**Graph Neural Networks for Tabular Data Prediction**

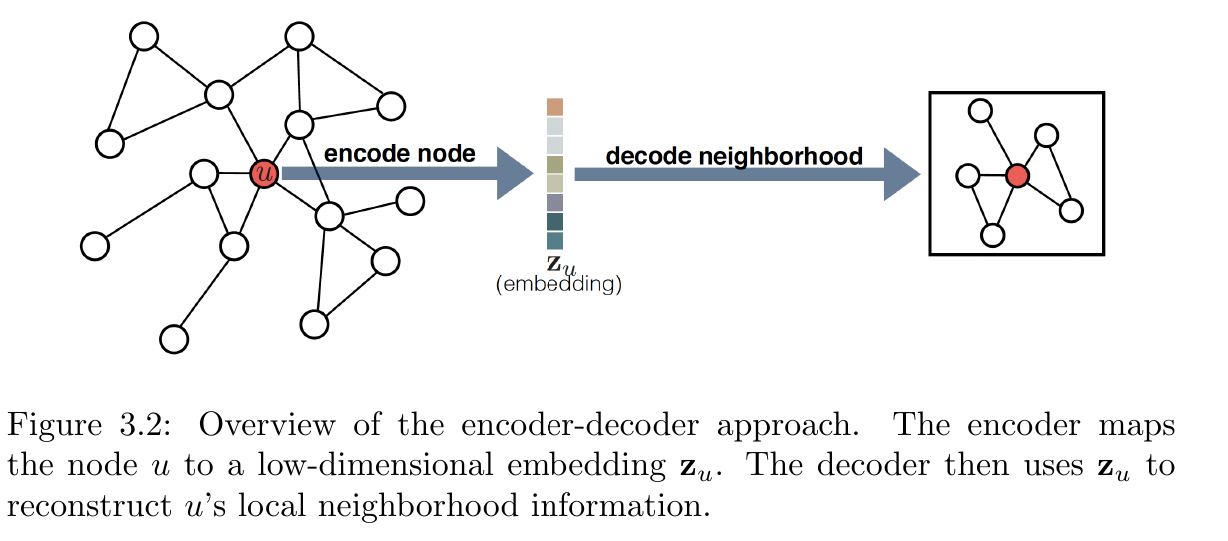
*Granick, Joseph; Gupta, Sarthak*

## Graph Structured Data: Problem/Task

A wide variety of problems are naturally structured as graphs or networks. Additionally many domains traditionally dominated by tabular data can leverage graph representation learning methods as tabular data can be easily structured in a graph. For many domains this representation will be more expressive, capturing latent structural information that can be flattened in a tabular representation, or difficult to represent due to complexity of relationships conforming to the limits imposed by the tabular schema. (Cheng et. al 2024)

## Limitations of Traditional Graph Learning Methods

### Node Embedding Methods



Graph representation learning can be viewed through the paradigm of **encoding-decoding graphs**. In this framework the problem is framed and decomposed into two primary **encoding** and **decoding** operations.

These consist of an *encoder model* that maps nodes to a lower dimensional vector embedding that is passed to a *decoder model that reconstructs* graph statistics captured in the node embedding. (Hamilton 2020, pg 30)

**Encoder**

**Decoder**

**Factorization Based Embeddings** The encoder-decoder paradigm can be thought of in the context of *matrix factorization* for a low-dimensional approximation of a generalized adjacency matrix that instead of reconstructing entries from the graph,represents a user defined measure of node-node similarity representing information from local node neighborhood structure. Most modern methods use an *inner-product based decoder*, based on the assumption that the similarity or neighborhood overlap between nodes should be proportional to the dot product of the node embeddings from the encoder stage. (Hamilton 2020, pg 32)

This decoder definition is then combined with a measure of mean-squared error where S[u,v] is the user defined similarity measure that differentiates different implementations of inner-product methods factorization based embeddings.

||

The loss function can then be minimized with a matrix-factorization algorithm. These methods aim to learn imbeddings for each node so that the inner product of the encoded vectors corresponds to a measure of node-similarity (Hamilton 2020, pg 34).

**Random Walk Embeddings** These methods like *Deep Walk* and *node2vec* use random walks to encode graph embeddings. (Hamilton, 2020 pg 34,37)

These type encoders use a *shallow embedding approach,* a simple embedding lookup from the node ID. Research from this approach has yielded many successful uses of these methods, but they have significant limitations. The direct encoding of a **unique embedding** for each node with no capacity for **parameter sharing** makes these methods inefficient and **computationally intractable** for large graphs. Shallow embeddings leave predictive power on the table as they do not incorporate **node features** which often encode variable information for observations in many graph datasets that can often have significant predictive power if represented in a model. The most important limitation with respect to **inductive** applications generalizing to new examples for predictive tasks is the **transductive** nature of shallow embeddings. These methods can only learn embeddings for nodes used in training dataset (Hamilton 2020, pg 37)

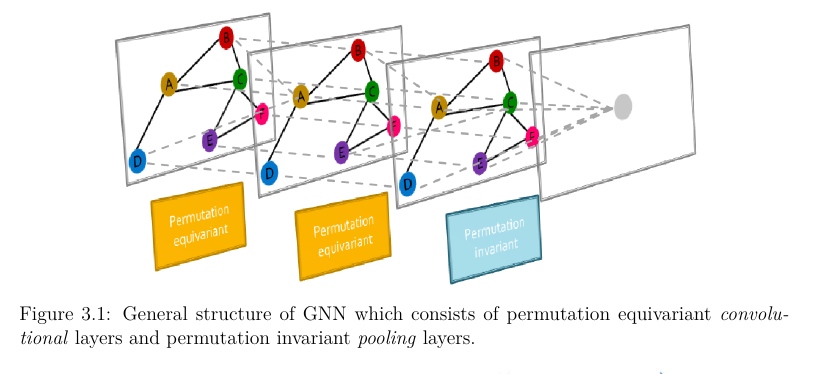
## Solution: Generalize beyond shallow embeddings with Graph Neural Networks

The *encoder-decoder* paradigm can be generalized beyond shallow embeddings with a more sophisticated encoder architecture called **graph neural networks,** which learn a representation more generally from the structure and attributes of the graph (Hamilton 202, pg. 31, 37).Key Properties for GNNs

A key design requirement when performing machine learning over graph structured data is that the model learned should not be dependent of th arbitrary node ordering in the adjacency matrix representation. They should be either **Permutation Invariant** meaning the function doesn’t depend on arbitrary ordering of rows and column in an adjacency matrix or **Permutation Equivariant** meaning the output of the function is permuted in a way consistent with the adjacency matrix(Hamilton 2020, pg 47)

**Permutation Invariance:**

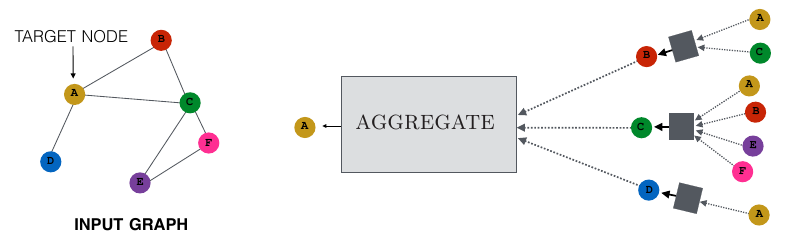
**Permutation Equivariance:**



In practice to preserve these properties the general GNN structure is composed from permutation invariant convolution layers and permutation invariant pooling layers.

### Message Passing

*Neural message passing* is the defining feature of the GNN paradigm, where nodes exchange information through vector messages and are updated with neural networks (Hamilton 2020, pg. 48).



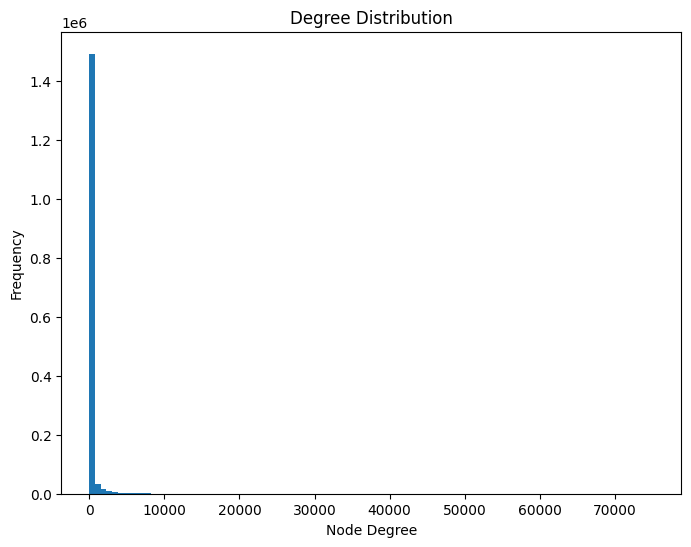
For each node a *hidden embedding* is **updated** at each iteration of message- passing based on information **aggregated** from each node’s neighborhood

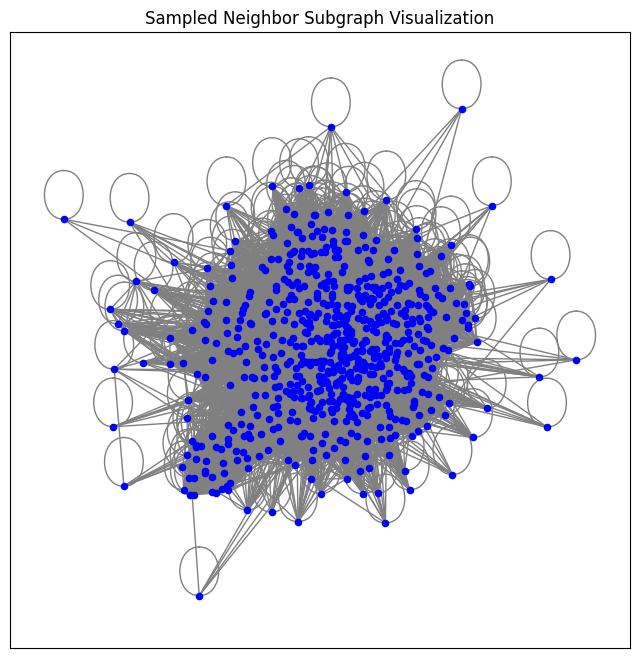
# Implementation

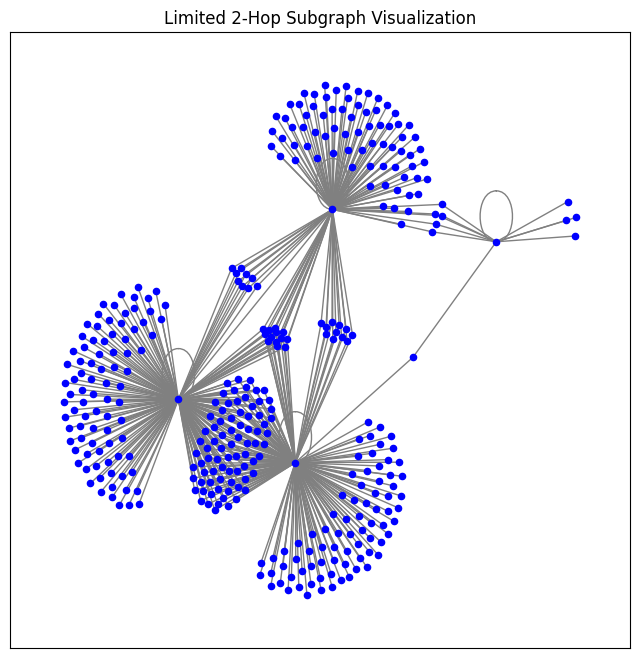
## Graph Structure of Dataset

The dataset used is part of the Amazon Product dataset, and the tasks traditionally performed with methods over tabular data is to learn a model for recommender system.

The dataset is composed of users and items. This user-item relationship is naturally represented as a bipartite graph. Using a GNN for a recommender system is most naturally cast as the edge-level task of link prediction, with the goal fo correctly predicting an edge between a user and an item, representing a correct recommendation.

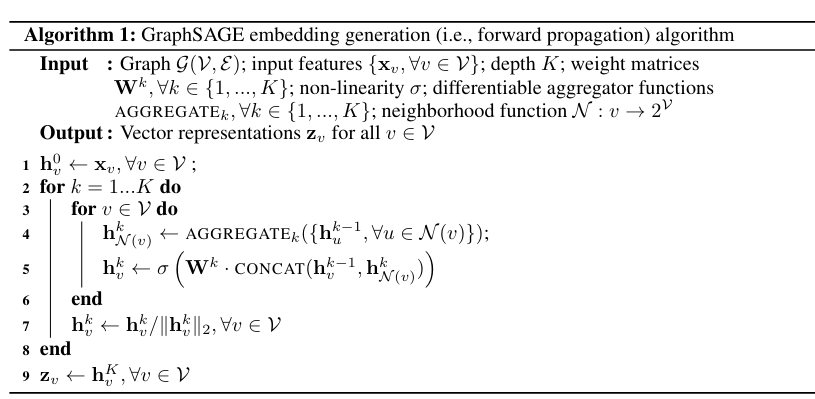




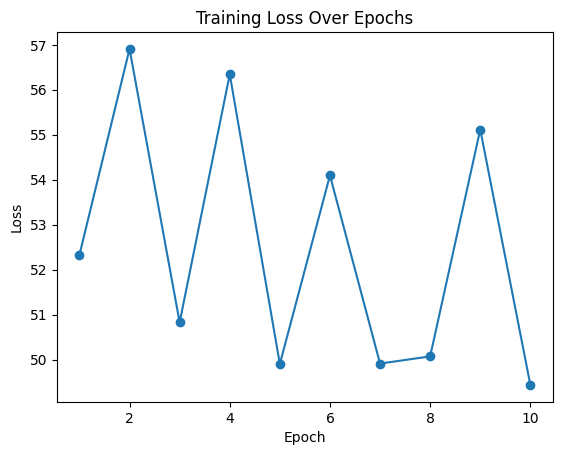


## Model Architecture

The dataset was used to train a model following the **GraphSAGE** GNN architecture. This architecture but uses sampling to work in an *inductive learning frame work*, and scale to larger graphs (Hamilton, YIng, Leskovec 2018)



This model is composed of two convolutional layers and a ReLU activations function. The model was trained over 10 Epochs, with a batch size of 1024



## Results

Evaluating the results we used variations of Hits@K, which measured the percentage of True Positive based on purchase of at least one item in the k-item basket of recommendations. In this test we selected a K of 10 items at a time.

**Validation Hits@10:** 0.8768, **NDCG@10:** 0.6970

**Test Hits@10:** 0.8832, **NDCG@10:** 0.7138

# References

Li, Cheng-Te; Tsai, Yu-Che; Chen, Chih-Yao; Liao, Jay Chiehen 2024. *Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy & Directions*

Hamilton, W.L. 2020. *Graph Representation Learning*. Morgan/Claypool

Bishop, C.M.; Bishop H, 2024. *Deep Learning: Foundations and Concepts*. Springer

Hamilton, William L; Ying, Rex; Leskovec, Jure 2018. *Inductive Representation Learning on Large Graphs*