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

VIRGINIA TECH

Introduction

⇨ Hyperbolic spaces have recently gained prominence for representation learning in graph processing tasks such as link prediction and node classification.

⇨ 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, there is a need in the community for a unified framework that eases the development and analysis of both Euclidean and hyperbolic graph networks.

⇨ We present a novel framework, GraphZoo, that makes learning, designing and applying graph processing pipelines/models systematic through abstraction over the redundant components.

⇨ Using GraphZoo, researchers can:

- ❒ 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.
- ❒ Quickly develop new models with the help of APIs to various graph-based hyperbolic/Euclidean layers and manifolds.
- ❒ Compare new models against state-of-the-art baselines on standard datasets with the help of pre-defined evaluation pipelines.
- ❒ Perform hyper-parameter tuning by parallelly running a rapid grid search over a set of configuration files.

GraphZoo Library

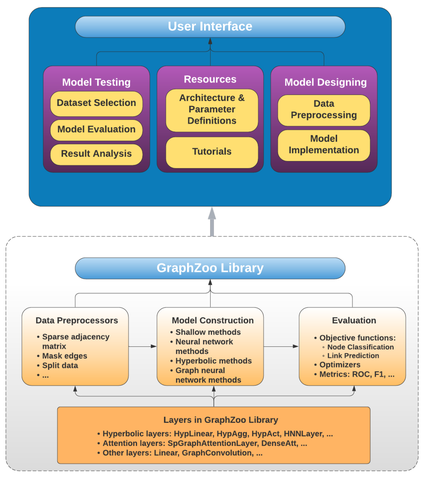

Fig. 1: An overview of the GraphZoo library.

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. In our library, as shown in Figure 2, these three modules are independent of each other.

⇨ 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.

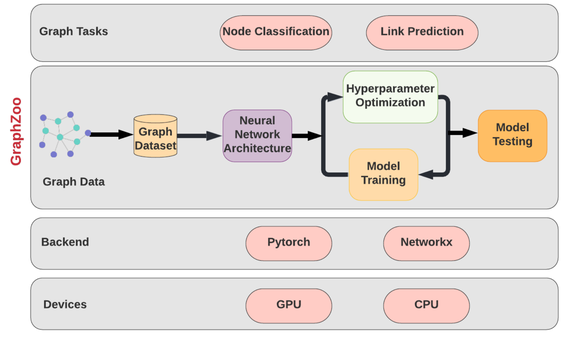

Fig. 2: An overview of the GraphZoo library.

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()
```

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

⇨ 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.

Figure 3 provides an example code snippet from the library to train H-GCN on CORA dataset for the node classification task.

Reproducibility Experiments

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

⇨ We 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.

⇨ From the results, provided in Table 2, we note that we are able to reproduce the baseline results (within a margin of error).

⇨ Table 3, which presents the time and memory required by the algorithms, clearly shows that the requirements are in line with the original implementation

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.

Acknowledgements

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

SCAN ME

Contact anoushkav@vt.edu for any questions.

Link to code: https://github.com/AnoushkaVyas/GraphZoo