



Deep Learning for Healthcare

Graph Neural Networks

Jimeng Sun

Outline



- Fundamentals of deep learning on graphs
- Graph convolutional networks (GCN)
- Message passing neural network (MPNN)
- Graph attention networks (GAT)
- Applications

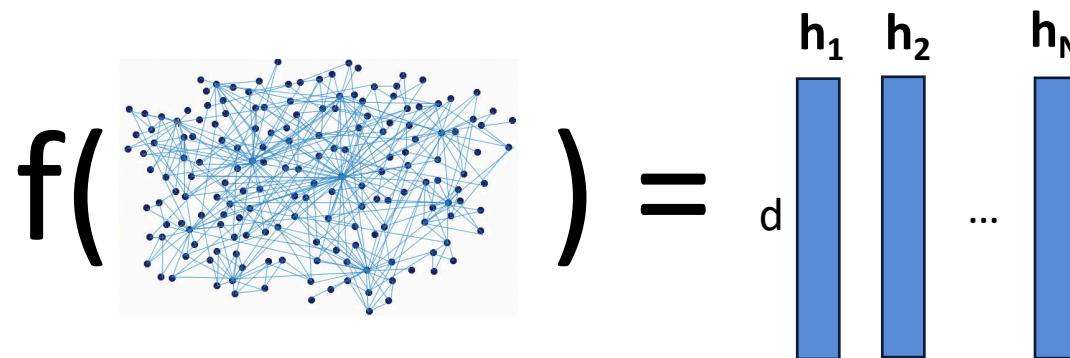
GNN fundamental

- Node embedding
- Tasks on graphs
- Challenges with GNN
- Key ideas of GNN

Node embedding



- Goal: Map nodes to d-dimensional vectors such that similar nodes are close to each other

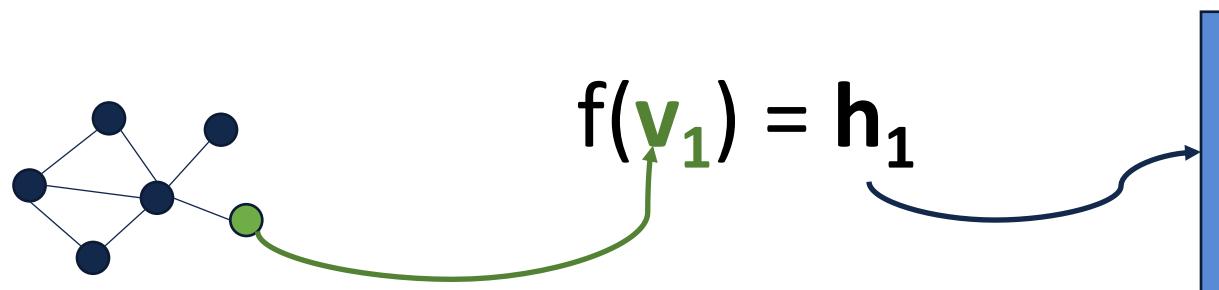


How to learn embedding function f ?

Key components



- **Embedding function:** map a node to a vector



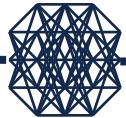
- **Similarity function:** define relationships of embedding vectors

$$S(v_1, v_2) = h_1^T h_2$$

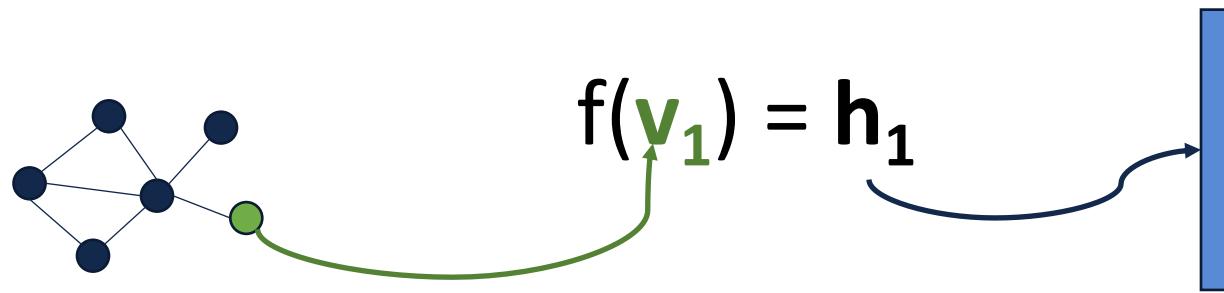
Similarity between nodes

Inner product between embeddings

Graph neural network



- **Embedding function:** map a node to a vector

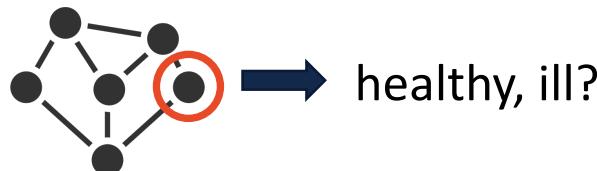


Embedding function is a deep neural network that leverages the graph structure

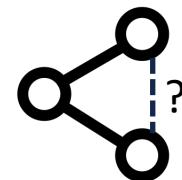
Tasks on graphs



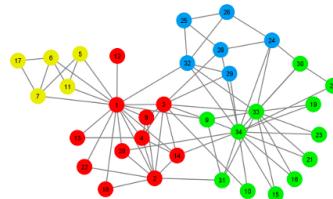
- Node classification



- Link prediction



- Community detection



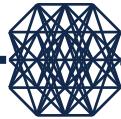
- Graph property prediction



- Graph generation



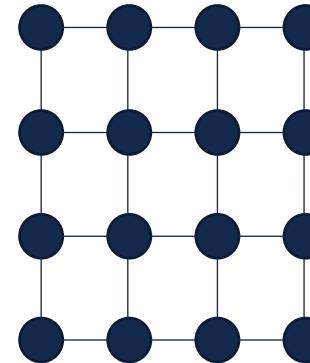
Challenges of graph neural network



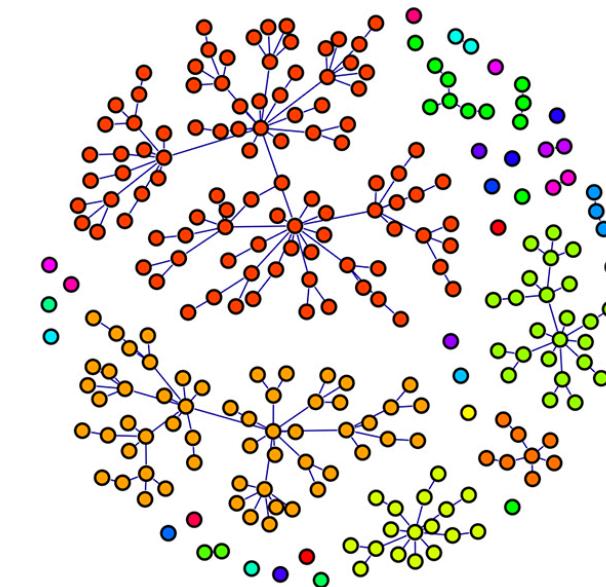
- Arbitrary size and complex structure
- No fixed node ordering
- Dynamic
- Heterogeneous features



Text

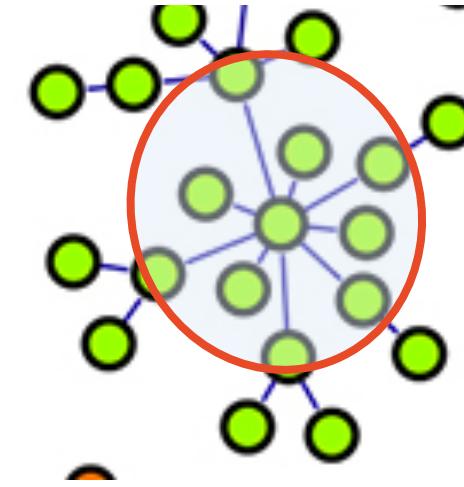
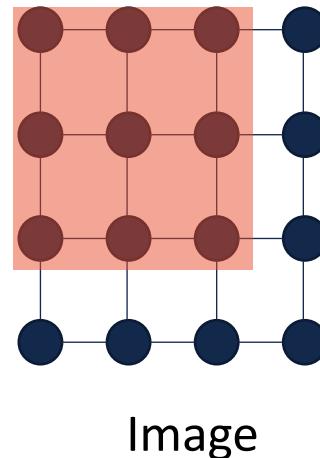
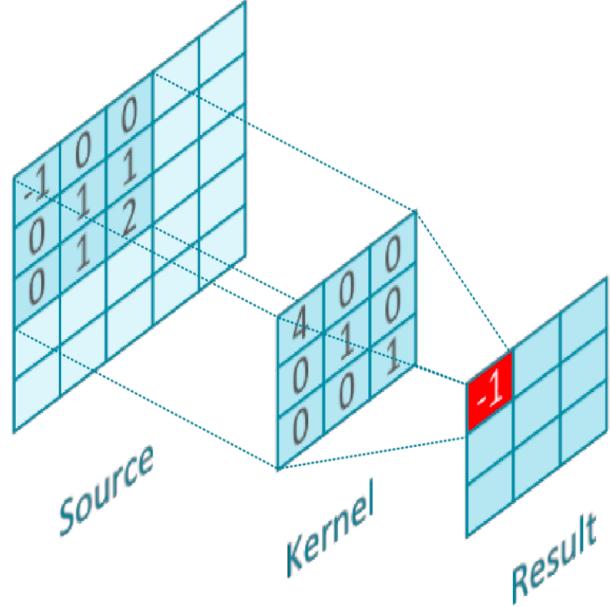


Image



Graph

Generalize convolution to graph



Graph convolution

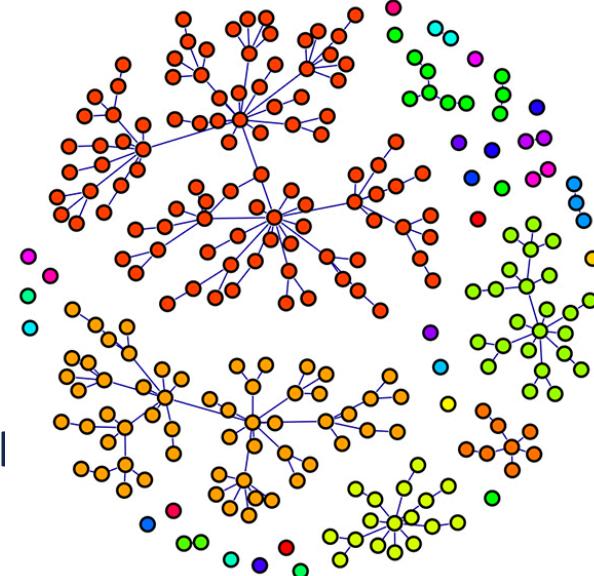
- Aggregate information from neighbors
- Combine them

$$\sum W h_i$$

Input



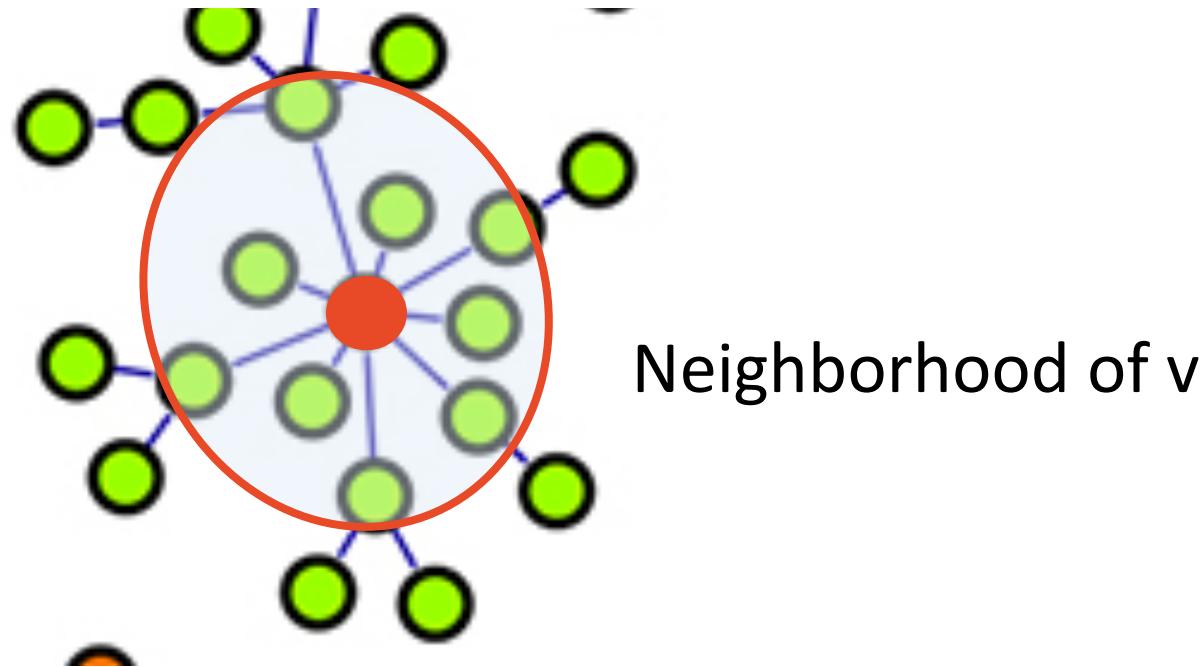
- Graph $G = (V, E, X)$
 - V is the node set
 - E is the edge set
 - A is the adjacency matrix
 - X is the node features
 - E.g., chemical information in molecule graph
 - No feature:
 - one-hot encoding for a node



Idea 1: aggregation

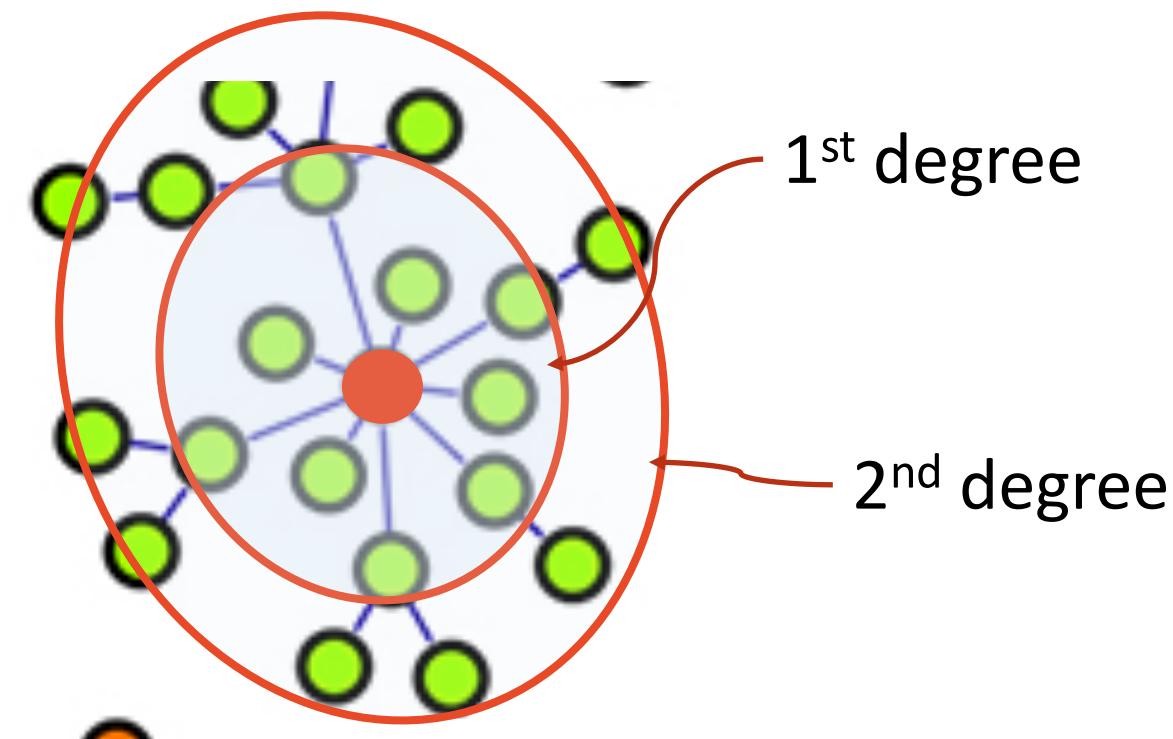


- Neighbors of a node defines the aggregation set



Idea 2: layering

- Embedding can have multiple layers
- Layer 0: x
- Layer 1: $h^{(1)}$ aggregate one hop neighbors
- Layer K: $h^{(k)}$ aggregate k-hop neighbors



Aggregation of GNN



- Approach: average neighboring embeddings and apply a neural network

Embedding at layer l

$$h_i^{(l+1)}$$

Embedding at layer l-1

$$h_i^{(l)}$$

Activation function

$$W^l + \sum_{j \in \mathcal{N}_i} h_j^{(l)} W^l)$$

Aggregate neighbor's embeddings

Aggregation of GNN



- Approach: average neighboring embeddings and apply a neural network

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{h}_i^{(l)} \mathbf{W}^l + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)} \mathbf{W}^l)$$

Weight parameters

Loss function for node classification



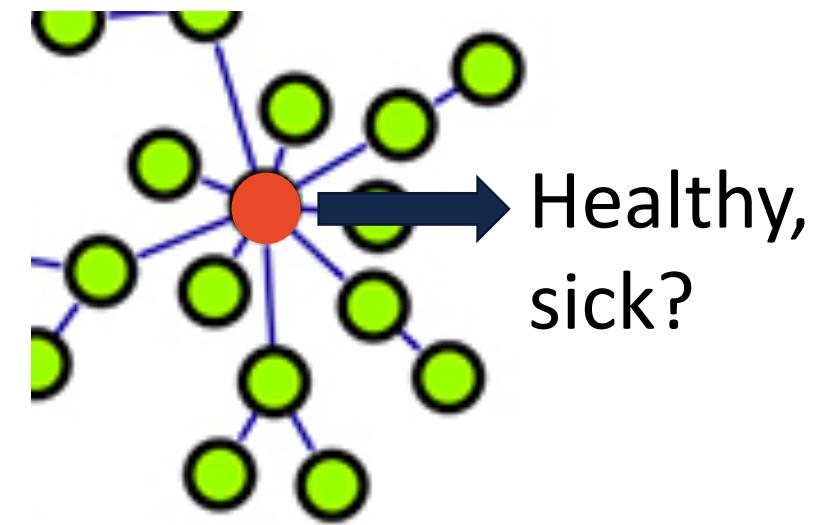
- Train a GNN model for a supervised learning task (such as node classification)

$$\mathcal{L} = \sum_i y_i \log(\sigma(\mathbf{h}_i^\top \boldsymbol{\theta})) + (1 - y_i) \log(1 - \sigma(\mathbf{h}_i^\top \boldsymbol{\theta}))$$

Class label for node v_i

Embedding for node v_i

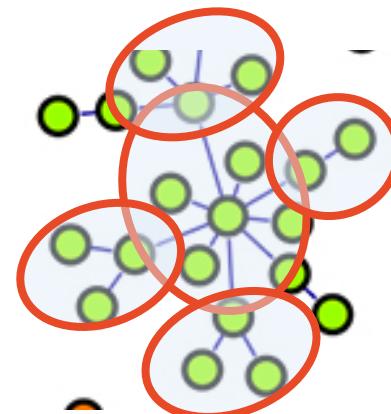
Cross entropy loss



GNN general design



1. Define a neighborhood aggregation function
2. Define a loss function on the embeddings
3. Train on a set of nodes based on local neighborhoods
4. Generate embeddings for any node based on node features
 - Even the ones that are not trained on – **Inductive learning**

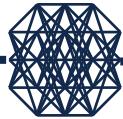


Outline

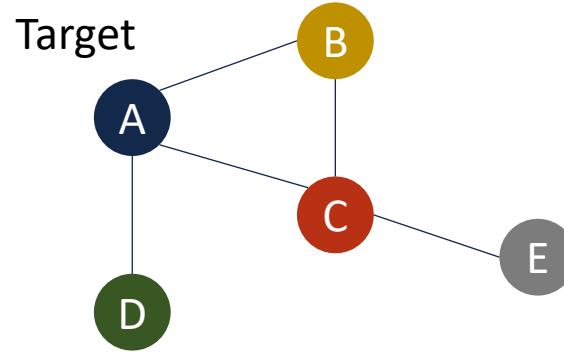


- Fundamentals of deep learning on graphs ✓
- Graph convolutional networks (GCN)
- Message passing neural network (MPNN)
- Graph attention networks (GAT)
- Applications

Graph convolutional networks (GCN)



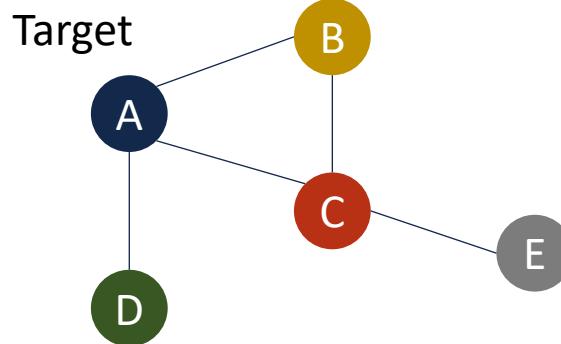
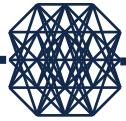
- Idea: compute node embedding based on neighboring embeddings



$$h_a = \sigma(h_a W_a + \text{aggregate}(h_u | u \in N_a))$$

$$h_a = \sigma(h_a W_a + \text{aggregate}(B, C, D))$$

Neighbor aggregation function



- Mean
 - Aggregation = average(B C D)
- Pooling
 - E.g., Element-wise max over h_u for $u \in N_a$
- Recurrent neural networks
 - Apply RNN to random shuffled neighbors

Matrix view of GCN



- Aggregation can be performed in parallel using matrix operations
- Vector view:

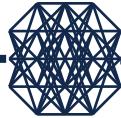
$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{h}_i^{(l)} \mathbf{W}^l + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)} \mathbf{W}^l)$$

- Matrix view

$$\mathbf{H}^{(l+1)} = \sigma \left((\mathbf{I} + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \mathbf{H}^{(l)}) \mathbf{W}^{(l)} \right)$$

Matrix view is much more efficient for training

Speedup GCN training via sampling



$$\text{For each node } i \quad \mathbf{h}_i^{(l+1)} = \sigma(\mathbf{h}_i^{(l)} \mathbf{W}^l + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)} \mathbf{W}^l)$$

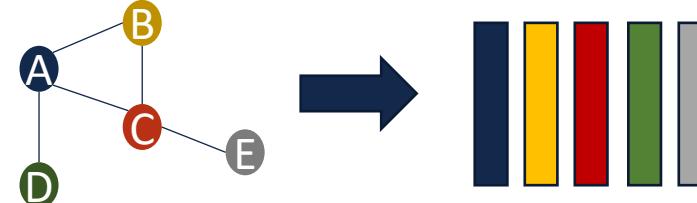
Training bottleneck

- GraphSage [1] samples a random subset of edges for each node
- FastGCN [2] biased samples a subset of nodes with higher degrees, then sample random edges from the nodes

[1] Hamilton, William L., Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs.” *NIPS 2017*

[2] Chen, Jie, Tengfei Ma, and Cao Xiao. “FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling.” *ICLR 2018*.

Summary of GCN



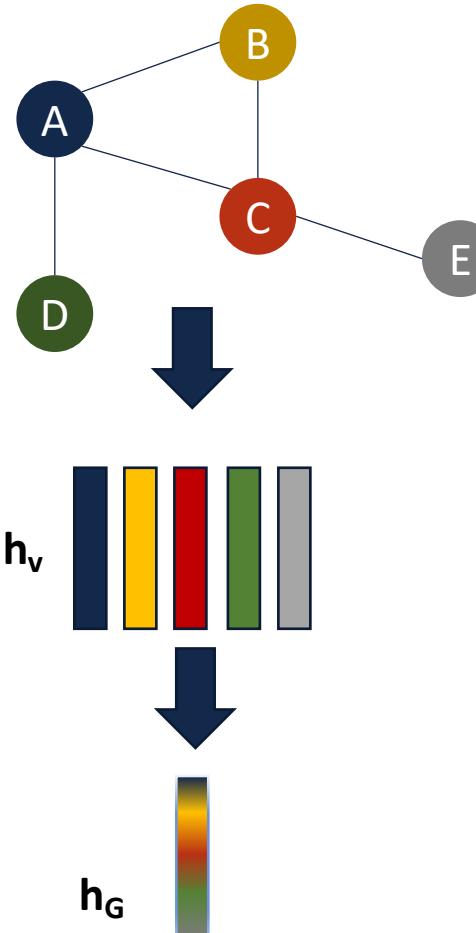
- GCN is an efficient method for generating node embeddings on graphs
- Keys
 - Aggregation function on local neighborhood
 - Loss function
 - Multiple layers are allowed like a standard DNN
 - Sample nodes and edges from a graph or multiple graphs
 - Shared Layer-wise parameters across nodes
- Remarks
 - GCN embeddings are invariant to node orderings
 - Limitation: 1) node features are important to have; 2) Edge features are not supported

Outline



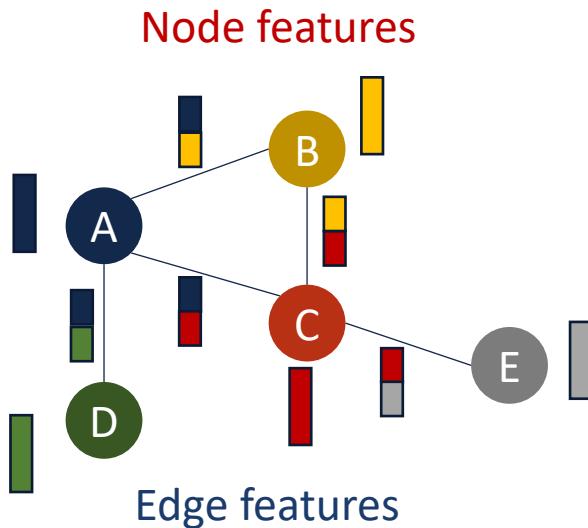
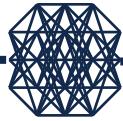
- Fundamentals of deep learning on graphs ✓
- Graph convolutional networks (GCN) ✓
- Message passing neural network (MPNN)
- Graph attention networks (GAT)
- Applications

Message passing neural network (MPNN)



- Goal: Generate both **node embeddings** and **graph embedding**
- MPNN is a GNN framework that handles both node and edge features
- **Message passing stage:**
 - compute the message \mathbf{m}_v for node v
 - update the node embedding \mathbf{h}_v
- **Readout stage**
 - Produce an embedding \mathbf{h}_G for the entire graph

Message passing phase



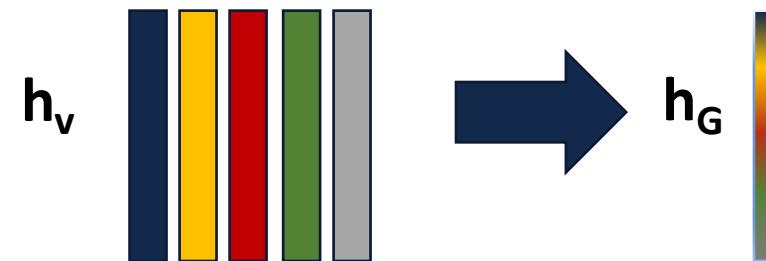
Message passing $\mathbf{m}_v^{t+1} = \sum_{w \in \mathcal{N}_v} M_t(\mathbf{h}_v^t, \mathbf{h}_w^t, \mathbf{e}_{vw})$

$$\mathbf{m}_A = M_t(\mathbf{h}_A, \mathbf{h}_B, \mathbf{e}_{AB}) + M_t(\mathbf{h}_A, \mathbf{h}_D, \mathbf{e}_{CD}) + M_t(\mathbf{h}_A, \mathbf{h}_E, \mathbf{e}_{EA})$$

Node embedding $\mathbf{h}_v^{t+1} = U_t(\mathbf{h}_v^t, \mathbf{m}_v^{t+1})$

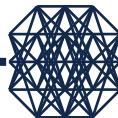
$$\mathbf{h}_v^{t+1} = U_t(\mathbf{h}_v^t, \mathbf{m}_A)$$

Readout phase

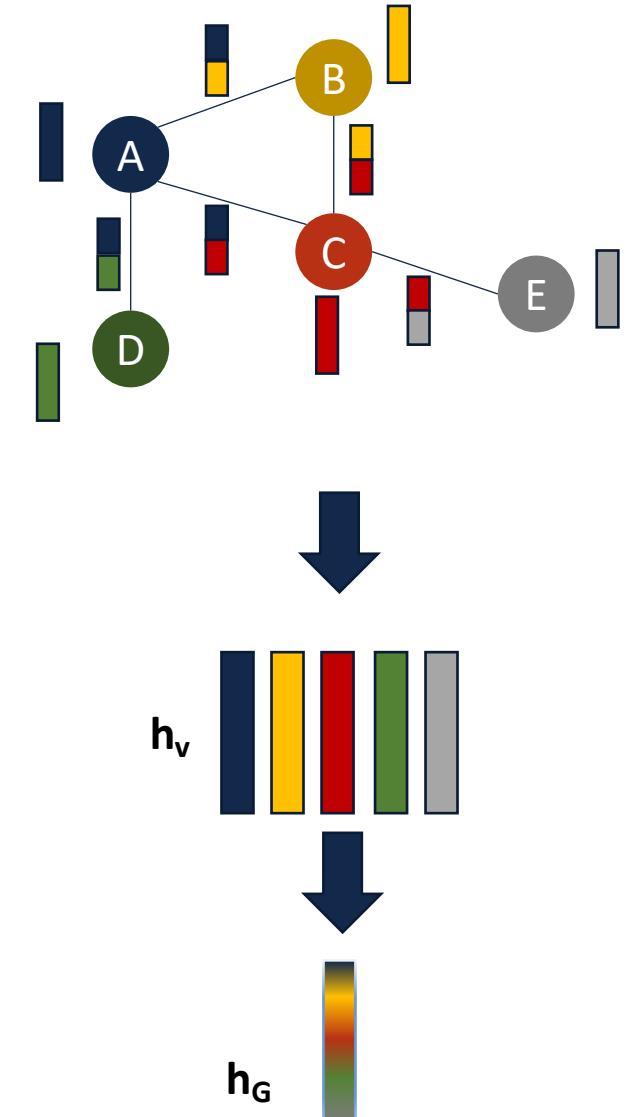


- Input: node embedding h_1, h_2, \dots, h_N
- Output: an embedding vector for the entire graph h_G
- Algorithm options
 - Average(h_1, h_2, \dots, h_N)
 - Max-pool(h_1, h_2, \dots, h_N)
 - RNN over random-shuffle(h_1, h_2, \dots, h_N)

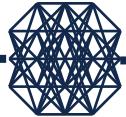
Summary of MPNN



- MPNN is a GNN that handles both node features and edge features.
- Two phases
 - Message passing phase
 - Readout phase
- Limitation:
 - computationally more expensive to train and only can handle smaller graphs such as molecule graphs.

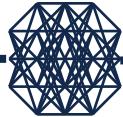


Outline



- Fundamentals of deep learning on graphs ✓
- Graph convolutional networks (GCN) ✓
- Message passing neural network (MPNN) ✓
- Graph attention networks (GAT)
- Applications

Graph attention networks (GAT)



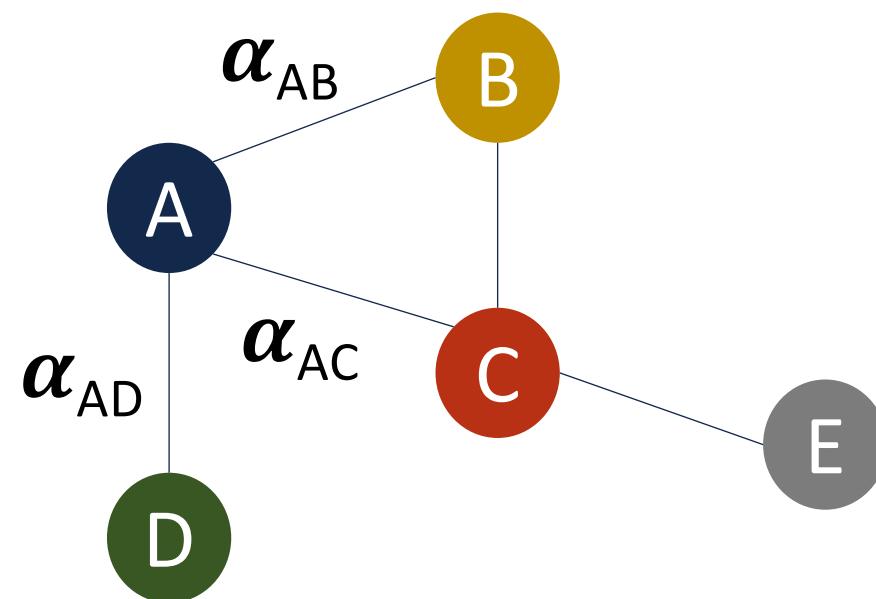
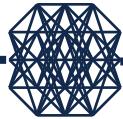
- **Recall** GCN aggregates neighborhoods uniformly with equal weight

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{h}_i^{(l)} \mathbf{W}^l + \sum_{j \in \mathcal{N}_i} \mathbf{h}_j^{(l)} \mathbf{W}^l)$$

Equal importance
for each neighbor

- **Idea:** assign different weights for different neighbors based on their importance

Graph attention networks (GAT)



- Attention mechanism on graph: compute attention weight α_{ij} from embeddings of neighboring nodes

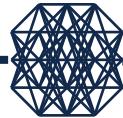
Alignment : $e_{ij} = \text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])$

concatenation

Attention weight $\alpha_{ij} = \text{Softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{v_j \in \mathcal{N}_i} \exp(e_{ij})}$

Attention α_{ij} is only computed for neighbors
(the rest has 0 attention)

Multihead attention



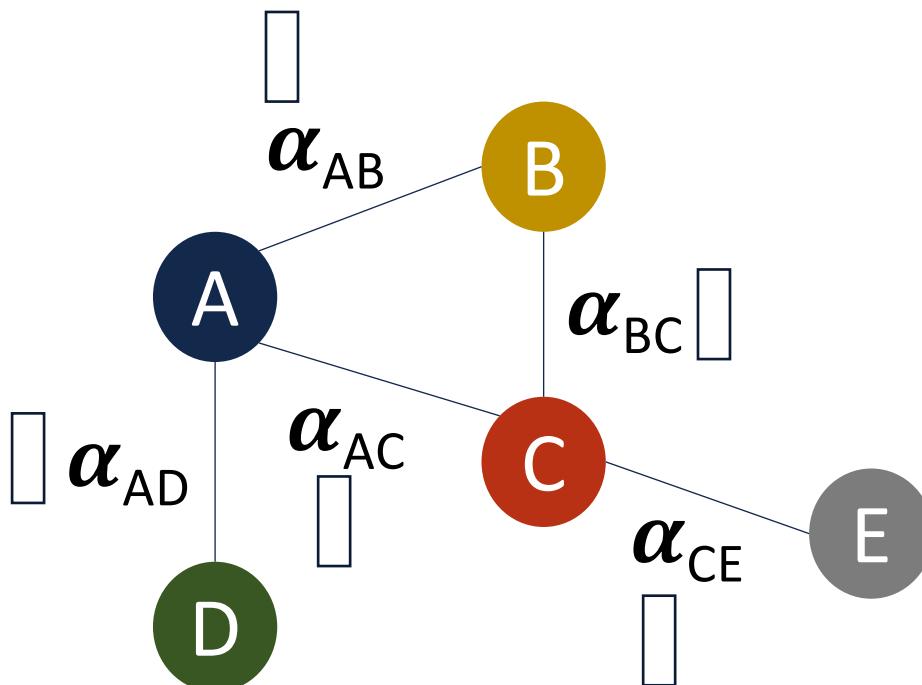
- Repeat the attention process R times

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i || \mathbf{W}\mathbf{h}_j])$$

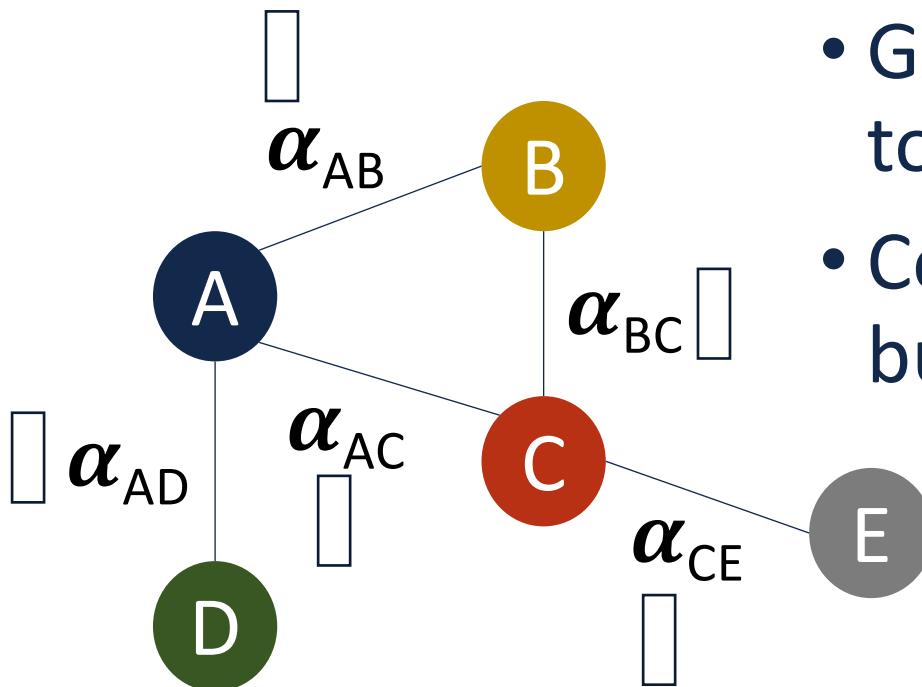
Repeat R times
with different W

$$\alpha_{ij} = \text{Softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{v_j \in \mathcal{N}_i} \exp(e_{ij})}$$

- Benefit: Stabilize the learning process of attention mechanism
- Attention weight α_{ij} becomes a vector of size R



Summary of Graph attention network



- GAT is a combination of attention mechanism and graph convolutional network
- GAT allows assigning different importance α_{ij} to different neighbors
- Computationally: more efficient than MPNN but slower than GCN

Summary

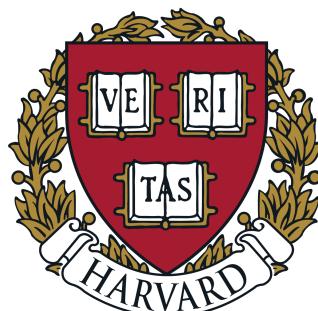


- Fundamentals of deep learning on graphs ✓
- Graph convolutional networks (GCN) ✓
- Message passing neural network (MPNN) ✓
- Graph attention networks (GAT) ✓
- Applications
 - Side-effect prediction
 - Antibiotic discovery

Learning actionable representations of biomedical data

Marinka Zitnik

Assistant Professor, Department of Biomedical Informatics
Associate Member, Broad Institute of Harvard and MIT



zitniklab.hms.harvard.edu

I ILLINOIS

Polypharmacy



Patients take multiple drugs to treat complex or co-existing diseases

46% of people over 65 years take more than 5 drugs

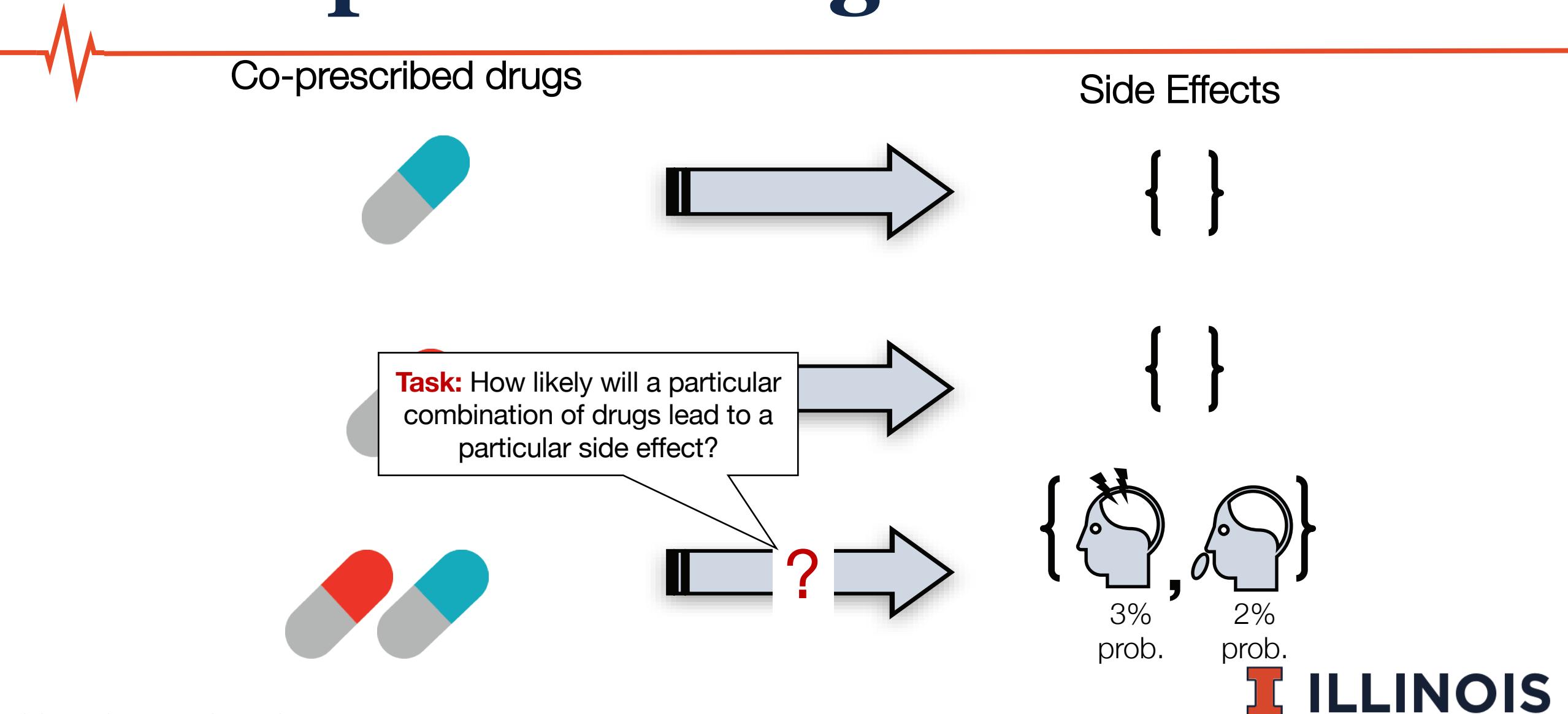
Many take more than **20** drugs to treat heart diseases, depression or cancer

15% of the U.S. population affected by unwanted side effects

Annual costs in treating side effects exceed **\$177** billion in the U.S. alone

[Ernst and Grizzle, JAPA'01; Kantor et al., JAMA'15]

Unexpected Drug Interactions

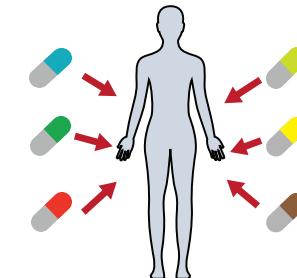


Why is modeling polypharmacy hard?



Combinatorial explosion

- >13 million possible combinations of 2 drugs
- >20 billion possible combinations of 3 drugs



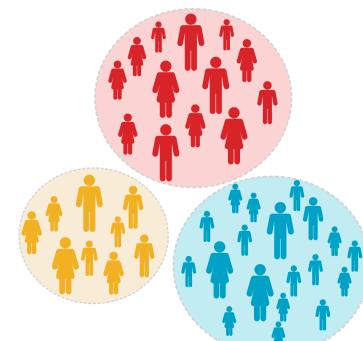
Non-linear & non-additive interactions

- Different effect than the additive effect of individual drugs



Small subsets of patients

- Side effects are interdependent
- No info on drug combinations not yet used in patients



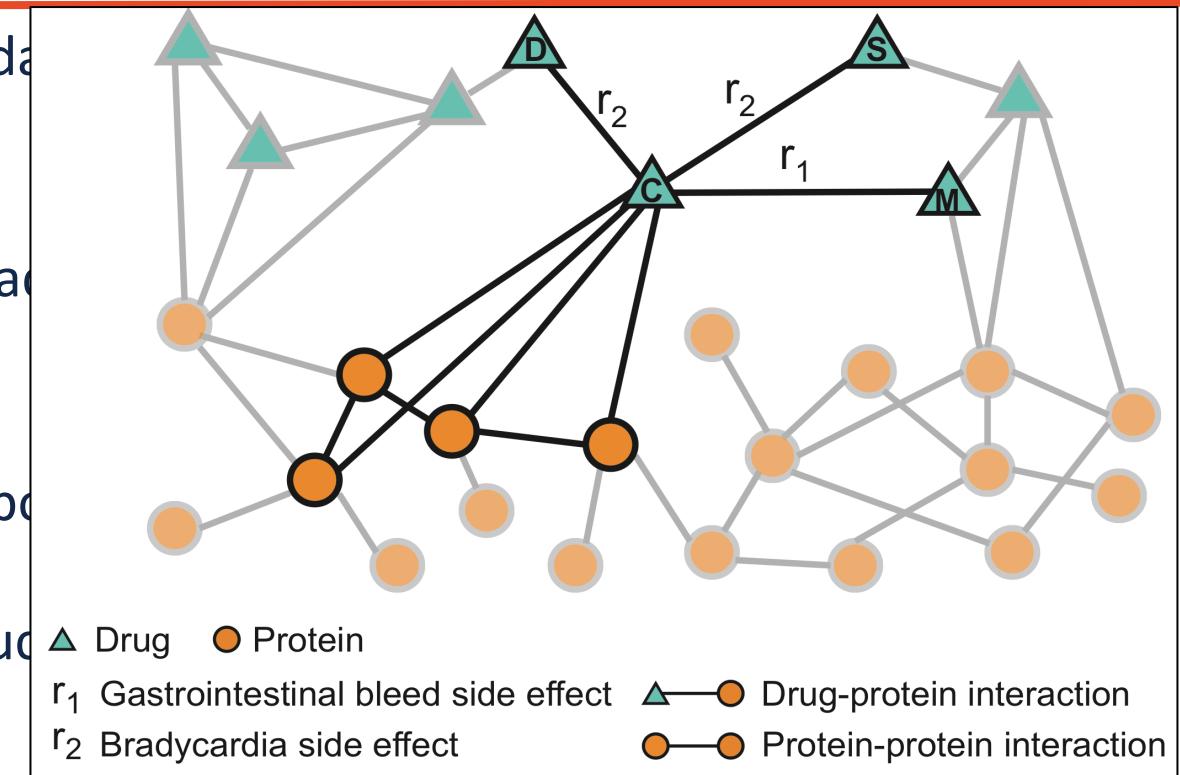
We need polypharmacy dataset



Objective: Capture molecular, drug, and patient data prescribed in the U.S.

The authors build a unique dataset:

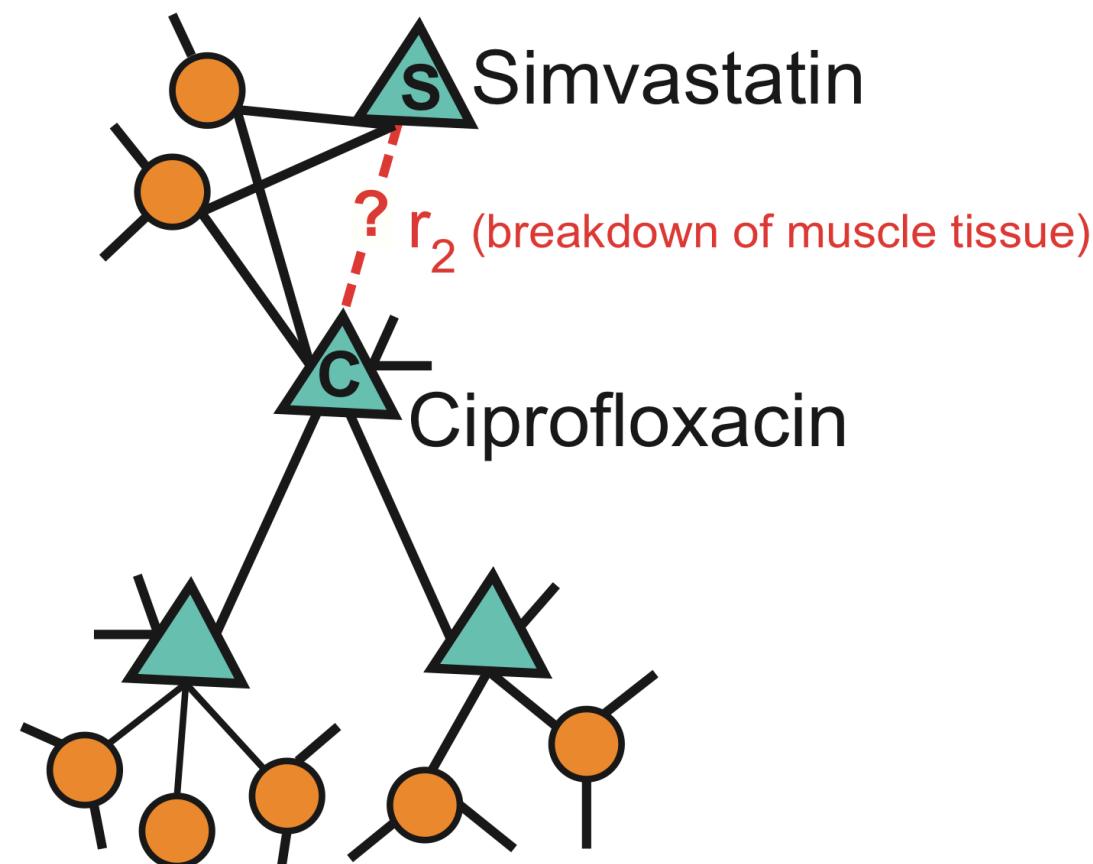
- **4,651,131 drug-drug edges:** Patient data from a test set of 10,000 patients tested for confounders [FDA]
- **18,596 drug-protein edges**
- **719,402 protein-protein edges:** Physical, metabolic, and signaling interactions
- **Drug and protein features:** drugs' chemical structure and membership in pathways



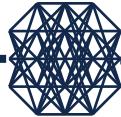
A polypharmacy network with over 5 million edges and over 1,000 different edge types

We apply our deep methods to the polypharmacy network

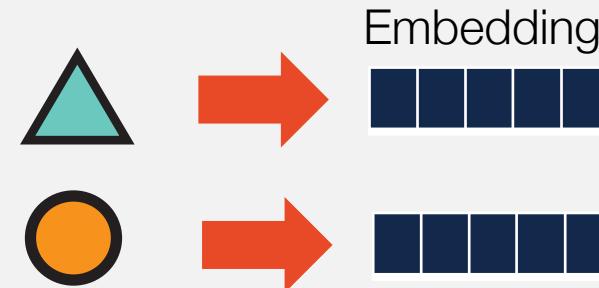
E.g.: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?



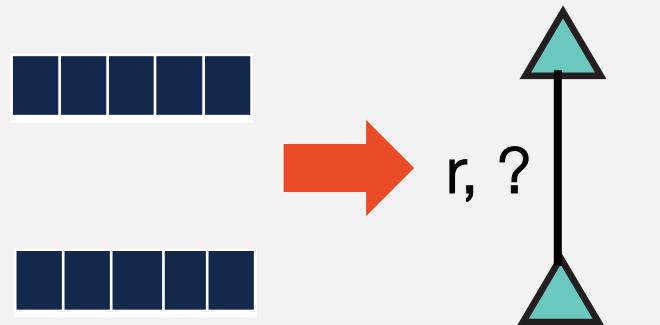
Decagon: Graph Neural Net



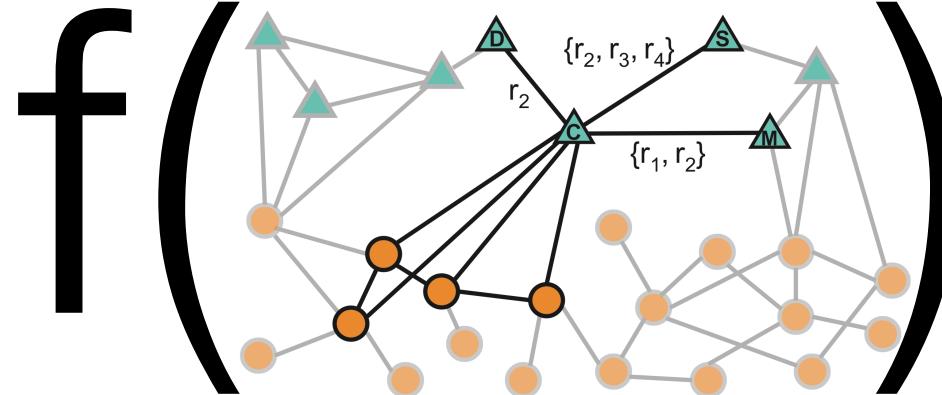
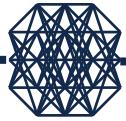
1. Encoder: Take the graph and learn an *embedding* for every node



2. Decoder: Use the learned embeddings to predict side effects

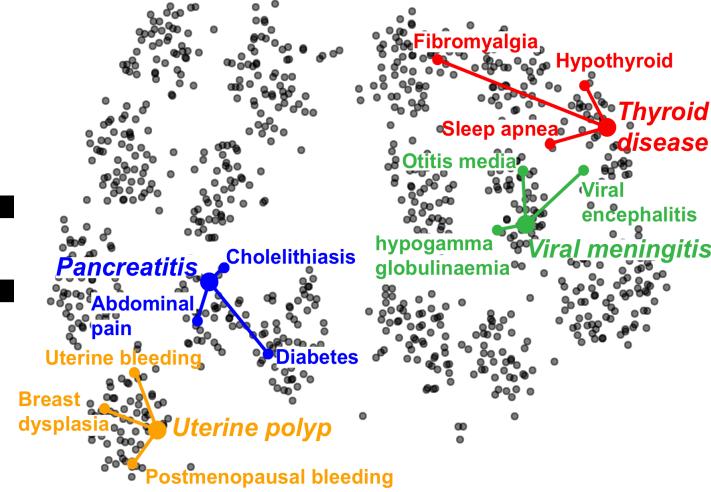


Embedding Nodes



Heterogeneous
graph

=



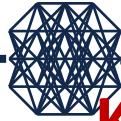
2-dimensional node
embeddings



How to learn f ?

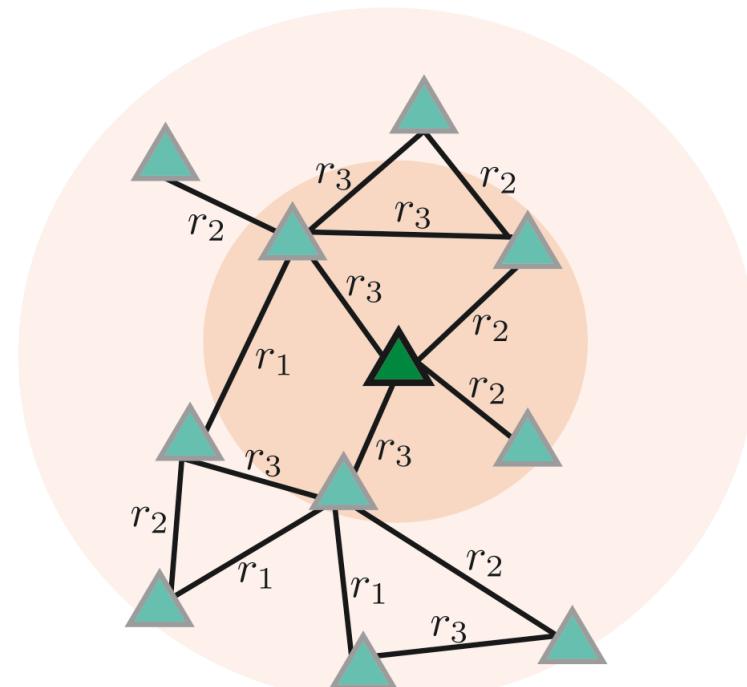
Intuition: Map nodes to d -dimensional embeddings such that similar nodes in the graph are embedded close together

Encoder: Principle

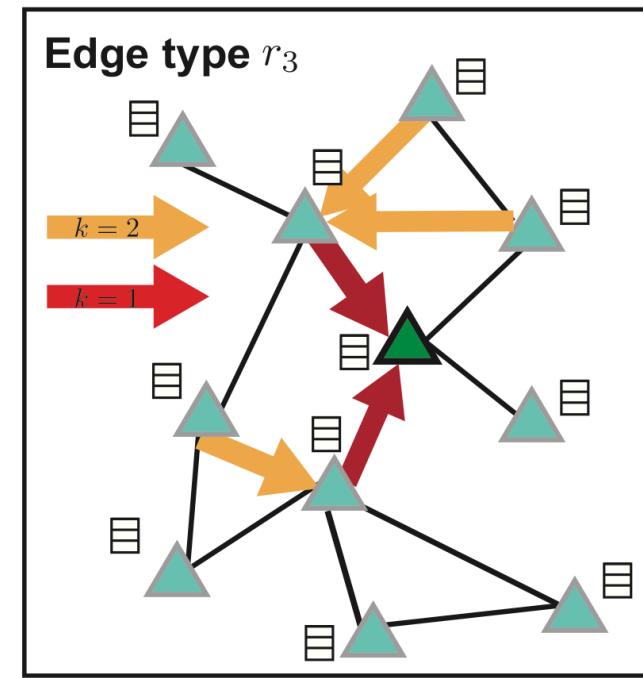


Key idea: Generate node embeddings based on local network neighborhoods

Each edge type is modeled separately



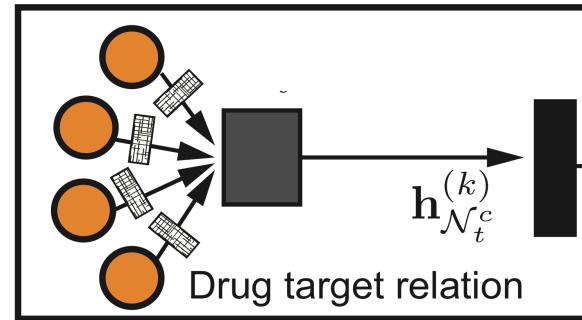
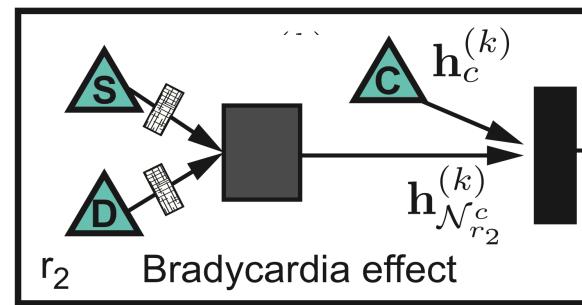
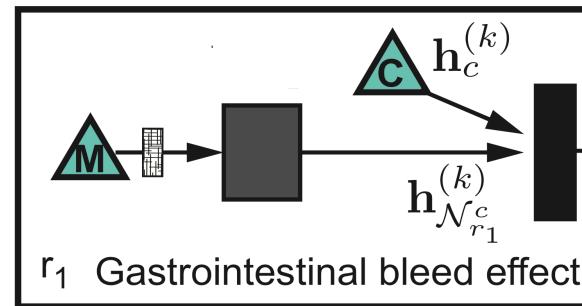
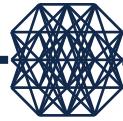
Determine a node's computation graph



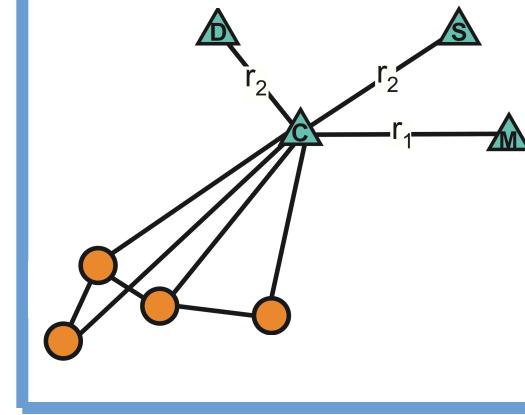
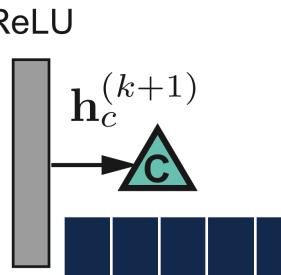
Learn how to transform and propagate information across the graph

ILLINOIS

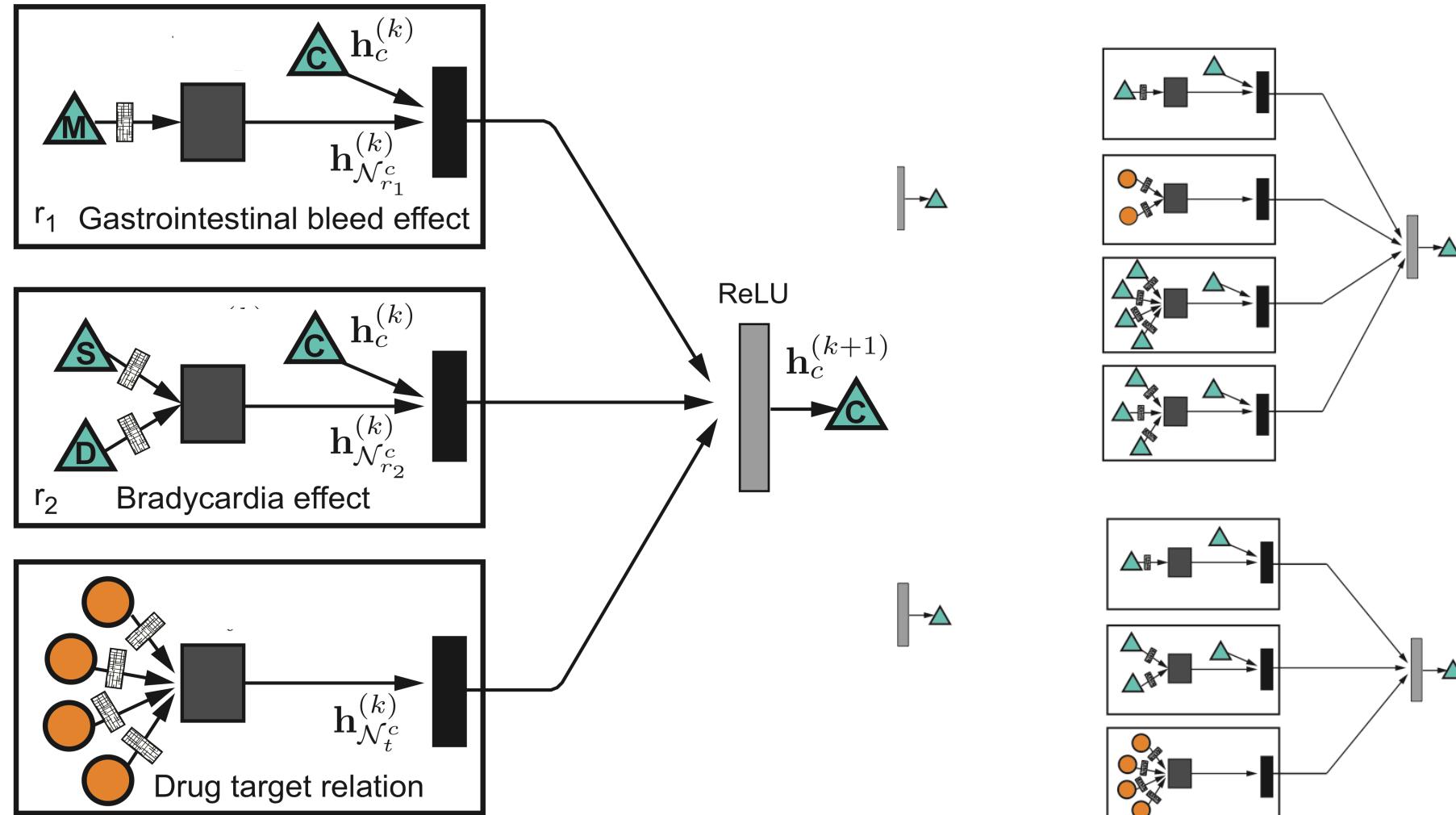
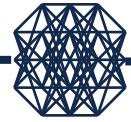
Encoder: Embeddings



One-layer computation graph
for drug $\triangle C$

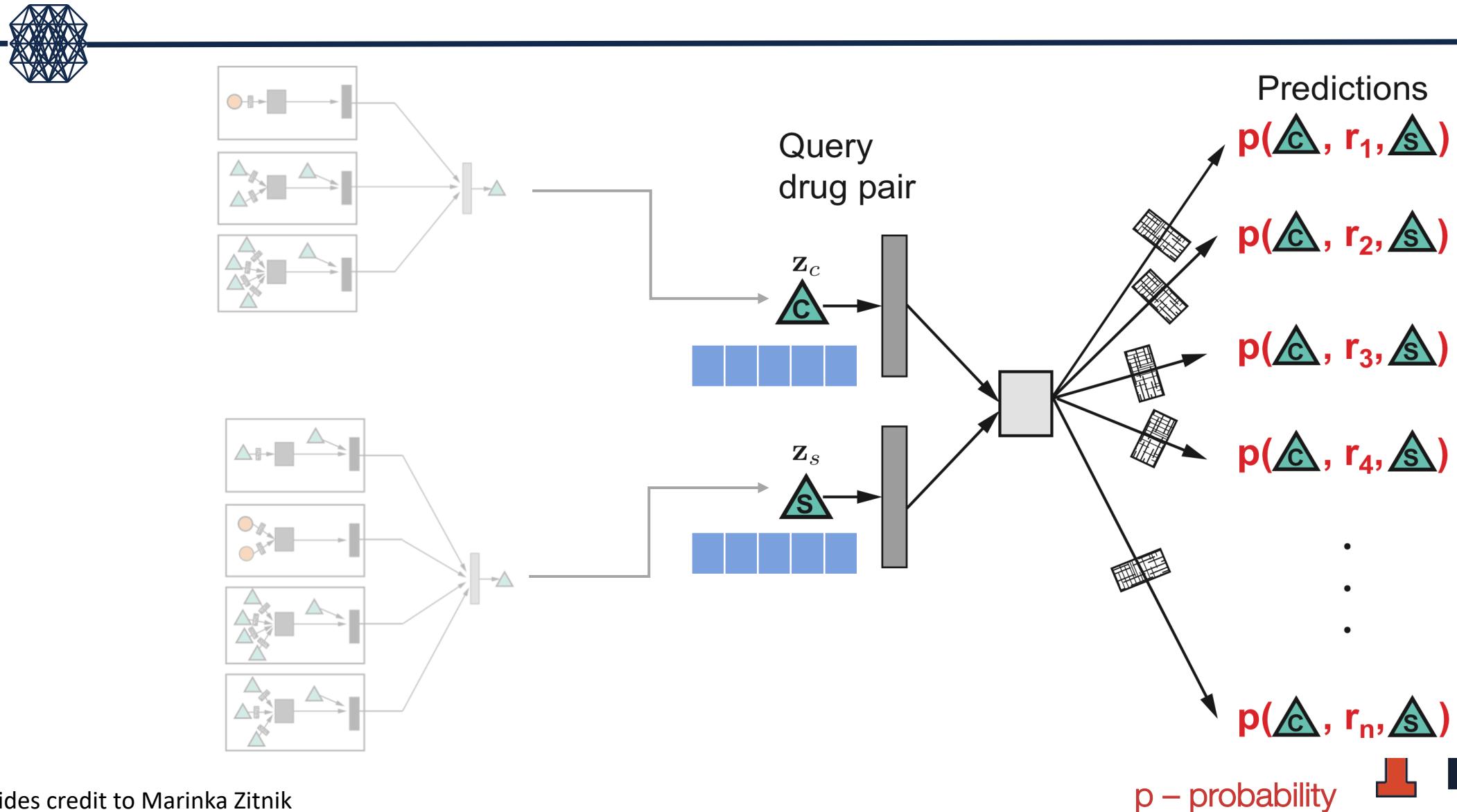


Encoder: Embeddings

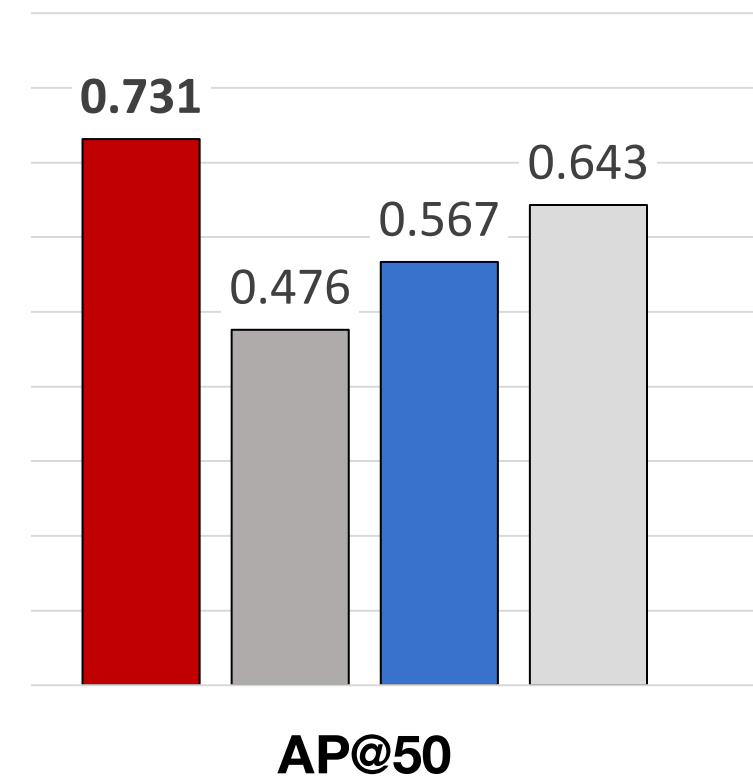
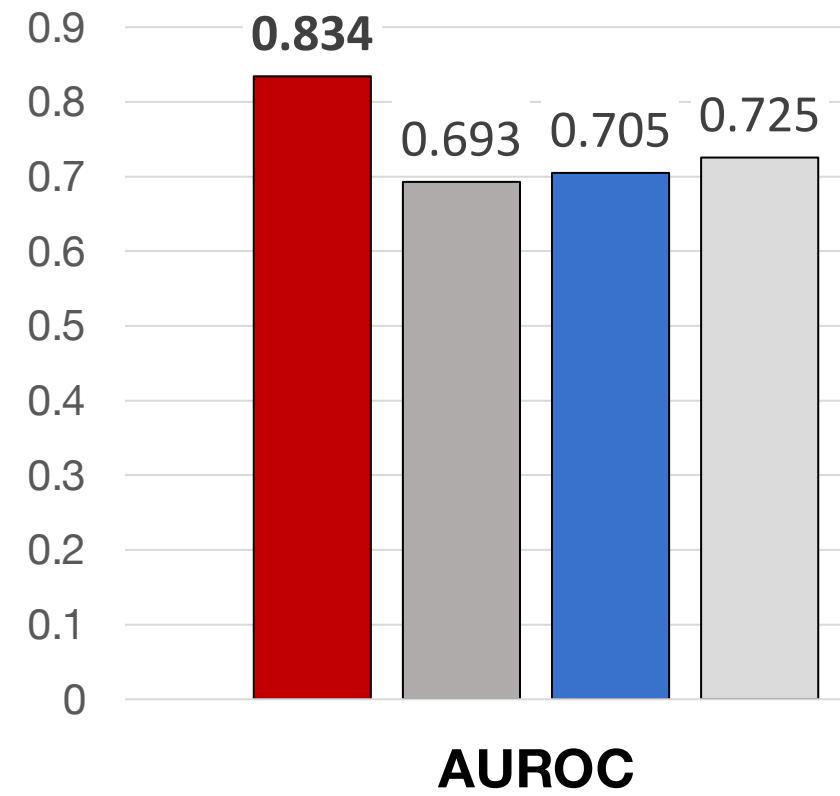


A batch of computation graphs

Decoder: Link Prediction



Results: Polypharmacy Side Effect Prediction



- Our method (Decagon)
- RESCAL Tensor Factorization [Nickel et al., ICML'11]
- Multi-relational Factorization [Perros, Papalexakis et al., KDD'17]
- Shallow Network Embedding [Zong et al., Bioinformatics'17]

New Predictions



Next: Can the method generate hypotheses and give:

- **Doctors** guidance on whether it is a good idea to prescribe a particular combination of drugs to a particular patient
- **Researchers** guidance on effective wet lab experiments and new drug therapies with fewer side effects

First AI method to predict side effects of drug combinations, even for combinations not yet used in patients

New Predictions



Approach:

- 1) Train deep model on data generated **prior to 2012**
- 2) How many **predictions** have been **confirmed after 2012?**

Rank	Drug	Drug	Side effect	Evidence found
1	Pyrimethamine	Aliskiren	Sarcoma	
2	Tigecycline	Bimatoprost	Autonomic neuropathy	
3	Telangiectases	Omeprazole	Dacarbazine	
4	Tolcapone	Pyrimethamine	Blood brain barrier	

Case Report

Severe Rhabdomyolysis due to Presumed Drug Interactions between Atorvastatin with Amlodipine and Ticagrelor

7	Anagrelide	Azelaic acid	Cerebral thrombosis
8	Atorvastatin	Amlodipine	Muscle inflammation
9	Aliskiren	Tioconazole	Breast inflammation
10	Estradiol	Nadolol	Endometriosis

A Deep Learning Approach to Antibiotic Discovery

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay,
James J. Collins

Stokes et al. 2020. "A Deep Learning Approach to Antibiotic Discovery." *Cell* 180 (4): 688–702.e13.

Antibiotic Discovery

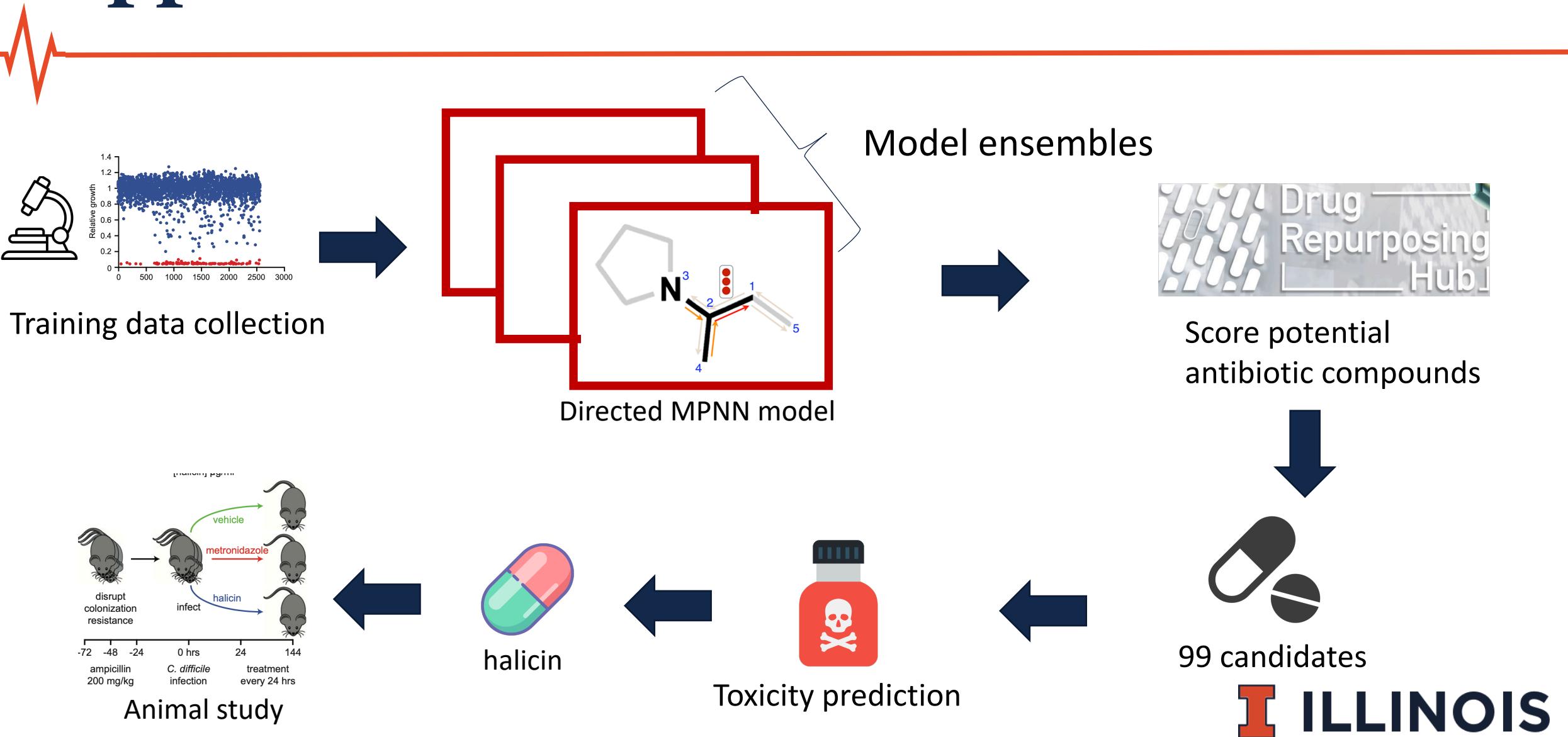


- Antibiotic drug is essential for modern medicine
- Continued efficacy of existing antibiotics is uncertain due to antibiotic-resistance determinants
- Lack of economic incentive for new antibiotic development
- The discovery of new antibiotics is becoming increasingly difficult

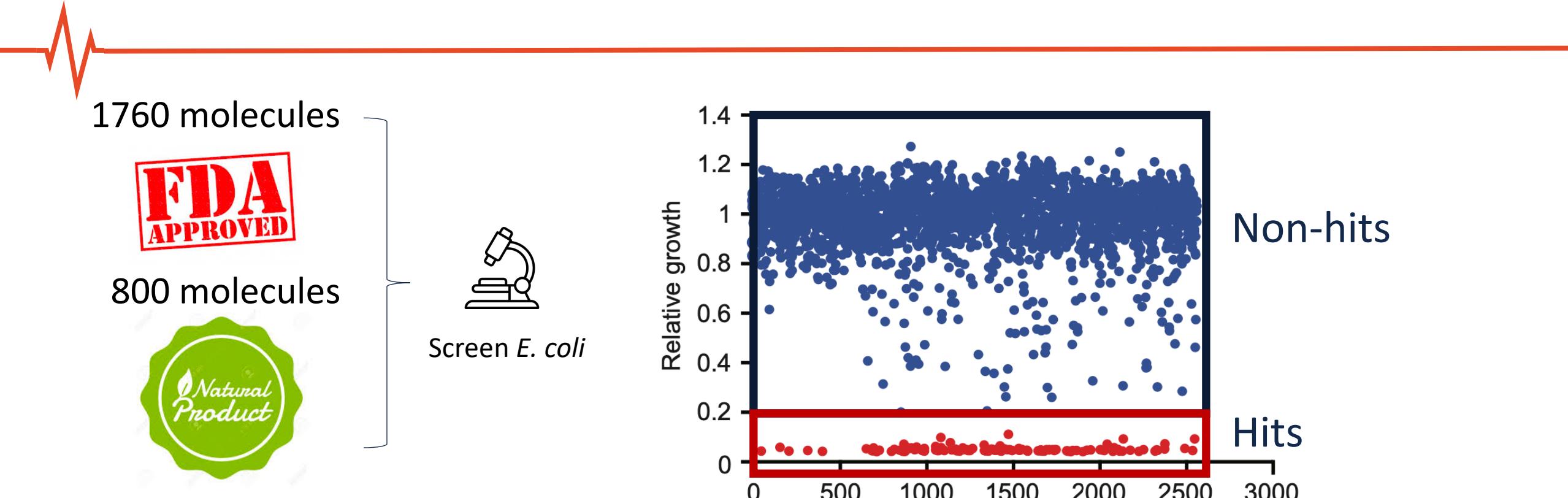
New approaches to antibiotic discovery are needed

Franco et al. 2009. “The Determinants of the Antibiotic Resistance Process.” *Infection and Drug Resistance* 2 (April): 1–11.

Approach



Data collection

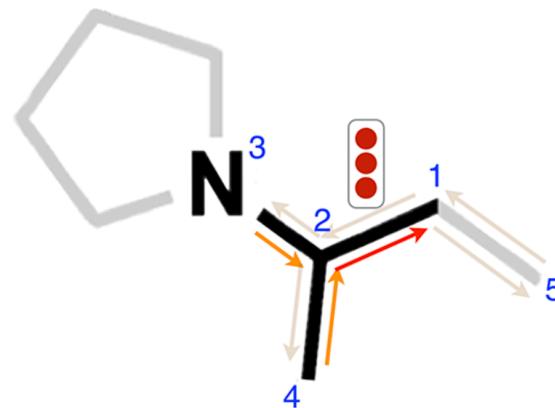


- 2,335 compounds from the primary training dataset were binarized as hit or non-hit.

Directed MPNN model

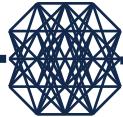


- Consider edge directions
- Update edge representation with neighboring edges
- Ignore the impact of reversal edges
 - Update h_{vw} from neighboring edges except the reversal h_{wv}



Yang, et al. 2019. "Analyzing Learned Molecular Representations for Property Prediction." *Journal of Chemical Information and Modeling* 59 (8): 3370–88.

Node and Edge features



Node features

feature	description	size
atom type	type of atom (ex. C, N, O), by atomic number	100
# bonds	number of bonds the atom is involved in	6
formal charge	integer electronic charge assigned to atom	5
chirality	unspecified, tetrahedral CW/CCW, or other	4
# Hs	number of bonded hydrogen atoms	5
hybridization	sp, sp ² , sp ³ , sp ³ d, or sp ³ d ²	5
aromaticity	whether this atom is part of an aromatic system	1
atomic mass	mass of the atom, divided by 100	1

Edge features

feature	description	size
bond type	single, double, triple, or aromatic	4
conjugated	whether the bond is conjugated	1
in ring	whether the bond is part of a ring	1
stereo	none, any, E/Z or cis/trans	6

Edge embedding update



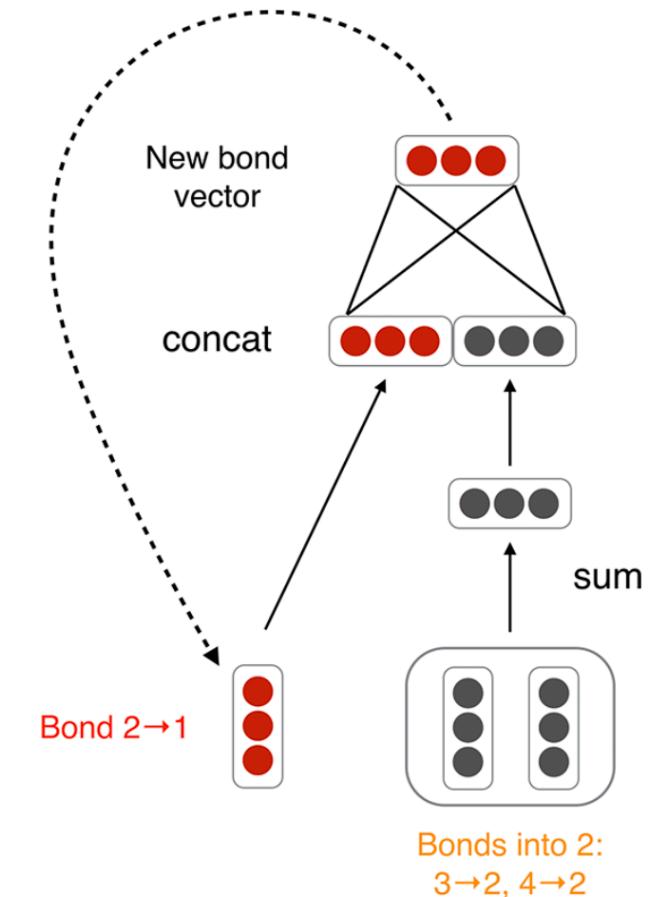
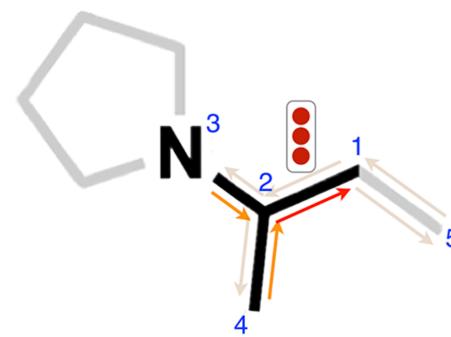
Directed message passing

$$m_{vw}^{t+1} = \sum_{k \in \{N(v) \setminus w\}} M_t(x_v, x_k, h_{kv}^t) = h_{vw}^t$$

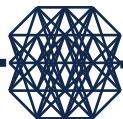
$$h_{vw}^{t+1} = U_t(h_{vw}^t, m_{vw}^{t+1}) = \text{ReLU}(h_{vw}^0 + W_m m_{vw}^{t+1})$$

Initialization

$$h_{vw}^0 = \tau(W_i \text{ cat}(x_v, e_{vw}))$$



Node and Graph embedding



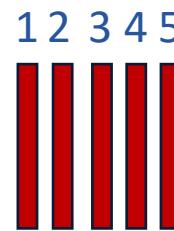
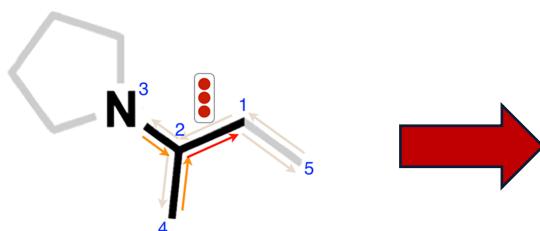
Node embedding updates

$$m_v = \sum_{k \in N(v)} h_{kv}^T$$

$$h_v = \tau(W_a \text{ cat}(x_v, m_v))$$

Readout operation:
graph embedding

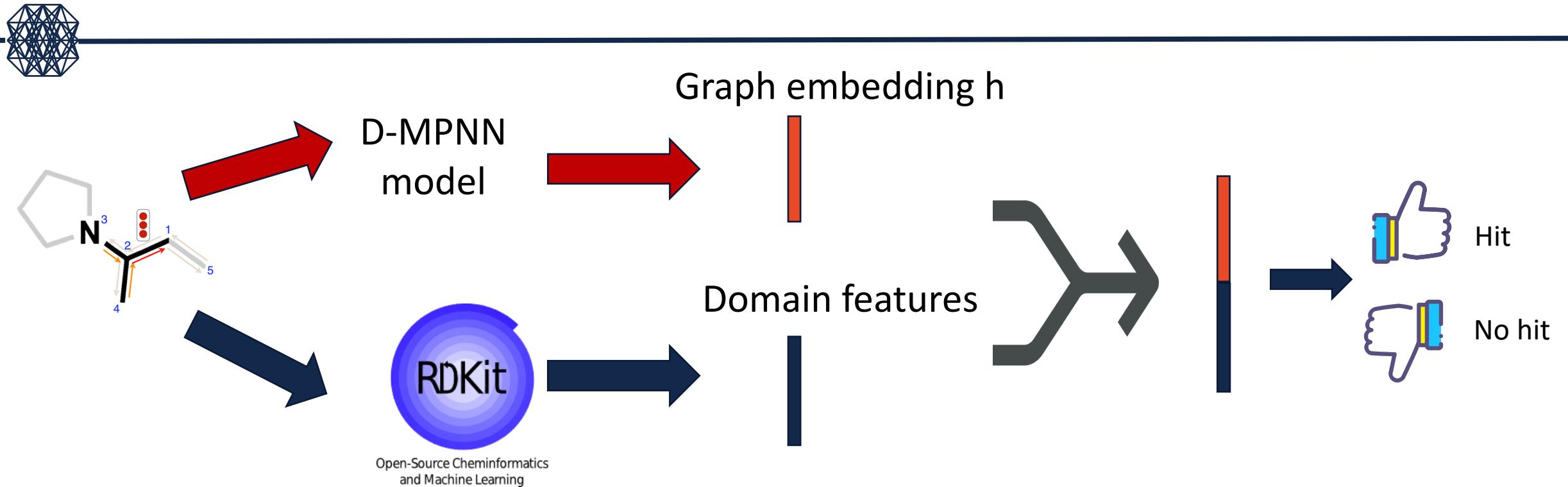
$$h = \sum_{v \in G} h_v$$



A large red arrow pointing to the right, indicating the final output of the readout operation.

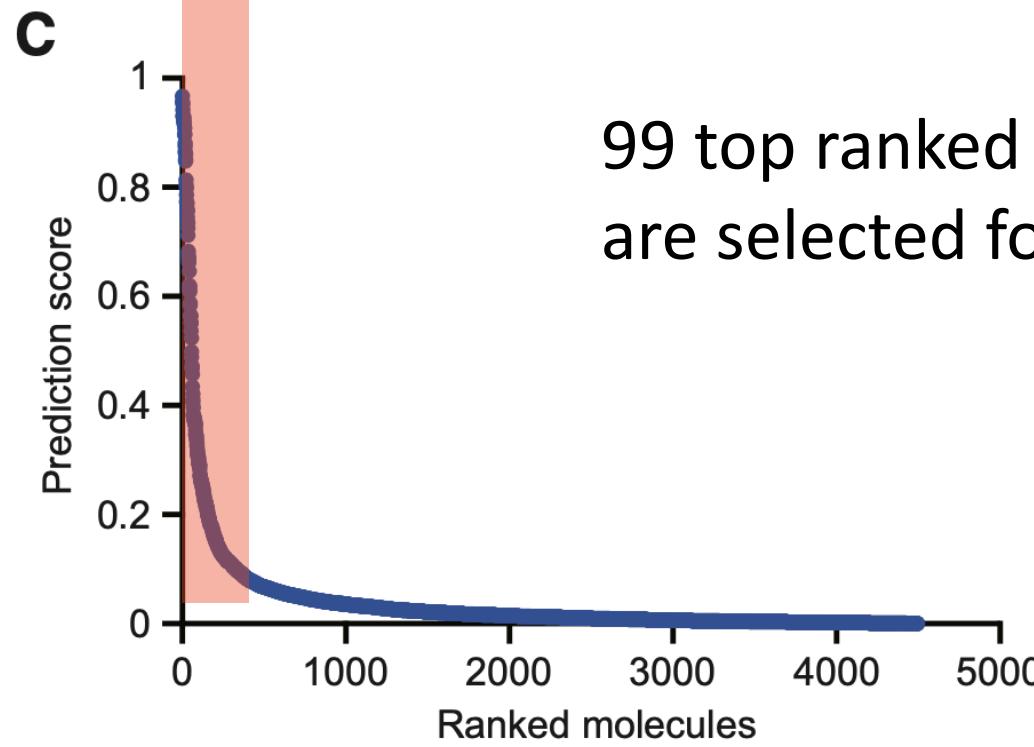
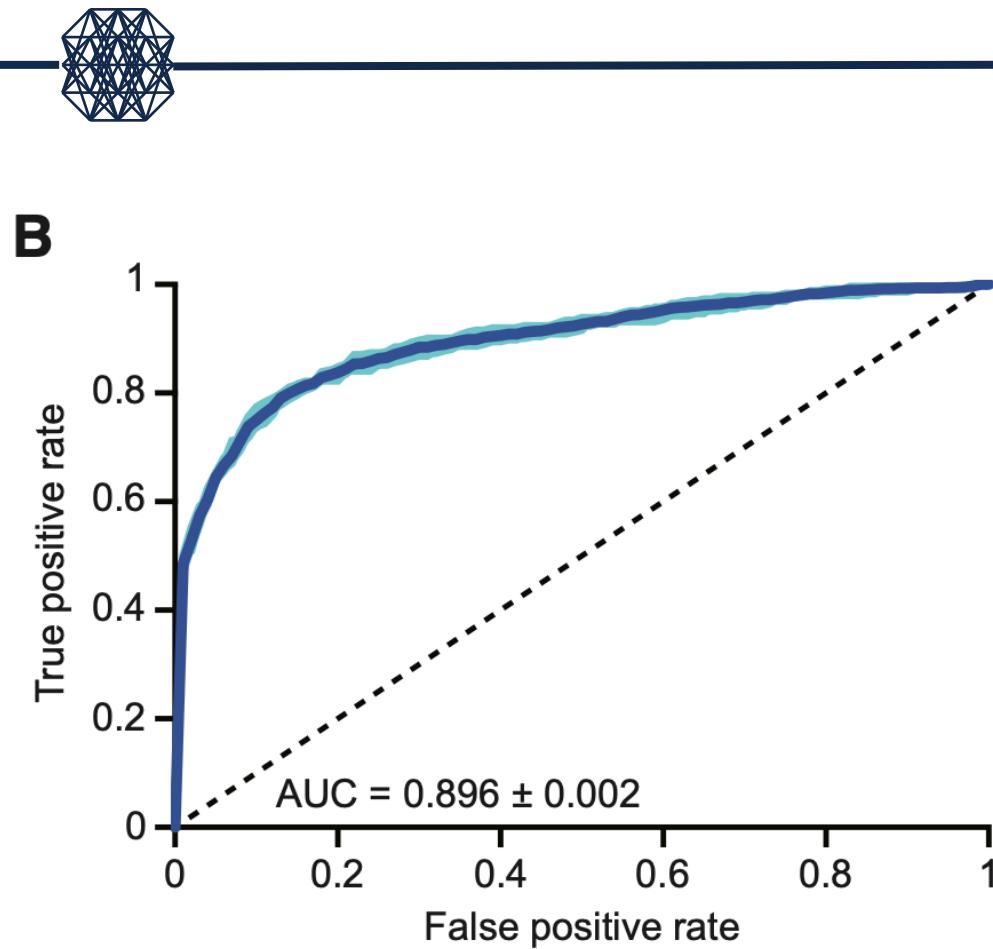
Hit or no hit?

Hit prediction

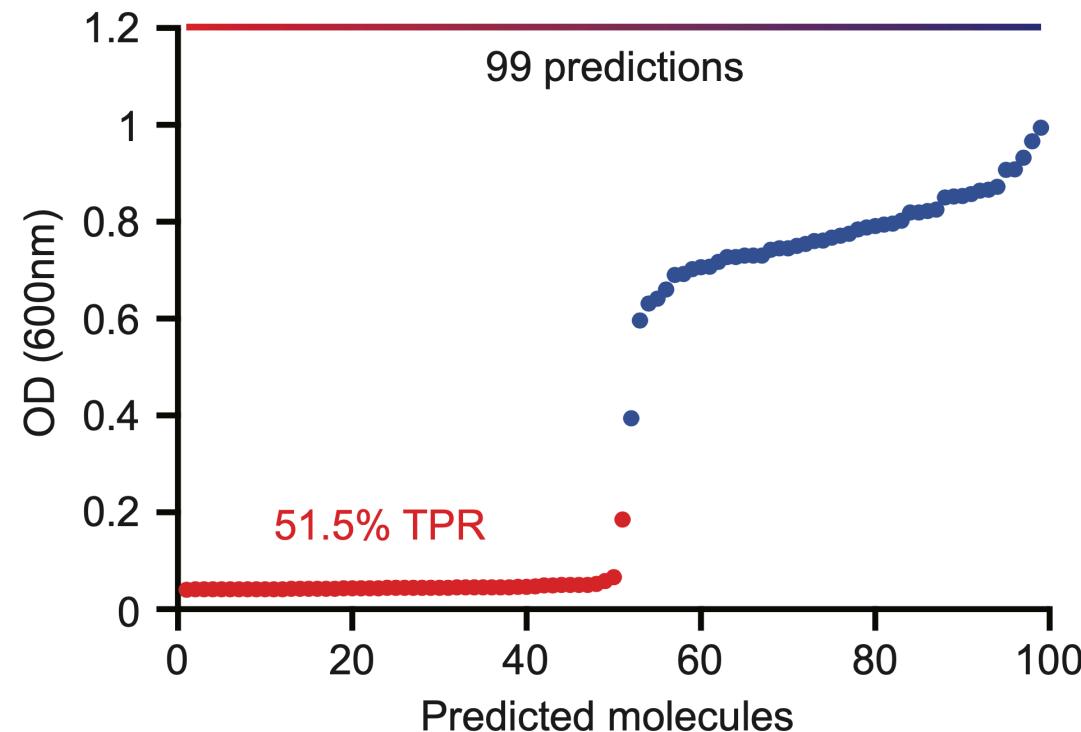


- 20-fold cross validation are used to select the best parameters
 - 80-10-10 split on train, validation and test
- An ensemble of 20 models are used for the final prediction

Result: Initial DL model performance

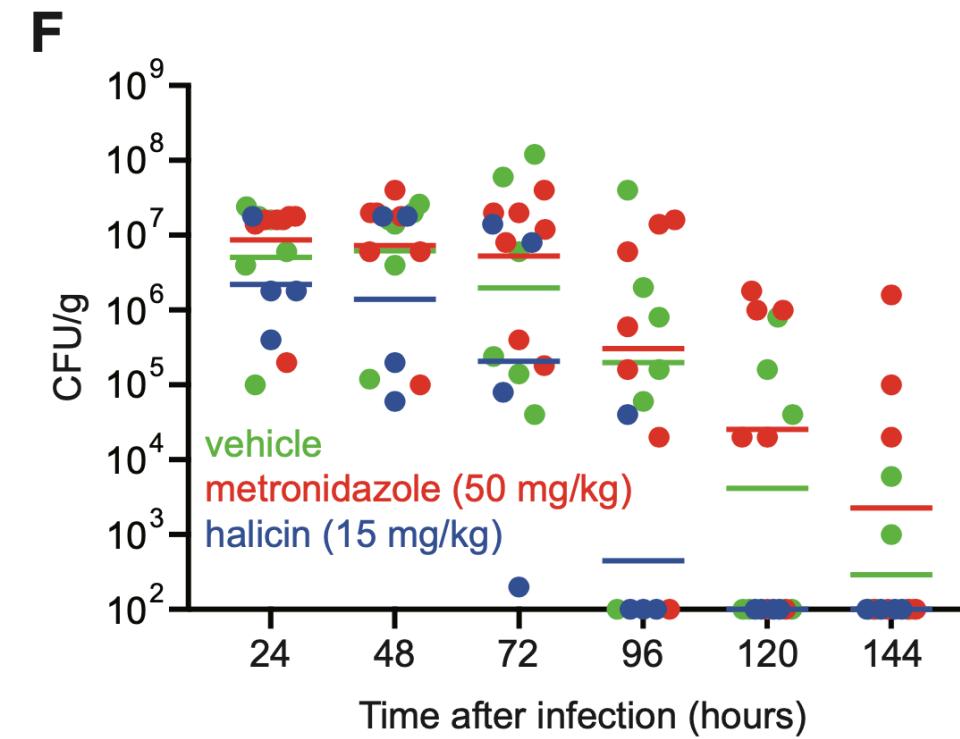
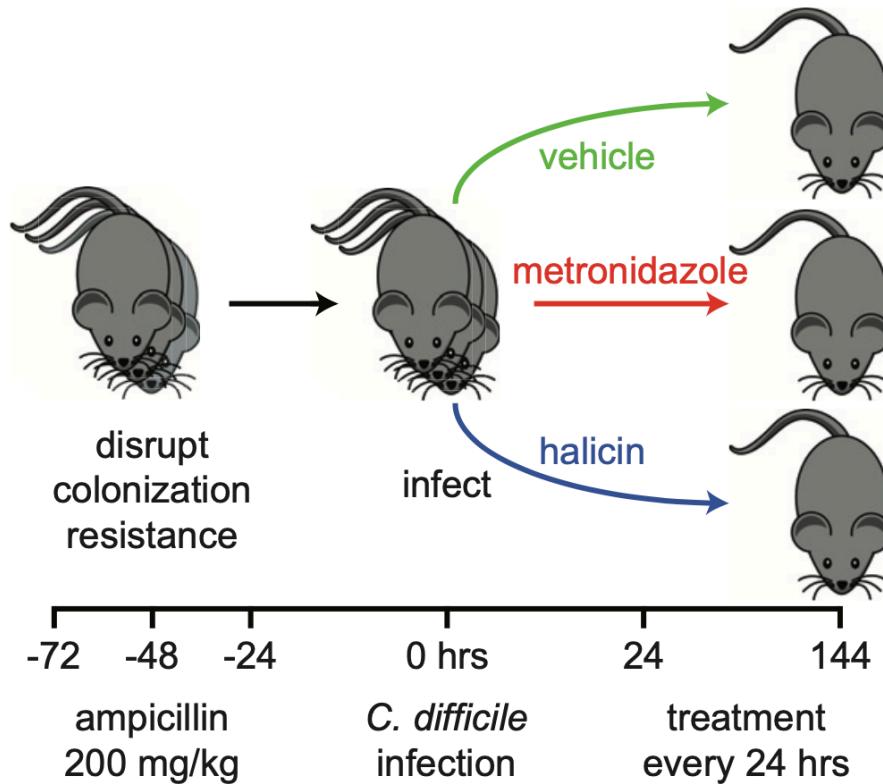


Result: Growth inhibition of *E. coli*



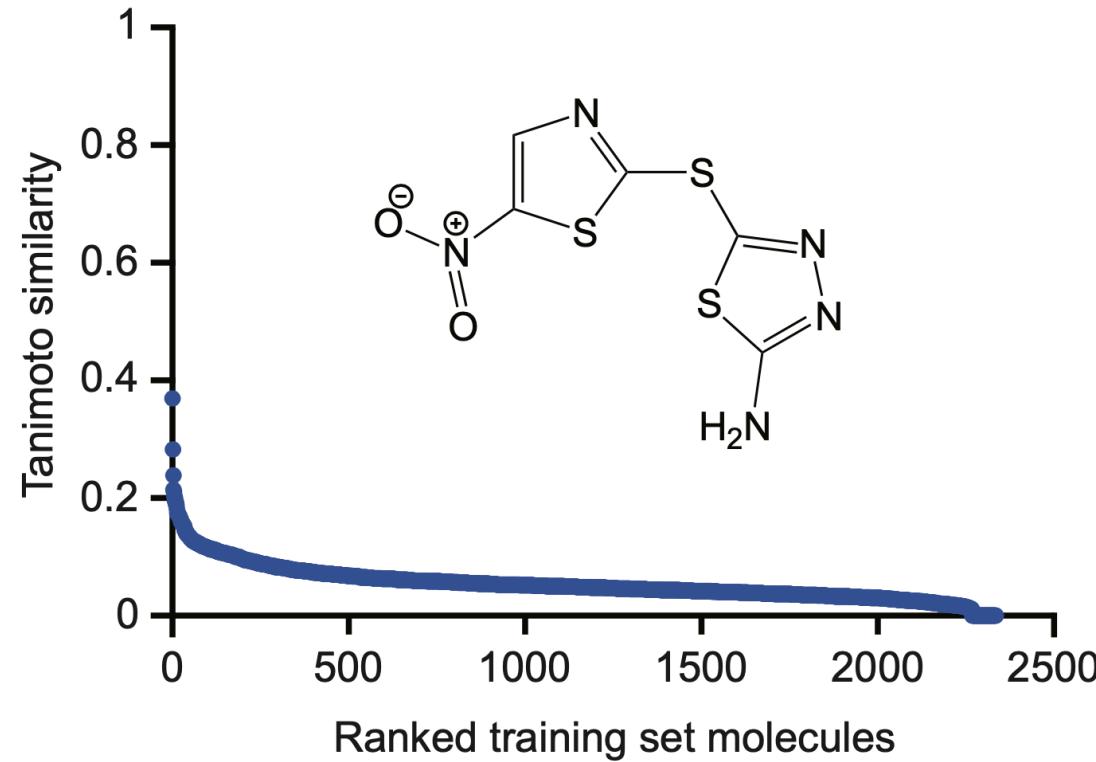
51.5% True Positive Rate

Result: Effectiveness of treating bacteria



- **Halicin** (the new antibiotic) works much better than existing antibiotic **Metronidazole**

Result: Structural diversity to known antibiotics



- The identified target Halicin is structurally different from existing antibiotics

Summary



- Fundamentals of deep learning on graphs ✓
- Graph convolutional networks (GCN) ✓
- Message passing neural network (MPNN) ✓
- Graph attention networks (GAT) ✓
- Applications
 - Side-effect prediction
 - Antibiotic discovery