Schema Is the Benchmark: Testing Cross-Schema Transfer in Graph Foundation Models

Anonymous Authors[1]

Abstract

Graph foundation models promise to pretrain once and adapt across graph tasks, domains, and feature spaces, but many evaluations publish a single dataset-specific rendering, making it hard to separate graph transfer from fit to that rendering. We argue that cross-schema transfer is a missing test: can a model preserve task behavior when labels, evidence, and splits are fixed but the graph grammar changes? SchemaWeaver is a benchmark protocol, not a new model; its goal is to make representation-level transfer measurable before leaderboards conflate schema engineering with model capability. It turns relational databases and text-attributed graphs into semantically matched schema views, then requires task contracts, executable leakage audits, adaptation budgets, and metrics for utility, robustness, and calibration. A reproducible synthetic pilot illustrates the measurement issue: under a deliberately schema-local extractor, moving predictive attributes away from the prediction object induces a measurable AUROC range, while an oracle contract control changes the interpretation by spending an explicit adapter budget. The paper gives a concrete path from fixed-schema graph benchmarks to a community stress test for genuinely transferable GFMs.

1. Introduction

Graph machine learning has become a practical interface to relational structure: transactions link customers and merchants, molecules link atoms and bonds, citations link papers and authors, and operational databases link tables through keys (Hu et al., 2020; Robinson et al., 2024). Foundation-model thinking changes the expected contract (Bommasani et al., 2021). Instead of training a graph neural network per task and dataset, a graph foundation model (GFM) should reuse pretraining across node, edge, graph, temporal, and multimodal tasks with little supervision (Wang et al., 2025).

The field is moving quickly: unified graph prompts, text-attributed graphs, LLM-based graph reasoning, relational deep learning, and tabular foundation models all motivate unified evaluation (Liu et al., 2023; 2024; Chen et al., 2023; Fey et al., 2024; Chen et al., 2024; Hollmann et al., 2025). Yet the community still lacks a stress test for the property that makes a model foundational: *transfer across representational choices*. The same prediction problem can be written as a homogeneous graph, a heterogeneous graph, a relational database, a temporal event stream, or a text-enriched property graph. A model that succeeds only on one hand-authored schema is useful, but it has not demonstrated broad graph transfer. Cross-schema transfer should therefore be reported alongside accuracy, compute, and adaptation budget.

We propose SchemaWeaver, a benchmark and reporting protocol for evaluating GFM behavior under controlled schema shifts. SchemaWeaver does not require a new massive pretraining corpus. Instead, it wraps existing relational and graph datasets with semantic-preserving transformations: normalization and denormalization, entity split and merge, edge reification, attribute relocation, temporal bucketing, and natural-language paraphrase. The output is a family of graph views with matched labels and explicit invariance contracts. This gives researchers a sharper question: when a schema changes but the task semantics stay fixed, which models still transfer?

Our contributions are:

1. We identify *schema overfitting* as a central failure mode for GFMs and separate it from ordinary distribution shift.
2. We define an operational cross-schema protocol with controlled graph rewrites, task contracts, leakage audits, adaptation budgets, and normalized robustness metrics.
3. We provide a reproducible pilot study and audit artifact that demonstrate why local evidence access and contract adapters must be reported explicitly.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. SchemaWeaver adds task-contract-preserving schema rewrites to existing benchmark axes.

Benchmark	Fixed	Rewrites	Contract	Budget	Audit
OGB	yes	no	no	no	no
GOOD	yes	no	no	no	no
GraphLand	yes	no	no	no	no
RelBench	yes	no	partial	no	no
OFA/Text-GFM	yes	no	partial	partial	partial
SchemaWeaver	no	yes	yes	yes	yes

Scope. SchemaWeaver is a benchmark proposal; the pilot is a controlled sanity check, not an evaluation of GFMs or fraud models. The seed artifact includes a deterministic generator, schema renderers, a schema-card template, and an audit JSON. A full leaderboard should instantiate the same protocol with public datasets and strong GFM, GNN, relational, tabular, and language-model baselines.

2. Related Work and Gap

Graph foundation models. Recent work has moved from task-specific graph learning toward unified graph models, graph prompting, text-attributed graph representations, and LLM-assisted graph reasoning (Liu et al., 2023; 2024; Chen et al., 2023; 2024; Wang et al., 2025). OFA standardizes graph classification tasks through text-attributed graphs and nodes of interest (Liu et al., 2024); text-space GFM benchmarks expose the importance of language-aligned feature spaces and unified settings (Chen et al., 2024). These efforts address cross-task and cross-domain transfer. SchemaWeaver addresses a complementary question: whether transfer survives when the *same* task is encoded through a different graph schema.

Relational and graph benchmarks. OGB made large-scale graph evaluation more systematic (Hu et al., 2020); graph OOD benchmarks test distributional robustness (Gui et al., 2022); recent industrial graph benchmarks broaden dataset realism (Bazhenov et al., 2024); and RelBench reframes relational databases as graph-learning tasks (Robinson et al., 2024). These benchmarks are natural substrates for SchemaWeaver, but their tasks are usually attached to a fixed canonical representation. SchemaWeaver turns schema construction itself into the controlled variable.

Why this gap matters. In production, graph construction is part of modeling. A benchmark that reports one representation cannot tell whether a GFM learned portable relational structure or merely aligned with a convenient schema. Cross-schema stress testing is therefore not another dataset split; it is a test of whether a model claimed to be foundational has representation-level invariance.

3. The Missing Axis: Schema Transfer

Existing benchmark protocols commonly publish a fixed graph or relational rendering before learning begins (Hu et al., 2020; Gui et al., 2022; Robinson et al., 2024; Chen et al., 2024). Even when datasets vary by domain, the evaluation often keeps the downstream task representation stable: the node types, edge types, feature locations, and label spaces are known. This is convenient, but it hides a practical deployment problem. In real systems, the graph is rarely a canonical object. It is a modeling decision layered over logs, tables, documents, and business rules.

Schema overfitting. We define schema overfitting as high performance on one representation of a task and low performance on another representation with matched labels, allowed evidence, and splits. Unlike standard covariate or graph OOD shifts, where train and test distributions differ (Quinonero-Candela et al., 2009; Gui et al., 2022), schema overfitting changes the rendering map while holding the task contract fixed. A customer–merchant transaction can be encoded as an attributed edge, a transaction node, a fact-table row, or a text-described event; these encodings change local neighborhoods, path lengths, feature sparsity, and message-passing routes while preserving the target concept.

What it is not. Cross-schema transfer is not data augmentation: transformed views are evaluation targets with audited contracts, not extra training examples. It is not ordinary graph OOD: the source phenomenon and task evidence are held fixed while the representation changes. It is not schema matching: SchemaWeaver does not ask models to align databases; it asks whether a claimed graph-transfer mechanism remains valid after a legitimate schema rendering change.

Why GFMs are exposed. GFMs often use text as a unifying space, prompts as task adapters, and large pretraining corpora as sources of reusable relational priors. These design choices make cross-schema generalization plausible. They also make it easy to conflate language memorization, label leakage, and genuine structural transfer. A benchmark should therefore ask whether a model keeps its behavior when the same task is rendered through multiple graph grammars.

4. SchemaWeaver Protocol

SchemaWeaver starts from a source dataset D with entities, relations, attributes, timestamps, optional text, and task labels. A schema view S_i defines how these objects become a learning instance: which objects become nodes, which relations become edges, which attributes become features, and which text is exposed to the model. A transformation

$T_{i \to j}$ produces a new schema view while preserving a task contract $\mathcal{C}_q$.

Objects. Let a source phenomenon be $D = (\mathcal{E}, \mathcal{R}, \mathcal{A}, \tau)$, with entities $\mathcal{E}$, relations $\mathcal{R}$, attributes $\mathcal{A}$, and optional timestamps τ. A schema view is a rendering map $\phi_i : D \mapsto G_i = (V_i, E_i, x_i^V, x_i^E, \theta_i^V, \theta_i^E, \xi_i, \tau_i)$ with node and edge attributes, node and edge types, exposed text, and timestamps. For task q, a contract is

$$\mathcal{C}_q = (I_q, y_q, \mathrm{Ev}_q, \mathrm{Split}_q, \mathcal{L}_q),$$

where I_q is the evaluated item set, y_q is the label map, Ev_q is the allowed evidence manifest, Split_q maps items to train/validation/test, and $\mathcal{L}_q$ specifies leakage tests.

Validity rules. A schema rewrite is admissible only if it satisfies three invariants. **Label equivalence**: for each view there is an evaluated-item bijection $h_i : I_q \to I_{i,q}$ and $y_i(h_i(a)) = y_j(h_j(a))$ for all canonical items a. **Evidence equivalence**: each view exposes the same allowed evidence atoms, even if they are attached to different graph objects; lossy rewrites must be flagged and separately approved. **Split equivalence**: chronological or entity-disjoint splits remain valid after the rewrite. These rules are what separate schema stress tests from arbitrary data augmentation.

4.1. Transformations

SchemaWeaver includes transformations that are common in real graph construction because they preserve task semantics while changing locality, type, temporal, or text interfaces:

Normalization and denormalization. A many-to-many relation can be represented as an attributed edge, a separate event node, or a fact table. The target is unchanged, but the graph diameter and feature locality change.

Entity split and merge. An entity type can be decomposed into subtype nodes, or multiple sparse entity types can be merged with type tokens. This tests whether the model relies on fixed type identities.

Edge reification. Edges with rich attributes become nodes. This is crucial for transactions, recommendations, cybersecurity events, and clinical visits, where the interaction itself carries semantics.

Attribute relocation. Attributes move among nodes, edges, and text fields when the information is semantically attached to more than one object. This probes whether models understand relational provenance.

Temporal bucketing. Fine-grained events are grouped into windows, sessions, or dynamic snapshots. This separates temporal reasoning from brittle timestamp formatting.

Text paraphrase. Node and edge descriptions are rewritten with controlled paraphrases and ontology aliases. Text-space GFMs are expected to be robust to these shifts without leaking labels.

Table 2. Schema rewrites and failure modes they expose.

Rewrite	Stress test	Common risk
Reify edge	feature locality	lost context
Normalize	relational joins	shortcut IDs
Merge types	type invariance	type-token overfit
Move attribute	provenance	hidden leakage
Bucket time	temporal scale	future evidence
Paraphrase text	grounding	label aliases

Table 3. SchemaWeaver adaptation modes.

Mode	Allowed updates	Purpose
Zero-shot	no labels or updates	schema prior
Prompt-only	fixed prompt budget	interface sensitivity
Adapter	declared trainable modules	efficient transfer
Few-label	fixed labels/trials	sample efficiency
Full-tune	all parameters, logged compute	upper bound

4.2. Task Contracts and Leakage Audits

For each transformation, SchemaWeaver publishes the task contract $\mathcal{C}_q$ and a machine-checkable schema card: prediction-unit IDs, view-specific ID maps, allowed columns and text fields, forbidden columns and tokens, timestamp cutoffs, candidate sets, negative-sampling rules, primary metrics, and non-bijective rewrite flags. A transformation is valid only if evaluated items preserve labels and splits, and no view exposes future information or label aliases unavailable in the source.

Leakage audits are mandatory and executable. Each view emits pass/conditional-pass/fail fields for label-token scans, forbidden identifiers, target-edge exposure during message passing, temporal cutoff assertions, future-derived aggregates, and identifier-ablation deltas. This is especially important for LLM and retrieval-augmented systems, and more broadly for benchmark validity under shortcut learning and leakage (Geirhos et al., 2020).

4.3. Evaluation Modes

SchemaWeaver separates model capability from tuning budget:

The central comparison is not only "which model is best," but "which model degrades least when the schema changes under the same adaptation budget." Prompt search, schema-specific preprocessing, validation labels, hyperparameter trials, trainable parameters, wall-clock time, and inference cost are all counted in the budget. We distinguish within-view evaluation, source-to-target transfer, and post-shift adaptation.

Figure 1. SchemaWeaver holds labels, allowed evidence, splits, and leakage rules fixed while changing the graph rendering; reports then separate utility from robustness, validity, calibration, and adaptation budget.

Algorithm 1 SchemaWeaver benchmark construction

Select source dataset D and prediction task q
Define $\mathcal{C}_q$: items, labels, evidence, splits, leakage rules
Choose schema views $\{\phi_i\}_{i=0}^{k}$ and admissible rewrites
for each view ϕ_i **do**
  Render $G_i = \phi_i(D)$
  Verify label, evidence, and split equivalence
  Run leakage audit and identifier ablations
end for
for each model M and adaptation mode b **do**
  Evaluate within-view, source-to-target, or post-shift mode
  Report utility, range, robustness, calibration, and cost
end for

5. Metrics

Let M be a model, S_0 a canonical source schema, and $\{S_1, \ldots, S_k\}$ matched schema views. Let $u_i = u(M, S_i)$ be a bounded, higher-is-better task utility such as AUROC, average precision, accuracy, or mean reciprocal rank, normalized to $[a_q, b_q]$ for task q. SchemaWeaver reports:

Schema robustness.

$$\mathrm{SR}_q(M) = 1 - \frac{1}{k} \sum_{i=1}^{k} \frac{|u_0 - u_i|}{b_q - a_q}.$$

We clip SR_q to $[0, 1]$ for reporting. Losses and lower-is-better metrics must be inverted or normalized before entering SR.

Schema range. Because some rewrites can improve utility while others harm it, we also report $R(M) = \max_{i \in \{0,\ldots,k\}} u_i - \min_{i \in \{0,\ldots,k\}} u_i$. This simple statistic exposes whether reported performance depends strongly on an arbitrary schema choice.

Worst-view drop. For deployment, the worst view often matters more than the average. We therefore report $D_{\mathrm{worst}}(M) = \max(0, u_0 - \min_{i \in \{1,\ldots,k\}} u_i)$ alongside the mean robustness score. A model with strong average transfer but one catastrophic schema failure should not be treated as robust.

Adaptation cost. We report labels used, trainable parameters, wall-clock time, and inference cost. GFMs should be judged by transfer efficiency, not only final accuracy.

Transfer slope. For few-label settings, we report zero-shot utility separately and finite differences $\Delta_{m,n} = (u(n) - u(m))/(\log n - \log m)$ for label counts $m, n \in \{1, 5, 20, 100\}$ per class or relation. This captures whether pretraining actually reduces supervision.

Calibration under schema shift. We report expected calibration error (Guo et al., 2017) and selective prediction curves (Geifman & El-Yaniv, 2017) per schema view. ECE uses a fixed binning rule declared by the task card; for binary pilot results we use 10 equal-width bins on predicted positive-class confidence. Cross-schema robustness without calibrated confidence is insufficient for high-stakes settings.

Fairness under schema shift. For tasks with protected or proxy attributes, reports include subgroup AUROC/AP/ECE, false-positive and false-negative rates at a declared threshold, worst-group utility, and cross-schema subgroup range. These metrics are audit fields, not deployment approval.

Leakage and identifier sensitivity. We include ablations that remove raw identifiers, paraphrase text, mask type

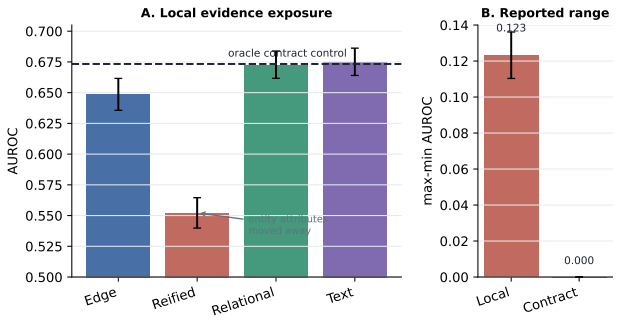

Figure 2. Pilot schema-shift effect. Local LR has unequal local evidence across views; reification moves predictive entity attributes away from the prediction object, while the oracle contract control spends an explicit canonicalization budget on structured views.

Figure 3. The pilot keeps the task contract fixed but changes where evidence lives. A schema-local learner can only use fields at the prediction object, whereas the oracle contract control canonicalizes allowed evidence.

Table 4. Pilot results over 20 random seeds. ECE is uncalibrated ECE for the positive class; extractor-level range and SR are shown in Figure 2.

Extractor	View	AUROC	AP	ECE
Local LR	Edge	0.649±0.013	0.249±0.021	0.322±0.009
Local LR	Reified	0.552±0.012	0.186±0.013	0.341±0.008
Local LR	Relational	0.673±0.011	0.276±0.022	0.312±0.008
Local LR	Text	0.675±0.011	0.276±0.020	0.312±0.008
Contract LR	Edge	0.673±0.011	0.277±0.022	0.313±0.010
Contract LR	Reified	0.673±0.011	0.277±0.022	0.313±0.010
Contract LR	Relational	0.673±0.011	0.277±0.022	0.313±0.010

names, and delete high-risk attributes. A model whose performance collapses only after label-like text is removed is not demonstrating graph transfer.

6. Pilot Artifact

To make the protocol concrete, we include a small reproducible pilot benchmark. The generator creates timestamped customer–merchant transactions with public user attributes, merchant attributes, transaction attributes, and a fraud label sampled from latent user and merchant risks plus observable relational factors. The same prediction problem is rendered as four views: an attributed user–merchant edge graph, a reified transaction-node graph, a denormalized relational table, and a natural-language transaction description. Each run uses 1,200 users, 400 merchants, and 25,000 transactions; results are averaged over 20 seeds. Splits are chronological by sorted transaction time, with the first 60% used for training, the middle 20% left as an unused temporal gap, and the last 20% used for testing. Identifiers are excluded from the learner, and the artifact emits a JSON audit file.

The pilot compares two deliberately simple learners. **Local LR** trains logistic regression on the features directly exposed at the prediction object in each view. Edge exposes transaction attributes, merchant category, and past endpoint counts; reified exposes only transaction-node attributes; relational exposes joined user and merchant attributes; text exposes a controlled serialization of selected public attributes. **Contract LR** is an oracle canonicalization control: it recovers a fixed contract feature frame across structured views, then trains the same logistic regression. Its invariance is by design, not an empirical robustness result; it is included to show how explicitly budgeted canonicalization changes the meaning of a score. It is not a proposed model and should not be read as evidence of GFM performance.

Table 4 shows the intended failure mode. Local LR drops on the reified graph because the local extractor is denied entity attributes that remain present only on adjacent nodes. Contract LR has zero measured schema range across the structured views by construction because it spends an oracle canonicalization budget to recover the contract frame. Without cross-schema reporting, model quality can be partly an artifact of graph construction or unreported adaptation.

What the pilot demonstrates. The pilot demonstrates, in a toy controlled setting, that a schema-local baseline can produce different scores when semantically related information is placed at different graph objects. It does not prove that a particular architecture is superior, that a contract adapter is desirable, or that the synthetic fraud process captures deployment risks. The right community next step is to instantiate the same protocol on public relational and text-attributed graph datasets with stronger baselines: GNNs, relational deep learning models, graph prompt models, text-space GFMs, LLM pipelines, and tabular foundation models.

7. Initial Task Suite

SchemaWeaver can be bootstrapped from four task families that map naturally onto the workshop's themes.

Relational prediction bridges graph and tabular foundation models through churn, purchase, fraud, and item adoption tasks. Views compare relational tables, entity graphs, and event-node graphs. **Text-attributed node classification** uses citation, product, and knowledge graphs to test whether text-space GFMs transfer across feature descriptions and structural encodings. **Temporal link prediction** exposes

whether a model handles schema rewrites without violating causal time splits. **Graph-level risk scoring** covers molecular, cybersecurity, and financial subgraphs where attributes and motifs move across nodes and edges.

The suite is intentionally modular. A workshop version can start with public datasets and a small number of high-quality transformations, then grow into a community leaderboard once task contracts and leakage audits are stable. A strong first release would wrap RelBench-style databases with edge, event-node, and table views; add text-attributed variants for selected entity and relation descriptions; and provide scripts that regenerate every view from a single source dataset.

8. Baselines and Research Questions

SchemaWeaver is designed to compare families rather than crown a single architecture. Strong baselines should include task-specific GNNs, graph prompt models, text-space GFMs, LLM-assisted graph reasoning systems (Chen et al., 2023), relational deep learning models (Fey et al., 2024; Robinson et al., 2024), and tabular foundation models (Hollmann et al., 2025) applied to engineered relational views. This creates useful tension: GNNs may win when structure is clean, tabular FMs may win on compact relational views, and LLM pipelines may win when text semantics dominate. The interesting model remains competitive as the schema changes.

The benchmark also creates concrete research questions:

- Which pretraining objectives, including masked-attribute or masked-structure objectives (Hou et al., 2022), learn invariants that survive edge reification and entity splitting?
- Do text descriptions help because they align schemas, or because they leak label semantics?
- Can tabular foundation models and GFMs be composed so that tables provide local feature priors and graphs provide relational context?
- How much of cross-domain transfer is actually cross-schema transfer?
- Which positional and structural encodings remain stable under relational rewrites?

9. Roadmap to a Community Benchmark

SchemaWeaver can be adopted incrementally. The first milestone is a *schema card* for each dataset: source tables or graph objects, target definitions, split policy, allowed evidence, forbidden leakage channels, and supported rewrites. The second is a public renderer that regenerates all schema views from a single source artifact. The third is a leaderboard that reports not only a primary metric, but also schema range, schema robustness, adaptation budget, and calibration.

Table 5. Concrete v0 release plan. Each task requires renderers, schema cards, and audit outputs.

Source	Task	Views	Primary check
RelBench DB	relational pred.	table/edge/event	evidence map
Citation TAG	node class.	text/node/type	token mask
Temporal graph	link pred.	edge/event/window	causal split
Cyber/fraud logs	risk scoring	edge/event/text	ID ablation

Table 6. Minimum report card for a SchemaWeaver leaderboard entry.

Field	Purpose
Primary utility	task performance
Schema range/drop	failure under rewrites
Adaptation mode	budget accounting
Leakage audit	validity of evidence
Calibration	confidence under shift
Runtime/cost	practical transfer

This reporting format creates incentives that match the GFM agenda. A large pretrained model that wins on one canonical graph but fails under edge reification is not yet robust. A smaller model with lower peak AUROC but stable behavior across views may be preferable for deployment. A text-space GFM that improves after natural-language paraphrase but collapses when label-like tokens are masked should be treated as a leakage risk rather than a transfer success.

10. Discussion

SchemaWeaver is deliberately a protocol before it is a leaderboard. The GFM community needs shared rules for what counts as transfer. Without them, the field risks optimizing for increasingly large pretraining pipelines while leaving the deployment bottleneck untouched: adapting to messy, changing schemas. Cross-schema evaluation would make benchmark results more comparable, highlight useful failure modes, and invite collaboration across graph learning, relational databases, LLMs, and tabular foundation models.

There is a practical reason to start now. Relational and graph datasets already exist, many transformations are deterministic, and the GFM community is still early enough to shape its evaluation norms. A workshop benchmark can be built quickly if the community agrees on task contracts, leakage audits, and adaptation budgets. SchemaWeaver provides that scaffold.

11. Threats to Validity and Responsible Use

Schema-preserving transformations are never perfectly neutral. Some rewrites change inductive biases in ways that favor particular model classes, and some source datasets may

contain historical or demographic biases that become harder to inspect after graph construction. Our pilot is synthetic and uses linear models, so it should be treated as a protocol sanity check rather than empirical evidence about current GFMs. A real benchmark release must include public datasets, nontrivial GFM baselines, ablations over text leakage and identifiers, and independent review of each schema contract. Models evaluated under this protocol should not be deployed in high-stakes domains without domain validation, fairness analysis, and monitoring for schema drift. Benchmark scores are not deployment approval; fraud, finance, medical, and cybersecurity uses require subgroup harm analysis, human review, monitoring, and rollback criteria.

The strongest version of SchemaWeaver will require community participation: dataset maintainers to validate schema cards, model builders to submit under comparable adaptation budgets, and domain experts to flag invalid rewrites. The present paper should therefore be read as a concrete benchmark specification and seed artifact, not as the final benchmark itself.

LLM Use Disclosure

GPT-5.5 was used for low-level grammar and wording edits. The human authors are responsible for all ideas, claims, citations, experiments, and final content.

References

Bazhenov, G., Platonov, O., and Prokhorenkova, L. GraphLand: Evaluating graph machine learning models on diverse industrial data. *arXiv preprint arXiv:2409.14500*, 2024.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Chen, Z., Mao, H., Li, H., Jin, W., Wen, H., Wei, X., Wang, S., Yin, D., Fan, W., Liu, H., and Tang, J. Exploring the potential of large language models (LLMs) in learning on graphs. *arXiv preprint arXiv:2307.03393*, 2023.

Chen, Z., Mao, H., Liu, J., Song, Y., Li, B., Jin, W., Fatemi, B., Tsitsulin, A., Perozzi, B., Liu, H., and Tang, J. Text-space graph foundation models: Comprehensive benchmarks and new insights. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Fey, M., Hu, W., Huang, K., Lenssen, J. E., Ranjan, R., Robinson, J., Ying, R., You, J., and Leskovec, J. Position: Relational deep learning – graph representation learning on relational databases. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 13592–13607, 2024.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, 2017.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020.

Gui, S., Li, X., Wang, L., and Ji, S. GOOD: A graph out-of-distribution benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1321–1330, 2017.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 594–604, 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, 2024.

Liu, Z., Yu, X., Fang, Y., and Zhang, X. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pp. 417–428, 2023.

Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (eds.). *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2009.

Robinson, J., Ranjan, R., Hu, W., Huang, K., Han, J., Dobles, A., Fey, M., Lenssen, J. E., Yuan, Y., Zhang, Z., He, X., and Leskovec, J. RelBench: A benchmark for deep learning on relational databases. *arXiv preprint arXiv:2407.20060*, 2024.

Wang, Z. et al. Graph foundation models: A comprehensive survey. *arXiv preprint arXiv:2505.15116*, 2025.