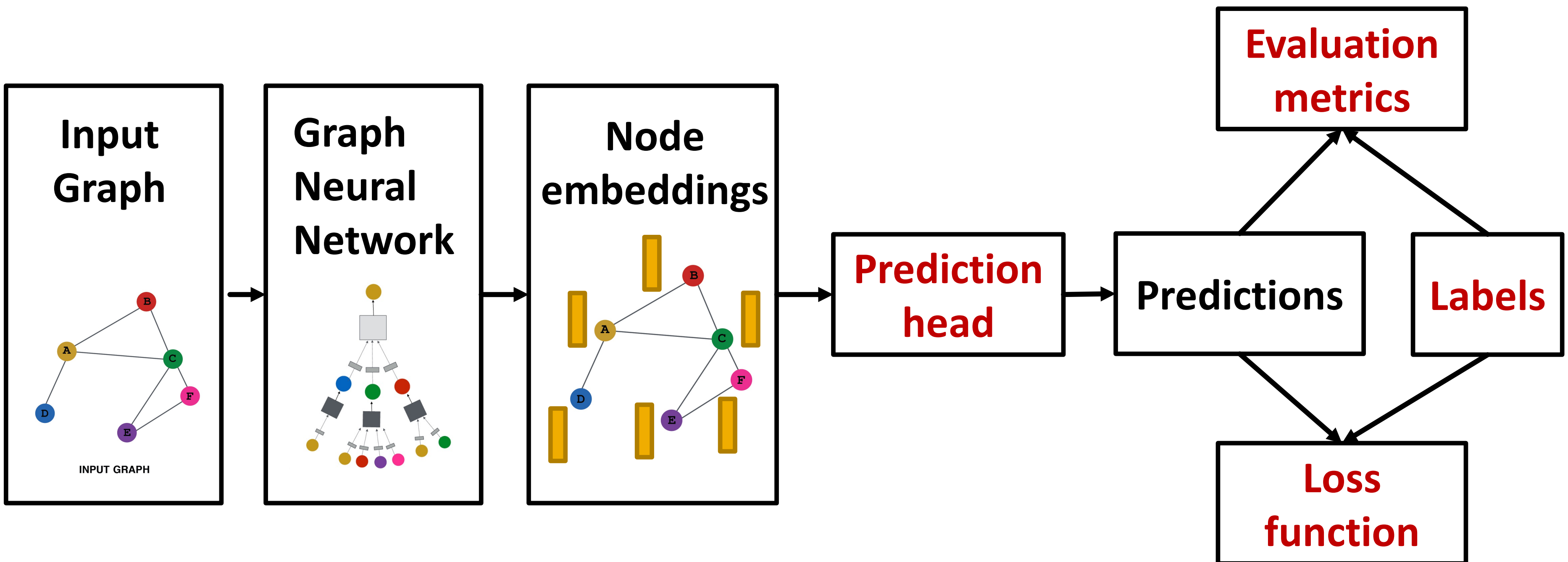
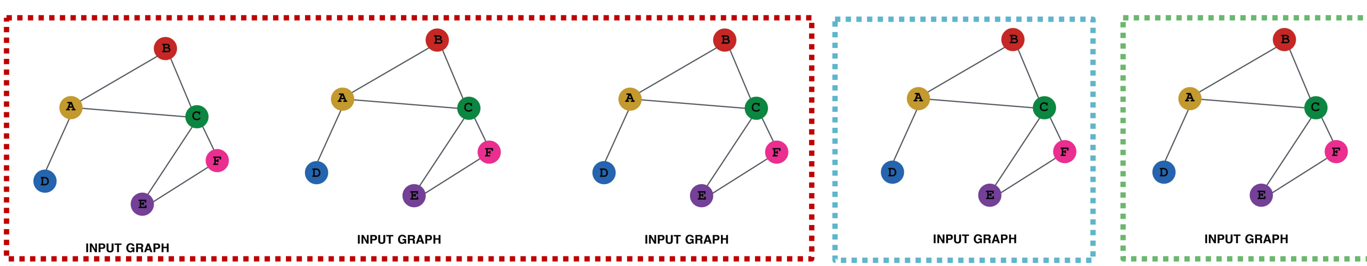
Setting-up GNN Prediction Tasks

GNN Training Pipeline (5)



(5) How do we split our dataset into train / validation / test set?



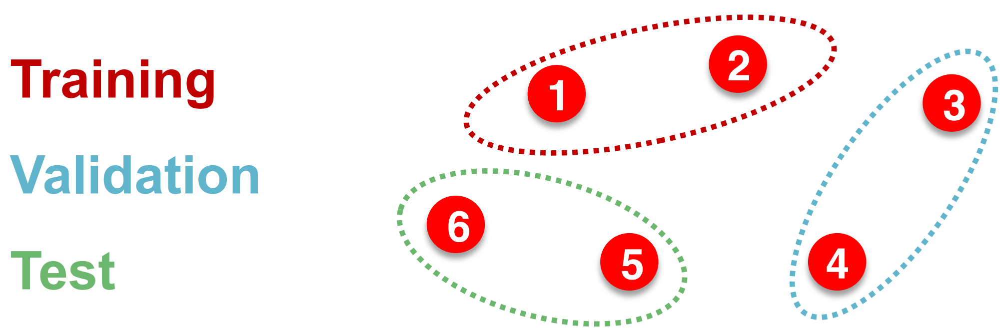
Dataset split

Dataset Split: Fixed / Random Split

- **Fixed split:** We will split our dataset **once**
  - **Training set**: used for optimizing GNN parameters
  - **Validation set**: develop model/hyperparameters
  - **Test set**: held out until we report final performance
- **A concern:** sometimes we cannot guarantee that the test set will really be held out
- **Random split:** we will **randomly split** our dataset into training / validation / test
  - We report **average performance over different random seeds**

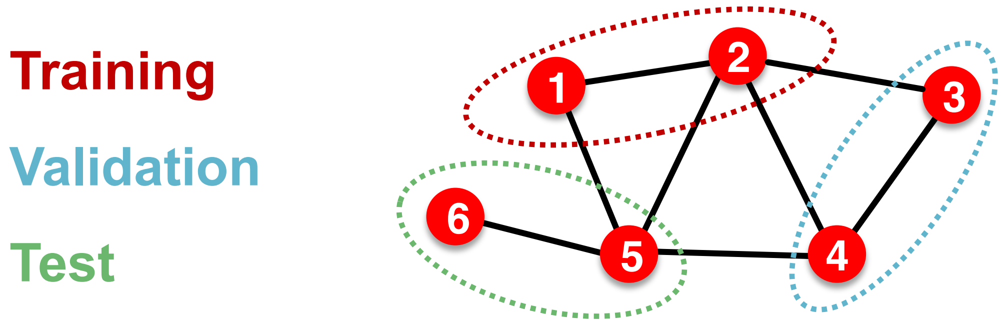
Why Splitting Graphs is Special

- Suppose we want to split an image dataset
  - **Image classification:** **Each data point is an image**
  - Here **data points are independent**
    - **Image 5 will not affect our prediction on image 1**



Why Splitting Graphs is Special

- **Splitting a graph dataset is different!**
  - **Node classification: Each data point is a node**
  - Here **data points are NOT independent**
    - **Node 5 will affect our prediction on node 1,** because it will participate in message passing → affect node 1's embedding

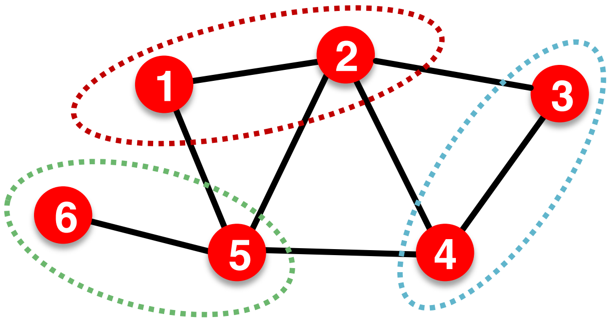


- **What are our options?**

Why Splitting Graphs is Special

- **Solution 1 (Transductive setting): The input graph can be observed in all the dataset splits (training, validation and test set).**
- **We will only split the (node) labels**
  - **At training time**, we compute embeddings **using the entire graph**, and train **using node 1&2's labels**
  - **At validation time**, we compute embeddings **using the entire graph**, and **evaluate on node 3&4's labels**

Training
Validation
Test



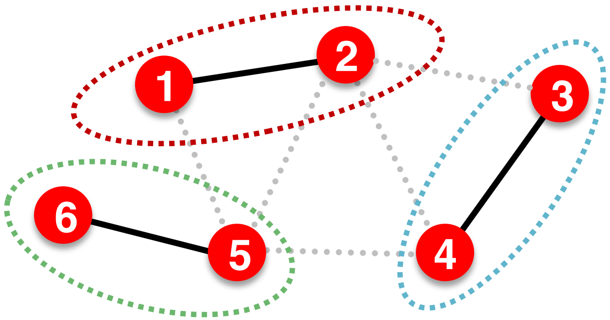
Why Splitting Graphs is Special

- **Solution 2 (Inductive setting): We break the edges between splits to get multiple graphs**
  - **Now we have 3 graphs that are independent. Node 5 will not affect our prediction on node 1 any more**
  - **At training time**, we compute embeddings **using the graph over node 1&2**, and train **using node 1&2's labels**
  - **At validation time**, we compute embeddings **using the graph over node 3&4**, and **evaluate on node 3&4's labels**

Training

Validation

Test

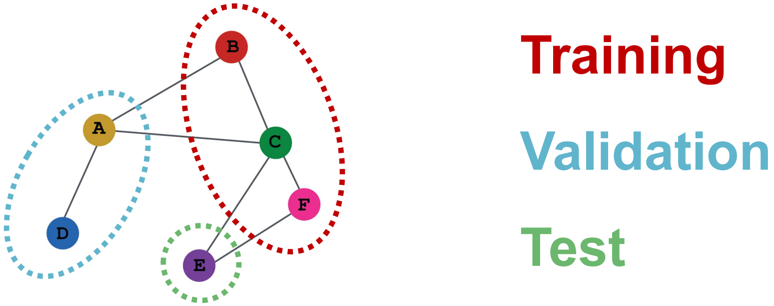


Transductive / Inductive Settings

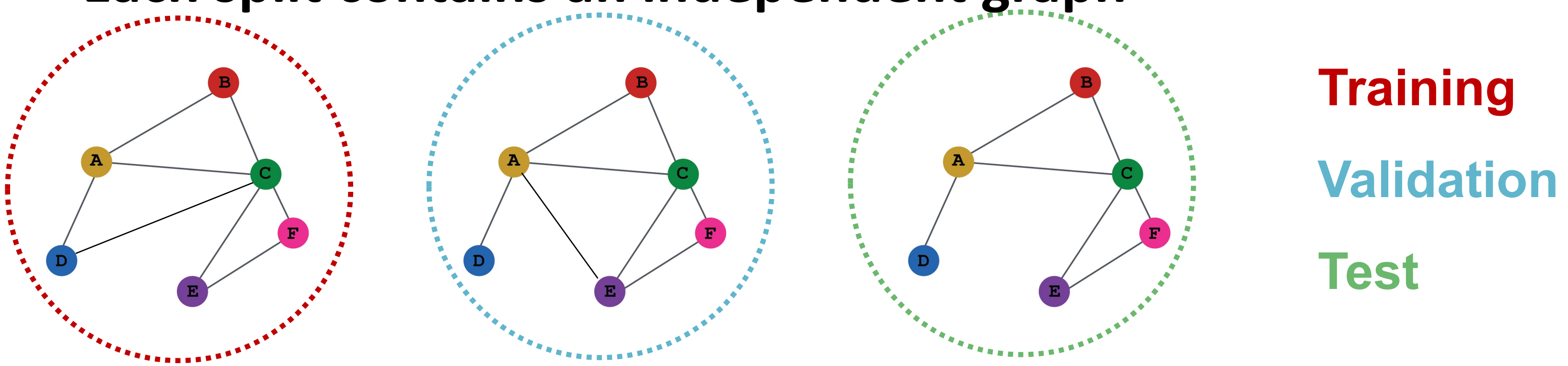
- **Transductive setting:** training / validation / test sets are **on the same graph**
  - The **dataset consists of one graph**
  - **The entire graph can be observed in all dataset splits, we only split the labels**
  - Only applicable to **node / edge** prediction tasks
- **Inductive setting:** training / validation / test sets are **on different graphs**
  - The **dataset consists of multiple graphs**
  - Each split can **only observe the graph(s) within the split**. A successful model should **generalize to unseen graphs**
  - Applicable to **node / edge / graph** tasks

Example: Node Classification

- **Transductive** node classification
  - **All the splits can observe the entire graph structure**, but can only observe the labels of their respective nodes

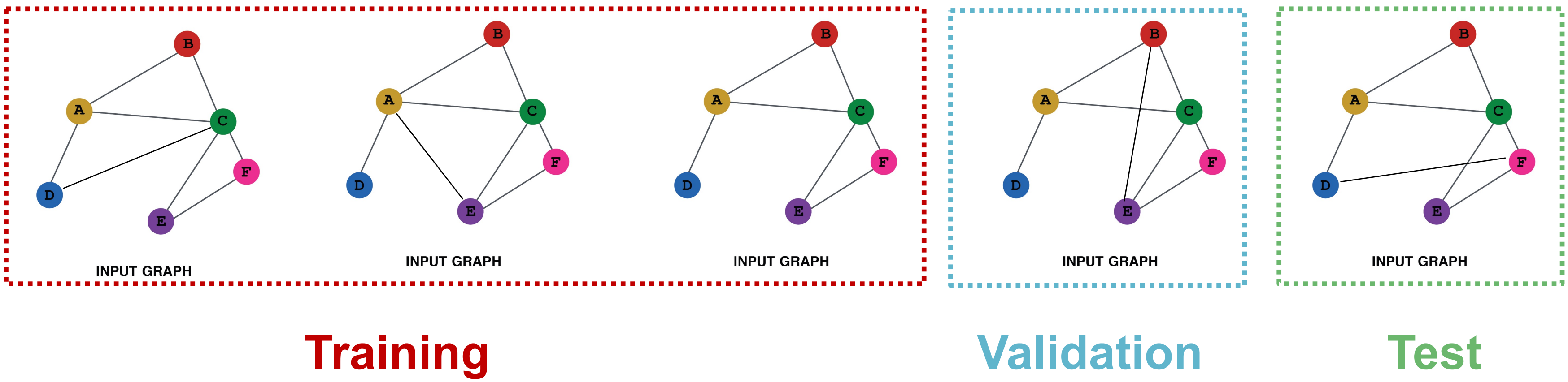


- **Inductive** node classification
  - Suppose we have a dataset of 3 graphs
  - **Each split contains an independent graph**



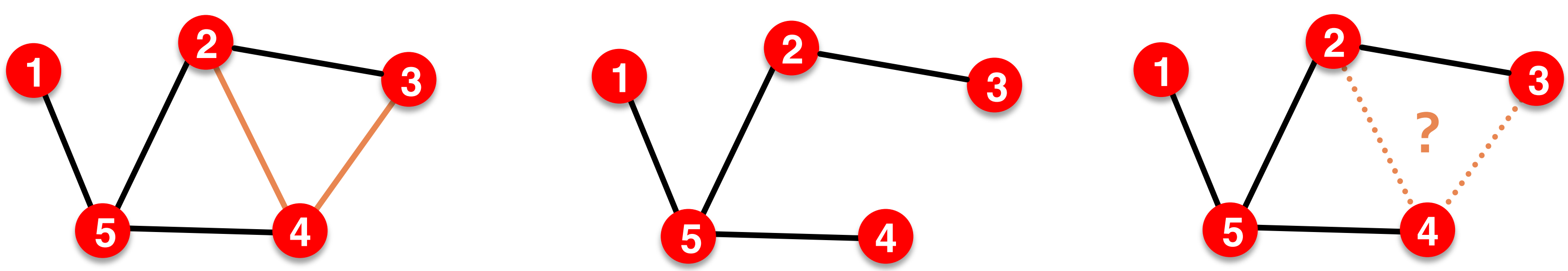
Example: Graph Classification

- Only the **inductive setting** is well defined for **graph classification**
  - Because **we have to test on unseen graphs**
  - Suppose we have a dataset of 5 graphs. Each split will contain independent graph(s).



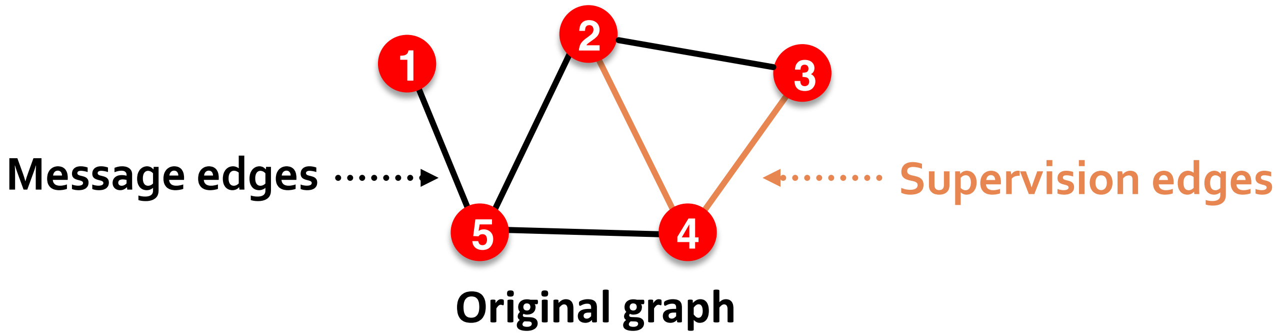
Example: Link Prediction

- **Goal of link prediction: predict missing edges**
- **Setting up link prediction is tricky:**
  - Link prediction is an unsupervised / self-supervised task. We need to **create the labels** and **dataset splits** on our own
  - Concretely, we need to **hide some edges from the GNN** and the **let the GNN predict if the edges exist**



Original graph **Input graph to GNN** **Predictions made by GNN**

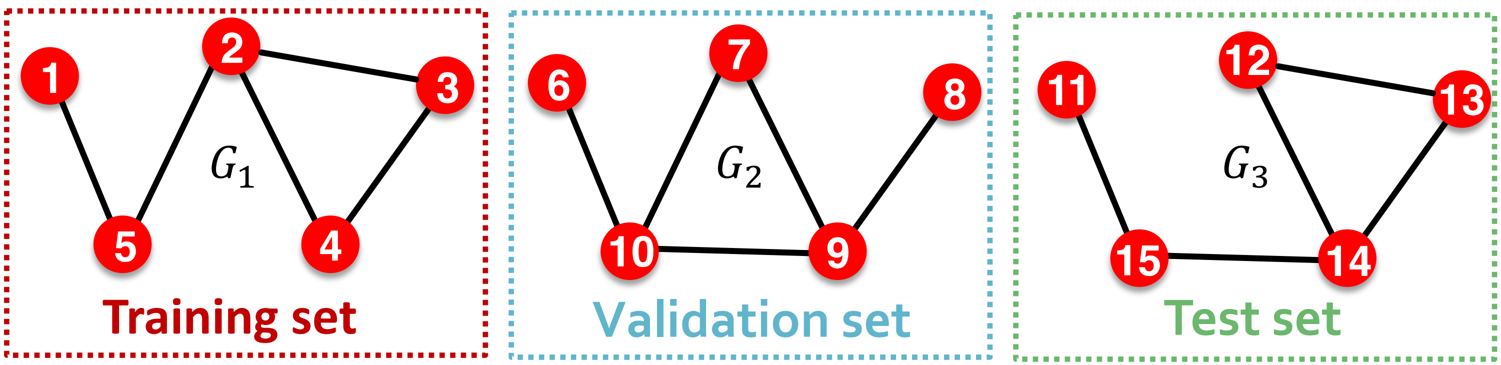
Setting up Link Prediction



- For link prediction, we will split edges twice
- Step 1: Assign 2 types of edges in the original graph
  - Message edges: Used for GNN message passing
  - Supervision edges: Use for computing objectives
  - After step 1:
    - Only message edges will remain in the graph
    - Supervision edges are used as supervision for edge predictions made by the model, will not be fed into GNN!

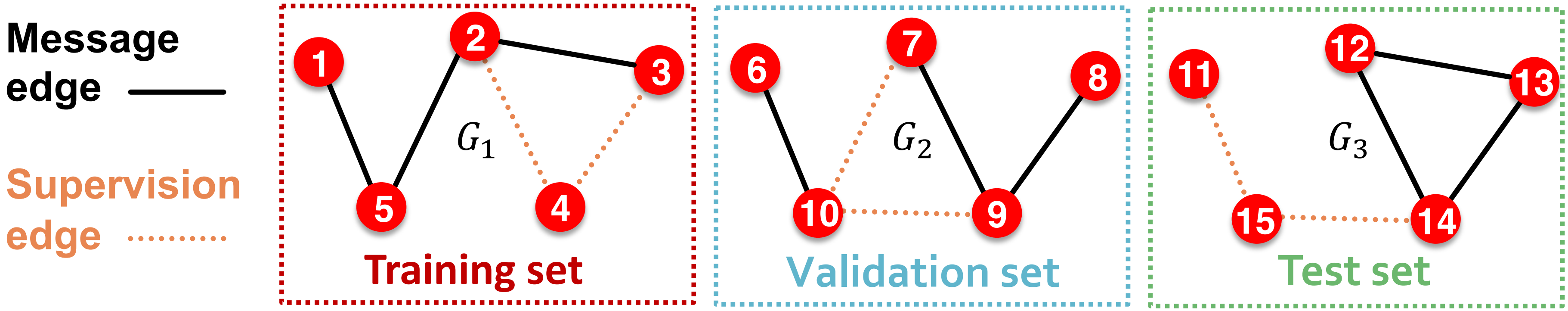
Setting up Link Prediction

- **Step 2: Split edges into train / validation / test**
- **Option 1: Inductive link prediction split**
  - **Suppose we have a dataset of 3 graphs. Each inductive split will contain an independent graph**



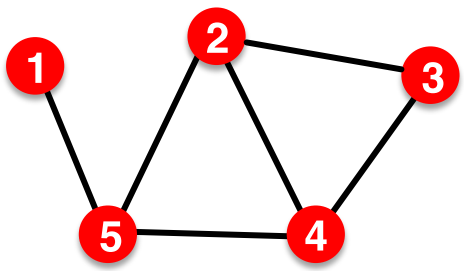
Setting up Link Prediction

- **Step 2: Split edges into train / validation / test**
- **Option 1: Inductive link prediction split**
  - **Suppose we have a dataset of 3 graphs. Each inductive split will contain an independent graph**
  - **In train** or **val** or **test** set, **each graph will have 2 types of edges: message edges** + **supervision edges**
    - Supervision edges are not the input to GNN



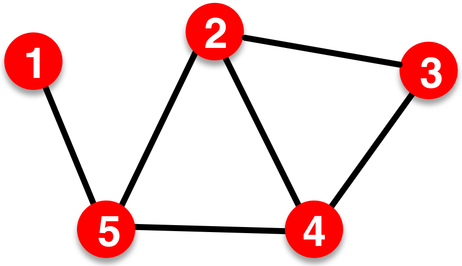
Setting up Link Prediction

- Option 2: Transductive link prediction split:
  - This is the default setting when people talk about link prediction
  - Suppose we have a dataset of 1 graph



Setting up Link Prediction

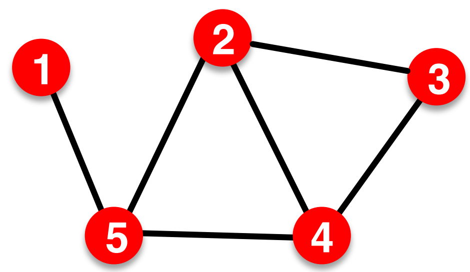
- **Option 2: Transductive link prediction split:**
  - **By definition of "transductive", the entire graph can be observed in all dataset splits**
    - **But since edges are both part of graph structure and the supervision, we need to hold out validation / test edges**
    - **To train the training set, we further need to hold out supervision edges for the training set**



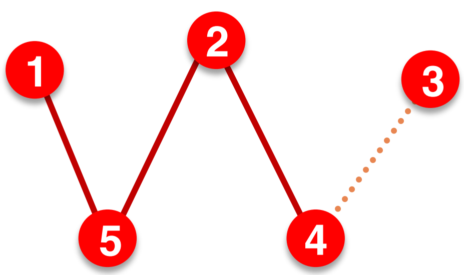
- **Next:** we will show the exact settings

Setting up Link Prediction

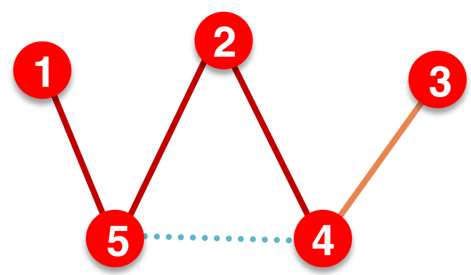
- **Option 2: Transductive link prediction split:**



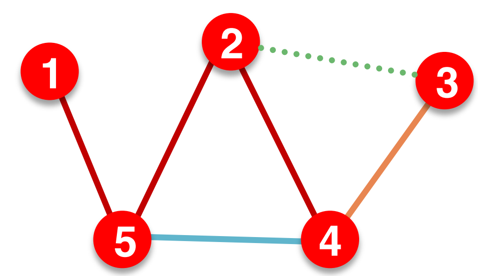
The original graph



(1) At training time:
Use **training message edges** to predict **training supervision edges**



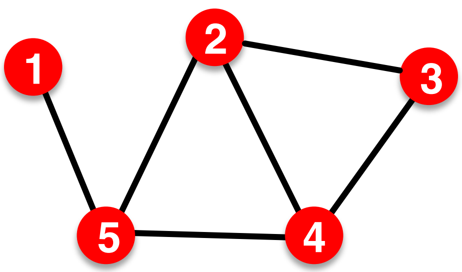
(2) At validation time:
Use **training message edges & training supervision edges** to predict **validation edges**



(3) At test time:
Use **training message edges & training supervision edges & validation edges** to predict **test edges**

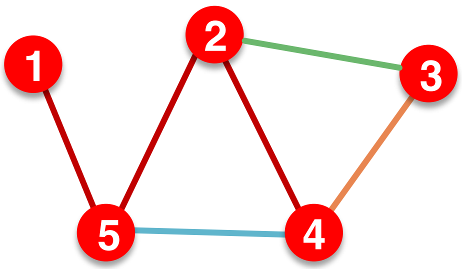
Setting up Link Prediction

- **Summary: Transductive link prediction split:**



The original graph

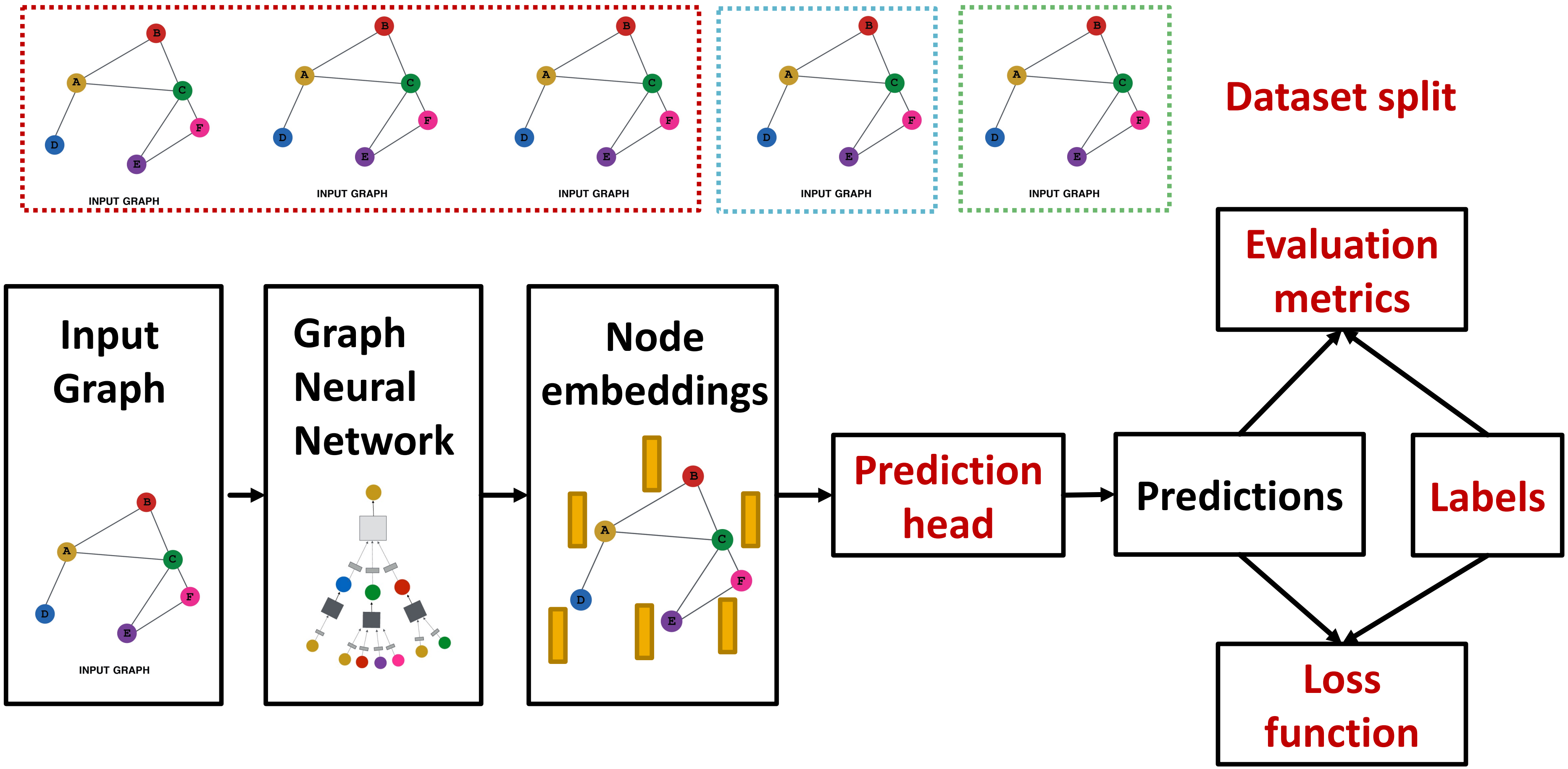
Split



Split Graph with
4 types of edges

- **Note:** Link prediction settings are tricky and complex. You may find papers do link prediction differently.
- Luckily, we have full support in **PyG and** [GraphGym]

GNN Training Pipeline



Implementation resources:

DeepSNAP provides core modules for this pipeline

GraphGym further implements the full pipeline to facilitate GNN design

Summary of the Lecture

- **We introduce a general GNN framework:**
  - **GNN Layer**:
    - Transformation + Aggregation
    - Classic GNN layers: GCN, GraphSAGE, GAT
  - **Layer connectivity**:
    - The over-smoothing problem
    - Solution: skip connections
  - **Graph Augmentation:**
    - Feature augmentation
    - Structure augmentation
  - **Learning Objectives**
    - The full training pipeline of a GNN