Robust Synthetic GNN Benchmarks with GraphWorld

John Palowitch, Anton Tsitsulin, Brandon Mayer, Bryan Perozzi
palowitch,tsituslin,bmayer,hubris@google.com
Google Research
USA

ABSTRACT

Despite advances in GNNs, only a small number of datasets are 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 are likely to have datasets which are very different from academic benchmarks. In this work we introduce `GraphWorld`, a novel methodology for benchmarking GNN models on an arbitrarily-diverse population of *synthetic* graphs for *any* GNN task. We present insights from `GraphWorld` experiments regarding the performance characteristics of eleven GNN models over millions of benchmark datasets. 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.

1 INTRODUCTION

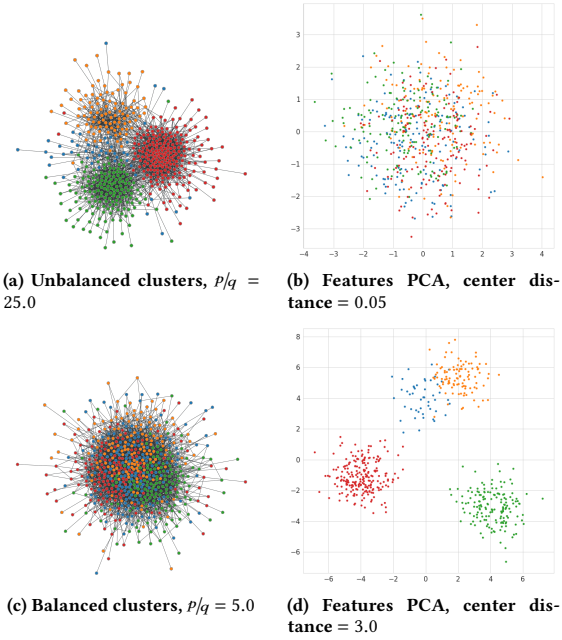
Graph Neural Networks (GNNs) have extended the benefits of deep learning to the non-Euclidean domain, allowing for standardized and re-usable machine learning approaches to problems that involve relational (graph-structured) data [33]. GNNs now admit an extremely wide range of architectures and possible tasks, including node classification, whole-graph classification, and link prediction [7]. With this growth has come increased calls for proper GNN experimental design [26, 35], refreshed benchmark datasets [14], and fair comparisons of GNN models in reproducible settings [10, 20].

Despite the proliferation of new GNN models, only a few hand-picked benchmark datasets are currently used to evaluate them [14]. The limited scope of these datasets makes it hard for practitioners to infer which models will generalize well. Relatedly, new architectures are proposed only when they beat existing methods on these datasets, which can cause *architectural overfitting* [22, 24].

In this work, we introduce `GraphWorld`, the first tunable, scalable, and reproducible method for analyzing the performance of GNN models on a statistically diverse set of benchmark datasets for *any given GNN task* (i.e. all {un/semi}supervised node/graph problems [33]). Using arbitrarily-tunable random graph generation models, `GraphWorld` allows comparisons between GNN models and architectures that are not possible with standard benchmarks. As seen in Figure 2, GNN models change sharply in performance ranking when tested on `GraphWorld` datasets that are *distant* in graph property space from standard real-world datasets.

In this short paper we provide overviews of the contributions from the longer version [23]:

(1) **Problem Formulation**. We pose the question: how can we measure GNN model performance across graph datasets with high statistical variance? Modern benchmark datasets, however well-maintained, can be limited in scope, and can be computationally inaccessible to the average researcher.



(a) Unbalanced clusters, p/q = 25.0 **(b) Features PCA, center distance = 0.05**

(c) Balanced clusters, p/q = 5.0 **(d) Features PCA, center distance = 3.0**

Figure 1: `GraphWorld` node classification datasets from the SBM with Gaussian node features. The datasets above differ in graph signal-to-noise ratio (p/q) and the feature center cluster distance.

(2) **Methodology**. We provide `GraphWorld`, a graph sampling and GNN training procedure which is capable of testing state-of-the-art 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.

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, such as node classification on the CORA dataset. In general, such experiments are used to arrive at some insights about how new architectures will perform in realistic scenarios. However, most studies feature 1 task and <5 datasets (graph-classification datasets such as [21] are considered one dataset), which greatly limits the confidence in the ability of any particular paper's results to generalize.

In this paper, we question whether or not this status quo of GNN evaluation is enough to measure progress in the field. Relying only on a handful of graph datasets over time could be detrimental to the field, for the following three reasons:

Inadequate generalization. Each curated graph dataset represents just one point in the space of all possible datasets that can be associated with the particular GNN task at-hand. The graph(s) in a 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 cause overfitting of new architectures to the datasets, as observed in NLP [22] and computer vision [24]. This effect (and the related "inadequate generalization" effect described above) is illustrated in Fig. 2.

Un-scalable development. In recent years, there has been a particular focus on scalability in GNN research and benchmarks [14]. While this is important for investigating GNN capacity, 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 without access to institution-scale resources.

Possibly due to the existence of well-studied random graph models such as the Stochastic Block Model [2, 15], 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 [8, 9, 25, 29, 30, 36]. 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`.

3 GRAPHWORLD

To address the problems above, we introduce `GraphWorld`: a trivially-parallel method for generating diverse *populations* of GNN benchmarks using random graphs and extracting marginal population-level insights about GNN architectures.

Graph Generation. `GraphWorld` simulates GNN datasets using random graphs and feature generators. Each `GraphWorld` dataset comes from a 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 graph space, $\mathcal{F}$ is a feature space, and $\mathcal{L}$ is a label space. 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 $\mathcal{P}$.

For example, we can generate *node classification* datasets with the Stochastic Block Model (SBM) [c.f. 2]: cluster assignments double as class labels, and node features $F \in \mathcal{F}$ are drawn from a Normal mixture. The class sizes, feature cluster centers, edge homogeneity, and many more dataset properties are tunable with the SBM parameters, e.g. Fig. 1. `GraphWorld` samples all parameters from a wide range, producing a diverse sample of benchmarks.

Training, Testing, and Evaluation. The `GraphWorld` method simulates a pre-specified number of GNN test datasets $D_1, D_2, \ldots \in \mathcal{D}$, and then trains and tests an arbitrary list of GNN models on each dataset. Similarly to the test data distributions, the hyperparameters of GNN model m (for a particular dataset) is determined by a sample or specification $(h_{m1}, h_{m2}, \ldots) \in \mathcal{H}_m = H_{m1} \times H_{m2} \times \ldots$. The complete set of inputs to the `GraphWorld` method are (1) a list of models to test; (2) a task formulation $\mathcal{P}(\pi_1, \pi_2, \ldots)$; (3) a task metric; (4) N: number of datasets. With these inputs, the `GraphWorld` system is trivially parallel over N samples.

Efficient analysis. 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 π_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 [28] 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.

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 Methods

We experiment on 11 representative GNNs and 3 baselines to illustrate the strengths of our proposed approach: **ARMA** [3], **APPNP** [18], **FiLM** [5], **GAT** [31], **GATv2** [6], **GCN** [17], **GIN** [34], **GraphSAGE** [13], **SGC** [32], **SuperGAT** [16], and **Transformer** [27] as GNN models; **Linear Regression** (*graph property prediction only*),

Multi-Layer Perceptron, **Personalized PageRank** [4], **Link prediction heuristics** [20] for baselines. By design, it is trivial to add additional models into `GraphWorld`.

4.2 Tasks

Here we describe three tasks that will be explored with three separate `GraphWorld` pipelines. In Section 5 we show results from each task run in each of the hyperparameter tuning modes described in Appendix B.2.

4.2.1 Node Classification (NC). As aforementioned, we generate NC datasets with the SBM. Cluster labels are generated from a multinomial distribution, and 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. Other parameters and training details are given in [23].

4.2.2 Link Prediction (LP). Again we generate SBM graphs. However, to craft 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 [11].

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 unit one-dimensional feature to each node.

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 Appendix B.2). We applied the `GraphWorld` F-statistic exploration technique to Modes 2-3, sampling *only one* parameter from the generator, holding the other parameters fixed at default. All results below come from the Mode 2 pipelines.

5.0.1 Marginal Parameter Analysis. We now cover marginal parameter insights from the node classification (NC) task, leaving results from the other two tasks discussed above for the main paper [23]. 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. Figures 2 and 3 present marginal parameter results, which we discuss further below. We produced those plots using data from `GraphWorld` Mode 2, using only samples in each plot from which the corresponding parameter was varied.

5.0.2 Insight: GNN models switch ranks outside of standard benchmark space. To establish our key empirical result, as seen in the three plots inside Figure 2, 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 [36]. 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 2, 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.

5.0.3 Insight: GNNs respond surprisingly to graph characteristics. Our `GraphWorld` experiments on the node classification tasks offer both intuitive and counter-intuitive insights about GNN responsiveness to graph and node feature distributions. We make the following observations (more in [23]):

- **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, when studying GNN sensitivity to graph characteristics like edge homogeneity, it may be sufficient to test new GNN architectures on small graphs produced by `GraphWorld`, rather than on large real-world graphs. However, as a limitation of this study, we note that small graphs such as those in this `GraphWorld` experiment cannot test GNN handling of long-range dependencies, studied e.g. in [18] and [12].
- **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. 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.

6 CONCLUSIONS AND FUTURE WORK

The insights described in the previous section are not possible without `GraphWorld`, which (1) has the capability to generate millions of test datasets with diverse characteristics and (2) *logs* the dataset characteristics along with test metrics for each model. More broadly, `GraphWorld` addresses limitations of GNN experimentation (discussed in Section 2) 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

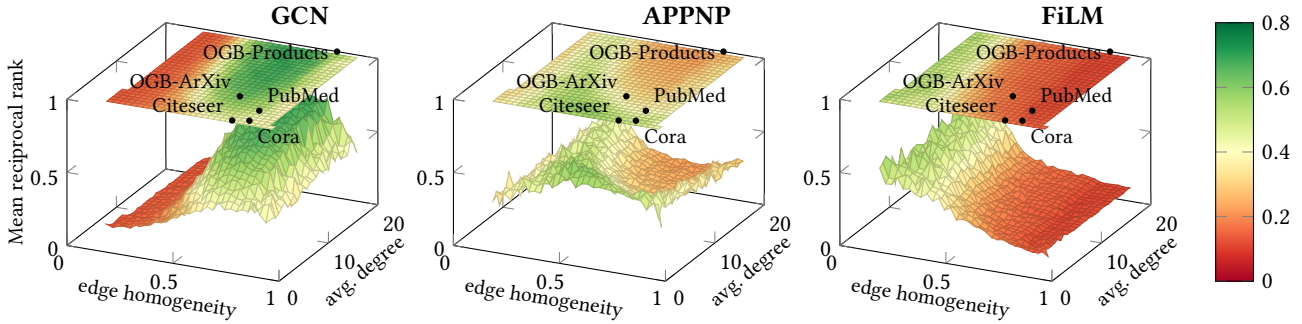


Figure 2: `GraphWorld` uses synthetic data to elucidate fundamental differences between graph neural network convolutions. Shown here are the relative performance results of GCN [17], APPNP [18], and FiLM [5] across 50,000 distinct node classification tasks. The x and y axes group the synthetic graphs by their structural properties, while the z-axis shows the mean reciprocal-rank (MRR) relative to other baselines (Section 4.1). We find that standard GNN benchmark datasets (Cora, OGB, etc.) exist in regions of the `GraphWorld` where model rankings do not change. `GraphWorld` can discover previously unexplored graphs which reveal new insights about GNN architectures.

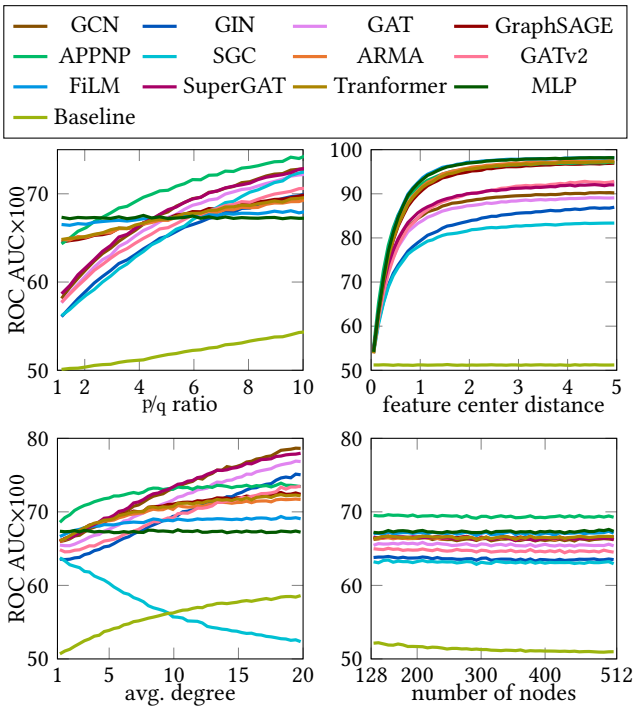


Figure 3: `GraphWorld` node classification results (mode 2).

collection of benchmark datasets. As shown in our results (especially Figure 2), the marginal analysis of these parameters can generate insights about GNN architectures that are unavailable from any small collection of re-used datasets.

(2) **Reproducibility without overfitting.** With `GraphWorld`, a researcher can 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 Appendix A, `GraphWorld` does not require excessive resources, and can 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 less resources. Thus `GraphWorld` can help facilitate GNN research in smaller labs.

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 architecture development. At Google, we are integrating `GraphWorld` with GNN experimental pipelines and unit tests, as well as with TF-GNN [1].

One limitation of the analyses in Section 5 is that most random graph models (including the SBM) are simple, relative to the complexity of real-world data. A complementary and follow-up line of research to our work could be the development of new random graph models with tunable properties that target classes of GNN architectures, or certain types of complex graph structures. For instance, we may wish to design clustered graph models with tunable numbers of certain graph motifs, or attributed graph models with non-trivial feature correlations. `GraphWorld` is the perfect tool to understand how these variations in these graph properties cause differential responses from various GNN architectures.

Overall, `GraphWorld` facilitates cheap, comprehensive, and principled investigation into the nuances of GNN model performance. By releasing our (forthcoming) codebase and integrating `GraphWorld` with TF-GNN, we hope to make GNN experimental results more robust and transferable – helping researchers reach more reliable conclusions when developing new architectures.

ACKNOWLEDGMENTS

We thank Google Cloud for supporting this project.

REFERENCES

[1] 2021. Tensorflow GNN. https://github.com/tensorflow/gnn.
[2] Emmanuel Abbe. 2017. Community detection and stochastic block models: recent developments. *JMLR* (2017).
[3] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
[4] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* (1998).
[5] Marc Brockschmidt. 2020. Gnn-film: Graph neural networks with feature-wise linear modulation. In *ICML*. PMLR.
[6] Shaked Brody, Uri Alon, and Eran Yahav. 2021. 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. 2020. 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. 2020. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025* (2020).
[9] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982* (2020).
[10] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A fair comparison of graph neural networks for graph classification. In *ICLR*.
[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.
[12] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. 2020. Implicit graph neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 11984–11995.
[13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.
[14] 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. In *NeurIPS*.
[15] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical review E* 83, 1 (2011), 016107.
[16] Dongkwan Kim and Alice Oh. 2021. How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision. In *ICLR*.
[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[18] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).
[19] Guohao Li, Jesus Zarzar, Hesham Mostafa, Sohil Shah, Marcel Nassar, Daniel Cummings, Sami Abu-El-Haija, Bernard Ghanem, and Matthias Müller. 2021. DEEPERBIGGERBETTER for OGB-LSC at KDD cup 2021. (2021).
[20] Alexandru Mara, Jefrey Lijffijt, and Tijl de Bie. 2021. Reproducible Evaluations of Network Representation Learning Models Using EvalNE. *WWW'21, Workshop on Graph Learning Benchmarks* (2021).
[21] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).
[22] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial NLI: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599* (2019).
[23] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. 2022. GraphWorld: Fake Graphs Bring Real Insights for GNNs. *arXiv preprint arXiv:2203.00112* (2022).
[24] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2018. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451* (2018).
[25] Neil Shah. 2020. Scale-Free, Attributed and Class-Assortative Graph Generation to Facilitate Introspection of Graph Neural Networks. *WWW'21, Workshop on Graph Learning Benchmarks* (2020).
[26] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
[27] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2021. Masked label prediction: Unified message passing model for semi-supervised classification. (2021).
[28] George W Snedecor. 1957. Statistical methods.
[29] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2020. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904* (2020).
[30] Anton Tsitsulin, Benedek Rozemberczki, John Palowitch, and Bryan Perozzi. 2021. Synthetic Graph Generation to Benchmark Graph Learning. *WWW'21, Workshop on Graph Learning Benchmarks* (2021).
[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[32] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*. PMLR.
[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* (2020).
[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[35] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. In *NeurIPS*.
[36] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *arXiv preprint arXiv:2006.11468* (2020).

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 $\leq$ 10 hours. Table 2 shows the number of virtual-CPU hours needed to complete the nine pipelines discussed in Section 4. 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+ benchmark datasets, with no GPUs and negligible RAM. By comparison, the experiment with real-world OGB data from [19] involved only 3 models, only 1 Open Graph Benchmark (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 OGB experiment completed in >40hrs, whereas our GraphWorld experiments took 10hrs total.

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 the node classification pipeline. We also include tabular experiment results from each `GraphWorld` pipeline.

B.1 Models

In Table 1 we 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).

B.1.1 Graph Property Prediction. In order to generate the global readout of the node state, we take the final layer's activations for all

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

the nodes and apply mean pooling to create a graph representation. This representation is then used for regressing substructure counts

B.2 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:

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

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.

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

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 2: Resource complexity for `GraphWorld` experiments.