Applications of Graph Neural Networks
Three topics for today:
1. GNN recommendation (PinSage)
2. Heterogeneous GNN (Decagon)
3. Goal-directed generation (GCPN)
PinSAGE: GNN for Recommender Systems
Recommender Systems

- **Users interacts with items**
  - Watch movies, buy merchandise, listen to music
- **Goal: Recommend items users might like**
  - Customer X buys Metallica and Megadeth CDs
  - Customer Y buys Megadeth, the recommender system suggests Metallica as well

![Diagram showing interactions between users and items]

“Interactions”

“You might also like”
Goal: Learn what items are related

- For a given query item(s) $Q$, return a set of similar items that we recommend to the user

Idea:

- User interacts with a set of items
- Formulate a query $Q$
- Search the items and return recommendations
Example: Pinterest

Query:

Chocolate Strawberry Shake
This healthier chocolate strawberry shake is like sipping a...
One Lovely Life
Danielle Benzaia
Strawberries
**Query:**

**Recommendations:**

- **Healthy Chocolate Strawberry Shake**
  - Description: This healthier chocolate strawberry shake is like sipping a...
  - Source: One Lovely Life
  - Likes: 249

- **Chocolate Dipped Strawberry Smoothie**
  - Description: Chocolate Dipped Strawberry Smoothie. Just in time for...
  - Source: Be Whole. Be You.
  - Likes: 5.3k

- **Tropical Orange Smoothie**
  - Source: Ed Todd Drinks Smoothies
  - Likes: 80.1k

- **8 Staple Smoothies (That You Should Know How To Make)**
  - Likes: 5.2k

- **Quick & Nutritious Vanilla Pumpkin Smoothie**
  - Likes: 11.4k

- **Spinach-Pear-Celery Smoothie**
  - Description: Drink this daily and watch the pounds come off without fuss...
  - Source: Greenreset.com
  - Likes: 60

- **Easy Breezy Tropical Orange Smoothie**
  - Source: Marybeth @ Bab... Best Comfort Food...
  - Likes: 80k

Example (2): Pinterest

Query:

Chocolate Strawberry Shake
This healthier chocolate strawberry shake is like sipping a...
One Lovely Life

Healthy Chocolate Peanut Butter Chip Muffins
Healthy Chocolate Peanut Butter Chip Muffins made with greek...
The First Year

Healthy Chocolate Peanut Butter Chip Cookies
The ULTIMATE Healthy Chocolate Chip Cookies -- so buttery...
Amy's Healthy Baking

Healthy Recipes

The Ultimate Healthy Soft & Chewy Chocolate Chip Cookies
Robin Guerin

healthy cooking
Query:

Healthy Chocolate Strawberry Shake
This healthier chocolate strawberry shake is like sipping a...
One Lovely Life
Danielle Benzaia
Strawberries

HEALTHY CHOCOLATE PEANUT BUTTER CHIP MUFFINS

Healthy Chocolate Peanut Butter Chip Muffins
Healthy Chocolate Peanut Butter Chip Muffins made with greek...
The First Year
Katie - You Brew... Healthy Recipes

Recommendations:

Skinny Banana Chocolate Chip Muffins
Almost fat free, healthy, satisfying chocolate chip... Amorfose Kitchen

The Ultimate Healthy Soft & Chewy Chocolate Chip Cookies
The ULTIMATE Healthy Chocolate Chip Cookies -- so buttery... Amy's Healthy Baking
Robyn Guerin

Healthy Peanut Butter Chocolate Chip Oatmeal Bars
Portion Control Peanut Butter Chocolate Chip Oatmeal Bar

Healthy Peanut Butter Chocolate Chip Oatmeal Bars

Healthy Peanut Butter Chocolate Chip Oatmeal Bars

Chocolate Dipped Strawberry Hearts
Chocolate Dipped Strawberries. Just in time for Valentine's Day

Chocolate Dipped Strawberry Hearts

Dark Chocolate Sea Salt Amorites
Sea Salt Amorites

Dark Chocolate Sea Salt Amorites

Healthy Chocolate Chip Cookie Dough Bites

Healthy Chocolate Chip Cookie Dough Bites

Easy Apple Pie Bars
With 4 ingredients, this Easy Apple Pie is a healthier version of the classic apple pie! Vegan, gluten-free, refined sugar-free.

Easy Apple Pie Bars

Tropical Orange Smoothie

COPYCAT Cinammons Cinnamon Rolls

Quick & Delicious Vanilla Pumpkin Smoothie

Quick & Delicious Vanilla Pumpkin Smoothie

Skinny Raspberry Greek Yogurt Smoothie
Having a universal similarity function allows for many applications:

- **Homefeed** (endless feed of recommendations)
- **Related pins** (find most similar/related pins)
- **Ads and shopping** (use organic for the query and search the ads database)
Question: How do we define similarity?

1) Content-based: User and item features, in the form of images, text, categories, etc.

2) Graph-based: User-item interactions, in the form of graph/network structure
   - This is called collaborative filtering:
     - For a given user X, find others who liked similar items
     - Estimate what X will like based on what similar others like
How do we define similarity:

- **(1) Gathering “known” similarities**
  - How to collect the data about what users like

- **(2) Extrapolating unknown similarities from the known ones**
  - Mainly interested in high unknown similarities
  - We are not interested in knowing what you don’t like but what you like

- **(3) Evaluating methods**
  - How to measure success/performance of recommendation methods
- 300M users
- 4+B pins, 2+B boards
Pinterest: Human curated collection of pins

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

**Pin:** Image, text, link

**Board:** A collection of ideas (pins having something in common)
Two sources of signal:

Features:
- Image and text of each pin

Graph:
- Graph is dynamic: Need to apply to new nodes without model retraining
Goal: Learn embeddings for items

- **Related Pins Query**: Which pin to recommend when a user interacts with a pin \( v_3 \)?
- **Answer**: Find the closest embedding (\( v_4 \)) to \( v_3 \) by nearest neighbor. Recommend it.
Recommendations via Embeddings

- **Goal 1**: Efficiently learn embeddings for billions of pins (items, nodes) using neural networks
- **Goal 2**: Perform nearest neighbor query to recommend items in real-time

![Diagram showing query pin and related pin in embedding space, with text indicating the closer the embeddings are, the more similar the pins are]
Overview: Pin Recommendation

**Task:** Recommend related pins to users

**Query pin**

**Task:** Learn node embeddings $z_i$ such that

$$d(z_{\text{cake1}}, z_{\text{cake2}}) < d(z_{\text{cake1}}, z_{\text{sweater}})$$

**Predict whether two nodes in a graph are related**

$d(z_1, z_2)$
PinSage: Graph Neural Networks

Predict whether two nodes in a graph are related

Approach:
- Pins have embeddings at each layer
- Layer-0 embedding of a node are its features:
  - Text, image, ...

12/5/19
PinSage: Why it Works

- **PinSage** graph convolutional network:
  - **Goal:** Generate embeddings for nodes (e.g., pins) in the 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
  - Pin embeddings are essential to many different tasks. Aside from the “Related Pins” task, it can also be used in:
    - Recommend related ads
    - Homefeed recommendation
    - Cluster users by their interest
1. **Collect** billions of training pairs from logs.
   - **Positive pair:** Two pins that are *consecutively saved into the same board* within a time interval (1 hour)
   - **Negative pair:** A random pair of 2 pins
     - With high probability the pins are not on the same board
1. **Collect** billions of training pairs from logs.
   - **Positive pair:** Two pins that are *consecutively saved into the same board* within a time interval (1 hour)
   - **Negative pair:** A random pair of 2 pins
     - With high probability the pins are not on the same 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.
Train so that **pins that are consecutively pinned have similar embeddings**

**Max-margin loss:**

\[
\mathcal{L} = \sum_{(u,v) \in D} \max(0, -z_u^T z_v + z_u^T z_m + \Delta)
\]

- **Set of training pairs from user logs**
- "positive"/true training pair
- "negative" example
- "margin" (i.e., how much larger positive pair similarity should be compared to negative)
Key Innovation (1)

- **Four key innovations:**
  1. **On-the-fly graph convolutions**
     - Sample the neighborhood around a node and dynamically construct a computation graph

Minibatch of neighborhoods
Key Innovation (1)

- Four key innovations:
  1. On-the-fly graph convolutions
     - Perform a localized graph convolution around a particular node
     - Does not need the entire graph during training

At every iteration, only source node embeddings are computed
Four key innovations:

2. Selecting neighbors via random walks
   - Performing aggregation on all neighbors is infeasible:
     - How to select the set of neighbors of a node to convolve over?
   - Personalized PageRank can help!
   - Define Importance pooling: Define importance-based neighborhoods by simulating random walks and selecting the neighbors with the highest visit counts
Key Innovation (2): Random Walks

- **Proximity to query node(s) $Q$**

```python
ALPHA = 0.5
QUERY_NODES = { }  
for i in range(N_STEPS):
    board_node = pin_node.get_random_neighbor()
    pin_node = board_node.get_random_neighbor()
    pin_node.visit_count += 1
    if random() < ALPHA:
        pin_node = QUERY_NODES.sample_by_weight()
```

---

5 5 5 5 5 5 14 9 Q 16 7 8 8 8 8 1 1 1

Yummm  Strawberries  Smoothies  Smoothie Madness!
Key Innovation (2): Random Walks

- Proximity to query node(s) $Q$
- Importance pooling
  - Choose nodes with top $K$ visit counts
  - Pool over the chosen nodes
  - The chosen nodes are not necessarily neighbors

5 5 5 5 5 5 14 9 Q 16 7 8 8 8 8 8 1 1 1

Yummm  Strawberries  Smoothies  Smoothie Madness!
- **Example:** suppose $K = 5$
- Rank nodes based on Random Walk visit counts
- Pick **top $K$ nodes** and normalize counts
  
  $$
  \begin{align*}
  16 & \quad 14 & \quad 9 & \quad 8 & \quad 8 \\
  55' & \quad 55' & \quad 55' & \quad 55' & \quad 55
  \end{align*}
  $$

- Aggregate messages from the top $K$ nodes
Key Innovation (2): Importance Pooling

- Pick top K nodes and normalize counts
  
  \[
  \begin{array}{cccccc}
  16 & 14 & 9 & 8 & 8 \\
  \end{array}
  \]
  
  \[
  \frac{16}{55} \quad \frac{14}{55} \quad \frac{9}{55} \quad \frac{8}{55} \quad \frac{8}{55}
  \]

- **GraphSAGE mean pooling**
  - Average the messages from direct neighbors

- **PinSAGE Importance pooling**
  - Use the normalized counts as weights for weighted mean of messages from the top K nodes

- **PinSAGE uses** $K = 50$
  - Negligible performance gain for $K > 50$
Four key innovations:

3. Efficient MapReduce inference
   - **Problem:** Many repeated computation if using localized graph convolution at inference step
   - Need to avoid repeated computation
Recall how we obtain negative examples

\[ \mathcal{L} = \sum_{(u,v) \in D} \max(0, -z_u^T z_v + z_u^T z_n + \Delta) \]

- **set of training pairs from logs**
- **“positive”/true example**
- **“negative” example**
- **“margin” (i.e., how much larger positive pair similarity should be compared to negative)**

Positive Example  
Random Negative
Goal: Identify target pin among 3B pins

- **Issue:** Need to learn with resolution of 100 vs. 3B
- **Massive size:** 3 billion nodes, 20 billion edges
- **Idea:** Use harder and harder negative samples

\[
\mathcal{L} = \sum_{(u,v) \in D} \max(0, -z_u^T z_v + z_u^T z_n + \Delta)
\]

- **set of training pairs from logs**
- **“positive”/true example**
- **negative examples**
- **“margin” (i.e., how much larger positive pair similarity should be compared to negative)**

Force model to learn subtle distinctions between pins
Key Innovation (4)

- **Hard negative examples** improve performance

  Positive pair

  Positive Example

  Random Negative

  Query

  Hard Negative

  Harder to distinguish from the positive pair

- **How to obtain hard negatives:** Use **random walks**:
  - Use nodes with visit counts ranked at 1000-5000 as hard negatives
  - Have something in common, but are not too similar
Hard negative examples improve performance

Curriculum training on hard negatives
- Start with random negative examples
- Provide harder negative examples over time
PinSage: Experiments

Related Pin recommendations

- Given a user just saved pin $Q$, predict what pin $X$ are they going to save next
- **Setup**: Embed 3B pins, find nearest neighbors of $Q$

**Baseline embeddings:**

- **Visual**: VGG visual embeddings
- **Annotation**: Word2vec embeddings
- **Combined**: Concatenate embeddings

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit-rate</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual</td>
<td>17%</td>
<td>0.23</td>
</tr>
<tr>
<td>Annotation</td>
<td>14%</td>
<td>0.19</td>
</tr>
<tr>
<td>Combined</td>
<td>27%</td>
<td>0.37</td>
</tr>
<tr>
<td>max-pooling</td>
<td>39%</td>
<td>0.37</td>
</tr>
<tr>
<td>mean-pooling</td>
<td>41%</td>
<td>0.51</td>
</tr>
<tr>
<td>mean-pooling-xent</td>
<td>29%</td>
<td>0.35</td>
</tr>
<tr>
<td>mean-pooling-hard</td>
<td>46%</td>
<td>0.56</td>
</tr>
<tr>
<td>PinSage</td>
<td>67%</td>
<td>0.59</td>
</tr>
</tbody>
</table>

**MRR**: Mean reciprocal rank of the positive example $X$ w.r.t $Q$

**Hit rate**: Fraction of times the positive example $X$ is among top K closest to $Q$
**Example Pin Recommendations**

**Pixie (graph-based):** the method of simulating random walks starting at query Pin using the Pixie algorithm in class. Items with top scores are retrieved as recommendations.

**Visual, Annot. (feature-based):** nearest neighbor recommendation using visual (CNN) and annotation features of pins.
Comparing against Prod (1)
Comparing against Prod (2)

Query

PinSAGE
1. GNN recommendation (PinSage)
2. Heterogeneous GNN (Decagon)
3. Goal-directed generation (GCPN)
DECAGON: Heterogeneous GNN
So far we only applied GNNs to simple graphs
- GNNs do not explicitly use node and edge type information
- Real networks are often heterogeneous
- How to use GNN for heterogeneous graphs?
Polypharmacy Side Effects

Polypharmacy: use multiple drugs for a disease
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
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
How likely with a pair of drugs $c, d$ lead to side effect $r$?

Model and predict side effects of drug pairs
Heterogeneous (multimodal) graphs: graphs with different node types and/or edge types.
Goal: Given a partially observed graph, predict labeled edges between drug nodes

Query: Given a drug pair $c, d$, how likely does an edge $(c, r_2, d)$ exist?

Co-prescribed drugs $c$ and $d$ lead to side effect $r_2$
Task Description

- 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 \(r_2\)

Predictions:
**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

GNN for Edge type: $r_1$

GNN for Edge type: $r_2$

GNN for Edge type: drug-target

---

Diagram showing the GNN model with different edge types and their corresponding features.
Making Edge Predictions

- **Key Insight:** 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

Input:
- Node embeddings of query drug pairs

Output:
- Predicted edges
Decoder: Link Prediction

Predictions

\[ p(C, r_1, S) \]
\[ p(C, r_2, S) \]
\[ p(C, r_3, S) \]
\[ p(C, r_4, S) \]
\[ \vdots \]
\[ p(C, r_n, S) \]

Query drug pair

\[ z_C \]
\[ z_S \]

p – probability
Experiment Setup

- **Data:**
  - **Graph over Molecules:** protein-protein interaction and drug target relationships
  - **Graph over Population:** Side effects of individual drugs, polypharmacy side effects of drug combinations

- **Setup:**
  - Construct a heterogeneous graph of all the data
  - **Train:** Fit a model to predict **known associations** of drug pairs and polypharmacy side effects
  - **Test:** Given a query drug pair, predict **candidate polypharmacy side effects**
## Prediction Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>AUROC</th>
<th>AUPRC</th>
<th>AP@50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decagon (3-layer)</td>
<td>0.834</td>
<td>0.776</td>
<td>0.731</td>
</tr>
<tr>
<td>Decagon (2-layer)</td>
<td>0.809</td>
<td>0.762</td>
<td>0.713</td>
</tr>
<tr>
<td>RESCAL</td>
<td>0.693</td>
<td>0.613</td>
<td>0.476</td>
</tr>
<tr>
<td>Node2vec</td>
<td>0.725</td>
<td>0.708</td>
<td>0.643</td>
</tr>
<tr>
<td>Drug features</td>
<td>0.736</td>
<td>0.722</td>
<td>0.679</td>
</tr>
</tbody>
</table>

- Up to **54% improvement** over baselines
- **First opportunity** to computationally flag polypharmacy side effects for follow-up analyses
### De novo Predictions

<table>
<thead>
<tr>
<th>Rank</th>
<th>Drug $c$</th>
<th>Drug $d$</th>
<th>Side effect $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pyrimethamine</td>
<td>Aliskiren</td>
<td>Sarcoma</td>
</tr>
<tr>
<td>2</td>
<td>Tigecycline</td>
<td>Bimatoprost</td>
<td>Autonomic neuropathy</td>
</tr>
<tr>
<td>3</td>
<td>Omeprazole</td>
<td>Dacarbazine</td>
<td>Telangiectases</td>
</tr>
<tr>
<td>4</td>
<td>Tolcapone</td>
<td>Pyrimethamine</td>
<td>Breast disorder</td>
</tr>
<tr>
<td>5</td>
<td>Minoxidil</td>
<td>Paricalcitol</td>
<td>Cluster headache</td>
</tr>
<tr>
<td>6</td>
<td>Omeprazole</td>
<td>Amoxicillin</td>
<td>Renal tubular acidosis</td>
</tr>
<tr>
<td>7</td>
<td>Anagrelide</td>
<td>Azelaic acid</td>
<td>Cerebral thrombosis</td>
</tr>
<tr>
<td>8</td>
<td>Atorvastatin</td>
<td>Amlodipine</td>
<td>Muscle inflammation</td>
</tr>
<tr>
<td>9</td>
<td>Aliskiren</td>
<td>Tioconazole</td>
<td>Breast inflammation</td>
</tr>
<tr>
<td>10</td>
<td>Estradiol</td>
<td>Nadolol</td>
<td>Endometriosis</td>
</tr>
</tbody>
</table>
## De novo Predictions

<table>
<thead>
<tr>
<th>Rank</th>
<th>Drug $c$</th>
<th>Drug $d$</th>
<th>Side effect $r$</th>
<th>Evidence found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pyrimethamine</td>
<td>Aliskiren</td>
<td>Sarcoma</td>
<td>Stage et al. 2015</td>
</tr>
<tr>
<td>2</td>
<td>Tigecycline</td>
<td>Bimatoprost</td>
<td>Autonomic dysreflexia</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Omeprazole</td>
<td>Dacarbazine</td>
<td>Telangiectasia</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tolcapone</td>
<td>Pyrimethamine</td>
<td>Breast discharge</td>
<td>Bicker et al. 2017</td>
</tr>
<tr>
<td>5</td>
<td>Minoxidil</td>
<td>Paricalcitol</td>
<td>Cluster headache</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Anagrelide</td>
<td>Azelaic acid</td>
<td>Cerebral thrombosis</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Atorvastatin</td>
<td>Amlodipine</td>
<td>Muscle inflammation</td>
<td>Banakh et al. 2017</td>
</tr>
<tr>
<td>9</td>
<td>Aliskiren</td>
<td>Tioconazole</td>
<td>Breast inflammation</td>
<td>Parving et al. 2012</td>
</tr>
<tr>
<td>10</td>
<td>Estradiol</td>
<td>Nadolol</td>
<td>Endometriosis</td>
<td></td>
</tr>
</tbody>
</table>

### Case Report

**Severe Rhabdomyolysis due to Presumed Drug Interactions between Atorvastatin with Amlodipine and Ticagrelor**
1. GNN recommendation (PinSage)  
2. Heterogeneous GNN (Decagon)  
3. Goal-directed generation (GCPN)
GCPN: Goal-Directed Graph Generation (an extension of GraphRNN)


Recap: Graph Generative Models

- **Given**: Graphs sampled from $p_{data}(G)$
- **Goal**:
  - Learn the distribution $p_{model}(G)$
  - Sample from $p_{model}(G)$

$p_{data}(G)$

Learn & Sample

$p_{model}(G)$
Generating graphs via sequentially adding nodes and edges
Quick Summary of GraphRNN:

- Generate a graph by generating a two level sequence
- Use RNN to generate the sequences
Imitating Given Graphs

Grid

Training

GraphRNN

Baselines

(Kronecker) (MMSB) (B-A)
Imitating Given Graphs

Can we do more than imitating given graphs?

(Kronecker)  (MMSB)  (B-A)
**Question:** Can we learn a model that can generate **valid** and **realistic** molecules with **high** value of a given chemical property?

**Model** ➔ **output** ➔ **that optimizes** ➔ **Property**

**e.g., drug likeness = 0.95**

---

Molecules as Heterogenous Graphs

- **Node types**: C, N, O, ...
- **Edge types**: single bond, double bond, ...
- **Note**: “H”s can be automatically inferred via chemical validity rules, thus are ignored in molecular graphs
Generating 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)**
  - e.g., Imitating a molecule graph dataset

Generating graphs that:

- **Optimize a given objective** (High scores)
  - e.g., drug-likeness
- **Obey underlying rules** (Valid)
  - e.g., chemical validity rules

Including “Black-box” in ML:

Objectives like drug-likeness are governed by physical law, which are assumed to be unknown to us!

---

Solution: Reinforcement Learning

- A 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:** Environment is a **blackbox** to the agent
**Policy-based RL**

- **Policy**: Agent behavior, which maps observation to action
- **Policy-based RL**: An agent directly learns an optimal policy from data
Graph Convolutional Policy Network combines graph representation + RL:

- **Graph Neural Network** captures complex structural information, and enables validity check in each state transition *(Valid)*
- **Reinforcement learning** optimizes intermediate/final rewards *(High scores)*
- **Adversarial training** imitates examples in given datasets *(Realistic)*
Overview of GCPN

- (a) Insert nodes/scaffolds
- (b) Compute state via GCN
- (c) Sample next action
- (d) Take action (check chemical validity)
- (e, f) Compute reward
Learn to take valid action
- At each step, assign small positive reward for valid action

Optimize desired properties
- At the end, assign positive reward for high desired property

Generate realistic graphs
- At the end, adversarially train a GCN discriminator, compute adversarial rewards that encourage realistic molecule graphs
Reward: $r_t = \text{Final reward} + \text{Step reward}$

- **Final reward** = Domain-specific reward
- **Step rewards** = Step-wise validity reward
How Do We Train?

- **Two parts:**
  - (1) **Supervised training:** Train policy by imitating the action given by real observed graphs. Use gradient.
  - (2) **RL training:** Train policy to optimize rewards. Use standard policy gradient algorithm (refer to any RL course, e.g., CS234).
GCPN Architecture

Graph $G_t$ → GCPN → Gradient

Query dataset → Real graph $G^*_t+1$ → Cross entropy loss

Generated graph $G^*_t+1$ → Final reward

Graph $G_t$ → Generated graph $G^*_t$ → Environment → Step reward

Policy gradient

Supervised Training

RL Training
GCPN Architecture

- **Gradient**
  - Query dataset
  - Graph $G_t$
  - Run one step
  - Run until stop

- **Validity**
  - Real graph $G_{t+1}^*$
  - Generated graph $G_{t+1}^*$
  - Cross entropy loss: 0.6

- **Score**
  - Step reward: 0.1
  - Final reward: 1

- **Realistic**
  - Adversarial reward: 0.3

- **Environment**
  - Discriminator

**Policy gradient**

- Supervised Training
- RL Training
GCPN: Tasks

- Property optimization
  - Generate molecules with high specified property score
- Property targeting
  - Generate molecules whose specified property score falls within given range
- Constrained property optimization
  - Edit a given molecule for a few steps to achieve higher specified property score
Data and Baselines

- **ZINC250k dataset**
  - 250,000 drug like molecules whose maximum atom number is 38

- **Baselines:**
  - **ORGAN:** String representation + RL  
    [Guimaraes et al., 2017]
  - **JT-VAE:** VAE-based vector representation + Bayesian optimization  
    [Jin et al., 2018]
Quantitative Results

Property optimization

- +60% higher property scores

Table 1: Comparison of the top 3 property scores of generated molecules found by each model

<table>
<thead>
<tr>
<th>Method</th>
<th>Penalized logP</th>
<th>QED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>ZINC</td>
<td>4.52</td>
<td>4.30</td>
</tr>
<tr>
<td>ORGAN</td>
<td>3.63</td>
<td>3.49</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>5.30</td>
<td>4.93</td>
</tr>
<tr>
<td>GCPN</td>
<td>7.98</td>
<td>7.85</td>
</tr>
</tbody>
</table>

logP: octanol-water partition coef., indicates solubility
QED: indicator of drug-likeness
Quantitative Results

Property targeting

- 7x higher success rate than JT-VAE, 10% less diversity

Table 2: Comparison of the effectiveness of property targeting task.

<table>
<thead>
<tr>
<th>Method</th>
<th>$-2.5 \leq \log P \leq -2$ Success</th>
<th>$5 \leq \log P \leq 5.5$ Success</th>
<th>$150 \leq MW \leq 200$ Success</th>
<th>$500 \leq MW \leq 550$ Success</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZINC</td>
<td>0.3%</td>
<td>1.3%</td>
<td>1.7%</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0.919</td>
<td>0.909</td>
<td>0.938</td>
<td>0.898</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>11.3%</td>
<td>7.6%</td>
<td>0.7%</td>
<td>16.0%</td>
</tr>
<tr>
<td></td>
<td>0.846</td>
<td>0.907</td>
<td>0.824</td>
<td>0.898</td>
</tr>
<tr>
<td>ORGAN</td>
<td>0</td>
<td>0.2%</td>
<td>15.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td></td>
<td>–</td>
<td>0.909</td>
<td>0.759</td>
<td>0.907</td>
</tr>
<tr>
<td>GCPN</td>
<td><strong>85.5%</strong></td>
<td><strong>54.7%</strong></td>
<td><strong>76.1%</strong></td>
<td><strong>74.1%</strong></td>
</tr>
<tr>
<td></td>
<td>0.392</td>
<td>0.855</td>
<td>0.921</td>
<td>0.920</td>
</tr>
</tbody>
</table>

**logP**: octanol-water partition coef., indicates **solubility**

**MW**: molecular weight an indicator of **drug-likeness**

**Diversity**: avg. pairwise Tanimoto distance between Morgan fingerprints of molecules
Constrained property optimization
- +180% higher scores than JT-VAE

Table 3: Comparison of the performance in the constrained optimization task.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>JT-VAE</th>
<th>GCPN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Improvement</td>
<td>Similarity</td>
</tr>
<tr>
<td>0.0</td>
<td>1.91 ± 2.04</td>
<td>0.28 ± 0.15</td>
</tr>
<tr>
<td>0.2</td>
<td>1.68 ± 1.85</td>
<td>0.33 ± 0.13</td>
</tr>
<tr>
<td>0.4</td>
<td>0.84 ± 1.45</td>
<td><strong>0.51 ± 0.10</strong></td>
</tr>
<tr>
<td>0.6</td>
<td>0.21 ± 0.71</td>
<td>0.69 ± 0.06</td>
</tr>
</tbody>
</table>
Qualitative Results

Visualization of GCPN graphs: Property optimization

(a) Penalized logP optimization

7.98
7.48
7.12
23.88*

(b) QED optimization

0.948
0.944
0.941
0.945
Qualitative Results

Visualization of GCPN graphs: constrained optimization

Starting structure

-8.32

-5.55

Finished structure

-0.71

-1.78

(c) Constrained optimization of penalized logP
Summary of Graph Generation

- Complex graphs can be successfully generated via **sequential generation**
- Each step a decision is made based on **hidden state**, which can be
  - **Explicit**: intermediate generated graphs, decode with GCN
  - **Implicit**: vector representation, decode with RNN
- Possible tasks:
  - **Imitating** a set of given graphs
  - **Optimizing** graphs towards given goals
References

PinSage:

Decagon:
- Website: [http://snap.stanford.edu/decagon/](http://snap.stanford.edu/decagon/)

GCPN:
- Code: [https://github.com/bowenliu16/rl_graph_generation](https://github.com/bowenliu16/rl_graph_generation)
What Next?

- **Project write-ups:**
  - Tue Dec 10 *(11:59PM)* Pacific Time
    - 1 team member uploads PDF to Gradescope
    - Don’t forget to tag your other team members!

- **Poster session:**
  - Thu Dec 12, 12:15 – 3:15 pm in Huang Foyer
    - All groups with at least one non-SCPD member must present
    - There should be 1 person at the poster at all times
    - Prepare a 2-minute elevator pitch of your poster
    - More instructions on Piazza

No late days!
What Next? Our Courses

- **CS246: Mining Massive Datasets (Winter 2020)**
  - Data Mining & Machine Learning for Big Data
    - (big==doesn’t fit in memory/single machine), SPARK

- **CS341: Project in Data Mining (Spring 2020)**
  - Groups do a research project on Big Data
  - We provide interesting data, projects and access to the Google Cloud infrastructure
  - Nice way to finish up CS224W project & publish it!
What Next?

- **Conferences / Journals:**
  - **KDD:** Conf. on Knowledge Discovery & Data Mining
  - **ICML:** Intl. Conf. on Machine Learning
  - **NeurIPS:** Neural Information Processing Systems
  - **ICLR:** Intl. Conf. on Learning Representations
  - **WWW:** ACM World Wide Web Conference
  - **WSDM:** ACM Web search and Data Mining
  - **ICWSM:** AAAI Int. Conf. on Web-blogs & Social Media
  - **Journal of Network Science**
  - **Journal of Complex Networks**
What Next? Other Courses

- Other relevant courses:
  - CS229: Machine Learning
  - CS230: Deep Learning
  - MSE231: Computational Social Science
  - MSE334: The Structure of Social Data
  - CS276: Information Retrieval and Web Search
  - CS245: Database System Principles
  - CS347: Transaction Processing & Databases
Thank you Michele and TAs!!

Teaching Assistants

Christina Yuan
Head TA

Lingzi (Liz) Guo

Benjamin (Ben) Hannel

Kuangcong (Cecilia) Liu

Vasco Portilheiro

Andrew Wang

Alexis Goh Weiying

Zhitao (Rex) Ying

Thank You
In Closing...

- You Have Done a Lot!!!
- And (hopefully) learned a lot!!!
  - Answered questions and proved many interesting results
  - Implemented a number of methods
  - And are doing excellently on the class project!

Thank You for the Hard Work!!!