IGB: Addressing The Gaps In Labeling, Features, Heterogeneity, and Size of Public Graph Datasets for Deep Learning Research

Arpandeep Khatua
UIUC
USA

Vikram Sharma Mailthody
NVIDIA
USA

Bhagyashree Taleka
UIUC
USA

Tengfei Ma
IBM Research
USA

Xiang Song
AWS AI
USA

Wen-mei Hwu
NVIDIA/UIUC
USA

ABSTRACT

Graph neural networks (GNNs) have shown high potential for a variety of real-world, challenging applications, but one of the major obstacles in GNN research is the lack of large-scale flexible datasets. Most existing public datasets for GNNs are relatively small, which limits the ability of GNNs to generalize to unseen data. The few existing large-scale graph datasets provide very limited labeled data. This makes it difficult to determine if the GNN model’s low accuracy for unseen data is inherently due to insufficient training data or if the model failed to generalize. Additionally, datasets used to train GNNs need to offer flexibility to enable a thorough study of the impact of various factors while training GNN models.

In this work, we introduce the Illinois Graph Benchmark (IGB), a research dataset tool that the developers can use to train, scrutinize and systematically evaluate GNN models with high fidelity. IGB includes both homogeneous and heterogeneous academic graphs of enormous sizes, with more than 40% of their nodes labeled. Compared to the largest graph datasets publicly available, the IGB provides over 162x more labeled data for deep learning practitioners and developers to create and evaluate models with higher accuracy. The IGB dataset is a collection of academic graphs designed to be flexible, enabling the study of various GNN architectures, embedding generation techniques, and analyzing system performance issues for node classification tasks. IGB is open-sourced, supports DGL and PyG frameworks, and comes with releases of the raw text that we believe foster emerging language models and supports DGL and PyG frameworks, and comes with releases of the raw text that we believe foster emerging language models.

CCS CONCEPTS

- Computing methodologies → Neural networks; Knowledge representation and reasoning; Natural language processing.

KEYWORDS

Datasets, Graph neural networks (GNNs), Graphs, Deep learning

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

KDD, August 06–10, 2023, Long Beach, CA
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.1145/3580305.3599843

1 INTRODUCTION

Graph neural networks (GNNs) are a class of neural networks that operate on graph-structured data. GNNs have shown to be effective in addressing a variety of real-world applications such as fraud detection [18, 47, 74], recommendation systems [61, 71, 72], predicting molecular and protein structure [29, 57], knowledge representation [15], and more recently helping in fine-tuning large language models [73]. Their popularity has led to the development of various optimized frameworks and libraries [19, 63, 76] that enable the fast application of GNNs on new domains, making it easier for researchers and practitioners to leverage the power of GNNs in their work.

However, high-quality frameworks and libraries are necessary but not sufficient for enabling fast research progress in GNN. One of the major challenges in GNN research is the lack of large-scale datasets. This is because large graph datasets are typically proprietary and most publicly available ones are rather small. The small size of these datasets makes it difficult to train GNNs that can handle large-graph structures and prevents the use of powerful pre-training techniques [16, 32, 59, 65]. These challenges make it difficult to fully leverage GNN potential and its applications.

To address these challenges, recent work such as OGBN and MAG [26, 27] have proposed open large graph benchmark suites providing up to 244 million nodes and 1.7 billion edge graphs. These datasets contain a diverse set of graphs and have been widely used in the research community to benchmark GNNs’ performance. However, most existing datasets, including OGBN, provide very limited labeled data. As GNN downstream tasks are often trained as supervised learning tasks, having large labeled data matters, especially for multi-label classification problems. However, both OGBN and MAG use Arxiv [26, 27] class labels which provide 1.4 million labeled nodes, meaning only about 1% of the overall dataset is labeled! With such small labeled data usage during training, it becomes challenging to determine if the GNN model’s low accuracy for unseen data is inherently due to insufficient training data or if the model itself fails to generalize [28, 37, 45, 62, 78].
Furthermore, the lack of flexible datasets hinders the researcher’s ability to scrutinize and systematically evaluate the scalability of the GNN models, frameworks, and systems. Ideally, a dataset should provide (a) capability to study the impact of embedding generation techniques and their properties on the GNN model’s accuracy, (b) provide sub-datasets of varying graph sizes and embeddings maintaining consistent homophily, and (c) provide a range of multi-class classification tasks with varying degrees of complexity. Without such flexible datasets, it is difficult to train models on small graph datasets and then evaluate their accuracy and execution efficiency on larger data corpus, a common scenario in industrial settings. Moreover, the framework and system scalability problems encountered with small datasets are different from those with large datasets, making it challenging to study the system requirements of GNNs.

To this end, this work proposes Illinois Graph Benchmark (IGB), a research dataset tool that the researchers can use to scrutinize and systematically evaluate the GNN models and their execution efficiency. IGB provides access to enormous academic graph datasets for deep learning research consisting of both homogeneous and heterogeneous graphs, providing a diverse range of graph structures for training and evaluating GNN models. IGB homogeneous (IGB-HOM) graph dataset has up to 269 million nodes and about four billion edges. IGB heterogeneous (IGB-HET) graph dataset has up to 547 million nodes and about six billion edges. Both of these datasets come with more than 220 million labeled nodes created with a novel approach and with two complex node classification tasks (19 and 2983 unique classes). Compared to the largest graph dataset publicly available [26, 27], IGB provides over 162× more labeled data, providing ample opportunities for GNN and deep learning practitioners to generalize the models to unseen data.

Furthermore, as a research tool, IGB offers flexible design choices for GNN model developer to investigate and systematically assess their models’ performance and execution efficiency. IGB provides variable-sized embeddings, can generate embeddings from different language models and provides a range of variable-size graphs with consistent homophily. To demonstrate the usefulness of such flexibility, we conduct an extensive ablation study to show the impact of node embedding generation on the GNN model accuracy. We find that using the Roberta NLP embeddings provides better accuracy on GNN models and that a larger embedding dimension further assists the GNN models’ performance and execution efficiency. IGB provides (a) capability to study the impact of embedding generation on the GNN model accuracy, (b) provide sub-datasets of varying graph sizes and embeddings maintaining consistent homophily, and (c) provide a range of multi-class classification tasks with varying degrees of complexity. Without such flexible datasets, it is difficult to train models on small graph datasets and then evaluate their accuracy and execution efficiency on larger data corpus, a common scenario in industrial settings. Moreover, the framework and system scalability problems encountered with small datasets are different from those with large datasets, making it challenging to study the system requirements of GNNs.

Overall this work makes the following key contributions:

1. We propose IGB, a research dataset tool that innovatively fills the critical gap in labeling, features, heterogeneity, and size of public graph datasets.
2. We show IGB offers flexible design choices enabling the exploration of different GNN designs, embedding generation schemes, the effect of labeled data, and how system performance evolves with increasing dataset size.

IGB is compatible with popular GNN frameworks like DGL [63] and PyG [19], and comes with several predefined popular models. IGB is available for public usage including raw text used to generate embedding and with a public leaderboard soon [30].

2 BACKGROUND AND MOTIVATION

In this section, we provide a brief overview of GNNs and their applications. We then cover existing GNN datasets and discuss the importance of high-quality, large-scale datasets. Additional overview of Graph Neural networks are explained in Appendix A.

2.1 GNN Dataset Generation Techniques

With the increasing popularity of GNNs, researchers have opted to generate GNN datasets based on their needs to simulate and test their model performance. Some of the popular GNN dataset generation techniques include: (a) Real-world graph datasets: This technique takes an existing real-world database such as a transaction database or webpages and extracts graph structure from it [26, 31]. (b) Graph sampling: This technique selects a subset of nodes and edges from existing large graphs to create a sample dataset [33, 74]. (c) Synthetic graph generation: This method creates synthetic graphs by randomly connecting nodes according to certain probability distributions such as power law or by using Kronecker graph generators [13, 43]. This technique can also be used as data augmentation to increase the dataset diversity. IGB datasets are created using real-world datasets and different configurations are created by performing graph sampling operations. We will describe the IGB dataset generation methodology in §3.

2.2 Existing Datasets And Their Problems

The existing dataset collection for GNNs are relatively small in size limiting the ability of GNN to generalize to unseen data [3, 5–9, 14, 17, 22, 24, 33, 34, 39, 42, 50, 52–55, 58, 60, 66, 69, 70, 74]. More recently, work such as OGBN [26, 27, 36] have proposed datasets providing up to 244 million nodes and 1.7 billion edge graphs. These datasets contain a diverse set of graphs and have been widely used by the research community to benchmark GNN’s performance.

Table 1 summarizes the various types of publicly disclosed graph datasets. We define flexibility in datasets as the ability to provide various configurations while maintaining homophily. This includes subgraphs with different node-degree distributions, variable-size embeddings, and different language models used to generate embeddings. Homophily refers to the tendency for individual nodes to be connected to other nodes with similar properties. It is fundamental for GNNs because it allows the model to learn a more accurate representation of the nodes in the graph. With datasets that have similar homophily, researchers can better understand and develop GNN methods.
models by training with smaller graphs and evaluating accuracy on larger graphs.

Table 1 covers both synthetic and real large graph datasets that are used to study the GNN model performance in both closed and open-source environments. PinSAGE [71] is one of the few publicly disclosed proprietary industrial datasets that have more than three billion nodes and eighteen billion edges with an unknown number of labels. Among the real-world open source datasets, MAG240M from OGB-LSC [26] provides the maximum number of nodes to evaluate multi-class node classification tasks. Yet, as noted from Table 1, most existing datasets including OGBN datasets provide a tiny set of labeled data, provide no flexibility and the embeddings are generated in closed source.

As the GNN downstream tasks are often trained as supervised learning tasks, having large labeled data helps in increasing the accuracy of multi-label classification problems. As several prior works have shown [28, 37, 45, 62], with such a small number of usable labeled nodes during training, it becomes challenging to determine if the GNN’s low accuracy for unseen data is primarily due to insufficient training data or inability to generalize.

Besides, it is crucial to have access to flexible datasets when training GNN models in order to fully understand the impact of various factors on their accuracy. One key aspect is the generation of embeddings associated with the nodes or edges in the graph. As NLP models are used to generate the embeddings for GNN models, the quality of the NLP model can have a direct effect on the GNN model’s accuracy and their downstream tasks. Thus, the datasets should provide mechanisms to study different embedding generation techniques along with methods to vary their properties such as node embedding dimensions, and pruning.

Furthermore, as GNNs become increasingly popular, it is crucial to comprehend how their accuracy and efficiency (wall clock time) scale with the size and complexity of the graph. In this regard, researchers are developing optimized hardware and software frameworks tailored to the needs of the GNN applications [2, 21, 23, 35, 77]. However, the lack of flexible datasets hinders their ability to evaluate the scalability of their systems as the datasets grow larger. Ideally, a dataset should provide sub-datasets of varying sizes that maintain the same graph structural properties to enable such comprehensive research on GNN systems. This is because the challenges faced with small datasets differ from those encountered with large datasets.

For instance, training a small IGB-HOM graph can be completed within a reasonable time frame as the graph datasets and features can fit within a single system’s memory. On the other hand, training a full version of the IGB-HET dataset with a single system with state-of-the-art software stack is currently challenging due to large memory requirements and complex software management techniques. To fully understand the potential and scalability of GNNs, it is essential to study their performance on a range of graph sizes and complexities.

3 IGB DESIGN
Illinois Graph Benchmark (IGB) datasets are designed to address the limitations discussed in § 2.2. IGB datasets are created using real-world data extracted from Microsoft Academic Graph [49] and SemanticScholar datasets [31] as discussed in § 3.2.

3.1 IGB Overview
The IGB brings unique opportunities to assist in understanding the impact of dataset creation on the GNN model’s accuracy. To this end, IGB addresses following set of challenges:

1. **Open data licensing and text accesses**: The use of datasets containing graphs, text, and embeddings to train emerging GNN models [51, 68, 73] is on the rise. However, data sources used to create the datasets must have open licensing for ease of adoption and derivative product creation. IGB meets open-data-license requirements (ODC-By-1.0) and provides a collection of datasets containing graphs, text, and embeddings for GNN and language model training.

2. **Large ground-truth**: IGB offers substantial ground-truth labels extracted from human-annotated data, eliminating the need for manually labeling millions of nodes. The majority of the IGB datasets are fully labeled, and the largest datasets have at least 40% of their nodes labeled, which is achieved by fusing multiple databases to form a large labeled dataset while preserving the accuracy of the information.

3. **Flexibility for ablation study**: The lack of flexible datasets limits the ability to evaluate the impact of the various components of the datasets on the performance of GNN models and understand their execution efficiency with different hardware and software systems. IGB overcomes this by offering a flexible research tool providing variable-sized embeddings, the ability to generate embeddings from different language models, and a range of variable-sized graphs with consistent homophily.

4. **Task complexity**: The IGB includes a range of multi-class classification tasks with varying degrees of complexity, which is crucial for evaluating the capabilities of GNN models. This is because while a GNN model can perform well on a binary classification task, it might not be effective for more fine-grain classification tasks. This feature in IGB also enables the investigation of efficient transfer learning techniques for fine-tuning downstream tasks.

3.2 IGB Dataset Generation Methodology
Generating IGB datasets involves curating data from various sources. Each of the IGB dataset has a ground truth labels, and node embeddings. We will first describe how we generate the graph by extracting information from real-world data.

**Input Data**: While there are a number of publicly available datasets that can be used as input source [1, 39, 53, 60], many of them are either small or lack the necessary information for building large-scale graph datasets for GNN application. To this end, IGB graphs are primarily created using the Microsoft Academic Graph (MAG) [49] database and carefully added human-annotated labels from both MAG and SemanticScholar Corpus [31] databases. Semantic Scholar data corpus is closely tied to MAG w.r.t to papers, as 89.4% of all Semantic Scholar paperIDs contain a field for unique reverse mapping with the MAG (a unique magPaperID).
Table 1: Comparison of IGB with the largest publicly disclosed existing dataset. Labelled nodes as a percentage of total nodes. input implies the sizes are dependent on the values provided to the generator.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Date</th>
<th>Availability</th>
<th>Type</th>
<th>#Nodes</th>
<th>Labelled*</th>
<th>#Edges</th>
<th>Dim</th>
<th>RawText</th>
<th>Flexibility</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN-SAGE</td>
<td>2018</td>
<td>Private</td>
<td>Real</td>
<td>3000 M</td>
<td>UNK</td>
<td>18 B</td>
<td>1024</td>
<td>No</td>
<td>No</td>
<td>Multi-class classification</td>
</tr>
<tr>
<td>papers10M</td>
<td>2020</td>
<td>Public</td>
<td>Real</td>
<td>111 M</td>
<td>&lt; 1%</td>
<td>1.6 B</td>
<td>128</td>
<td>No</td>
<td>No</td>
<td>172-class classification</td>
</tr>
<tr>
<td>mag-240M</td>
<td>2021</td>
<td>Public</td>
<td>Real</td>
<td>244 M</td>
<td>&lt; 1%</td>
<td>1.7 B</td>
<td>768</td>
<td>No</td>
<td>No</td>
<td>153-class classification</td>
</tr>
<tr>
<td>GraphWorld</td>
<td>2022</td>
<td>Public</td>
<td>Syn</td>
<td>input</td>
<td>N.A.</td>
<td>input</td>
<td>N.A.</td>
<td>No</td>
<td>No</td>
<td>System Design</td>
</tr>
<tr>
<td>SemanticScholar</td>
<td>2023</td>
<td>Public</td>
<td>Real</td>
<td>205 M</td>
<td>UNK</td>
<td>3 B</td>
<td>768</td>
<td>No</td>
<td>No</td>
<td>Multi-class classification</td>
</tr>
<tr>
<td>SynGen</td>
<td>2023</td>
<td>Private</td>
<td>Syn</td>
<td>input</td>
<td>N.A.</td>
<td>input</td>
<td>N.A.</td>
<td>No</td>
<td>Yes</td>
<td>System Design</td>
</tr>
<tr>
<td>IGB-HOM</td>
<td>2023</td>
<td>Public</td>
<td>Real</td>
<td>&gt; 260 M</td>
<td>&gt; 81%</td>
<td>4 B</td>
<td>128 to</td>
<td>Yes</td>
<td>Yes</td>
<td>19 or 2983-class classification</td>
</tr>
<tr>
<td>IGB-HET</td>
<td>2023</td>
<td>Public</td>
<td>Real</td>
<td>&gt; 547 M</td>
<td>&gt; 40%</td>
<td>6 B</td>
<td>1024</td>
<td>Yes</td>
<td>Yes</td>
<td>and System Design</td>
</tr>
</tbody>
</table>

Figure 1: IGB-HET dataset schema.

The MAG database is particularly useful as it contains a wide range of relationships and information about different types of data points, including papers and authors, as well as information about paper citations, authorship, affiliations, and field of study. SemanticScholar corpus [31], while slightly smaller than MAG, also provides a similar set of functionalities. However, what particularly sets both of these datasets apart is the explicit permission we received to release the raw text data under the ODC-By-1.0 licensing scheme.

The MAG comprises over 260 million entries for papers and about four billion relations representing a citation between two papers. The schema used by MAG has many tables among which paper, author, affiliation, URLs, conference/journals, and field of study tables are of interest. The paper table contains information such as title, published date, authors, citations, and names of journals or conferences where the article is published. The paper citations are stored in a file using coordinate (COO) graph storage format where the first paper ID cites the paired paper ID. The author table includes data on the author’s name, affiliations, and papers they wrote. SemanticScholar corpus [31] has a similar schema providing up to 205 million entries for papers and about three billion relations representing citation.

**IGB Dataset Graphs:** IGB provides two types of graphs: homogeneous and heterogeneous. The IGB homogeneous graph (IGB-HOM) is created by extracting only the paper nodes and the paper-cites-paper relation between these nodes. The IGB heterogeneous graph (IGB-HET) is created by extracting four different types of nodes: papers, authors, institutions, and fields of study. These different types of nodes are connected by various types of edges as shown in Figure 1.

The IGB dataset collection (homogeneous and heterogeneous) graphs are created only using the MAG database. Although the Semantic Scholar and MAG databases exhibit a close association with paper nodes, the relationships with other node types such as authors and institutions pose a significant challenge. As a result only the relations from the MAG database are utilized during the construction of IGB graphs. Combining the graph structure using Semantic Scholar and MAG would give us an additional 20 million (0.5%) edges but will introduce data inconsistencies. For example, MAG and SemanticScholar have author names mismatches despite referring to the same individual. Some of these inconsistencies can be addressed by emerging entity resolution techniques which are still being discussed in the research. We hope by releasing the raw data as a part of IGB we can help build better entity resolution techniques and eventually assist in building much larger graphs.

**IGB Ground-Truth Labels:** The challenge of obtaining a large collection of real-world ground truth labels is a major issue pertaining to dataset generation especially when manual annotation at the million scales is unattainable. Currently, the largest graph datasets such as MAG240M [26] and OGBN-paper100M utilize ArXiv [1] labels and have a maximum of 1.4 million nodes categorized into 172 paper topics and eight subject areas. However, such a tiny set of labeled nodes is insufficient for real-world applications as the model cannot accurately predict unseen data.

To address this, IGB extracts human annotated label information from the MAG [49] and SemanticScholar [31] databases. Extraction of information and aligning two databases pose a challenging problem. First, the two databases are distinct, and the nodes from both sides must be correctly aligned. We address this by leveraging the reverse mapping of MAG paper IDs from the SemanticScholar database and merging the databases to create the IGB graph.

Second, the two databases have different labeling methodologies. To handle this issue, we carefully create a union of distinct labels from the MAG and SemanticScholar databases. Third, the two databases can have different distinct labels for each paper and can create a merge conflict. We address this merge conflict by manually checking the very small number of nodes that have this issue and assigning the right label that appropriately defines the paper from the union of distinct labels from MAG and SemanticScholar databases. Using this merged database, IGB datasets can cover a large portion of the data with labeled nodes. As shown in Table 1, IGB provides

---

\*It is important to note that the MAG dataset is now fully deprecated and we thank the MAG database owners for giving us explicit open-sourcing permission.
more than 81% and 40% of nodes labeled for homogeneous and heterogeneous graphs.

The paper nodes in the IGB dataset collection are labeled using the below labeling rules. These rules are carefully crafted to ensure label consistency in the data while merging labels from both MAG and Semantic Scholar databases:

- **MAG has labels for the given paper node (58% of papers):** Regardless of whether Semantic Scholar has a label or not, if MAG has a label, we use it directly.
- **MAG does not have a label and Semantic Scholar has a label (25% of papers):** If MAG does not provide a label but Semantic Scholar has a label and a unique magID reverse mapping, we use the label from Semantic Scholar.
- **Neither MAG nor Semantic Scholar provides a label (17% of papers):** For papers that do not have labels in either MAG or Semantic Scholar, we consider them unlabeled.

As we prioritize MAG labels over Semantic Scholar labels, we do not have label consistency issues in terms of under/over merges.

**IGB Downstream Tasks:** The IGB dataset collections are designed for multi-class classification problems with two different numbers of classes (19 and 2983) depending on the degree of complexity. The 19 class task is curated by combining classes from MAG and Semantic Scholar and mapping them into a common structure. Examples of 19-class labels are history, geology, economics, and many more. The 2983 class task is created by bucketing all papers with the same set of paper topics from the set of labels provided by Semantic Scholar corpus. Examples of class labels are pediatrics, criminology, and computer engineering. The 19-class task is intended for model development and testing while the 2983-class task can be used to stress test the models and develop robust GNN models for noisy real-world data. Although the IGB dataset comprises node classification problems to solve, it is easy to extend the downstream task to train for edge prediction tasks such as citation recommendation or reviewer recommendation. For instance, an edge prediction task can be constructed by masking our existing edges in the graph, training an edge prediction model and use the masked edges as the ground truth output.

**IGB Embedding Generation:** GNN models operate on graph structure and their embeddings. Embeddings can be associated with a node or an edge and capture the nature of the nodes as well as the nature of relationship between the nodes and help the GNN models to extract deeper semantic information from the graph. In the past, one-hot vectors and word dictionaries were used to generate these embeddings. However, with the introduction of word2vec embeddings and more recently deep learning-based word and text embeddings, GNNs are being initialized with embeddings generated using foundational language models. For IGB datasets, we generate embeddings for each node in the graph 3. Node embeddings are generated by passing the paper titles and abstracts through a Sentence-BERT model [48] and obtaining a 1024-dimensional vector for each paper node. Sentence-BERT, based on Siamese network, can generate semantically meaningful sentence embeddings by modifying a pre-trained BERT model [16] such as RoBERTa [38] or GPT [10] while maintaining high accuracy with least run-time overheads.

As IGB graphs are extracted from scientific data [31, 49], it is possible to create domain-specific embeddings instead of general language model-based embeddings. For scientific text, SPEcTRE [12], an embedding model created using SciBERT [4], a variant of BERT, can be used. SPEcTRE [12] uses positive and negative samples of the papers from the Semantic Scholar corpus [31] to optimize the embeddings for the scientific text, leading to a 1.5% gain in the F1 score in the 19-class task compared to the Sentence-BERT model when tested on papers from the MAG dataset.

We chose to use sentence-BERT embeddings trained on web data as the default IGB embeddings, as we want to benchmark the GNN model’s ability to extract structural information from the embeddings that do not contain inherent fine-tuning towards a specific domain. This also reflects the practical considerations of real-world industrial settings where fine-tuning language models for each domain-specific task is time-consuming and expensive.

The IGB author node embeddings are generated by taking the average of the node embeddings of all the papers written by the author, a methodology followed by Hu, et al. [26]. Institute node embeddings are generated similarly taking the average of the node embeddings of all the authors affiliated with the particular institute.

---

1. Although IGB can generate edge embeddings, from the data management perspective it is quite challenging due to the sheer size of the generated dataset.
Field of study node embeddings is generated using the topic name provided in the field of study in the database.

3.3 IGB Datasets

The IGB dataset suite offers five datasets for both homogeneous and heterogeneous graphs with varying sizes for deep learning practitioners as shown in Table 2 and Table 3. Each dataset is larger than the previous one (by about an order of magnitude) and is designed for a specific purpose. Each of the smaller datasets are created by carefully sampling the graph such that we maintain similar homophily across different dataset variations. Homophily of the IGB dataset varies from 47.75% to 59.93% consistent with the homophily of prior released citation graphs [1, 26]. The tiny dataset is meant for testing and development of GNN models and can be run on a laptop or mobile or an edge device. The small and medium datasets are ideal for training and testing new GNN models during development and can be trained with lower to mid-end GPUs or a powerful CPU. The small and medium datasets are also representative of large-scale labeled datasets currently available to the public [26]. The large dataset requires significant computing resources and can be trained on high-end accelerators and is suitable for building robust GNN models and testing by system designers. The full dataset is a massive dataset for GNN developers to construct practical models and stress test the distributed training platforms.

Each dataset is initialized with 1024-dimensional node embeddings and has two different output classes that increase the difficulty to stress testing the GNN models and optimizing model development. The datasets are randomly split with 60% for training, and 20% each for validation and testing. The dataset also includes the year of publication metadata for every paper node in case the user wants to set a specific splitting rule.

4 IGB DATASET CASE STUDIES

The IGB dataset flexibility enables extensive ablation study to understand the impact of dataset generation technique on the GNN model performance. Through these case studies, we make the following key observations:

- GNN models for node classification tasks observe up to 12.96% boost in their accuracy as the fraction of labeled nodes in the dataset increased from 1/150 to fully labeled.
- Using an NLP model for initializing node embeddings results in 40% increase in GNN accuracy over random node embeddings.
- Among the different generic NLP model tests, RoBERTa NLP model provides the best overall performance.
- Reducing the embedding dimension from 1024-dim to 384-dim results in 2.67% memory saving with a maximum 3.55% average loss in the GNN models accuracy.

4.1 Setup

Models: We present performance benchmarks for three commonly used GNN models (GCN [32], GraphSAGE [25], and GAT [59]) on the IGB homogeneous dataset and for three popular GNN models (RGCN [51], RSAGE\(^4\) and RGAT [11]) on the IGB heterogeneous dataset. All models are trained with 0.01 learning rate with two layers. Most of the models are trained to three epochs and numbers are reported. Unless explicitly stated, we used a batch size of 10K to maximize GPU utilization and used IGB-medium as the default dataset. Lastly, most evaluations are reported with homogeneous IGB datasets but are equally applicable to heterogeneous datasets.

System: We conducted our evaluations on a high-performance server system with dual AMD EPYC 7R32 processors, 256GB of DRAM, and an NVIDIA A100-40GB PCIe GPU with NVMe SSDs. The experiments were carried out using the DGLv0.9 framework [63] built on top of PyTorch [44] as obtained from the NVIDIA NGC repository. To reflect real-world cost considerations in industry, all experiments were run on a single EC2 instance.

4.2 Impact Of Labeled Nodes

In Section 2.2, we emphasized the importance of labeled data. In this section, the impact of labeled data is evaluated for the IGB medium dataset. To achieve this, sub-datasets with varying fractions (i.e., 1/150, 1/100, and so on.) of labeled data were created and then trained and evaluated using the GCN, GraphSAGE and GAT models on the 19-class node classification task. 1/150 labeled sub-dataset is representative of the largest publicly available labeled dataset, MAG240M [26] and is created by picking one labelled node every 150 nodes. As shown in Figure 2, the model accuracy improved by up to 10% as more labeled data was added. Increasing the classification task complexity from 19-class to 2983-class shows a similar trend in the performance. This is expected behavior as these models are trained as supervised learning tasks and more labeled data should help in boosting the model’s performance.

The experiment was also performed on three sub-variants of the homogeneous dataset and on medium size IGB heterogeneous dataset. As shown in Table 4, adding more labeled data helps
Table 5: Impact of labeled nodes on IGB using SAGE compared to MLP.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLP (2 layer)</th>
<th>SAGE</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGB-HOM-tiny (1/150 labeled)</td>
<td>47.89%</td>
<td>63.97%</td>
<td>16.08</td>
</tr>
<tr>
<td>IGB-HOM-tiny (all labeled)</td>
<td>64.00%</td>
<td>69.38%</td>
<td>5.38</td>
</tr>
<tr>
<td>IGB-HOM-small (1/150 labeled)</td>
<td>52.90%</td>
<td>57.20%</td>
<td>4.30</td>
</tr>
<tr>
<td>IGB-HOM-small (all labeled)</td>
<td>73.18%</td>
<td>75.49%</td>
<td>2.31</td>
</tr>
<tr>
<td>IGB-HOM-medium (1/150 labeled)</td>
<td>52.11%</td>
<td>58.15%</td>
<td>6.01</td>
</tr>
<tr>
<td>IGB-HOM-medium (all labeled)</td>
<td>70.75%</td>
<td>70.30%</td>
<td>-0.45</td>
</tr>
</tbody>
</table>

Table 6: Selected Sentence Transformer Models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Emb dim</th>
<th>Avg. Acc</th>
<th>Model size</th>
</tr>
</thead>
<tbody>
<tr>
<td>all-MiniLM-L6-v2</td>
<td>384</td>
<td>58.80%</td>
<td>80 MB</td>
</tr>
<tr>
<td>distiluse-base-multilingual</td>
<td>512</td>
<td>45.59%</td>
<td>480 MB</td>
</tr>
<tr>
<td>all-mpnet-base-v2</td>
<td>768</td>
<td>63.30%</td>
<td>420 MB</td>
</tr>
<tr>
<td>all-roberta-large-v1</td>
<td>1,024</td>
<td>61.64%</td>
<td>1,360 MB</td>
</tr>
</tbody>
</table>

Figure 3: NLP embeddings help the GNN model to learn both structural and node properties while random embeddings can only learn from the structure.

Improve the model performance significantly even with the IGB dataset variants including heterogeneous datasets. This is promising mainly because, with datasets that are 100% labeled, the GNN developer can develop more accurate models and not be constrained by a lack of labeled data.

Comparison with MLP: The strength of GNNs, typically manifests when the label fraction is relatively small. According to the Table 5 above, GNN models exhibit better performance compared to the supervised MLP baseline for the homogeneous graph, when the labeled data constitutes a small fraction of the overall dataset. However, when the dataset is fully labeled, the benefit provided by the GNN models is comparable to the naive MLP baseline with two layers.

While the accuracy of the GNN model surpasses that of the MLP baseline when a small fraction of the dataset is labeled, its throughput is an order of magnitude lower than that of the MLP baseline. From a system design perspective, this raises the question of whether we should consider developing a more complex MLP architecture for homogeneous graph datasets.

4.3 Embedding Generation Ablation Study

Embeddings are key to the GNN model’s performance but none of the prior work has shown how they impact GNN’s performance. In this section, we will discuss the importance of embedding, and its generation process in-depth.

Random vs. NLP embeddings: The GNN performance depends on the node information provided to the model, which is initialized as node embeddings. In this evaluation, we compared the difference in performance between randomly initialized vectors and RoBERTa [38] NLP-based embeddings on three IGB homogeneous datasets as shown in Figure 3. Each bar graph in Figure 3 is an average accuracy score of three GNN models (GCN, GraphSAGE, and GAT) on the respective datasets. Our results demonstrate that the NLP embeddings significantly increase the model’s performance (up to 3x) because the GNN model learns both the structural information and the node’s properties. With the random embeddings model can only learn from the graph structure and cannot perform well in the multi-class classification task.

Selecting Right Language Model For Embeddings: The quality of embeddings generated by the language model can have a significant effect on the GNN model accuracy. To this end, we evaluated the GNN accuracy using a number of widely used language models (see Table 6) using the sentence transformer package provided by the Huggingface library [64]. Table 6 summarizes the average language model performance on NLP tasks, their respective model size, and generated text embedding dimensions.

Using these language models, we evaluate the performance of GNN models on the IGB-tiny dataset. To do this, we generated embeddings using each of the language models independently and trained all the GNN models using the respective language embeddings. Figure 4 shows the impact of these language models on the GNN accuracy. The average accuracy is calculated by averaging the reported accuracy of GCN, GraphSAGE, and GAT models. Although with this evaluation, it is not possible to concisely determine if the large performance range is due to the accuracy of the model or the embedding dimensions, it is clear that the language model itself has a direct impact on the GNN performance. In our next evaluation, we will isolate the impact of the model by reducing the dimensions of the embeddings.

Impact Of Embedding Dimension: We normalized the different-sized embeddings from various language models using principal component analysis (PCA), a dimensionality reduction technique [20]. We used the GPU implementation of PCA from the NVIDIA cuML library [46]. The results of the experiments using GCN, GraphSAGE, and GAT models with varying embedding dimensions are presented in Figure 5 and Figure 6.

From Figure 5, both mpnet [56] and RoBERTa [38] models outperform the other language models while training GNN. This is
because both these NLP models are inherently better language models (see Table 6) and their respective accuracy improvements are reflected in the GNN model’s accuracy.

From the figures, reducing the embedding dimensions using PCA had an impact on the accuracy across all the GNN models. This is not surprising as the PCA is a lossy pruning technique. For RoBERTa [38] embeddings, reducing the dimensions from 1024 to 384 results in at least 2.67× memory capacity savings during training and inference operation with up to 3.55% loss in accuracy for the GCN, GraphSAGE, and GAT models, respectively. The GAT model was found to be more resilient to reductions in dimensionality when compared to the GraphSAGE and GCN models. A similar observation is noted in other language models.

Reducing the dimensions from 1024 to 384 results in at least 2.67× memory capacity savings during training and inference operation with up to 3.55% loss in accuracy for RoBERTa embeddings. Further reducing the dimensionality offers significant memory footprint reduction but results in a significant drop in the GNN model performance as shown in Figure 6. A similar set of observations are also seen with different IGB datasets including IGB-HOM graphs and are not reported. Based on this analysis, the IGB dataset collection provides downloadable embeddings from the RoBERTa language model and provides a toolkit to generate other embeddings in our open-source codebase [30].

We also analyzed the impact of language that is used to train NLP model makes any difference in the GNN performance. The results of this study can be found in Appendix B.

4.4 Overall Summary
The overall results of the IGB datasets for all the GNN models with two complex tasks are summarized in Table 7 and Table 8. The IGB-HOM-tiny and IGB-HOM-small datasets are trained for 20 epochs while the rest are trained for three epochs due to the long training time. The accuracy of the models generally decreases in the more complex 2983 tasks as the model must identify finer differences in the nodes to classify them into a large number of classes. We observe up to 14.93% drop in performance for IGB-HOM dataset. Longer training (more epochs) may help recover some of these accuracy drops, but ideally, better models are needed to handle the finer classification task.

Surprisingly the relational GNN models are tolerant to complex multi-class problems on the IGB-HET dataset. Despite an increase in task complexity from 19 classes to 2983 classes, the accuracy of relational GNN models does not decline. We believe the extra structural information present in the heterogeneous graph enables these models to classify with greater precision. Further investigation is necessary to fully comprehend this phenomenon.

4.5 Existing System Limitations
The existing system is unable to handle the complexity of training GNN models using the full IGB dataset. This is due to the large size of the embedding tables and graphs (see Table 2 and Table 3), which require large memory capacity. Despite the use of state-of-the-art systems and software, training GNN models on the full IGB dataset on a single system is still a challenge and time-consuming task. Let’s briefly describe how GNN training occurs before discussing the system limitations.

GNN training operation can be split into three key stages: sampling, aggregation, and computation. The first stage in GNN training is to sample nodes or sub-graphs from the graph to form a mini-batch. In the aggregation stage, information from the neighborhood of each node present in the mini-batch is aggregated to form a node representation. This process involves reading node embeddings and forming an aggregated representation for the node. In the final stage, the aggregated node representation is fed into a neural network to train the model.

As graphs and embeddings come in varying sizes, frameworks such as DGL [63] and PyG [19] provide a different mechanism to optimally place graphs and embeddings in the system for efficient training execution. If the dataset fits in the GPU memory (e.g. IGB-HET-tiny and IGB-HET-small), then both graphs and embeddings can be preloaded to the GPU memory and then GNN models can be trained. Our observation is that, when the datasets fit in the GPU memory, we achieve up to 80% GPU utilization (average 50%) during training.

If the dataset fits in the host CPU memory (e.g. IGB-HET-medium and IGB-HET-large), the graph is either placed in the GPU memory or the host memory, depending on the size of the graph, while the embeddings are loaded in the host memory. During the training operation, the GPU threads can directly sample the graphs from GPU memory and issue zero-copy memory-mapped I/O access to the embeddings using the DGL-UVA [40, 41] technique for efficient execution. Our measurement shows that, in this case, we achieve up to 80% GPU utilization (average 50%).

If the graph and embeddings do not fit in the host memory of a single system, as is the case of the IGB dataset, then embeddings can be memory-mapped ( mmap ) and be stored in fast storage medium like NVMe SSDs. The intuition here is that, even though each
Table 7: IGB-HOM datasets benchmark results using the same GNN model parameters (* trained for 3 epochs). N.G. stands for cannot finish training within time limit.

<table>
<thead>
<tr>
<th>Model</th>
<th>GCN</th>
<th>SAGE</th>
<th>GAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset (# class)</td>
<td>19</td>
<td>2983</td>
<td>19</td>
</tr>
<tr>
<td>IGB-HOM-tiny</td>
<td>68.06</td>
<td>53.13</td>
<td>71.87</td>
</tr>
<tr>
<td>IGB-HOM-small</td>
<td>70.46</td>
<td>63.28</td>
<td>75.49</td>
</tr>
<tr>
<td>IGB-HOM-medium*</td>
<td>70.63</td>
<td>62.70</td>
<td>70.35</td>
</tr>
<tr>
<td>IGB-HOM-large*</td>
<td>50.29</td>
<td>N.G.</td>
<td>64.89</td>
</tr>
<tr>
<td>IGB-HET*</td>
<td>48.59</td>
<td>N.G.</td>
<td>54.95</td>
</tr>
</tbody>
</table>

Table 8: IGB-HET datasets benchmark results using the same GNN model parameters (* trained for 3 epochs). N.G. stands for cannot finish training within time limit.

<table>
<thead>
<tr>
<th>Model</th>
<th>RGCN</th>
<th>RSAGE</th>
<th>RGAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset (# class)</td>
<td>19</td>
<td>2983</td>
<td>19</td>
</tr>
<tr>
<td>IGB-HET-tiny</td>
<td>66.73</td>
<td>67.66</td>
<td>67.79</td>
</tr>
<tr>
<td>IGB-HET-small</td>
<td>71.35</td>
<td>71.64</td>
<td>72.54</td>
</tr>
<tr>
<td>IGB-HET-medium*</td>
<td>71.85</td>
<td>71.79</td>
<td>72.65</td>
</tr>
<tr>
<td>IGB-HET-large*</td>
<td>N.G.</td>
<td>N.G.</td>
<td>N.G.</td>
</tr>
<tr>
<td>IGB-HET*</td>
<td>N.G.</td>
<td>N.G.</td>
<td>N.G.</td>
</tr>
</tbody>
</table>

Figure 7: Execution time breakdown of the three stages involved while training GCN and RGCN model. The majority of the time is spent either on the sampling operation or on the aggregation step.

mini-batch training requires a small working set size (~600MB for IGB-HET in the GPU memory, the entire graph and embeddings need to be accessible during a training operation. This allows the frameworks like DGL [63] and PyG [19] to work on the embedding tables that exceed the host memory capacity of a single system.

Figure 7 illustrates the distribution of the time spent for each of the three stages of training GCN and RGCN models on various IGBs graphs. As shown in Figure 7, the use of mmap approach leads to a considerable slowdown, even when the graphs and embeddings fit in the host memory, as much as 4.5x (see IGB-large column). Unlike UVA, the mmap abstraction requires the CPU threads to perform the sampling and aggregation stages, accessing the embeddings stored in NVMe SSDs through the operating system page cache, leading to significant overhead. For IGB-HET graphs, the node aggregation stage consumes the most significant fraction of iteration time when using the mmap approach, causing low average GPU utilization, less than 5%. Profiling reveals that the mmap approach can only achieve up to 25% of storage bandwidth (1Gbps) and is mainly limited by the system’s page fault handler and page-cache throughput.

5 DISCUSSION

The main motivation for IGB datasets creation is to have a dataset that can be used by academia and the industry for system designs and GNN model development. IGB enables:

- Better experiments for GNN model development: IGB is created to highlight the inefficiency in existing GNN models. As shown in §4, most widely used GNN models only provide a marginal benefit over the naive MLP baseline when datasets are fully labeled, indicating that much more research is needed to further improve GNN model’s accuracy.

- Raw text and embedding: IGB addresses the lack of large-scale graph datasets with rich labeling information and adaptable node features. This dataset also offers GNN model developers access to raw text data which is crucial for training emerging GNN models like GraphFormer [67] and GLEM [75]. This also enables evaluating the importance of embedding generation and the significance of embedding quality during GNN model training. The IGB dataset is currently the only large-scale dataset for these purposes.

- System scaling: IGB is designed to cater support for emergent research and engineering topics for applying GNN in the industry as well. Most of the existing datasets focus on GNN model performance while IGB, in additional to model performance, enables scalable and efficient system and framework design for GNN deployment in the industry. The IGB presents new system design challenges that previous datasets could not capture. For instance, when training with the DGL on the SAGE model using the OGB MAG240M dataset, we never encountered out-of-memory (OOM) issues as the dataset fits within the << 1TB CPU memory (the largest single GPU instance in AWS ~ p4dn.24x has only 1TB memory).

However, training with the full IGB datasets, whether homogeneous or heterogeneous, on a single CPU server is impractical and requires the development of efficient distributed GNN training methodologies or efficient access to storage. This necessitates the discovery of efficient graph partitioning schemes, communication primitives, and evaluating systems with strong and weak scaling studies. Many of these experiments cannot be effectively conducted due to the lack of large, scalable, and meaningful datasets. The IGB dataset collection addresses these requirements head-on.

6 CONCLUSION

This work introduces IGB, a research tool for deep learning practitioners to thoroughly evaluate and test GNN models with accuracy. It provides access to a large graph dataset that includes both homogeneous and heterogeneous graphs, with more than 40% of their nodes labeled. IGB has been designed to be flexible, offering options for the examination of various GNN architectures, embedding generation techniques, and analyzing system performance while training or inferencing GNN models. IGB is open-sourced, compatible with DGL and PyG frameworks, and includes raw text data that...
can inspire new research at the intersection of natural language processing and graph neural network research topics.

ACKNOWLEDGMENTS

We would like to acknowledge all of the help from members of the IMPACT research group, NVIDIA GNN team, and AWS Graph ML team without which we could not have achieved any of the above reported results. Special thanks to Dr. Zaid Qureshi from NVIDIA Research and Jeongmin Park from the IMPACT research group for their valuable suggestion on writing and performance evaluations. We also would like to acknowledge feed-backs received anonymous reviewers and Dr. Piotr Bigaj from NVIDIA. This work is partly funded by the Amazon Research Awards. This work uses GPUs donated by NVIDIA, and is partly supported by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) and the IBM-ILLINOIS Discovery Accelerator Institute (IDIA). The datasets in this work is released to public in cooperation with Amazon using AWS Open Data Sponsorship Program.

REFERENCES

A GNN OVERVIEW

GNNs are inspired by the idea of extending neural networks that are usually defined for vector inputs to graph-structured inputs. The key idea of GNNs is to pass messages between the nodes in a graph. They use a neural network to propagate information along the edges of the graph while updating the representations of the nodes as the model learns. This enables the GNN to take into account the relationships between the nodes while learning their representations and properties. GNNs can be used to solve a wide variety of tasks such as node classification, link prediction, and graph classification, commonly found in applications such as fraud detection [18, 47, 74], protein structure discovery [29, 57], and recommendation system [61, 71, 72].

GNNs can be applied to both homogeneous graphs and heterogeneous graphs. Different types of GNN models have been developed to operate on these graphs. For homogeneous graphs, the most popular GNN models are graph convolutional network (GCN) [32], GraphSAGE [25], and graph attention network [59]. These models primarily differ in how they pass messages between the nodes to learn the representation. For heterogeneous graphs, the most popular ones are relational-GCN [51] and relational-GAT(RGAT) [11]. While there are several GNN models, we can formalize all their update functions into a single equation:

$$h_u^k \leftarrow \text{UPDATE}(h_u^k, \text{SAMPLE}(h_u^{k-1}))$$

Depending on the number of types of relations and neighbor sampling method we can derive respective updated functions of the popular graph neural net models as shown in Table 9, where,

- $h_u^k$: embedding of node $v$ in the $k$ layer of the GNN.
- $\sigma$: non-linear activation.
- $W_r^k$: weight matrix of GNN layer $k$ (for relation $r \in R$), $W_r$ for $r = 1$.
- $R$: set of all the relations in the graph.
- $\alpha_{uv}$: attention coefficient between node $u$ and $v$.
- $c_{uv}$: degree normalizing factor between node $u$ and $v$.
- $\mathcal{N}(v)$: neighborhood of node $v$ (for $r \in R$).

Note that these equations ignore the self-node consideration and the bias term which remains the same for all models.

B GNN INPUT LANGUAGE INFLUENCE

Scientific databases such as MAG [49] and SemanticScholar [31] consist of papers that are written in more than 80 languages. For instance, MAG [49] database has more than 12 languages with over 1 million papers and only 50% of the total papers are written in English. In order to understand the impact of language on the accuracy of GNN models, it is important to consider the language models used to generate node embeddings, which are typically trained on web corpus data [10, 38].

Our goal is to determine if the language used in the dataset to train the NLP model makes any difference in the GNN performance. To do this, the study would require a language model trained exclusively on a specific language. However, finding such a pre-trained large language model is currently impossible. As a result, the study used multilingual language models that included or excluded specific languages.

We generated two types of embeddings: one using a language model trained in English and the other using a multi-lingual language model that included Japanese, Spanish, and French (multilingual embedding). The results, shown in Table 10, indicate that there is no significant performance improvement achieved by including specific languages, regardless of the model dimensions and language model used. We found similar results when running different embedding dimension models on the French and Spanish datasets. This makes us believe that the GNNs models are likely language agnostic.

Table 9: Popular GNN models’ update function comparison.

<table>
<thead>
<tr>
<th>Model</th>
<th>IGB-japanese</th>
<th>IGB-spanish</th>
<th>IGB-french</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN [32]</td>
<td>62.89</td>
<td>59.77</td>
<td>60.39</td>
</tr>
<tr>
<td>GraphSage [25]</td>
<td>62.83</td>
<td>59.01</td>
<td>59.48</td>
</tr>
<tr>
<td>GAT [59]</td>
<td>768 – eng</td>
<td>64.82</td>
<td>61.63</td>
</tr>
<tr>
<td>R-GCN [51]</td>
<td>768 – all</td>
<td>64.74</td>
<td>61.77</td>
</tr>
<tr>
<td></td>
<td>512 – v2</td>
<td>61.61</td>
<td>58.42</td>
</tr>
<tr>
<td>Average</td>
<td>61.72 ±0.11</td>
<td>58.64 ±0.22</td>
<td>57.26 ±0.22</td>
</tr>
</tbody>
</table>

Received 26 February 2023; accepted 5 June 2023