Graph Neural Networks

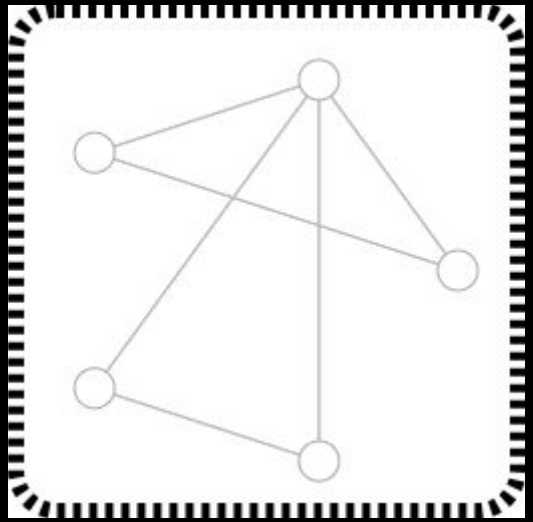
& Hybrid Approaches for OrbitAI

David Orjuela

February 20th, 2025

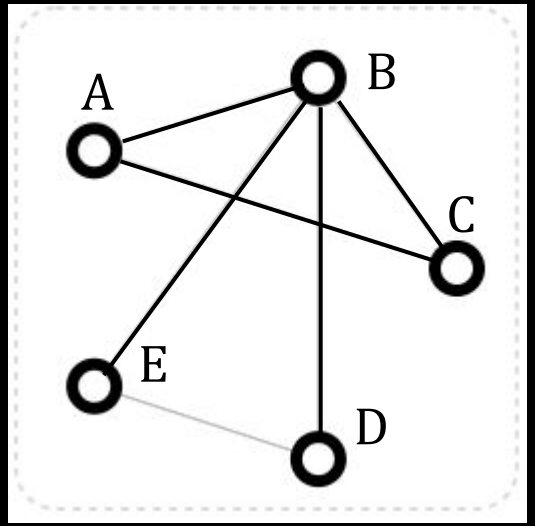
Introduction

- What are graphs?



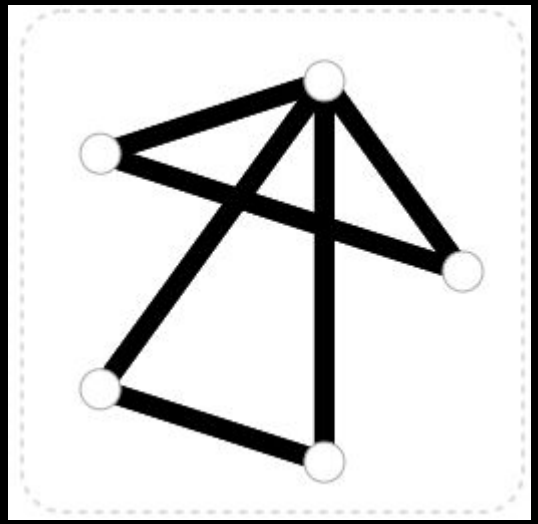
Introduction

- What are graphs?
- Label/Identity
- Number of Neighbors



Introduction

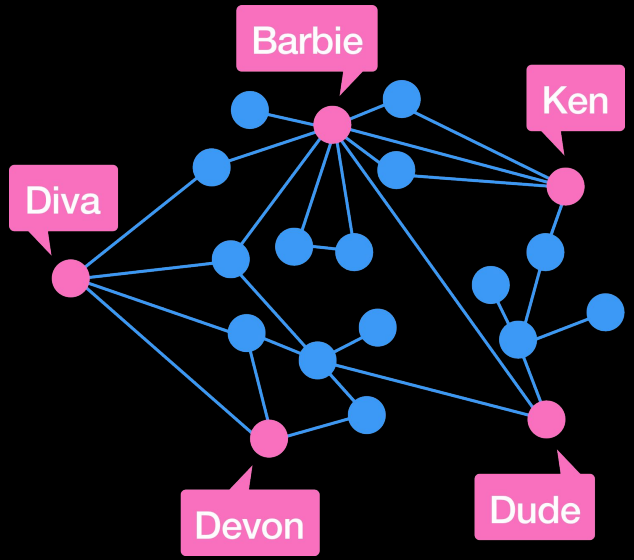
- What are graphs?
- Undirected
- Directed



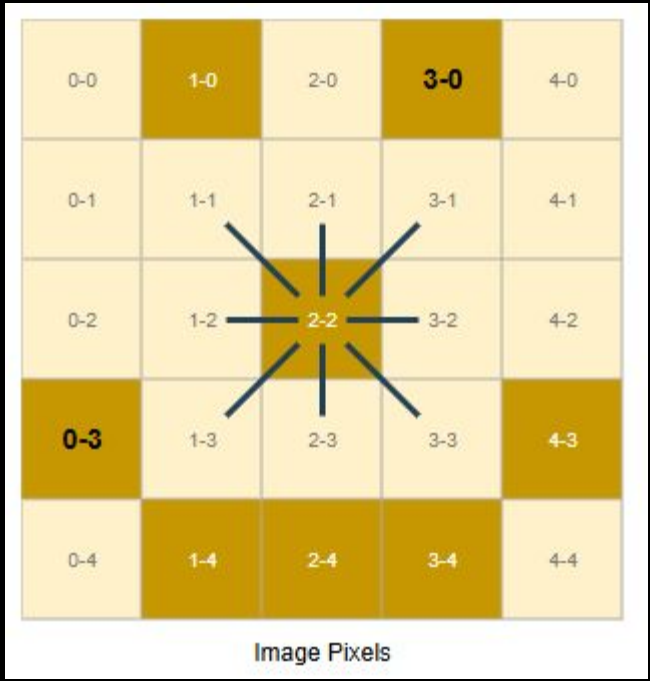
Common Graphs

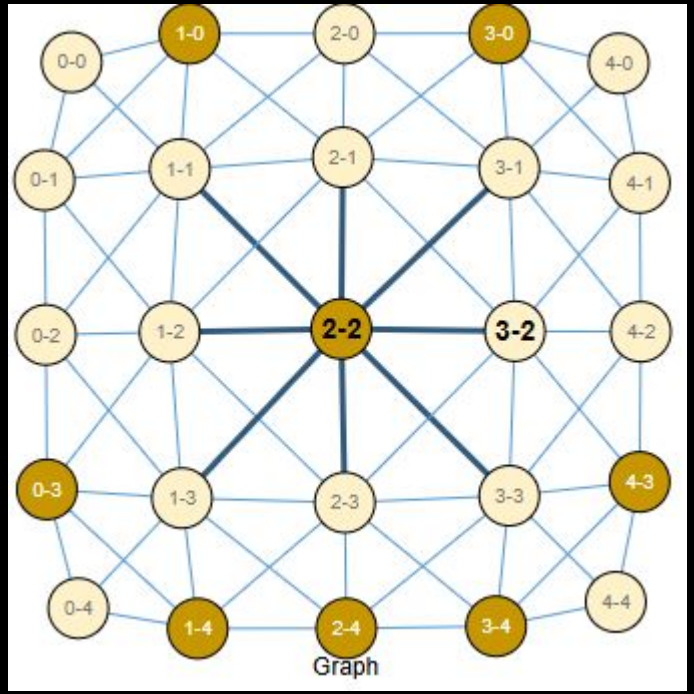
Intuitively:

- Social Networks



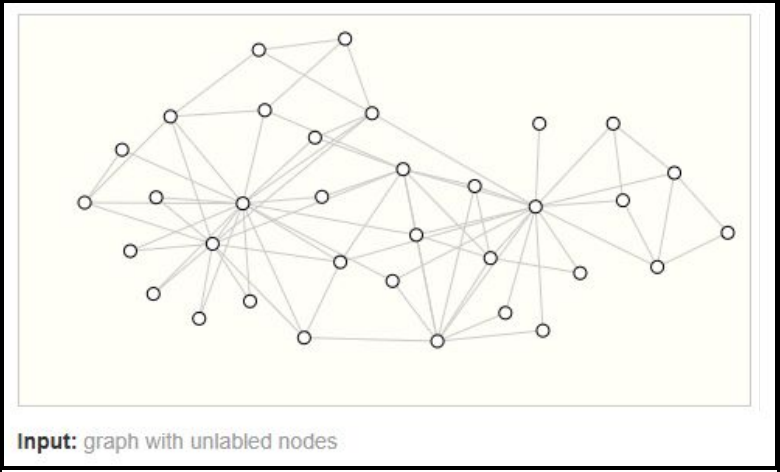
Common Graphs



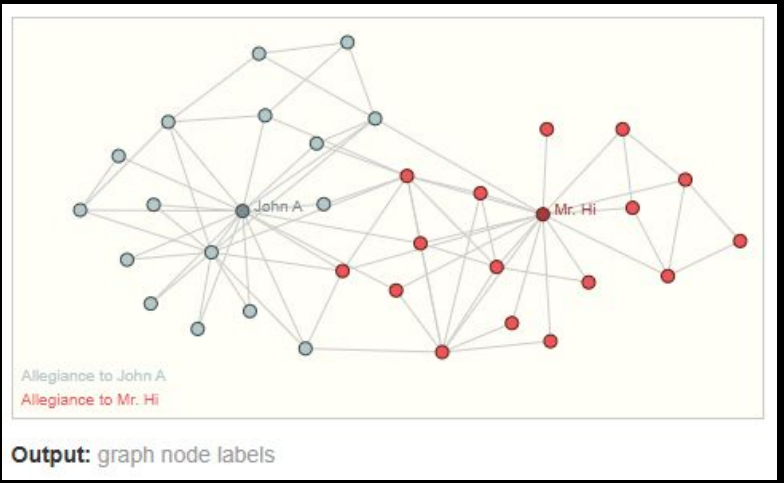


Tasks to Perform

- Three general types of prediction tasks:
  - Graph-Level
  - Node-Level
  - Edge-Level



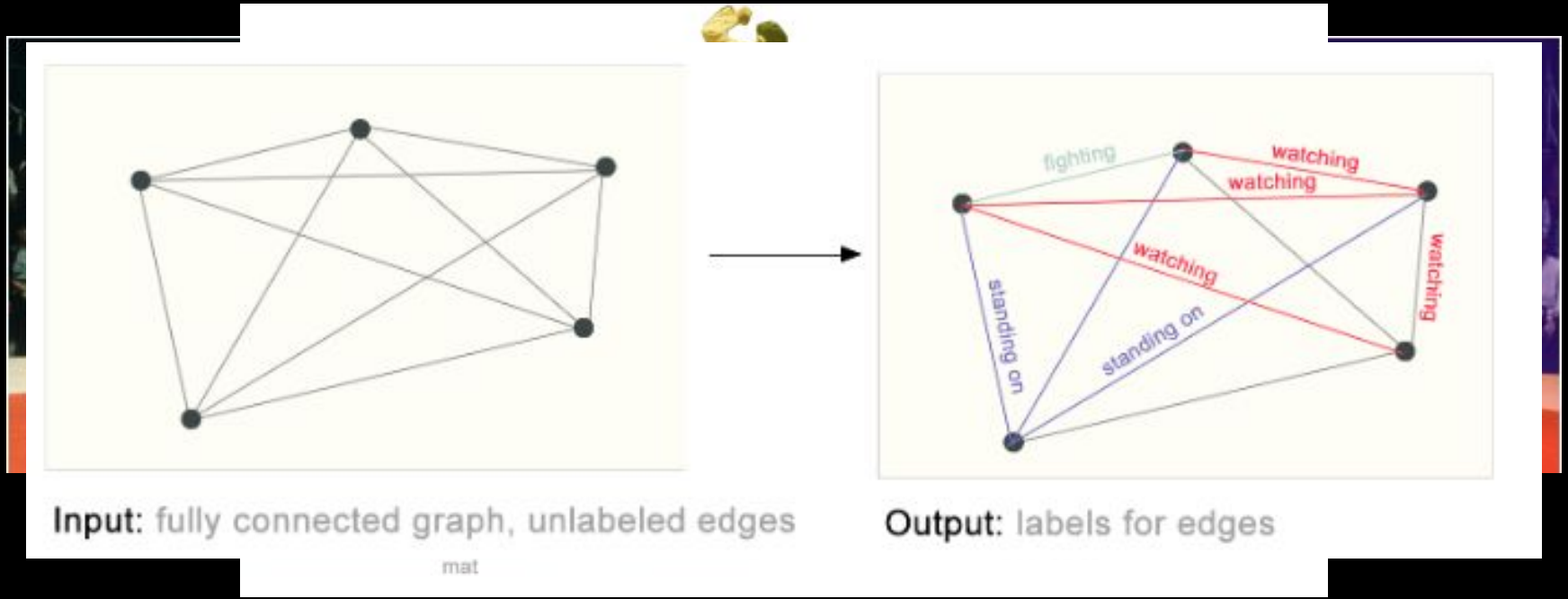
Input: graph with unlabled nodes



Output: graph node labels

Tasks to Perform

Edge-Level



Why Consider GNNs

- What if we could combine both RNNs and GNNs into our project?
- Our space environment is inherently a graph
  - Nodes: Satellites, debris, space stations.
  - Edges: Potential collision paths, proximity-based relationships, orbital interactions.
- Limitations of RNN-only approaches:
  - RNNs process temporal sequences well but don’t model relationships between multiple objects effectively.
  - Predicting an individual satellite’s movement isn’t enough—we need to understand interactions.

Building the Graph (Using NORAD's TLE Data)

Steps to construct the graph:

1. Nodes: Each **tracked** satellite or debris object is a node.
2. Edges formed based on:
    - Thresholds on distance (e.g., within 5 km)
    - Historical close approaches
    - Similar orbits (objects may have future collision risk)
3. Edge Features: Include velocity, past collision risk, and trajectory patterns.

Models space traffic and potential collision zones.

Proposal – Hybrid RNN + GNN Model

What if we combine both RNNs and GNNs?

- RNNs excel at sequential (time) predictions (e.g., individual satellite trajectories over time)
- GNNs excel at relational modeling (e.g., how satellites and debris interact).

- Combining the two would allow us to predict future movements and then evaluate collision risk.

Hybrid Approach 1 – Node State Updates

- Each node (satellite or debris) contains an RNN to process its individual trajectory.
- The RNN predicts the next state of the object based on:
  - Position (x, y, z)
  - Velocity (Vx, Vy, Vz)
  - Acceleration
  - Fuel & maneuver capabilities
- GNN takes these RNN-updated states and propagates risk information across the graph.

Hybrid Approach 2 – Edge Features

- Instead of RNNs predicting only node states, they can also predict edge weights dynamically.
- Edges store historical and predicted proximity trends:
  - Objects frequently moving close to each other get stronger edges.
  - An RNN predicts future proximity values over time.
- The GNN aggregates edge-level information to refine global collision risk assessment.

Comparison – Node Updates vs. Edge Features

Feature	RNN as Node Updates	RNN as Edge Features
Tracks individual orbits	✅Yes	❌ No
Predicts inter-object dynamics	✅ Yes (indirectly)	✅ Yes (directly)
Best for tracking individual satellites	✅ Yes	❌ No
Best for understanding cluster movements	❌ No	✅ Yes
Handles dynamic graphs (objects enter/exit space)	✅ Yes	✅ Yes
Implementation Complexity	Medium	High
Scalability	Moderate	Lower (depends on # of edges)

RNN Only vs GNN-RNN Hybrid

Metric	RNN-Only	GNN-RNN Hybrid
Trajectory Prediction Accuracy	✅ High	✅ High
Collision Risk Prediction	❌ Limited to pairwise comparisons	✅ Models system-wide risks
Computational Cost	✅ Lower	❌ Higher
Scalability for Large Satellite Networks	❌ Limited	✅ Better for constellations
Interpretability of Space Traffic Behavior	❌ Individual-based	✅ Cluster-based

How to Modify Our RNN for This Approach

- Current RNN Architecture:
  - LSTMs/GRUs predict individual satellite positions.
  - Dense layers classify collision risk.
- Modifications for Hybrid Approach:
  - Each node runs an RNN for trajectory prediction.
  - A GNN processes interactions based on RNN-updated node states.
  - Edge weights evolve over time using a separate RNN.
- Outcome: A dynamic graph that updates in real-time based on predicted satellite movements.

Final Decision – Which Architecture is Better?

- Option 1: RNN for Node State Updates + GNN (Easier to implement, better individual tracking)
- Option 2: RNN for Edge Features + GNN (More complex, better at tracking large-scale interactions)
- Maybe combination of both?