# Visualizing Graph Neural Networks with CorGIE: Corresponding a Graph to Its Embedding

Zipeng Liu, Yang Wang, Jürgen Bernard, Tamara Munzner

**Abstract**—Graph neural networks (GNNs) are a class of powerful machine learning tools that model node relations for making predictions of nodes or links. GNN developers rely on quantitative metrics of the predictions to evaluate a GNN, but similar to many other neural networks, it is difficult for them to understand if the GNN truly learns characteristics of a graph as expected. We propose an approach to corresponding an input graph to its node embedding (aka latent space), a common component of GNNs that is later used for prediction. We abstract the data and tasks, and develop an interactive multi-view interface called CorGIE to instantiate the abstraction. As the key function in CorGIE, we propose the K-hop graph layout to show topological neighbors in hops and their clustering structure. To evaluate the functionality and usability of CorGIE, we present how to use CorGIE in two usage scenarios, and conduct a case study with two GNN experts.
**Availability:** Open-source code, supplemental materials & video at https://osf.io/tr3sb/.

**Index Terms**—Visualization for machine learning, graph neural network, graph layout.

## 1 Introduction

Graph neural networks (GNNs) are machine learning (ML) models that have received substantial recent attention due to their ability to deal with abstract concepts like relationships and interactions. GNN models are widely used in downstream ML applications such as fraud detection, traffic modeling, and product recommendation, in addition to the classic ML application domains of natural language processing and computer vision.

During training, a GNN model takes in a node-link graph and as output generates a **node embedding**; that is, a representation of all discrete nodes in the graph as fixed-dimensional vectors in a continuous **latent space**. Proximities between embedded nodes in the latent space represent meaningful similarities between nodes in the input graph learned during the training. The node embedding leverages both the topology – connectivity between neighboring nodes – and node features – information associated with each node – of the input graph. (Information associated with nodes, typically called node features in the ML literature, is sometimes called node attributes in the visualization literature.) The node embedding is then available for feeding into downstream ML applications, for example to make predictions about nodes and links. Fig. 1a summarizes the standard GNN pipeline.

To achieve optimal performance, model developers must evaluate whether the training has succeeded in producing a GNN model that has in fact learned the important characteristics from the input graph. Model developers often need to determine the stopping criteria for training a GNN model; in other words, to answer the question "are we there yet?" when training models or tuning hyperparameters. In addition to global questions such as when to stop training and tuning, model developers need to debug and trace errors and sub-optimal states in the trained model. For example, they may want to identify and understand the groups of nodes that suffer from mis-classification the most. In some cases, they also need to compare results from different model architectures or hyperparameters to choose the best ones for production use.

The most prevalent current approach for model evaluation is to compute quantitative metrics based on the downstream ML applications [1], such as precision and F1 for node classification and Hit@k for link prediction, which are used for cross-validation during training. Some researchers have proposed algorithms to compute explanations for prediction results in terms of influential nodes, or what features are important in the determination of a specific node [2]. Another common approach is to conduct qualitative sanity checks, such as manual inspection of node or link instances, for example by visualizing with multi-class scatterplots of the dimensionally reduced node embedding [3]. Fig. 1b summarizes these previous approaches to evaluation.

However, these existing evaluation approaches fall short because they do not sufficiently support developers in directly understanding the correspondences between input graphs and their node embeddings. Developers must contend with a GNN model as a black box, because it is hard to check whether all important information is codified in the embedding. Even if they find errors in the downstream predicted node labels, it is hard to even trace these errors back to the upstream GNNs, or to refine the GNNs to avoid them.

Our key idea is to aid GNN evaluation by surfacing the correspondences, or lack thereof, between an input graph and its node embedding produced by a GNN. Fig. 1c illustrates this approach. Geometrically speaking, a GNN

- *Z. Liu and T. Munzner are with the Department of Computer Science, University of British Columbia. Email: zipeng, tmm@cs.ubc.ca*
- *Yang Wang is with Facebook Inc. Email: yvv@fb.com*
- *J. Bernard is with the Department of Computer Science, University of Zurich. Email: bernard@ifi.uzh.ch*

*Manuscript received June 15, 2021; revised XXX.*

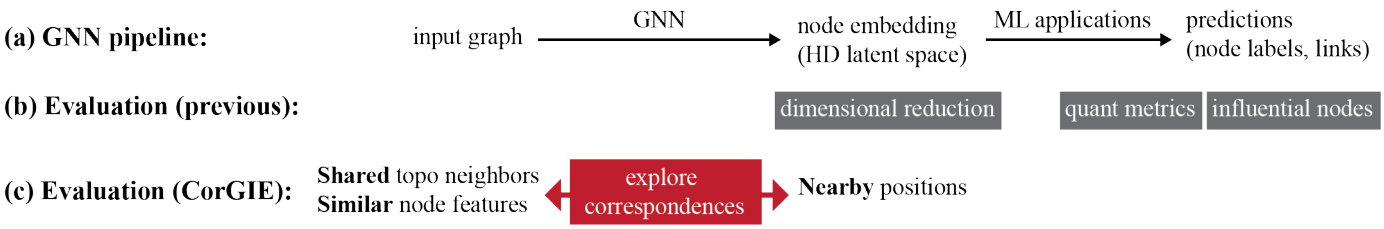


Fig. 1. CorGIE motivation. (a) Standard pipeline for graph neural network (GNN) training and usage. (b) Previous approaches to evaluate GNNs mostly focus on the predictions [1], [2], [3], or to separately inspect the node embedding; (c) our approach with CorGIE is to directly support exploration of correspondences between an input graph and its node embedding.

should learn to place graph nodes with similar neighbors and feature values in a nearby location in the high dimensional latent space. Therefore, we can evaluate a GNN by verifying how well it preserves the characteristics of the input graph in the node embedding. For instance, one can explore how a cluster in the node embedding corresponds to the similarity of topology and/or node features in the input graph. Conversely, one can explore how close two nodes sharing many topological neighbors are represented in the node embedding. To achieve these goals, we present three contributions.

Our first contribution, presented in Sec. 3.3, is a data and task abstraction for visually exploring correspondences between an input graph and the derived node embedding, to understand if a GNN model has learned important characteristics of the input graph to construct the embedding The abstraction is the result of extensive iterative refinement.

To instantiate the abstraction, our second contribution is the design and implementation of an interactive multi-view interface, CorGIE (Corresponding a Graph to Its Embedding). CorGIE is agnostic to specific GNN models, with a "grey-box" approach that ignores the internal model details like neurons and weights and only assumes that the GNN aggregates information from neighboring nodes. We describe the design of CorGIE in Sec. 4.

To fulfill the important task of showing topological neighbors for nodes of interest, our third contribution is a new visualization technique, the K-hop graph layout. The K-hop layout reveals node neighbors hop by hop, similarly to how a GNN aggregates information, and reveals the clustering structures within the node neighborhoods. Previous graph representation and layout algorithms do not suffice for this scenario. We present the details of this technique in Sec. 5.

We validate our claims in two ways. We provide two usage scenarios that walk through how the CorGIE design supports the task and data abstractions in detail. We also discuss the results of a study with two GNN experts that provide preliminary evidence of the utility and usability of CorGIE.

## 2 Background

We first provide the minimum necessary technical background on GNNs to make this work self-contained. We then describe two concrete usage scenarios that motivate CorGIE. We defer our discussion of the related work to Sec. 7.

### 2.1 Graph neural networks

The widespread encoder-decoder perspective is a useful starting point for understanding general graph neural networks [4]. The encoder combines topology, and optionally node features, to produce a node embedding. The decoder takes the node embedding to make inferences in downstream ML applications. In the GNN literature, the node embedding is an intermediate representation within the pipeline that is not necessarily of direct interest outside it. Our perspective is different: we emphasize inspection of the learned embedding as the key to understanding. Moreover, the downstream ML applications are usually considered as an intrinsic component of the GNN pipeline. In contrast, we separate the downstream ML applications from the preceding pipeline so that CorGIE can support multiple downstream applications. CorGIE's agnostic approach supports both node classification that predicts properties of single nodes, as used for movie tagging or document topic labeling), and link prediction that predicts properties of node pairs, as used for movie recommendation.

A GNN is usually trained using either node labels or node connections depending on the application. For node labels, training can be supervised by separating the nodes into training, validation, and test sets if all node labels are available, or semi-supervised if only a small number of labels are available. For node connections, training is unsupervised. Two nodes connected by an edge are considered a positive node pair, whereas a negative node pair has no edge between them; both positive and negative pairs are sampled during training.

A GNN learns from the input graph with **neighborhood aggregation** (aka message passing), where the embedding of a node is computed by traversing to its topological neighbors to collect and aggregate node features from them. A GNN has multiple layers, allowing aggregation to take place hop by hop: each node aggregates the output of its neighbors from one layer to the next. The input layer takes the node features, the last layer outputs a high dimensional node embedding, and the $k$-th layer aggregates from the $k$-th hop neighbor. The number of layers $K$ is usually small, with $K \leq 3$, indicating that it is a shallow network. GNNs differ in the aggregation functions, ranging from a simple mean to complicated non-linear functions [5]. Our approach does take $K$ as a parameter, but is fully agnostic to the details of the aggregation function.

### 2.2 Motivating scenarios

In this paper, we refer to the target users of CorGIE as *GNN developers*, ranging from users who consume pre-trained GNNs to experienced researchers who can propose new model architectures.

Before diving into the details of our approach and tool, we present two representative scenarios of a hypothetical GNN developer, Alice. For each, we introduce a dataset, which consists of an input graph and the embedding constructed by the trained GNN model, and a set of visualization tasks that apply to it. We provide extensive additional examples of concrete tasks in Supplemental Sec. S1.

#### 2.2.1 Movie: recommendation

Alice has a bipartite movie-user graph[1], where an edge indicates that a user has watched or rated a movie. Node features in this graph depend on the node type: the movie node features are budget, popularity, vote average, #cast, #crew; the user node features are vote average and #votes. The goal is to recommend movies to users; that is, the downstream ML application is link prediction. Alice trains a GCN (graph convolutional network, a well-known GNN model [6]) with the node connections and node features in the bipartite graph, to produce a 16-dimensional node embedding. She chooses the hyperparameter setting of 16 based on her previous experience.

To evaluate the training results, Alice wants to understand what the GNN has learned. Besides the typical quantitative metrics like accuracy, she wants to examine:

1) The overall clustering structure of the node embedding, to answer questions like *Are the movies separated from the users? Are there clusters of users?*
2) Within a user cluster, she is interested in why nodes are grouped together: *Do they watch similar movies; that is, do the movies share first hop neighbors? Do they rate movies with similar scores? Do movies they have watched have similar budgets?* Similarity in topology and/or feature values indicate the GNN has done a good job in grouping these users.
3) Between two user clusters, she wants to see their differences: *Do they watch different sets of movies? Do they rate movies differently?* Significant differences indicate that the GNN has done a good job in differentiating the two user groups.
4) Instance-level inspection of movie recommendations is Alice's next undertaking. She wants to spot-check the list of recommended movies for some user, for example a recommendation of *Interstellar* for a viewer who has seen *The Matrix* and *Guardians of the Galaxy*. She wants to understand why the GNN has generated them: *Are recommendations determined by shared viewing patterns, as with collaborative filtering [7]? Are they motivated by similar node features? Do they capture topic similarity, like science fiction?*

#### 2.2.2 Cora: paper topic labeling

The Cora dataset is commonly used as a benchmark in the GNN literature [8]. It is an academic paper citation graph, where the node features are a dictionary of 1433 one-hot vectors showing whether specific words exist in papers.The downstream ML application is to classify the papers according to a set of given topics (classes). Alice trains a GAT (graph attention network, another well-known GNN model [9]) with the node labels in the Cora graph, and produces a 7-dimension node embedding because there are 7 classes of paper topics. She thus converts the values for each dimension to probabilities of belonging to that class, which is a common practice in the GNN literature.

Following a similar analysis process for the movie scenario, Alice first explores the clustering structure by visualizing shared topological neighbors and similarity of node features from an overview level. Then, she tries to understand the pattern of misclassified nodes. She selects groups of these nodes that are either located in the same area of latent space or have the same predicted label; that is, the same error. She inspects the shared neighbors and the words within groups to determine whether there are commonalities to blame for the misclassification.

1. Extracted from a Kaggle dataset: https://www.kaggle.com/rounakbanik/the-movies-dataset

## 3 Data and Task Abstraction

After considering many examples of concrete tasks faced by GNN developers, we generalized them to understand the problem at a higher level. We formed the data and task abstractions iteratively, over several months, through four parallel thrusts: review of the GNN and visual analytics literatures, informal interviews of four GNN developers within three different organizations, development and use of prototypes of the CorGIE interface, and reflection between paper authors.

### 3.1 Overview

We generalize the many concerns faced by GNN developers as they train their models into three driving questions. For memorability, we frame them as questions that might be asked on a road trip:

- Q1: **are we there yet:** should we train or tune more?
- Q2: **are we lost:** does it behave as we expect?
- Q3: **what's that:** what does this exemplar do?

The first big-picture question concerns the potential need to keep training the model for more epochs, tuning the hyperparameters, or reconsider how they construct the graph. It would be answered by understanding whether the results are satisfying. The second question, also big-picture, concerns developers' understanding and trust in whether the GNN learns inherent characteristics from the input graph; that is, does it behave as they expect. These two big-picture questions are particularly difficult to answer with previous evaluation approaches, which rely heavily on quantitative metrics for the downstream ML application; we aim to provide visual and qualitative evidence and insights to answer these two questions in a more detailed and thorough way. The third question is more specific, concerning investigating exemplar instances of nodes or links. Some previous approaches like the GNNExplainer [2] do support such single-instance inspections, but we seek to make them easier and faster.

We posit that finding the correspondences among 1) the node positions and clustering structure in the latent space, 2) the topological neighbors in the input graph, and 3) the node feature distributions in the input graph, can shed light on these questions. We thus discuss three abstract data spaces, as illustrated in Fig. 2: latent space, topology space, and feature space.

### 3.2 Data Space Abstraction

Our data abstraction features three sub-spaces: the latent space, the topology space, and the feature space. We compute distance metrics in each of these spaces. We also incorporate prediction results if available. We follow the order of views in the CorGIE interface to introduce the three spaces.

#### 3.2.1 Latent space

The **latent space** is where the node embedding learned by the GNN lives. Each node in the input graph has its own high-dimensional vector representation. The number of dimensions is usually less than a thousand, although this number does not affect the complexity of the tasks and thus is not a constraint for the CorGIE design. CorGIE assumes K$\leq$3, where $K$ is the number of layers in the network.

In the latent space, the absolute vector values are usually not directly interpretable. We are interested in the vector similarities and particularly, whether there is a clustering structure among the node vectors. For example, in the *Cora* scenario, a relevant question is whether there are exactly seven clusters of papers in the latent space, which would match the seven topics in the ground truth.

The scope of this paper is node embedding; less popular cases like edge embedding and graph embedding are left for future work.

#### 3.2.2 Topology space

The **topology space** consists of the topological connectivity of one single input graph. The input graph could be either homogeneous, with a single type of nodes, or heterogeneous, with multiple node types. The design target for CorGIE is to handle up to 6 node types, which covers many useful cases for heterogeneous graphs [10]. For graph size, CorGIE will accommodate up to 20K nodes, to handle many prevalent benchmark datasets in the GNN literature [1].

In the topology space, we are interested in the topological neighborhoods and the connections (edges) between node pairs. GNNs aggregate information along edges to compute the node embedding. We thus derive node's **neighbor set** containing its k-hop neighbors; that is, the set of all of the nodes reachable from a node $v$ within $k$ topological hops.

We limit our scope to one input graph, leaving the multiple graph case to future work. We also leave the support of multiple edge types for the future.

#### 3.2.3 Feature space

The **feature space** consists of all features (often referred to as attributes in the visualization literature) of the graph nodes. We distinguish between dense and sparse features. Dense features are numeric or boolean features that can be collected and understood independently, with interpretable semantic meanings for each one, such as *budget* or *average vote* from the *Movie* scenario.

Sparse features are usually collected together and share a combined semantics, where interpretation typically takes place across the entire set considered as a whole. Typically they are stored as one-hot encoded categorical features like city names or the dictionary of words from the *Cora* scenario, where the collection is represented by hundreds or thousands of bit vectors.

In the feature space, we are interested in the distributions of feature values, which involves deriving aggregations. We deep-dive into the details of feature aggregation in Sec. 4.2.4.

The design target of CorGIE is to handle up to a dozen dense features and up to 2K sparse features. Note that we deem edge or graph features out of the scope of this paper.

#### 3.2.4 Distance metrics

To quantify correspondences between spaces, we introduce a distance metric between any pair of node within each space, so that scalar distance values across different spaces can be compared. In the latent space, we derive the cosine distance of the embedding vectors, which is commonly used in the downstream ML applications. In the topology space, we derive a topologically oriented distance measure, the inverse Jaccard index for the full set of k-hop neighbors of each pair of nodes in the input graph. This measure is based on the Jaccard Index: the intersection over union of the neighbor set. It is intuitive and easy to compute, but the flattened neighbor sets are sometimes an oversimplification because neighbors in different hops cannot be distinguished from each other. In the feature space, we derive the Euclidean distance of feature vectors (scaled linearly to $[0, 1]$), which is also familiar to GNN developers.

#### 3.2.5 Prediction results (optional)

After training is complete, the GNN can be used for prediction in the downstream ML applications (Fig. 1). We leverage prediction data by comparing the predicted results to ground truth when available, to derive true/false negatives and positives. For node classification applications, we compare the true and predicted node labels to derive a label correctness value (*correct* vs *wrong*) for each node. For link prediction applications, we obtain the predicted positive and negative node pairs, and compare these to the edges of the input graph. We derive labels for two interesting categories of node pairs: *false positives*, where unconnected node pairs are predicted as connected (recommended pairs); and *false negatives* which are node pairs connected with an edge but predicted as disconnected. We also provide options to derive the true positives and negatives if desired. Although the fundamental data and task abstractions do not rely on these prediction results, they can be used for filtering.

### 3.3 Task abstraction

We identify a unifying task abstraction built around an iterative cycle of two phases, as shown in Fig. 2: **specify** items in any of the three data spaces, and **correspond** items between either the latent space and topology space,

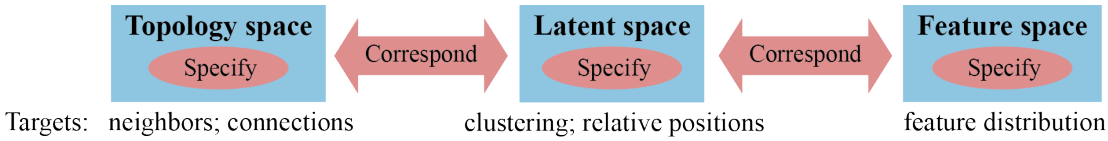


Fig. 2. Data and task abstraction. Users can specify items within all three data spaces, and investigate correspondences between the latent space and topology or feature spaces. The targets of activity in each space are listed below it.

or the latent space and feature space. Notably, our task abstraction does not entail finding correspondences between the topology and feature spaces, because that exploration would focus only on characteristics within input graph and would not provide direct insight into GNN behaviors.

The *specify* step allows users to indicate nodes of interest in any of the spaces, based on the targets visible within each of them. In the latent space, they could specify nodes based on relative positions with respect to any visible cluster structure, or with respect to the latent distance distribution. In the topology space, they could specify nodes based on neighbors, cliques, or the topological distance distribution. In the feature space, they could specify nodes of specific feature values. If available, they can also specify a collection of nodes based on the prediction results such as label correctness or newly predicted links between nodes. Below, we refer to a group of specified nodes as a focal group, which has a double meaning: it is the mental focus within a user's thoughts at the current step of their data analysis process, and it is the focus of actions and computation within the CorGIE interface.

The *correspond* step then allows users to explore to what extent the characteristics of the specified nodes in one space correspond to those in the other spaces. For example, for a tight cluster in the latent space, users could verify whether the nodes in the cluster share many neighbors in the topology space and have similar features in the feature space. If so, the GNN has successfully learned how to group these nodes together. The extent to which these spaces line up with each other is a qualitative judgement call, not a precisely computable metric. The intent of this abstraction is to support GNN developers in obtaining higher confidence and trust in their models than would be possible from purely quantitative summary metrics such as accuracy.

The abstraction supports the process of *specify* and *correspond* as an iterative loop, where exploring correspondences can trigger the user's interest in specifying other sets of nodes. For instance, the user might start by specifying a cluster in the latent space, and after checking its correspondences to the topology space would become curious to then specify a sub-cluster within it to explore further connections. Multiple cycles of refining and changing the sets of specified nodes based on the visual correspondences allow users to connect the dots and find answers all the three of the driving questions.

This task abstraction encompasses all three of our driving questions. It was developed through considering the commonalities between all of the concrete tasks that we analyzed, not only those presented in the motivating scenarios (Sec. 2.2) but also many additional concrete examples summarized in Supplemental Sec. S1. During the abstraction process, we initially considered a much more complex set of targets and actions, such as analyzing outliers with respect to a cluster, or analyzing one node vs. all other nodes, or whether verifying differences should be treated differently than verifying similarities. In the end, we realized that almost all of these questions could be framed much more simply, in terms of specifying a very small number of groups as a target and then checking on correspondences. It was rare to require even three groups: one or two groups were sufficient for almost all analysis questions. Our final design is optimized for up to two groups, although it is possible to specify several more to handle edge cases.

## 4 CorGIE Design

Based on the data space and task abstraction, we design CorGIE, a multi-view tool that reveals the correspondence between an input graph and its embedding. We describe the design of each view, then the view coordination between them, and finally the implementation.

### 4.1 Overview

Fig. 3 shows an overview of the CorGIE interface on the movie scenario, which has seven main views. The LATENT SPACE VIEW is on the top left. Two views at the bottom right show the topology space, the GLOBAL TOPOLOGY VIEW and a novel graph layout for a user-chosen subset of nodes in the K-HOP TOPOLOGY VIEW. The feature space is showcased at the top right in NODE FEATURES VIEW. To enable direct and quick correspondence exploration, we have two views in the middle that can incorporate two or three spaces simultaneously. We introduce the LATENT NEIGHBOR BLOCKS VIEW to connect the latent and topology space, and the DISTANCE COMPARISON VIEW to connect all three spaces. Finally, there is a SETTINGS view at the bottom left.

To support the two steps of *specify* and *correspond* in the task abstraction, the interaction design of CorGIE involves three actions: *hover*, *select*, and *focus*.

Node specification occurs through a *focus* action, and correspondence exploration is triggered through *hover* and *select* actions. The *focus* action creates one or a few special node groups of interest, denoted by *foc-0*, *foc-1*, and so on. When *hovered* or *selected*, nodes are then highlighted across views. When nodes are *focused*, there will be a novel graph layout showing their k-hop topological neighbors.

### 4.2 View design

We introduce the visual encoding and design rationale for each view.

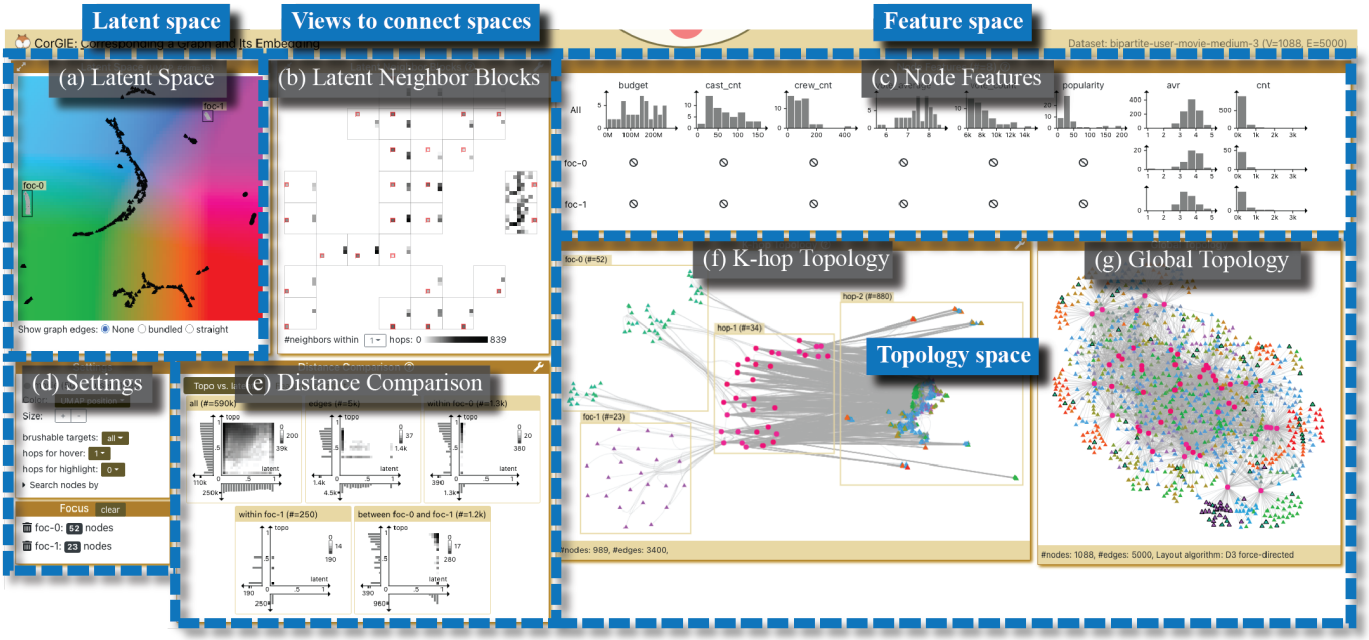


Fig. 3. Full screenshot of CorGIE interface on the *Movie* dataset, with two focal groups of user nodes. The views, with names shown in grey boxes, are laid out in four major areas on the screen, shown in blue: (a) the LATENT SPACE VIEW is for 2D node positions in latent space; (c) the NODE FEATURES VIEW is for feature distribution of all and focal nodes; (f & g) the TOPOLOGY VIEWS show local topological neighbors and global topology; (b) the LATENT NEIGHBOR BLOCKS VIEW and the (e) DISTANCE COMPARISON VIEW connect different spaces. The fifth area is for toggles and menus: (d) the SETTINGS VIEW.

#### 4.2.1 *Latent space*

The LATENT SPACE VIEW on the upper left of the interface (Fig. 3a, Fig. 6a, Fig. 8a) shows the clustering structure and the relative positions of nodes in the latent space. We use UMAP [11] to project the latent space to two dimensions and plot the dimensionally reduced nodes as a scatterplot. GNN developers are typically familiar with dimensionality reduction and are thus aware that loss of information is inevitable in this process.

CorGIE supports color coding nodes by their UMAP positions in all views, and in that mode a rainbow background for the entire 2D latent space view provides a salient reference for positions within the latent space, as shown in Fig. 3. We chose a highly saturated CIELAB colormap (x axis for the variable $a$, and y axis for $b$) that has substantial color exploitation and strong performance for the synoptic localization task of detecting groups of nodes [12]. The combination of dimensionality reduction and this colormap shows latent space distances, so that similar nodes receive similar colors whereas dissimilar nodes receive dissimilar colors.

Nodes are shape-coded for heterogenous graphs with multiple node types, with the same shape preserved across the latent and topology space views. Edges are off by default to avoid clutter but can be shown on demand in this view, as straight or bundled edges.

#### 4.2.2 *Global topology*

We present graph topology with node-link diagrams in CorGIE. We adopt the D3 force-directed layout [13] for the GLOBAL TOPOLOGY VIEW (Fig. 3g, Fig. 8b). It visualizes the topology of an entire input graph, and is computed once in the pre-processing step. As in the LATENT SPACE VIEW, we encode graph nodes as circles for homogenous graphs and glyphs of different shapes for different node types for heterogeneous graphs. We use straight lines for graph edges and curved lines when edge bundling is enabled.

Although this straightforward backstop layout may provide useful insights in smaller graphs, such global views can be extremely cluttered for larger ones. In that case, our custom layout for exploring local neighborhoods of user-specified subsets of the graph is crucial, in the k-hop topology view.

#### 4.2.3 *K-hop topology*

The K-HOP TOPOLOGY VIEW (Fig. 3f, Fig. 7b & e, Fig. 8d, Fig. 9b, Fig. 10d & e) shows the topological neighbors of the focal node sets specified by users. We devise the k-hop layout algorithm to mimic how information is aggregated in GNNs, with boxes enclosing meaningful groups. On the far left are the focal nodes, with hop-1 neighbors in the middle, hop-2 neighbors to the right, and hop-3 neighbors on the far right. Within the boxes, nodes are clustered using the topological (Jaccard) distance. Full algorithmic details on the layout are provided in Sec. 5.

#### 4.2.4 *Node features*

The NODE FEATURES VIEW on the top right (Fig. 3c) shows feature distributions for up to three sets of nodes: all nodes, and each of the two possible sets of focal nodes (*foc-0* and *foc-1*). Dense and sparse features are visualized differently due to their scale differences.

For dense features, we choose histograms as they are intuitive and well-known for visualizing and comparing distributions of scalar values. They also scale well for the available display space, to handle the design target of up

to a dozen dense attributes. We organize these feature histograms as a matrix. Each column represents one feature, whereas the rows show different collections of nodes. The first row shows the full distribution of all nodes; the next row shows the distribution of the focal node set *foc-0* and underneath that is the set *foc-1*. For example, in Fig. 3c, the two focal groups of user nodes have different distribution in average vote (the second last *avr* column), but little difference in the number of votes (the last column). Note that for heterogeneous graphs, different types of nodes can have different features, so focal node sets could end up with some empty histograms.

In contrast to dense features, there could be thousands of sparse features, and their ordering is not explicitly meaningful. We designed a two-level custom view to make effective use of the screen space in a compact area, as shown in Fig. 8f. The top part is a *feature strip* overview that aggregates the information in the more detailed *feature matrix* part below it. The lower **feature matrix** is a heatmap containing one square cell per feature (e.g., a word `train`), and its luminance encodes the count of nodes with that feature (e.g., the number of papers with the word `train` in the Cora dataset). Conceptually, the heatmap contains a single very long line that is simply wrapped to fit into a rectangular aspect ratio. It is not a true 2D matrix, so the absolute position of a cell in terms of a row/column is not meaningful. The upper **feature strip** aggregates the information in the heatmap into a highly compressed pixel-based depiction where many consecutive features are combined into the same vertical line, with a luminance representing the maximum values across the range. The number of features to aggregate is determined by the available horizontal pixel budget of the view, so that the strip fits within it.

As with the dense features, there is a row for all nodes on top, with a row for each of the two focal groups below to show partial feature distributions. In addition, there is a lower *diff* row derived by subtracting the feature values of the two focal node groups. Fig. 8f shows an example (bottom row) that visualizes differences in the word distribution of *foc-0* and *foc-1*, where the dark strips indicate that there are large difference in those word counts.

In each row, the *feature matrix* can be hidden to save space, or expanded to inspect detail. The luminance values are separately normalized within each row since the value ranges could be quite different, so they each require a separate legend.

#### 4.2.5 Latent neighbor blocks

The LATENT NEIGHBOR BLOCKS VIEW (Fig. 3b, Fig. 8c) overlays topological neighbor distributions on the 2D latent space. The challenge is to show the positions of each node's neighbors in the latent space, for each of the nodes. We do so with aggregation and nesting. We aggregate by partitioning the space into an $8\times8$ grid, to create a coarser representation of the space with 64 *blocks*, and map each node as belonging to the block that encloses it. Within each block, we nest a complete copy of the latent space itself, again at a coarse resolution as an $8\times8$ grid of *cells*. Blocks that do not contain any nodes are omitted from the drawing.

This view is inspired by origin-destination (OD) maps for geo-spatial networks [14], but we show neighbor set distributions rather than geographical movement. Within each high-level block, we outline the single low-level cell that corresponds to its block index with red, like the *origin* in an OD map. For every other cell in the block, we encode with luminance the number of neighbors of nodes that fall within the red-outlined origin cell, like the *destinations* in an OD map. If the red-outlined block and its surrounding cells are darker than others in all blocks, that pattern indicates neighbors in the topology space are still neighbors in the latent space, and the GNN has successfully preserved neighborhood structure. Fig. 8c for the *Cora* scenario illustrates this case.

However, there is a limitation to this visual encoding: it is less informative for a bipartite graph, such as the *Movie* scenario, where movie nodes are only allowed to have user nodes as their first hop neighbors. In this case, as shown in Fig. 3b, the pattern of neighbors separated in the latent space does not reflect evidence of poor GNN performance.

#### 4.2.6 Distance comparison

The DISTANCE COMPARISON VIEW (Fig. 3e, Fig. 7e & 7f) shows distributions of distances in each space and supports comparison between spaces. In the data abstraction (Sec. 3.2.4), we derive one distance metric in each space. This view reveals matches and mis-matches of distances between the topology and the latent space in one tab, and between the feature and the latent space in a second tab. A match constitutes a positive correlation of the distances in two spaces. This view shows pairs of nodes, which may either represent an edge or be disconnected in the input graph.

To present and compare two scalar distributions simultaneously, we combine two histograms and a gridded scatterplot into one chart, where the x-axis represents the distance distribution in the latent space, and the y-axis represents that in the topology or feature space. The number of items to plot in the scatterplots can be huge, so we use a grid-binning approach to avoid over-drawing, and to speed up computation we also down-sample the node pairs if there are more than a million.

Each tab accommodates multiple charts to show different sets of node pairs side by side, such as all, within a focal group, between two focal groups, or a user-customized filtered set based on connectivity and link prediction values (when available). The customization options are shown in Supplemental Fig. S17, including a choice between linear and log scale to handle the variance in data.

For example, the three bottom histograms in Fig. 6e show the latent distance distribution within *foc-0*, within *foc-1*, and between the two groups respectively. We observe that the between-group distances are much larger than the within-group distances, indicating that the two focal groups are two distant clusters in the latent space. Fig. 9a shows a pursuit of problematic node pairs, where brushing and highlighting the bottom-right area in the scatterplot signals a mis-match with a negative correlation between latent and topology distance.

### 4.3 View coordination

To visually underscore the linkages between the views, we carefully maintain the consistency of visual encodings for

nodes across views including the shape, color, and size. The node color is user configurable in the SETTINGS view (Fig. 3d), to either the latent space position (the similarity rainbow), a specific node feature (a sequential ramp), node type (distinguishable categorical colors), or node classification labels if available (also categorical). The shape depends on the node type for heterogeneous graphs, or is a circle in the homogeneous case. The size is also globally configurable in SETTINGS.

We also develop user interactions with consistent semantics across all views. In CorGIE, there are three major interactions from light weight to heavy weight: *hover*, *select*, and *focus*. Typically, they are used in order: *hover* is for quick and temporary exploration, then *select* offers a more persistent visual prompt for nodes of interest that are identified with hover, and finally *focus* stores the currently selected nodes into a persistent group. The look and feel of the interaction is shown in the supplemental video.

The *hover* action is triggered when users mouse over objects, such as a node or edge in the TOPOLOGY VIEWS, and a block in the LATENT NEIGHBOR BLOCKS VIEW. On *hover*, CorGIE reacts with strong visual prompts (Fig. 7c & e): strokes on activated nodes and edges while desaturating the background with a half-transparent mask, black partial distributions on top of the node feature histograms, tooltips and node labels.

The *select* action is triggered by clicking a node/edge or brushing multiple nodes. The visual prompt is highly similar to that for *hover* (Fig. 6c & d), but persists even when users move the cursor away. Users can control whether the neighbors of target nodes are highlighted during the *hover* and *select* actions, with drop-down menus in the SETTINGS VIEW.

The heavy-weight *focus* action enables users to find correspondences for one or two groups of specified nodes. When a focal group is created or modified many views are updated, with changes to the FEATURES and DISTANCE views and the computationally intensive operation of creating a new K-HOP TOPOLOGY layout. CorGIE supports several focus actions: create a new focal group, add-to/single-out/remove-from existing groups, and clear groups. These actions enable users refine and change the specified nodes, as required by the iterative nature of the tasks (Sec. 3.3). Since only selected nodes can be the target of focus actions, the focus menu to choose one of these actions appears only after nodes are selected. It drops down from the top of the window and looks like a corgi dog's paw, in keeping with the system name, as shown in the video and Supplemental Fig. S6. The focal nodes are bounded by boxes and labelled with *foc-0* or *foc-1* in both the FOCAL LAYOUT and LATENT SPACE views.

### 4.4 Implementation

CorGIE is implemented using ReactJS [2] and Redux [3] as the frontend scaffold. We choose Canvas over SVG in most of the views for browser performance. For a graph with more than a thousand nodes, there would be thousands of DOM elements such that re-rendering SVGs on user interactions would be very expensive. We develop a layering system using the Konva library [4] to add visual elements on top of a static canvas in order to avoid the expensive re-render whenever possible. As the *focus* action involves heavy computation that takes seconds and even minutes, we offload it to multiple Web Workers to keep the application responsive, and use the Comlink library [5] to communicate between threads. We open-source the code at https://github.com/zipengliu/CorGIE.

### 4.5 Design alternatives

During the iterative refinement process for the CorGIE interface, we experimented extensively with different alternatives for exploring the neighbor sets of a single focus group, including an Upset-based [15] view for neighbor sets and a partial adjacency matrix with roll-up histograms. We discuss these design alternatives in detail in Supplemental Sec. S3. However, the pixel space required would be infeasible for realistic sizes of graphs (e.g. graphs with hundreds of nodes), and there was no obvious extension to comparing neighbor sets between two focus groups. Our final design relies more extensively on interactive exploration to explore neighbor sets for one or two groups at a time, rather than attempting to visually encode all of possibilities simultaneously. In the case study sessions, a domain expert suggested that we could use the traditional bipartite layout for the *Movie* scenario, but we consider the force-directed layout a more general solution for all kinds of graphs.

## 5 K-HOP LAYOUT

We present a new visual encoding technique, the K-hop graph layout, which is used in CorGIE's K-HOP TOPOLOGY VIEW (Fig. 3f, Fig. 7b & e, Fig. 8d, Fig. 9b, Fig. 10d & e). We present the computational scalability of the K-hop layout in this section, and evaluate its suitability for the tasks in Sec. 6. We also discuss three design alternatives that we considered and their limitations.

### 5.1 Algorithm

We first provide an overview of the layout algorithm, then the technical detail and rationale of each step, and finally we discuss its scalability.

The K-hop layout aims to organize topological neighbors of user-specified nodes by the number of hops and their clustering structure. As illustrated in Fig. 4, it is computed in four main steps: 1) divide relevant nodes into groups, 2) bound nodes within boxes and lay out the boxes, 3) lay out nodes within each box independently, and 4) perform transformations to optimize global readability. An optional final step is 5) edge bundling.

**Divide neighbors.** We divide the nodes into groups: user-specified focal groups, all 1st-hop neighbors of focal nodes, all 2nd-hop neighbors of focal nodes, and so on until the Kth hop (where $K \leq 3$). The rest of nodes are deemed irrelevant and thus discarded, so this view shows only a subset of the nodes in the global layout. This division

2. https://reactjs.org/
3. https://redux.js.org/
4. https://konvajs.org/
5. https://github.com/GoogleChromeLabs/comlink

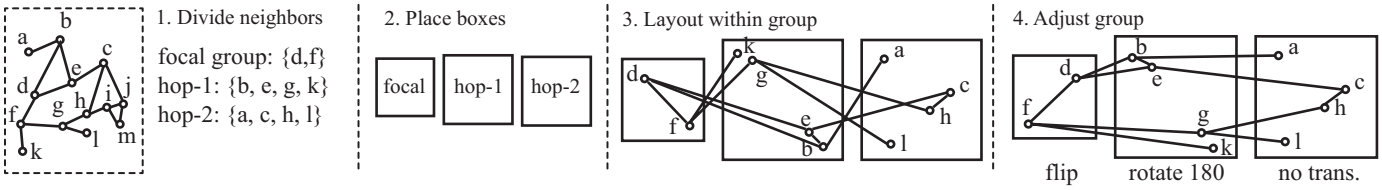


Fig. 4. K-hop layout algorithm in four steps. The optional fifth step is bundle edges (not shown).

naturally matches the neighborhood aggregation in GNNs (Sec. 2.1), by design. Note that a node can only appear in one group to avoid confusing duplication, with priority given to the leftmost view in which it appears.

**Place boxes.** We bound the nodes of groups within boxes and place the boxes from left to right. For the focal groups on the left, multiple focal nodes are placed from top to bottom. This layout mimics how the information flow is typically thought about by GNN developers, as we verified in our interviews with them.

**Lay out within group.** We lay out the nodes within each group independently. To reveal the clustering structure of nodes, we use a dimensionality reduction technique UMAP [11] to reduce a high-dimensional topological distance matrix of nodes down to two dimensions. Users can choose between local distance (connections to previous hop only) or global distance (all K hop connections). The layout falls back to D3 force-directed layout if there are not enough nodes to run UMAP. Because of the independence between groups, we can parallelize the UMAP processes for performance improvement. It would be possible to use alternative layout algorithms in this step like t-SNE [16]; we choose UMAP for its speed and strength in revealing global structure [17].

**Adjust group.** We need to fine-tune the layout from previous step since purely local layout decisions could endanger the global readability. For example, there can be many edge-edge crossings in step 3 of Fig. 4, even though the local layout within each box has been optimized. We use an approach similar to Procrustes analysis [18] to potentially reorient each group. Specifically, we can apply six possible transformations to each group: rotation (0, 90, 180, 270 degrees), and flip (horizontal and vertical). These rigid transformations do not change the relative positions within the group. We find the optimal transformations for each group with a simple enumeration algorithm, that is, to enumerate all 6 possible transformations for all groups for a total of $6^B$ combinations, where $B$ is the number of groups. The node-repulsion Linlog function [19] is used as the readability metric to find the optimum. Note that only node pairs between boxes are considered in the measurement, and we sample them randomly to compute an approximation of the function for performance speedup.

**Bundle edges.** Finally, to further reveal the pattern of connections, we apply edge bundling to the graph layout to reduce visual clutter. Out of several edge bundling algorithms [20], we choose the multilevel agglomerative one [21] for its speed and competitive visual performance. We note that it can introduce distorted edges, which are harder to trace than to unchanged ones, so we allow the user to toggle between straight and bundled edges.

| Dataset | N/E total | N k-hop | E k-hop | N box | B | Times (sec) UMAP (step 3) | Times (sec) total |
|---|---|---|---|---|---|---|---|
| Movie | 1K/5K | 1088 | 5000 | 633 | 4 | 4.1 | 5.5 |
| Movie | 1K/5K | 1088 | 5000 | 370 | 5 | 2.5 | 4.7 |
| Cora | 3K/5K | 989 | 2010 | 576 | 3 | 3.3 | 3.9 |
| Cora | 3K/5K | 2116 | 4222 | 1028 | 4 | 8.5 | 9.6 |
| Coauthor | 10K/54K | 9528 | 43317 | 4254 | 3 | 47.1 | 57.6 |
| Coauthor | 10K/54K | 9951 | 44261 | 6544 | 4 | 110.1 | 123.9 |

TABLE 1
Computational benchmarks for the K-hop layout. N is #nodes, E is #edges: approximate for full graph and exact for used within k-hop layout. N box is maximum #nodes in any box (key dependency), B is #boxes; times in seconds.

## 5.2 Scalability

To aid the perceptual scalability of our approach, our emphasis is on reducing edge-edge crossings, which would hinder users' judgement about topology and connectivity. Our choice to reveal clustering structure through the K-hop layout constitutes a trade-off, where there could be substantial node-node overlap in the K-hop layout.

For computational scalability, we instrument the code to obtain elapsed (wall-clock) timings, and measure the running times with different interactively chosen focal groups for three datasets: *Movie* (N=1K, E=3K) and *Cora* (N=3K, E=5K) as introduced in Sec. 2.2, and *Coauthor* (N=10K, E=54K) extracted from a larger graph of 69K nodes [22]. The experiment is conducted in Chrome on a 2020 MacBook Pro with 2.3GHz 8-core i9 CPU. We present the results in Tab. 1. In addition to the total graph size, we provide the number of nodes and edges used within for the k-hop graph layout (the combination of the focal nodes and their neighbors within k hops). We see that sometimes the k-hop focal layout incorporates all nodes and edges but sometimes shows only a subset of them, depending on the choice of focal groups.

Graphs with 1K nodes like those in the *Movie* dataset take a few seconds to compute, whereas those with 10K nodes take significantly longer. The UMAP computation (step 3) is the computational bottleneck in our algorithm, as shown the timing numbers on the two rightmost columns. We note that the times do not depend on the total number of nodes, but instead, the largest number within a single bounding box, because we run UMAP processes in parallel across multiple worker threads for each box. We thus include the node count within the maximum box in the table. For example, the time for the last run (with 6544 nodes in the most numerous box) of 110 seconds doubles that of the previous run (4254 nodes maximum) of 50 seconds on the same *Coauthor* dataset. Compared to the UMAP computation, the time required for step 1 (divide neighbors), 2 (place boxes), and 4 (adjust group) is negligible; the optional 5th

step (edge bundling) takes about $10-20\%$ of the total time.

### 5.3 Design alternatives

We tried three design alternatives to the K-HOP TOPOLOGY, which were less effective at showing neighbor hops and clustering structure than our final choice: D3 force-directed, WebCola, and space-filling spirals.

The D3 force-directed layout is versatile as we can configure the type of strength of forces. Similarly to the K-hop layout, we use topological distance as the pulling force between nodes, instead of the usual approach of using the edges to control the forces. Fig. 5 (a) and (b) shows two versions of this layout. The first take is bounded by boxes like the K-hop layout, but the nodes are pulled towards other groups and positioned alongside the group boundary as an undesired artifact. It does not show clustering structure within a group. The second take is not bounded strictly but has a strong repulsion force to separate nodes of different neighbor groups. The separation of hops is not obvious enough, especially for homogeneous graphs where the nodes are not shape coded.

The WebCola layout [23] (Fig. 5c) allows us to specify positional constraints between nodes, but due to its force-directed nature, it suffers from a similar problem as the D3 version in Fig. 5a.

The space-filling curve layout [24] is one of the fastest layout algorithms. We choose a spiral curve as it can separate nodes of different groups from center to peripheral. On a polar coordinate system, we use the topological distance as the distance on the curve between two consecutive nodes. From Fig. 5d, we can see that it has some spatial division between different groups but is even worse than the D3 version. It also comes with an artifact that sometimes proximity in the polar coordinate system does not match the proximity in topology: nodes are placed with different radii but similar angles. Moreover, such a spiral space-filling design does not use space effectively, with a lot of unused space in the peripheral area where the less important neighbors reside.

## 6 RESULTS

We evaluate CorGIE in two ways: we use it to address the questions in the two motivating scenarios, and we recruited two GNN experts to participate in a user study to evaluate its utility and usability.

We first present our two usage scenarios. We then describe our study design, and discuss highlights of one of the expert user sessions using CorGIE on the popular Cora dataset. We then summarize the study participants' feedback. We describe the second expert user session in Supplemental Sec. S4.4.

Below, we systematically describe four aspects of CorGIE usage: the *visualization task* being conducted, the *visual operation* used within the CorGIE interface, the *direct observation* that is possible from the CorGIE display, and any resulting *inference* of authors or participants. Here we tag only the visualization task with color; in Supplemental Sec.4 we provide a version of the text with all four aspects color-coded.

### 6.1 Usage scenario 1: Movie Recommendation

To obtain an overview of the dataset, we first color the nodes by their node type (user and movie). Fig. 6a shows that we can see two clusters of movies in blue on the right-hand side and many clusters of users in orange in the rest of the latent space. After coloring the nodes by the underlying positional colormap in the latent space, we observe in the GLOBAL TOPOLOGY view (Fig. 3g) that most user nodes (triangles) are laid out around movie nodes (circles). Many nodes that are connected by only one edge are placed around the periphery of the layout, indicating that there are many users who only watched one movie. In Fig. 6b, we notice that the peripheral users that are connected to the same movies have similar colors: for example, the triangles on the bottom right are all green. We can thus infer that the GNN does a good job in grouping these one-time users reviewing the same movie.

To compare user clusters, we select and focus on two clusters from the LATENT SPACE, one in the green zone (*foc-0*) on the left, the other in the purple zone (*foc-1*) on the upper right, as shown in Fig. 3. In the K-HOP TOPOLOGY (Fig. 3b), there appears to be three green sub-clusters in *foc-0*, and the green nodes connect to many nodes in the hop-1 box, while *foc-1* does not show any salient internal structure. To understand the topological difference between the two focal groups, we explore their connections in the K-HOP TOPOLOGY with hover and select actions. Fig. 6c and 6d show the visual highlights in the K-HOP TOPOLOGY when selecting the green *foc-0* nodes and the purple *foc-1* nodes respectively. We can see that the green nodes in the top *foc-0* box connect to all hop-1 neighbors, but the purple nodes in the lower *foc-1* box only connect to four of those neighbors. We infer that the GNN has learned the topological difference and thus separates them in the latent space.

Besides graph topologies, we also want to check whether distances match between spaces. In the DISTANCE COMPARISON VIEW (Fig. 6e), we can see that the within group distances in latent space are small, while the between group distances are large, which confirms that we have picked two distant clusters from the LATENT SPACE. We notice that the topological distances within the green cluster (*foc-0*) are relatively large considering they belong to the same cluster (leftmost vertical histogram in Fig. 6e), which actually matches the existence of three sub-clusters in the K-HOP TOPOLOGY. The feature distances in all three charts of Fig. 6f are small and similar, indicating that the node features (#votes and average vote of a user) cannot distinguish *foc-0* and *foc-1*. We further confirm the ineffectiveness of the features by reading the feature distances by picking some other node groups (not shown in figure), which indicates that they are not useful features, and could perhaps be removed from the dataset.

After exploring the clustering structure, we inspect instances of recommendation. In Fig. 7a we select the user node *5355*, and list its top 5 recommended movies. The first one is *The Lord of the Rings: the Return of King* (*LotR*). We would like to understand why GNN decides to recommend *LotR*, so we focus this recommendation by clicking on it. CorGIE automatically creates two focal groups with the user *5355*

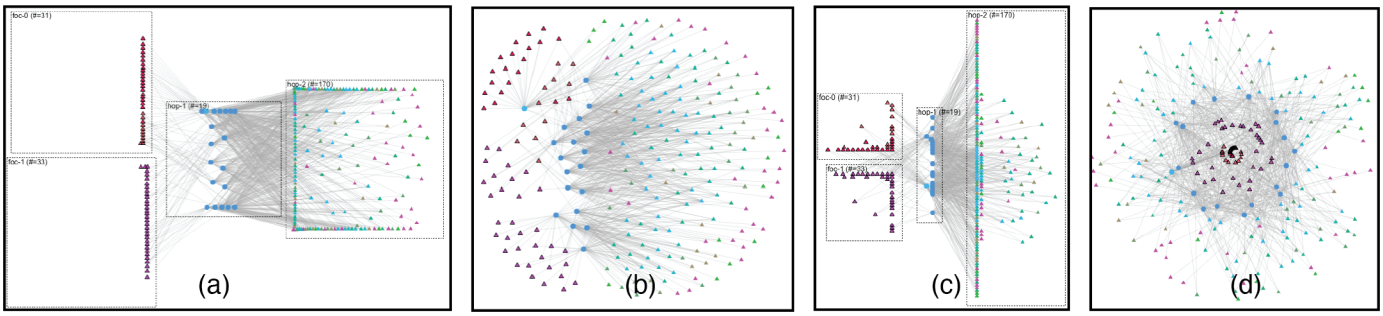


Fig. 5. Design alternatives to K-hop layout: (a) D3 force-directed layout with strict bounding box; (b) unbounded D3 force-directed layout with repulsion force between groups; (c) WebCola force-directed layout with constraints; (d) space-filling curve layout using spiral curve.

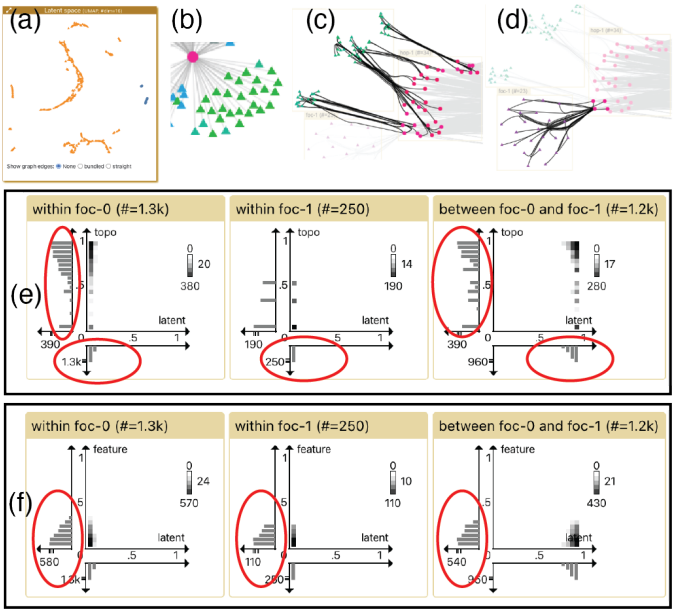


Fig. 6. Check overview of the *Movie* scenario: (a) LATENT SPACE, movie nodes in blue and user nodes in orange; (b) inset of GLOBAL TOPOLOGY showing similar peripheral nodes in similar green color (closeup of lower right for screenshot in Figure 3). When comparing two clusters of user nodes, (c) we select nodes in *foc-1* of the K-HOP TOPOLOGY, highlighting themselves and four neighbors in the hop-1 box, and also in the (d) GLOBAL TOPOLOGY. (e) We compare distances between topology and latent space, and (f) between feature and latent space.

in *foc-0* and movie *LotR* in *foc-1* (Fig. 7b). The K-HOP TOPOLOGY shows that user *5355* watched two movies (in pink) before. When we hover on the two pink nodes in the hop-1 box (Fig. 7c), we find out that each only shares a few users with the recommended movie *LotR*; for example, *Suicide Squad* has only 5 hop-1 shared neighbors. We believe that this recommendation is poor, potentially indicating that we *are not there yet* with GNN training. We further confirmed this problem by reading the topological 0.05 and latent 0.97 distances between user *5355* and movie *LotR* (Fig. 7e), which seem to be correlated negatively.

We later find a recommendation that makes sense: user *587* and movie *LotR*. As shown in Fig. 7e, the two movies that *587* watched, *Inception* and *The Dark Night*, share many users with the recommended movie *LotR* (many hop-1 neighbors are highlighted when hovering on *Inception*). Moreover, the topological distance is 0.72 in this case, which is relatively small compared to the overall distance distribution (not shown in figure). CorGIE thus shows evidence that this recommendation is well supported.

We repeat this process to check other recommendations, and we find many that do not make sense. We conclude that this training result is not satisfactory. It could be due to the "cold start" problem in such a small dataset, where most users only watch 1 or 2 movies. Also, the node features were not very useful: e.g., we know that #votes cannot distinguish users effectively. To improve the recommendation, we might try a different model or fix the dataset problems.

### 6.2 Usage scenario 2: Cora

As with the *Movie* scenario, we first check the clustering structure of paper nodes. We color the nodes by the predicted labels to see if the label distribution makes sense. In LATENT SPACE (Fig. 8a), we can see that the different classes of papers are roughly separated to different areas. In GLOBAL TOPOLOGY (Fig. 8b), although the force-directed layout lacks much structure, we can still see that nearby nodes are in similar colors. In LATENT NEIGHBOR BLOCKS (Fig. 8c), we can see that the red-outlined origin and its surrounding cells are darker for all the blocks. All three observations indicate that the GNN has done a good job in classifying papers using the graph topology.

Next, we inspect and compare a few clusters. For example, when we compare (select and focus) the entire red cluster and a left part of the blue one (Fig. 8d), the K-HOP TOPOLOGY (Fig. 8e) shows that most nodes of the same color connect to each other, and the diff chart in the NODE FEATURES VIEW (Fig. 8f) signals a considerable amount of different words between the two paper clusters (many visible dark strips on the diff row). These observations reinforce our good impression on the training result.

As the GNN seems to classify most nodes successfully, we look for problematic nodes and edges. We brush the distance scatterplot to highlight the node pairs with large latent distances but small topological distances (i.e. the bottom right area), as shown in Fig. 9a, and we focus one of these problematic node pairs. In the K-HOP TOPOLOGY (Fig. 9b), we find through a few rounds of interactive hover that the two focal nodes only share one 1st-hop neighbor (circled in red), which has many nodes connected in the

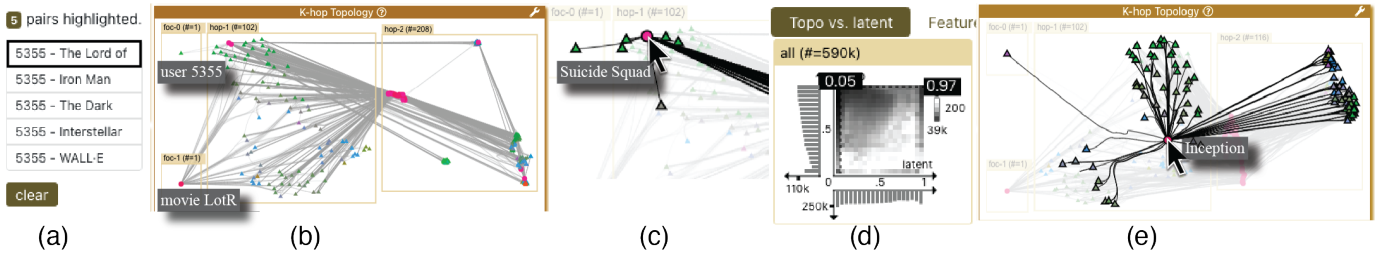


Fig. 7. Check movie recommendations: (a) top 5 recommended movies for user *5355*; (b) K-HOP TOPOLOGY focusing user *5355* and movie *LotR*; (c) hover on the movie *Suicide Squad* that the user has watched before; (d) topological distance between *5355* and *LotR* is large (0.97); (e) K-HOP TOPOLOGY for user *587* and movie *LotR*.

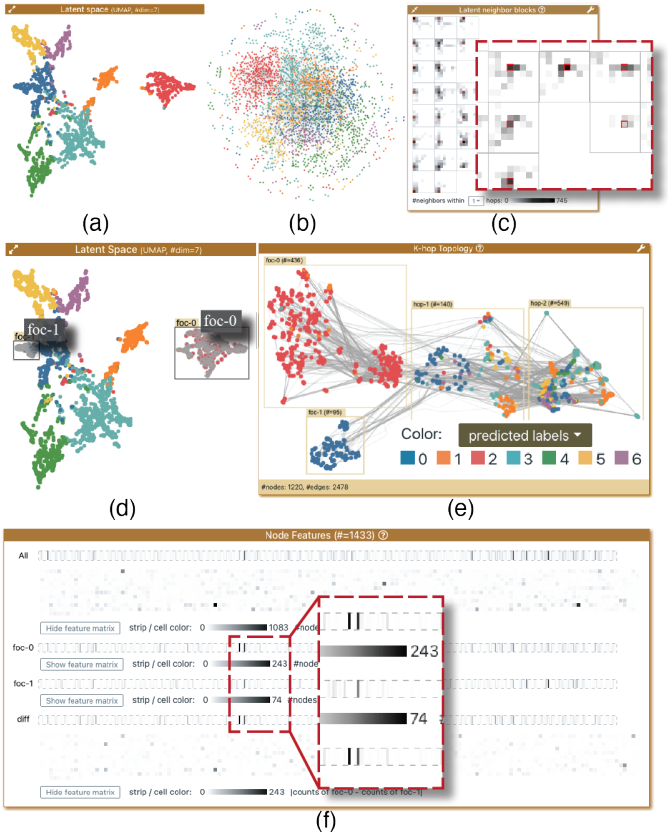


Fig. 8. Explore overview of *Cora* dataset, with nodes colored by predicted label: (a) LATENT SPACE; (b) GLOBAL TOPOLOGY; (c) LATENT NEIGHBOR BLOCKS with zoomed-in inset showing that the red-outlined cells are darker than others. Compare two paper clusters: (d) focus on the red and blue clusters in the LATENT SPACE; (e) K-HOP TOPOLOGY; (f) NODE FEATURES VIEW with inset showing the word count differences between *foc-0* and *foc-1*.

second hop. As the Jaccard distance accounts for multiple hops of neighbors, the large number of shared hop-2 neighbors can explain the low topological distance (0.23). We conjecture that the GNN decides to locate them far from each other due to the large difference in the first hop. Further investigation showed that the node in *foc-0* is mis-classified, which hints at the limitations of this GNN model to deal with such situations.

### 6.3 Expert study overview

One of the two GNN experts is a PhD student (P1), and the other is an industrial lab researcher (P2). Both have

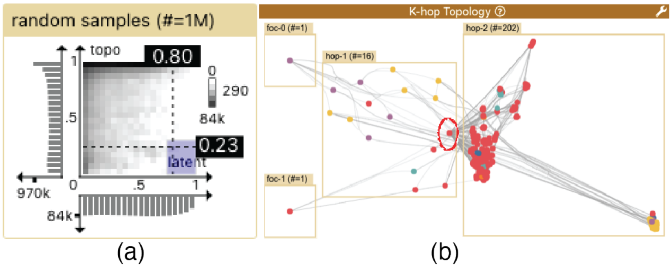


Fig. 9. Find problematic node pairs in the *Cora* dataset: (a) select node pairs with high latent distance but low topological distance; (b) K-HOP TOPOLOGY focusing on a buggy node pair, with only one shared hop-1 neighbor (circled) and many shared hop-2 neighbors.

published multiple GNN papers in top-tier venues in recent years. We had a meeting with each participant a few months before the study when we were iterating on the tasks and design of CorGIE. In this meeting, we discussed the pain points in their workflow of model training and confirmed that our correspondence approach matches their mental model, but did not show any version of the CorGIE interface to the participants at that time. In the remote study session, the first author introduced and demonstrated a near-final version CorGIE for about 40 minutes. Then participants used that version of CorGIE on their own datasets for about 30 minutes, followed by a semi-structured interview to gather their feedback.

The results provide preliminary evidence of the utility of CorGIE in fulfilling the tasks we proposed in Sec. 3.3, as the experts were highly positive about the effectiveness of CorGIE.

### 6.4 Expert session: Cora decision boundary

In this case study, expert P1 first chose to perform quite similar tasks as in our usage scenario. Then he wanted to check the overview to study misclassification. He colored the nodes by label correctness (Fig. 10a), where he saw that the red (misclassified) nodes are distributed across different clusters. He used a filtered brush selection to highlight and focus the wrongly predicted nodes within the middle cluster in the LATENT SPACE. He colored the nodes by the predicted labels (Fig. 10d) and true labels (Fig. 10e) respectively, so he could understand which classes are wrong. It appeared that the GNN predicted these nodes as the orange class similar to their hop-1 neighbors, but the ground truth labels state that they belong to multiple different classes (blue,

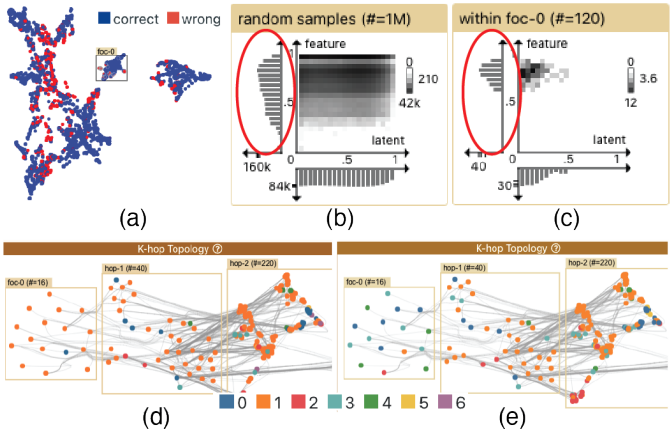


Fig. 10. P1 explored the decision boundary in the *Cora* dataset: (a) he focused on the wrongly predicted nodes within a cluster (in the bounding box) in the LATENT SPACE; he compared (b) feature distances of the entire graph to (c) those within the focused nodes; he inspected neighbors in the K-HOP TOPOLOGY that are colored by (d) predicted label and (e) ground-truth label.

green, red, etc) although the hop-1 neighbors are still orange. Further inspection with hover and brush selection shows that the focal nodes are loosely connected to their hop-1 neighbors, but the hop-1 neighbors are tightly connected to the hop-2 ones. Based on the variance in true labels and the sparsity in topology, P1 inferred that these nodes were sitting at the decision boundary between two or multiple classes in the graph topology. It was his first time to visually inspect the boundary nodes even though he had used this dataset for years. He emphasized the importance of understanding the decision boundary if he needed to improve the GNN model.

After exploring the topology, P1 turned to DISTANCE COMPARISON for more information on the node features, where he found the feature distances within the focal nodes are relatively large compared to the overall distribution (Fig. 10c&b). He conjectured that the current model performed as he expected, and that it would be unlikely to classify the boundary nodes correctly even if he kept training more epochs. He also conjectured that the GNN should use the node features in a better way to correct these nodes. As he confirmed that the GNN has done a good job for most nodes except some boundary ones, he concluded that there is no need to keep training more epochs, and training should be stopped to avoid over-fitting.

### 6.5 Expert feedback summary

We summarize the participants' feedback throughout the sessions, including the questions and discussions while using CorGIE the semi-structured interview at the end. Both participants were impressed after they watched our demonstration and used the tool themselves for the first time.

**Task abstraction.** They confirmed that the two-step task framework (*specify* and *correspond*) is powerful for evaluating GNNs, especially for the iterative refinement of focal nodes, and both thought CorGIE is useful for debugging node embeddings (P1: *"finding decision boundary nodes"*, P2: *"debug [models] quickly in an intense leaderboard competition in ML research"*).

**CorGIE interface.** They stated that most views are straightforward to understand after the introductory explanation, but the heatmap and heat-strips in the NODE FEATURES VIEW are less intuitive. The three-level interaction is easy to use and suitable for the tasks. As for the usability and learnability, the author observed that both participants picked up the the main functions very quickly, which the participants also verbally confirmed in the interview.

However, they criticized the interface as being visually busy/complex, making it difficult to locate a specific object at first glance. We thus improved the interface layouts according to their advice, e.g., we made the *focus* action menu much more salient, changing it to the eye-catching final design of a corgi paw dropping down from the top. Both expressed the excitement to incorporate CorGIE into their research workflow once it is available for deployment.

**K-hop layout.** Both participants consider the K-HOP TOPOLOGY the most useful feature in CorGIE, which validates the success of our novel K-hop layout algorithm. The representation aligns with how they would think about GNNs.

## 7 Related Work

CorGIE relates to previous research on explainable AI, which uses visualization to explain machine learning models. In this section we relate and compare CorGIE to explainable visualizations for non-graph models, latent space, and graph neural networks. We also discuss the related work for our new K-hop layout.

### 7.1 Visualizations for machine learning models

Many visualization tools have been developed to explain and evaluate machine learning models, especially in the recent five years. Readers can refer to many survey papers to have a comprehensive understanding on this topic [25], [26]. Most such tools target models for images/videos and text sequences, most of which are not modeled as graphs. Although the seminal GNN work, GCN [6] stems from convolutional neural networks (CNNs) for images (as they both use convolution operation), it is not practical to simply retarget tools for CNNs to GNNs. The key difference is that pixel neighbors in an image are intrinsically different from topological neighbors in a graph. Sequence data like text in natural languages is even more different. Unlike many visualizations that try to open the "black box" of a neural network and those that keep the "black box" completely opaque, we propose a "grey box" approach that only leverages a key concept in GNN - neighborhood aggregation (Sec. 2.1) - to balance the generalizability and specificity for GNN models.

### 7.2 Latent space interpreter

The study of the latent space, also known as the embedding space, has attracted substantial attention, especially in text-based ML practices. Embeddings eliminate the error-prone process of feature selection and they can be pre-trained for many different downstream applications [27]. As CorGIE

supports exploration of the correspondences between a latent space and a graph, it directly relates to much previous work on latent space visualization.

One theme is to reveal local and global structures of one or multiple latent spaces. Ghosh et al. [28] developed VisExPres, an interactive toolkit for user-driven evaluation of embeddings. Heimerl et al. [29] compare embeddings based on different quantitative metrics, while Cutura et al. does so with dimensional reduction techniques and matrix visualizations [30]. Liu et al. [31] analogize the process of mapping and comparing semantic dimensions within latent spaces to latent space cartography. This theme mainly focuses on digging deep inside latent spaces, which is different from our approach that tries to connect a latent space to the input.

Another theme, dimension reduction (DR), is also loosely related as many non-linear DR techniques produce latent spaces [32]. There are many visual tools for DR results: some for presenting the data points [33], and some connect the DR results back to their semantically meaningful high-dimensional spaces [34], [35]. Their original input is usually tabular data, where a data item has multiple features (aka attributes), but none of them consider graph topology like CorGIE.

### 7.3 GNN interpreter

There has been some effort in the GNN community on GNN interpretation. Most of the previous work focuses on generating algorithms or models to conduct feature and neighbor analysis [36]. Huang et al. [37] propose GraphLIME to produce a few most representative features in the neighborhood of a node. One of the most well-known work is the GNNExplainer [2] by Ying et al., who propose a method to compute the most important nodes of one or a group of user-specified nodes based on information theory. Rao et al. [38] follow up on GNNExplainer and propose xFraud to target at fraud detection specifically. Pope et al. [39] extend explainability algorithms for CNN like saliency map, class activation mapping, and back-propagation to Graph-CNN. This thread of work does not sufficiently support human-in-the-loop visual exploration, and so falls short in providing an overview of how well a GNN learns from the input graph and in connecting the exemplar inspections iteratively.

Two recent papers visualize graph models. Li et al. [40] propose EmbeddingVis to compare multiple graph embeddings generated from different models. It focuses on how the node metrics (e.g. degree, centrality) are preserved in the embeddings, but does not directly support topological neighborhood exploration nor node features. As this research project was conducted before the recent trends of graph neural networks that use neighborhood aggregation, their embeddings are not generated by GNNs but other graph models. A paper by Jin et al. [41], who developed GNNVis to diagnose errors in GNN, is the most similar related work to CorGIE. It only targets one downstream ML application, node classification, which limits its scope of usage. Unlike CorGIE, GNNVis ignores the latent space and takes another route: it supports finding errors in the classification predictions, and comparing against two surrogate models to approximate the erroneous component in GNN.

### 7.4 Graph layout

We discuss the related work for our technical contribution, the K-hop layout. A graph layout survey by Gibs et al. [42] categorizes graph layouts into three approaches: force-directed layouts, dimensional reduction layouts (e.g., MDS, t-SNE, UMAP), and multi-level graph layouts (see survey by McGee et al. [43]). Our K-hop layout combines the DR and multi-level approaches. Out of dozens of layout algorithms, ours is most related to those focusing on clusters. IPSepCoLa [23] by Dwyer et al. is a force-directed layout specializes at separation constraints. Our investigation of its utility for this setting shows that it generates undesired artifacts (Fig. 5c). Our K-hop layout is similar to the Group-In-a-Box layout proposed by Rodrigues et al. [44] for category-based partitions of social networks. It uses the space-filling treemap techniques to separate the clusters, whereas our division into groups is dynamic based on user selections and combines DR techniques. The LinLog layout [19] proposed by Noack is an energy-based model for cluster separation, and it inspires our readability metric in the adjustment step.

## 8 Discussion and Future Work

On reflection, after the whole process of characterizing the problem, designing the interface, and evaluating it, we believe that the high-level idea of exploring correspondences among input, output, and internal data structures is useful for GNN interpretation and may be generalizable to other deep neural networks. As a neural network itself is usually not interpretable for many reasons (e.g. non-linearlity, layering), the approach to "open the black box" completely demands substantial complexity of interpretation, and demands high expertise from the users. We prefer a "grey-box" approach balancing the exposure of internal structure to a nonzero but minimum level. Though often not exposed to the end users for interpretation, the node embedding is universal to all GNNs. By finding correspondences between it and the input data (graph), we can infer if the GNN has achieved satisfactory results without fiddling around the internal structure of a neural network. We imagine this approach could be generalized for other types of neural networks and ML applications.

We would also like to promote our notion of data spaces. Our data abstraction consists of the three data spaces – latent, topology, and feature – alongside a task abstraction that connects them. This mental model of connecting data spaces helps us design the views: some are dedicated for a single space, while some connect multiple spaces. It also contributes to the usability and learnability of CorGIE, for which we have preliminary evidence through the feedback from two GNN experts. It would be interesting future work to introduce more data spaces, such as a geospatial space to deal with GNNs that specialize in geospatial data [45].

It would be useful future work to handle larger graphs. Due to the K-hop layout computation, the current version of CorGIE cannot guarantee a smooth interactive user experience for graphs with more than 20K nodes. Although many popular benchmark datasets are within this scale, the ML community is moving forward with larger datasets, such as those on the OGB platform [46].

To further extend CorGIE, we could incorporate the algorithms like GNNExplainer [2] either in the K-HOP TOPOLOGY VIEW implicitly or in a dedicated view explicitly. This idea was also independently brought up by one of the GNN experts. We could also support more ways to specify nodes, such as topological statistics. Finally, we could enable the comparison of multiple node embeddings of the same input graph, to help evaluate model re-architecting and hyperparameter tuning. A future paper could study how to enable GNN developers to explore the correspondences between multiple input graphs and their node/graph embeddings.

## 9 Conclusion

In this work, we present a task abstraction for exploring the correspondences between an input graph and the latent space created by a GNN, to understand if GNN has learned important characteristics from the graph and to find bugs in the latent space. Based on this abstraction, we develop an interactive multi-view tool, CorGIE, which is validated through usage scenarios and case studies with GNN experts. As the most important component in CorGIE, we propose the K-hop graph layout to reveal how GNNs aggregate information for nodes of interest. Both case studies and expert studies validate the effectiveness of bringing CorGIE into a GNN model development life-cycle. We envision that our novel data and task abstraction, in conjunction with our design rationales and implementation considerations, could serve as a stepping stone for future researches.

## Acknowledgments

The authors would like to thank Madison Elliott, Steve Kasica, Michael Oppermann, Ben Shneiderman, and Mara Solen for helpful comments on paper drafts, and the anonymous participants in the user study. This work was funded in part by Uber Technologies and by NSERC RGPIN-2014-06309.

## References

[1] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[2] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems (NeurIPS*, vol. 32, pp. 9240–9251, 2019.

[3] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding." in *WWW*. ACM, 2015.

[4] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *ArXiv*, vol. abs/2005.03675, 2020.

[5] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Trans. Association for Computational Linguistics (ACL)*, vol. 5, pp. 101–115, 2017.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[7] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[8] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[10] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Trans. Knowledge and Data Engineering (TKDE)*, 2020.

[11] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[12] J. Bernard, M. Steiger, S. Mittelstädt, S. Thum, D. Keim, and J. Kohlhammer, "A survey and task-based quality assessment of static 2d colormaps," in *Visualization and Data Analysis 2015*, vol. 9397. International Society for Optics and Photonics, 2015, p. 93970M.

[13] M. Bostock, V. Ogievetsky, and J. Heer, "$D^3$ data-driven documents," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 17, no. 12, pp. 2301–2309, 2011.

[14] J. Wood, J. Dykes, and A. Slingsby, "Visualisation of origins, destinations and flows with OD maps," *The Cartographic Journal*, vol. 47, no. 2, pp. 117–129, 2010.

[15] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, "UpSet: Visualization of intersecting sets," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 20, no. 12, pp. 1983–1992, 2014.

[16] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[17] A. Coenen and A. Pearce, "Understanding UMAP," https://pair-code.github.io/understanding-umap/, 2019.

[18] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[19] A. Noack, "Energy models for graph clustering." *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 453–480, 2007.

[20] H. Zhou, P. Xu, X. Yuan, and H. Qu, "Edge bundling in information visualization," *Tsinghua Science and Technology*, vol. 18, no. 2, pp. 145–156, 2013.

[21] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger, "Multilevel agglomerative edge bundling for visualizing large graphs," in *2011 IEEE Pacific Visualization Symposium*, pp. 187–194.

[22] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA, 2008, p. 990–998.

[23] T. Dwyer, Y. Koren, and K. Marriott, "IPSep-CoLa: An incremental procedure for separation constraint layout of graphs," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 12, no. 5, pp. 821–828, 2006.

[24] C. Muelder and K.-L. Ma, "Rapid graph layout using space filling curves," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 14, no. 6, pp. 1301–1308, 2008.

[25] A. Chatzimparmpas, R. M. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren, "The state of the art in enhancing trust in machine learning models with the use of visualizations," *Computer Graphics Forum*, vol. 39, no. 3, pp. 713–756, 2020.

[26] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 25, no. 8, pp. 2674–2693, 2019.

[27] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[28] A. Ghosh, M. Nashaat, J. Miller, and S. Quader, "VisExPreS: A visual interactive toolkit for user-driven evaluations of embeddings," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, pp. 1–1, 2020.

[29] F. Heimerl, C. Kralj, T. Moller, and M. Gleicher, "embcomp: Visual interactive comparison of vector embeddings," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, pp. 1–1, 2020.

[30] R. Cutura, M. Aupetit, J.-D. Fekete, and M. Sedlmair, "Comparing and exploring high-dimensional data with dimensionality reduction algorithms and matrix visualizations," in *Proc. Intl. Conf. Advanced Visual Interfaces (AVI)*. ACM, 2020.

[31] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent space cartography: Visual analysis of vector space embeddings," *Computer Graphics Forum*, vol. 38, pp. 67–78, 06 2019.

[32] L. G. Nonato and M. Aupetit, "Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and

layout enrichment," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 25, no. 8, pp. 2650–2673, 2018.
[33] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," *arXiv preprint arXiv:1611.05469*, 2016.
[34] J. Stahnke, M. Dörk, B. Müller, and A. Thom, "Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 22, no. 1, pp. 629–638, 2016.
[35] R. Faust, D. Glickenstein, and C. Scheidegger, "DimReader: Axis lines that explain non-linear projections," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 25, no. 1, pp. 481–490, 2019.
[36] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *arXiv preprint arXiv:2012.15445*, 2020.
[37] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "GraphLIME: Local interpretable model explanations for graph neural networks," *arXiv preprint arXiv:2001.06216*, 2020.
[38] S. X. Rao, S. Zhang, Z. Han, Z. Zhang, W. Min, Z. Chen, Y. Shan, Y. Zhao, and C. Zhang, "xFraud: Explainable fraud transaction detection on heterogeneous graphs," *arXiv preprint arXiv:2011.12193*, 2020.
[39] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 764–10 773.
[40] Q. Li, K. S. Njotoprawiro, H. Haleem, Q. Chen, C. Yi, and X. Ma, "EmbeddingVis: A visual analytics approach to comparative network embedding inspection," in *IEEE Conf. Visual Analytics Science and Technology (VAST)*, 2018, pp. 48–59.
[41] Z. Jin, Y. Wang, Q. Wang, Y. Ming, T. Ma, and H. Qu, "GNNVis: A visual analytics approach for prediction error diagnosis of graph neural networks," *arXiv preprint arXiv:2011.11048*, 2020.
[42] H. Gibson, J. Faith, and P. Vickers, "A survey of two-dimensional graph layout techniques for information visualisation," *Information Visualization*, vol. 12, no. 3-4, pp. 324–357, 2013.
[43] F. McGee, M. Ghoniem, G. Melançon, B. Otjacques, and B. Pinaud, "The state of the art in multilayer network visualization," in *Computer Graphics Forum*, vol. 38, no. 6. Wiley Online Library, 2019, pp. 125–149.
[44] E. M. Rodrigues, N. Milic-Frayling, M. Smith, B. Shneiderman, and D. Hansen, "Group-in-a-box layout for multi-faceted analysis of communities," in *IEEE Third International Conf. Privacy, Security, Risk and Trust and IEEE Third International Conf. Social Computing*, 2011, pp. 354–361.
[45] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.
[46] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.