Open Graph Benchmark
Tutorial 1

Antonio Longa

https://antoniolonga.github.io/

Advance PyG Tutorials

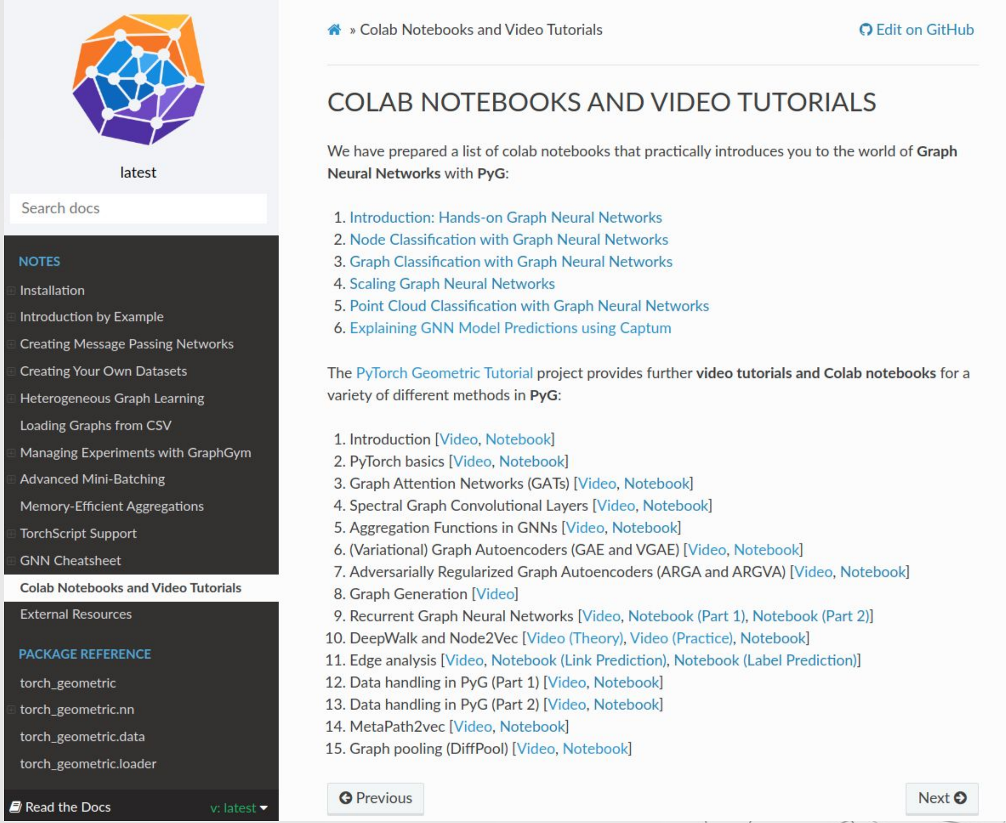
- Welcome :)

Advance PyG Tutorials

- Welcome :)
- From 3 to 26 guys, join us here.

Advance PyG Tutorials

- Welcome :)
- From 3 to 26 guys, join us here.
- We are on the official library of PyG



latest

Search docs

NOTES

- Installation
- Introduction by Example
- Creating Message Passing Networks
- Creating Your Own Datasets
- Heterogeneous Graph Learning
- Loading Graphs from CSV
- Managing Experiments with GraphGym
- Advanced Mini-Batching
- Memory-Efficient Aggregations
- TorchScript Support
- GNN Cheatsheet
- Colab Notebooks and Video Tutorials
- External Resources

PACKAGE REFERENCE

- torch_geometric
- torch_geometric.nn
- torch_geometric.data
- torch_geometric.loader

Read the Docs v: latest

» Colab Notebooks and Video Tutorials Edit on GitHub

COLAB NOTEBOOKS AND VIDEO TUTORIALS

We have prepared a list of colab notebooks that practically introduces you to the world of **Graph Neural Networks** with **PyG**:

1. Introduction: Hands-on Graph Neural Networks
2. Node Classification with Graph Neural Networks
3. Graph Classification with Graph Neural Networks
4. Scaling Graph Neural Networks
5. Point Cloud Classification with Graph Neural Networks
6. Explaining GNN Model Predictions using Captum

The PyTorch Geometric Tutorial project provides further **video tutorials and Colab notebooks** for a variety of different methods in **PyG**:

1. Introduction [Video, Notebook]
2. PyTorch basics [Video, Notebook]
3. Graph Attention Networks (GATs) [Video, Notebook]
4. Spectral Graph Convolutional Layers [Video, Notebook]
5. Aggregation Functions in GNNs [Video, Notebook]
6. (Variational) Graph Autoencoders (GAE and VGAE) [Video, Notebook]
7. Adversarially Regularized Graph Autoencoders (ARGA and ARGVA) [Video, Notebook]
8. Graph Generation [Video]
9. Recurrent Graph Neural Networks [Video, Notebook (Part 1), Notebook (Part 2)]
10. DeepWalk and Node2Vec [Video (Theory), Video (Practice), Notebook]
11. Edge analysis [Video, Notebook (Link Prediction), Notebook (Label Prediction)]
12. Data handling in PyG (Part 1) [Video, Notebook]
13. Data handling in PyG (Part 2) [Video, Notebook]
14. MetaPath2vec [Video, Notebook]
15. Graph pooling (DiffPool) [Video, Notebook]

Previous Next

Advance PyG Tutorials

- Welcome :)
- From 3 to 26 guys, join us here.
- We are on the official library of PyG
- AIMS:
  - Learn together
  - Share GNN news
  - Discuss

Advance PyG Tutorials

- Welcome :)
- From 3 to 26 guys, join us here.
- We are on the official library of PyG
- AIMS:
  - Learn together
  - Share GNN news
  - Discuss
  - Spread GNN to the world :)

Advance PyG Tutorials

TIMETABLE:

- Two times per month

Advance PyG Tutorials

TIMETABLE:

- Two times per month
- On Friday at 3.00pm Italian Time. Do not worry, you will receive a google calendar invite :)

Advance PyG Tutorials

TIMETABLE:

- Two times per month
- On Friday at 3.00pm Italian Time. Do not worry, you will receive a google calendar invite :)
- **Where:** Google meet

Advance PyG Tutorials

TIMETABLE:

- Two times per month
- On Friday at 3.00pm Italian Time. Do not worry, you will receive a google calendar invite :)
- **Where:** Google meet

DATE	TOPIC	AUTHOR
Fri, 15 Oct	Open Graph Benchmark	Antonio Longa
Fri, 29 Oct	Graph Gym	Gabriele Santin
Fri, 12 Nov	Graph Gym practice	To decide
Fri, 26 Nov	Heterogeneous graph learning	To decide
Fri, 10 Dec	Advanced mini-batching	To decide
Fri, 17 Dec	Memory-Efficient aggregations	To decide

Advance PyG Tutorials

TIMETABLE:

- Two times per month
- On Friday at 3.00pm Italian Time. Do not worry, you will receive a google calendar invite :)
- **Where:** Google meet

DATE	TOPIC	AUTHOR
Fri, 15 Oct	Open Graph Benchmark	Antonio Longa
Fri, 29 Oct	Graph Gym	Gabriele Santin
Fri, 12 Nov	Graph Gym practice	To decide
Fri, 26 Nov	Heterogeneous graph learning	To decide
Fri, 10 Dec	Advanced mini-batching	To decide
Fri, 17 Dec	Memory-Efficient aggregations	To decide

Do you want to present?

- Text us and you will be added

Advance PyG Tutorials

TIMETABLE:

- Two times per month
- On Friday at 3.00pm Italian Time. Do not worry, you will receive a google calendar invite :)
- **Where:** Google meet

Do you want to present?

- Text us and you will be added

DATE	TOPIC	AUTHOR
Fri, 15 Oct	Open Graph Benchmark	Antonio Longa
Fri, 29 Oct	Graph Gym	Gabriele Santin
Fri, 12 Nov	Graph Gym practice	To decide
Fri, 26 Nov	Heterogeneous graph learning	To decide
Fri, 10 Dec	Advanced mini-batching	To decide
Fri, 17 Dec	Memory-Efficient aggregations	To decide

TABLE OF CONTENTS

01 Open Graph Benchmark

The Open Graph Benchmark (OGB)

- Collection of realistic, large-scale, and diverse benchmark datasets for machine learning on graphs.

- Automatically downloaded, processed, and split using the OGB Data Loader.

- **Multiple task categories**: predicting the properties of nodes, links, and graphs.
- **Diverse scale:** Small-scale graph datasets, medium- and large-scale graphs.
- **Rich domains:** Graph datasets come from diverse domains and include biological networks, molecular graphs, academic networks, and knowledge graphs.

01 Open Graph Benchmark

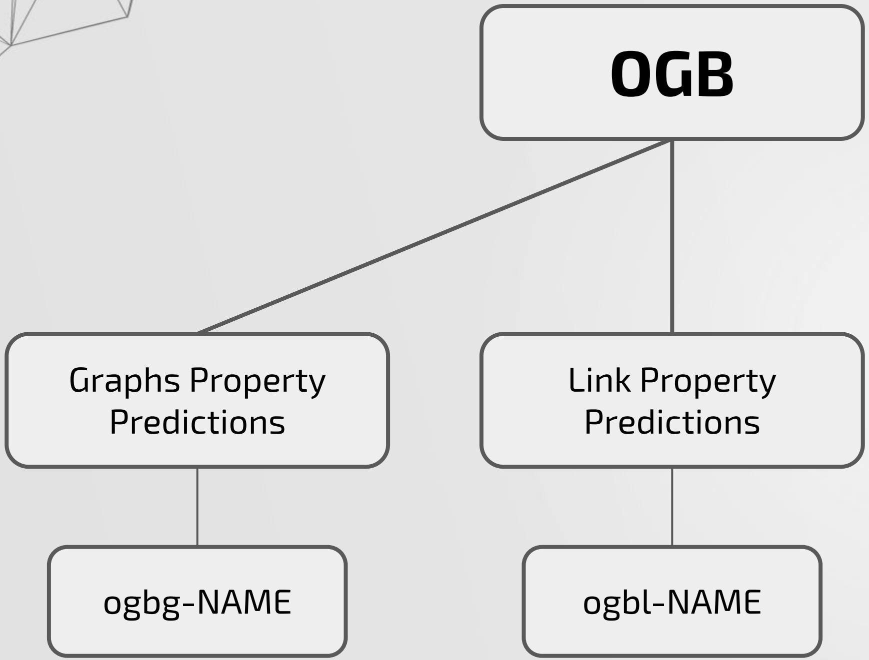
OGB

01 Open Graph Benchmark

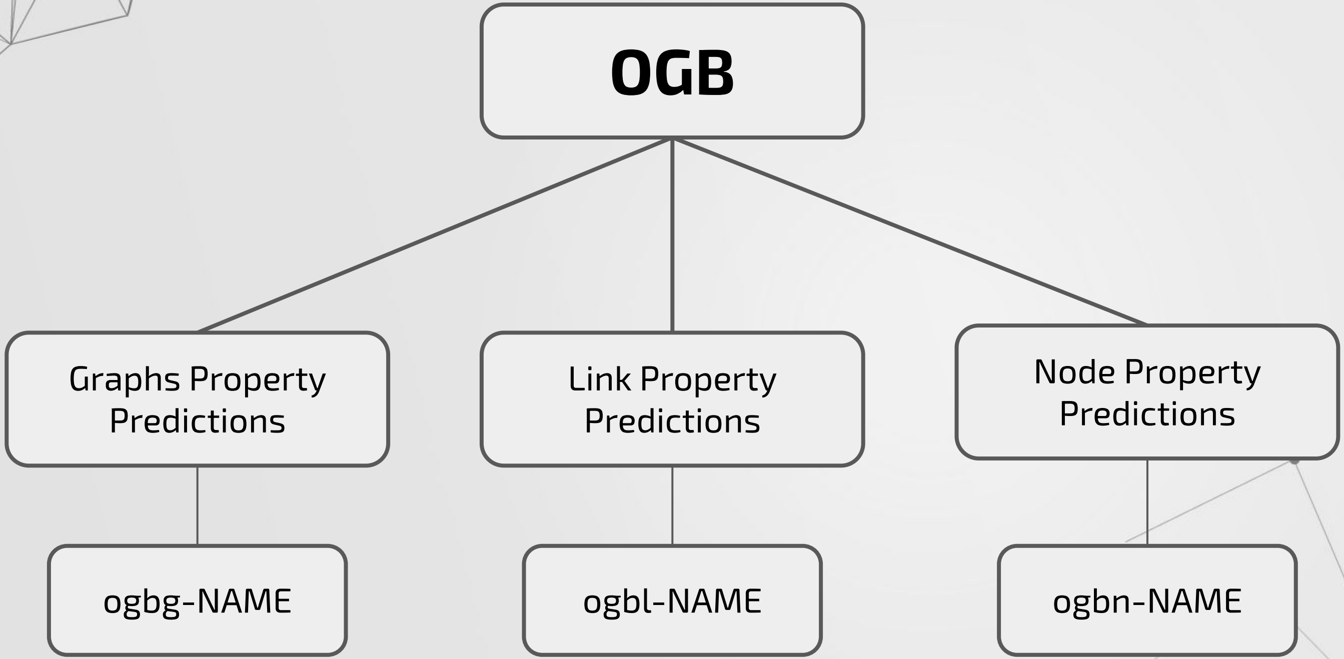
OGB

- Graphs Property Predictions
  - ogbg-NAME

01 Open Graph Benchmark



01 Open Graph Benchmark



01 Open Graph Benchmark

Node Property Prediction

The task is to predict properties of single nodes.

Summary

- Datasets

Scale	Name	Package	#Nodes	#Edges*	#Tasks	Split Type	Task Type	Metric
Medium	ogbn-products	>=1.1.1	2,449,029	61,859,140	1	Sales rank	Multi-class classification	Accuracy
Medium	ogbn-proteins	>=1.1.1	132,534	39,561,252	112	Species	Binary classification	ROC-AUC
Small	ogbn-arxiv	>=1.1.1	169,343	1,166,243	1	Time	Multi-class classification	Accuracy
Large	ogbn-papers100M	>=1.2.0	111,059,956	1,615,685,872	1	Time	Multi-class classification	Accuracy
Medium	ogbn-mag	>=1.2.1	1,939,743	21,111,007	1	Time	Multi-class classification	Accuracy

01 Open Graph Benchmark

Node Property Prediction

The task is to predict properties of single nodes.

Summary

- Datasets

Scale	Name	Package	#Nodes	#Edges*	#Tasks	Split Type	Task Type	Metric
Medium	ogbn-products	>=1.1.1	2,449,029	61,859,140	1	Sales rank	Multi-class classification	Accuracy
Medium	ogbn-proteins	>=1.1.1	132,534	39,561,252	112	Species	Binary classification	ROC-AUC
Small	ogbn-arxiv	>=1.1.1	169,343	1,166,243	1	Time	Multi-class classification	Accuracy
Large	ogbn-papers100M	>=1.2.0	111,059,956	1,615,685,872	1	Time	Multi-class classification	Accuracy
Medium	ogbn-mag	>=1.2.1	1,939,743	21,111,007	1	Time	Multi-class classification	Accuracy

01 Open Graph Benchmark

Dataset `ogbn-products` (Leaderboard):

Graph: The `ogbn-products` dataset is an undirected and unweighted graph, representing an Amazon product co-purchasing network [1]. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together. We follow [2] to process node features and target categories. Specifically, node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100.

Prediction task: The task is to predict the category of a product in a multi-class classification setup, where the 47 top-level categories are used for target labels.

Dataset splitting: We consider a more challenging and realistic dataset splitting that differs from the one used in [2] Instead of randomly assigning 90% of the nodes for training and 10% of the nodes for testing (without use of a validation set), we use the *sales ranking* (popularity) to split nodes into training/validation/test sets. Specifically, we sort the products according to their sales ranking and use the top 8% for training, next top 2% for validation, and the rest for testing. This is a more challenging splitting procedure that closely matches the real-world application where labels are first assigned to important nodes in the network and ML models are subsequently used to make predictions on less important ones.

Note: A very small number of self-connecting edges are repeated (see here); you may remove them if necessary.

References

[1] http://manikvarma.org/downloads/XC/XMLRepository.html

[2] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 257–266, 2019.

License: Amazon license

01 Open Graph Benchmark

Dataset `ogbn-products` (Leaderboard):

Graph: The `ogbn-products` dataset is an undirected and unweighted graph, representing an Amazon product co-purchasing network [1]. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together. We follow [2] to process node features and target categories. Specifically, node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100.

Prediction task: The task is to predict the category of a product in a multi-class classification setup, where the 47 top-level categories are used for target labels.

Dataset splitting: We consider a more challenging and realistic dataset splitting that differs from the one used in [2] Instead of randomly assigning 90% of the nodes for training and 10% of the nodes for testing (without use of a validation set), we use the *sales ranking* (popularity) to split nodes into training/validation/test sets. Specifically, we sort the products according to their sales ranking and use the top 8% for training, next top 2% for validation, and the rest for testing. This is a more challenging splitting procedure that closely matches the real-world application where labels are first assigned to important nodes in the network and ML models are subsequently used to make predictions on less important ones.

Note: A very small number of self-connecting edges are repeated (see here); you may remove them if necessary.

References

[1] http://manikvarma.org/downloads/XC/XMLRepository.html
[2] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 257–266, 2019.

01 Open Graph Benchmark

Dataset `ogbn-products` (Leaderboard):

Graph: The `ogbn-products` dataset is an undirected and unweighted graph, representing an Amazon product co-purchasing network [1]. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together. We follow [2] to process node features and target categories. Specifically, node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100.

Prediction task: The task is to predict the category of a product in a multi-class classification setup, where the 47 top-level categories are used for target labels.

Dataset splitting: We consider a more challenging and realistic dataset splitting that differs from the one used in [2] Instead of randomly assigning 90% of the nodes for training and 10% of the nodes for testing (without use of a validation set), we use the *sales ranking* (popularity) to split nodes into training/validation/test sets. Specifically, we sort the products according to their sales ranking and use the top 8% for training, next top 2% for validation, and the rest for testing. This is a more challenging splitting procedure that closely matches the real-world application where labels are first assigned to important nodes in the network and ML models are subsequently used to make predictions on less important ones.

Note: A very small number of self-connecting edges are repeated (see here); you may remove them if necessary.

References

[1] http://manikvarma.org/downloads/XC/XMLRepository.html

[2] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), pp. 257–266, 2019.

License: Amazon license

01 Open Graph Benchmark

Leaderboard for ogbn-products

The classification accuracy on the test and validation sets. The higher, the better.

Package: >=1.1.1

Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	**SAGN+SLE (4 stages)+C&S**	0.8485 ± 0.0010	0.9302 ± 0.0003	Chuxiong Sun (CTRI)	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Sep 21, 2021
2	**SAGN+SLE (4 stages)**	0.8468 ± 0.0012	0.9309 ± 0.0007	Chuxiong Sun (CTRI)	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Sep 21, 2021
3	**GAMLP+RLU**	0.8459 ± 0.0010	0.9324 ± 0.0005	Wentao Zhang (PKU Tencent Joint Lab)	Paper, Code	3,335,831	Tesla V100 (32GB)	Aug 19, 2021
4	Spec-MLP-Wide + C&S	0.8451 ± 0.0006	0.9132 ± 0.0010	Huixuan Chi (AML@ByteDance)	Paper, Code	406,063	Tesla V100 (32GB)	Jul 27, 2021
5	**SAGN+SLE**	0.8428 ± 0.0014	0.9287 ± 0.0003	Chuxiong Sun	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Apr 19, 2021
6	**MLP + C&S**	0.8418 ± 0.0007	0.9147 ± 0.0009	Horace He (Cornell)	Paper, Code	96,247	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020
7	**GAMLP**	0.8354 ± 0.0009	0.9312 ± 0.0003	Wentao Zhang (PKU Tencent Joint Lab)	Paper, Code	3,335,831	Tesla V100 (32GB)	Aug 22, 2021

02 Why OGB

- Easy access to datasets

02 Why OGB

- Easy access to datasets
- Load your own dataset

02 Why OGB

- Easy access to datasets
- Load your own dataset
- State-of-the-art performance of GNN performance

02 Why OGB

- Easy access to datasets
- Load your own dataset
- State-of-the-art performance of GNN performance
- Leaderboards

02 Why OGB

Leaderboards

Public leaderboards allow researchers to keep track of state-of-the-art methods and encourage reproducible research.

Important: Please make sure your experimental protocol follows the rules here.

How Leaderboards Work?

Once you have developed your model and got results, you can submit your test results to our leaderboards. For each dataset, we require you to submit the following information.

- **OGB version**: The OGB version used to conduct the experiments. **Must satisfy the version requirement for each dataset.**
- **Method**: The name of the method.
- **Dataset**: The name of an OGB dataset that you use to evaluate the method.
- **Test performance**: Raw test performance output by OGB model evaluators, where **average (`torch.mean`) and unbiased standard deviation (`torch.std`) must be taken over 10 different random seeds.** You can either not fix random seeds at all, or use the random seeds from 0 to 9. We highly discourage you to tune the random seeds. For the large `ogbn-papers100M`, you only need to use 3 random seeds to report the performance.
- **Validation performance**: Validation performance of the model that is used to report the test performance above.
- **Contact**: A person's name and email address to contact about the method and code.
- **Code**: The Github repository or directory containining all code to reproduce the result. **A placeholder repository is not allowed.**
  - We recommend using Pytorch.
  - The authors are responsible for addressing any inquiry about their code.
  - **Please add `README` and provide enough instruction (i.e., exact command) to reproduce the submitted result.** A good example is this.

03 OGB and PyG

- Nothing to say :)

03 OGB and PyG

- Nothing to say :)
- The downloaded object is ready for PyG

04 Practice

Jupyter notebook

Conclusion

- OGB is an amazing project!

Conclusion

- OGB is an amazing project!
    - Easy model performance comparison

Conclusion

- OGB is an amazing project!
  - Easy model performance comparison
  - Fast way to download dataset in PyG format

Conclusion

- OGB is an amazing project!
    - Easy model performance comparison
    - Fast way to download dataset in PyG format
    - Upload your own dataset

Conclusion

- OGB is an amazing project!
  - Easy model performance comparison
  - Fast way to download dataset in PyG format
  - Upload your own dataset
  - Upload your own model (like in Kaggle.com)

THANKS

Does anyone have any questions?

longaantonio@gmail.com
https://antoniolonga.github.io/