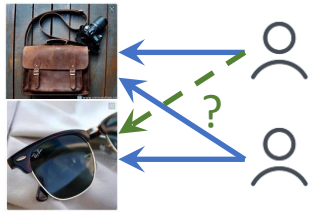
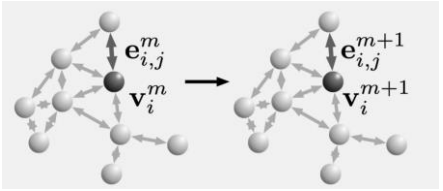
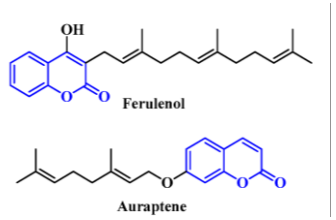
Graph Neural Networks Applications

Rex Ying



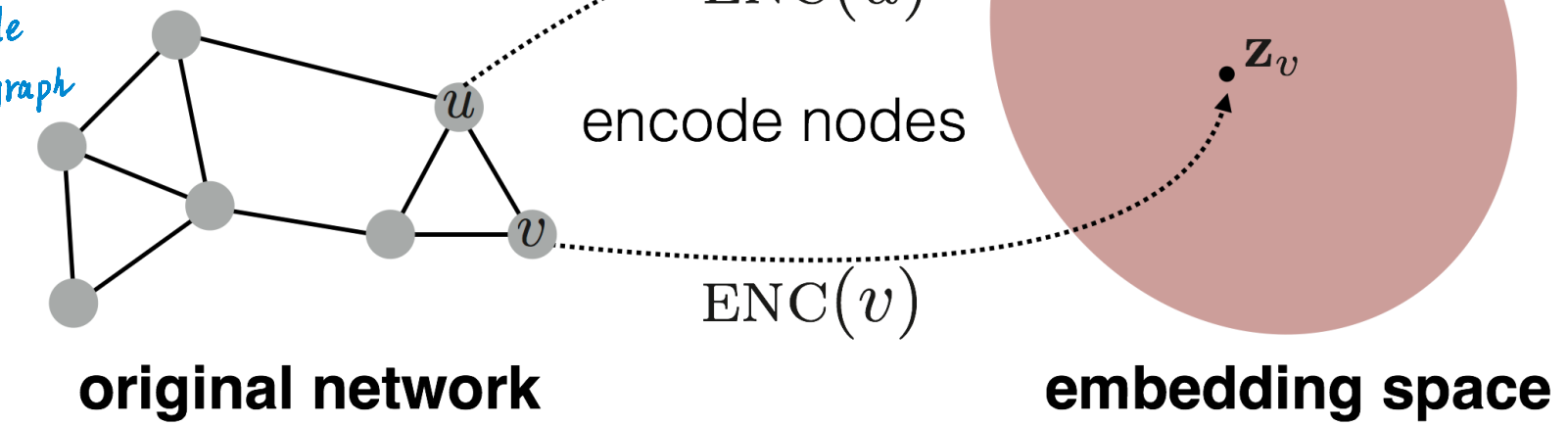




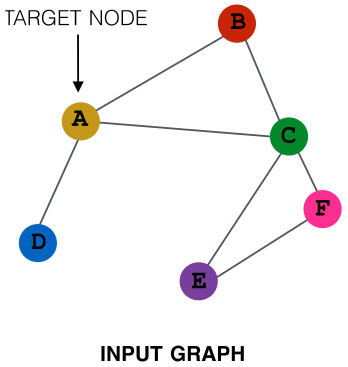
Background: Graph Representation Learning

- Given one or more input graphs, we use a (deep) encoder to map nodes to high-dimensional embedding space $u \rightarrow z_u$
- Objectives
  - **Node** property prediction eg. use z_u to do multi-class classification
  - **Link** / **edge** property prediction use z_u and z_v to predict if there is link between node u and v
  - **Graph** / **subgraph** prediction

average over all node embeddings to one graph embedding → make prediction



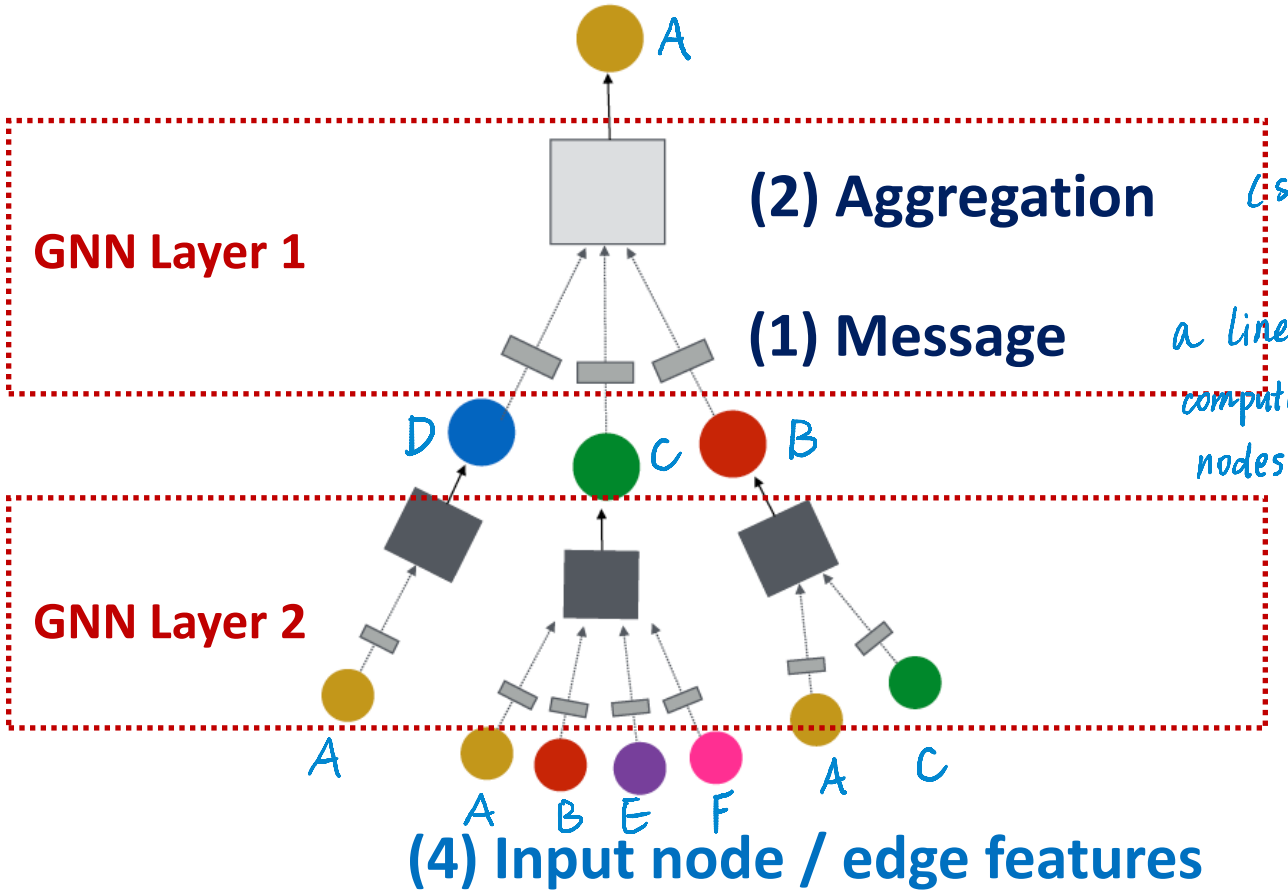
Recap: Graph Neural Networks (GNNs)



INPUT GRAPH

Compute node embedding for node A

(5) Learning objective loss func for node classification / edge prediction

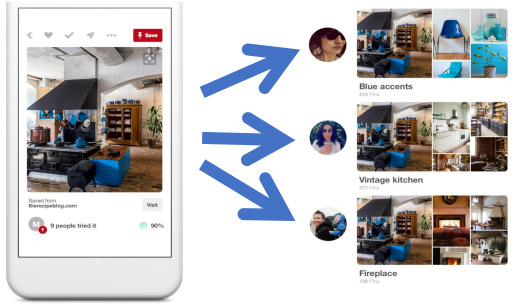


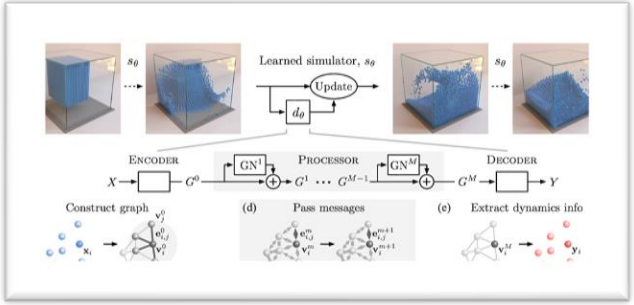
(3) Layer connectivity

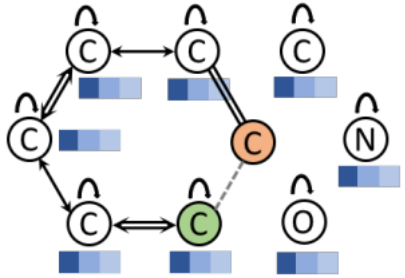
(4) Input node / edge features

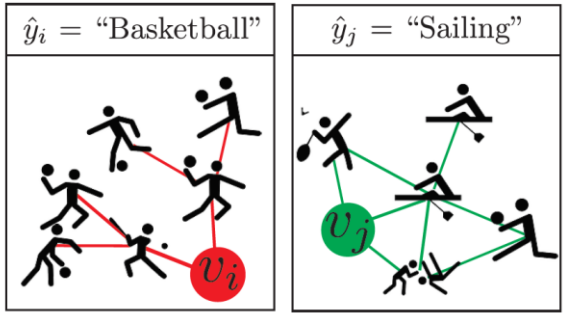
GNN Applications

- **Social Networks** → **Recommender System**
- **Natural Science** → **Physical Simulation**
- **Medicine** → **Drug Side-effects**, **Molecule Generation**
- **Explainability of GNNs**



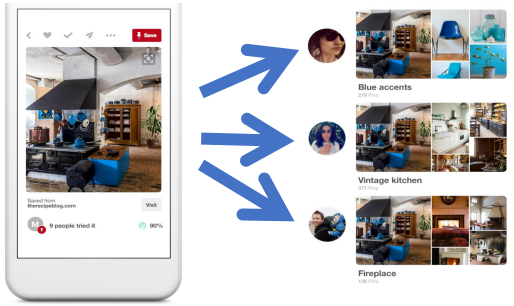






GNN Applications

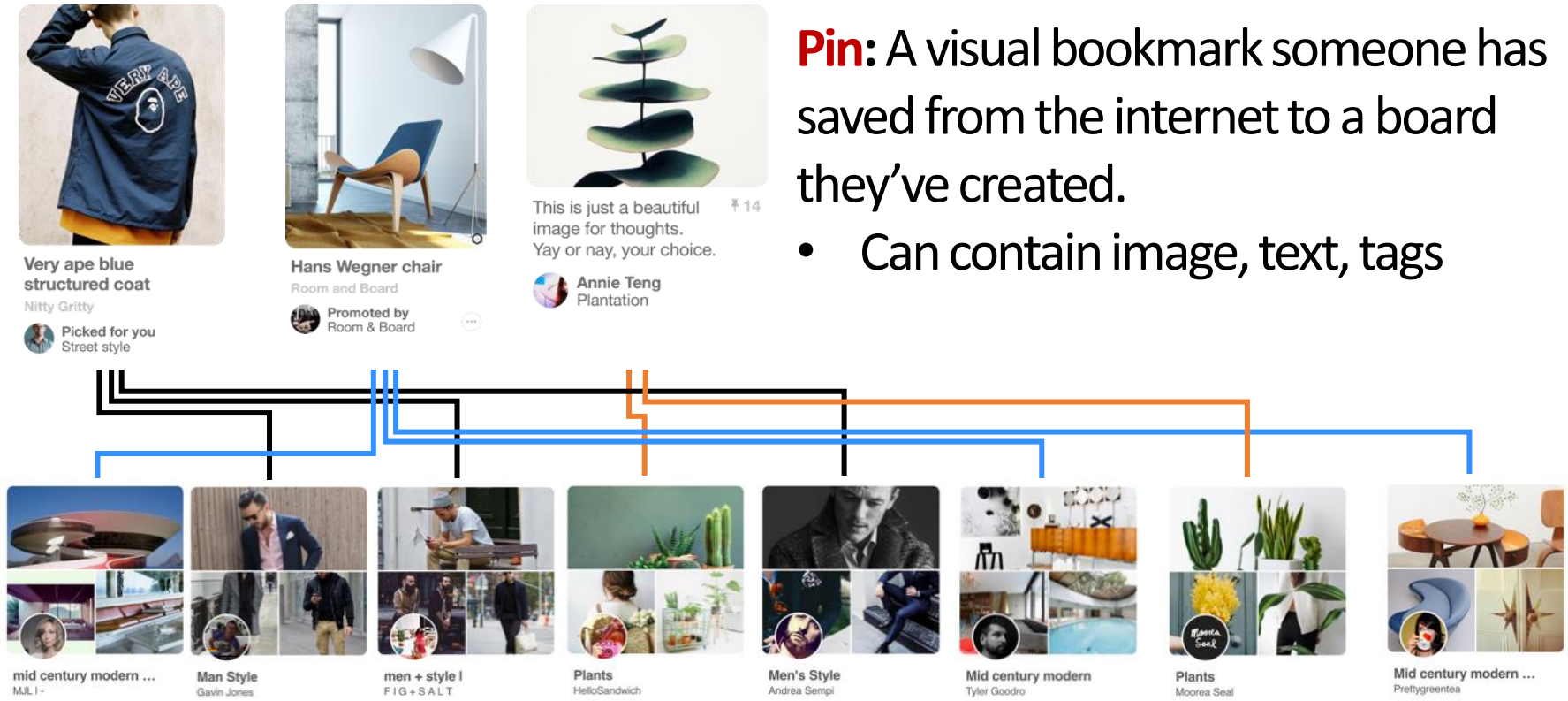
- **Social Networks** → **Recommender System**



- Natural Science
- Medicine
- Explainability of GNNs

Pinterest Recommender System

Human curated collection of pins



Pin: A visual bookmark someone has saved from the internet to a board they've created.

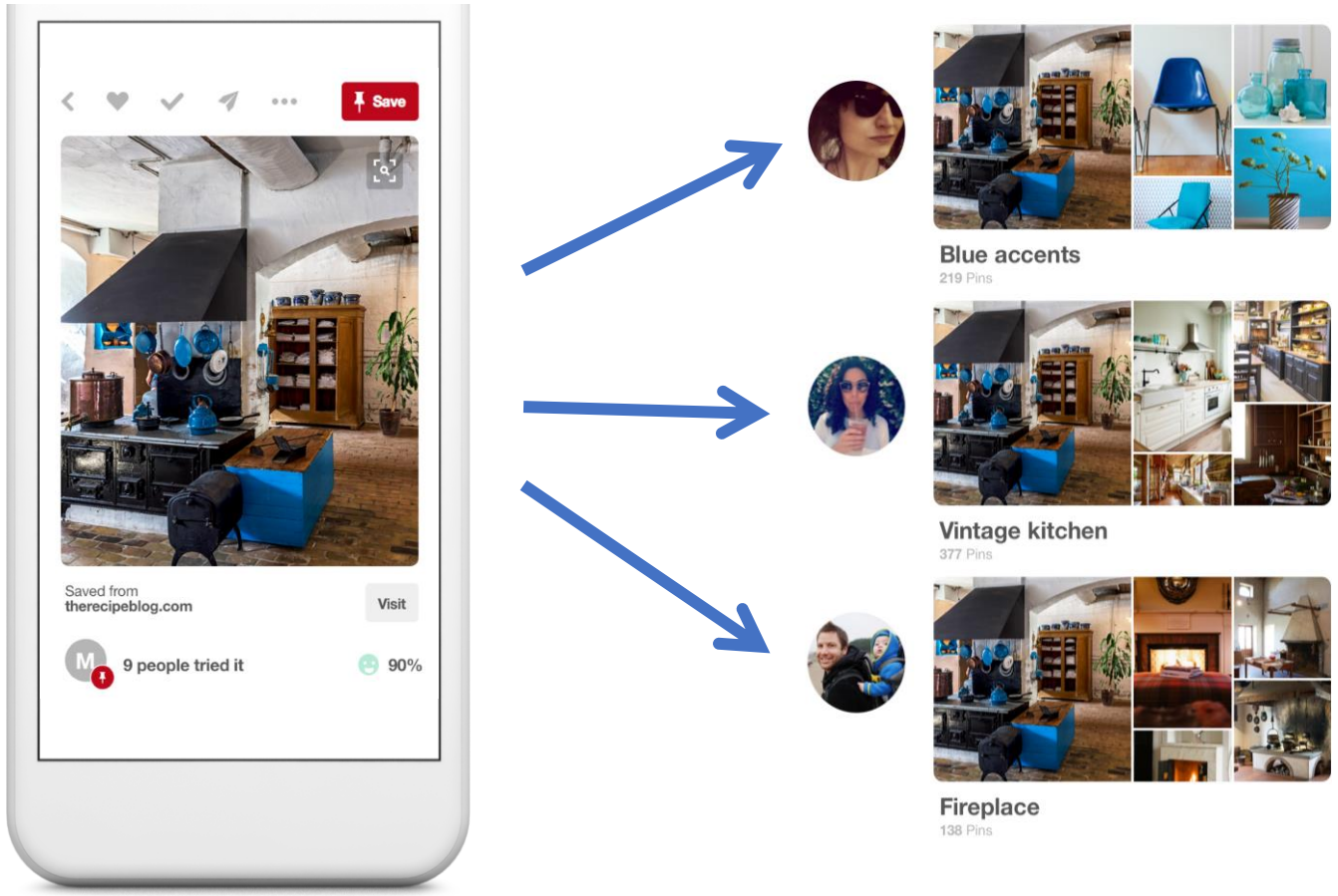
- Can contain image, text, tags

Board: A collection of ideas (pins having something in common)

Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD 2018

Pinterest Recommender System

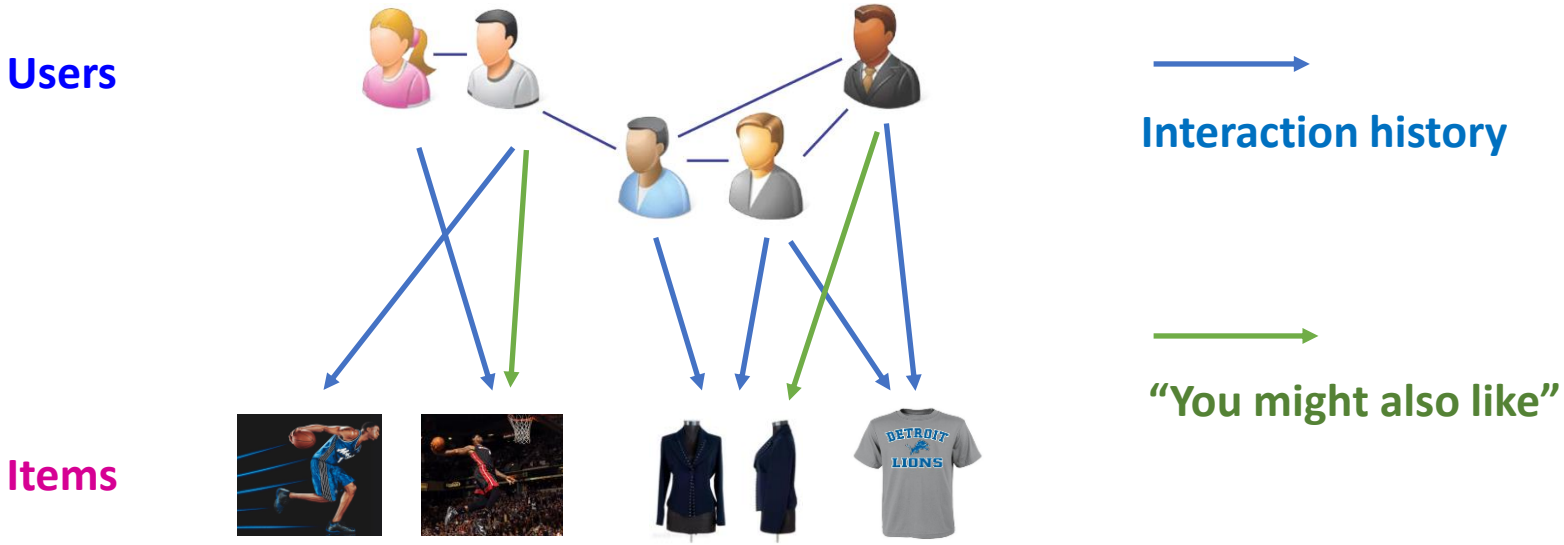
- Social network for idea collections
- 300M users
- 4B+ pins, 2B+ boards



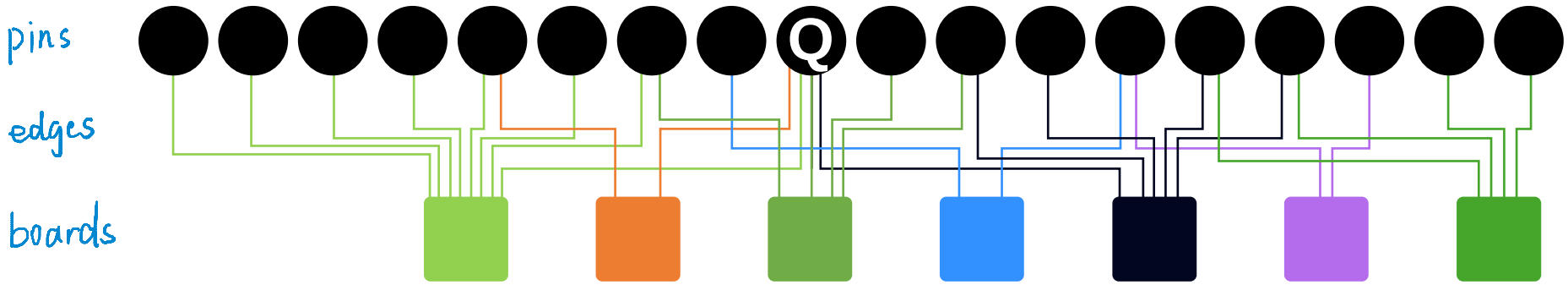
Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD 2018

PinSage: Web-scale Recommender System

- Users are related by social network
- Users interact with items (watch movie, buy merchandise, listen to music)
- Predict future interactions from history



Pinterest Graph

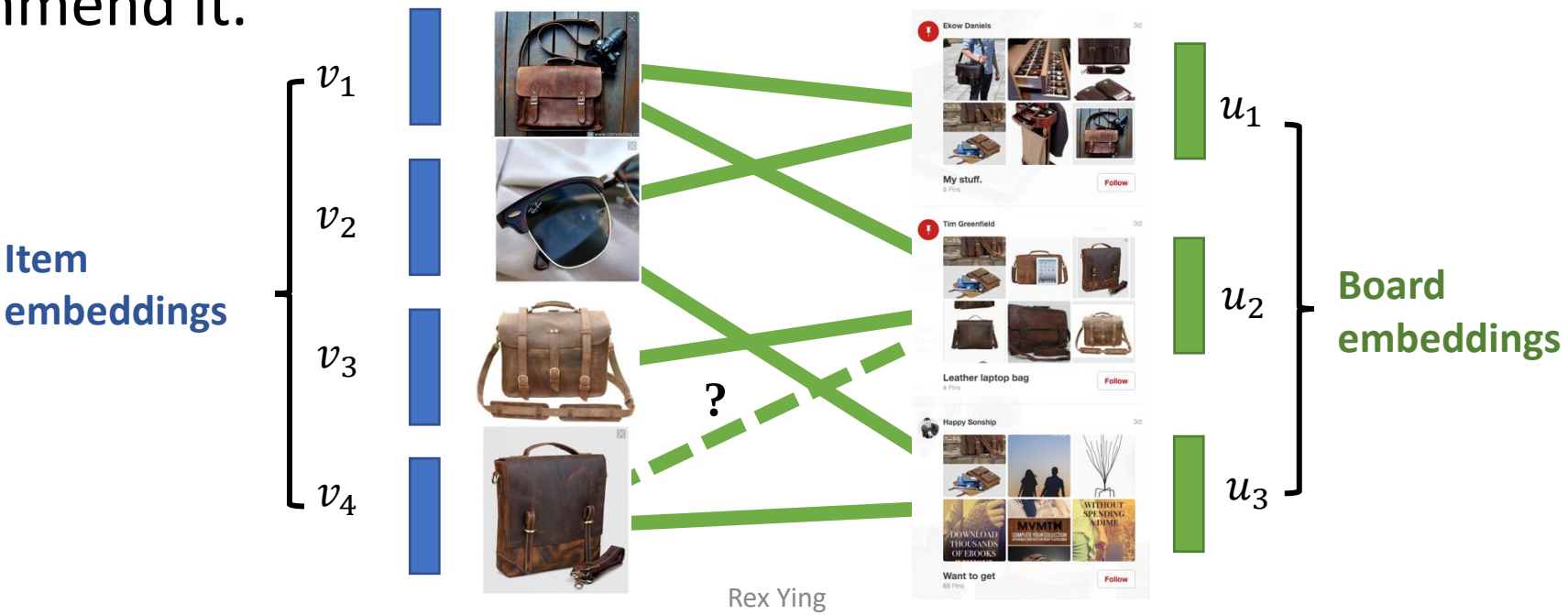


Graph: 2B pins, 1B boards, 20B edges

- **Graph is dynamic:** Need to apply to new nodes without model retraining
- **Rich node features:** Content, images

Recommend by embeddings

- Learn embeddings for **items** and **boards** (users)
- **Query**: which item to recommend to the board with embedding u_2 ?
- **Answer**: find the closest embedding (v_4) by nearest neighbor. in item embeddings Recommend it.



Why is it Hard?

scalability

How to scale the training as well as inference of node embeddings to graphs with billions of nodes and tens of billions of edges?

- Scaling up is difficult:
  - Existing collaborative filtering and distributed node embedding methods are inefficient when the underlying graph has billions of nodes and whose structure is constantly evolving

PinSage Overview

- **PinSage** graph convolutional network:
  - **Goal:** Generate embeddings for nodes (e.g., Pins/images) in a web-scale Pinterest graph containing billions of objects
  - **Key Idea:** Borrow information from nearby nodes
    - E.g., bed rail Pin might look like a garden fence, but gates and beds are rarely adjacent in the graph

gates pin and beds pin are in different boards though they look similar in CV so we need graph structure to model pins relationship and boards

    - Pin embeddings are essential to various tasks like recommendation of Pins, classification, clustering, ranking
      - Services like “Related Pins”, “Search”, “Shopping”, “Ads”

PinSage Pipeline

1. **Collect** billions of training pairs from logs.
   - **Positive pair**: a pin and a board that contains it
   - **Negative pair**: a random pair of pin and board
     - With high probability the pin is not pinned to the board

2. **Train GNN** to generate similar embeddings for training pairs.
3. **Inference**: generate embeddings for all pins.
4. **Nearest neighbor search** in embedding space to make recommendations.

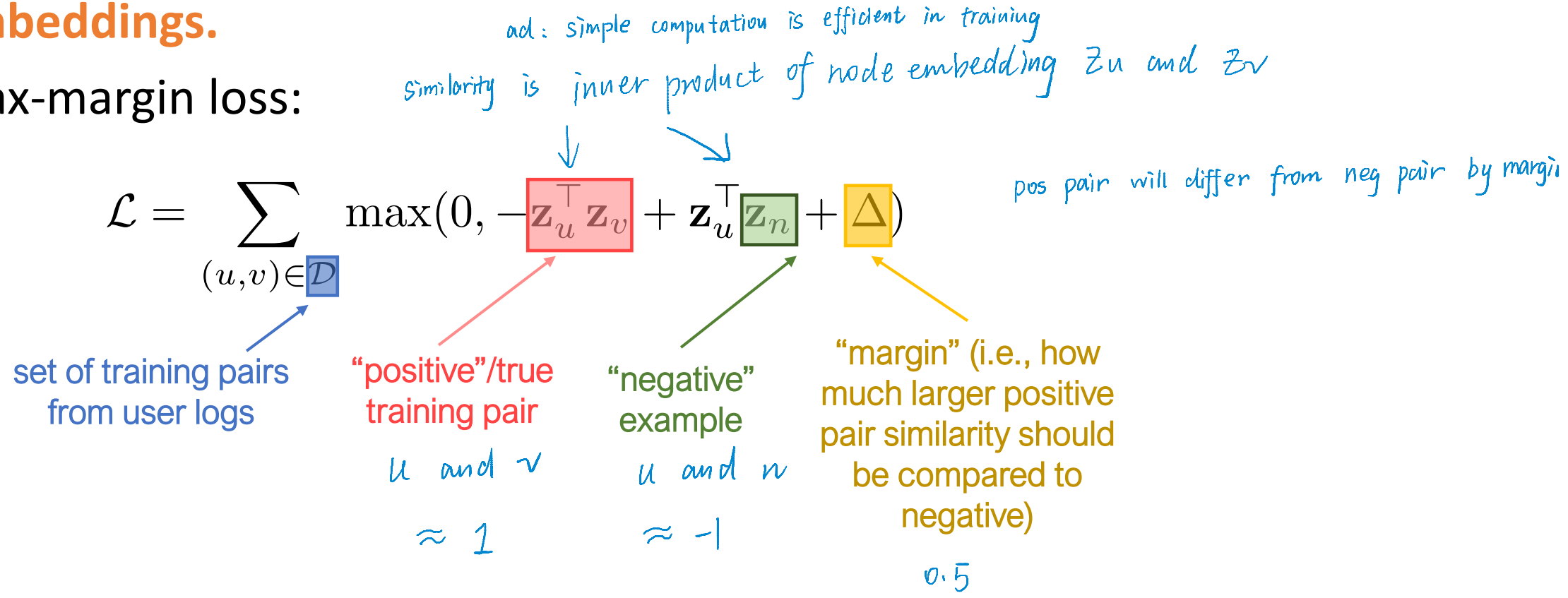
Neighborhood Sampling

Constructing convolutions via random walks

- Performing convolutions on full neighborhoods is infeasible: bc we can't fit the whole graph to GPU . we have to do mini-batch training , look at few nodes and its neighbors only pass selected neighbor nodes to GPU and compute node embeddings
  - How to select the set of neighbors of a node to convolve?
- **Personalized PageRank can help!** what are the important nodes (pins) for a node (board)
- **Importance pooling:** Define importance-based neighborhoods by simulating random walks and selecting the neighbors with the highest visit counts

Training Objective

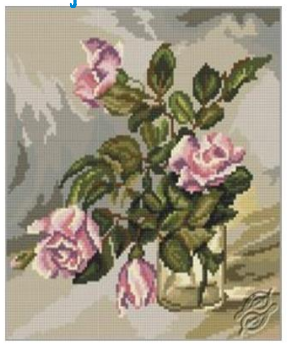
- Train so that **pins that are consecutively clicked have similar embeddings.**
- Max-margin loss:



Negative Sampling

- **Hard negative sampling**

刺绣 embroidery
十字绣 cross stitch


Query

flower 刺绣


Positive Example

flower 刺绣


Random Negative

帽子 real


Hard Negative

bird 刺绣

same texture 刺绣
but not same object (bird)

(confusing example)

Harder to distinguish from positive

- Use personalized PageRank (PPR)
  - Use nodes that have PPR score ranked at 1000-5000 as hard negatives
  - Have something in common, but not too similar

Negative Sampling

- **Hard negative sampling**

- **Curriculum training** on hard negatives
  - provide harder and harder examples over time

After training

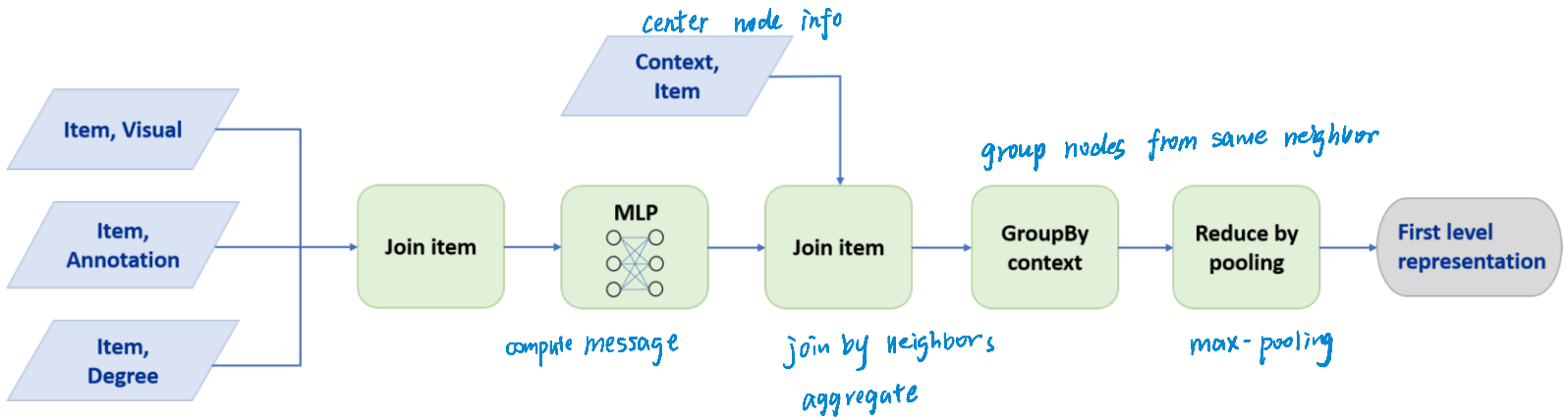
Efficient Inference

Efficient inference via **MapReduce**

- Bottom-up aggregation of node embeddings lends itself to MapReduce
  - Decompose each aggregation step across all nodes into three operations in MapReduce, i.e., *map*, *join*, and *reduce*
- Avoid repeated computation

bc we train using mini-batch (a subset of nodes)
so there will be overlap between trained neighbor nodes

For every layer:



Evaluation

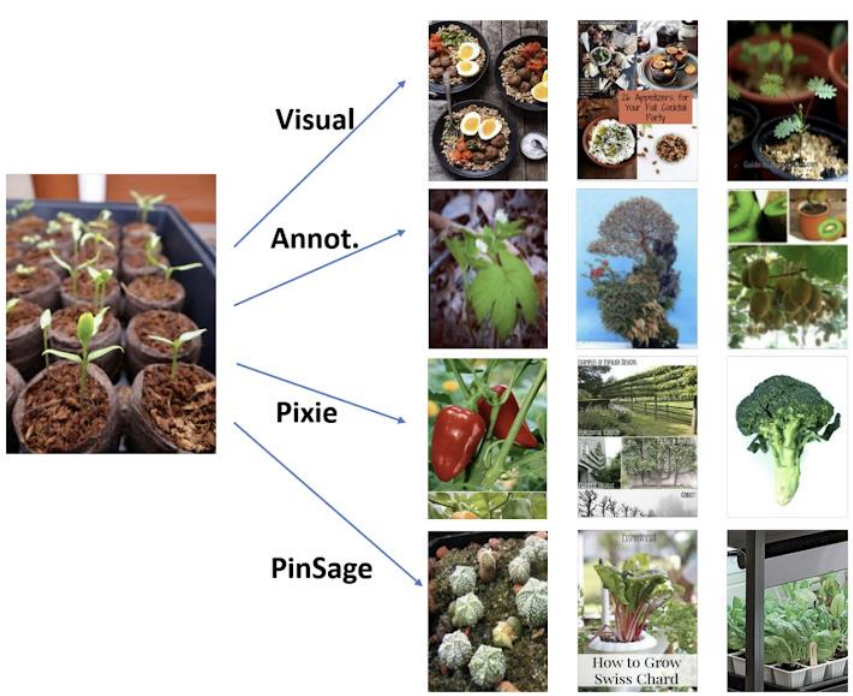
- Baselines: (pretrained)
  - **Visual**: VGG visual embeddings for recommendations
  - **Annotation**: Word2vec embeddings
  - **Combined**: Concatenate embeddings:
    - Uses exact same data and loss function as PinSage

recommend based on → near neighbor

Method	Hit-rate	MRR
Visual	17%	0.23
Annotation	14%	0.19
Combined	27%	0.37
max-pooling	39%	0.37
mean-pooling	41%	0.51
mean-pooling-xent	29%	0.35
mean-pooling-hard	46%	0.56
PinSage	**67%**	**0.59**

PinSage gives 150% improvement in hit rate and 60% improvement in MRR over the best baseline

Evaluation



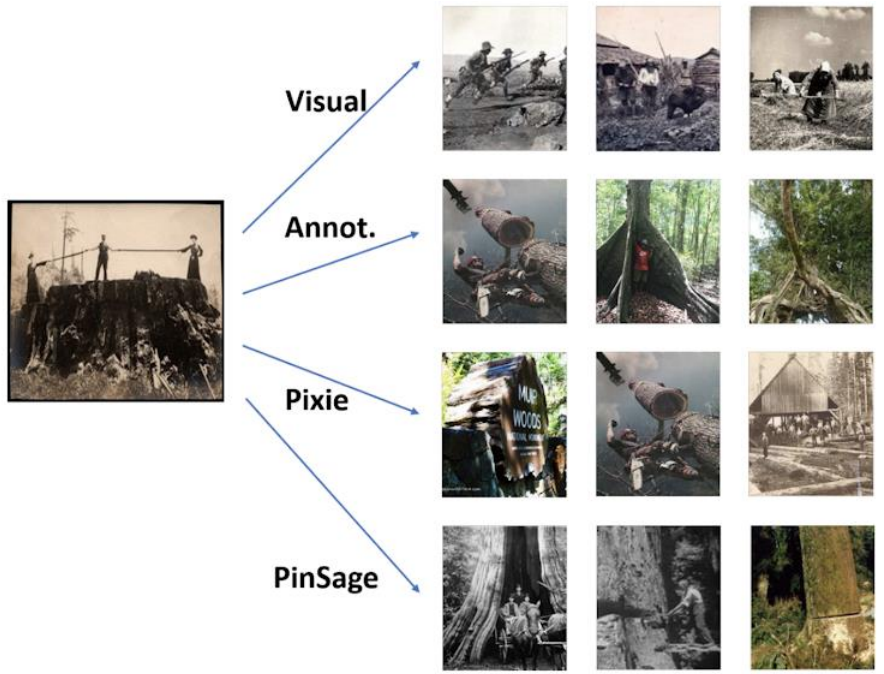


Image information

Text information

Graph information

Combined!

image + text + graph info
visual + context + category similar

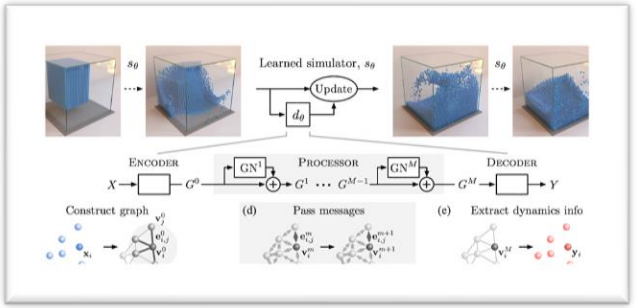
Pixie is a purely graph-based method that uses biased random walks to generate ranking scores by simulating random walks starting at query Pin. Items with top scores are retrieved as recommendations [Eksombatchai et al., 2018]

Summary

- Many online platforms can be modeled as graphs, where nodes are users, items and edges represent **interactions**
  - View, purchase, review, follow, likes
- We use **link prediction objective** to train a GNN-based recommender systems
  - Efficient training and inference on a billion-scale graph
- The learned graph embedding captures image/text feature as well as graph structure (category feature)
- The learned embedding is useful in many downstream tasks beyond recommendation!

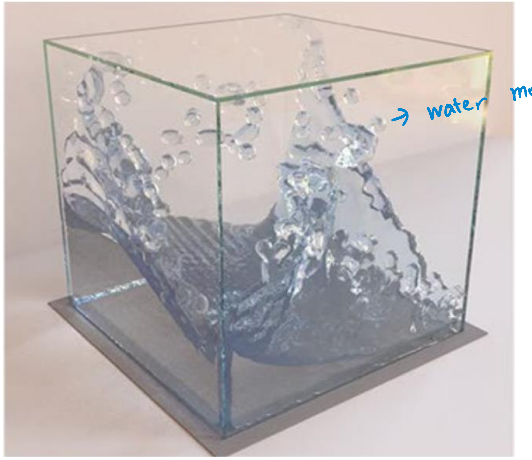
GNN Applications

- Social Networks
- **Natural Science** → **Physical Simulation**
- Medicine
- Explainability of GNNs

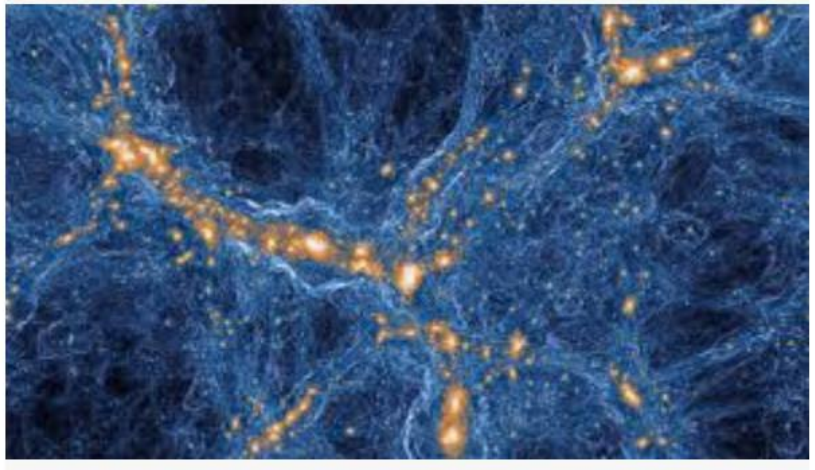


Simulations in Science and Engineering

1. Particle-particle interactions:



Water simulation



Galaxy formation

node : stars
edge : gravity force

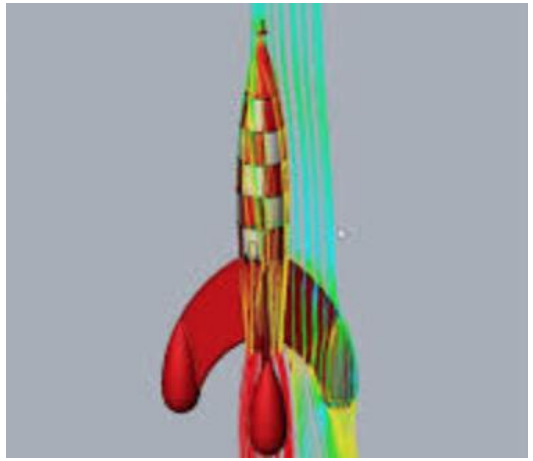
We construct graphs with particles as nodes and interactions as edges

Simulations in Science and Engineering

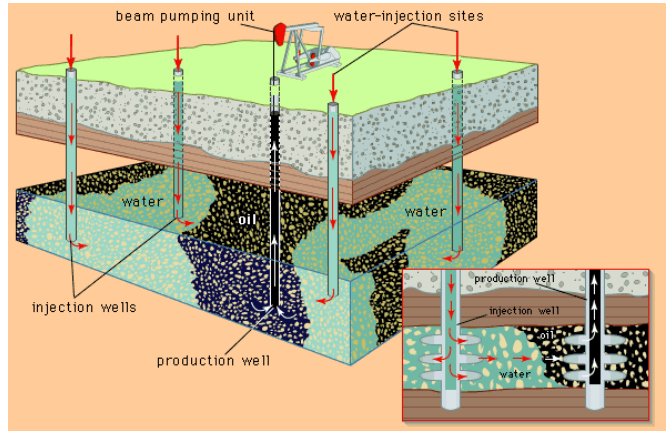
2. PDEs (on grid or mesh)

Partial Differentiation Equation

Weather prediction



Aerodynamics



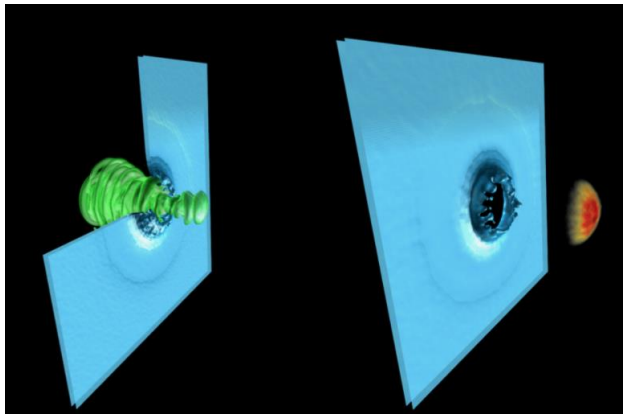
oil Reservoir simulation

Graph based on grid / mesh structure

alternative 3-D U-net

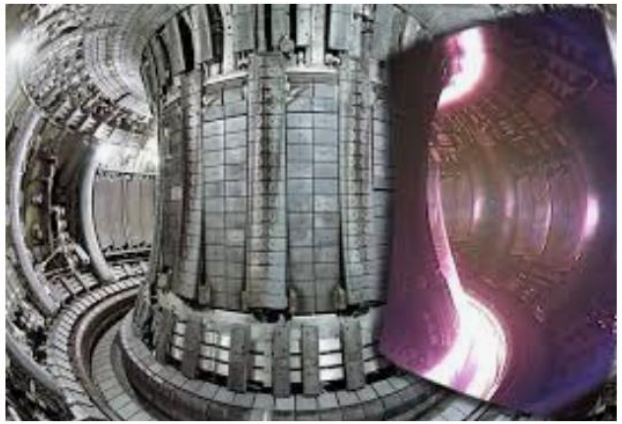
Simulations in Science and Engineering

3. Particle-in-Cell (involves both grid and particles)



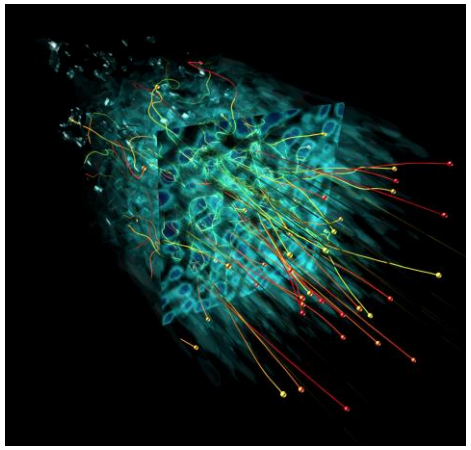
Laser-plasma particle acceleration

激光等离子加速



Fusion

核聚变



Cosmic-ray acceleration

宇宙射线加速

Graph combining grid / mesh with particles!

Why Learning Simulation

Engineered simulators:

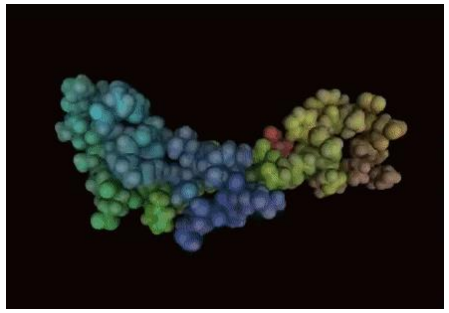
1. Substantial effort to build
2. Substantial resources to run
3. Only as accurate as the designer
4. Not always suitable for solving inverse problems

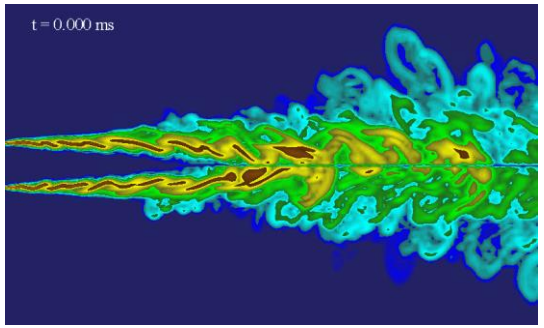
eg inverse problem: How to tune params to stablize particles for los?

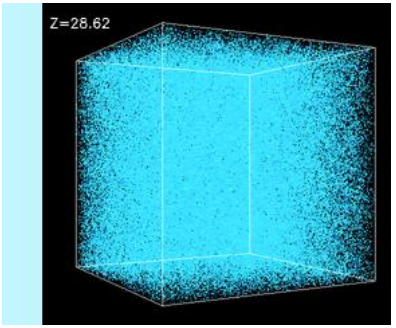
Learned simulators:

1. **Shared** architectures bc generalizability
2. Can be directly optimized for **efficiency**
3. Can be as accurate as the available **data**
4. Gradient-based search for **control**

use gradient to backtrace param





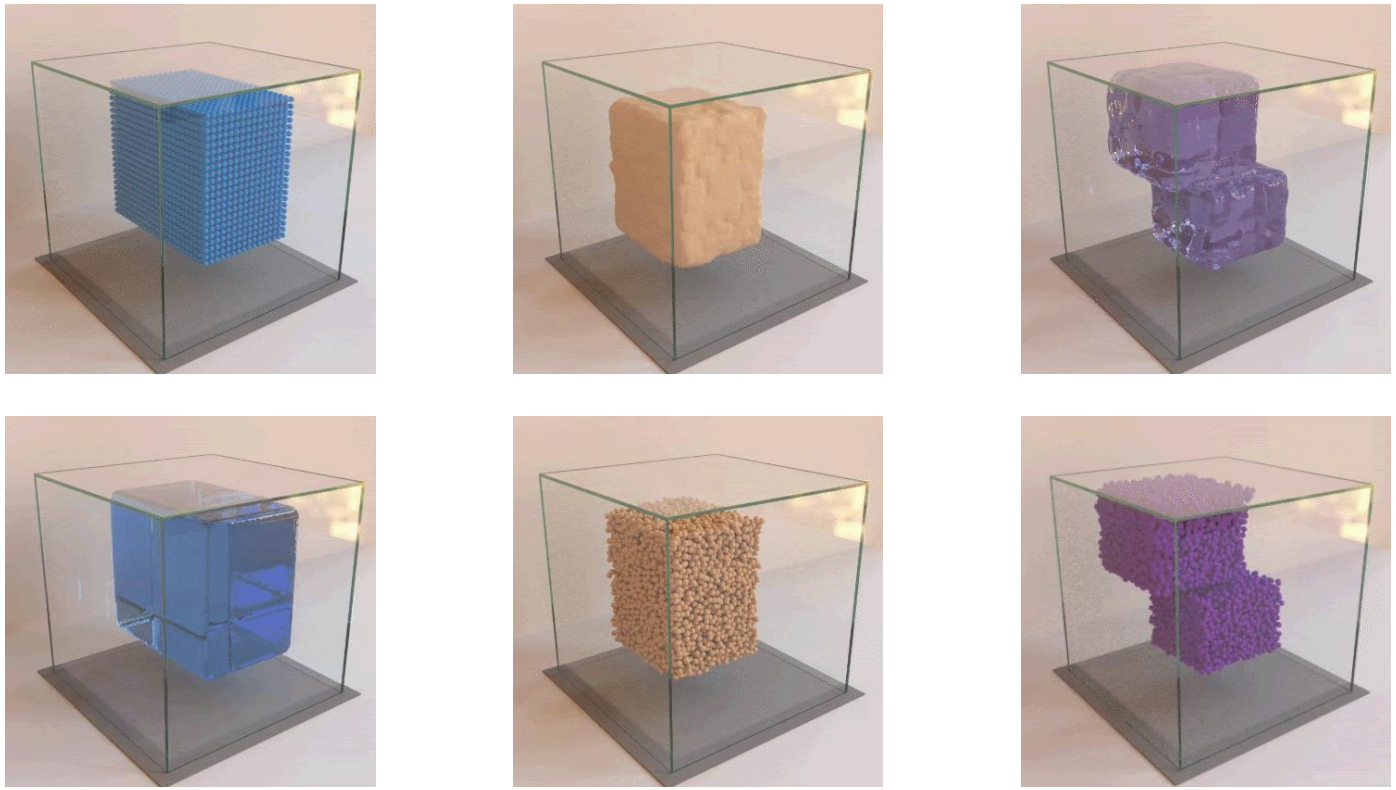


Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Dynamic predictions – Learning Simulation

initial state → future state

- Simulating complex fluids and other materials:

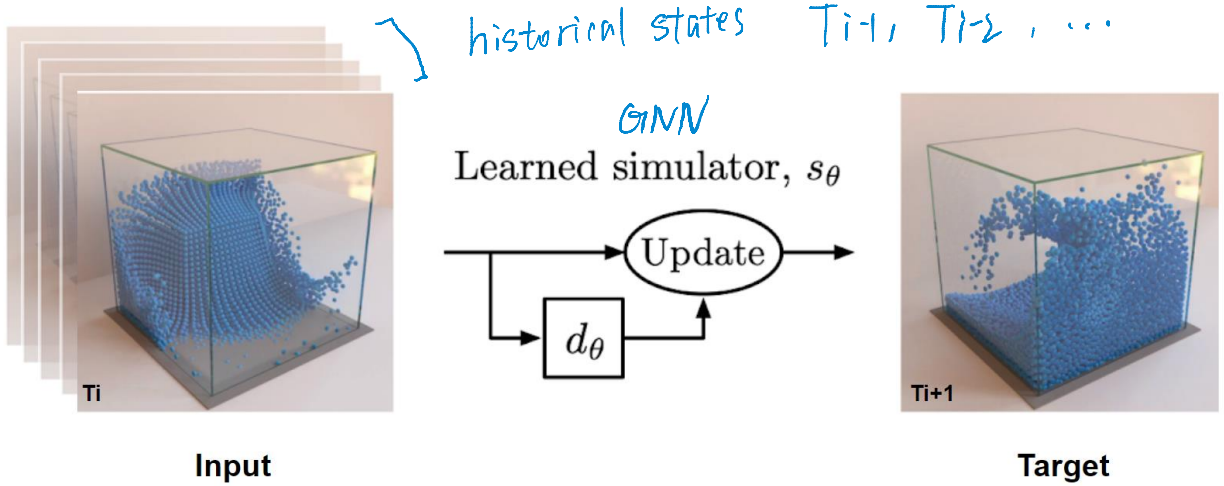


Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Use historical particle information

- During training, we are given **particle properties** (position, velocity …) for a time period $0 - T$
  - Equal and discrete time interval
- We **sample** batches of particle states at consecutive time steps
- Historical particle information

Multiple instances per minibatch

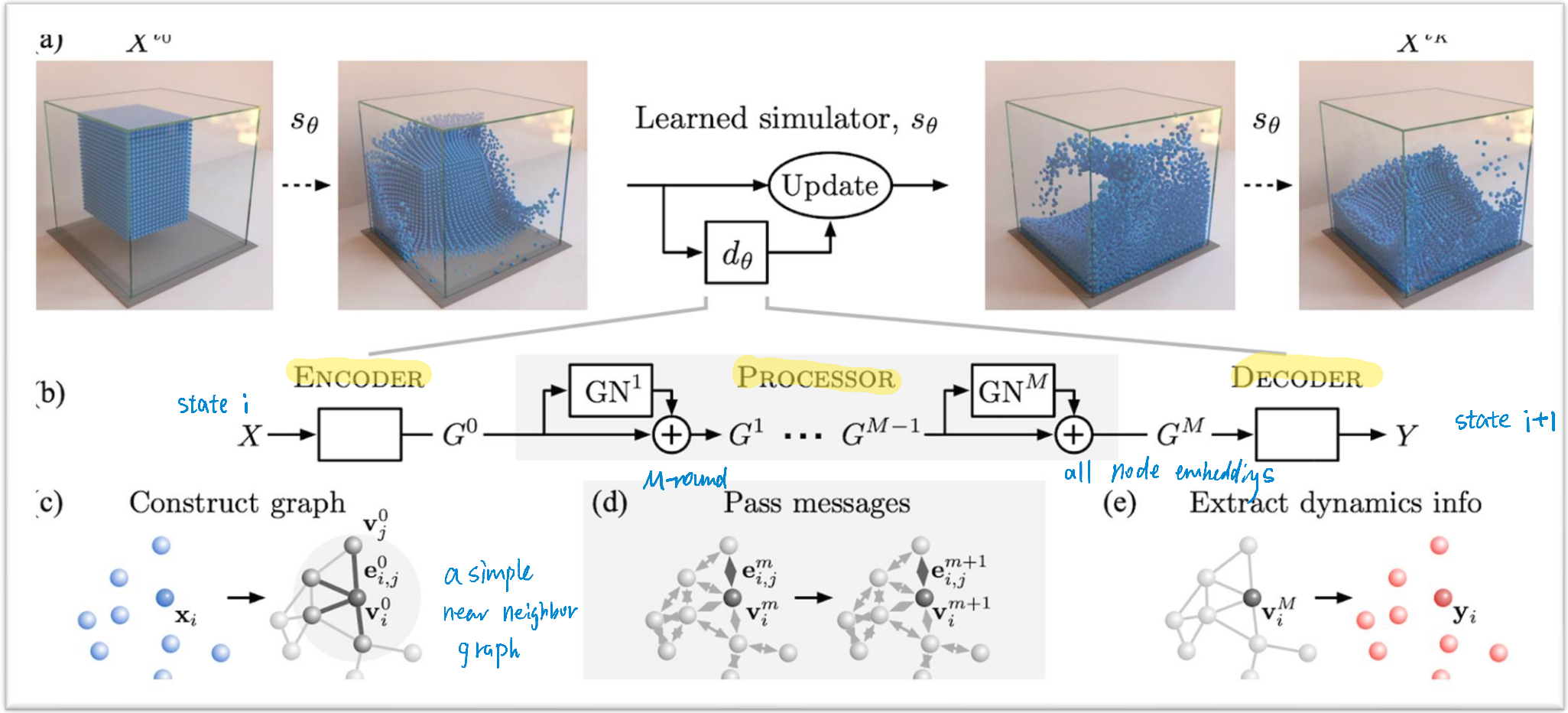


Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Simulation Architecture

training : state t_i and t_{i+1} pairs

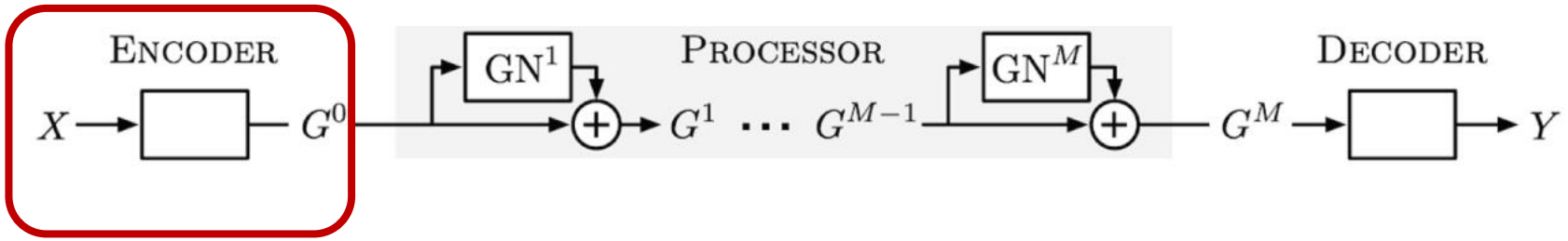
inference : input state t_0, can predict state at any time point



Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Graph Network Simulator (GNS) Model

Encoder



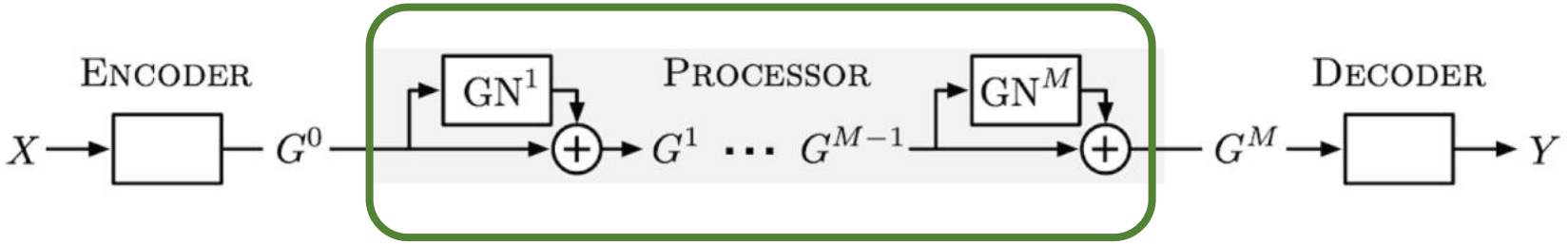
- Node input features:
  - Position, previous 5 velocities, particle type (water/sand)
- Edge input features: displacements
  - Embed features with MLP
- Construct neighbourhood graph
  - K-nearest neighbor (kNN) edges

choice depends on inductive bias for specific problem

Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Graph Network Simulator (GNS) Model

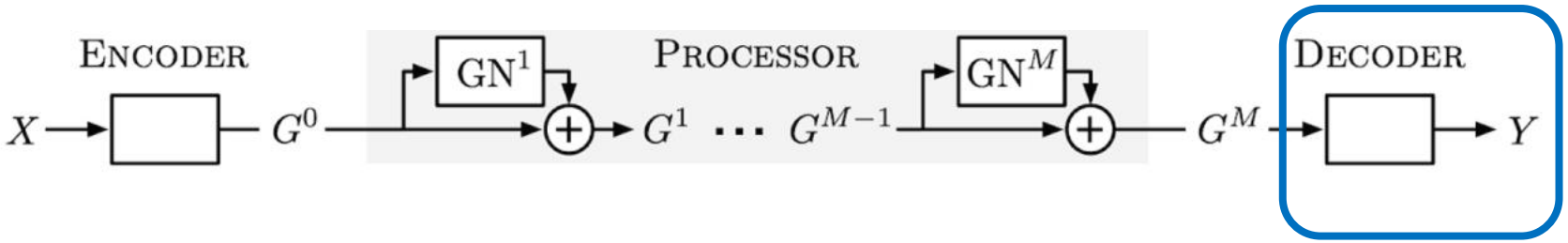
- **Processor**



- Message-passing layers (x10) on the kNN graph
  - Concatenate node and edge features, and compute message function through multi-layer perceptron (MLP)
  - Outputs embeddings for each particle
    - Used to predict next step dynamics

Graph Network Simulator (GNS) Model

Decoder



- Decode acceleration 加下标和个当

- Feed into Euler integrator to obtain position and velocity

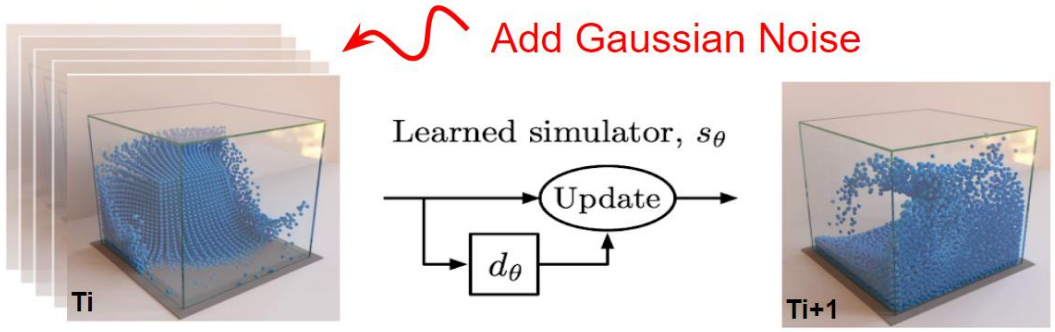
- Sum over L2 Loss for all particles in each pair

Learning to Simulate Complex Physics with Graph Networks, ICML 2020

Graph Network Simulator (GNS) Model

- Training time: One-step minibatch training

t_i – t_{i+1} pair



Gaussian

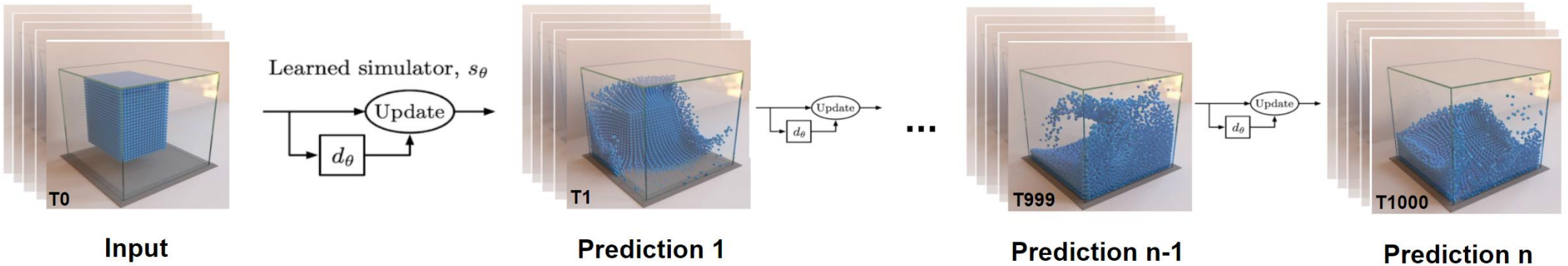
Use noise at training to prevent error accumulation at evaluation

make GNN robust

variant: use Recurrent GNN

- Test time: 1000s of steps

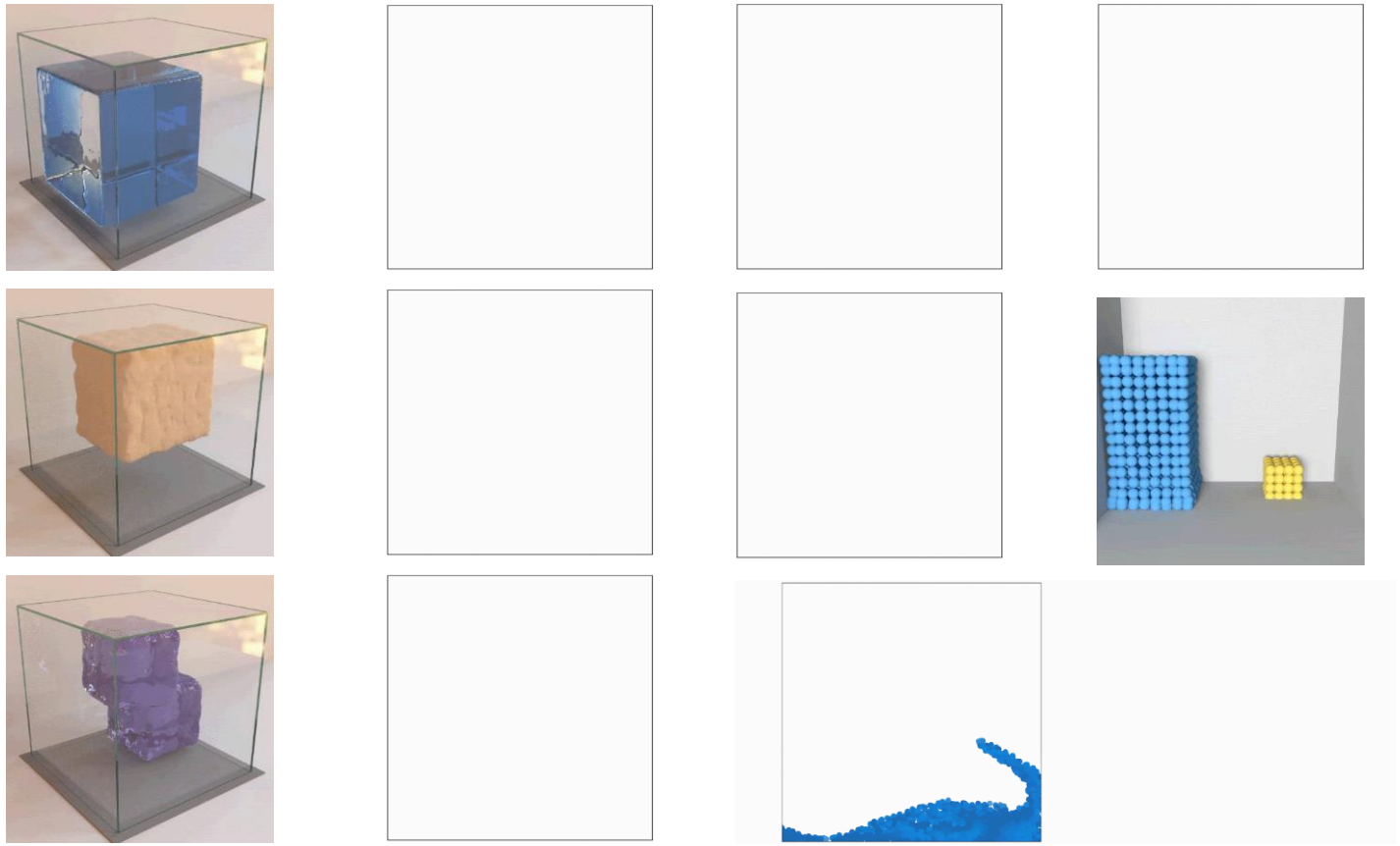
prediction serve as input



Generalization

- **Same model and hyperparameters across datasets**

local interaction is key to simulation



Learning to Simulate Complex Physics with Graph Networks, ICML 2020

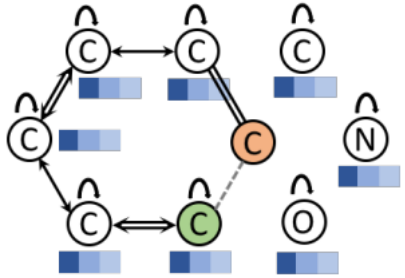
Summary

- In simulation, we are given an initial condition, and use the model to make predictions of the **evolution of the system** over time.
- We model the systems through **particles** (nodes) and **interactions** between particles (edges).
- The model **generalizes** to many different scenarios since the model learns the underlying rules of physical interactions.
- Open question
  - more complex interactions
  - more efficient simulation of large systems

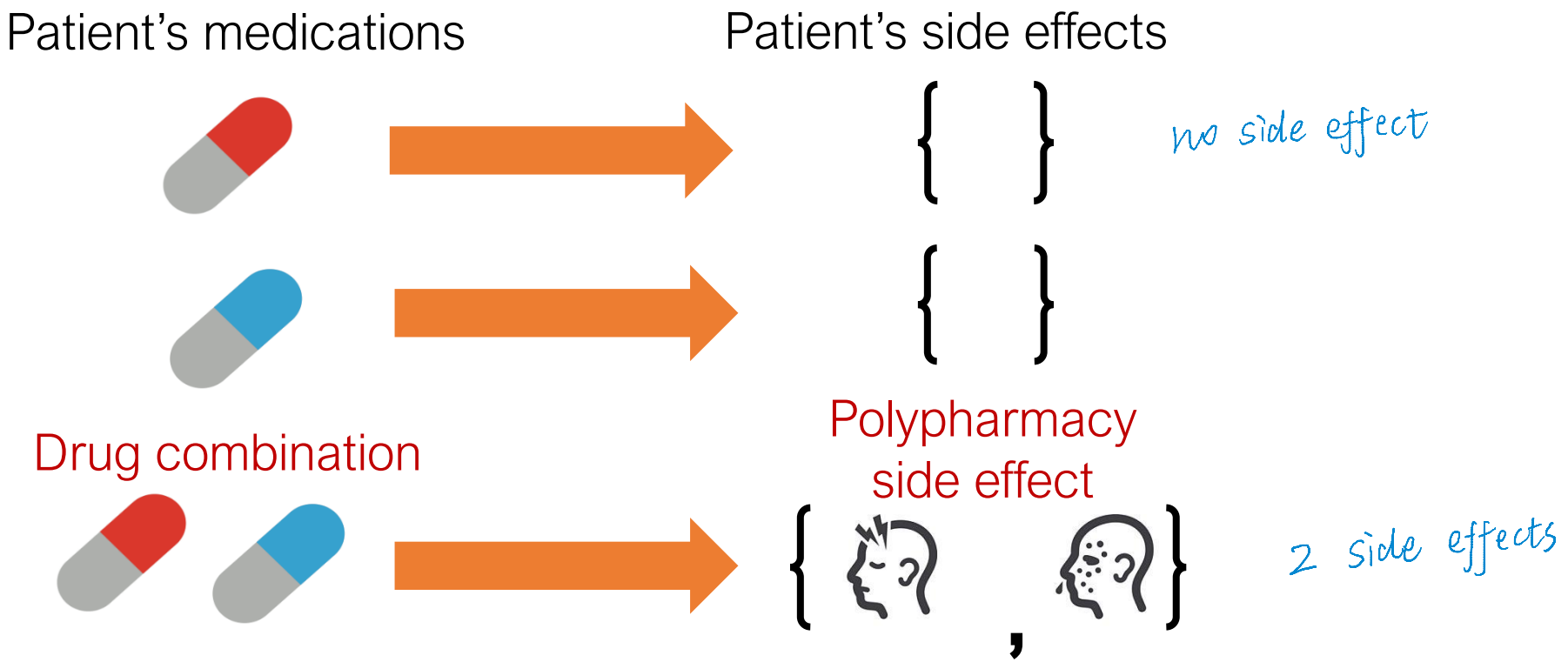
GNN Applications

- Social Networks
- Natural Science
- **Medicine**
- Explainability of GNNs

Drug Side-effects
Molecule Generation



Polypharmacy Side Effects



Polypharmacy: use multiple drugs for a disease

Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

Polypharmacy Side Effects

- Polypharmacy is common to treat complex diseases and co-existing conditions
- High risk of side effects due to interactions
- **15%** of the U.S. population affected
- Annual costs exceed **$177 billion**
- Difficult to identify manually:
  - Rare, occur only in a subset of patients
  - Not observed in clinical testing

Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

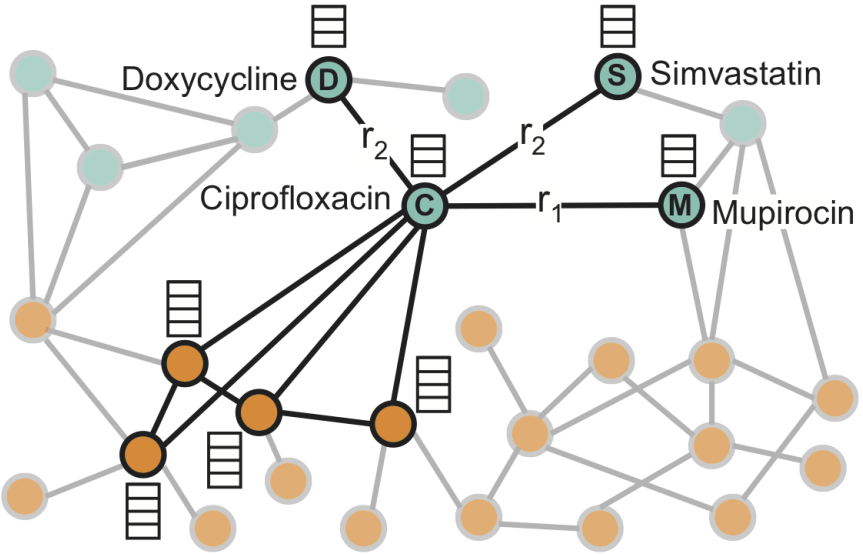
Modeling Polypharmacy

- Systematic experimental screening of drug interactions is challenging
- Idea: Computationally screen/predict polypharmacy side effects
  - Use molecular, pharmacological and patient population data
  - Guide translational strategies for combination treatments in patients

Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

Data: Heterogeneous Graphs

- **Heterogeneous (multimodal) graphs:** graphs with different node types and/or edge types



2 node types

4 edge types

Drug Gene

Feature vector

r_1 Gastrointestinal bleed effect

Drug target interaction

r_2 Bradycardia effect

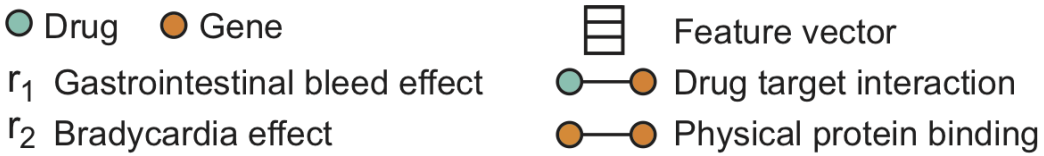
Physical protein binding

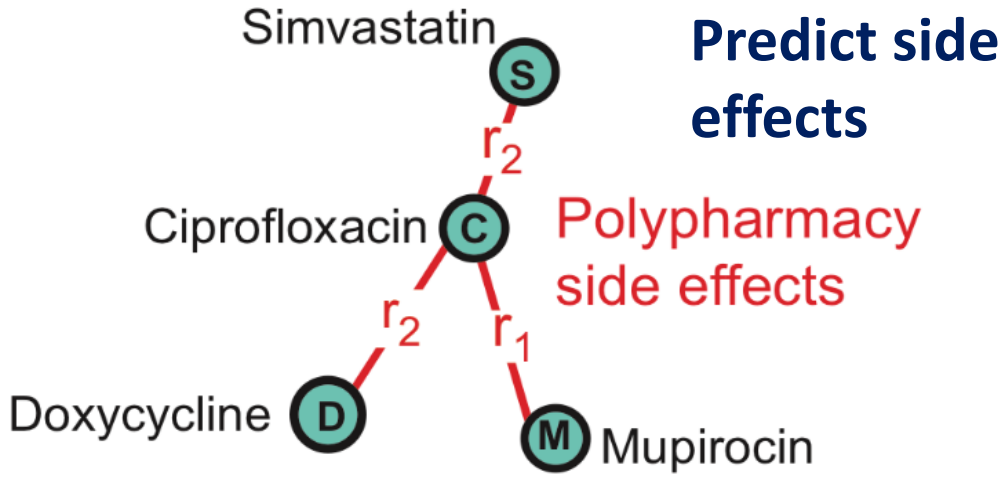
side effect

Task Description

Link prediction

- Predict labeled edges between drugs nodes
  - i.e., predict the likelihood that an edge (c, r_2, s) exists between drug nodes c and s
  - Meaning: Drug combination (c, s) leads to polypharmacy side effect r_2

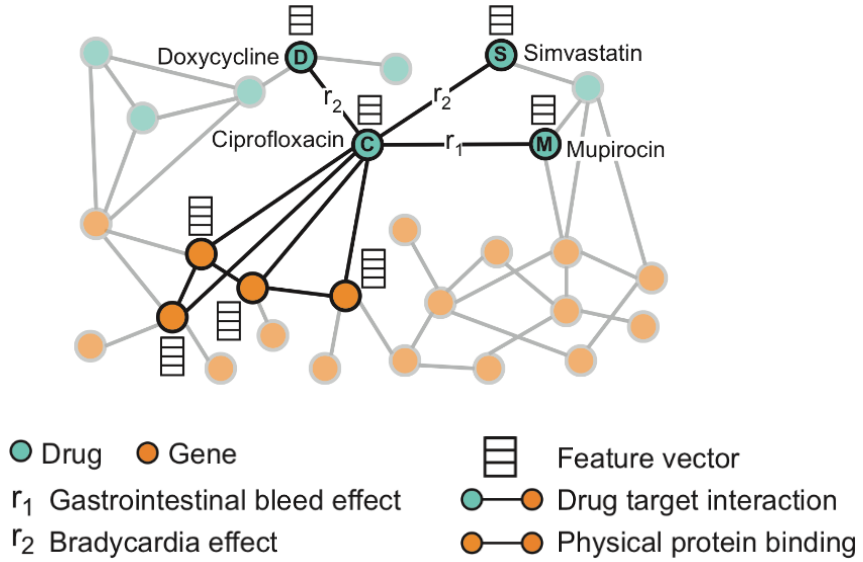




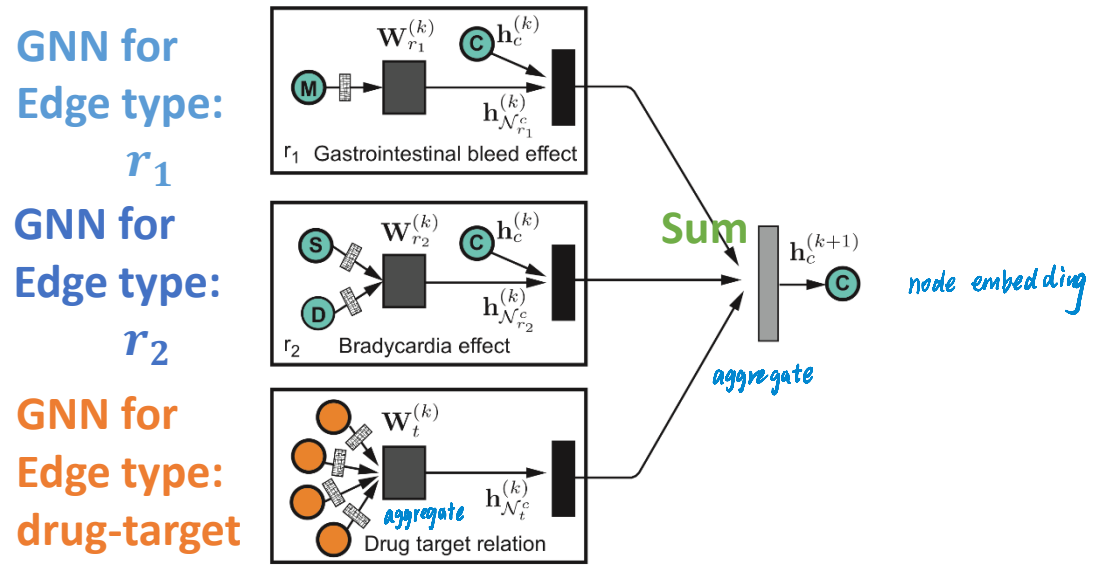
Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

Model: Heterogenous GNN

- **Key Insight:** Compute GNN messages from each edge type, then aggregate across different edge types
  - **Input:** heterogenous graph
  - **Output:** node embeddings



One layer of Heterogeneous GNN

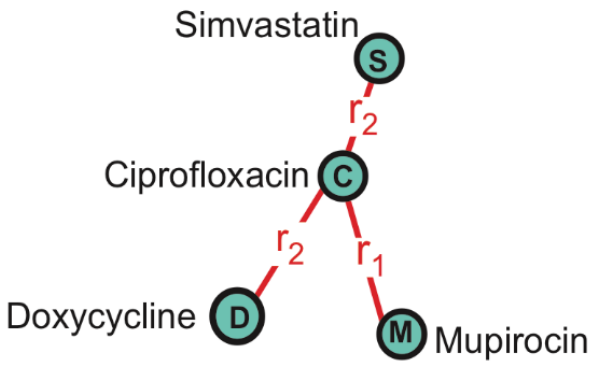


Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

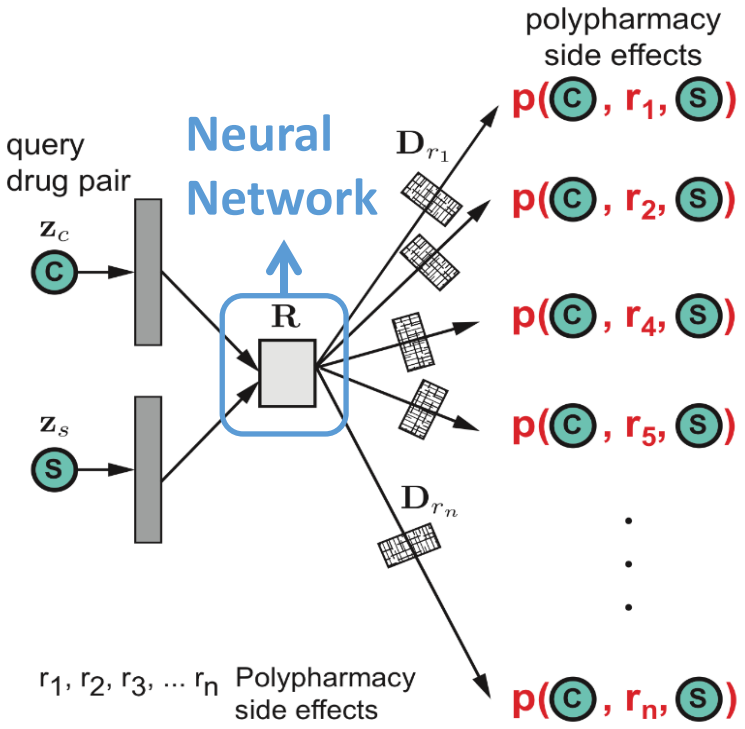
Making Edge Predictions

multi-class classification

- **Inference:** Use pair of computed node embeddings to make edge predictions
  - **Input:** Node embeddings of query drug pairs
  - **Output:** predicted edges



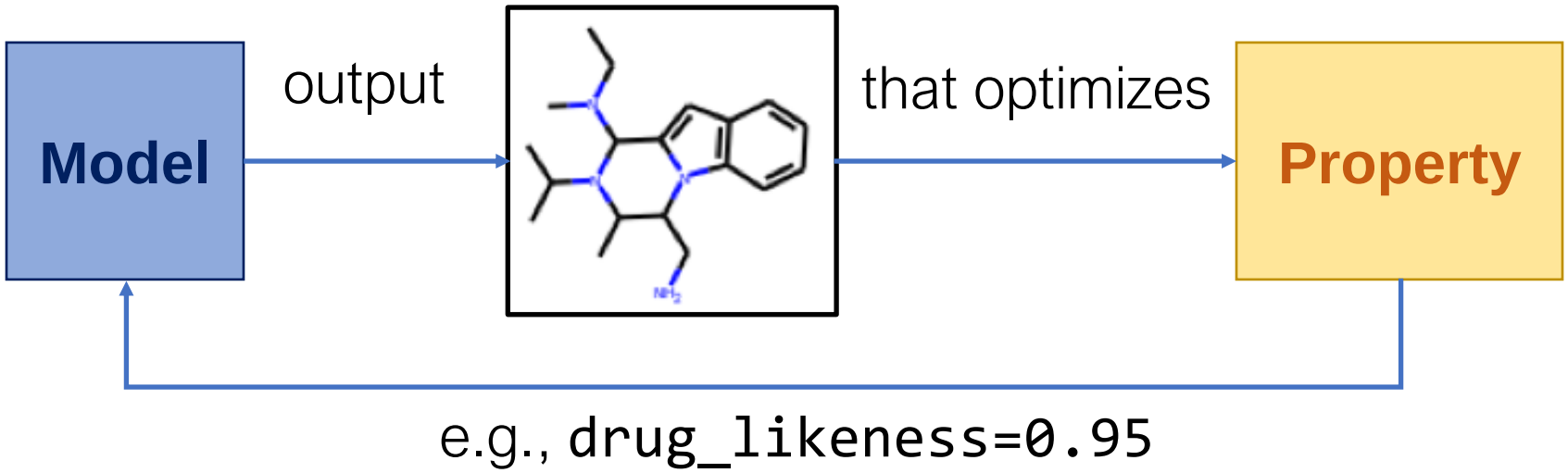
Predict possible edges with NN



Modeling Polypharmacy Side Effects with Graph Convolutional Networks, Bioinformatics 2018

Application: Drug Discovery

Question: Can we learn a model that can generate **valid** and **realistic** molecules with **optimized** property scores?



Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

Goal-Directed Graph Generation

The goal is to generate graphs that:

- **Optimize a given objective** (High scores)
  - e.g., drug-likeness
- **Obey underlying rules** (Valid)
  - e.g., chemical validity rules
- **Are learned from examples** (Realistic)
  - Imitating a molecule graph dataset

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

The Hard Part:

The goal is to generate graphs that:

- **Optimize a given objective** (High scores)

  Only available at the end of generation

- **Obey underlying rules** (Valid)

  Often not differentiable

- **Are learned from examples** (Realistic)

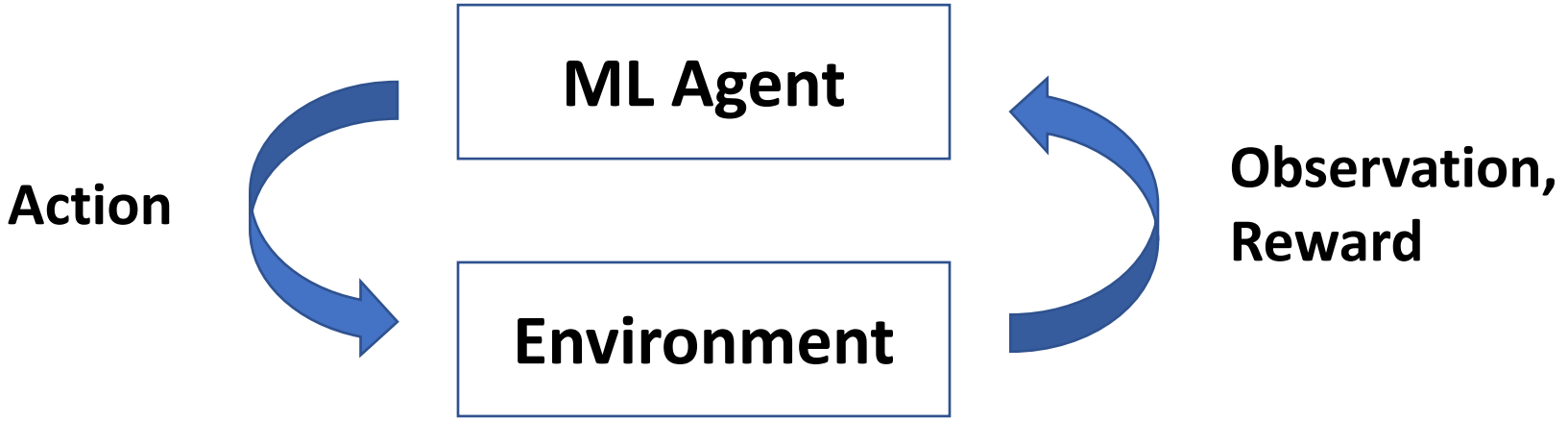
  Requires the model to learn the distribution of graph structures

generate atom by atom
bond by bond
no reward in intermediate step
only has reward in the end

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

Idea: Reinforcement Learning

- An ML agent **observes** the environment, takes an **action** to interact with the environment, and receives positive or negative **reward**
- The agent then **learns from this loop**
- **Key idea**: Agent can directly learn from environment, which is a **blackbox** to the agent



Solution: GCPN

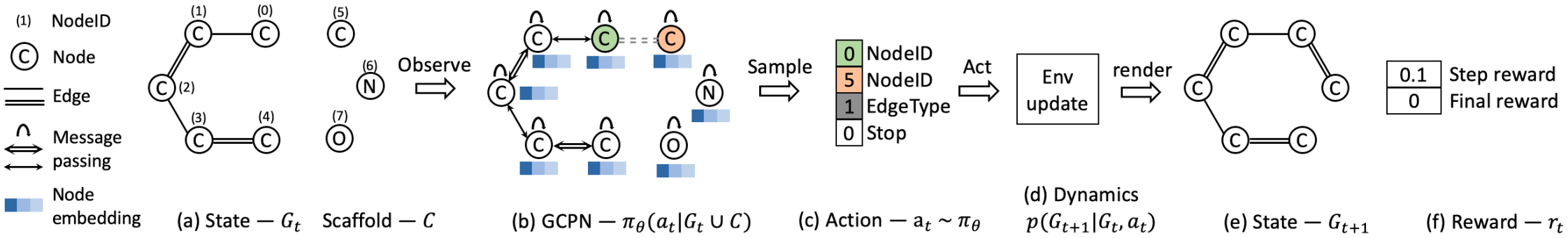
Graph Convolutional Policy Network (GCPN) combines **graph representation** and **reinforcement learning**

Key component of GCPN:

- **Graph Neural Network** captures graph structural information
- **Reinforcement learning** guides the generation towards the desired objectives
  - Generation step-by-step: every step we generate additional edges (bonds) and nodes (atoms) to attach to the generated molecule
  - Delayed reward: we only obtain the final metric (drug-likeness) at the end of the generation process
- **Supervised learning** imitates examples in given datasets

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

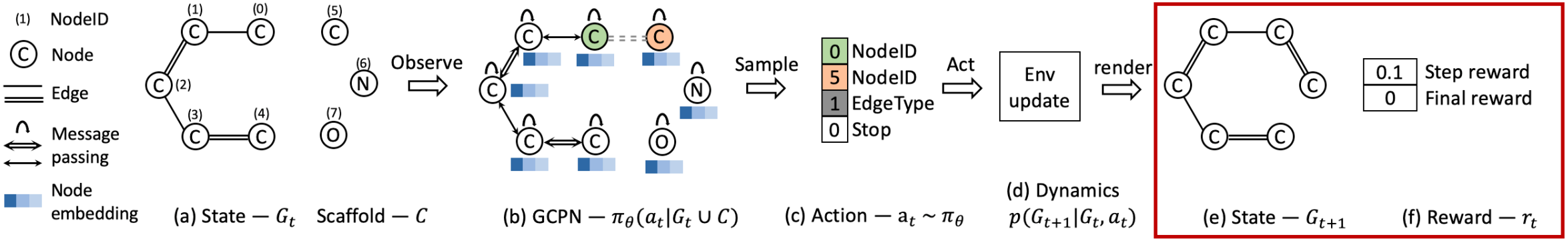
Overview of GCPN



- **(a)** Insert nodes
- **(b,c)** Use GNN to predict which nodes to connect
- **(d)** Take action (check chemical validity)
- **(e, f)** Compute reward

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

How Do We Set the Reward?



- **Step reward:** Learn to take valid action
  - At each step, assign small positive reward for valid action (satisfy valency rules)
- **Final reward:** Optimize desired properties
  - At the end, assign positive reward for high desired property

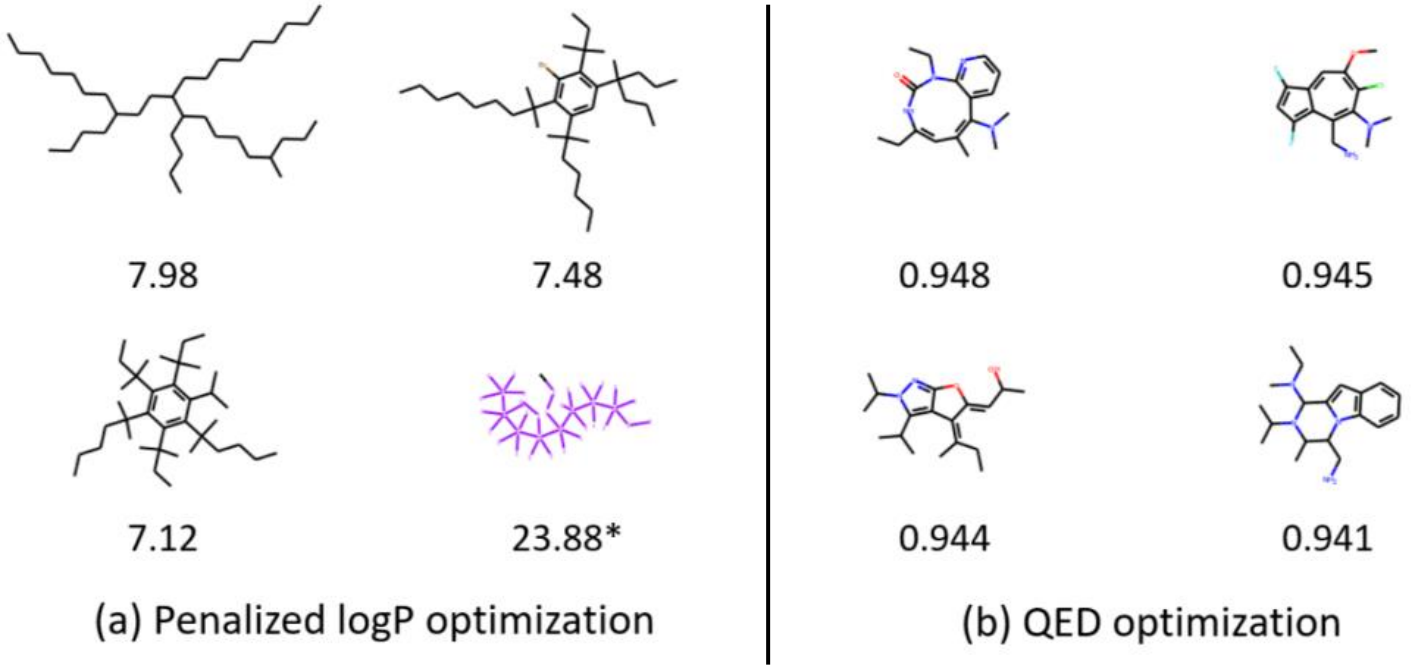
Reward = **Final reward** + **Step reward**

Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, NeurIPS 2018.

Qualitative Results

Visualization of GCPN graphs:

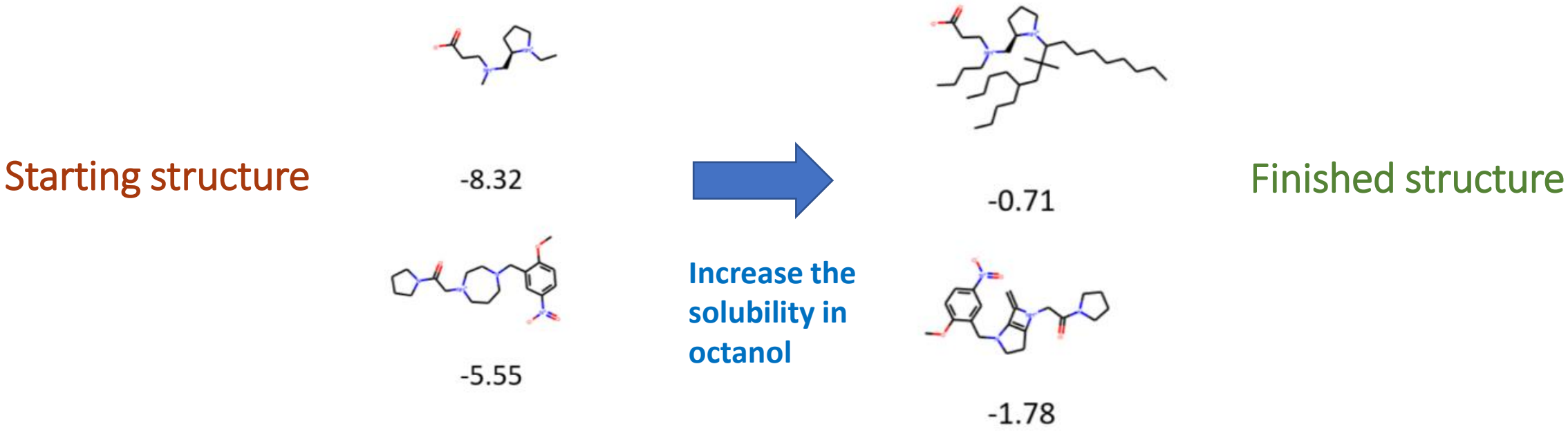
- Property optimization Generate molecules with high specified property score



Qualitative Results

Visualization of GCPN graphs:

- Constrained optimization: Edit a given molecule for a few steps to achieve higher property score



Constrained Optimization of Penalized LogP

Summary

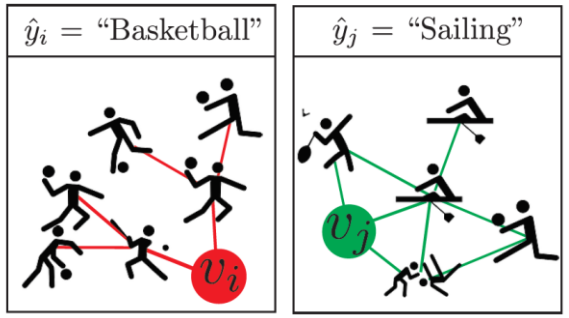
- Many biological applications can be formulated as learning on **heterogeneous** graphs
- An ML algorithm to generate graphs (molecules) is the central problem of drug discovery
  - Imitating a set of given graphs
  - Optimizing graphs towards given goals
  - Other interesting tasks:
    - Retro-synthesis; molecule conformation; drug / molecule property prediction; prediction of the effects of gene knockouts etc.

分子构造

- Many other interesting tasks!
  - Protein folding, single-cell analysis …

GNN Applications

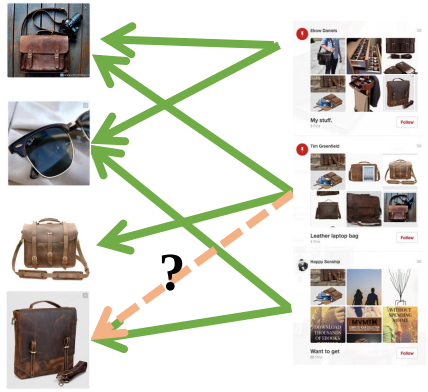
- **Social Networks**
- **Natural Science**
- **Medicine**
- **Explainability of GNNs**



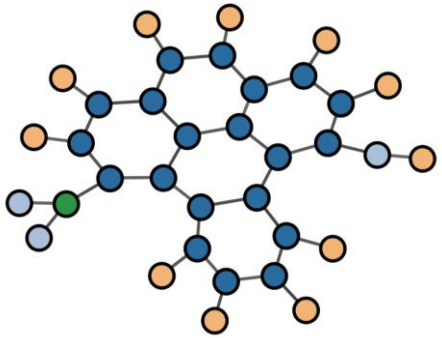
Scalable GNN Explanations

- Many questions after training GNNs:
  - Why is an item recommended to a user? — Explain link prediction
  - Why is the molecule mutagenic? 诱发的 — Explain graph classification
  - Why is the user classified as fraudulent? 诈骗的 — Explain node classification
- Being able to answer these questions are important for domain experts

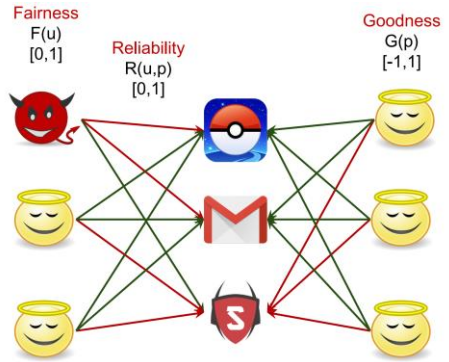
fraud detection



Recommender System



Mutagenic



Anomaly/Abuse Detection

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

Why is it hard

- Explain predictions for multiple tasks
  - Node classification
  - Graph classification
  - Link prediction

- Model agnostic (*post-hoc*)
  - Can be applied to all models covered in class: GCN, GraphSAGE, GAT etc.

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

Recap: GNN Framework

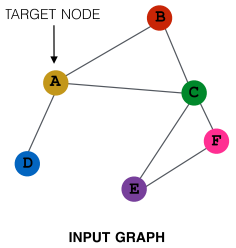
- Message Computation

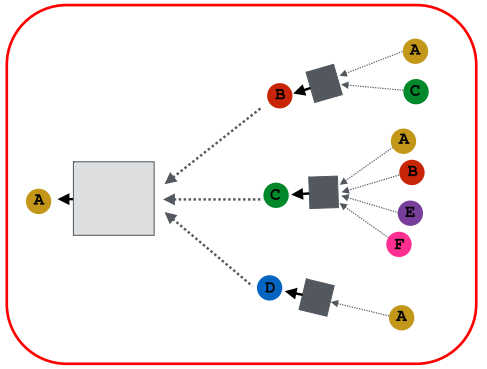
$$m_{ij}^{l} = \text{Msg}(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}, e_{ij})$$

Previous layer neighbor embedding

- Aggregation

$$M_i^{l} = \text{Agg}\left(\{m_{ij}^{l} | v_j \in N_{v_i}\}\right)$$





Information used by GNN

- Representation update

$$\mathbf{h}_i^{l} = \text{Update}\left(M_i^{l}, \mathbf{h}_i^{l-1}\right)$$

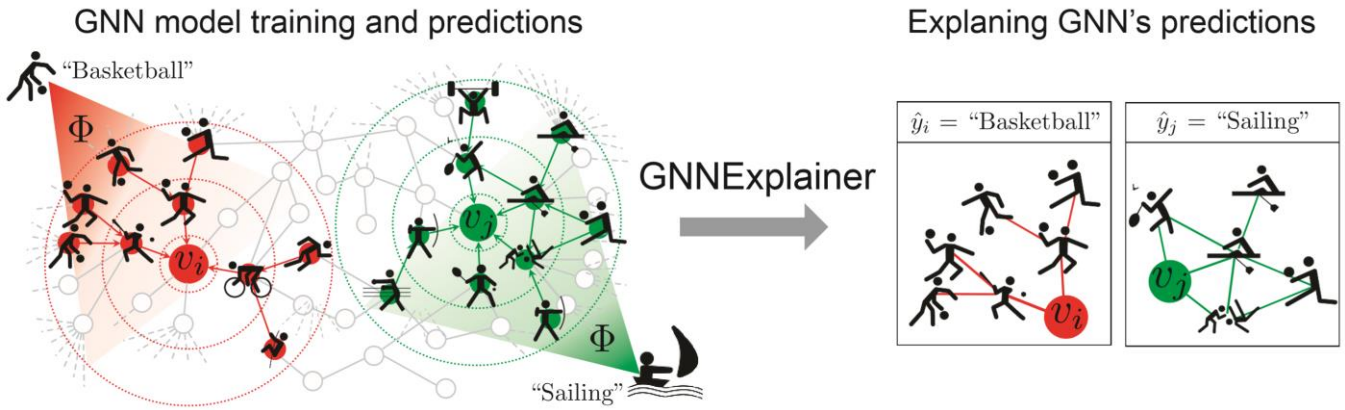
- Stack Multiple layers

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

Model Explanation

- **Training time:**
  - Optimize GNN on training graphs
  - Save the trained model
- **Test time:**
  - Explain predictions made by th GNN
  - On unseen instances (nodes, edges, graphs)

node prediction

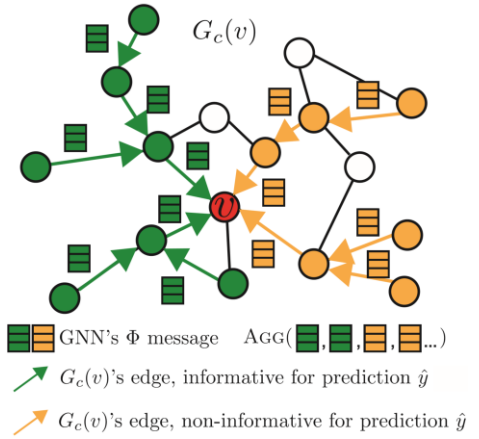


GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

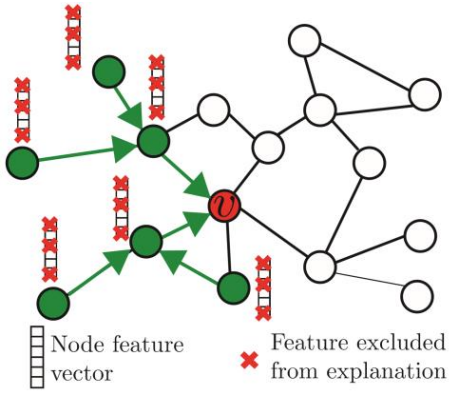
GNNExplainer

- Message passing structure
- The importance of node features
- Explain both aspects simultaneously { structure, feature

- To explain a given node, learn
  - Important edges in its neighborhood
  - Important node feature dimensions
- Mutual information objective



Structural explanation

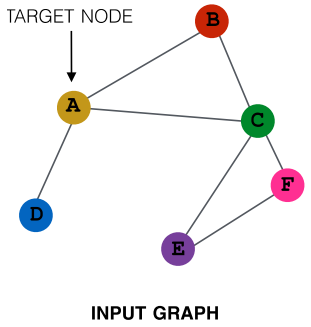


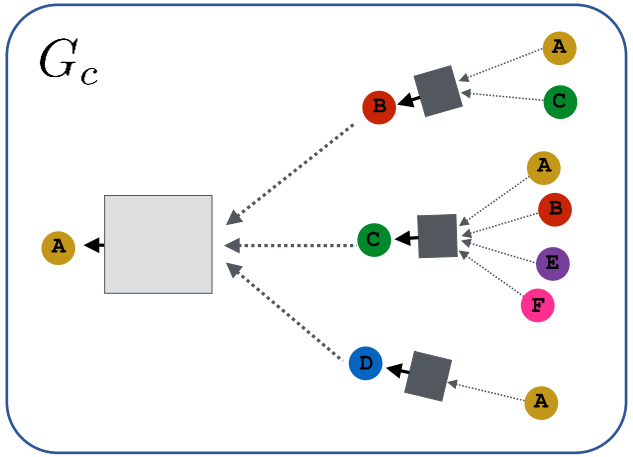
Feature explanation

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

GNN-Explainer Input

- Consider node classification task:





Suppose GNN predicts label $\hat{y}$ for node v

- Input computation graph: $G_c(v)$
- Adjacency matrix: $A_c(v) \in \{0,1\}^{n \times n}$
- Node Feature: $X_c(v) = \{x_j | v_j \in G_c(v)\}$

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

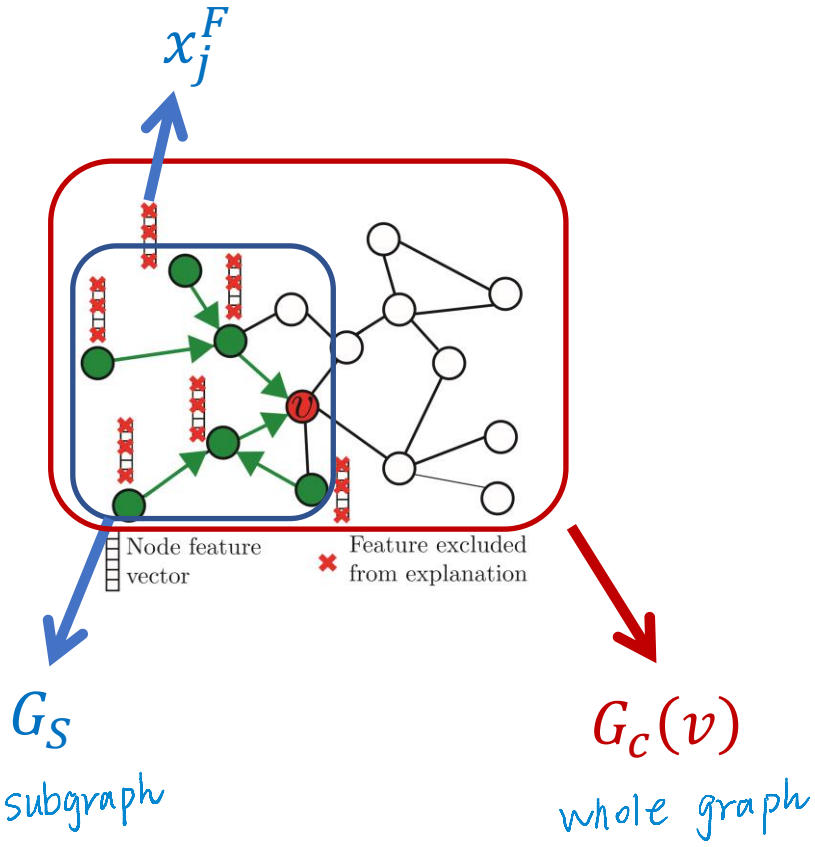
GNN-Explainer Output

subgraph and subgraph node features

- GNN model $\boldsymbol{\phi}$ learns $P_{\phi}(Y \mid G_c(v), X_c(v))$
- Y denotes predicted label of v
- **GNNExplainer** outputs (G_S, X_S^F)

whole graph

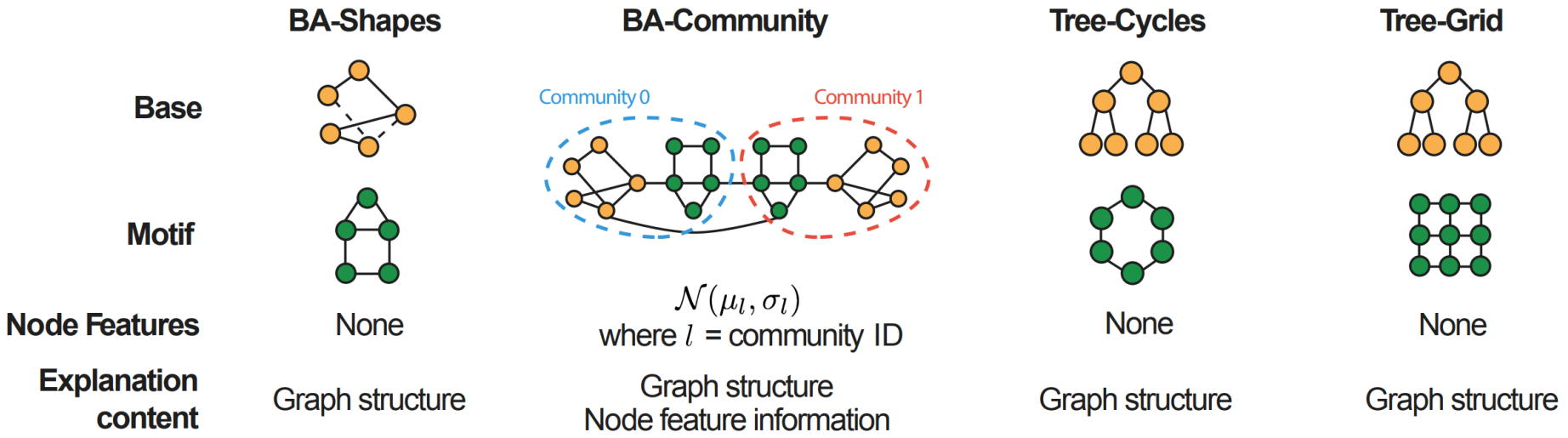
- G_S is a small subgraph of $G_c(v)$ (omit v)
- $X_S^F = \{x_j^F | v_j \in G_S\}$ are features for G_S
- Mask F masks out unimportant dimensions



GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

Experiments

- Synthetic task: is a node part of a motif?



- Real-world social networks, molecules
  - Graph classification

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

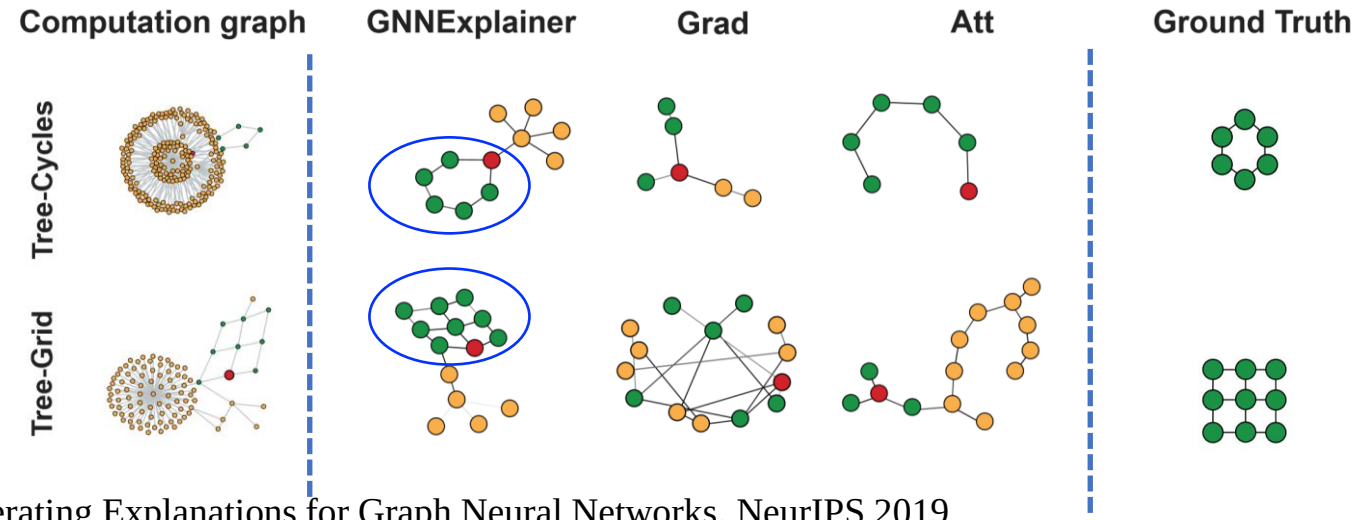
Explainability Method Comparison

- GNN saliency map based on gradients
  - Large gradient: more important
- Graph Attention Networks (GAT)
  - Edge importance indicated by attention
  - Average attention weights across layers

GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

GNNExplainer

- Explain node classification using GCN
- Base: **Barabasi-Albert Graph**; addon: **Cycle, Grid**



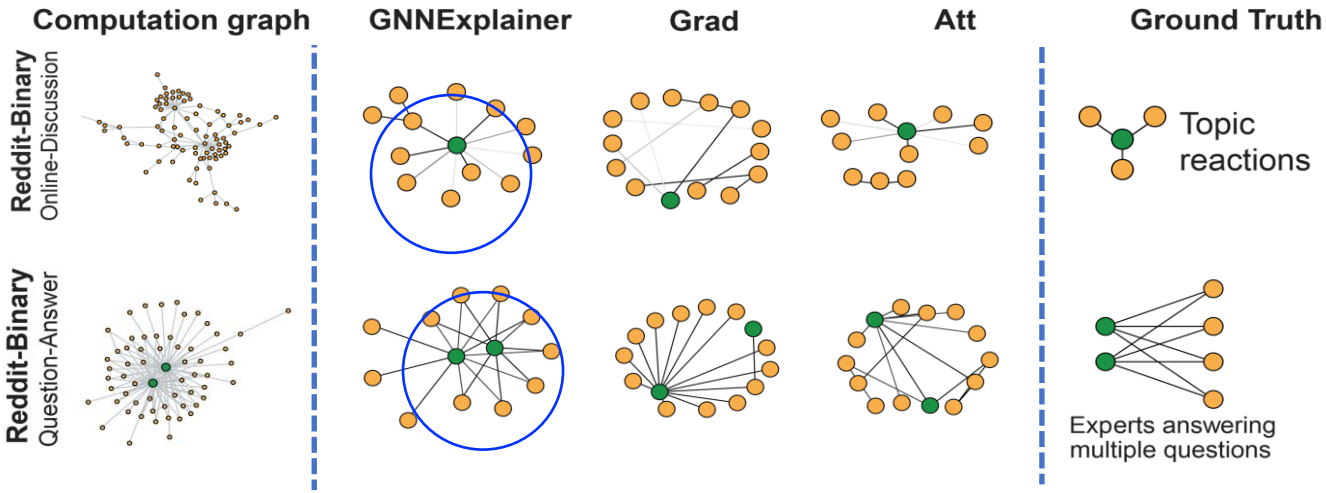
GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

GNNExplainer

Graph Classification

Reddit-Binary: 2000 threads, 2 classes

Discussion community vs. QA community



GNNExplainer: Generating Explanations for Graph Neural Networks, NeurIPS 2019

Conclusion

- We have covered a variety of graph learning applications, in domains including **online social networks**, **natural sciences** and **medicine**.
- Many of the problems were not always formulated as a graph problem!
- Interesting directions to explore
  - Novel ways to **model** problems through the angle of graph learning
  - **Scalable GNN algorithms** for extremely large datasets
  - **AutoML**, **explainability** for GNN applications

Deep Learning on Graph-Structured Data

- New course on **Deep Learning on Graph-structured Data** (Fall 2022)
- The scope will be primarily focused on **theory**, **algorithms** and **applications** related to GNNs and geometric deep learning
  - We also encourage students who primary work on ML for visual, text or other forms of data to explore potential incorporation of graph techniques in their research problems