CS 224W Project (Fall 2025)

Table of Contents

0. Overview
1. Real-world applications of GNNs
    1.2 Examples of good projects
    1.3 Project Proposal [13.33%]
    1.4 Project Milestone [6.67%]
    1.5 Project Report [80%]
2. Tutorial on PyG functionality
    2.1 Example tutorials and good projects from previous years
    2.2 Project Proposal [13.33%]
    2.3 Project Milestone [6.67%]
    2.4 Project Report [80%]
3. Implementation of cutting-edge research
    3.1 Example research ideas
    3.2 Project Proposal [13.33%]
    3.3 Project Milestone [6.67%]
    3.4 Project Report [80%]
4. Project logistics
    4.1 What to submit
    4.2 Project Feedback and Mentor Assignments
    4.3 Writing blog posts on Medium
    4.4 Instructions on Google Colab
    4.5 Compute resources
5. Extra credit for approved PRs
6. Tips on writing blog posts

0. Overview

The goal of this year's CS224W course project is to help you create long-lasting resources for both your technical profiles and the graph machine learning community at large. To this end, we are inviting three types of projects:

1) **Real-world applications of GNNs [blog post + Google Colab]**: in this project, you will identify a specific use case (e.g., predicting drug interactions, fraud detection) and demonstrate how GNNs + PyG can be used to solve this problem.

2) **Tutorial on PyG functionality [blog post + Google Colab]**: in this project, you will develop a tutorial that explains how to use existing PyG features and functionality, such as methods for GNN explainability or graph sampling.
3) **Implementation of cutting-edge research [blog post + PR]**: in this project, you will identify a recent research paper in graph ML that develops interesting methods and implement the work as a pull request (PR) to the contrib package in PyG.

Graph ML is one of the hottest emerging topics in machine learning, with wide applicability across domains and industries. We hope your project will first serve as an opportunity for you to dive deep into graph ML algorithms and get first-hand experience implementing them. Furthermore, the outputs of your projects will become great resources for the broader graph ML community, particularly for newcomers who want to learn about where GNNs can be applied and how to use them.

Through your projects, you will gain experience using PyG (PyTorch Geometric), the most popular graph deep learning framework built on top of PyTorch. PyG is suitable for quickly implementing GNN models, with abundant graph models already implemented and datasets ready to load. Furthermore, GraphGym, available as part of PyG, allows a training/evaluation pipeline to be built in a few lines of code.

Your projects are also an opportunity to demonstrate your knowledge of graph ML to a wide audience. We will publish all blog posts on our course's Medium page and encourage you to share your hard work on your websites / social media. We will also publish the best blog posts at PyG.org and hope to integrate your high-quality PRs into the main PyG library!

We recommend project groups of 3 students, but groups of 2 are also allowed. Solo projects are discouraged; please see Ed. **Project proposals are due on Tuesday, October 21, 2025; the project milestone is due Thursday, November ~~6~~11, 2025; and final project reports are due on December 11, 2025.** Please see Project Logistics for details.

1. Real-world applications of GNNs

Output: blog post, Google Colab

In this project, you will identify a specific use case and demonstrate how GNNs and PyG can be used to solve this problem. We welcome use cases across domains (e.g., biological, social, financial), but we want to see use cases that are motivated by real-world problems, e.g., an application that a company would actually use. Given a use case, we expect you to identify an appropriate *public* dataset, formulate the use case as a clear graph ML problem, and demonstrate how GNNs can be used to solve this problem.

Note that it is not enough to simply *apply* an existing GNN model to a new dataset. We expect to see some form of **novelty** (e.g. developing a new GNN method for your specific problem,

improving on existing methods in a non-trivial way) or **comprehensive analyses** (e.g. ablation studies, comparison between multiple model architectures) for the project.

1.1 Example graphs & datasets

Below, we list several real-world application domains, specifying how graphs and tasks are defined in these domains. We also provide representative graph ML models and public datasets that you may use, such as from Open Graph Benchmark (OGB)* and the datasets available in PyG. These are just examples; you are free to explore other application domains, models, and public datasets yourself.

*Note that some OGB datasets are quite large and require a decent amount of computational resources to handle. We recommend using small-scale OGB datasets that are more tractable.

Recommender systems

Graphs:

- Nodes: Users, items
- Edges: User-item interactions

Tasks:

- Predicting the edge ratings. Metric: RMSE
- Predicting edge existence. Metric: Recall@K

Example model(s): LightGCN

Public datasets: MovieLens, Recsys repository

Product co-purchasing graph

Graphs:

- Nodes: Products
- Edges: Product co-purchasing relations

Tasks:

- Predicting product categories. Metric: Classification accuracy

Example model(s): See the leaderboard

Datasets: ogbn-products

Note: This is a medium-scale OGB dataset that requires more computational resources than the other datasets.

Fraud detection in transaction graphs

Graphs:

- Nodes: Financial users (customers, banks)
- Edges: Transaction (money and amount sent)

Tasks:

- Edge classification - predict which edges are fraudulent. Metric: Hits@50

Example model(s): See https://github.com/safe-graph/graph-fraud-detection-papers

Datasets: Bitcoin Fraud Dataset (only use labeled data!)

Friend recommendation

Graphs: social network
- Nodes: users
- Edges: potentially heterogeneous -- friend, follow, reply to, message, like, etc.

Tasks:
- Recommending/ranking new friends for users. Metrics: Hits@K, NDCG@K, MRR

Example model(s): GraFRank (paper, GitHub)
Datasets: Facebook, Google+, Twitter

Paper citation graphs
Graphs:
- Nodes: Papers
- Edges: Paper citations

Tasks:
- Predicting subject areas of papers. Metric: Classification accuracy

Example model(s): See the leaderboard
Datasets: ogbn-arxiv

Author collaboration networks
Graphs:
- Nodes: Authors
- Edges: Author Collaboration

Tasks:
- Predicting future author collaboration. Metric: Hits@50

Example model(s): See the leaderboard
Datasets: ogbl-collab

Heterogeneous academic graph
Graphs:
- Nodes: Papers, authors, institutions, fields of study
- Edges: an author is “affiliated with” an institution, an author “writes” a paper, a paper “cites” a paper, and a paper “has a topic of” a field of study

Tasks:
- Predicting paper publication venues. Metric: Classification accuracy

Example model(s): See the leaderboard
Datasets: ogbn-mag
Note: This is a medium-scale OGB dataset that requires more computational resources than the other datasets.

Knowledge graphs
Graphs:
- Nodes: Entities
- Edges: Knowledge triples

Tasks:
- Predicting missing triples. Metric: Mean Reciprocal Rank (MRR)

Example model(s): TransE, DistMult, ComplEx, RotatE
Datasets: FB15k-237, WN18RR

Molecule classification
Graphs:
- Nodes: Atoms
- Edges: Bonds

Tasks:
- Predicting properties of molecules. Metric: ROC-AUC

Example model(s): See the leaderboard
Datasets: ogbl-molhiv

Protein-protein interaction networks
Graphs:
- Nodes: Gene nodes
- Edges: Interaction between gene nodes

Tasks:
- Node classification - protein function prediction. Metric: Classification accuracy

Example model(s): See methods on OGB node classification leaderboard
Datasets: https://snap.stanford.edu/biodata/datasets/10013/10013-PPT-Ohmnet.html

Drug-drug interaction networks
Graphs:
- Nodes: FDA-approved or experimental drug
- Edges: interactions between drugs (joint effect of taking the two drugs together)

Tasks:
- predict drug-drug interactions - Metric: Hits@K

Example model(s): See the leaderboard
Datasets: ogbl-ddi

1.2 Examples of good projects

Here we provide examples of good projects from the last two years' students, of applications of GNNs to impactful real-world use cases. You can find all of the blog posts from our course here: https://medium.com/stanford-cs224w.

- Stock Market Forecasting with Differential Graph Transformer by Xiang Li and Farzad Pourbabaee
- Tracing Historical Political Leaning in Newspapers with GNNs by Amy Guan, Poonam Sahoo, Karsen Wahal
- Predicting Water Quality in Unmonitored Networks with RGCNs by Tim Chen, Zhenyu Zhang, and Shuojia Fu
- Code Similarity Using Graph Neural Networks by Abhay Singhal, Suhas Chundi, and Patrick Donohue

- graphE2G: Linking Genes and Their Regulatory Elements Using Graph Neural Network by Valeh Valiollah Pour Amiri, Tony Kyle Zeng, Garyk Brixi
- Spread No More: Twitter Fake News Detection with GNN by Li Tian, Sherry Wu, and Yifei Zheng
- Learning Complex Dynamics of Particle Interactions through a Social Dynamics Lens by Nitish Gudapati, Shashvat Jayakrishnan, and Shreya Agrawal
- Automated Theorem Proving with Graph Neural Networks by Dan Jenson, Julian Cooper, and Daniel Huang
- Augmenting Your Notes Using Graph Neural Networks by Arjun Karanam and Michael Elabd
- Re-Mix and Match! by Jay Yu, Laura Wu, and Jiacheng Hu
- Fraudulent Activity Recognition in Graph Networks by Ansh Khurana, Aman Kansal, and Soumya Chatterjee
- Traffic Signal Control Using Decentralized MARL via Graph Modeling by Kevin Chen, Kelvin Christian, Haoyu Wang
- Spotify Track Neural Recommender System by Eva Batelaan, Thomas Brink, and Benjamin Wittenbrink
- RingSAGE: Cycle-Aware Molecular Regression with Virtual Nodes by Pranava Singhal, Herumb Shandilya and Harshvardhan Agarwal

1.3 Project Proposal [13.33%]

The project proposal is **13.33%** of the project grade and **4%** of your total grade.
For this project type, we expect you to cover the following in your proposal:

- Application domain
    - Which dataset(s) are you planning to use?
        - Is the dataset public? Do you plan to make it public before submitting the project? [*Note: this is a strict requirement]*
    - Describe the dataset/task/metric. Cite as relevant, especially the data.
    - Why did you choose the dataset?
- Graph ML techniques that you want to apply
    - Which graph ML model are you planning to use? (Be sure to cite!)
    - Describe the model(s) (try using figures and equations).
    - Why is/are the model(s) appropriate for the dataset(s) you have chosen?

1.4 Project Milestone [6.67%]

The project milestone is **6.67%** of the project grade and **2%** of your total grade. *It is graded on a credit/no credit basis.*
The purpose of the milestone is to encourage you to get started on the project early, and for the TAs to have something more substantial to give feedback on.
For this project type, possible deliverables for the milestone include:

- Code for
    - Processing the dataset
    - Training/evaluating the model
    - Any other programs required by your project
- Report draft
    - Problem description/motivation
    - Dataset description and processing
    - Model design and metrics

Note that you do not need to submit everything listed in the above bullet points. As long as you show that you have done substantial work by the submission deadline, you will receive full credit.

1.5 Project Report [80%]

This final submission is **80%** of the project grade and **24%** of your total grade.
Deliverables for the final submission and their weights are:

- Blog post (50 points)
    - Motivation & explanation of data/task (10 points)
    - Appropriateness & explanation of model(s) (10 points)
    - Insights + results (10 points)
    - Figures (10 points)
    - Code snippets (10 points)
- Colab (25 points)
    - Code: correctness, design (15 points)
    - Documentation: class/function descriptions, comments in code (10 points)
- Medium account name (5 points)

2. Tutorial on PyG functionality

Output: blog post, Google Colab

In this project, you will develop a tutorial that explains how to use existing PyG features and functionality, such as methods for GNN explainability or graph sampling. Compared to the first project, this project is much more focused on PyG functionality and less about demonstrating how GNNs can be used for a specific use case, although you will want to demonstrate the functionality concretely on datasets, such as those provided by PyG.

As a great example, you can take a look at a tutorial that PyG members Jan and Matthias wrote on link prediction on heterogeneous graphs with PyG. You can see that they explore the entire ML pipeline with PyG: how to construct a HeteroData object to represent the heterogeneous graph, how to split and load the data using transforms.RandomLinkSplit and loader.LinkNeighborLoader, how to create the heterogeneous GNN with nn.SageConv and

nn.to_hetero(), and how to train and evaluate the GNN. Each functionality that is explained is accompanied by side-by-side code examples. Your tutorial should similarly explore many PyG functionalities, with code snippets, and clearly explain where/how/why to use them, as well as how they all fit together.

2.1 Example tutorials and good projects from previous years

We provide several ideas below for potential tutorials. Feel free to use one of these ideas or to come up with one of your own. It may also be helpful to take a look at the list we provided of example datasets, so that you can demonstrate PyG functionality on concrete data.

- Tutorial on PyG's explainability module. For example, the tutorial could explore explainability in heterogeneous graphs or link prediction, and should make use of different evaluation metrics. The tutorial could also compare different explainer algorithms, such as GNNExplainer, CaptumExplainer, etc.
- Tutorial on PyG's methods for graph sampling (see the loader and sampler modules). For example, the tutorial could explore sampling for heterogeneous graphs and the consequences of different sampling parameters. You could also explore negative sampling for link prediction and how it's used in different PyG data loaders.
- Tutorial on GraphGym. GraphGym is a platform for designing and evaluating GNNs, introduced in the paper "Design Space for Graph Neural Networks" (You, Ying, and Leskovec, NeurIPS 2020). GraphGym is now officially supported as a part of PyG. A tutorial of GraphGym should explain how to run experiments with GraphGym and how to understand its results, essentially transforming its current usage description into an engaging blog post.

For reference, here are a few example tutorials that PyG has released:

- Understanding Graph Neural Networks with GraphGym: A Comprehensive Tutorial by Riya Sankhe, Emma E Passmore, Ashlee Kupor
- GNN Explainability with PyG by Bohao He
- Jan Eric Lenssen and Matthias Fey, "Link Prediction on Heterogeneous Graphs with PyG"
- Guohao Li, "A Principled Approach to Aggregations"

Here are examples of good projects from students in the previous years:

- A tour of PyG's data loaders by Grant Uy and Huijian Cai
- A Primer on Explainers, Explanations, and their Metrics in PyG: or how to explain explanations by Samy Cherfaoui and Eric Tang
- Introducing DistMult and ComplEx for PyTorch Geometric by David Kuo and Riya Sinha

2.2 Project Proposal [13.33%]

The project proposal is **13.33%** of the project grade and **4%** of your total grade.
For this project type, we expect you to cover the following in your proposal:

- Functionality
    - What general ML problem are you tackling, e.g., link prediction on heterogeneous graphs, explainability, GNN design?
    - Which PyG functionalities (e.g., modules, classes, functions) are you going to cover that address this problem?
- Demonstration
    - What dataset(s) will you use to concretely demonstrate these functionalities? Be sure to cite.
    - Which specific steps in the ML pipeline will you walk through? How will they showcase the classes and functions you plan to cover?

2.3 Project Milestone [6.67%]

The project milestone is **6.67%** of the project grade and **2%** of your total grade. *It is graded on a credit/no credit basis.*
The purpose of the milestone is to encourage you to get started on the project early, and for the TAs to have something more substantial to give feedback on.
For this project type, possible deliverables for the milestone include:

- Code for
    - Dataset processing
    - Demonstrating PyG functions
    - Any other programs required by your project
- Report draft
    - Problem explanation
    - Outline of the tutorial

2.4 Project Report [80%]

This final submission is **80%** of the project grade and **24%** of your total grade.
Deliverables for the final submission and their weights are:

- Blog post (50 points)
    - Motivation & explanation of ML problem (10 points)
    - Explanation of PyG functionalities (20 points)
    - Figures (10 points)
    - Code snippets (10 points)
- Colab (25 points)
    - Code: correctness, design (15 points)
    - Documentation: class/function descriptions, comments in code (10 points)
- Medium account name (5 points)

Note that you do not need to submit everything listed in the above bullet points. As long as you show that you have done substantial work by the submission deadline, you will receive full credit.

3. Implementation of cutting-edge research

Output: PyG PR, short blog post

In this project, you will identify a recent research paper in graph ML that develops interesting methods and implement the work as a pull request (PR) to the contrib package in PyG. We recommend that you take a look at PyG's contribution guidelines. Your code should build on existing PyG functionality wherever possible so that your new functionality would integrate smoothly into the PyG framework. Your code should also include testing of your new functionality. Finally, you should write a short blog post explaining your implementation and how it can be used. We will give extra credit for PRs that are approved by PyG!

Note: this project is more manageable if you are already comfortable with PyTorch and deep learning. We would also strongly recommend working in a group of 3 for this project.

3.1 Example research ideas

In collaboration with the PyG team, we have built a list (below) of recommended research papers and ideas. You're welcome to choose one of these ideas or to come up with your own. If you're looking for potential papers, it may be useful to look at the proceedings of the top ML and data conferences, including KDD, NeurIPS, ICML, ICLR, AAAI, and WebConf.

- Time series forecasting with Dynamic Graphs: paper
- Travel Demand Prediction with Spatial-Temporal GNNs: paper
- Scaling GNNs with Approximate PageRank: paper
- Relational Deep Learning
    - Building scalable GNN infrastructure for Large Databases
        - Must extend beyond the RAM capabilities of a single machine
        - Can provide a few options on graph DBs and KV stores
        - Investigate data loading and caching algorithms
    - Related Work: Survey; RelGNN (GNN); RelGT (Graph Transformer)
- Knowledge Graphs: Embeddings and Generalizations
    - ComplEx: Complex embeddings for simple link prediction
    - NodePiece: Compositional and Parameter-Efficient Representations of Large Knowledge Graphs
    - RotatE: Knowledge Graph Embeddings by relational rotation in complex space.
    - GoldE: Generalizing Knowledge Graph Embedding with Universal Orthogonal Parameterization
    - ULTRA: A Foundation Model for KG Reasoning
- Geometric/Equivariant GNNs

    - E(n) Equivariant Graph Neural Networks
    - Geometric Latent Diffusion Models for 3D Molecule Generation
    - Equivariant Flow Matching
    - Enabling Efficient Equivariant Operations in the Fourier Basis via Gaunt Tensor Products
- Temporal Graph Learning
    - Comprehensive survey blog post
- Graph Transformer
    - Survey paper
    - Graphormer paper
    - Transformer-M paper
    - TokenGT (Kim et al.)
    - GraphGPT (Gao et al.)
    - CoBFormer (Xing et al.)
- LLMs x GNNs
    - Survey GitHub
    - GNNs as Retrieval/RAG mechanism for LLMs: paper
        - Reducing hallucinations in LLMs: paper, survey
    - Using LLMs to create better node embeddings for GNNs: paper
    - Using GNNs to create better tokens for LLMs: approach 1, approach 2

3.2 Project Proposal [13.33%]

The project proposal is **13.33%** of the project grade and **4%** of your total grade.
For this project type, we expect you to cover the following in your proposal:

- Research paper
    - Which paper do you intend to implement? Describe the overall problem that the paper is trying to solve and the specific methods in the paper that you will implement. (Be sure to cite!)
    - Why are these methods interesting? What is their significance in graph ML, e.g., allowing us to scale to larger graphs, improving expressivity?
- Implementation in PyG
    - Sketch out a high-level API for how you intend to implement these methods. What new classes, functions, etc., will you introduce? Where will you utilize existing PyG and PyTorch functionality so that your PR integrates smoothly into the PyG framework? (for example, your code could be a subclass of an existing PyG or PyTorch base class, such as torch.nn.Module)
    - What does your new functionality allow us to do in PyG that we couldn't do before?

3.3 Project Milestone [6.67%]

The project milestone is **6.67%** of the project grade and **2%** of your total grade. *It is graded on a credit/no credit basis.*
The purpose of the milestone is to encourage you to get started on the project early, and for the TAs to have something more substantial to give feedback on.
For this project type, possible deliverables for the milestone include:

- Code
    - Algorithm implementation (WIP)
    - PyG API documentation
- Report draft
    - Problem description and motivation
    - Paper summary
    - Implementation explanation

Note that you do not need to submit everything listed in the above bullet points. As long as you show that you have done substantial work by the submission deadline, you will receive full credit.

3.4 Project Report [80%]

This final submission is **80%** of the project grade and **24%** of your total grade.
Deliverables for the final submission and their weightages are:

- Blog post (25 points)
    - Motivation & explanation of research method (10 points)
    - Explanation of implementation (10 points)
    - Code snippets (5 points)
- Pull request (50 points)
    - Correctness (15 points)
    - Documentation: class/function descriptions, comments in code (15 points)
    - Ease of integration into PyG (10 points)
    - Ease of use (10 points)
- Medium account name (5 points)

4. Project logistics

We recommend project groups of 3 students, particularly for Project 3, but groups of 2 are also allowed. Solo projects are discouraged; please see Ed. The project is worth 30% of your course grade.

4.1 What to submit

What to submit for project proposal [13.33% of *project* grade, 4% of *total* grade]
Due October 21, 2025, 11:59pm PT

- Who is in your group
- Please refer to the Project Proposal subsection under your project type for instructions on what to include in your proposal
- **Format:** please use the NeurIPS'22 style file and aim for 2 pages without references. The abstract is not needed.

What to submit for project milestone [6.67% of *project* grade, 2% of *total* grade]
Due November 11, 2025, 11:59pm PT

- Please refer to the Project Milestone subsection under your project type for instructions on what to include in your proposal

What to submit for final project [80% of *project* grade, 24% of *total* grade]
Due December 11, 2025, 11:59pm PT

- Private draft link to blog post (see instructions below)
- Name of Medium account
- Google Colab (Projects 1+2) or PR to PyG's contrib (Project 3)

4.2 Project Feedback and Mentor Assignments

Each group will be assigned a course assistant as a mentor for their project whose purpose is to provide feedback for your project as you progress through the quarter. Meeting with your assigned mentor is not required, but it is recommended that you meet with them at least once during the quarter to make sure you are headed in the right direction.

Mentor assignments will begin after the due date for submitting project proposals (October 21, 2025).

4.3 Writing blog posts on Medium

For all three project types, a final product of your project will be your blog post. For this course, you will be writing and we will be releasing your blog posts on Medium. Writing a blog post in Medium is easy and intuitive; see step-by-step instructions below. We also provide tips on how to write an engaging blog post.

1. Sign up / sign in on Medium.
2. For each group, one member should set up a draft
    a. To start a new draft, go here.
    b. To restart from the existing draft, go here.
3. Editing the draft
    a. Only one member (who owns the draft) can directly edit the blog post.

    b. If you want to write the blog together with your members, we suggest you work on the Google doc first and then copy-paste the eventual texts/images to the Medium draft.
4. Submitting your draft
    a. Your project submission will include the **draft link** to your blog post. To get the link: on the editing page, click the "..." button (located right next to the "publish"), then click "Share draft link."
    b. Please also share your account name with us.
    c. **Note: Do NOT publish your blog without our review.**
5. After the course
    a. Publishing on the CS224W Medium page
        i. After the course, you will have a chance to incorporate final feedback into your blog post, then we will work on publishing it on our course's Medium page.
        ii. To do this, we will add you as a writer to the stanford-cs224w publication page.
        iii. Once you are added as our writer, you can click the "..." button and further click "Add to publication".
    b. We will also work with the PyG team to select blog posts of interest to publish at PyG.org and on our social media.
    c. You will get grading regardless of whether your blogs are published or not.

4.4 Instructions on Google Colab

Your Google Colab should include:

- A high-level summary of what the code is about and what the task is
- All the code to reproduce your results in the blog posts (including data preprocessing, model definition, and train/evaluation pipeline).
- Detailed comments of what each cell does.

See here for examples of good Google Colabs.

4.5 Compute resources

Google cloud retrieval link.

1. You will be asked for a name and email address, which needs to match your school domain (either @stanford.edu or @cs.stanford.edu).
2. A confirmation email will be sent to you with a coupon code.
3. When you redeem your coupon, make sure you are logged into a **personal Google account** (unlike in step 1).
4. You can only request ONE code per unique email address.
5. To create a VM, you can take a look at the guidelines here (starting from **Step 3**) and here (see **Set Up Google Cloud VM Image**). If you need GPU for your project, you will need to request GPU quota -- see the instructions under **Request GPU Quota** in the

second link. Please do this soon, since it can take a few days for your request to be approved.

Once you have the GCP credits, follow this tutorial to link them to your Colab notebook.

NOTE: Google also provides $300 in credits for new customers, so definitely check that out too if it applies to you! Please let us know if you have any issues with redeeming credits or finding sufficient compute resources for your projects.

4.6 Project sharing policy

Policy on sharing a final project with another course:

- You should make sure that you follow all the guidelines and requirements for the CS224W project (in addition to the requirements of the other class). So, if you'd like to combine your CS224W project with a class X but class X's policies don't allow for it, you cannot do it.
- You cannot turn in an identical project for both classes, but you can share common infrastructure/code base/datasets across the two classes.
- In your milestone and final report, clearly indicate which part of the project is done for CS224W and which part is done for a class other than CS224W. For shared projects, we also require that you submit the final report from the class you're sharing the project with.

5. Extra credit for approved PRs

- You will also be considered for up to an extra 1-3% of your course grade if you create *approved* pull requests to OGB, PyG, or GraphGym that add useful functionalities or fix bugs, depending on the significance of the contribution.
- In the spirit of open-source development, you are highly encouraged to participate in improving these resources as you work with them throughout the course and in your final projects. For example, for each project, we will give extra credit for the following:
    - Project 1: contribute your final project to the OGB leaderboard together with open-source code using PyG. This is particularly exciting if your model is able to achieve SoTA performance!
    - Project 1 or 2: write a blog post that we choose to feature on the PyG website
    - Project 3: have your project PR (or part of it) approved by the PyG team
- Also, we encourage students doing any project to:
    - Find bugs and create issues
    - Propose missing functionality and implement it
    - Contribute useful modules to the GraphGym pipeline
- We highly recommend consulting the TAs before contributions so that we can confirm the details of incorporating the contribution and granting the extra grade.

6. Tips on writing blog posts

First, please read this article carefully to learn about how to write machine learning blog posts. For this course project, please also follow the instructions below.

In the blog posts, you should include the following:

- At the beginning of your blog post, include "By XXX, YYY, ZZZ as part of the Stanford CS224W course project.", where XXX, YYY, ZZZ are the names of the team members.
- Wherever applicable:
    - The domain(s) that you are applying graph ML to.
    - Dataset descriptions (source, pre-processing etc).
    - Results that you obtain using the model on the dataset
- Paper references as [1], [2], etc., see this blog post as an example.
- Step-by-step explanation of graph ML techniques you are using
    - You can assume the following for the readers.
        - Readers are familiar with machine learning (e.g., CS229) and deep learning (e.g., CS230) concepts. You do not need to explain them in detail.
        - Readers are familiar with PyTorch.
        - Readers are not familiar with graph ML.
    - Some code snippets of how you used PyG/PyTorch to implement the techniques
- **Visualizations** that would make the blog posts intriguing to read.
    - Gifs > Images > Text to show your methods and results.
    - Try to use videos, images, flow charts as much as possible.
    - The more visualization, the better. Reading text-occupied blogs is often painful.
    - Provide image credits if you are adopting figures from other places (we encourage you to make your own figures).
- Link to your Google Colab that can be used to reproduce your results.
- Avoid criticizing research / research orgs. You are here to showcase your work, not to write opinion pieces.

A good blog post should

- be fun to read with many figures and visualizations.
- be easy to follow even for graph ML novice.
- clearly convey the potential of graph ML.
- contain a good amount of PyG code snippets, with explanations, to understand how PyG is used in the project.
- be around 10 minutes to read (although this is not a hard constraint).

Examples of good blog posts:

- http://peterbloem.nl/blog/transformers
- https://tkipf.github.io/graph-convolutional-networks/
- https://towardsdatascience.com/graph-neural-networks-as-neural-diffusion-pdes-8571b8c0c774

- https://blog.twitter.com/engineering/en_us/topics/insights/2021/temporal-graph-networks