Input: Network

Predictions: Node labels, New links, Generated graphs and subgraphs
Key Questions for GNN Design

- **GNN architectural design:**
  - How to find a good GNN design for a specific GNN task?
- **Important but challenging:**
  - Domain experts want to use SOTA GNN on their specific tasks, however...
    - There are tons of possible GNN architectures
      - GCN, GraphSAGE, GAT, GIN, ...
    - **Issue:** Best design in one task can perform badly for another task
    - Redo hyperparameter grid search for each new task is NOT feasible
- **Topic for today:**
  - Study for the _GNN design space and task space_
  - **GraphGym**, a powerful platform for exploring different GNN designs and tasks
Design: a concrete model instantiation
- E.g., a 4-layer GraphSAGE

Design dimensions characterize a design
- E.g., the number of layers $L \in \{2, 4, 6, 8\}$

Design choice is the actual selected value in the design dimension
- E.g., the number of layers $L = 2$

Design space consists of a Cartesian product of design dimensions

Task: A specific task of interest
- E.g., node classification on Cora, graph classification on ENZYMES

Task space consists of all the tasks we care about
Recap: GNN Design Space

Intra-layer Design:
GNN Layer = Transformation + Aggregation
• We propose a general instantiation under this perspective
Inter-layer Design

- We explore different ways of organizing GNN layers

Intra-layer Design: 4 dims
- Linear
- BatchNorm
- Dropout
- Activation
- Aggregation

Inter-layer Design: 4 dims
- MLP Layer
- GNN Layer
- GNN Layer
- MLP Layer
- MLP Layer

Pre-process layers:
Important when expressive node feature encoder is needed
E.g., when nodes are images/text

Skip connections:
Improve deep GNN’s performance

Post-process layers:
Important when reasoning or transformation over node embeddings are needed
E.g., graph classification, knowledge graphs

Learning Configuration: 4 dims
- Batch size
- Learning rate
- Optimizer
- Training epochs
Recap: GNN Design Space

Learning configurations

- Often neglected in current literature
- But we found they have high impact on performance
Overall: A GNN design space

Intra-layer design
- Batch Normalization: True, False
- Dropout: False, 0.3, 0.6
- Activation: ReLU, PReLU, Swish
- Aggregation: Mean, Max, Sum

Inter-layer design
- Layer connectivity: Stack, Skip-Sum, Skip-Cat
- Pre-process layers: 1, 2, 3
- Message passing layers: 2, 4, 6, 8
- Post-process layers: 1, 2, 3

Learning configuration
- Batch size: 16, 32, 64
- Learning rate: 0.1, 0.01, 0.001
- Optimizer: SGD, ADAM
- Training epochs: 100, 200, 400

In total: 315K possible designs

Our Purpose:
- We don’t want to (and we cannot) cover all the possible designs
- A mindset transition: We want to demonstrate that studying a design space is more effective than studying individual GNN designs
Categorizing GNN tasks

- **Common practice:** node / edge / graph level task
- Reasonable but not precise
  - **Node prediction:** predict clustering coefficient vs. predict a node’s subject area in a citation networks – completely different task
- But creating a precise taxonomy of GNN tasks is very hard!
  - Subjective; Novel GNN tasks can always emerge

**Our innovation:** a quantitative task similarity metric

- Purpose: understand GNN tasks, transfer the best GNN models across tasks
Quantitative task similarity metric

1) Select “anchor” models \((M_1, \ldots, M_5)\)

2) Characterize a task by ranking the performance of anchor models

3) Tasks with similar rankings are considered as similar

<table>
<thead>
<tr>
<th>Task Similarity Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor Model Performance ranking</td>
</tr>
<tr>
<td>Task A</td>
</tr>
<tr>
<td>Task B</td>
</tr>
<tr>
<td>Task C</td>
</tr>
</tbody>
</table>

Task A is similar to Task B
Task A is not similar to Task C

How do we select the anchor models?
Selecting the anchor models

1) Select a small dataset
   - E.g., node classification on Cora

2) Randomly sample \( N \) models from our design space, run on the dataset
   - E.g., we sample 100 models

3) Sort these models based on their performance: evenly select \( M \) models as the anchor models, whose performance range from the worst to the best
   - E.g., we sample 12 models in our experiments

Goal: Cover a wide spectrum of models: A bad model in one task could be great for another task
A General GNN Task Space

- **We collect 32 tasks**: node / graph classification

  (We include link prediction results in the Appendix)

  - **6 Real-world node classification tasks**
    - Predict node properties:
      - Clustering coefficient
      - PageRank

  - **12 Synthetic node classification tasks**
    - Predict node properties:
      - Clustering coefficient
      - PageRank

  - **6 Real-world graph classification tasks**
    - Predict graph properties:
      - Average path length

  - **8 Synthetic graph classification tasks**
    - Predict graph properties:
      - Average path length
Evaluating a design dimension:
- “Is BatchNorm generally useful for GNNs?”

The common practice:
- (1) Pick one model (e.g., a 5-layer 64-dim GCN)
- (2) Compare two models, with BN = True / False

Our approach:
- Note that we have defined $315K \times 32 \approx 10M$ model-task combinations
- (1) Sample from 10M possible model-task combinations
- (2) Rank the models with BN = True / False

How do we make it scalable & convincing?
Evaluating GNN Designs

- Evaluating a design dimension: Controlled random search
  - a) Sample random model-task configurations, perturb $\text{BatchNorm} = [\text{True}, \text{False}]$
  - Here we control the computational budget for all the models

(a) Controlled Random Search

<table>
<thead>
<tr>
<th>GNN Design Space</th>
<th>GNN Task Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchNorm</td>
<td>Activation</td>
</tr>
<tr>
<td>True</td>
<td>relu</td>
</tr>
<tr>
<td>False</td>
<td>relu</td>
</tr>
<tr>
<td>True</td>
<td>relu</td>
</tr>
<tr>
<td>False</td>
<td>relu</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>True</td>
<td>prelu</td>
</tr>
<tr>
<td>False</td>
<td>prelu</td>
</tr>
</tbody>
</table>
Evaluating GNN Designs

- **b) Rank** BatchNorm = [True, False] by their performance (lower ranking is better)
- **c) Plot Average / Distribution of the ranking of** BatchNorm = [True, False]

**Summary:** Convincingly evaluate any new design dimension, e.g., evaluate a new GNN layer we propose.

---

### (b) Rank Design Choices by Performance

<table>
<thead>
<tr>
<th>GNN Design Space</th>
<th>Experimental Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchNorm</td>
<td>Val. Accuracy</td>
</tr>
<tr>
<td>True</td>
<td>0.75</td>
</tr>
<tr>
<td>False</td>
<td>0.54</td>
</tr>
<tr>
<td>True</td>
<td>0.88</td>
</tr>
<tr>
<td>False</td>
<td>0.88</td>
</tr>
<tr>
<td>True</td>
<td>0.89</td>
</tr>
<tr>
<td>False</td>
<td>0.36</td>
</tr>
</tbody>
</table>

### (c) Ranking Analysis
Results 1: A Guideline for GNN Design

- Certain design choices exhibit **clear advantages**
  - Intra-layer designs:

  **Explanation:**
  GNNs are hard to optimize

  **Explanation:**
  This is our new finding!

  **Explanation:**
  GNNs experience underfitting more often

  **Explanation:**
  Sum is the most expressive aggregator
Results 1: A Guideline for GNN Design

- Certain design choices exhibit clear advantages
  - Inter-layer designs

Optimal number of layers is hard to decide
Highly dependent on the task

Explanation:
Skip connection enable hierarchical node representation
Results 1: A Guideline for GNN Design

- Certain design choices exhibit clear advantages
  - Learning configurations

Optimal batch size and learning rate is hard to decide

Explanation:
Adam is more robust
More training epochs is better
Best GNN designs in different tasks vary significantly

Motivate that studying a task space is crucial
Results 2: Understanding GNN Tasks

- **Build a GNN task space**

  We compute pairwise similarities between all GNN tasks

  **Recall how we compute task similarity**

<table>
<thead>
<tr>
<th>Task A</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor Model Performance ranking</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similarity to Task A</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task B</td>
<td>M1</td>
<td>M3</td>
<td>M2</td>
<td>M4</td>
<td>M5</td>
</tr>
<tr>
<td>Similarity to Task A</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task C</td>
<td>M5</td>
<td>M1</td>
<td>M4</td>
<td>M3</td>
<td>M2</td>
</tr>
<tr>
<td>Similarity to Task A</td>
<td>-0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  **Task similarity computation is cheap:** Using 12 anchor models is a good approximation!
Results 2: Understanding GNN Tasks

- **GNN task space is informative**

  **Group 1:**
  Tasks rely on **feature information**
  - Node/graph classification tasks, where input graphs have high-dimensional features
    - Cora graph has 1000+ dim node feature

  **Group 2:**
  Tasks rely on **structural information**
  - Nodes have few features
  - Predictions are highly dependent on graph structure
    - Predicting clustering coefficients
Results 2: Understanding GNN Tasks

- GNN task space is informative

<table>
<thead>
<tr>
<th>Task</th>
<th>Pre layers</th>
<th>MP layers</th>
<th>Post layers</th>
<th>Connectivity</th>
<th>AGG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task A</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>skip-sum</td>
<td>sum</td>
</tr>
<tr>
<td>Task B</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>skip-sum</td>
<td>sum</td>
</tr>
<tr>
<td>Task C</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>skip-cat</td>
<td>mean</td>
</tr>
</tbody>
</table>
Results 3: Transfer to Novel Tasks

- **Case study:** generalize best models to **unseen** OGB ogbg-molhiv task:
  - ogbg-molhiv is unique from other tasks: 20x larger, imbalanced (1.4% positive) and requires out-of-distribution generalization

- **Concrete steps for applying to a novel task:**
  - **Step 1:** Measure 12 anchor model performance on the new task
  - **Step 2:** Compute similarity between the new task and existing tasks
  - **Step 3:** Recommend the best designs from existing tasks with high similarity
Results 3: Transfer to Novel Tasks

- Our task space can **guide best model transfer to novel tasks!**

We pick 2 tasks:
- **Task A:** Similar to OGB
- **Task B:** Not similar to OGB

**Findings:**
- Transfer the best model from Task A achieves SOTA on OGB
- Transfer the best model from Task B performs badly on OGB

<table>
<thead>
<tr>
<th></th>
<th>Task A: graph-scalefree-const-path</th>
<th>Task B: node-CoauthorPhysics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best design in our design space</td>
<td>(1, 8, 3, skipcat, sum)</td>
<td>(1, 4, 2, skipcat, max)</td>
</tr>
<tr>
<td>Task Similarity with ogbg-molhiv</td>
<td>0.47</td>
<td>-0.61</td>
</tr>
<tr>
<td>Performance after transfer to ogbg-molhiv</td>
<td>0.785</td>
<td>0.736</td>
</tr>
</tbody>
</table>

Previous SOTA: 0.771
GNN Design Space: Summary

- Systematic investigation of:
  - General guidelines for GNN design
  - Understandings of GNN tasks
  - Transferring best GNN designs across tasks
  - **GraphGym**: Easy-to-use code platform for GNN

---

**GNN Design Space**

- Intra-layer Design: 4 dims
  - Linear
  - BatchNorm
  - Dropout
  - Activation
  - Aggregation

- Inter-layer Design: 4 dims
  - MLP Layer
  - Pre-process layers
  - GNN Layer
  - Layer connectivity
  - GNN Layer
  - Message passing layers
  - GNN Layer
  - Post-process layers
  - MLP Layer

**GNN Task Space**

- Task format: Level-Dataset
  - node-AmazonComputers
  - node-AmazonPhoto
  - node-CiteSeer
  - node-CoauthorCS
  - node-CoauthorPhysics
  - node-Cora
  - graph-BZR
  - graph-COX2
  - graph-PROTEINS
  - graph-DD
  - graph-ENZYMES
  - graph-MDB
  - graph-scalefree
  - graph-smallworld
  - node-scalefree
  - node-smallworld

**Controlled Random Search**

<table>
<thead>
<tr>
<th>GNN Design Space</th>
<th>GNN Task Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>BatchNorm</td>
<td>Connectivity</td>
</tr>
<tr>
<td>True</td>
<td>level</td>
</tr>
<tr>
<td>False</td>
<td>dataset</td>
</tr>
<tr>
<td>False</td>
<td>node</td>
</tr>
</tbody>
</table>

- **Experimental Results**

<table>
<thead>
<tr>
<th>Task</th>
<th>Pre layers</th>
<th>MP layers</th>
<th>Post layers</th>
<th>Connectivity</th>
<th>AGG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task A</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>skip-sum</td>
<td>sum</td>
</tr>
<tr>
<td>Task B</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>skip-sum</td>
<td>sum</td>
</tr>
<tr>
<td>Task C</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>skip-cat</td>
<td>mean</td>
</tr>
</tbody>
</table>

- **Rank Design Choices by Performance**

<table>
<thead>
<tr>
<th>Val. Accuracy</th>
<th>Design Choice Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>0.54</td>
<td>2</td>
</tr>
<tr>
<td>0.88</td>
<td>1 (a tie)</td>
</tr>
<tr>
<td>0.86</td>
<td>1 (a tie)</td>
</tr>
</tbody>
</table>

- **Ranking Analysis**

- **Accuracy Ranking Distribution**

- **False Batch Normalization**

---

J. You, R. Ying, J. Leskovec. *Design Space of Graph Neural Networks*, NeurIPS 2020
Pre-Training
Graph Neural Networks
Graph ML in Scientific Domains

- **Chemistry**: Molecular graphs
  - Molecular property prediction

  \[ f(\text{molecule}) = \text{toxic?} \]

- **Biology**: Protein-protein association graphs
  - Protein function prediction

  \[ f(\text{protein association}) = \text{biological activity?} \]

Our running example today
GNNs for Graph Classification

- GNNs obtain an embedding of an entire graph by following two steps
  - **Iteratively aggregate neighboring information** to obtain node embeddings
  - **Pool node embeddings** to obtain a graph embedding
GNNs for Graph Classification

- Node embeddings capture local neighborhood structure
- The embedding of an entire graph is a global aggregation of such node embeddings
Challenges of Applying ML

- Two fundamental challenges in applying ML to scientific domains

1. Scarcity of labeled data
   - Obtaining labels requires expensive lab experiments
     → ML models overfit to small training data

2. Out-of-distribution prediction
   - Test examples tend to be very different from training examples
     → ML models extrapolate poorly
Deep learning models have a lot of parameters to train (e.g., in the order of millions).

\[ \text{(Labeled training data)} \ll \text{(Parameters)} \]

Deep learning models are extremely prone to overfitting on small labeled data.
Deep learning models extrapolate poorly

- Models often make predictions based on spurious correlations in a dataset [Sagawa et al. ICML 2020]
- Ex) Image classification between “polar bear” and “brown bear”
- During training:
  - Most “polar bears” have the snow background
  - Most “brown bears” have the grass background
  - Model can learn to make prediction based on the image background, rather than the animal itself.
- At test time, what if we see “polar bear” on the grass?
Goal: Improve model’s out-of-distribution prediction performance even with limited data.

Key idea: Inject domain knowledge into a model before training on scarcely-labeled tasks!

- The model already knows the domain knowledge before training on data
- So that the model can
  - Generalize well without many task-specific labeled data
  - Extract essential (non-spurious) pattern that allows better extrapolation.
Effective Solution: Pre-Training

- We **pre-train a model on relevant tasks**, where data is abundant.
  - After pre-training, the model parameters already contain domain knowledge.
- For **downstream tasks** (what we care about, typically with small #labeled data)
  - We start from the pre-trained parameters and fine-tuning them.
Pre-Training is Successful

- Pre-training has been hugely successful in computer vision and natural language processing.
  - Pre-training improves label-efficiency.
  - Pre-training improves out-of-distribution performance [Hendrycks et al. ICML 2019]
- Pre-training is a powerful solution to the two ML challenges in scientific applications
  - Scarce labels
  - Out-of-distribution prediction
Pre-training GNNs

- Let’s consider pre-training GNNs!
- We design GNN pre-training strategies and systematically investigate

Q1. How effective is pre-training GNNs?

Q2. What is the effective pre-training strategy?
Let’s think about molecular property prediction for drug discovery.

- **Naïve strategy**
  Multi-task supervised pre-training on relevant labels.
Molecule classification

**Task:** Binary classification. ROC-AUC as metric

\[ f(\text{molecule}) = \{0,1\} \]

**Supervised pre-training data**

- 1310 diverse binary bioassays annotated over ~450K molecules

**Downstream task** *(what we care about!)*

- 8 molecular classification datasets (relatively-small, 1K—100K molecules)

**Data split:** Scaffold (test molecules are out-of-distribution)
How Effective is Pre-training GNNs?

- **Naïve strategy**: Multi-task supervised pre-training on relevant labels.
  - Limited performance improvement on downstream tasks. Often leads to negative transfer.
**Key idea**: Pre-train both node and graph embeddings.

→ GNN can capture domain-specific knowledge of both local and global structure.
What is the Effective Strategy?

- **Key idea**: Pre-train both node and graph embeddings.

Embedding illustration

Naïve strategies

(a.iii) **Node-level +**  
**Graph-level pre-training**

(a.i) **Node-level**  
**pre-training only**

(a.ii) **Graph-level**  
**pre-training only**
Proposed Pre-Training Methods

<table>
<thead>
<tr>
<th></th>
<th>Node-level</th>
<th>Graph-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute prediction</td>
<td>Attribute Masking</td>
<td>Supervised Attribute Prediction</td>
</tr>
<tr>
<td>Structure prediction</td>
<td>Context Prediction</td>
<td>Structural Similarity Prediction</td>
</tr>
</tbody>
</table>

Self-supervised
(No need for external labels)
Attribute Masking: Algorithm

- Mask node attributes
- Use GNNs to generate node embeddings.
- Use the embeddings to predict masked attributes.
Intuition

Through solving the masked attribute prediction task, a GNN is forced to learn domain knowledge, e.g., chemical rules.
Context Prediction: Algorithm

- For each graph, sample one center node.
- Extract neighborhood and context graphs.
- Use GNNs to encode neighborhood and context graphs into vectors.
- Maximize/minimize the inner product between true/false (neighborhood, context) pairs.
Intuition

Subgraphs that are surrounded by similar contexts are semantically similar.

In natural language processing, this is called distributional hypothesis, and is exploited in the word2vec model [Mikolov et al. NIPS 2013].
Supervised Attribute Prediction

- Multi-task supervised training on many relevant labels.
Overall Strategy

1. Node-level pre-training
2. Graph-level pre-training
3. Fine-tuning on downstream tasks

- Self-supervised node-level pre-training
- Supervised graph-level pre-training
  - Toxicity A? Bioactivity A?
  - Toxicity B? Bioactivity B?

GNN

Node pre-train

Graph pre-train

Downstream task 1

Fine-tune

Downstream task N
Results of Our Strategy

- Avoids negative transfer.
- Significantly improve the performance.

Avoids negative transfer

Molecule classification performance

ROCAUC improvement over no pre-training

Downstream datasets

Non pre-trained GNNs

Our strategy

Naïve strategy

Avoids negative transfer
Comparison of GNN models

- When different GNN models are pre-trained, the most expressive model (GIN) benefits the most from pre-training.
- Intuition: Expressive model can learn to capture more domain knowledge than less expressive models.

<table>
<thead>
<tr>
<th></th>
<th>Chemistry</th>
<th>Biology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-pre-trained</td>
<td>Pre-trained</td>
</tr>
<tr>
<td>GIN</td>
<td>67.0</td>
<td>74.2</td>
</tr>
<tr>
<td>GCN</td>
<td>68.9</td>
<td>72.2</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>68.3</td>
<td>70.3</td>
</tr>
<tr>
<td>GAT</td>
<td>66.8</td>
<td>60.3</td>
</tr>
</tbody>
</table>
Pre-Training GNNs: Summary

- GNNs have important applications in scientific applications, but they present challenges of
  - Label scarcity
  - Out-of-distribution prediction
- Pre-training is promising to tackle the challenges.
- However, naïve pre-training strategy gives sub-optimal performance and even leads to negative transfer.
- Our strategy: Pre-train both node and graph embeddings → Leads to significant performance gain on downstream tasks.
CS224W: Wrap-Up
Modern deep learning toolbox is designed for simple sequences & grids
This Course

How can we develop neural networks that are much more broadly applicable?

Graphs are the new frontier of deep learning
Main question:
How do we take advantage of relational structure for better prediction?

Knowledge Graphs
Image credit: Maximilian Nickel et al

Scene Graphs
Image credit: math.hws.edu

Code Graphs
Image credit: ResearchGate

Molecules
Image credit: MDPI

3D Shapes
Input: Network

Predictions: Node labels, New links, Generated graphs and subgraphs
Models of Interest: Invariances

- **Perceptrons**: Function regularity
- **CNNs**: Translation
- **Group-CNNs**: Translation + Rotation, Global groups
- **LSTMs**: Time warping

- **DeepSets / Transformers**: Permutation
- **GNNs**: Permutation
- **Intrinsic CNNs**: Isometry / Gauge choice
There is exciting relational structure in many real-world problems

- Molecules/Proteins as strings vs. graphs
- Travel time duration over the map graph

Identifying and harnessing this relational structure leads to better predictions

- AlphaFold
- Biomedicine
- Recommender systems
You learned a lot!

- **Theory:**
  - Models, architectures, approaches
- **Practice:**
  - Collab notebooks
  - Homeworks
- **Creative research:**
  - Course project
- **The real-world use cases and applications**
What Next?

- **Project write-ups:**
  - Thurs Dec. 14, Midnight *(11:59PM)* Pacific Time

- **Courses:**
  - **CS246: Mining Massive Datasets (Spring)**
    - Data Mining & Machine Learning for big data
      - (big==doesn’t fit in memory/single machine)
      - Fast clever algorithms for real-world problems
      - Distributed data processing frameworks: MapReduce, Spark

No late days!
Thank you, team!!!

Instructor
- Jure Leskovec

Course Assistants
- Xikun Zhang
  Head CA
- Hamed Nilforoshan
- Aditya Agrawal
- Abhinav Garg

Guest Instructor
- Joshua Robinson
- Matthew Jin
- Yunqi Li
- Tolu Oyeniyi
- Chenshu (Jupiter) Zhu
- Pratham Soni
- Anirudh Sriram
You Have Done a Lot!!!
And (hopefully) learned a lot!!!
- Answered questions and proved many interesting results
- Implemented a number of methods
- And did excellently on the project!

Thank You for the Hard Work!!!