GraphWorld: Fake Graphs Bring Real Insights for GNNs

Anonymous Author(s)
Affiliation
Address
email

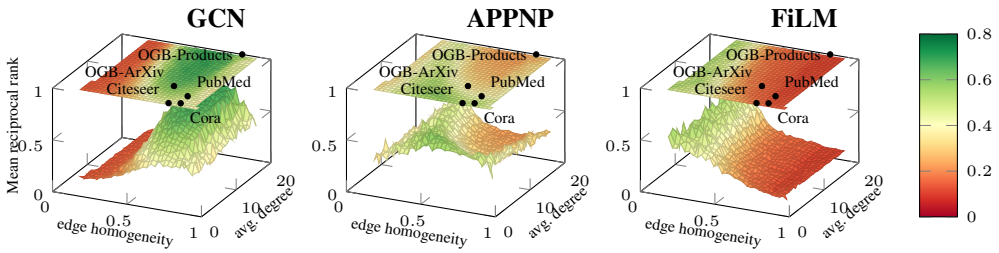

Figure 1: We present relative performance results of GCN [22], APPNP [23], and FiLM [5] across 50,000 synthetic node classification tasks. The x and y axes group the synthetic graphs by their structural properties, while the z-axis shows the mean reciprocal-rank (MRR) relative to other baselines. Standard GNN benchmark datasets (Cora, OGB, etc.) exist in graph property space where model rankings do not change.

Abstract

Despite advances in the field of Graph Neural Networks (GNNs), only a small number (~5) of datasets are currently used to evaluate new models. This continued reliance on a handful of datasets provides minimal insight into the performance differences between models, and is especially challenging for industrial practitioners who have datasets which are very different from academic benchmarks.

In this work we introduce `GraphWorld`, a novel methodology and system for benchmarking GNN models on an arbitrarily-large population of *synthetic* graphs for *any* conceivable GNN task. `GraphWorld` allows a user to efficiently generate a *world* with millions of statistically diverse datasets. It is accessible, scalable, and easy to use. `GraphWorld` can be run on a single machine without specialized hardware, or it can be easily scaled up to run on arbitrary clusters or cloud frameworks. Using `GraphWorld`, a user has fine-grained control over graph generator parameters, and can benchmark arbitrary GNN models.

We present insights from `GraphWorld` experiments on the performance of thirteen GNN models and baselines over millions of benchmark datasets. We show that `GraphWorld` efficiently explores regions of benchmark dataset space uncovered by standard benchmarks, revealing comparisons between models that have not been historically obtainable. Using `GraphWorld`, we also are able to study in-detail the relationship between graph properties and task performance metrics, which is nearly impossible with the classic collection of real-world benchmarks.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

1 Introduction

Graph Neural Networks (GNNs) have extended the benefits of deep learning to the non-Euclidean domain, allowing for standardized and re-usable machine learning approaches to problems that involve relational (graph-structured) data [46]. GNNs now admit a wide range of architectures and possible tasks, including node classification, whole-graph classification, and link prediction [7]. With this growth has come increased calls for proper GNN experimental design [35, 48], refreshed benchmark datasets [17], and fair comparisons of GNN models in reproducible settings [13, 29].

Despite the proliferation of new GNN models, only a few hand-picked benchmarked datasets are currently used to evaluate them [17]. The limited scope of these datasets introduces a number of problems. First, it makes it hard for practitioners to infer which models will generalize well to unseen datasets. Second, new architectures are proposed only when they beat existing methods on these datasets, which can cause *architectural overfitting* [33, 31]. Finally, recent efforts at expanding the diversity of available GNN benchmark data have focused on size – increasing the cost of evaluating models without significantly increasing the variation of graphs considered.

In this work, we introduce `GraphWorld`, the first tunable, scalable, and reproducible method for analyzing the performance of GNN models *on synthetic benchmark data* for *any given GNN task* (i.e. all {un/semi}supervised node/graph problems [46]). With its ability to generate a vast and diverse "world" of graph datasets for any specified task, `GraphWorld` allows comparisons between GNN models and architectures that are not possible with the handful of standard graph datasets on which the current literature depends. As seen in Figure 1, GNN models change sharply in performance ranking when tested on `GraphWorld` synthetic datasets that are *distant* in graph property space from standard real-world datasets. This paper's contributions are:

1. **Problem Formulation**. We pose the question: how can we measure GNN model performance across graph datasets with high statistical variance? GNN datasets, however well-maintained, are limited in scope, and can be computationally inaccessible to the average researcher.
2. **Methodology**. We provide `GraphWorld`, a graph sampling and GNN evaluation procedure which is capable of testing GNNs on task datasets beyond the scope of any existing benchmarks.
3. **Insights**. We use `GraphWorld` to conduct large-scale experimental study on over 1 million graph datasets for each of three GNN tasks – node classification, link prediction, and graph property prediction. We provide a novel method to explore the GNN model performance across all locations in the graph worlds that we generate.

Importantly, the `GraphWorld` methodology enables experimental analysis of GNNs across *any* task that can be expressed in the theoretical framework we introduce in Section 3, including tasks not mentioned above such as node regression. Therefore, particular random graph models for synthetic dataset generation are not the focus of our investigation.

The rest of this paper is as follows. First we discuss recent work and current challenges in benchmarking GNNs. We then formally propose `GraphWorld` as a novel benchmarking system that features many advantages unavailable to GNN experiments that depend only on natural datasets from the literature. Next, in Section 4, we demonstrate `GraphWorld` on node classification, link prediction, and graph property prediction tasks, and show brand-new aggregate performance metrics on GNN models, some of which raise surprising conclusions about prior work and common-sense intuitions. We close with a discussion about the `GraphWorld` platform, including its scalability and accessibility compared with large-scale experiments on real-world graphs.

2 GNN Experiments: Past and Future

Progress on GNN architectures is recorded, in large part, by comparing the empirical performance of proposed and existing architectures on particular tasks. In these empirical studies, each task is associated with a number of datasets on which GNN models attempt to perform well.

In general, such experiments are used to arrive at some insights about how new architectures will perform in realistic scenarios. Most studies feature 1-3 tasks and 1-5 datasets per-task. Some datasets contain many graphs, such as a datasets containing molecules represented as atom graphs [30]. Multiple-graph datasets have primarily been used to train GNNs on the task of classifying whole

graphs, or estimating some property of them. Such datasets can be considered equivalent to a dataset with only one (usually much larger) graph, given that GNN performance on each dataset is measured by a single task and a single set of metrics. Even when the dataset collection itself is large, such as TUDataset [30], which currently contains more than 130 graph collections, researchers can not utilize them all due to the lack of space, limiting the reporting to 3–6 most common ones.

These concerns are not specific to the field of GNNs — the broader machine learning community has identified problems in benchmarking protocols and reporting in other subfields [27, 10, 33]. New benchmarks are being actively created in nearly all areas of machine learning [24, 44, 9, 37]. In the field of GNNs, recent comparative benchmarking studies [20, 28, 13, 49, 12] limit themselves to just a few datasets, mainly targeting fair experimental settings and fair hyperparameter tuning.

Possibly due to the existence of well-studied random graph models such as the Stochastic Block Model [19, 1], there has been a very recent trend of featuring small synthetic datasets in GNN research, to tease apart model differences that would be harder to observe on standard datasets [41, 12, 42, 34, 51, 8]. However, to date, there is no generalized methodology for producing synthetic, tunable *populations* of GNN task datasets at-scale, nor a concept of how to analyze GNN performance on such populations. This is the main problem we aim to solve with `GraphWorld`.

2.1 Pitfalls of Standard GNN Benchmarks

In this paper, we question if the status quo of GNN evaluation as described above, is enough to measure progress in the field. While there has been much recent work on improving and standardizing GNN benchmark datasets, relying only on a handful of graph datasets over time is detrimental to the field, for the following three reasons:

Inadequate generalization. Each curated graph dataset is just one point in the space of all possible datasets that can be associated with the particular GNN task at-hand. The graph (or graphs) in a particular dataset may have properties that favor some GNN models over others, whereas yet-unseen graphs will have different characteristics that could reverse any insights made from the singular trial.

Incremental overfitting. As it is in many machine learning subfields, GNN task datasets are successively re-used across papers, to accurately measure incremental improvements of new architectures. However, this can easily cause, over time, overfitting of new architectures to the datasets, as observed for NLP tasks [31] and computer vision tasks [33]. This effect will be especially pronounced if the main collection of benchmark graphs have similar structural and statistical properties.

Un-scalable development. In recent years, there has been a particular focus on scalability in GNN research. The Open Graph Benchmark [17] increased the size of experiment-friendly benchmark citation graphs by over 1,000x the number of nodes. From one perspective, this can only be natural as computing capabilities grow and graph-based learning problems become increasingly data-flush. On the other hand, while the availability of giant graphs is important for testing GNN software, platforms, and model complexity, it is not obvious that giant graphs are needed to test GNN *accuracy* or *scientific usefulness*. As the field's benchmark graphs become ever-larger, standardized graph datasets for testing GNN expressiveness become less accessible to the average researcher.

3 GRAPHWORLD

The three problems described in the previous section – inadequate generalization, incremental overfitting, and un-scalable development – are *inherent* to the need of GNN researchers to benchmark on datasets sourced from the real world. Any handful of such datasets will be necessarily limited, and it is in the best interest of researchers to re-use well maintained datasets from prior work to show how novel methods improves over existing architectures in the field.

Given that these problems can not be solved with the status quo approach to benchmark design, a *complementary* system is necessary to fill in the gaps that re-usable benchmark datasets cannot. For this, we propose `GraphWorld`: a distributed framework for simulating diverse *populations* of GNN benchmark datasets, tuning and testing an arbitrary number of GNN models on the population, and extracting population-level insights from the logs of the system. As we explain in this section, and demonstrate with our experiment results, `GraphWorld` provides *generalizable* GNN insights in a *scalable* manner that is accessible to researchers with low computational resources.

3.1 Graph Generation

The core component of `GraphWorld` is the simulation of GNN test datasets using attributed-graph and label generators. Each test dataset is a realization of a parameterized probability distribution $\mathcal{P}(\pi_1, \pi_2, \ldots)$ on $\mathcal{D} = \mathcal{G} \times \mathcal{F} \times \mathcal{L}$, where $\mathcal{G}$ is a collection of graphs, $\mathcal{F}$ is a collection of features, and $\mathcal{L}$ is a collection of labels. (This formulation supports graph classification datasets, which can be represented as a single graph of disjoint graph examples.) The space of possible test datasets $\mathcal{D}$ is fully determined by regions of generator parameters $\pi_1 \in \Pi_1, \pi_2 \in \Pi_2, \ldots$ provided by the user to the workers via the manager. Each worker in a `GraphWorld` pipeline generates a *single* realization $D \in \mathcal{D}$ by sampling a generator parameter set $(\pi_1, \pi_2, \ldots)$, and then sampling a single dataset from $\mathcal{P}(\pi_1, \pi_2, \ldots)$. Note: each dataset sampled is actually a combination $D = [D_{train}, D_{test}]$ of training data and testing data for the GNN models.

3.2 Training, Testing, and Evaluation

The `GraphWorld` method simulates a pre-specified number of GNN test datasets $D_1, D_2, \ldots \in \mathcal{D}$, in the manner described above, and trains and tests an arbitrary list of GNN models on each dataset. Note that the particular task is fully-generalizable beyond node classification; we demonstrate link prediction and graph property prediction in Section 4. Similarly to the test data distributions, the hyperparameters of GNN model m (for a particular dataset) is determined by a sample or specification $(h_{m1}, h_{m2}, \ldots) \in \mathcal{H}_m = H_{m1} \times H_{m2} \times \ldots$. `GraphWorld`'s complete set of inputs are:

- **Models**: list of models $m_1, m_2, \ldots$ and associated hyperparameter spaces $\mathcal{H}_1, \mathcal{H}_2, \ldots$.
- **Task formulation**: space of possible datasets $\mathcal{D}$, and a probability distribution $\mathcal{P}(\pi_1, \pi_2, \ldots)$ defined on $\mathcal{D}$.
- **Task metric**: test accuracy function `EvalMetric`(m, D).
- **Size** N: number of datasets to sample on the graph world.

With these inputs, the `GraphWorld` system is trivially parallel over N location on the graph world. We give more architecture details in Appendix A. In Section 4, we specify the above inputs for three different `GraphWorld` pipelines, and discuss the results of those pipelines in Section 5.

3.3 Efficient exploration of graph worlds

A key aspect of `GraphWorld` is the ability to analyze the response of GNN models to the task generator parameters described previously. However, not all configurations $(\pi_1, \pi_2, \ldots)$ in parameter space will provide equivalent insights. As a trivial example, extremely small values of $\pi_i =$ number of vertices (such as 2 or 3) will clearly not be useful to exploring other parameters, like edge density, or the skewness of the degree distribution, since GNN models will either perform poorly or perfectly on trivially-sized graphs regardless of other parameters.

We provide a methodology to mine a large (random) sample of `GraphWorld` generator configurations for the most "affective" configuration, meaning that deviations from that configuration affect GNN model performance most strongly. Assume we have generated a graph world for a given task T with generator parameter space Π, and at each location k at the graph world we have a sampled configuration $\hat{\Pi}_k \in \Pi$ and an average GNN test metric z_k. Conceptually, a sampled configuration $\hat{\Pi} = (\pi_1, \pi_2, \ldots)$ is most affective if for every $\pi_i \in \hat{\Pi}$, changing the value of any other parameter π_j produces variance in the test metrics of GNN models.

To find such a configuration, we perform marginal optimization on the space of parameters Π. Using the samples $\{(\hat{\Pi}_k, z_k)\}$ on an initial run of a `GraphWorld` pipeline, we find a (locally) optimal setting for each π_i in the following manner. We first bin each dimension of Π into a fixed number of quantile bins. Then for each quantized value $\pi_i = x$, we compute the average F statistic [38] between the other parameter values π_j and the test metric z (on graph world locations where $\pi_i = x$). We then set π_i to the x value that produced the highest F statistic. This produces an optimal generator configuration from which we can efficiently sample a smaller but still-interesting graph world. In Section 4 we describe how we apply this technique to `GraphWorld` experiments.

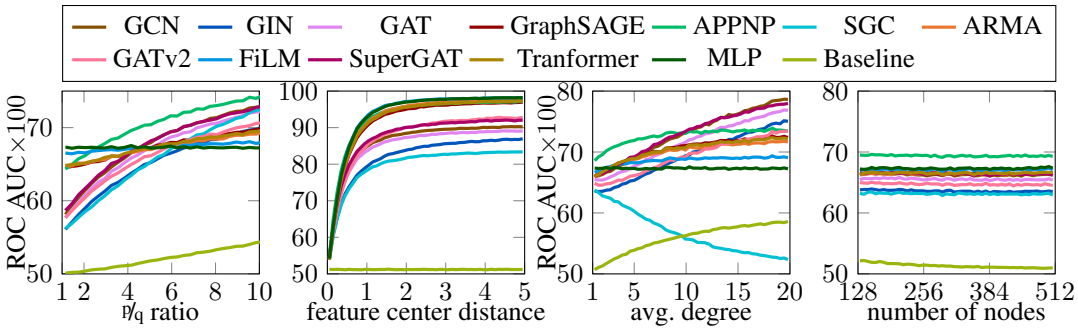

Figure 2: `GraphWorld` node classification results (mode 2).

4 Experimental Design

In this section, we introduce novel experimental design for the `GraphWorld` method, showing how to efficiently sample a useful part of any graph world. We describe three tasks – node classification, graph classification, and link prediction – and how they are generated in various graph worlds. We also list the GNN models tested with the `GraphWorld` applications, and present novel `GraphWorld` modes of hyperparameter tuning and inference.

4.1 Hyperparameter Optimization

GNN hyperparameter tuning is essential for understanding model performance, and is an aspect of GNN experimentation that can be efficiently explored with `GraphWorld`. A `GraphWorld` pipeline can be run in one of three hyperparameter modes:

Mode 1 Each model m is trained and tested with a random draw $(h_{m1}, h_{m2}, ...) \in \mathcal{H}_m$, its hyperparameter configuration space.

Mode 2 Assume a `GraphWorld` pipeline has already been run in Mode 1. For any model m, let $\hat{H}_i$ be the i-th unique configuration sampled (at any location in the graph world). Let $\mathcal{D}_i$ be the collection of `GraphWorld` datasets for which $\hat{H}_i$ was sampled for m. Mode 2 is to run another `GraphWorld` pipeline with the best config H^*_m defined as:

$$H^* = \arg\max_{\hat{H}_i} |\mathcal{D}_i|^{-1} \sum_{D \in \mathcal{D}_i} \texttt{EvalMetric}(m(\hat{H}_i), D). \tag{1}$$

Intuitively, we pick the hyperparameters that achieve the best average performance across all `GraphWorld` samples.

Mode 3 Each model m receives a budget of t tuning rounds, and the hyperparameter configuration which performed best on a held-out validation set is used to compute the test metric.

We provide more model and hyperparameter details in Appendix B.1.

4.2 Tasks

Here we describe three tasks that will be explored with three separate `GraphWorld` pipelines. We provide the ranges of generator parameters in Appendix B.2. In Section 5 we show results from each of tasks run in every hyperparameter tuning mode described previously in Section 4.1.

4.2.1 Node Classification (NC)

In this `GraphWorld` experiment, we generate graphs using the Stochastic Block Model (SBM). First, node labels (classification targets) are generated from a multinomial distribution, which define the node clusters. Edges are generated as Bernoulli random variables following within-block probability p and between-block probability q ($p \geq q$). Node features are generated from a within-cluster multivariate Normal distribution, with unit (diagonal) covariance, and cluster centers are drawn from a prior multivariate Normal. The variance of the prior controls the degree of separation between the cluster feature centers. The number of clusters, the cluster size, and the power law of the expected

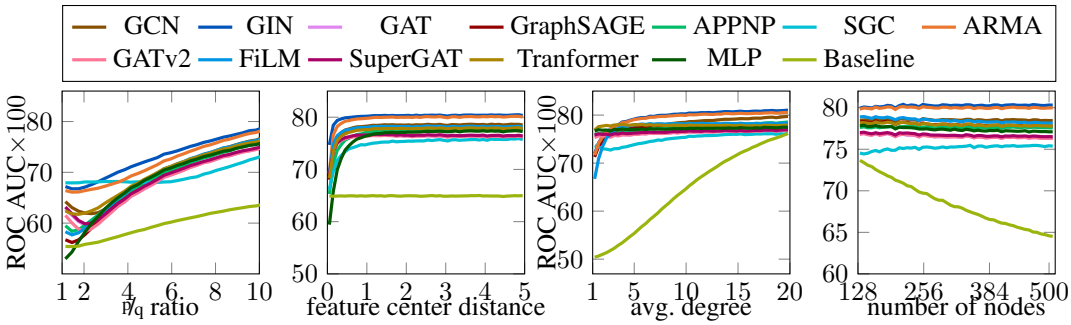

Figure 3: `GraphWorld` link prediction results (mode 2).

degree distribution, are also varied as described further in the Appendix. We tune and test GNN models with ROC-AUC one-vs-rest (AUC-ovr). We train models on a random sample of 5 nodes per-class, tune on a (disjoint) random sample of 5 nodes per-class, and test on the rest of the nodes.

4.2.2 Link Prediction (LP)

In this `GraphWorld` experiment, we generate graphs using the SBM, as for node classification. However, to simulate a link prediction setting, we randomly split edges into training, validation, and test sets. The task is to predict the "unseen" edges in the test set, which we evaluate with the ROC-AUC metric against randomly chosen negatives, imitating the setting in [15]. We train on 80% of the edges, tune on 10% of the edges, and test on the remaining 10%.

4.2.3 Graph Property Prediction (GPP)

In this `GraphWorld` experiment, we generate a dataset of small Erdős-Renyi random graphs. The task is to infer the number of a certain motif in each test graph. In this paper, we evaluate tailed-triangle motif counting. We evaluate the models with scaled mean-squared-error (S-MSE): $\sum(y_i - \hat{y}_i)^2 / \sum(y_i - \bar{y})^2$, which is comparable across datasets with different scales of motif counts. As in [8], we give a dummy feature to each node. Here, the number of training graphs is a variable parameter (see Appendix B.2 for details). We tune on 20% of the data, and test on the remainder.

5 Results and Insights

In this section, we present preliminary results from the `GraphWorld` pipeline. Following the experimental design described in the previous section, we ran nine `GraphWorld` pipelines, one for each of three tasks, and with all three hyperparameter optimization modes per task (see Section 4.1). To sample more efficiently from useful regions of the graph worlds, we applied the `GraphWorld` exploration technique described in Section 3.3 to the Mode 1 experiments, extracting default configurations to use for Mode 2 and Mode 3. In those modes, we sample *only one* parameter from the generator, holding the other parameters fixed at the default config. We provide the default configuration values in Section B.2.

The following two sections cover *global*-distribution and *marginal*-distribution insights uncovered by `GraphWorld`. While our primary aim is not to confirm or overturn established results in the literature, to illustrate the utility of `GraphWorld`, we point out previously-unseen behavior of GNN models which `GraphWorld` has exposed with synthetic graph generation.

5.1 Global Results

5.1.1 Insight: Globally Optimal Hyperparameters Work Well

As shown in Tables 1, 2, and 3, most models tested in `GraphWorld` Mode 2 (training each model with its best-performing hyperparameter set from Mode 1) uniformly outperform their counterparts tested in Mode 1 (with the exception of baselines like PPR, which do not have tunable hyperparameters). While Mode 2 models still do not uniformly outperform Mode 3 models (which each receive a budget of 100 tuning rounds), they do come close, especially on the link prediction task. This shows that `GraphWorld` can cheaply find hyperparameters that work for a large variety of graphs.

model	Mode 1	Mode 2	Mode 3
APPNP	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.00
ARMA	1.03 ± 0.00	0.97 ± 0.00	0.92 ± 0.01
FiLM	1.06 ± 0.01	1.01 ± 0.00	1.01 ± 0.00
GAT	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.00
GATv2	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.00
GCN	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.00
GIN	$\mathbf{0.86 \pm 0.07}$	$\mathbf{0.21 \pm 0.01}$	$\mathbf{0.33 \pm 0.03}$
GraphSAGE	1.04 ± 0.00	1.01 ± 0.00	1.01 ± 0.00
LR	$\mathbf{0.52 \pm 0.00}$	$\mathbf{0.52 \pm 0.00}$	$\mathbf{0.51 \pm 0.03}$
MLP	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.00
SGC	1.08 ± 0.00	1.02 ± 0.00	1.03 ± 0.01
SuperGAT	1.04 ± 0.00	1.02 ± 0.00	1.03 ± 0.01
Transformer	1.04 ± 0.00	1.00 ± 0.00	1.00 ± 0.01

Table 1: Graph property prediction performance averages in terms of scaled MSE. Lower is better.

model	Mode 1	Mode 2	Mode 3
APPNP	62.31 ± 0.05	$\mathbf{74.16 \pm 0.10}$	79.16 ± 0.33
ARMA	$\mathbf{65.03 \pm 0.05}$	71.43 ± 0.10	77.97 ± 0.33
FiLM	$\mathbf{66.07 \pm 0.05}$	71.52 ± 0.11	72.00 ± 0.34
GAT	63.38 ± 0.05	70.42 ± 0.09	$\mathbf{80.32 \pm 0.36}$
GATv2	63.40 ± 0.05	69.79 ± 0.09	$\mathbf{80.24 \pm 0.36}$
GCN	63.50 ± 0.05	71.31 ± 0.09	$\mathbf{81.46 \pm 0.36}$
GIN	61.29 ± 0.04	68.18 ± 0.08	76.05 ± 0.36
GraphSAGE	$\mathbf{64.58 \pm 0.05}$	71.55 ± 0.10	78.41 ± 0.33
MLP	64.14 ± 0.05	71.46 ± 0.11	70.92 ± 0.34
PPR	59.62 ± 0.03	52.28 ± 0.03	59.59 ± 0.28
SGC	58.36 ± 0.04	66.06 ± 0.09	71.79 ± 0.42
SuperGAT	63.58 ± 0.05	71.56 ± 0.09	$\mathbf{81.26 \pm 0.35}$
Transformer	64.07 ± 0.05	71.52 ± 0.10	77.67 ± 0.33

Table 2: Node classification performance averages in terms of ROC-AUC. Higher is better.

model	Mode 1	Mode 2	Mode 3
APPNP	$\mathbf{70.41 \pm 0.02}$	76.48 ± 0.01	76.80 ± 0.11
ARMA	69.21 ± 0.03	$\mathbf{79.49 \pm 0.01}$	$\mathbf{79.77 \pm 0.13}$
Heuristics	69.65 ± 0.03	65.92 ± 0.02	65.91 ± 0.19
FiLM	66.28 ± 0.03	77.40 ± 0.01	77.22 ± 0.14
GAT	60.40 ± 0.03	75.45 ± 0.01	75.48 ± 0.13
GATv2	59.62 ± 0.03	75.44 ± 0.01	75.50 ± 0.13
GCN	64.13 ± 0.03	77.54 ± 0.01	78.00 ± 0.12
GIN	$\mathbf{71.06 \pm 0.03}$	$\mathbf{79.97 \pm 0.01}$	$\mathbf{79.79 \pm 0.13}$
GraphSAGE	61.89 ± 0.04	76.94 ± 0.01	75.20 ± 0.14
MLP	56.28 ± 0.02	76.01 ± 0.01	75.90 ± 0.14
SGC	66.56 ± 0.03	74.62 ± 0.01	75.76 ± 0.11
SuperGAT	60.41 ± 0.03	75.66 ± 0.01	75.54 ± 0.13
Transformer	65.37 ± 0.03	77.21 ± 0.01	77.10 ± 0.14

Table 3: Link prediction mode averages in terms of ROC-AUC. Higher is better.

Furthermore, as seen in Appendix Figure 4, many of the top-performing hyperparameter configurations from `GraphWorld` Mode 1 have similar average test accuracy (there is no stand-out best configuration). Figure 4 displays the dropoff in performance of all unique sampled hyperparameter configurations for the `GraphWorld` Link Prediction Mode 1 experiment. Each configuration has an average test metric score, averaged over each graph world location at which it was sampled. The lines in this plot represent the *ordered* scores for each model, the x-axis representing the inverse percentile rank of the score. This depiction illustrates that for most models, there is no "elbow" [40] or clear break between top-performing hyperparameter configuration and the next 10–20 top performing configurations.

While we are cautious about how these observations apply to graphs with more complex features, it does suggest that finding good hyper-parameters for models in `GraphWorld` is low-cost. As a result of this finding all figures and tables (other than Tables 1–3) contain data from just `GraphWorld` Mode 2 experiments, as in Mode 2 we are able to sample more graphs than in Mode 3 with the same computational budget.

Another important observation from GraphWorld mode comparisons regard one family of models—variations of Graph Attention Networks—that exhibit high sensitivity for hyperparameters in the node classification experiments in Table2. While not achieving the absolute best performance, the difference between Mode 2 and Mode 3 is the most profound for GAT models. We can also compare the improvements to the original GAT architecture. We observe that SuperGAT achieves significantly better performance in Mode 3, whereas GATv2 struggles to improve over its parent model.

5.1.2 Insight: Most GNNs Can't Count Substructures

Table 1 shows that among GNNs we tested on the Graph Property Prediction graph world, only GIN achieved better-than-mean-fitting MSE, along with simple linear regression with edge density as a feature, which outperformed all other GNNs. In fact, all other methods performed no better (on average) than the naive mean-predictor which produces a scaled MSE of exactly 1.0. This result both accords with and contrasts with various results from the paper "Can graph neural networks count substructures?" [8], which contains a synthetic data experiment that GraphWorld replicates thousands of times. We make the following observations and comparisons:

- The conclusion of [8] generally holds that "Message-Passing Neural Networks cannot count substructures". Interestingly however, the GIN architecture is able to learn a representation which has some utility for the task, significantly outperforming the linear regression baseline. This makes intuitive sense, as GIN was developed specifically to be more-expressive for whole-graph encoding tasks.
- We show that simple linear regression using edge density as a feature can count substructures better than most GNNs.

With GraphWorld, we reveal a more controlled and reproducible study into substructure counting, using appropriately-scaled MSE, showing that most GNNs fail on the task, but surprisingly GIN does not (even though none of the GNNs are given meaningful features). Due to the lack of good performance of most GNNs on this task, for the rest of this section we analyze GraphWorld results only on Link Prediction and Node Classification tasks.

5.2 Marginal Results

We now turn to marginal analysis of GNN models. Marginal parameter analysis is a unique and powerful property of GraphWorld, allowing us to examine the average response of GNN models to *particular, explainable* characteristics of the task. We rely on plots in figures 1, 2, and 3 for these insights. We produced those plots using data from GraphWorld Mode 2, using only samples in each plot from which the corresponding parameter was varied.

5.2.1 Insight: GNN models switch ranks outside of standard benchmark space.

To establish our key empirical result, as seen in the three plots inside Figure 1, we project the GraphWorld node classification task distribution space into a 2-D plane measuring each graph's average degree and edge homogeneity, which is the proportion of edges that connect nodes in the same class [51]. Our first finding is that standard benchmark graphs (shown as black points on the plot) cover only a small region of this graph space that GraphWorld is able to cover via synthetic graph generation. This adds to the strong overall motivation for the GraphWorld method described in Section 2, since these statistics should (intuitively) strongly affect graph convolutions.

On the z-axis of each plot in Figure 1, we measure the mean reciprocal rank of GCN, APPNP, and FiLM (respectively) against the other 12 models. Our second finding is that—indeed, as expected—GNN models exhibit high ranking variance across this slice of synthetic graph space. We find sharp MRR phase transitions around 0.5 edge homogeneity, and for lower values of average degree. Furthermore, importantly, standard benchmark datasets mostly avoid regions of phase transitions. This strongly suggests that standard benchmark datasets are insufficient to produce generalizable rankings of models and that there is a serious risk of overfitting to the small number of available benchmark datasets for GNNs. We are hopeful that more comprehensive benchmarking by the means of GraphWorld will help the field to continue to make forward progress.

5.2.2 Insight: GNNs respond surprisingly to graph characteristics

Our GraphWorld experiments on node classification (NC) and link prediction (LP) tasks offer both intuitive and counter-intuitive insights about GNN responsiveness:

- **Number of vertices doesn't matter.** Across NC and LP tasks, the size of the graph (number of vertices) has negligible effect on test AUC. This suggests that in many cases, it may be sufficient to test new GNN architectures on small graphs produced by GraphWorld, rather than focusing on large graphs that are currently being proposed as a one-size-fits-all solution [17].
- **NC: Differential sensitivity to graph and feature signal.** For NC, most models increased test AUC as the p-to-q ratio (graph cluster signal) and feature-center-distance (feature cluster signal) increased. Interestingly, attention-based methods (GAT, GATv2) did not respond as well to these parameters, and seemed to respond negatively to stronger feature signal. FiLM and MLP, which depend strongly on the features, do not respond at all to p/q, but are among the top-performers as the feature signal increases.
- **LP: weaker dependence on cluster strength.** For link prediction, interestingly, the distance between feature clusters and p/q ratio do not have as strong of an effect on models as in the NC task, and some models even seem to exhibit local-maxima behavior in response to these parameters. This suggests that these models (GAT, GATv2, and FiLM) can not harness very powerful node features and translate them to positional embedding for link prediction.

The insights described in this section and Section 5.1 are not possible without a method like GraphWorld.The insights above are certainly not all that could be gleaned from GraphWorld experiments, or even the particular GraphWorld experiments that we ran. We hope that future GNN researchers will include GraphWorld studies as complements to real data analyses.

5.3 Cost and scale

It is relatively cheap to perform large-scale GNN model analyses such as those in this paper with GraphWorld. Our node classification experiments featured in the paper cost under $120, involving 13 models on 1M+ synthetic benchmark datasets, with no GPUs and negligible RAM. By comparison, the experiment with real-world OGB data from [26] involved only 3 models, only 1 OGB dataset, 4 GPUs, and >480GB of RAM per >24 CPUs. These resources would cost >$500 on modern cloud compute platforms (see https://cloud.google.com/compute). Additionally, that single OGB experiment completed in >40hrs, whereas our GraphWorld experiments took 10hrs total.

6 Conclusions and Future Work

In this paper, we have shown that GraphWorld addresses these problems via the following features:

1. **Generalizable analyses.** With tunable parameters for GNN dataset generators, GraphWorld can simulate graphs with far-wider ranges of graph properties than currently exist in any collection of benchmark datasets. As shown in our results (especially Figure 1), the marginal analysis of these parameters and statistics can generate insights about GNN architectures that are unavailable from any small collection of re-used benchmark datasets.
2. **Reproducibility without overfitting.** Using our code and platform, a researcher can use GraphWorld to test their model as easily as with any existing collection of GNN benchmarks, but without the risk of overfitting to graphs with a limited set of properties.
3. **Accessibility.** As described in Section 5.3, GraphWorld does not require excessive resources, and can actually test many more models at a time for lower cost than standard benchmarks. Furthermore, our experiments show that assessing GNN test performance does not depend on having natural, society-scale graph data. Combining these observations, we have shown that with GraphWorld it is possible to derive new insights with constrained resources.

These characteristics make GraphWorld the perfect complement to GNN experiments on graph datasets sourced from nature. While performance on such natural datasets will always be of scientific interest and essential for new research, GraphWorld can expose when progress on them may not transfer to other datasets. More importantly, GraphWorld can help uncover certain distributions of graphs that have not yet been used to test GNNs, which we hope will inspire new architectures.

References

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *JMLR*, 2017.

[2] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 2003.

[3] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[4] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 1998.

[5] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *ICML*. PMLR, 2020.

[6] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[7] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

[8] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

[9] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020.

[10] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *RecSys*, 2019.

[11] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 1945.

[12] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[13] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *ICLR*, 2020.

[14] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[15] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

[18] Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 1912.

[19] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[20] Megha Khosla, Vinay Setty, and Avishek Anand. A comparative study for unsupervised network representation learning. *TKDE*, 2019.

[21] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *ICLR*, 2021.

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

[23] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[24] Pang Wei Koh, Shiori Sagawa, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

[25] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 2006.

[26] Guohao Li, Jesus Zarzar, Hesham Mostafa, Sohil Shah, Marcel Nassar, Daniel Cummings, Sami Abu-El-Haija, Bernard Ghanem, and Matthias Müller. Deeperbiggerbetter for ogb-lsc at kdd cup 2021. 2021.

[27] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[28] Enxhell Luzhnica, Ben Day, and Pietro Liò. On graph classification networks, datasets and baselines. *arXiv preprint arXiv:1905.04682*, 2019.

[29] Alexandru Mara, Jefrey Lijffijt, and Tijl de Bie. Reproducible evaluations of network representation learning models using evalne. *WWW'21, Workshop on Graph Learning Benchmarks*, 2021.

[30] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[31] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*, 2019.

[32] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 2002.

[33] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

[34] Neil Shah. Scale-free, attributed and class-assortative graph generation to facilitate introspection of graph neural networks. *WWW'21, Workshop on Graph Learning Benchmarks*, 2020.

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

[36] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. 2021.

[37] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020.

[38] George W Snedecor. Statistical methods, 1957.

[39] Th A Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 1948.

[40] Robert L Thorndike. Who belongs in the family? *Psychometrika*, 1953.

[41] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*, 2020.

[42] Anton Tsitsulin, Benedek Rozemberczki, John Palowitch, and Bryan Perozzi. Synthetic graph generation to benchmark graph learning. *WWW'21, Workshop on Graph Learning Benchmarks*, 2021.

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

[44] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *ACL*, 2020.

[45] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*. PMLR, 2019.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

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

[48] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. In *NeurIPS*, 2020.

[49] Wentao Zhao, Dalin Zhou, Xinguo Qiu, and Wei Jiang. A pipeline for fair comparison of graph neural networks in node classification tasks. *arXiv preprint arXiv:2012.10619*, 2020.

[50] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 2009.

[51] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *arXiv preprint arXiv:2006.11468*, 2020.

Task	Mode	Samples (N)	Tuning Rounds	vCPU hours
LP	1	1e6	0	1,681
LP	2	7e5	0	1,672
LP	3	7e3	100	1,896
NC	1	1e6	0	1,047
NC	2	7e5	0	755
NC	3	7e3	100	937
GPP	1	1e6	0	9,767
GPP	2	4e5	0	3,399
GPP	3	4e3	100	3,553

Table 4: Resource complexity for `GraphWorld` experiments.

Hyperparameter	Values
Learning Rate	$[0.01, 0.001, 0.0001]$
Hidden Channels	$[4, 8, 16]$
Number of Layers	$[1, 2, 3, 4]$
Dropout	$[0, 0.3, 0.5, 0.8]$
α (APPNP, SGC, and PPR baseline)	$[0.1, 0.2, 0.3]$
Iterations (APPNP and SGC)	$[5, 10, 15]$
# of attention heads (GATs and Transformer)	$[1, 2, 3, 4]$

Table 5: Hyperparameter values for all models used by all `GraphWorld` experiments.

A Implementation and Cost

Design goals for `GraphWorld` focused on accessibility, scalability and efficiency; any researcher should be able to run `GraphWorld` simulations with minimal setup, while having the system automatically scale up experiments to available resources only as needed. To this end, `GraphWorld` is implemented as a containerized Apache Beam[1] pipeline allowing researchers to run a hermetic copy of Graph World on any infrastructure i.e., a local machine, compute cluster, or cloud framework. Experiments in this paper were run on Google Cloud Platform (GCP) using Cloud Dataflow. Experiments were allowed to scale up to a maximum of 1000 workers using n1-standard-1 machines capable of sampling millions of graphs in ≤ 10 hours.

Figure 5 shows the design of the `GraphWorld` distributed processing system. Table 4 shows the number of virtual-CPU hours needed to complete the nine pipelines discussed in Section 4.

B Experiment Details

In this Appendix section, we provide more details about `GraphWorld` pipelines, the task dataset generators, and the model architectures. In particular, we specify all the `GraphWorld` configuration elements (see Section 3) of each pipeline described in Section 4.

B.1 Models

For the experiments in this paper, we choose 11 representative GNNs and 3 baselines to illustrate the strengths of our proposed approach.

- **ARMA** [3]: A GNN with auto-regressive moving average filters.
- **APPNP** [23]: One of the first 'linear' GNNs, accelerating propagation using Personalized PageRank.
- **FiLM** [5]: A model that modulates an incoming message by the features of the target node.

[1] https://beam.apache.org/

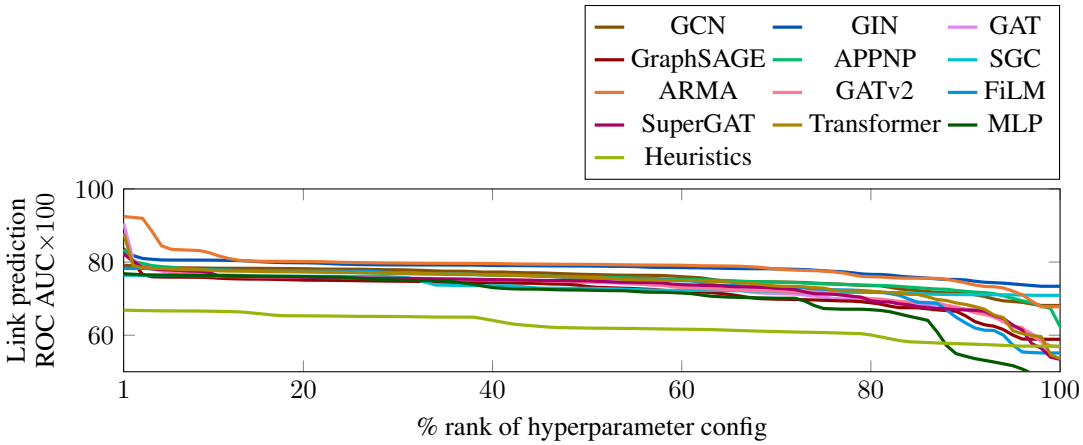

Figure 4: Each line is the accuracy dropoff of hyperparameter configurations for a particular model, from the `GraphWorld` Link Prediction Mode 1 experiment. The x-axis is the inverse percentile rank of the hyperparameter configuration, and the y-axis is the accuracy. There is no "elbow" [40] or clear break between top-performing hyperparameter configuration and the next 10–20 top performing configurations. See Section 5.1 for context.

- **GAT** [43]: An early model of graph attention.
- **GATv2** [6]: An improved variant of graph attention that allows any node to attend to any other one.
- **GCN** [22]: A seminal model that averages neighbor state at each iteration.
- **GIN** [47]: This model uses MLPs to transform summations of neighbor features.
- **GraphSAGE** [16]: A variant of the GCN which adds uses sampling and improved propagation of the hidden state.
- **SGC** [45]: Linear GNNs using matrix multiplication.
- **SuperGAT** [21]: An approach to improve the graph attention layer.
- **Transformer** [36]: A multi-head attention-based model.

In addition we examine several baselines which are not GNNs:

- **Linear Regression**: (*graph property prediction only*) Simple ordinary least-squares with edge density as the sole feature.
- **Multi-Layer Perceptron**: (*features only*) Transforms the node features via a DNN for classification.
- **Personalized PageRank** [4]: (*graph only*) Predicts node labels for unseen nodes via Personalized PageRank seeded by the labelled nodes. For each unseen node, we compute the total probability mass that comes from the vertices of each label, and pick the label with the highest score.
- **Link prediction heuristics** [29]: (*graph only*) Creates a link prediction ranking using eight different reweighting schemes for counting common neighbors. We pick one of the following schemes: (i) Sørensen–Dice coefficient [39, 11], (ii) cosine similarity, (iii, iv) hub-promoted and hub-suppressed similarity [32], (v) Jaccard similarity [18], (vi) Adamic–Adar index [2], (vii) Resource Allocation index [50], and (viii) Leicht–Holme–Newman similarity [25].

The GNN models use the reference implementations from the PyTorch-Geometric library [14]. We note that by design it is trivial to add additional models into `GraphWorld`.

We now list hyperparameter values available to each GNN model (and some non-GNN models) for tuning. We note that `GraphWorld` experiments are focused on a comparison of the convolution layers introduced by each model (defined by its corresponding convolution implementation in PyTorch-geometric).

Graph Property Prediction. In order to generate the global readout of the node state, we take the final layer's activations for all the nodes and apply mean pooling to create a graph representation. This representation is then used for regressing substructure counts.

B.2 Generator parameters

In Tables 6 and 7, we list generator parameters for task dataset generators. Table 6 has parameter values for the Node Classification and Link Prediction pipelines. Table 7 has parameter values for the Graph Property Prediction pipelines. Each table contains the parameter names, their description, and their default values found using the technique described in Section 3.3.

Parameter Name	Description	Values	NC	LP
nvertex	Number of vertices in the graph.	$[128, 512]$	343	481
p/q ratio	the ratio of in-cluster edge probability to out-cluster edge probability	$[1.0, 10.0]$	3.98	12.40
avg. degree	the average expected degrees of the nodes	$[1.0, 20.0]$	1.75	10.12
feature center distance	the variance of feature cluster centers, generated from a multivariate Normal	[0.0, 5.0]	0.20	2.20
num clusters	the number of unique node labels	[2, 6]	5	4
cluster size slope	the slope of cluster sizes when index-ordered by size	[0.0, 0.5]	0.08	0.92
power exponent	the value of the power law exponent used to generate expected node degrees	[0.5, 1.0]	1.0	1.0

Table 6: Node Classification and Link Prediction default generator values used in `GraphWorld` Mode 2 and Mode 3 experiments.

Parameter Name	Description	Values	Default
ngraphs	Number of graphs in each dataset.	$[100, 500]$	499
num vertices	Number of vertices in each graph	$[5, 30]$	6
edge prob	the edge probability of the Erdos-Renyi graph	$[0.1, 0.75]$	0.72
train prob	number of graphs in the dataset used for training	$[0.2, 0.6]$	0.60

Table 7: Graph Property Prediction generator values used in `GraphWorld` Mode 2 and Mode 3 experiments.

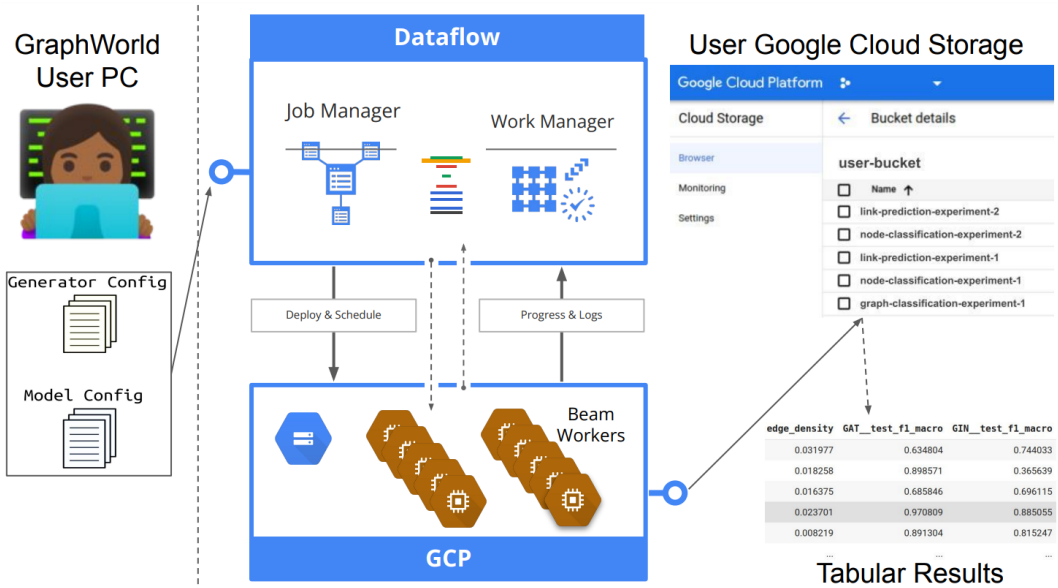

Figure 5: On their PC, a `GraphWorld` user specifies dataset generator and GNN model configurations, and then launches a `GraphWorld` pipeline, which is submitted on a remote manager. The records of GNN tests are written to Google Cloud Storage (GCS), which we then accumulate into a data table via the GCS API.