GraphZoo: A Development Toolkit for Graph Neural Networks with Hyperbolic Geometries

Anoushka Vyas, Nurendra Choudhary, Mehrdad Khatir, Chandan K. Reddy
Department of Computer Science, Virginia Tech, Arlington, VA, USA
{anoushkav,nurendra,khatir}@vt.edu,reddy@cs.vt.edu

ABSTRACT

Hyperbolic spaces have recently gained prominence for representation learning in graph processing tasks such as link prediction and node classification. Several Euclidean graph models have been adapted to work in the hyperbolic space and the variants have shown a significant increase in performance. However, research and development in graph modeling currently involve several tedious tasks with a scope of standardization including data processing, parameter configuration, optimization tricks, and unavailability of public codebases. With the proliferation of new tasks such as knowledge graph reasoning and generation, there is a need in the community for a unified framework that eases the development and analysis of both Euclidean and hyperbolic graph networks, especially for new researchers in the field. To this end, we present a novel framework, GraphZoo, that makes learning, designing and applying graph processing pipelines/models systematic through abstraction over the redundant components. The framework contains a versatile library that supports several hyperbolic manifolds and an easy-to-use modular framework to perform graph processing tasks which aids researchers in different components, namely, (i) reproduce evaluation pipelines of state-of-the-art approaches, (ii) design new hyperbolic or Euclidean graph networks and compare them against the state-of-the-art approaches on standard benchmarks, (iii) add custom datasets for evaluation, (iv) add new tasks and evaluation criteria.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; *Learning latent representations*; • **Information systems** → *Open source software*.

KEYWORDS

graph learning, graph neural network, hyperbolic models, software

ACM Reference Format:
Anoushka Vyas, Nurendra Choudhary, Mehrdad Khatir, Chandan K. Reddy. 2022. GraphZoo: A Development Toolkit for Graph Neural Networks with Hyperbolic Geometries. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487553.3524241

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04.
https://doi.org/10.1145/3487553.3524241

1 INTRODUCTION

Graphs have been an integral part of research in the web domain. From inter-connected webpages that form the internet to the relational databases that support the software infrastructure, development of graph processing techniques and tools has been critical for several web applications [9]. Recently, the advancements in this topic has been primarily driven by the advent of neural networks, specifically, graph neural networks (GNNs). Similar to other fields such as Natural Language Processing and Computer Vision, research in the graph domain is now suffering an overload of models, i.e., too many new models are being developed at a rapid pace. Such a rapid growth in model development increases the complexity of tracking research in this area, specifically in terms of reproducibility or reusability of the models. Furthermore, without standardization, new researchers in this area will need to spend a significantly higher time exploring and understanding the entire literature, and further reproducing the models for their applications. The situation is further aggravated by the lack of public codebases or the availability of stand-alone codebases that work on specific datasets/tasks. In such cases, one needs to perform several tedious tasks such as data processing, parameter configuration, and optimization tricks to apply the models to their problem setting. The other related domains overcame these issues by standardization with systematic frameworks such as MatchZoo [6] and ModelZoo[1]. Inspired by this, we introduce a systematic framework for research in graph processing called GraphZoo to overcome the challenges in the graph research community. GraphZoo contains two interfaces: (i) the GraphZoo library that contains standard graph datasets and preprocessors, extensible modules of Riemannian manifolds, hyperbolic layers, and evaluation methods, and (ii) the GraphZoo investigator that allows researchers to integrate their datasets, models, or evaluation pipelines to compare the baselines in their experimental setups.

Using our GraphZoo framework[2], researchers can:

(1) **Systematically train models** by using the training/testing pipelines that include graph data pre-processors, off-the-shelf state-of-the-art layers, models, optimization methods, objective functions, and standard evaluation pipelines for both Euclidean and hyperbolic spaces.
(2) **Quickly develop new models** with the help of APIs to various graph-based hyperbolic/Euclidean layers and manifolds.
(3) **Compare new models** against state-of-the-art baselines on standard datasets with the help of pre-defined evaluation pipelines.
(4) **Perform hyper-parameter tuning** by parallelly running a rapid grid search over a set of configuration files.

[1] https://modelzoo.co/
[2] Implementation: https://github.com/AnoushkaVyas/GraphZoo

The GraphZoo system is built upon open-source framework [1] with updated libraries and new interfaces. There are other libraries in this domain such as DGL [8] and torch-geometric [4]. However, their focus is on building layers and data processing modules for Euclidean networks. GraphZoo is uniquely focused on improving the accessibility to hyperbolic networks [2, 3, 5] and providing complete evaluation pipelines for further development.

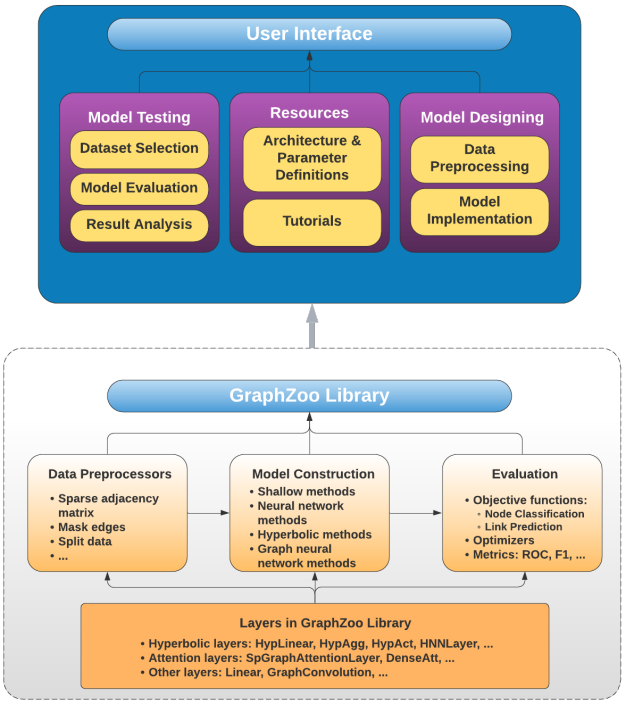

Figure 1: An overview of the GraphZoo framework.

2 GRAPHZOO LIBRARY

The overall architecture of the framework is given in Figure 1. The framework aids the three stages of model development, namely, data preparation, model construction, and evaluation. The library provides a number of pre-processed datasets, popular neural networks (such as graph neural networks and hyperbolic neural network models) as well as task-specific evaluation metrics and loss functions. Moreover, it is convenient to change various parameters related to data preparation, hyper-parameter tuning, and model selection in the library during experimentation. The framework also eases the process of training and testing models on new datasets and creating novel graph processing frameworks.

The GraphZoo library primarily provides experimentation pipelines for node classification and link prediction on graph data. To perform these tasks, the library has three main modules, namely data preparation, model construction, and evaluation. In our library, as shown in Figure 2, these three modules are independent of each other to enable user flexibility to perform all the tasks together or separately with their own custom parameters, datasets, and models.

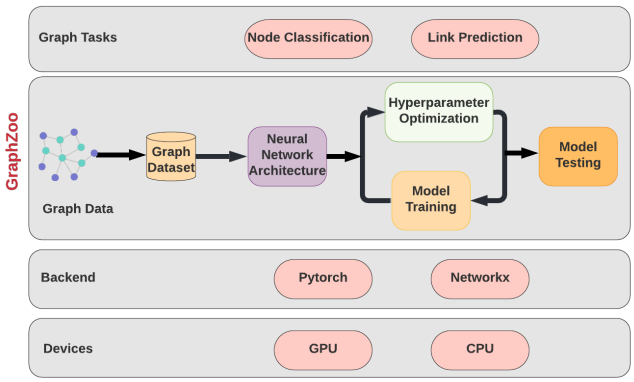

Figure 2: An overview of the GraphZoo library.

2.1 Data Preprocessors

The data preprocessor aims to convert the raw graph data into the input format for the model. This module, by default, handles benchmark graph datasets like Cora, Pubmed, Disease, and Airport [1]. In addition, it provides a number of processing units to convert raw graph data to the format required depending on the downstream task. All the processing units, take as input the fundamental adjacency matrix and node features and can be trivially combined together to achieve the format requirements of different tasks. After converting the raw dataset to the desired format, this module provides functions to split the dataset into train, test, and validation data depending on the default splits or user provided splits.

2.2 Model Construction

In the model construction module, we extend the PyTorch library with hyperbolic manifolds and layers to help users conveniently build new architectures. The GraphZoo library provides a set of common layers widely used in graph neural network models, such as graph convolution (GCN), graph attention (GAT), and hyperbolic layers (H-MLP, H-GCN, and H-GAT). The library also contains pre-implemented modules and common graph network baselines for easy reference and reproducibility:

- **Manifolds:** Euclidean, Hyperboloid, and Poincaré Ball (includes hyperbolic objective functions).
- **Optimizers:** Riemannian Stochastic Gradient Descent (RSGD), and Riemannian Adam (RADAM).
- **Activations:** Hyperbolic activation.
- **Layers:** Hyperbolic linear regressor, hyperbolic non-linear regressor, Graph Convolution, Graph Attention, Hyperbolic Graph Convolution, and Hyperbolic Graph Attention.
- **Models:** Encoder-decoder models for all the layers.

2.3 Evaluation

For experimentation, the library provides a variety of objective functions for node classification and link prediction, e.g., binary cross entropy and L2-norm for Euclidean and hyperbolic node classification, respectively. Furthermore, the library provides several widely adopted evaluation metrics, such as Precision, ROC, and

F1 score. The framework provides a choice of manifolds, layers, optimizers, activations, models, and evaluation metrics that can be seamlessly integrated to create novel model architectures for systematic development and evaluation of graph processing pipelines.

3 GRAPHZOO INVESTIGATOR

```
#Importing Libraries
import graphzoo as gz
import torch
from graphzoo.config import parser

#Defining Parameters
params = parser.parse_args(args=[])

#Preparing Data
params.dataset='cora'
params.datapath='data/cora'
data = gz.dataloader.DataLoader(params)

#Building Model
params.task='nc'
params.model='HGCN'
params.manifold='PoincareBall'
params.dim=128
model= gz.models.NCModel(params)

#Defining Optimizer
optimizer = gz.optimizers.RiemannianAdam(
            params=model.parameters(),
            lr=params.lr,
            weight_decay=params.weight_decay)

#Training and Testing
trainer=gz.trainers.Trainer(params,model,optimizer,data)
trainer.run()
trainer.evaluate()
```

Figure 3: A GraphZoo code snippet to run H-GCN model for node classification task on CORA dataset.

```
#Importing Libraries
import graphzoo as gz
import torch
from graphzoo.models import Encoder

class DemoModel(Encoder):
    """
    Pseudo Code for Demo Encoder Class
    args: input parameters (user defined)
    c: hyperbolic radius
    adj: adjacency matrix
    x: embeddings
    """
    def __init__(self, args, c):
        super(DemoModel, self).__init__()
        #Define neural network layers
        self.c = c
        self.encode_graph = False
        self.layers = ...

    def encode(self, x, adj):
        #Define forward pass
        if self.encode_graph:
            input = (x, adj)
            output, _ = self.layers.forward(input)
        else:
            output = self.layers.forward(x)
        return output
```

Figure 4: Pseudocode for creating a new model in GraphZoo.

The GraphZoo investigator provides an interactive notebook interface for new users to easily configure and run models which are already implemented to learn and explore state-of-the art graph neural network models. A comprehensive tutorial including theoretical descriptions and implementation details of various model components is provided for speedy initiation of new researchers into the development framework. The model is also well documented for a better understanding to advanced users. Figure 3 provides an example code snippet from the library to train H-GCN on Cora dataset for the node classification task. Various other graph processing pipelines can be similarly run with a trivial change of parameters. Advanced users can create their own models and adopt new datasets by inheriting the various layer APIs and data preprocessors provided in the toolkit. A tutorial for the same is also included in the framework. Figure 4 provides a pseudo-code that shows the creation of custom encoder model to perform experiments.

4 REPRODUCIBILITY EXPERIMENTS

In this section, we discuss the datasets, and reproduce the results of various baselines for the downstream tasks of node classification and link prediction in networks. We also perform time and memory analysis of the models to show the efficiency of our library[3].

Datasets: The library is tested on various benchmark graph datasets which are described below:

- **Citation Networks:** Cora and Pubmed are standard benchmarks describing citation networks where nodes are scientific papers and edges are the citations between them. The node labels in these datasets are academic (sub)areas. Cora contains 2,708 scientific publications divided into 7 classes, and Pubmed has 19,717 publications in the area of medicine grouped into 3 classes.
- **Disease Propagation Tree:** This dataset is taken from [1] where they simulate the SIR disease spreading model [7]. The label of a node is whether the node was infected or not. Based on the model, they build tree networks, where node features indicate the susceptibility to the disease.
- **Flight Networks:** Airport is a transductive dataset where nodes represent airports and edges represent the airline routes as from OpenFlights.org. This dataset has the size of 2,236 nodes. This dataset is taken from [1], where the authors also augment the graph with geographic information (longitude, latitude, and altitude), and GDP of the country where the airport belongs to. They use the population of the country where the airport belongs to as the label for node classification.

Algorithms: We aim to reproduce the results of our algorithms, as provided in this paper [1]. The algorithms considered in our experiments are: Euclidean Linear (E-Linear), Multilayer Perceptron (MLP), Graph Convolution (GCN) and their hyperbolic variants (H-Linear, H-MLP, and H-GCN). Additionally, we also test the Graph Attention model (GAT) and the hyperbolic variant H-GAT.

Training Setting: For the given methods, we perform a hyperparameter grid search on a validation set over initial learning rate, weight decay, dropout, number of layers, and activation functions. We measure the models' performance on the final test set over 10 random parameter initializations. The number of dimensions is the same for all methods, which is 128. We optimize all models using Adam, except Poincaré embeddings which are optimized using RiemannianAdam. The train, validation, and test splits are the same as the ones used in [1]. We evaluate link prediction by measuring area under the ROC curve on the test set and evaluate node classification by measuring F1 score except for Cora and Pubmed where we report accuracy as is standard in the literature.

[3]Our experiments are conducted on a RTX8000 GPU within a limit of 48 GB VRAM.

Table 1: AUC values for Link Prediction (LP) and F1 scores for Node Classification (NC) tasks with their corresponding standard deviations over 10 random parameter initializations. Best results are given in bold.

Dataset	CORA		PUBMED		DISEASE		AIRPORT	
Algorithms	LP	NC	LP	NC	LP	NC	LP	NC
E-Linear	83.62 ± 0.34	23.30 ± 0.13	85.98 ± 0.44	34.30 ± 0.57	57.74 ± 0.21	33.25 ± 0.19	92.79 ± 0.13	62.79 ± 0.54
H-Linear	84.50 ± 0.13	25.50 ± 0.37	88.21 ± 0.32	51.60 ± 0.54	61.81 ± 0.14	46.55 ± 0.36	93.49 ± 0.64	69.74 ± 0.24
MLP	84.52 ± 0.26	53.00 ± 0.74	82.60 ± 0.36	67.60 ± 0.77	72.13 ± 0.57	41.00 ± 0.35	67.86 ± 0.64	89.79 ± 0.75
H-MLP	91.28 ± 0.14	54.60 ± 0.47	94.11 ± 0.65	67.20 ± 0.43	75.02. ± 0.76	65.57 ± 0.53	92.21 ± 0.18	79.05 ± 0.24
GCN	91.32 ± 0.66	81.50 ± 0.94	85.98 ± 0.15	78.85 ± 0.49	64.72 ± 0.37	66.67 ± 0.54	90.58 ± 0.12	78.95 ± 0.17
H-GCN	**94.01 ± 0.38**	78.70 ± 0.54	**96.55± 0.16**	79.40± 0.18	90.54± 0.20	**73.24± 0.30**	**96.30± 0.74**	**90.43± 0.76**
GAT	93.32 ± 0.77	**81.90 ± 0.34**	94.54 ± 0.66	77.30 ± 0.56	70.12 ± 0.55	68.34 ± 0.48	90.65 ± 0.31	80.54 ± 0.38
H-GAT	90.01 ± 0.18	79.50 ± 0.51	93.55± 0.12	**79.20± 0.34**	**91.52± 0.25**	70.21± 0.32	94.13± 0.73	87.13± 0.46

Table 2: Time (T) given in seconds and Memory (M) given in Mebibytes (MiB) required for Link Prediction (LP) and Node Classification (NC) tasks for various algorithms.

Dataset	CORA				PUBMED				DISEASE				AIRPORT			
Task	LP		NC		LP		NC		LP		NC		LP		NC	
Algorithms	T	M	T	M	T	M	T	M	T	M	T	M	T	M	T	M
E-Linear	109	911	2	935	23	5499	1	2471	3	909	31	747	40	855	1	759
H-Linear	162	875	52	955	149	5643	25	2553	5	893	51	771	88	935	4	781
MLP	28	1007	1	929	238	4035	2	913	16	839	30	767	21	743	40	893
H-MLP	27	1149	4	1053	221	5101	5	1579	7	923	3	815	255	1095	3	823
GCN	20	1063	2	985	22	4253	6	1065	11	1041	46	825	8	969	19	803
H-GCN	34	1241	7	1125	162	5363	9	1759	71	1003	119	825	428	1059	21	903
GAT	35	1209	14	1103	1051	4303	44	4203	16	1231	29	839	16	1263	21	823
H-GAT	54	1441	24	1254	92	10313	34	5247	51	1713	200	1335	568	2101	187	1615

Results and Analysis: From the results, provided in Table 1, we note that we are able to reproduce the baseline results [1] (within a margin of error). Table 2, which presents the time and memory required by the algorithms, clearly shows that the requirements are in line with the original implementation, as well as, within the limits of a standard training/evaluation setup for the graph processing models.

5 DEMO PLAN

We will present our toolkit in the following way: (i) use a poster to describe the overall flow of a graph processing pipeline and explain the role of our system's modules in the different components, (ii) guide the audience through the entire process of building and evaluating a new model, including the dataset preprocessing, model construction, hyper-parameter tuning, and evaluation, (iii) describe the mathematical fundamentals of hyperbolic models as needed in the design of hyperbolic networks with their benefits and limitations, and (iv) provide future avenues of applicability in the area of graph processing and hyperbolic networks.

6 CONCLUSION

We presented GraphZoo, a versatile library which helps in systematically learning, using and designing graph processing pipelines. While there has been considerable amount of work done for each of the independent modules/models, this methodical way of combining them enables the framework to quickly deliver what can be of significant value to the researchers working with graphs.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grants IIS-1838730 and Amazon AWS credits.

REFERENCES

[1] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, Vol. 32. Curran Associates, Inc., Vancouver, Canada.

[2] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2021. Self-Supervised Hyperboloid Representations from Logical Queries over Knowledge Graphs. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 1373–1384. https://doi.org/10.1145/3442381.3449974

[3] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. 2022. ANTHEM: Attentive Hyperbolic Entity Model for Product Search. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (Virtual Event, AZ, USA) *(WSDM '22)*. Association for Computing Machinery, New York, NY, USA, 161–171. https://doi.org/10.1145/3488560.3498456

[4] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[5] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *Advances in neural information processing systems*. 5345–5355.

[6] Jiafeng Guo, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2019. MatchZoo: A Learning, Practicing, and Developing System for Neural Text Matching. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France). ACM, New York, NY, USA, 1297–1300.

[7] Raymond S. Koff. 1992. Infectious diseases of humans: Dynamics and control. By R.M. Anderson and R.M. May, 757 pp. Oxford: Oxford University Press, 1991. $95.00. *Hepatology* 15, 1 (1992), 169–169.

[8] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li,

and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[9] Ping Wang, Khushbu Agarwal, Colby Ham, Sutanay Choudhury, and Chandan K Reddy. 2021. Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. In *Proceedings of the Web Conference 2021*. 2946–2957.