates, decrypts, and evaluates the submissions, updating the leaderboard accordingly. Each submission is assessed using the Macro F_1-Score.

C06: DiaGraph

The **DiaGraph** (Robust Diabetes Prediction on Noisy Graphs) is a transductive node-level classification task where each sample corresponds to a patient represented as a node in a homogeneous graph $G = (V, E)$. Nodes denote patients, while edges encode similarity relationships based on latent clinical affinities. The objective is to learn a mapping $f(v_i) \to y_i$ that predicts whether a patient is diabetic (1) or non-diabetic (0).

Participants are provided only with node features and the adjacency matrix for all nodes, while labels are available only for the training nodes. For evaluation, participants generate a prediction file containing binary labels for all nodes in the hidden test set. The predictions are encrypted locally using the provided public key prior to submission, and only encrypted files are submitted to the repository. An automated evaluation pipeline decrypts the submissions, validates their format, computes the official score against hidden test labels, and updates the public leaderboard. Each submission is evaluated using the macro F1-score.

C07: GRAPE

The **GRAPE** (**G**raph **R**etinal **A**nalysis for **P**rediction and **E**valuation) challenge is a graph-level classification benchmark designed for diabetic retinopathy (DR) detection from retinal vessel graphs. Each sample corresponds to a retinal vascular network represented as a spatial graph $G_k = (V_k, E_k)$, where nodes denote vessel bifurcation or intersection points and edges represent vessel segments connecting them. Given a collection of retinal graphs $\mathcal{G} = \{G_k\}$, the objective is to learn a mapping $f(G_k) \to y_k$ that predicts whether the retinal image corresponds to a healthy subject (0) or a diabetic retinopathy patient (1).

The benchmark is constructed from retinal vessel graphs extracted from three publicly available retinal imaging datasets: DRIVE, STARE, and HRF. In total, the dataset contains 70 graphs, comprising 48 healthy and 22 diabetic retinopathy cases. Specifically, the DRIVE subset contributes 20 graphs (17 healthy, 3 DR), STARE contributes 20 graphs (16 healthy, 4 DR), and HRF contributes 30 graphs (15 healthy, 15 DR). The dataset is partitioned into 55 training graphs and 15 hidden test graphs using

stratified sampling to preserve class proportions across splits. Each retinal graph preserves the vascular connectivity structure derived from vessel segmentation outputs.

Participants are provided with node features and the adjacency matrix for all graphs, while labels are available only for the training graphs. Participants are expected to generate a prediction file containing binary labels for all graphs in the hidden test set. The predictions are encrypted locally using the provided public key prior to submission, and only encrypted files are submitted to the repository. An automated GitHub Actions workflow securely injects the private decryption keys, decrypts the submissions, validates their format, computes the official score against the hidden test labels, and updates the public leaderboard. Each submission is evaluated using the macro F1-score.

C08: Liar Nodes

The **Liar Nodes** challenge is formulated as a transductive node classification problem over a homogenous graph derived from single-cell cancer transcriptomics data. Each node represents an individual cell, while graph edges encode contextual or biological similarity relationships between cells within the tumor microenvironment. Given a graph $G = (V, E)$, the objective is to learn a mapping $f(v_i) \rightarrow y_i$ that predicts whether a cell is cancerous or non-cancerous.

The underlying data originate from the **Cancer Single-cell Expression Map (CancerSCEM)** resource (Zeng et al., 2022), which contains large-scale single-cell RNA sequencing data across multiple cancer types. The original dataset provides UMI count matrices, cell-type annotations, functional molecular descriptors, and cell-cell interaction information. For the purposes of this benchmark, the data are transformed into graph learning instances suitable for GNN training and evaluation. Node embeddings are intentionally altered and partially corrupted in order to create contradictions between intrinsic node representations and graph context.

Participants are provided with graph-structured data derived from the CancerSCEM resource, together with training labels for supervised learning. Final predictions are submitted via a designated Google Form for evaluation and leaderboard ranking. An automated evaluation pipeline computes accuracy on hidden test labels and updates the public leaderboard accordingly.

C09: GraphFMD

The **GraphFMD** (**Graph**-based **F**inancial **M**isconduct **D**etection) challenge is a temporal inductive node classification task defined over the Bitcoin transaction network. Each sample is a transaction represented as a node v_i in a directed temporal graph $G = (V, E)$, where nodes denote individual Bitcoin transactions and edges denote the flow of Bitcoin between them. The objective is to learn a mapping $f(v_i) \rightarrow y_i$ that assigns each test-time transaction to one of two classes: illicit (1) or licit (2).

The dataset is derived from the Elliptic Bitcoin transaction graph (Weber et al., 2019) and is partitioned temporally: training nodes span time steps 1–34, comprising 16,658 labelled transactions, while test nodes are drawn from time steps 35 and beyond, comprising 8,896 transactions. The graph contains approximately 5 million directed edges. Each node is associated with a 165-dimensional feature vector consisting of 94 local transaction features (e.g., transaction amounts, fees, and input/output counts) and 71 aggregated neighbourhood features derived from one-hop neighbours; Gaussian noise has been injected into these features to simulate real-world data degradation. The dataset exhibits strong class imbalance, with illicit transactions constituting only 1–3% of all nodes.

Participants are provided with node feature files and an edge list for the full graph, together with ground-truth labels for the training split. No handcrafted forensic descriptors (e.g., mixing patterns, chain-length heuristics, or address-clustering signals) are supplied. Participants are expected to generate a prediction table mapping each test transaction identifier to its predicted class $1, 2$. Submissions are encrypted and uploaded via a secure Google Form, which triggers a GitHub Actions workflow that validates the submission, computes the performance against the hidden label file, and updates the public leaderboard automatically. Performance is evaluated on the hidden test split using the macro F1-score.

C10: NetLinkArena

The **NetlinkArena** (Link Prediction Challenge) is a link prediction task defined on a real-world scientific citation network. Each sample corresponds to a pair of scientific papers represented within a homogenous citation graph $G = (V, E)$, where nodes denote research papers and edges represent

citation relationships between them. Given a partially observed graph and a set of candidate node pairs (u, v), the objective is to learn a mapping $f(u, v) \to y$ that predicts whether a citation link exists between the two papers, where $y \in \{0, 1\}$, with 1 indicating the presence of a link and 0 indicating no link.

The dataset is constructed from the original **CiteSeer** dataset (Sen et al., 2008), in which additional perturbations and synthetic noise were introduced to generate a more challenging and realistic link prediction benchmark. The graph contains 3,327 scientific papers, each represented by a 2,742-dimensional feature vector that captures the semantic and structural properties of each document. Citation relationships are provided as source-target node pairs, where positive edges correspond to observed citations and negative edges represent non-citation pairs sampled from the graph. The dataset is partitioned into training, validation, and hidden test splits for supervised evaluation.

Participants are provided with the node feature matrix, labeled source–target node pairs for training and validation, and unlabeled node pairs for inference. Participants are expected to generate a file mapping each test node pair to a predicted probability score indicating the likelihood of a citation link. Submissions are entered through a Google Sheets-based workflow, where each team records predictions in a designated submission sheet. A GitHub Actions workflow then validates the submission format and updates the leaderboard automatically. Performance is evaluated on the hidden test split using the AUC score.

C11: PARK-GNN

The **PARK-GNN** Challenge (**P**arkinson's **A**coustic **R**epresentation & **K**nowledge with **G**raph **N**eural **N**etworks) is a transductive node classification task defined over a homogenous graph constructed from acoustic voice recordings. Each node v_i in the graph $G = (V, E)$ represents a single voice recording, characterized by a 22-dimensional feature vector of acoustic measurements including vocal fundamental frequency statistics, jitter and shimmer variations, harmonics-to-noise ratios, and nonlinear dynamical complexity features. Edges are formed by a k-nearest-neighbor criterion ($k = 5$) over acoustic similarity in feature space, supplemented by subject-level connections that link recordings from the same individual. The objective is to learn a mapping $f(v_i) \to y_i \in \{0, 1\}$ that assigns each voice recording to either a healthy control (0) or a Parkinson's Disease patient (1).

The dataset is derived from the UCI Parkinson's Dataset (Little, 2007), comprising 195 voice recordings collected from 31 subjects, of whom 23 were diagnosed with Parkinson's Disease and 8 were healthy controls. The corpus is partitioned into a training set of 156 nodes (approximately 80%) and a hidden test set of 39 nodes (approximately 20%). Training labels are provided alongside the full graph structure; test labels are withheld and used exclusively within the automated evaluation pipeline.

Participants are provided with the serialized graph structure, node feature matrices, and a label table mapping each training node to its ground-truth class. No engineered biomarker composites or pre-computed diagnostic descriptors are supplied.Participants are expected to generate a prediction table mapping each test node identifier to its predicted class. Submissions are encrypted locally using the provided public-key scheme and uploaded through a pull-request-based workflow. An automated GitHub Actions pipeline then validates, decrypts, scores, and updates the public leaderboard. Performance is evaluated on the hidden test split using the macro-averaged F1-score.

C12: GNN-CoRA

The **GNN-CoRA** (Cora Node Classification Challenge) competition is a node classification task defined over a homogenous citation-network graph derived from the *Cora* benchmark dataset (Sen et al., 2008). The objective is to train a GNN that predicts the research-topic label of each publication node from both its textual feature representation and the surrounding citation structure. Formally, given a graph $G = (V, E)$ in which nodes correspond to publications and edges denote citation relationships, the task is to learn a mapping $f(v_i) \to y_i$ that assigns each node v_i to one of seven research-topic classes.

The dataset consists of 2,708 publication nodes connected by 5,429 citation edges. Each node is associated with a high-dimensional sparse bag-of-words representation extracted from the publication text, and the graph structure captures citation links between papers. The benchmark follows the standard citation-network split, containing 140 labelled training nodes, 500 validation nodes, and 1,000 hidden test nodes. To increase task difficulty and evaluate robustness, Gaussian noise is injected into

the node-feature representations during benchmark construction.

Participants are provided with graph connectivity information, node feature matrices, training and validation label tables, and identifiers for the hidden test nodes. The ground-truth labels for the test split remain concealed and are used exclusively within the automated evaluation pipeline. Participants are expected to generate a prediction table mapping each hidden test node to one of the seven target classes. Submissions are encrypted locally using the benchmark's public-key encryption scheme and submitted through a pull-request-based workflow. An automated GitHub Actions evaluation pipeline then validates, decrypts, scores, and updates the leaderboard accordingly. Performance is evaluated on the hidden test set using classification accuracy.

C13 : GTA

The **GTA** (**G**raph **T**opology **Ablation**) challenge is formulated as a supervised binary graph classification problem over molecular graphs derived from the **MUTAG** dataset, a widely used benchmark in chemical informatics. Each graph $G_k = (V_k, E_k)$ represents a chemical compound in which nodes correspond to atoms and edges correspond to chemical bonds. The objective is to learn a mapping $f(G_k) \rightarrow y_k$ that predicts whether a molecule is mutagenic or non-mutagenic with respect to *Salmonella typhimurium*. The dataset contains 188 molecular graphs with an average of approximately 18 nodes and 40 edges per graph. Node attributes consist of categorical atom labels interpreted as node features.

The benchmark is divided into two evaluation settings. In the *ideal graph topology* setting, participants evaluate their models on clean and unmodified node features. In the *perturbed graph topology* setting, node features are corrupted through a combination of feature distribution shift and Gaussian noise injection. Specifically, a constant offset δ is added to node features to simulate systematic domain shift:

$$x \leftarrow x + \delta,$$

where δ corresponds to a predefined feature shift parameter. In addition, Gaussian noise is injected into node attributes according to:

$$x \leftarrow x + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2),$$

where σ controls the noise magnitude. This competition falls under the generelization category due to pertubred test graphs.

The dataset is partitioned into training and hidden evaluation splits using stratified sampling. The training split contains labeled molecular graphs together with graph identifiers and binary labels, while the evaluation split provides only graph identifiers and hides the ground-truth labels used for leaderboard scoring. Models are therefore trained exclusively on clean graph features and subsequently evaluated on both clean and perturbed graph representations.

Performance is evaluated using the macro F1 score, which balances precision and recall across both classes and provides a more reliable evaluation criterion than classification accuracy alone. Each submission is scored separately under the clean and perturbed settings, yielding two metrics: F_1^{ideal} and $F_1^{\text{perturbed}}$. To quantify robustness, the benchmark additionally computes the **Robustness Gap**:

$$\text{Robustness Gap} = F_1^{\text{ideal}} - F_1^{\text{perturbed}}.$$

Smaller robustness gaps indicate models that maintain stable performance under feature corruption.

Participants submit encrypted prediction files for both clean and perturbed evaluation settings. An automated evaluation pipeline decrypts submissions securely, computes official metrics against hidden labels, and updates the public leaderboard automatically. Leaderboard ranking prioritizes the highest perturbed macro-F_1 score, followed by the smallest robustness gap as a secondary criterion.

C14: BrainAge

The **BrainAge** (Brain-Age Prediction Challenge) challenge is a graph-level regression task defined over structural neuroimaging data. The dataset consists of homogenous brain graphs $\mathcal{G} = \{G_k\}$, where each graph $G_k = (V_k, E_k)$ is constructed from structural and diffusion MRI data from a specific subject's imaging session. Nodes V_k correspond to cortical brain regions defined by the Desikan-Killiany atlas (Desikan et al., 2006) ($N = 68$), and edges E_k encode structural connectivity derived from diffusion MRI. Given a graph G_k, the objective is to learn a mapping $f(G_k) \rightarrow y_k$, where $y_k \in \mathbb{R}$ denotes the age of an individual at the time of image acquisition.

The dataset is constructed from the OASIS-3 longitudinal neuroimaging cohort (LaMontagne et al., 2019), restricted to subjects who remained cognitively normal across all visits. Cortical measurements were obtained from preprocessed

FreeSurfer (Fischl, 2012) outputs available through the Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC). Structural connectivity information is sourced from the PIT Bioinformatics Group (Kerepesi et al., 2017). The final dataset comprises 431 MRI sessions from 288 unique subjects across multiple longitudinal time points.

For every cortical region, two node features are provided: cortical thickness and gray matter volume. Edge weights are defined by diffusion MRI streamline counts between regions. To prevent information leakage arising from repeated longitudinal observations of the same participant, training, validation, and test splits were performed at the subject level using age-stratified sampling.

For evaluation, participants generate age predictions for test MRI sessions and submit them as an encrypted prediction table through a pull-request-based GitHub workflow. The automated GitHub pipeline validates the submission, decrypts the predictions, computes the evaluation score, and updates the leaderboard upon merge. Performance is evaluated on the hidden test split using MAE.

C15: OctoNode Cup

OctoNode Cup (GitHub Social Network Classification) is a transductive node classification task focused on classifying users in a homogenous GitHub social network. In this network, represented as a graph $G = (V, E)$, each node corresponds to a GitHub user and edges represent connections between users. The task aims to predict a binary label for each node based on its associated features. The objective is to learn a mapping $f(v_i) \rightarrow y_i$ where v_i denotes a GitHub user and $y_i \in \{0, 1\}$ represents the predicted class, with 1 corresponding to a Web Developer and 0 corresponding to a Machine Learning Developer.

The dataset is derived from a real GitHub developer network introduced by (Rozemberczki et al., 2019) and represents the platform as a graph. Each node is identified by a unique ID and is associated with a set of features used for prediction. The dataset contains approximately 37,700 nodes and 289,000 edges. It is divided into two partitions: a training set comprising 70% of the data, which includes both features and labels, and a test set comprising the remaining 30%, where only the features are provided while the labels remain hidden for evaluation purposes.

Participants are provided with input files required for training and evaluation. Predictions must be generated locally and encrypted prior to submission using the provided encryption method. Only encrypted submission files are uploaded through the repository-based submission system. An automated GitHub pipeline validates the submission, decrypts the predictions, and computes the evaluation score. Submissions are evaluated on the hidden test set using the Macro F1-score.

C16: Graph4ASD Challenge

The **Graph4ASD** (Graph Neural Networks for the classification of Autism Spectrum Disorder) challenge is a binary graph classification task over brain networks derived from resting-state functional magnetic resonance imaging (rs-fMRI) recordings of the human brain. Each sample corresponds to a single subject and is represented as a homogenous graph $G_k = (A_k, X_k)$, where $A_k \in \mathbb{R}^{n \times n}$ is the weighted adjacency matrix encoding functional connectivity between brain regions and X_k is the node feature matrix. Given a collection $\mathcal{G} = \{G_k\}$, the task is to learn a mapping $f(G_k) \rightarrow y_k \in \{0, 1\}$ that assigns each subject to one of two diagnostic classes: Typical Control (0) or Autism Spectrum Disorder (1).

The dataset is drawn from the Autism Brain Imaging Data Exchange (ABIDE) (Craddock et al., 2013), a multi-site repository of anonymised rs-fMRI scans collected at seventeen international acquisition sites. The full corpus contains 1,009 subjects (516 ASD, 51.14%; 493 controls, 48.86%), yielding a near-balanced two-class distribution. Each subject's brain volume is parcellated with the Craddock 200 atlas, so every graph has exactly $n = 200$ nodes corresponding to predefined regions of interest, with identical node ordering across samples. A challenge-specific subset, preprocessed exclusively for this benchmark, is partitioned into 484 training and 153 evaluation graphs; participants are required to use the data as released, without alternative preprocessing pipelines or auxiliary neuroimaging corpora.

For every subject in the training and evaluation partitions, participants receive a connectivity matrix and a node feature matrix; the training partition is additionally accompanied by subject-level diagnostic labels. Node features are restricted to identity descriptors, so any discriminative signal must be recovered from the connectivity structure over the 200 Craddock regions alone. The training and evaluation partitions are drawn from non-identical site distributions, requiring models to learn site-

invariant connectivity representations.

For evaluation, participants produce a prediction file mapping each test subject to a predicted class. The file is encrypted locally using the provided public key and submitted through a pull-request-based workflow. An automated pipeline validates, decrypts, scores, and updates the leaderboard. Performance on the hidden test split is reported using the macro F1-score.

C17: Mix2Print

Mix2Print (Learning Material Interaction Physics for identifying parameters of 3D Bioprinting) is a graph regression challenge focused on predicting optimal 3D bioprinting parameters from bioink formulation graphs. The competition models each bioink formulation as a homogenous graph, where nodes correspond to biomaterials and edges represent potential interactions between materials in the mixture. The task aims to learn the underlying material interaction physics governing extrusion-based bioprinting.

Given a graph formulation $G_i = (V_i, E_i, X_i)$, participants are required to predict three continuous targets: $f(G_i) \rightarrow y_i$ where $y_i = (\text{pressure}, \text{temperature}, \text{speed})$ corresponding to Pressure (kPa), which represents extrusion force; Temperature (°C), which represents printing temperature; and Speed (mm/s), which represents printhead velocity.

The dataset represents bioink formulations as fully connected graphs, reflecting the assumption that all biomaterials in a formulation can potentially interact with each other. Node features consist of one-hot encoded material identity features and normalized concentration value. Thus, the total node feature dimensionality is approximately $D \approx 31$. The dataset contains 423 bioink formulations collected from peer-reviewed publications, 30 biomaterials appearing at least five times ,303 training samples and 120 test samples. The train-test split follows a stratified grouping strategy based on printing temperature regimes (hydrogel versus thermoplastic formulations). The dataset captures diverse biomaterial categories including alginates, gelatins, polymers, ceramics, and natural biomaterials.

Participants are provided with graph-based input files for model development and evaluation. Participants are required to generate predictions for the hidden test set in CSV format. To preserve leaderboard integrity, all submissions must be encrypted prior to upload using the provided encryption utility. The encrypted submission file is then uploaded through a GitHub pull request. Submissions are evaluated using the Normalized Mean Absolute Error (NMAE) across three regression targets, where lower values indicate better performance. Final rankings are determined by sorting participants in ascending order of NMAE on the hidden test set.

C18: THInC

THInC (**T**umor **H**istopathology **In**ductive **C**lassification) is formulated as an **inductive node classification problem** over spatial cell graphs derived from H&E-stained histology images. Each graph $G_k = (V_k, E_k)$ corresponds to a single patient or tissue slide, where nodes represent individual cell nuclei and edges encode spatial relationships between nearby cells. Given a set of training graphs $\mathcal{G}_{train}$, the goal is to learn a mapping $f(v_i) \rightarrow y_i$ that assigns each cell to one of four classes: Tumor (0), Stromal (1), Lymphocyte (2), or Macrophage (3). Importantly, predictions must generalize to entirely unseen graphs corresponding to new patients, making this an inductive rather than transductive learning setting.

The dataset is derived from the **NuCLS** breast cancer histopathology dataset (Amgad et al., 2022) and is transformed into spatial cell graphs using a biologically grounded pipeline. Starting from raw histology images, cell nuclei are detected and represented as bounding boxes, which are then converted into centroid coordinates (x, y) forming graph nodes. Graph edges are constructed using a k-nearest neighbor (k-NN) approach with $k = 5$, where each cell is connected to its closest spatial neighbors within the same tissue image.

The inductive split is performed at the image level rather than the node level. Specifically, 80% of tissue images are used for training, while the remaining 20% are held out as completely unseen test patients. Node features consist of normalized spatial coordinates (x, y) along with morphological descriptors such as nucleus width and height, standardized to zero mean and unit variance to ensure stable training.

To ensure data privacy, participants are required to encrypt their submission CSV files prior to upload. The evaluation system decrypts and scores predictions in a secure environment, preventing exposure of ground-truth labels from unseen patients during the leaderboard process. Performance is

evaluated using accuracy.

B Human Evaluation Details

This appendix provides additional implementation details for the human evaluation track of GNN-CB.

B.1 Human Implementations

All human participants follow a uniform implementation protocol designed to ensure reproducibility, prevent test-set leakage, and isolate model design choices as the primary source of performance variation across submissions. Each competition enforces a fixed wall-clock budget of three hours of CPU time per task, with hyperparameter tuning capped at five training runs per task and a single final submission per benchmark version to discourage test-set overfitting. All random seeds are fixed to `seed = 25`, training loss curves must be plotted to verify proper convergence and detect overfitting (with early stopping permitted), and the exact source code that produced each submitted prediction file is retained alongside the recorded training time and hardware specifications.

Submitted solutions must be implemented using Graph Neural Network architectures only; convolutional, recurrent, and other non-graph spatial deep-learning models are prohibited, and copy-pasting external solutions is not permitted. LLM assistance is restricted to low-level utility functions and must not expose the model to any task-specific information (e.g., competition name, dataset identity, or feature schema) that could induce data leakage. For each competition, every participant maintains a single documentation file containing the hyperparameter-search strategy used (manual, grid, or random), all configurations attempted within the five-run budget, the recorded training time and hardware specifications, and the exact prompts and LLM-generated snippets reused in the solution.

B.2 Benchmark Composition Attempted by Human Participants

Figure 5 summarizes the composition of the 18 GNN-CB competitions attempted by human participants across four characterizing dimensions: graph category, difficulty tier, task type, and application domain. The benchmark is dominated by homogeneous graphs (55.6%), with the remainder distributed across spatial, generalization, temporal, heterogeneous, hierarchical, and molecular settings. Difficulty tiers are skewed toward Medium (44.4%) and Hard (33.3%) competitions, with only three Easy tasks and a single Expert-level task (C13). Node classification and graph classification together account for 83.3% of the tasks, complemented by one edge-prediction and two graph-regression competitions. Application domains span medical, biomedical, academic, neuroimaging, social, bioinformatics, bioprinting, cybersecurity, finance, and urban data, reflecting the benchmark's emphasis on real-world heterogeneity rather than a single application focus.

B.3 Geographic Distribution of Participants

GNN-CB draws on a globally distributed cohort of human contributors. Figure 6 shows the locations of the 18 participating universities across 12 countries spanning four continents, with each country annotated by the competition(s) co-designed by its contributing team(s). The cohort represents Africa (Algeria, Egypt, Ghana, Tunisia, Uganda), the Americas (Peru, USA), Asia (India, Nepal, Pakistan), and Europe (Turkey, UK), reflecting the benchmark's emphasis on broad geographic and institutional representation rather than concentration in a single research hub. This geographic diversity is intentional: it ensures that the curated competitions, datasets, and domain conventions in GNN-CB are not dominated by the priorities of any single regional research community, and that the human performance baseline reported in Table 2 reflects a heterogeneous expert pool rather than a homogeneous local cohort.

B.4 Human Evaluation Pipeline

Figure 7 summarizes the end-to-end human evaluation workflow used in GNN-CB, from repository fork through encrypted pull-request submission to automated GitHub Actions scoring.

C LLM Evaluation Details

This appendix provides additional implementation details for the LLM evaluation framework used in GNN-CB, including the frozen prompt construction, a representative plan-then-code example, the prompt template, and the orchestration pipeline used during API-based evaluation. Table 4 summarizes the set of LLMs evaluated in GNN-CB, grouped by access type (proprietary vs. open-source) and functional category (chat vs. reasoning). For each model, we report the exact API engine identifier used in our orchestration pipeline

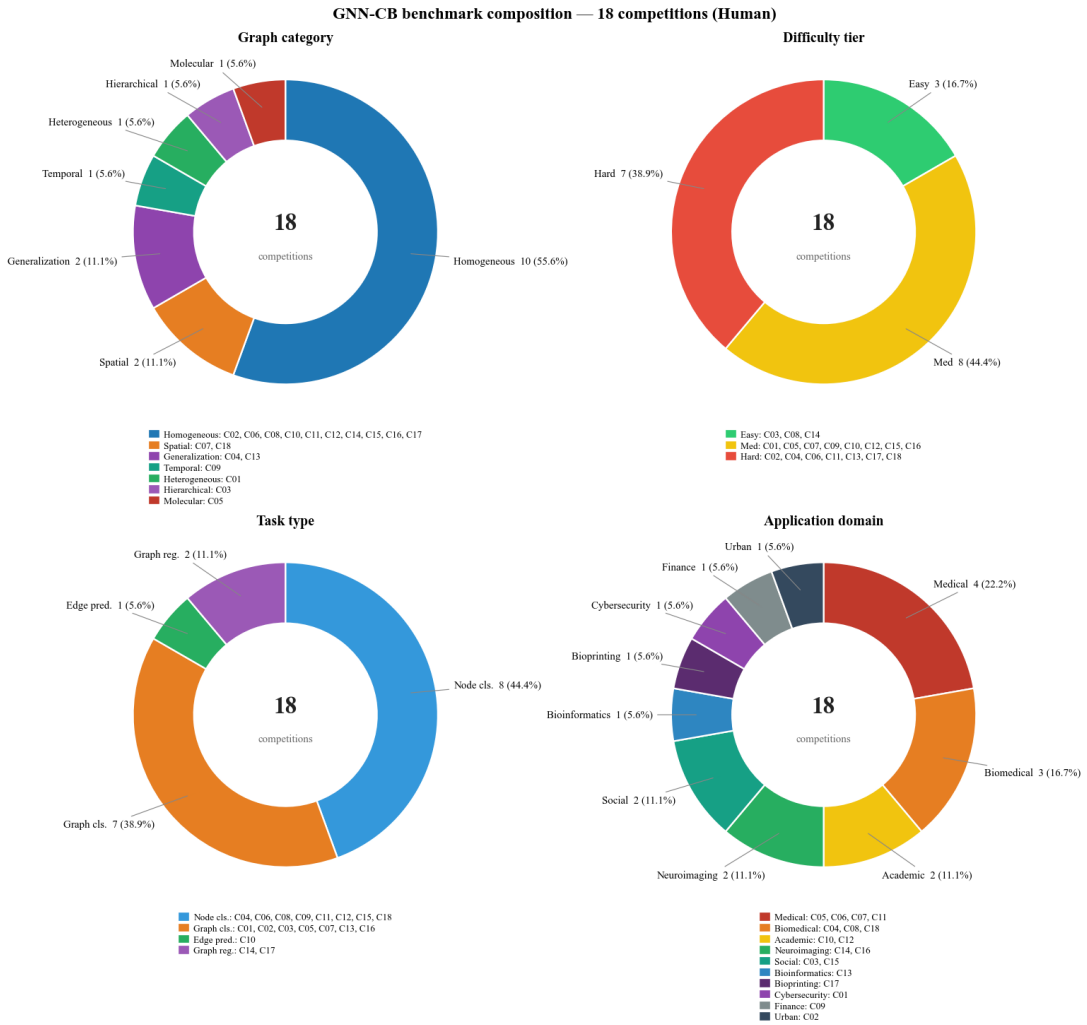


Figure 5: Composition of the 18 GNN-CB competitions attempted by human participants, summarized along four axes: graph category (top-left), difficulty tier (top-right), task type (bottom-left), and application domain (bottom-right). Per-slice counts and competition codes are listed in the legend below each donut.

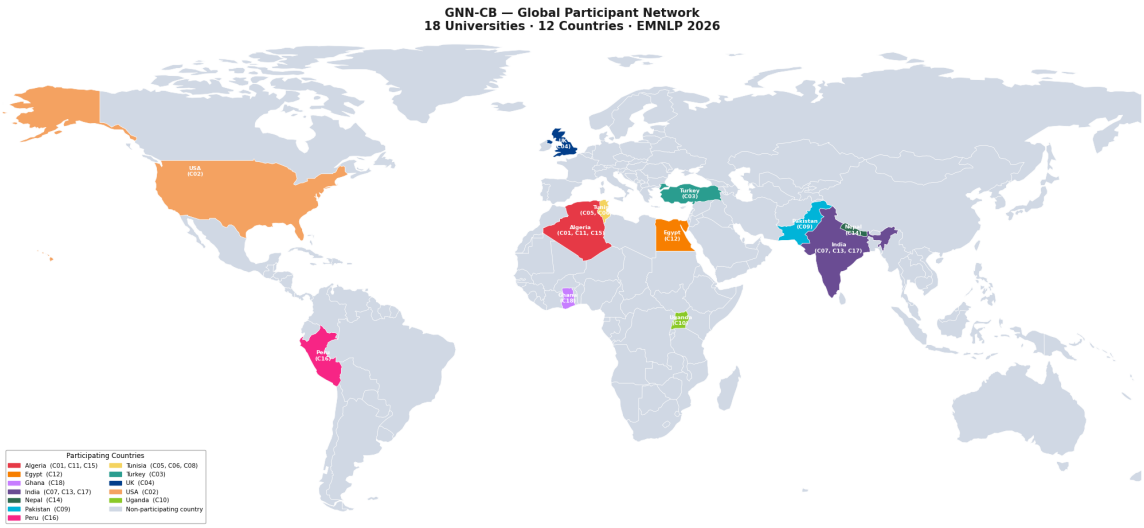


Figure 6: Global distribution of the 18 participating universities that co-designed and evaluated the GNN-CB benchmark. Each highlighted country is annotated with the competition(s) contributed by its team(s); the colour key appears in the legend. The cohort spans 12 countries across Africa, the Americas, Asia, and Europe.

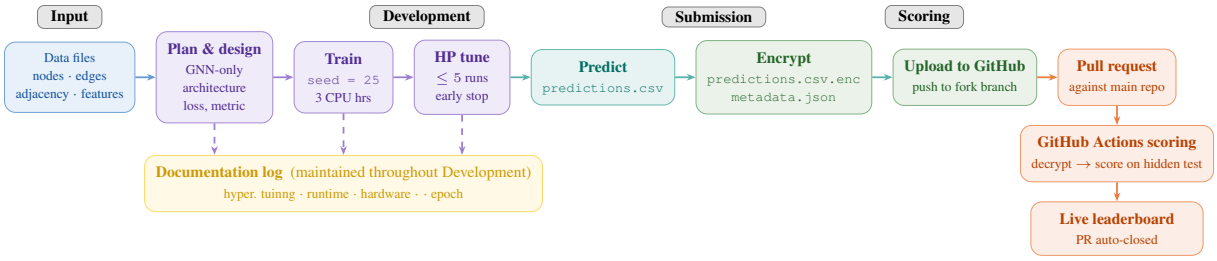


Figure 7: Human evaluation pipeline of GNN-CB. Participants prepare graph data (nodes, edges, adjacency, and feature matrices) from the released data files, then design and train a GNN-only solution under fixed reproducibility constraints (`seed = 25`, three CPU hours, at most five hyperparameter-tuning runs, early stopping permitted). They generate a `predictions.csv` file, encrypt it locally into `predictions.csv.enc` together with a `metadata.json`, push the encrypted submission to their GitHub fork, and open a pull request against the main repository. An automated GitHub Actions workflow decrypts the submission, scores it against the hidden test labels, updates the live leaderboard, and closes the pull request. A documentation log—maintained throughout development—records the hyperparameter-search strategy, runtime, hardware specifications, loss-curve verification, and any LLM-assisted utility usage.

along with a reference to the official provider documentation.

C.1 Frozen Prompt Construction

Figure 8 illustrates how repository components are inserted into structured slots without manual modification, producing an instantiated prompt shared across all competitions.

C.2 Plan-Then-Code Example

Figure 9 presents a representative plan-then-code output for C03 (Real or Fake!), with the high-level reasoning plan on the left and the LLM-generated executable code on the right.

C.3 Frozen Prompt Template

```
You are solving a GNN coding competition.

You will receive:
  (1) competition README
  (2) repository tree
  (3) dataset summary
  (4) submission format
  (5) allowed libraries

Produce:

<plan>
  ...structured reasoning plan...
</plan>

<code>
  ...standalone executable implementation...
</code>

Constraints:
  - runnable as `python solution.py`
  - CPU-only execution
  - no external downloads
  - must generate predictions.csv
```

Listing 1: Simplified frozen prompt template used across all 18 GNN-CB competitions.

To minimize prompt-engineering bias and ensure cross-model comparability, every LLM evaluation in GNN-CB uses the same task-frozen prompt template, instantiated through automatic slot filling from the competition repository. The template encodes the inputs the model receives, the structured output it must produce, and the execution constraints under which the generated solution will be evaluated. Listing 1 reproduces a simplified version of this template; the same string is shared verbatim across all 18 competitions, with only the slot contents (README, repository tree, dataset summary, submission specification, and allowed libraries) varying per task.

C.4 LLM Evaluation Pipeline Listings

```
prompt = instantiate_frozen_prompt(repo
    )
for attempt in range(REPAIR_LIMIT):
                # k <= 5
    response = call_llm(prompt)
                        # API request
    plan     = extract_plan(response)
    code     = extract_code(response)
    write_solution(code)
                                # ->
        solution.py
    result   = execute_solution()
                        # CPU-only sandbox
    if valid_submission(result):
                        # schema + content
        break
    prompt += format_traceback(result)
                # appended for repair
```

Listing 2: Simplified non-agent orchestration loop with bounded execute-and-repair iterations ($k \leq 5$).

The conceptual distinction between the non-agent (API-only) and agent-based evaluation

Table 4: List of LLMs evaluated in GNN-CB, grouped by access type (proprietary vs. open-source). Default decoding parameters are `temperature = 0` and `max_output_tokens = 10K`, applied where the provider's API exposes these settings. API-only (non-agent) runs follow this per-call output budget, while agent-mode runs may consume substantially more tokens across multiple internal tool calls and extended-thinking traces, governed by each agent's native session budget rather than the per-call cap. The **Model Engine Name** column lists the exact API identifier used in our orchestration scripts; the **Source** column links to the provider's official model documentation.

#	Model Name	Model Engine Name	Source
	Proprietary Chat LLMs		
1	Claude Opus 4.6 (Anthropic, 2026a)	`claude-opus-4-6`	Link
2	Gemini-3 Flash (Google DeepMind, 2026)	`gemini-3.0-flash`	Link
	Open-source Chat LLMs		
3	Llama-3.3 70B (Meta AI, 2024)	`Llama-3.3-70B-Instruct`	Link
4	Qwen2.5-Coder 32B (Alibaba Cloud, 2024)	`Qwen2.5-Coder-32B-Instruct`	Link
5	Kimi k2.6 (Moonshot AI, 2026)	`kimi-k2.6`	Link
	Open-source Reasoning LLMs		
6	DeepSeek v4 Pro (DeepSeek-AI, 2026)	`deepseek-v4-pro`	Link
	Proprietary Reasoning LLMs		
7	Gemini 2.5 Pro (Google DeepMind, 2025)	`gemini-2.5-pro`	Link
8	Claude Opus 4.7 (Anthropic, 2026b)	`claude-opus-4-7`	Link
9	GPT-5.4 Codex (OpenAI, 2026)	`gpt-5.4-codex`	Link

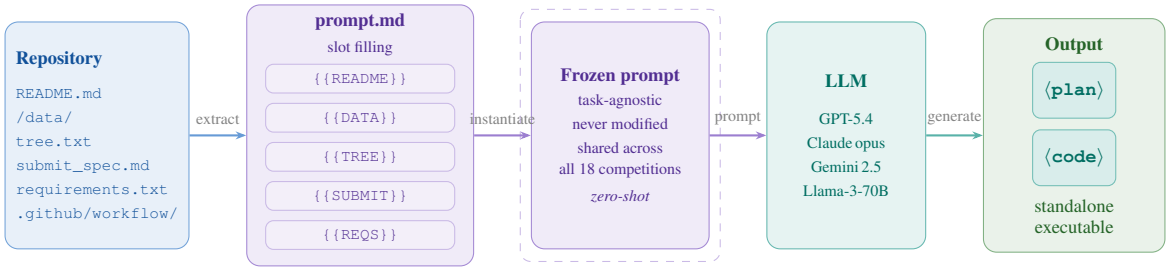


Figure 8: A frozen prompt construction is instantiated from the repo, and then it is passed to the LLM, which outputs a plan and code.

modes is described in Section 3.3.4. Listing 2 reproduces the simplified Python orchestration loop used in the non-agent mode; the agent-based mode has no equivalent participant-side script, as it relies on the autonomous coding system's native repository inspection, code execution, and repair capabilities, governed by each agent's session budget.

C.5 Performance Analysis of LLMs Across Competitions

To assess the comparative performance of leading LLMs against human baselines, we evaluate their capabilities across Easy, Medium, and Hard classification tasks in the GNN-CB benchmark from three perspectives: a gap analysis quantifying performance differences relative to the human-top score, an average human–LLM comparison, and a category-based comparison of how different model types perform across difficulty levels.

Gap analysis: For Easy classification tasks (Figure 10), proprietary models except from GPT 5.4 Codex show minimal deviation, with best-case gaps approaching zero and worst-case gaps remaining moderate. In contrast, open-source models exhibit larger negative gaps across all models, indicating instability and weaker alignment with human-top performance. On Medium tasks (Figure 11), the performance gap widens across all model categories. Proprietary models continue to outperform open-source models in terms of mean gap and exhibit more consistent behavior overall, whereas open-source models show substantially larger negative gaps and higher variability across competitions.

① Plan (zero-shot, plan-then-code)

Step 1 Feature engineering

Concatenate BERT (768) + spaCy (300) + profile (10) → 1078-dim; `StandardScaler` fit on training nodes only.

Step 2 Graph construction

Custom `Dataset` builds per-graph `Data`(x, edge_index, y, root_idx); node indices remapped to local $0..N-1$.

Step 3 Model architecture

Stacked `GATConv` + `global_mean_pool`, concatenated with a dedicated root-node MLP; sigmoid head.

Step 4 Training

Adam ($\eta = 10^{-3}$), `BCELoss`, batch = 64, early stopping on val accuracy (patience = 5).

Step 5 Submission

Inference on hidden test split → write `predictions.csv` with columns (`id`, `y_pred`).

② Code (LLM-generated, abridged)

```
# Step 1: features
x = np.hstack([bert, spacy,
               StandardScaler().fit_transform(profile_train)])
                    # 1078-d

# Step 2-3: model
class GossipGNN(nn.Module):
    def __init__(self, in_ch, hid=128):
        super().__init__()
        self.gat1 = GATConv(in_ch, hid, heads=4)
        self.gat2 = GATConv(hid * 4, hid)
        self.root = nn.Linear(in_ch, hid)
        self.head = nn.Linear(hid * 2, 1)

    def forward(self, x, edge_index, batch, root_idx):
        h = F.elu(self.gat1(x, edge_index))
        h = self.gat2(h, edge_index)
        g = torch.cat([global_mean_pool(h, batch),
                       self.root(x[root_idx])], dim=1)
        return torch.sigmoid(self.head(g))

# Step 4: training loop with early stopping
opt = Adam(model.parameters(), lr=1e-3)
for epoch in range(200):
    train_one_epoch(model, train_loader, opt, BCELoss())
    if early_stop(evaluate(model, val_loader)): break

# Step 5: submission
preds = (model(test_batch).sigmoid() > 0.5).long()
pd.DataFrame({"id": test_ids, "y_pred": preds})\
  .to_csv("predictions.csv", index=False)
```

Figure 9: Plan-then-code paradigm illustrated for C03 (Real or Fake!): a 5-step structured plan (left) and the LLM-generated, executable implementation skeleton (right).

On Hard tasks (Figure 12), the same overall trend persists, although the gap further increases for nearly all models. Notably, the performance of Gemini 2.5 Pro drops considerably, while Kimi k2.6 reaches a comparable level on several tasks. Despite these improvements, the overall gap between LLMs and human participants remains large on hard competitions. Across all difficulty levels, no substantial performance difference is observed between chat and reasoning LLMs.

Average human–LLM comparison: Figure 13 compares the overall average performance of LLMs with the Human Top average across all classification competitions in GNN-CB. LLMs lag behind the Human Top performance. Proprietary models achieve the strongest performance and exhibit the smallest gap to humans, whereas open-source models show lower averages and substantially larger gaps. Within both proprietary and open-source families, no significant performance difference is observed between chat-oriented and reasoning-oriented models. Figure 14 further analyzes LLM and human average scores across Easy, Medium, and Hard competitions on a per-competition basis. Although LLM averages occasionally surpass human averages on individual competitions, humans still achieve higher average scores in 10 out of the 16 classification competitions, indicating that human participants remain more consistently reliable overall. Moreover, as task difficulty increases from Easy to Hard, LLM performance declines similarly to human performance, with larger drops observed on harder tasks.

Category-based comparison: Finally, we analyse model performance by category across the Easy, Medium, and Hard classification tasks using each model's best, mean, and worst accuracy. On Easy tasks (Figure 15), proprietary models achieve high performance with a small variablity between best and worst scores, indicating strong consistency, while open-source models remain competitive but display wider variability. On Medium and Hard tasks (Figures 16 and 17), mean scores decline across nearly all models and the best–worst variance widens, indicating reduced stability as complexity increases. The steep decline in mean and worst-case accuracy highlights the limitations of current foundational architectures under higher complexity.

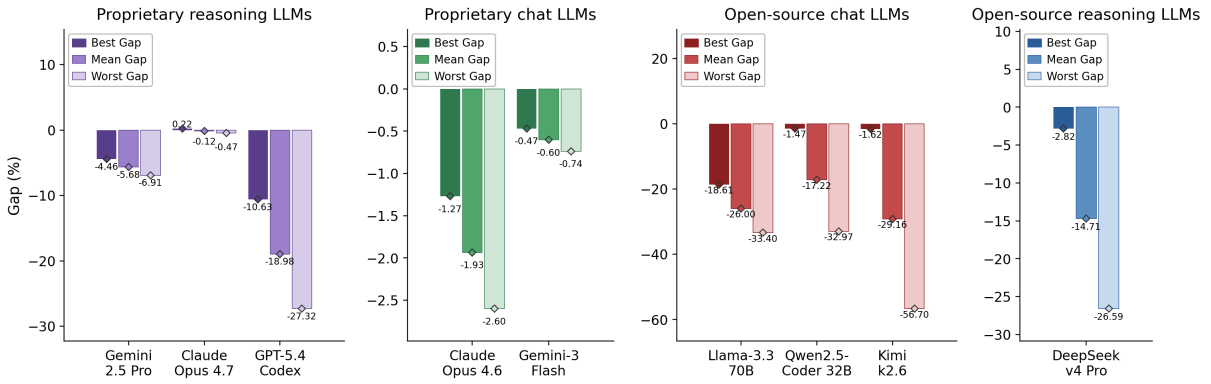


Figure 10: Best, Mean, and Worst Gap performance for leading LLMs on the **Easy** classification tasks of the GNN-CB benchmark, grouped by model family, where the Gap is defined as Gap = Score - Human Top

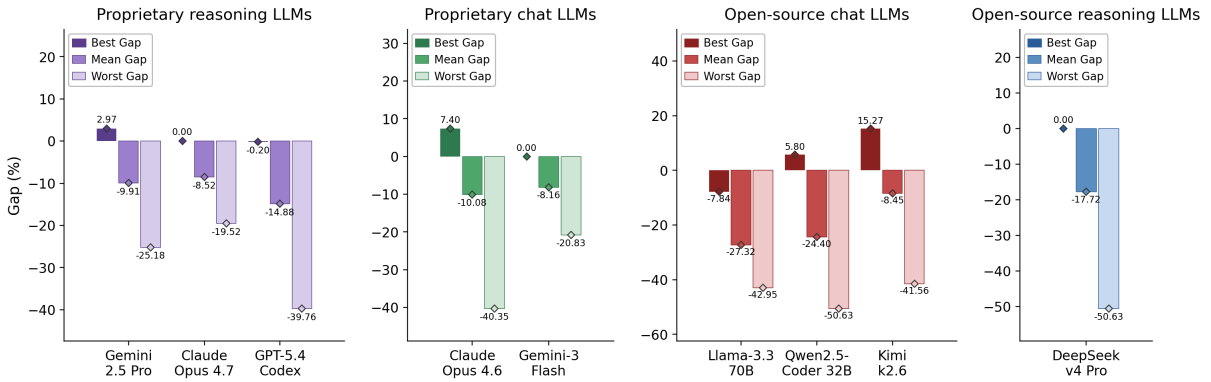


Figure 11: Best, Mean, and Worst Gap performance for leading LLMs on the **Medium** classification tasks of the GNN-CB benchmark, grouped by model family, where the Gap is defined as Gap = Score - Human Top

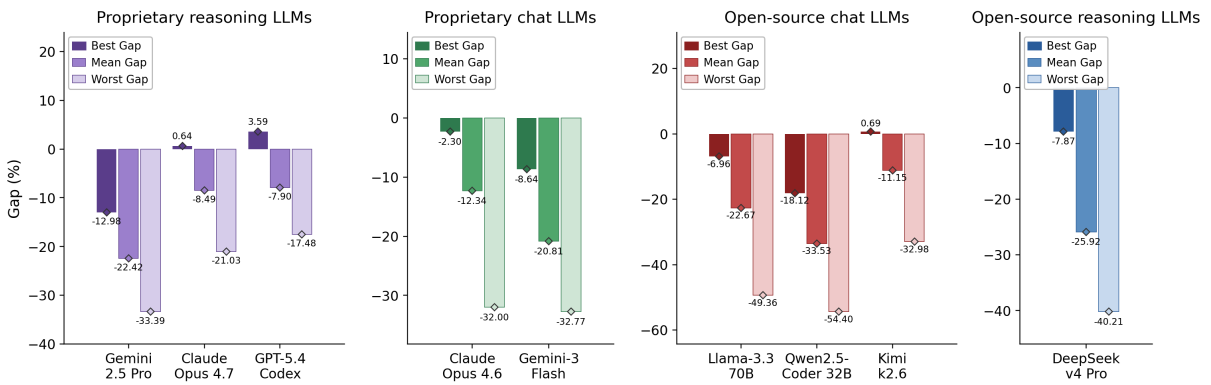


Figure 12: Best, Mean, and Worst Gap performance for leading LLMs on the **Hard** classification tasks of the GNN-CB benchmark, grouped by model family, where the Gap is defined as Gap = Score - Human Top

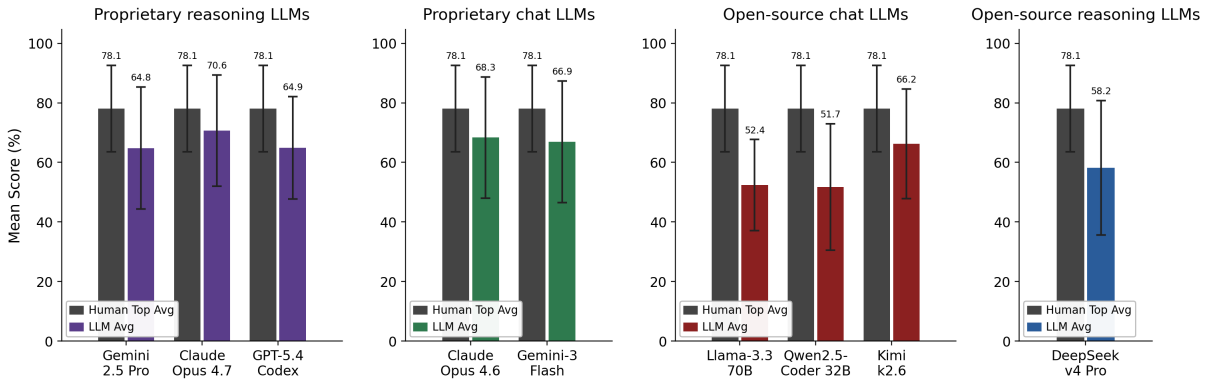


Figure 13: Avg Human Top vs Avg LLM score for leading LLMs on the classification tasks of the GNN-CB benchmark, grouped by model family.

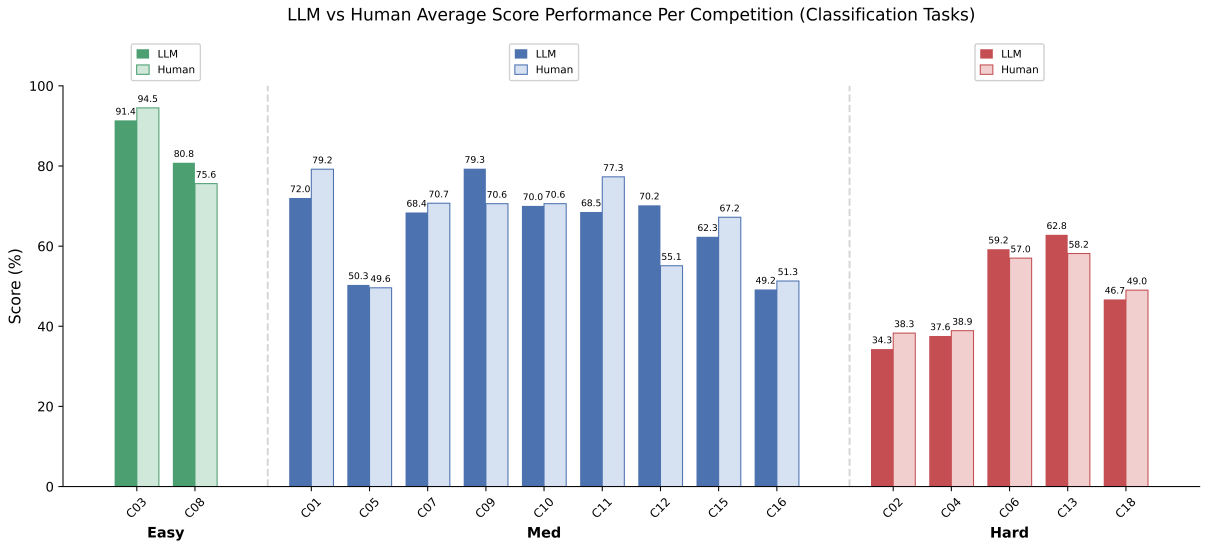


Figure 14: LLM vs Human Average Performance across difficulty levels

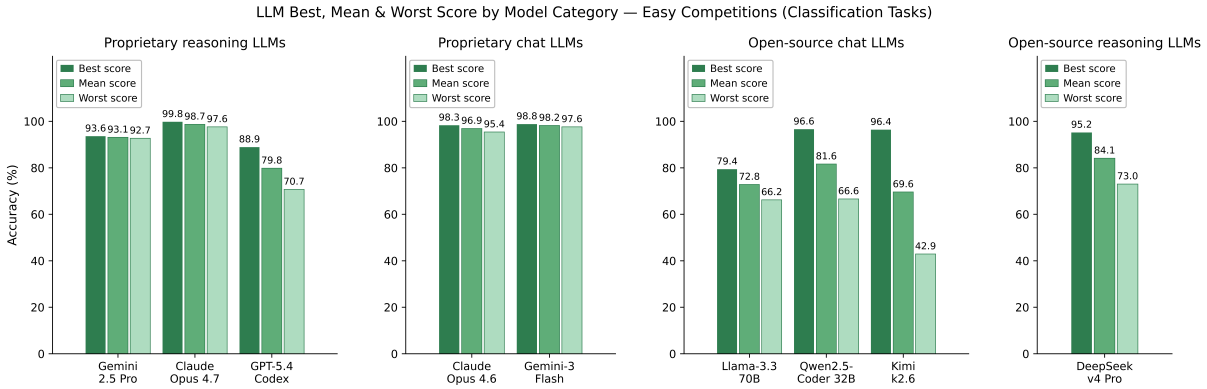


Figure 15: LLM performance grouped by model family on **Easy** classification tasks. Performance metrics capture the best, mean, and worst scores achieved by each model.

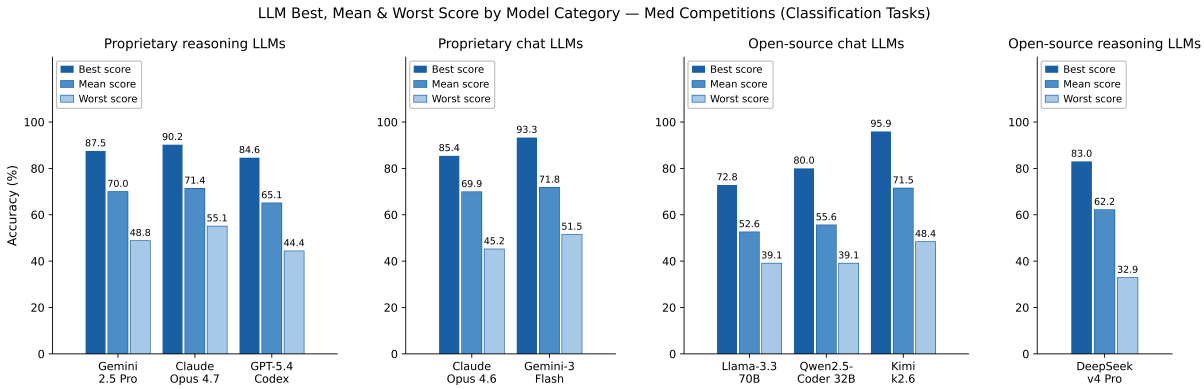


Figure 16: LLM performance grouped by model family on **Medium** classification tasks.

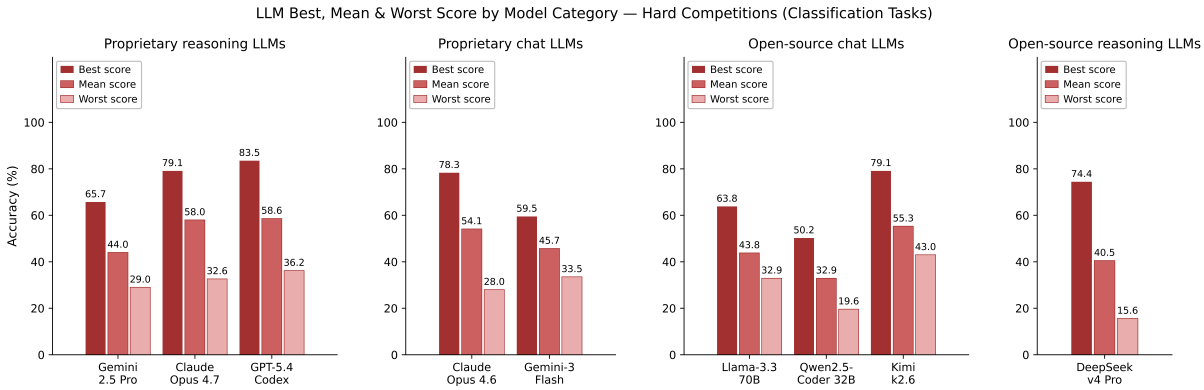


Figure 17: LLM performance grouped by model family on **Hard** classification tasks.