Achieve the State of the Art

GDL 2022
Group 01
Project 03

Mürsel Sinan Ayaz, Stefano Carlo Lambertenghi, Sofía Talancón Fernández

Abstract
The task of drug-drug interaction (DDI) prediction is extremely useful to allow medical experts to evaluate side effects in drug administration during clinical trials. In this project we attempt to place our implementation of this task in the top-3 positions of the The Open Graph Benchmark project Hu et al. [1] for the DDI dataset. We decided to use two approaches; For our first attempt we have implemented a newly proposed method, SkipGNN Huang et al. [2], which has not been used by competing leaderboard participants. Our solution obtains testing accuracy of 86.82% which places it at the fourth place of the DDI dataset rankings. As we were required to obtain top 3 places we chose to take inspiration from one of the top 3 ranking methods. We believe that just reusing an already highly placed submission and positioning it at the same rank would not be very innovative and neither useful, therefore we strived to improve it to place it in a higher rank. We have repurposed the second ranking submission, 2-GraphSAGE+Edge Attributes(Lu and Yang [3]), and created 3 modified versions, one of which managed to surpass the first place submission by obtaining 94.03%. The code of this project is available in this Github repository.

1 Introduction

For this project we were tasked to rank within the top 3 positions in one of the datasets from the Open Graph Benchmark project(Hu et al. [1]). We decided to work with one of the datasets from the Link Property prediction subset, called **ogbl-ddi**. The data from the dataset is made up of drug-drug interactions (DDI), which is when medications interact with one or more different medications. This prediction task is a challenging problem being researched because it's critical to be able to identify the potential DDI's in clinical trials. Existing machine learning models already deal with the task of link prediction, which normally involves dealing with incomplete graphs and being able to correctly predict a connection between two nodes.

2 Related works

Lately, using techniques such as machine learning to predict potential drug-drug interactions has become popular because of the effectiveness these computational methods can offer. There have even been articles focused on evaluating the different methods that have already been tested on DDIs. (Qiu et al. [4], Shtar et al. [5], Han et al. [6]). A popular approach is to use methods such as node2vec(Grover and Leskovec [7]), or DeepWalk(Perozzi et al. [8]) that have a skip-gram model with random walks. There are also methods such as Variational Graph Autoencoders(VGAE), Wu and Cheng [9], that use neural embedding. Recently, there have also been methods that use not only direct connection embedding but skip connections to capture higher order similarity (Xu et al. [10], Kipf and Welling [11], Abu-El-Haija et al. [12]).

3 Link Property Prediction

Among the machine learning tasks that are commonly applied to graphs, we have the task of Link prediction which implies training a model to learn important relationships between nodes and being able to predict useful interactions between them. In other words, for the task of Link Property prediction we have examples of node pairs as input and during training the pairs are either marked as connected or not.

3.1 Dataset

The dataset **ogbl-ddi** is a network that represents the drug-drug interactions, where each node represents an FDA-approved drug and the edges represent interactions between them. It is an undirected, homogeneous, unweighted graph. The dataset is split in such a way that the drugs from the validation and training sets target different proteins than those in the test set, this way it is possible to evaluate whether the model can properly predict interactions.

- The graph is made up of 4,267 nodes and 1,334,889 edges.
- The evaluation metric consists of counting the ratio of positive edges that are ranked at K-place or above (Hits@K). Each true drug interaction is ranked among randomly-sampled negative drug interactions.

3.2 Leaderboard

The Open Graph Benchmark project(Hu et al. [1]) maintains a leaderboard for the dataset that was chosen. The table below shows the status of it as of May 12, 2022.

3.3 Methods implemented for experiments

For our experiments we had two main approaches: the first one being to implement a new method based on an article recently published, while the second was to find one of the existing methods in the leaderboard and try and improve it further.

Table 1. Leaderboard for ogbl-ddi

Method	Test Hits@20
1 - `PLNLP`	0.9088 ± 0.0313
2 - `GraphSAGE + Edge Attr`	0.8781 ± 0.0474
3 - `CFLP (w/ JKNet)`	0.8608 ± 0.0198
4 - `GraphSAGE+anchor distance`	0.8239 ± 0.0437
5 - `DEA + JKNet`	0.7672 ± 0.0265
6 - `GraphSAGE+Edge Proposal Set`	0.7495 ± 0.0317
7 - `LRGA+GCN(Node2Vec+Augment)`	0.7385 ± 0.0871

Implementation of a new method: SkipGNN
As a first approach, we found the method called SkipGNN, Huang et al. [2], used for predicting molecular interactions. This model takes as input the original network and, first of all, constructs a skip graph. Which is essentially a new graph based on the original graph but with neural messages being passed through second-order interactions. The following figure taken from the original article, illustrates what this skip similarity is and the difference from the direct interactions.

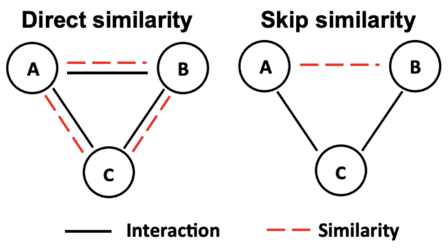

Figure 1. Direct versus skip similarity.

Once the skip graph has been generated we also construct the adjacency matrix to capture the skip similarity. Next, both the skip graph and the original graph are passed through two graph convolutions. Afterwards, we combine the embeddings with iterative fusion in the propagation step and they are then stored in the final layer.

Improving an existing model: 2-GraphSAGE+Edge Attr
The model we selected to improve is the current second place holder (Lu and Yang [3]) in the leaderboard for the dataset **ogbl-ddi**. The authors a modified version of GraphSAGE, in which they have change the aggregation stage to incorporate edge information, which is obtained from multiple anchor sets from random sampling.

In the original code, the SPD matrix calculates valuable edge attributes of randomly selected K-nodes (500) to every other node. It does this by iterating over every node, calculating the shortest distance between itself and every other node in the graph. Then, it appends this distance to the corresponding entry in the SPD matrix, however, if this distance is greater than 5 (max_spd=5), it appends 5 only, which was found to be another hyperparameter that could be optimized. We have tried to improve the overall performance by adding additional measurements to the SPD matrix, listed below.

Add attributes to the SPD Matrix The first improvement we have tried to implement was to add an additional attribute into the entries of the SPD matrix. We calculated the similarity rank of each of the K nodes in the SPD matrix and arbitrarily weighted the result (as it was too insignificant to make a major difference) and added that result into the calculation of that particular node in the SPD matrix.

Average distance instead of limiting the distance between nodes The second improvement we have tried was similar to the first one. We discarded of the max_spd=5 variable completely from the equation, and instead of calculating the similarity, we calculated the average of each of the K nodes in the SPD matrix and divided this average with the distance.

top-k degree node candidates for SPD Matrix Instead of randomly choosing K nodes to calculate the SPD matrix, another improvement we have tried to implement was to calculate the degree of each node in the graph, and instead calculate the SPD matrix of the nodes with top-K nodes with the highest degrees, and send this set to be the input of the SPD matrix calculation function instead.

3.4 Hyperparameters

Hyperparameter selection has been as closely as possible related to the original authors'.
In the 2-GraphSAGE+Edge Attributes variations, the only change in parameters has been "hidden-dimension", a decision commanded by resources limitations, in some variations, some hyperparameters were not used.
As the SkipGNN training logic is borrowed from the 2nd place positioning implementation, the hyperparameters relating to training and selections are the ones recommended by the authors.

Table 2. Hyperparameters settings

SkipGNN		2-GraphSAGE+Edge Attributes	
Hyperparameter	Value	Hyperparameter	Value
num layers	2	num layers	2
num samples	5	num samples	5
hidden channels	256	hidden channels	256
dropout	0.3	dropout	0.3
batch size	65536	batch size	65536
lr	0.003	lr	0.003
epochs	1000	epochs	1000
runs	8	runs	24(total)
-	-	spd size	500
hidden1	256	-	-
hidden2	256	-	-
hidden decode	16	-	-

3.5 Experimental setup

All of our models have been implemented to be used on Google Colab.
2-GraphSAGE+Edge Attributes's variations are implemented in a single notebook, where the original authors' code is repurposed with our additions.
SkipGNN has been implemented on another notebook, which is based on the same 2-GraphSAGE+Edge Att. training loop/dataloader with the new custom model.
The notebooks implementing our experiments can be found

here: Github repo. The complete list of hyperparameters is visible in table Table 2

3.6 Computational requirements

2-GraphSAGE+Edge Attributes's variations have been ran on a Tesla V100 with 16160MiB of GPURam as it was the best GPU available on Google Colab Pro+. We have found that 16160MiB were not enough to execute the original code, as the Embedding Network requires the full dataset as input and it would exceed GPU memory available to us.
We have reduced hidden dimensions to be able to perform training. GPU time is around 2 hours for each training loop, for a total of 16 hours for the 8 runs for each variation, in total 48 GPU hours were needed to present the discussed results.

SkipGNN can be executed using the resources allocated using a free license of Google Colab but, as a premium license was available to us, it was used with the purpose of reducing training times.
GPU time is around 1 hour for each training loop, for a total of 8 hours for each variation.

4 Results

All approaches have been tested over 1000 epochs and multiple training and testing runs have been performed to take into account any stochasticity caused by dataset augmentation and validation/testing input selection.
Hereafter are the found results of the different approaches analyzed. All of our results are evaluated using Hits@20 as the challenge focuses on this metric.

Results of SkipGNN
Our first approach, which novel to the Open Graph Benchmark project leader boards submissions would be placed fourth as visible in Table 3, where our results are compared with the best performing result of competing models. To analyze the results, 8 independent runs of the training/validation/testing process will be shown.

- **Training loss over epochs analysis.**
  Figure 2 displays the loss evolution over the 8 performed runs of our model training. Consistency in loss evolution in the different runs can be observed. The best performing run reaches a loss of **0.1360**

- **Validation accuracy over epochs analysis.**
  Figure 3 displays the validation accuracy evolution over the 8 performed runs of our model training.
  Here, even if loss seems to be consistent between runs, validation accuracy is more varied.
  To better discuss results, the best performing run is shown in detail in Figure 4. Observing the model behaviour, we can observe that accuracy increase is not strictly correlated to loss decrease, and as such we will consider the best achieved accuracy as our metric of quality. The best performing run reaches an overall best validation accuracy of **75.79%**

- **Testing accuracy over epochs analysis.**
  Figure 5 displays the testing accuracy evolution over the 8 performed runs of our model training.
  This is the metric on which we compete on the leaderboard to achieve top 3 results.
  Even more than in validation accuracy, different runs vary significantly in testing accuracy achieved.
  As per validation, epochs evolution do not strictly correlate to accuracy increase, and as such we will consider the best achieved result over the best run.
  The best run is presented in Figure 6
  The best performing run reaches an overall best testing accuracy of **86.82%**

Table 3. Leaderboard for ogbl-ddi after our experiments

Method	Test Hits@20
1 - `PLNLP`	$0.9088 \pm 0.0313 = 0.9401$
2 - `GraphSAGE + Edge Attr`	$0.8781 \pm 0.0474 = 0.9255$
3 - `CFLP (w/ JKNet)`	$0.8608 \pm 0.0198 = 0.8806$
4 - **`SkipGNN`**	**0.8682**
5 - `GraphSAGE+anchor distance`	$0.8239 \pm 0.0437 = 0.8676$
6 - `DEA + JKNet`	$0.7672 \pm 0.0265 = 0.7937$
7 - `GraphSAGE+Edge Proposal Set`	$0.7495 \pm 0.0317 = 0.7812$
8 - `LRGA+GCN(Node2Vec+Augment)`	$0.7385 \pm 0.0871 = 0.8256$

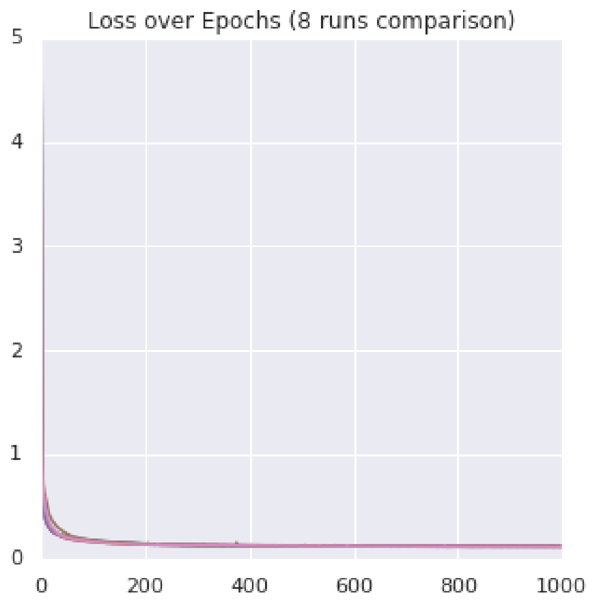

Figure 2. Training loss over 1000 epochs, for 8 runs.

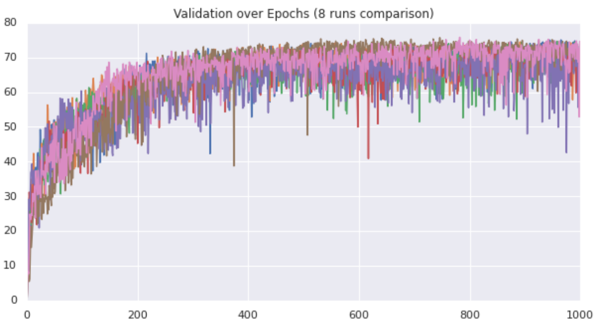

Figure 3. Validation accuracy over 1000 epochs, for 8 runs.

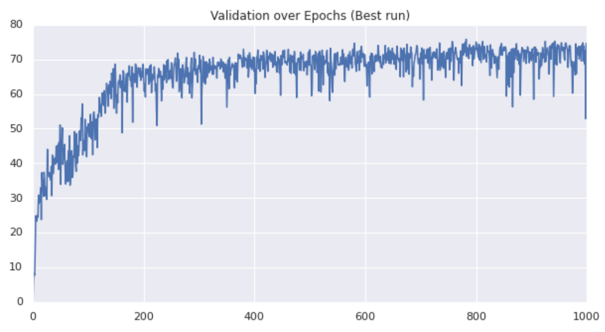

Figure 4. Validation accuracy over 1000 epochs, Best run.

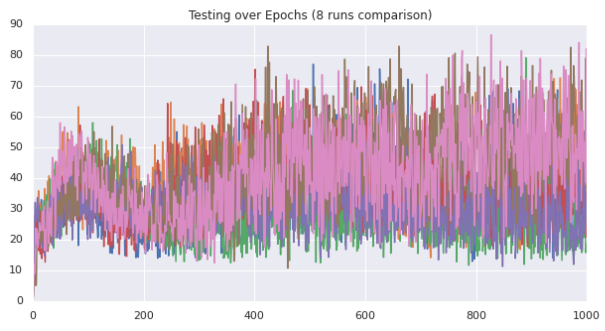

Figure 5. Testing accuracy over 1000 epochs, for 8 runs.

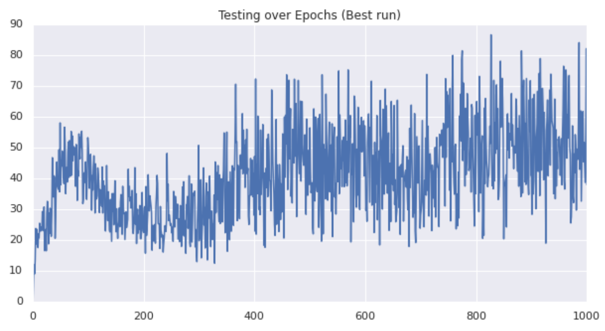

Figure 6. Testing accuracy over 1000 epochs, Best run.

Results of 2-GraphSAGE+Edge Attr. with Variations
To evaluate results of the variations over the original 2-GraphSAGE+Edge Attributes method, due to GPU Memory limitations, as mentioned in subsection **3.6 Computational requirements**, we had to reduce Hidden dimension from 512 to 256, as such all results here presented will be performed using the same reduced hidden dimension. In the same manner as with SkipGNN, we have performed 8 runs for each variation of the model. Instead of showing each run will take into account the overall best run for each of the variations.

- **Training loss over epochs analysis.**
  To evaluate the modifications in terms of validation performance, Figure 7 displays the training loss over epochs of the different model variations.
- **Validation accuracy over epochs analysis.**
  To evaluate the modifications in terms of validation performance, Figure 8 displays the validation accuracy over epochs of the model variations. The overall best results in validation accuracy can be compared in

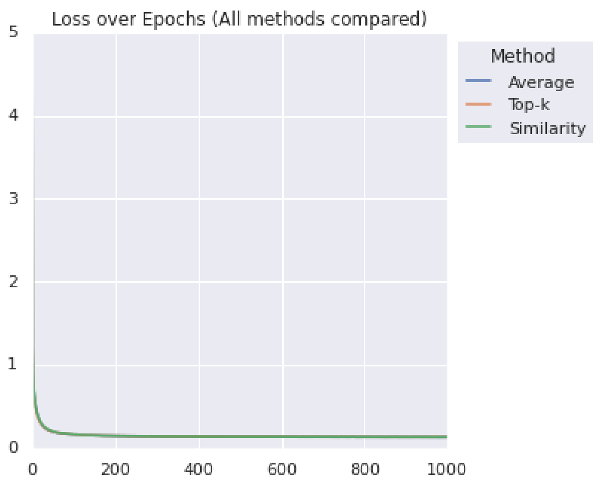

Figure 7. Training loss over 1000 epochs, all variations.

Figure 9. All of the validations accuracies are worse than the original model they are based on.

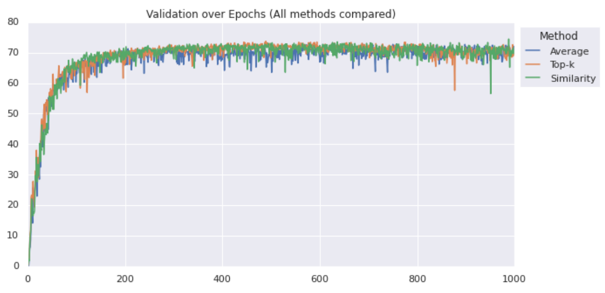

Figure 8. Validation accuracy over 1000 epochs, all variations.

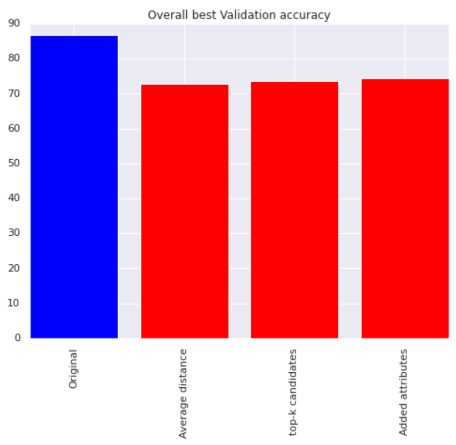

Figure 9. Overall best validation accuracy of model variations

- **Testing accuracy over epochs analysis.**
  To evaluate the modifications in terms of testing performance, Figure 10 displays the testing accuracy over epochs of the model variations. The overall best results in Testing accuracy can be compared in Figure 11.
  From the achieved results, we have observed that not all of our variations have given a beneficial impact on the expected performace, in particular (GraphSAGE+E.A. + Average distance) and (GraphSAGE+E.A.+ Added attributes to the SPD Matrix) has

achieved worse performance that the model it is based on.
Meanwhile the use of top-k candidates has improved the accuracy by around 2% placing it in the first place of the ogbl-ddi leaderboard, with an ever so slightly higher performance than the top-performing PLNLP method.
The variation placements are visible in Table 4 where the bold entries are our newly found results. As in Table 3, our results are compared with the best performing result of competing models.

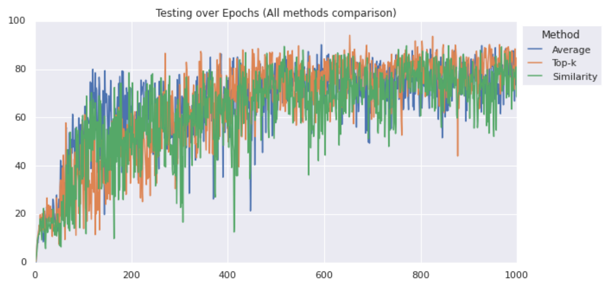

Figure 10. Testing accuracy over 1000 epochs, all variations.

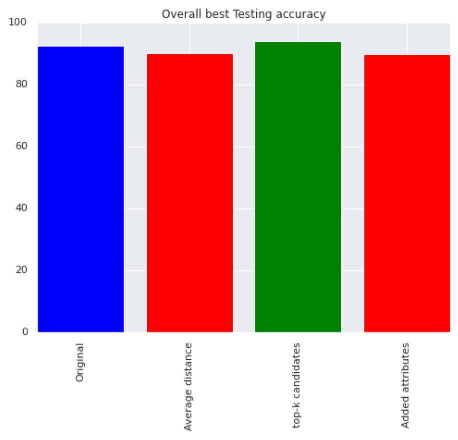

Figure 11. Overall best testing accuracy of model variations

Table 4. Leaderboard for ogbl-ddi after our experiments

Method	Test Hits@20
– **`GraphSAGE + E.A. + Top-k selection`**	**0.9403**
1 - `PLNLP`	$0.9088 \pm 0.0313 = 0.9401$
2 - `GraphSAGE + Edge Attr`	$0.8781 \pm 0.0474 = 0.9255$
– **`GraphSAGE + E.A. + Average distance`**	**0.9016**
– **`GraphSAGE + E.A. + Added attributes`**	**0.8974**
3 - `CFLP (w/ JKNet)`	$0.8608 \pm 0.0198 = 0.8806$
– **`SkipGNN`**	**0.8682**
4 - `GraphSAGE+anchor distance`	$0.8239 \pm 0.0437 = 0.8676$
5 - `DEA + JKNet`	$0.7672 \pm 0.0265 = 0.7937$

5 Discussion and conclusion

Link prediction for drug-drug interactions is a very important problem that has been shown to be solved by using several methods. In our experiments we showed different models that we tested with the dataset **ogbl-ddi** from OGB, and in doing so we managed to obtain a few models that give a performance in the top-3 and one other model in the top-10. The two approaches we had were to implement a published model that hadn't been applied to this dataset and to modify a model that was already in the leaderboard of the task. The SkipGNN model doesn't manage to rank within the top-3 but we see that it could be further improved by using additional modifications such as distance encoding and higher hidden layer sizes if higher resources are available. Higher model sizes could be beneficial also for our GraphSAGE variations, as we couldn't recreate the original model which is implemented to run on 32Gb GPU memory. Moreover due to time limitations, only 4 variations of attributes augmentation have been analyzed, many other graph proprieties could be explored.

References

[1] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[2] Kexin Huang, Cao Xiao, Lucas Glass, Marinka Zitnik, and Jimeng Sun. Skipgnn: Predicting molecular interactions with skip-graph networks, 2020. URL `https://arxiv.org/abs/2004.14949`.

[3] Shitao Lu and Jing Yang. Link prediction with structural information. *Open Graph Benchmark: Datasets for Machine Learning on Graphs*, 2021.

[4] Yang Qiu, Yang Zhang, Yifan Deng, Shichao Liu, and Wen Zhang. A comprehensive review of computational methods for drug-drug interaction detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2021. doi: 10.1109/TCBB.2021.3081268.

[5] Guy Shtar, Lior Rokach, and Bracha Shapira. Detecting drug-drug interactions using artificial neural networks and classic graph similarity measures. *CoRR*, abs/1903.04571, 2019. URL `http://arxiv.org/abs/1903.04571`.

[6] Ke Han, Peigang Cao, Yu Wang, Fang Xie, Jiaqi Ma, Mengyao Yu, Jianchun Wang, Yaoqun Xu, Yu Zhang, and Jie Wan. A review of approaches for predicting drug–drug interactions based on machine learning. *Frontiers in Pharmacology*, 12, 01 2022. doi: 10.3389/fphar.2021.814858.

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL `http://arxiv.org/abs/1607.00653`.

[8] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL `http://arxiv.org/abs/1403.6652`.

[9] Xinxing Wu and Qiang Cheng. Deepened graph autoencoders help stabilize and enhance link prediction. *CoRR*, abs/2103.11414, 2021. URL `https://arxiv.org/abs/2103.11414`.

[10] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks, 2018. URL `https://arxiv.org/abs/1806.03536`.

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

[12] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, 2019. URL `https://arxiv.org/abs/1905.00067`.