Explaining Robot Action Predictions Using Graph Embeddings in Imitation Learning

Lukas Probst
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: lukas.probst@student.kit.edu

Manuel Córcoles Villas
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: manuel.villas@student.kit.edu

***Abstract*—Imitation learning is increasingly leveraged to train robots on complex manipulation tasks by replicating expert demonstrations. However, the interpretability of robot action predictions remains a significant challenge, limiting trust and hindering real-world applicability. In this paper, we systematically investigate the use of graph embedding explainability techniques—specifically, GNNExplainer, PGExplainer, and AttentionExplainer—to interpret predictions made by Graph Neural Networks (GNNs) in robotic imitation learning contexts. Utilizing the RoboCasa simulation framework and focusing on a drawer-closing task, we quantitatively assess each explainer's effectiveness using adapted fidelity metrics for regression tasks and Euclidean distances. Complementary qualitative evaluations employing dimensionality reduction methods (UMAP, PCA, and t-SNE) further demonstrate the explanatory power and structural consistency of the embeddings. Our results indicate that PGExplainer consistently provides superior fidelity and interpretability, while attention-based methods offer computational efficiency. This work advances explainability in robotic learning, facilitating more transparent and trustworthy deployments.**

I. Introduction

Imitation learning has become a promising approach for enabling robots to acquire complex manipulation skills by observing expert demonstrations, thus overcoming limitations associated with traditional reinforcement learning methods such as sample inefficiency and sparse reward signals [1], [2]. However, a significant challenge in imitation learning arises from the "black-box" nature of neural network models, which often leads to a lack of interpretability and explainability, hindering trust and effective debugging in real-world deployments [3], [4].

Graph Neural Networks (GNNs) have emerged as a powerful tool for capturing structural relationships in graph-structured data, such as robotic environments or task representations, and have demonstrated superior performance in various predictive tasks [5], [6], [7]. However, interpreting GNN predictions remains challenging due to their inherently relational and complex decision-making processes. Several methods have been proposed to address this interpretability gap, such as GNNExplainer [8], PGExplainer [9], and AttentionExplainer [10], each targeting different aspects of explanation generation by identifying crucial nodes, edges, or subgraphs influencing model predictions.

In parallel, recent advances in embedding techniques have allowed efficient representation of complex data in lower-dimensional spaces, significantly improving performance in tasks involving images [11], [12], [13] and graph structures [14], [15], [6]. These embeddings provide a valuable basis for generating meaningful explanations by mapping high-dimensional decisions into interpretable forms.

In this report, we propose a systematic approach for explaining robot action predictions in imitation learning scenarios using graph embeddings. We leverage state-of-the-art GNN explainability methods and systematically evaluate their performance using standardized metrics such as fidelity and Euclidean distance, adapted specifically for continuous-valued regression tasks prevalent in robotics. Furthermore, we qualitatively assess embedding visualization techniques—UMAP [16], PCA [17], and t-SNE [18]—to determine their effectiveness in preserving local and global embedding structures.

Our experiments are conducted within the RoboCasa [19] simulation framework, focusing specifically on a manipulation task involving drawer closing. By combining quantitative and qualitative evaluations, we provide comprehensive insights into the capabilities and limitations of current GNN explanation techniques, ultimately contributing towards more transparent, reliable, and interpretable robotic systems.

II. Related Work

A. GraphFramEx

GraphFramEx [4] provides a systematic evaluation framework for explainability methods in Graph Neural Networks (GNNs). It benchmarks various techniques based on fidelity, stability, and robustness, helping assess their effectiveness in different graph settings.

The framework introduces standardized evaluation metrics and datasets to compare explainers fairly. It highlights key limitations in existing methods, such as instability in explanations and lack of generalizability across different graph structures. By providing a unified benchmark, GraphFramEx facilitates the development of more reliable and interpretable GNN explainability approaches.

B. Embeddings in machine learning

Embeddings play a crucial role in representing complex data in lower-dimensional spaces while preserving important structural information.

1) Image embeddings: Image embeddings map high-dimensional image data into a lower-dimensional feature space, capturing semantic and structural information. Convolutional Neural Networks (CNNs) [20] have been widely used to extract feature representations from images. Pre-trained models such as ResNet [11] and Vision Transformers (ViTs) [12] generate robust embeddings that can be used for downstream tasks like classification and retrieval. More recently, contrastive learning approaches, such as CLIP [13], align image and text representations, enabling zero-shot learning capabilities.

2) Graph embeddings: Graph embeddings transform nodes, edges, or entire graphs into continuous vector spaces while preserving graph structure and relationships. Traditional methods include spectral embeddings [21] and random walk-based approaches like DeepWalk [14] and Node2Vec [15]. More recently, Graph Neural Networks (GNNs) [5] have become the dominant approach for learning graph embeddings by iteratively aggregating neighborhood information. Frameworks like GraphSAGE [6] and GAT [7] improve embedding quality by using inductive learning and attention mechanisms, respectively.

By leveraging these embedding techniques, we can effectively process both image and graph data.

C. RoboCasa

RoboCasa is a large-scale simulation framework developed for benchmarking and evaluating generalist robotic agents across various everyday household tasks [19]. It provides realistic virtual environments featuring diverse manipulation scenarios, such as drawer manipulation, object relocation, and kitchen interactions, designed to test both task-specific performance and the agents' ability to generalize. By systematically varying environment configurations, distractors, and task complexity, RoboCasa allows rigorous assessment of model generalizability and robustness in realistic household settings [19].

III. Fundamentals

A. PCA

Principal Component Analysis (PCA) [17] is a widely used linear dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional subspace while preserving as much variance as possible. In the context of deep neural networks it has been used after feature extraction in CNNs to lower the number of dimensions of the embeddings for reducing the computation costs in the classification and analysis tasks[22]. PCA works by transforming a high dimensional dataset into a new coordinate system where the principal components (new axes) capture the most variance in the data [23]. Given an observation matrix X of size n × p, where p is the number of dimensions in the space and n the number of vectors, S is the variance-covariance matrix (derived from X), and A is a matrix of orthonormal eigenvectors of S, PCA effectively tries to replace the set of original variables with their linear combinations, which are contained in the matrix V.

$$V = A^T X \tag{1}$$

PCA is still widely used, as a result of its simplicity and efficiency for linearly separable data [24], and has been applied in numerous fields, including image recognition [25], image compression [26], medicine [27], and financial analysis [28].

B. UMAP

Uniform Manifold Approximation and Projection (UMAP) [16] is a non-linear dimensionality reduction technique grounded in Riemannian geometry and topological data analysis. UMAP has emerged as a powerful tool for various data types, including imaging flow cytometry (IFC) [29], spectral imaging of artworks [30] and genomic data analysis [31]. As explained in [16], it models high dimensional data as a weighted graph, capturing local neighborhood relationships using a fuzzy simplicial set representation. The method then optimizes a lower-dimensional embedding to best preserve these relationships while maintaining global structure effectively. Mathematically, UMAP constructs a weighted k-nearest neighbor graph and optimizes its low-dimensional representation by using the following cross-entropy loss function. The cross-entropy C measures the difference between two fuzzy sets (A, μ) and (A, ν), where A is a reference set, and μ and ν are membership strength functions mapping elements of A to values in $[0, 1]$. The function is defined as:

$$C((A,\mu),(A,\nu)) \triangleq \sum_{a\in A}\left(\mu(a)\log\frac{\mu(a)}{\nu(a)} + (1-\mu(a))\log\frac{1-\mu(a)}{1-\nu(a)}\right) \tag{2}$$

C. t-SNE

T-Distributed Stochastic Neighbor Embedding (t-SNE), proposed in [18], is a popular non-linear dimensionality reduction and data visualization method. Unlike PCA, which relies on linear projections to maximize variance, t-SNE models pairwise similarities between data points. For this purpose it uses conditional probabilities and optimizes a low-dimensional representation by minimizing the Kullback-Leibler (KL) divergence between a joint probability distribution in the high-dimensional space and a joint probability distribution in the low-dimensional space. The cost function C, given a joint probability distribution P in the high-dimensional space and a joint probability distribution Q in the low-dimensional space is defined as:

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}}. \tag{3}$$

p_{ij} and q_{ij} are conditional probabilities defined as:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)} \tag{4}$$

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}. \quad (5)$$

T-SNE has been widely used in different fields in recent years, such as in financial analysis [32], medicine [33] and social network analysis [34].

D. GNNExplainer

GNNExplainer [8] is a model-agnostic explainability method designed to interpret predictions of Graph Neural Networks (GNNs). It identifies the most influential subgraph and node features by optimizing a mask over the input graph, ensuring that the selected substructure retains the model's original prediction.

The method applies a differentiable mask to the adjacency matrix and node features, optimizing it using a loss function that balances prediction accuracy, sparsity, and interpretability. While effective, GNNExplainer is limited by scalability issues for large graphs and provides only local instance-based explanations. Despite these limitations, it remains a foundational approach in GNN interpretability and serves as a benchmark for newer methods.

E. PGExplainer

PGExplainer (Parameterized Graph Explainer) is a model-agnostic explanation technique developed to efficiently interpret predictions from Graph Neural Networks (GNNs) [9]. Unlike instance-specific methods such as GNNExplainer, PGExplainer employs a parameterized neural model that learns a generalizable mapping from graph inputs and node embeddings to edge importance scores. This learned model significantly improves scalability, as explanations for new instances can be generated rapidly through a simple forward pass, without additional optimization steps.

PGExplainer is initially trained using supervised signals provided by explanations generated from computationally expensive "teacher" explainers. Once trained, it quickly produces edge-level explanations, making it particularly suitable for real-time and large-scale deployments.

F. AttentionExplainer

Graph Attention Networks (GAT) [35] introduced an attention mechanism for graphs, allowing nodes to attend to their neighbors with different importance levels. A major improvement over GAT was proposed in GATv2 [10], which ensures dynamic attention weights by making the attention mechanism input-dependent.

The AttentionExplainer utilizes GATv2 to generate explanations for graph-based models by identifying important nodes and edges based on learned attention scores. The core of this approach is the attention mechanism, which updates node features as

$$\mathbf{x}_i' = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{i,j} \Theta \mathbf{x}_j, \quad (6)$$

where $\mathbf{x}_i'$ represents the updated feature vector of node i, $\mathcal{N}(i)$ denotes the set of neighbors of node i, and $\alpha_{i,j}$ are the learned attention coefficients. The trainable weight matrix Θ transforms node feature vectors before aggregation.

The attention coefficients $\alpha_{i,j}$ determine how much influence a neighboring node j has on node i and are computed as

$$\alpha_{i,j} = \frac{\exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\Theta \mathbf{x}_i + \Theta \mathbf{x}_j\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\Theta \mathbf{x}_i + \Theta \mathbf{x}_k\right)\right)}. \quad (7)$$

Here, $\mathbf{a}$ is a learnable attention vector that determines the importance of neighboring nodes, while the LeakyReLU activation introduces non-linearity into the attention mechanism.

If edge features $e_{i,j}$ are present, the attention mechanism extends to

$$\alpha_{i,j} = \frac{\exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\Theta \mathbf{x}_i + \Theta \mathbf{x}_j + \Theta_e e_{i,j}\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\mathbf{a}^\top \text{LeakyReLU}\left(\Theta \mathbf{x}_i + \Theta \mathbf{x}_k + \Theta_e e_{i,k}\right)\right)}, \quad (8)$$

where Θ_e is an additional trainable matrix that applies transformations to edge features.

AttentionExplainer leverages these learned attention weights to provide insights into model behavior by highlighting which nodes and edges contribute most to predictions. With GATv2's improvements over its predecessor, explanations generated by AttentionExplainer are more robust, as they reflect input-dependent attention dynamics rather than static weight assignments [10].

G. Fidelity metric for explainability in graph neural networks

The fidelity metric is a widely used evaluation measure for explainability in Graph Neural Networks (GNNs) [4], [8]. It assesses how well the identified explanatory subgraph aligns with the original model's decision-making.

For classification tasks, fidelity is commonly defined as:

- **Fidelity+**: Measures the change in the model's prediction when the explanatory subgraph G_S is removed from the original graph G. A high fidelity+ indicates that the removed part significantly influenced the decision.

$$fid_+ = 1 - \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left(\hat{y}_i^{G \setminus S} = \hat{y}_i\right),$$

  where N is the number of instances, and $\mathbb{1}$ is an indicator function.
- **Fidelity-**: Measures how well the explanatory subgraph G_S alone can reproduce the original prediction.

$$fid_- = 1 - \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left(\hat{y}_i^{G_S} = \hat{y}_i\right).$$

A lower fidelity- value is desirable, indicating that G_S captures the essential part of the decision-making process.

These definitions are suited for classification problems but need adaptation for regression tasks, as discussed in Section V-C1.

IV. Methods

A. Dimensionality reduction

In order to work with the dimensionality reduction techniques mentioned in Section III we used their implementation provided by the Scikit-learn [36] and UMAP-learn [16] libraries . Our main purpose was to apply the dimensionality reduction techniques for each of the model's output embeddings for a given dataset in order to be able to provide a visualization for each technique and a further comparison between them. We followed the following steps for each technique:

- Convert and prepare the model's output embeddings into an appropriate type for their future reduction.
- Initialization of the dimensionality reduction techniques.
- Application of the initialized technique to the prepared models embeddings.

In the initialization step, the techniques share most common attributes, but we only defined two and left the rest in their default setting, as a study of the effect of their parameters is out of the scope of this project. We used the *n_components* parameter, which is the number of dimensions of the resulting embeddings space, in our case 3. We also declared the *random_state* parameter, which serves for the random number generator, in order to have reproducible results across multiple function calls.

B. Explaining graph embeddings through the graph itself

We employ PyTorch Geometric (PyG), a widely-used library optimized for deep learning on graph-structured data [37], to generate explanations for Graph Neural Network (GNN) predictions. PyG provides an accessible, flexible, and efficient interface for implementing diverse explainability methods such as GNNExplainer [8], PGExplainer [9], and Attention-based explainers [10], enabling intuitive interpretation and evaluation of learned graph embeddings.

Our implementation uses a structured and modular codebase centered around the custom `GeometricExplainer` class. Each explainer algorithm is encapsulated independently, ensuring reusability and clarity in the evaluation pipeline. Specifically, our setup comprises the following key steps:

- Initialization of separate `GeometricExplainer` instances for training and testing datasets.
- Generation of baseline visualizations using the `create_base_graph` method.
- Application and comparative analysis of multiple explanation algorithms on both training and test trajectory datasets.
- Quantitative assessment using standardized metrics such as fidelity and Euclidean distance.
- Visualization of explanations in both static graph images and animated GIFs to aid qualitative analysis.

Below, we describe each explainer's configuration and rationale.

1) GNNExplainer: The GNNExplainer [8] algorithm identifies critical graph components contributing to the GNN predictions by optimizing masks applied to nodes and edges. In our implementation, we leverage PyG's built-in `GNNExplainer`, configured with these hyperparameters:

- Number of epochs: 100
- Learning rate (`lr`): 1×10^{-4}
- Edge mask type: `object`
- Node mask type: Evaluated both with and without an `object`-based node mask.

GNNExplainer explanations enhance interpretability by isolating influential subgraphs directly responsible for the model's decision-making process.

2) PGExplainer: PGExplainer (Parameterized Graph Explainer) [9] introduces a parameterized approach to graph explanation by learning a predictive explanation model, thus achieving higher efficiency and scalability compared to non-parameterized methods like GNNExplainer. Our setup for PGExplainer employs the following configuration:

- Number of epochs: 100
- Learning rate (`lr`): 1×10^{-2}
- Explanation type: `phenomenon`-level explanations, suitable for our scenario as it captures general behavior across multiple instances. PGExplainer does not support `model`-level explanations.
- Edge mask type: `object`, focusing explanations explicitly on object interactions within graphs.

Due to its learned predictive mechanism, PGExplainer is particularly effective in scenarios with large-scale datasets or when computational resources are constrained.

3) AttentionExplainer: The AttentionExplainer leverages attention coefficients from attention-based Graph Neural Network (GNN) architectures such as GAT, GATv2, and TransformerConv [35], [10], [38]. These attention scores indicate the importance of individual edges during model predictions, making AttentionExplainer particularly suitable for edge-level interpretability.

In our implementation, we utilize PyTorch Geometric's built-in `AttentionExplainer` configured as follows:

- Default reduction strategy for attention coefficients (`reduce`): `max`, aggregating attention scores across all layers and heads by selecting the maximum value.
- Explanation type: `model`-level, providing direct interpretability of the model's internal decision-making process.
- Edge mask type: `object`, since AttentionExplainer specifically provides explanations at the edge level and does not support node-level masking.

AttentionExplainer offers efficient, real-time interpretability by directly accessing the model's learned attention weights

without additional training overhead, thus enabling immediate insights into the critical edges influencing GNN predictions.

V. Experiments

To systematically evaluate and compare the explanatory methods introduced in the previous section, we perform a set of experiments using training and testing trajectories. These trajectories, containing continuous state-action embeddings generated by our agent, are stored in HDF5 format for efficient handling and reproducibility.

Specifically, our experimental evaluation includes:

- **Training dataset**: 49 trajectories consisting of 9317 individual data points, i.e. actions.
- **Test dataset**: 30 trajectories (23 successful and 7 not successful) consisting of 7454 individual data points, i.e. actions.

Experiments are structured into two complementary parts:

- A qualitative analysis that evaluates different dimensionality reduction techniques (PCA, UMAP, t-SNE) by applying a set of distance metrics to each consecutive action in the graph embeddings of the trajectories.
- A rigorous quantitative assessment of the graph explanation methods (GNNExplainer, PGExplainer, AttentionExplainer) using fidelity (adapted for regression tasks with cosine similarity) and Euclidean distance to systematically evaluate and compare their performance.

Moreover, we examine the explanations specifically within the context of the `CloseDrawer` task from the RoboCasa dataset [19], as detailed below.

A. Tasks

In our current evaluation, we focus solely on a single RoboCasa task to demonstrate our proposed explainability approach. As an extension, future experiments could explore multiple distinct RoboCasa tasks to comparatively analyze how explanation metrics differ across various task embeddings. By measuring distances between embeddings, one could investigate task-specific embedding structures and determine whether explanatory effectiveness varies across tasks.

1) CloseDrawer: We specifically consider the `CloseDrawer` task from the RoboCasa dataset [19], which involves closing a kitchen drawer within a simulated environment. This task represents a single-stage manipulation scenario, characterized by straightforward interactions between the robot and a drawer within a simulated kitchen scene. Each trajectory in the dataset represents a unique sequence of states and actions required to complete this task.

Figure 1 illustrates a representative frame from one of the training trajectories, demonstrating the task environment and the drawer object that must be closed.

This single-stage kitchen task provides a suitable baseline for evaluating explanation algorithms due to its simplicity, clearly defined goal, and relatively low complexity.

Fig. 1: Representative frame from the third training trajectory of the `CloseDrawer` task in RoboCasa.

B. Qualitative experiments

We qualitatively compare the dimensionality reduction techniques using three different distance metrics and two locality preservation metrics.

1) Distance metrics: To quantify the changes in graph embeddings across consecutive steps in the different trajectories we used the following distance metrics, which will be formally explained below.

- **Mean Squared Error**: measures the average squared difference between corresponding elements of two embeddings vectors, providing an estimate of their dissimilarity. This allows us to assess the stability of the learned embeddings. The metric ranges positive values starting from 0, where a higher MSE indicates a shift in the reduced representation. Given two embeddings E_t and E_{t+1} at consecutive steps t and $t+1$, MSE is computed as:

$$MSE(E_t, E_{t+1}) = \frac{1}{d}\sum_{i=1}^{d}(E_t^i - E_{t+1}^i)^2 \tag{9}$$

  where d is the dimensionality of the reduced embedding space, and E_t^i and E_{t+1}^i are the i-th components of the embeddings at time steps t and $t+1$.
- **Cosine Distance**: quantifies the angular separation between two vectors, capturing directional differences rather than magnitude differences. This is particularly useful for analyzing embedding spaces, where vector orientations may convey more meaningful relationships than absolute distances. It ranges from 0 to 2, where 0 indicates that there is minimal change (embeddings point in the same direction), and higher values suggest greater angular deviation. Given two embeddings E_t and E_{t+1} at consecutive steps t and $t+1$, the Cosine Distance is computed as:

$$D_{\cos}(E_t, E_{t+1}) = 1 - \frac{E_t \cdot E_{t+1}}{\|E_t\|\|E_{t+1}\|} \tag{10}$$

where $E_t \cdot E_{t+1}$ represents the dot product of the two embeddings, and $\|E_t\|$ and $\|E_{t+1}\|$ denote the Euclidean norms of the respective embeddings.

- **Euclidean Distance**: measures the straight-line distance between two points. It provides a direct measure of the absolute displacement between embeddings. A lower value indicates a small distance between the points, whereas a higher value suggests a more significant distance. Given two embeddings E_t and E_{t+1} at consecutive steps t and $t+1$, the Cosine Distance is computed as:

$$D_{\text{Euc}}(E_t, E_{t+1}) = \sqrt{\sum_{i=1}^{d}(E_t^i - E_{t+1}^i)^2} \tag{11}$$

where d is the dimensionality of the embedding space, and E_t^i and E_{t+1}^i represent the i-th components of the embeddings at time steps t and $t+1$.

2) Locality Preservation: To measure the extent to which the local and global relationships are preserved, we employ two complementary metrics: Spearman Rank Correlation and Degree of Locality Preservation. These methods help evaluate the quality of the dimensionality reduction techniques by assessing how well they retain the original structure of the data.

- **Spearman Rank Correlation**: This metric has been used in other projects, such as [39], to quantify how well the relative ordering of distances is maintained between the original and reduced spaces. Given a dataset with n points, we first compute the pairwise distance matrix in the original and reduced spaces. Let D_X and D_Y denote the flattened vectors of pairwise distances from these matrices. Spearman's correlation is defined as the Pearson correlation coefficient between the ranked versions of these two vectors:

$$\rho(D_X, D_Y) = 1 - \frac{6\sum_{i=1}^{m}(r_{X,i} - r_{Y,i})^2}{m(m^2-1)} \tag{12}$$

where $m = \frac{n(n-1)}{2}$ is the number of unique pairwise distances in the dataset, $r_{X,i}$ and $r_{Y,i}$ are the ranks of the i-th distance in the original and reduced space, respectively. A Spearman correlation close to 1 suggest that relative distances are well preserved, while lower values indicate distortions.

- **Degree of Locality Preservation (DLP)**: While Spearman correlation assesses global and relative distance preservation, the Degree of Locality Preservation (DLP) provides a more fine-grained evaluation of local structure retention in manifold learning techniques [40]. The DLP score measures how well the local neighborhood of each data point is maintained after dimensionality reduction. It is defined as:

$$DLP = \frac{1}{n}\sum_{i=1}^{n}\frac{|N_k(x_i) \cap N_k(y_i)|}{|N_k(x_i)|} \tag{13}$$

where $N_k(x_i)$ is the set of k nearest neighbors of point x_i in the original space, and $N_k(y_i)$ is the corresponding set in the reduced space. The DLP score ranges from 0 to 1, with higher values indicating stronger locality preservation.

This metric complements Spearman Rank Correlation by ensuring that the nearest neighbors in the high-dimensional space remain close in the lower-dimensional representation, providing an additional perspective on the quality of dimensionality reduction.

C. Quantitative experiments

We quantitatively assess the quality of explanations using two primary metrics: Fidelity and Euclidean distance. The fidelity metrics used here are commonly adopted in evaluating explanation methods for Graph Neural Networks [4], [8].

1) Adaptation for regression using cosine similarity: Since our problem is a regression task, the standard fidelity metric (Section III-G) must be adapted. Instead of using categorical comparisons common in classification tasks, we use cosine similarity to measure the alignment between continuous-valued predictions.

The cosine similarity between two prediction vectors $\mathbf{a}$ and $\mathbf{b}$ is given by:

$$\text{cosine similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}. \tag{14}$$

To ensure interpretability within $[0, 1]$, we scale the similarity:

$$\text{scaled cosine similarity}(\mathbf{a}, \mathbf{b}) = \frac{1 + \text{cosine similarity}(\mathbf{a}, \mathbf{b})}{2}. \tag{15}$$

Using this, we redefine the fidelity metrics for regression:

- **Fidelity+ (regression-adapted)**:

$$fid_+ = \frac{1}{N}\sum_{i=1}^{N}\frac{1 + \text{cosine similarity}(\hat{y}_i^{G_{\setminus S}}, \hat{y}_i)}{2}. \tag{16}$$

- **Fidelity- (regression-adapted)**:

$$fid_- = \frac{1}{N}\sum_{i=1}^{N}\frac{1 + \text{cosine similarity}(\hat{y}_i^{G_S}, \hat{y}_i)}{2}. \tag{17}$$

Higher values for Fidelity+ indicate that removing the explanatory subgraph significantly affects predictions, implying its importance. Conversely, a high Fidelity- means that the explanatory subgraph alone sufficiently retains the original prediction.

2) Euclidean distance: Euclidean distance complements fidelity by quantifying the magnitude of changes in predictions when subgraphs are added or removed:

- **Euclidean Distance (+)**: measures the average Euclidean distance between original predictions and predictions after removing the explanatory subgraph, highlighting its significance:

$$L2_+ = \frac{1}{N}\sum_{i=1}^{N}\|\hat{y}_i^{G_{\setminus S}} - \hat{y}_i\|. \tag{18}$$

- **Euclidean Distance (-)**: measures the average Euclidean distance between original predictions and predictions on the explanatory subgraph alone, indicating how closely the subgraph captures the prediction:

$$L2_{-} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_i^{G_S} - \hat{y}_i\|. \tag{19}$$

Additionally, we calculate the relative percentage difference between these two metrics to facilitate comparative analyses:

$$L2(\%) = \frac{L2_{-}}{L2_{+}} \times 100\%. \tag{20}$$

These adapted metrics provide robust and meaningful quantitative evaluation of explanations within regression-based GNN models.

VI. Results

A. Qualitative results

Here we show the qualitative results of the dimensionality-reduction techniques UMAP, PCA, and t-SNE applied to the training and test trajectories, with the latter divided between successful and not successful trajectories of the robot.

1) UMAP, PCA, and t-SNE visualizations: The following figures show the results of applying UMAP, PCA, and t-SNE to reduce the graph embeddings produced by the model for the data in train trajectory 1 to three dimensions. Since the trajectory consists of 193 actions, which would result in an overly crowded plot, we decided to plot only the average of the reduced projections at intervals of 10. The color of each point represents the order of the embeddings, this means that points belonging to the first steps in the trajectory are represented with darker colors and points in the last steps of the trajectory are represented with lighter colors. Figure 2a shows the embeddings after applying UMAP, figure 2b the results of PCA and figure 2c the results of t-SNE. Furthermore the results of the Spearman correlation, DLP and computation time for this trajectory are presented in table I.

TABLE I: Qualitative results for the train trajectory 1. Best results are highlighted in bold.

Metric	UMAP	PCA	t-SNE
Spearman Correlation	0.922	**0.971**	0.925
DLP	0.612	0.597	**0.835**
Computation time in seconds	7.914	**0.246**	1.119

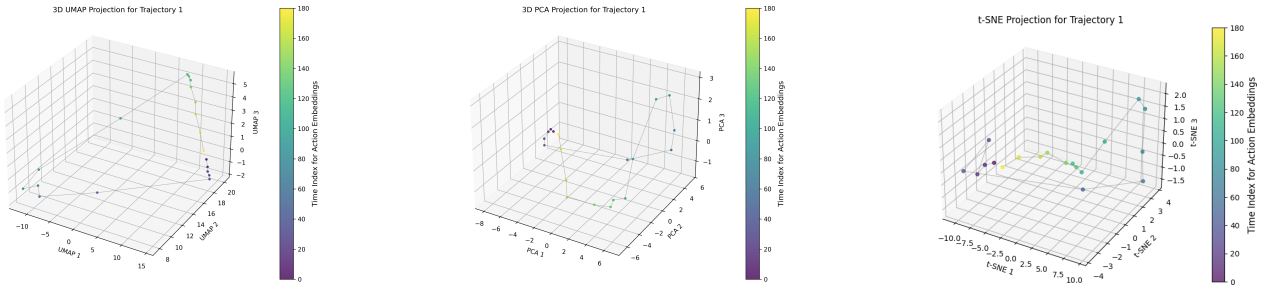
2) Distance metrics results: Figure 3 show the values of the cosine distance metric, calculated between each pair of consecutive points in the dimensionality-reduced space, for different trajectories with the PCA technique. More results of the cosine distance metric for the three algorithms can be found in the appendix A3. Figure 3a shows the results for the train trajectory 0. Figure 3b exhibits the results for the successful test trajectory 0. Finally, figure 3c presents the results for the unsuccessful test trajectory 3.

3) Similarities with end effector coordinates: Figures 4, 5 and 6 show various comparisons between the movement of the end effector and the dimensionality-reduced embeddings for a successful and unsuccessful trajectory and the cosine distances between points in the reduced embeddings. In figure 4 the PCA-reduced embeddings for the successful test trajectory 0 and the movement of the end effector are presented, where again darker blue colours represent earlier embeddings in the trajectory and lighter green values later embeddings. Figure 5 shows a similar comparison but with unsuccessful test trajectory 2. In figure 6 there is another comparison from the Euclidean distances between consecutive PCA-reduced embeddings (see figure 6a) and the movement of the end effector for the train trajectory 0.

4) Interpretation of results: The results presented in figure 2 show how UMAP, PCA, and t-SNE reduce the embeddings to a similar representation in which the embeddings in 3D form a path, or almost a cycle, where the first embeddings are located in a similar area to the last embeddings. The results provided in table I suggest that all three techniques have a similar Spearman Rank Correlation value, with PCA having the highest value. Regarding the DLP values, UMAP and PCA have similar results, but t-SNE provides a much higher value, which means that their reduced embeddings are more representative of the original high-dimensional embeddings. In appendix A there is a further comparison between the representation of the reduced embeddings in test and train trajectories, where we have learned that test trajectories for UMAP, PCA, and t-SNE do not form a similar cycle form, but rather a path where there is a clearer difference between the first and last embeddings.

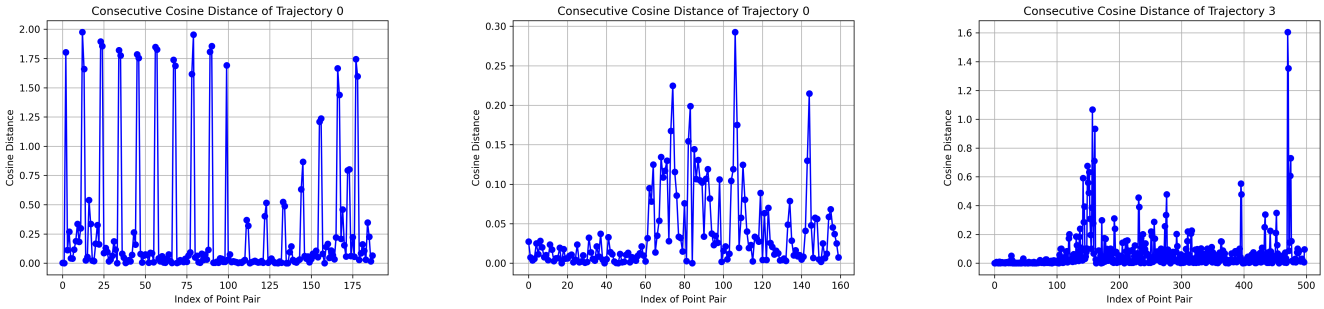
The results shown in figure 3 first demonstrate that the cosine distances are lower for the successful test trajectory than for the unsuccessful test trajectory and train trajectory. A common occurrence between the three types of trajectories are the spikes shown in the distance values, which are highly accentuated in the train and test trajectory. We believe that these greater distances in these type of trajectories are related to the robot trying different movements in the train or in the unsuccessful trajectory to achieve the task. The key takeaway from these results for us is that successful test trajectories present more compact embeddings, as the low cosine distances show, which can be caused by the trajectory being smoother. This effect can also be observed in the appendix A for the UMAP and t-SNE techniques.

The results of figure 4 demonstrate that the form of the path between the reduced embeddings for the successful test trajectory 0 and the movement of the end effector are highly similar. Furthermore this similarity between reduced embeddings and the movement of the end effector can be seen in figure 5. Here, the last embeddings and positions of the end effector are concentrated around the same region. In figure 5a the last reduced embeddings seem to be grouped closely in an unordered, almost erratic manner. In the figure 5b the last positions of the end effector are also close to one another, almost in the same coordinates, which are barely noticeable



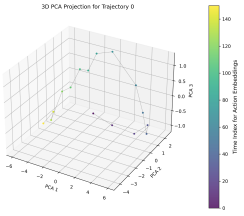
(a) UMAP (b) PCA (c) t-SNE

Fig. 2: Comparison of dimensionality reduction techniques (UMAP, PCA, t-SNE) for train trajectory 1.

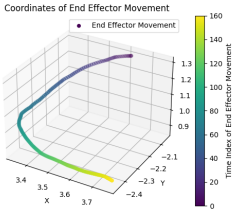


(a) Train trajectory 0 (b) Test trajectory 0 (successful) (c) Test trajectory 3 (unsuccessful)

Fig. 3: Visualization of the cosine distances between consecutive points for PCA across different trajectories.

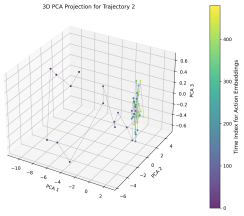


(a) PCA reduced embeddings

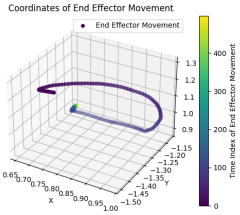


(b) End effector movement

Fig. 4: Comparison of PCA and end effector movement for successful test trajectory 0.



(a) PCA reduced embeddings



(b) End effector movement

Fig. 5: Comparison of PCA and end effector movement for unsuccessful test trajectory 2.

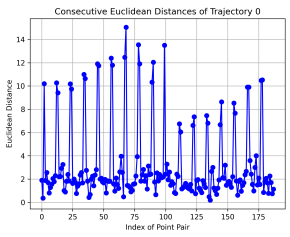
as all these last lighter points are blocked by the darker (and thus earlier embeddings) points. From figure 6 it can be seen that the spikes in the Euclidean distance values are possibly related to the movement of the end effector. At first sight, the movement seems smooth, but the drastic color changes in the path of the end effector mean that the end effector position is jumping from one coordinate to another along the drawn path.

B. Quantitative results

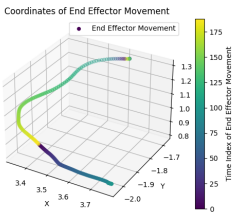
We present quantitative evaluations of the explanation techniques, namely the DummyExplainer, GNNExplainer, PGExplainer, and AttentionExplainer, alongside the BaseGraph, which serves as a reference visualization without any applied explanation methods. Evaluations were performed using the adapted fidelity and Euclidean distance metrics previously defined, applied to both training and test trajectories of the `CloseDrawer` task.

In the visual representations of the trajectories, nodes indicate specific entities within the RoboCasa environment:

- Node 0: Robot
- Node 1: Primary task-relevant object (e.g., drawer)
- Nodes $2, 3, \ldots$: Distractors (irrelevant objects within the environment).



(a) Euclidean Distances



(b) End effector movement

Fig. 6: Comparison of PCA and end effector movement for train trajectory 0.

1) BaseGraph (no explanation): The Base Graph illustrates the full graph structure without applying any masking or explanation algorithm. Thus, it serves as a baseline reference showing the model's original embedding structure, where all edges and nodes are equally visualized.

Figure 7 shows the complete graph structure for a single action in one trajectory, without any explanation applied. All nodes and edges are retained, providing a reference for comparison with explanation methods.

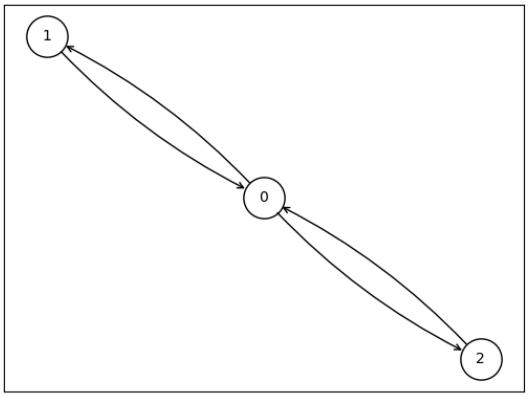


Fig. 7: Visualization of BaseGraph for train trajectory 0, action 44.

2) Baseline (DummyExplainer): The DummyExplainer exhibits fidelity values very close to 1, suggesting that it does not significantly perturb the original model's predictions. Euclidean distances are nearly identical, indicating that the DummyExplainer does not meaningfully alter the learned embeddings.

3) GNNExplainer: GNNExplainer without node masking achieves strong fidelity and low Euclidean distances, effectively capturing important subgraphs. However, applying node masking significantly reduces fidelity and drastically increases Euclidean distances, indicating that removing nodes directly harms explanation quality. This confirms that node features play a crucial role in preserving the model's decision-making process.

Figure 8 illustrates the result of applying GNNExplainer (without node masking) to the same action as figure 7. Compared to the BaseGraph, some edges are removed or deemphasized, highlighting the most relevant connections identified by the explainer. This demonstrates how GNNExplainer extracts a subgraph that retains essential structural information while reducing complexity.

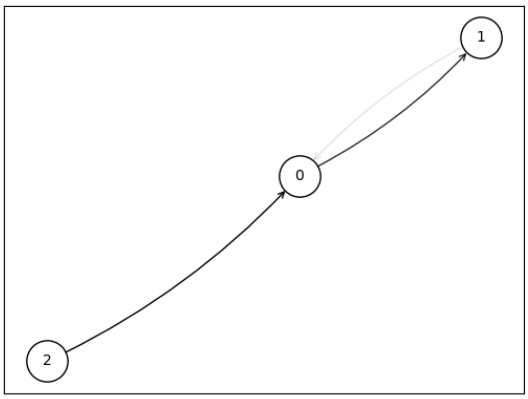


Fig. 8: Visualization of GNNExplainer_no_node_mask for train trajectory 0, action 44.

4) PGExplainer: PGExplainer provides the highest fidelity scores and the lowest Euclidean distances. This suggests that PGExplainer effectively isolates critical components of the model's decision process while preserving interpretability. Its learned parameterized approach enables generalization across different input graphs, making it particularly suitable for large-scale deployment.

5) AttentionExplainer: AttentionExplainer utilizes built-in attention coefficients from the model (GATv2), offering immediate and computationally lightweight explanations. It achieves consistently high fidelity and maintains relatively low Euclidean distances. This suggests that attention-based explanations align well with the model's internal decision-making process while preserving interpretability.

Quantitative comparisons using fidelity and Euclidean distance metrics are summarized for both train and test datasets in Tables II and III. These tables highlight each explainer's performance relative to the BaseGraph.

VII. Conclusion

In this paper, we systematically evaluated and compared multiple graph embedding explanation techniques—namely GNNExplainer, PGExplainer, and AttentionExplainer—in the context of imitation learning for robot action prediction tasks within the RoboCasa simulation framework. Our qualitative analysis using dimensionality reduction methods (UMAP, PCA, and t-SNE) provided explainability for the embeddings,

TABLE II: Quantitative results for the train set over 49 rollouts (trajectories). The GNN model is evaluated with and without node masking. Best results are highlighted in bold.

METRIC	DUMMY	GNN W/O MASK	GNN W/ MASK	PG	ATTENTION
FID+	0.982	0.984	0.626	**0.994**	0.990
FID-	0.982	0.984	0.734	0.997	**0.998**
L2+	3.049	2.927	16.740	**0.170**	3.441
L2-	3.057	2.918	13.169	**0.124**	1.557
L2(%)	100.289%	99.732%	78.660%	72.824%	**45.245%**

TABLE III: Quantitative results for the test set over 30 rollouts (trajectories). The GNN model is evaluated with and without node masking. Best results are highlighted in bold.

METRIC	DUMMY	GNN W/O MASK	GNN W/ MASK	PG	ATTENTION
FID+	0.983	0.984	0.706	**0.998**	0.991
FID-	0.983	0.985	0.789	**0.999**	0.998
L2+	3.047	2.923	14.974	**0.116**	3.354
L2-	3.054	2.920	12.011	**0.111**	1.563
L2(%)	100.187%	99.927%	80.209%	95.698%	**46.602%**

as their visualizations showed that the embeddings represent a path structure, similar to the motion of the end effector of the robot. T-SNE was the most effective in preserving local and global relationships in embeddings, confirmed through Spearman correlation and locality preservation metrics.

Quantitative evaluations demonstrated that PGExplainer consistently achieved the highest fidelity and lowest Euclidean distance, indicating superior explanatory quality and robustness. AttentionExplainer also performed well, offering a computationally efficient alternative by leveraging built-in attention coefficients from GATv2 models [10]. In contrast, applying node masking in GNNExplainer significantly deteriorated explanation quality, underscoring the importance of node features in capturing the GNN model's decision-making process [8].

Overall, our results suggest that parameterized explainers such as PGExplainer [9] and attention-based methods offer promising directions for scalable and interpretable graph embedding explanations in robotic imitation learning scenarios. Further exploration of these techniques across a broader range of tasks and embedding structures could yield deeper insights into their generalizability and practical utility.

VIII. FUTURE WORK

Several aspects remain to be addressed to enhance our proposed approach. Firstly, our evaluations were limited to relatively small-scale and simple graph structures. Future work should investigate how effectively the explored explainability methods scale to larger and more complex graphs, particularly to assess their capability to interpret robot action predictions in more realistic and demanding scenarios.

Additionally, our experiments currently focus on the `CloseDrawer` task within RoboCasa. Expanding this evaluation to a wider range of manipulation tasks could provide valuable insights into the generalizability of different explainability methods and clarify whether certain methods are particularly effective for specific types of tasks or actions.

Furthermore, incorporating human-subject studies could significantly enhance our understanding of practical explainability. Such studies would help determine how effectively these graph-based explanations support real-world users in interpreting robot behaviors, thus providing direct feedback on interpretability and trust.

REFERENCES

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[2] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[3] T. Chakraborti, S. Kambhampati, M. Scheutz, and Y. Zhang, "Emerging topics in explainable and interpretable ai," *arXiv preprint arXiv:2004.13988*, 2020.

[4] K. Amara, R. Ying, Z. Zhang, Z. Han, Y. Shan, U. Brandes, S. Schemm, and C. Zhang, "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2206.09677

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2017.

[6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1025–1035.

[7] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2018.

[8] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1903.03894

[9] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," 2020. [Online]. Available: https://arxiv.org/abs/2011.04573

[10] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" 2022. [Online]. Available: https://arxiv.org/abs/2105.14491

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[14] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2014, pp. 701–710.

[15] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 855–864.

[16] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[17] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://doi.org/10.1080/14786440109462720

[18] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[19] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, "Robocasa: Large-scale simulation of everyday tasks for generalist robots," in *Robotics: Science and Systems*, 2024.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[22] J. Ma and Y. Yuan, "Dimension reduction of image deep feature using pca," *Journal of Visual Communication and Image Representation*, vol. 63, p. 102578, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320319301932

[23] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/009830049390090R

[24] A. Name, "Dimensionality reduction: Pca, t-sne, and umap," Year of Publication, accessed: 2024-03-12. [Online]. Available: https://medium.com/@aastha.code/dimensionality-reduction-pca-t-sne-and-umap-41d499da2df2

[25] A. Alkandari and S. J. Aljaber, "Principle component analysis algorithm (pca) for image recognition," in *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*, 2015, pp. 76–80.

[26] C. Clausen and H. Wechsler, "Color image compression using pca and backpropagation learning," *pattern recognition*, vol. 33, no. 9, pp. 1555–1560, 2000.

[27] F. Kherif and A. Latypova, "Chapter 12 - principal component analysis," in *Machine Learning*, A. Mechelli and S. Vieira, Eds. Academic Press, 2020, pp. 209–225. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128157398000122

[28] R. M. Mbona and K. Yusheng, "Financial statement analysis: Principal component analysis (pca) approach case study on china telecoms industry," *Asian Journal of Accounting Research*, vol. 4, no. 2, pp. 233–245, 2019.

[29] I. Stolarek, A. SamelakCzajka, M. Figlerowicz, and P. Jackowiak, "Dimensionality reduction by umap for visualizing and aiding in classification of imaging flow cytometry data," *iScience*, 2022, accessed: 2024-03-12. [Online]. Available: https://www.cell.com/iscience/fulltext/S2589-0042(22)01414-6

[30] M. Vermeulen, K. Smith, K. Eremin, G. Rayner, and M. Walton, "Application of uniform manifold approximation and projection (umap) in spectral imaging of artworks," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 252, p. 119547, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1386142521001232

[31] A. Diaz-Papkovich, L. Anderson-Trocmé, C. Ben-Eghan, and S. Gravel, "Umap reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts," *PLoS Genet*, vol. 15, no. 11, p. e1008432, 2019. [Online]. Available: https://doi.org/10.1371/journal.pgen.1008432

[32] A. M. Lopes and J. A. T. Machado, "Dynamical analysis of the dow jones index using dimensionality reduction and visualization," *Entropy*, vol. 23, no. 5, p. 600, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/5/600

[33] M. P. Maurits, I. Korsunsky, S. Raychaudhuri, S. N. Murphy, J. W. Smoller, S. T. Weiss, T. W. J. Huizinga, M. J. T. Reinders, E. W. Karlson, E. B. van den Akker, and R. Knevel, "A framework for employing longitudinally collected multicenter electronic health records to stratify heterogeneous patient populations on disease history," *Journal of the American Medical Informatics Association: JAMIA*, vol. 29, no. 5, pp. 761–769, 2022. [Online]. Available: https://doi.org/10.1093/jamia/ocac008

[34] C. Yuan, Y. He, R. Lin, and Y. Tang, "Graph embedding for scholar recommendation in academic social networks," *Frontiers in Physics*, vol. 9, 2021. [Online]. Available: https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2021.768006

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018. [Online]. Available: https://arxiv.org/abs/1710.10903

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html

[37] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019. [Online]. Available: https://arxiv.org/abs/1903.02428

[38] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," 2021. [Online]. Available: https://arxiv.org/abs/2009.03509

[39] H. Huang, Y. Wang, C. Rudin, and E. P. Browne, "Towards a comprehensive evaluation of dimension reduction methods for transcriptomic data visualization," *Communications Biology*, vol. 5, p. 719, 2022. [Online]. Available: https://doi.org/10.1038/s42003-022-03628-x

[40] X. Zhu, Q. Liu, J. Wu, and P. S. Yu, "Degree of locality preservation: A metric for evaluating the quality of dimensionality reduction," *Communications Biology*, vol. 5, p. 1203, 2022. [Online]. Available: https://www.nature.com/articles/s42003-022-03628-x

APPENDIX

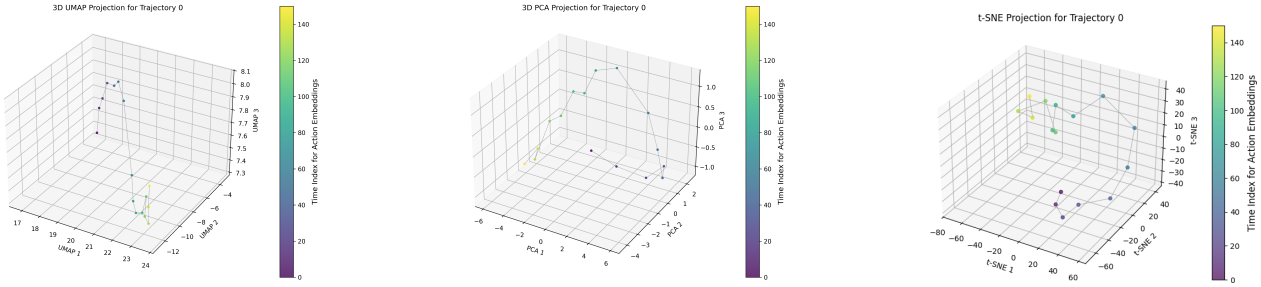
A. Qualitative results

1) Visualizations of UMAP, PCA, and t-SNE reduced embeddings for a successful test trajectory 0: Figure 9 shows the results of applying the techniques to the embeddings from the successful test trajectory 0. It can be seen that all the different reduced embeddings form a path.

2) Visualizations of UMAP, PCA, and t-SNE reduced embeddings for a unsuccessful test trajectory 2: Figure 10 shows the results of applying the techniques to the embeddings from the unsuccessful test trajectory 2. It can be seen that all the different reduced embeddings form a path but this time the last embeddings are grouped in the same region. As this trajectory is unsuccessful we can assume that this is due to the robot getting stuck in similar actions without achieving its goal (closing the drawer).

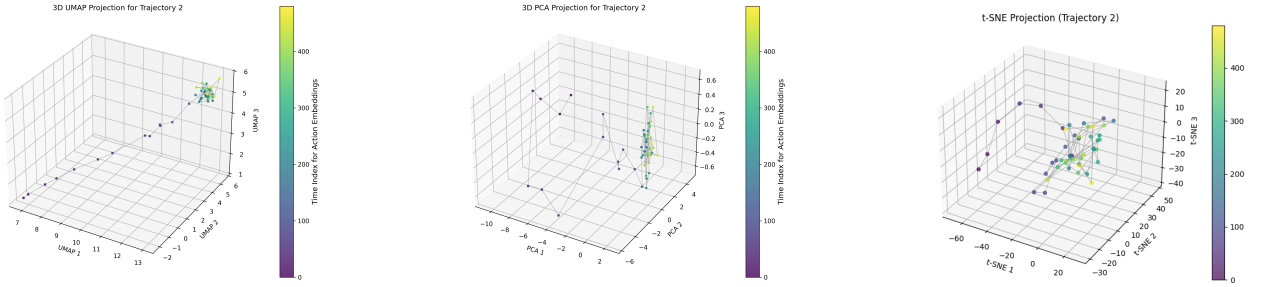
3) Cosine distances for UMAP and t-SNE for train trajectory 0, successful test trajectory 0 and unsuccessful test trajectory 3: Figures 11 and 12 show the results of the cosine distances for the train trajectory 0, successful test trajectory 0 and unsuccessful test trajectory 3. From the figures we can see that both techniques also generate spikes between some embeddings pairs, although the distances provided by UMAP are of lesser magnitude. It is also worth mentioning that UMAP's distances for the unsuccessful test trajectory 3 seem to be minimal, which does not happen for other unsuccessful test trajectories.

4) Comparison of MSE for original and reduced embeddings for each algorithm: In figure 13 we represent the MSE distance between consecutive points in the original embeddings space. On the other hand figure 14 shows the MSE for each reduction technique for the same trajectory. These results show that the original embeddings have much smaller MSE distances between consecutive points compared to the reduced embeddings. With the exception of a much higher MSE value in t-SNE for two consecutive embeddings, we can also see that all three reduced techniques show similar MSE spikes in their embeddings.



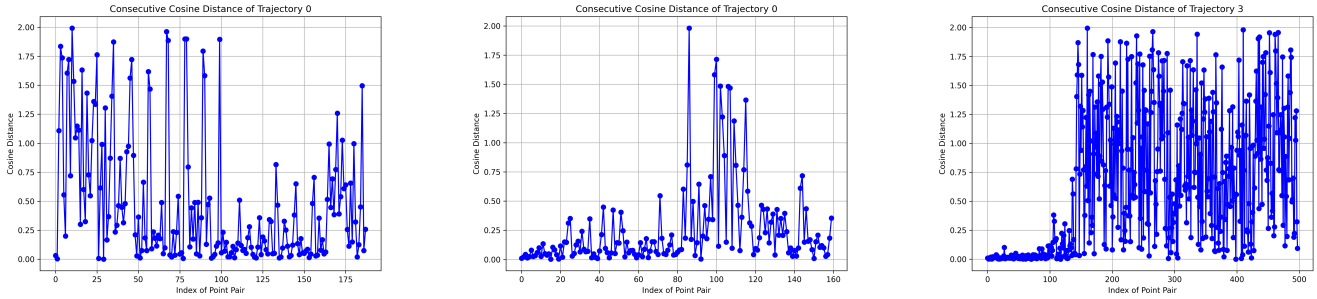
(a) UMAP (b) PCA (c) t-SNE

Fig. 9: Comparison of dimensionality reduction techniques (UMAP, PCA, t-SNE) for successful test trajectory 0.



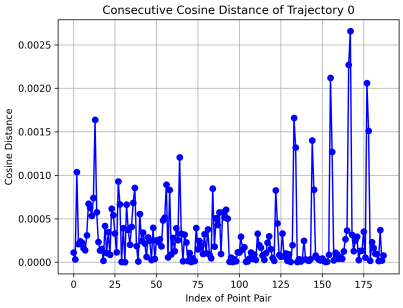
(a) UMAP (b) PCA (c) t-SNE

Fig. 10: Comparison of dimensionality reduction techniques (UMAP, PCA, t-SNE) for unsuccessful test trajectory 2.

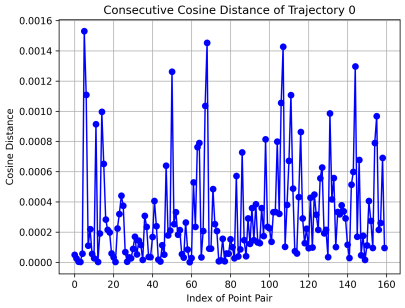


(a) t-SNE for Train trajectory 0 (b) t-SNE for Test trajectory 0 (successful) (c) t-SNE for Test trajectory 3 (unsuccessful)

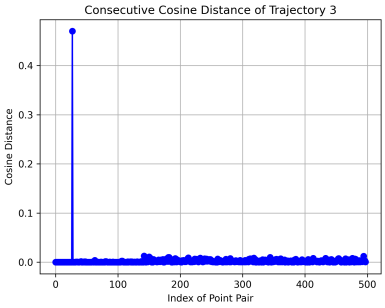
Fig. 11: Visualization of the cosine distances between consecutive points for t-SNE across different trajectories.



(a) UMAP for Train trajectory 0



(b) UMAP for Test trajectory 0 (successful)



(c) UMAP for Test trajectory 3 (unsuccessful)

Fig. 12: Visualization of the cosine distances between consecutive points for UMAP across different trajectories.

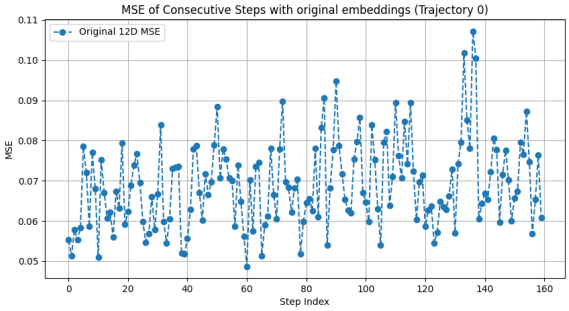
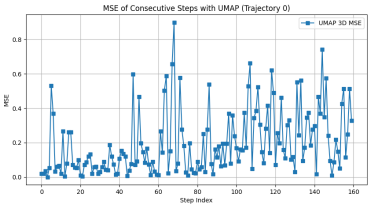
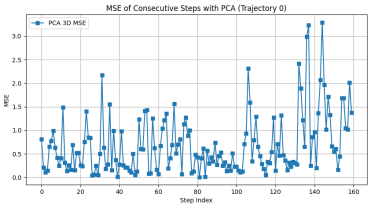


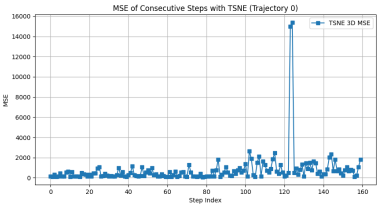
Fig. 13: MSE of original embeddings in successful trajectory 0.



(a) MSE of UMAP-reduced embeddings



(b) MSE of PCA-reduced embeddings



(c) MSE of t-SNE-reduced embeddings

Fig. 14: Visualization of the MSE distances between consecutive points for the reduced embeddings by each technique for the successful test trajectory 0.

B. Quantitative results (train set)

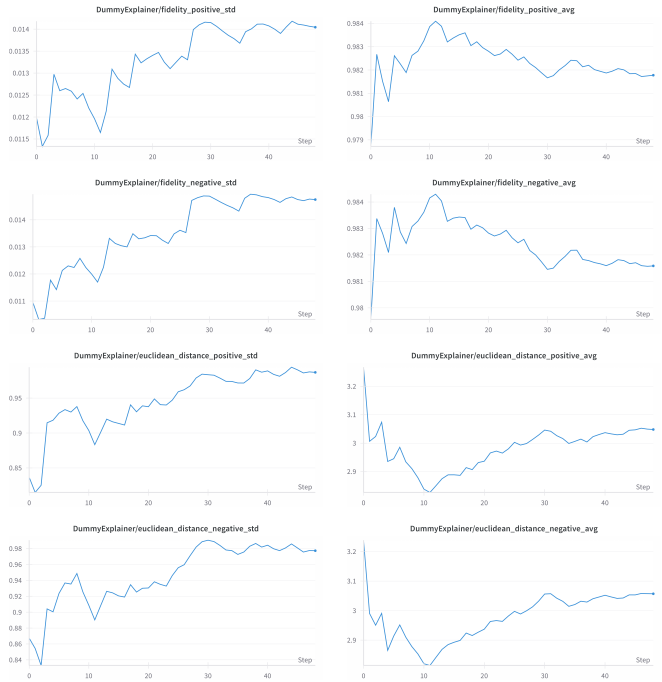


Fig. 15: Quantitative results for train trajectories using DummyExplainer, showing fidelity and Euclidean distance metrics.

Fig. 16: Quantitative results for train trajectories using GNNExplainer_no_node_mask, showing fidelity and Euclidean distance metrics.

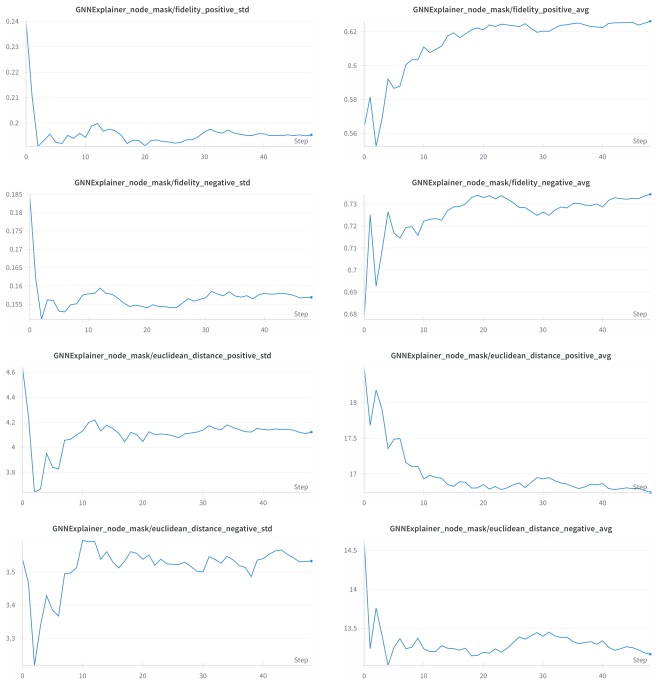


Fig. 17: Quantitative results for train trajectories using GNNExplainer_node_mask, showing fidelity and Euclidean distance metrics.

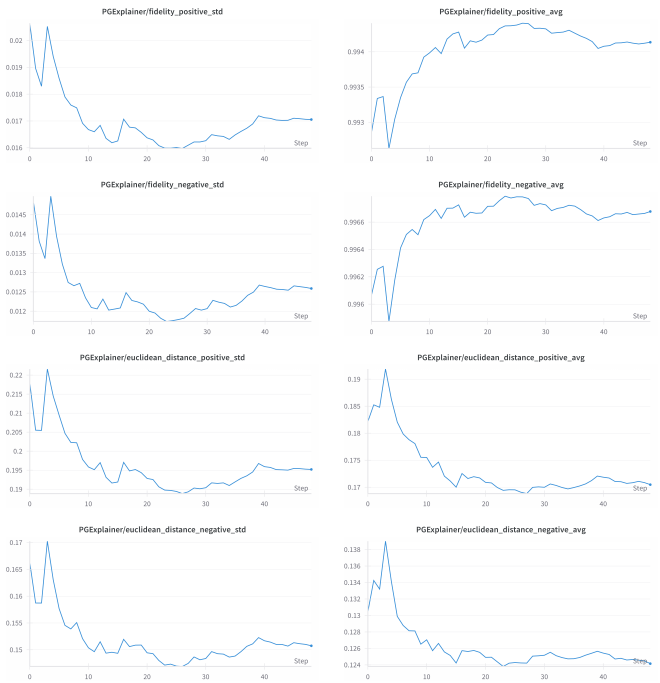


Fig. 18: Quantitative results for train trajectories using PGExplainer, showing fidelity and Euclidean distance metrics.

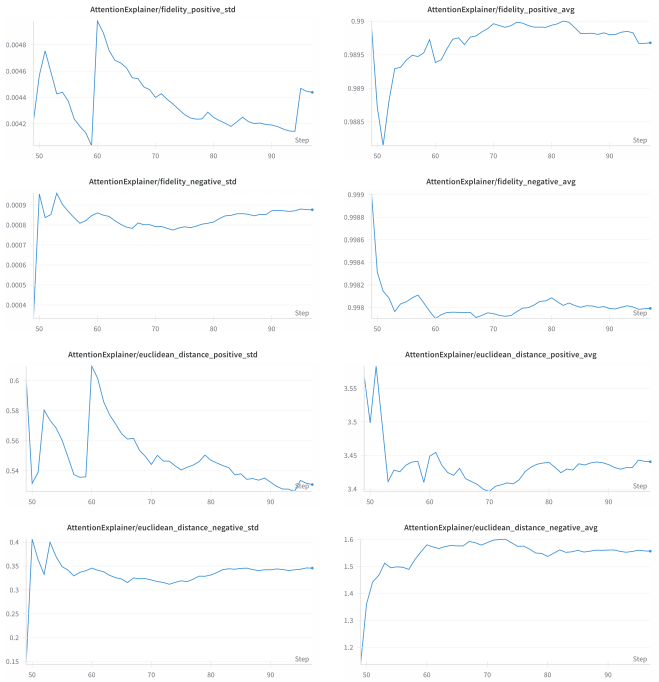


Fig. 19: Quantitative results for train trajectories using AttentionExplainer, showing fidelity and Euclidean distance metrics.

C. Quantitative results (test set)

Fig. 20: Quantitative results for test trajectories using DummyExplainer, showing fidelity and Euclidean distance metrics.

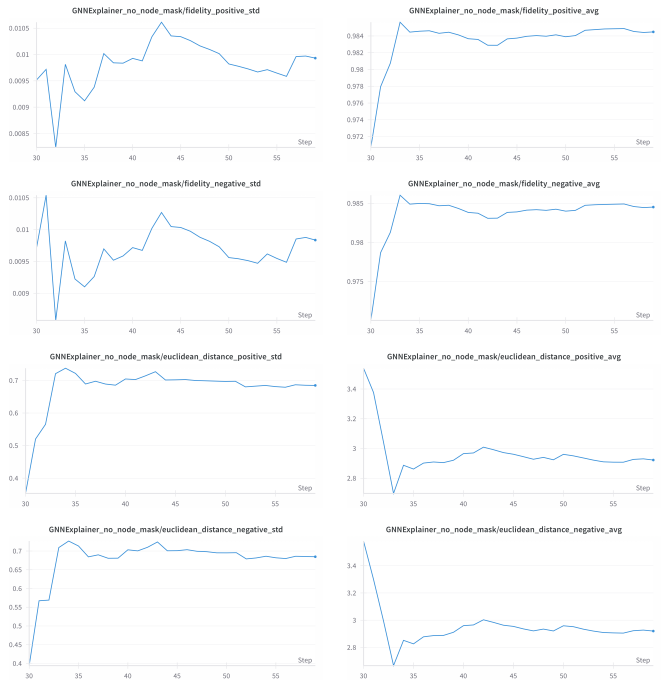


Fig. 21: Quantitative results for test trajectories using GNNExplainer_no_node_mask, showing fidelity and Euclidean distance metrics.

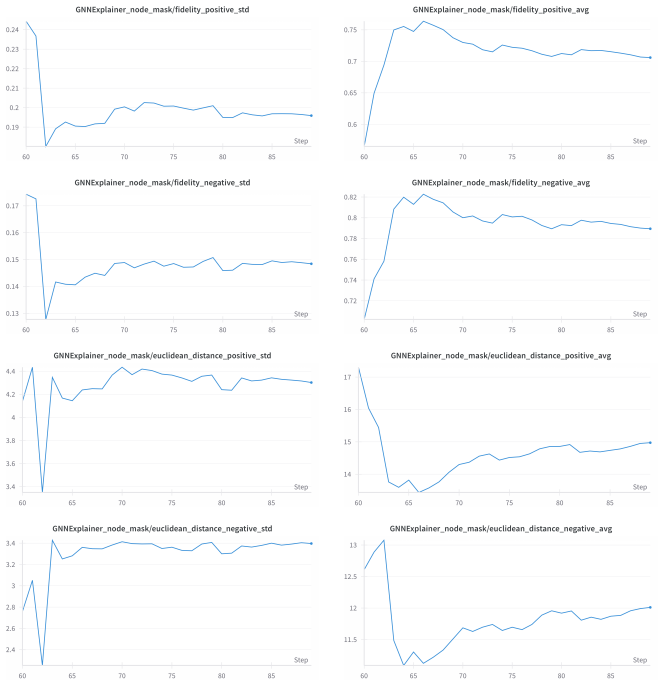


Fig. 22: Quantitative results for test trajectories using GNNExplainer_node_mask, showing fidelity and Euclidean distance metrics.

Fig. 23: Quantitative results for test trajectories using PGExplainer, showing fidelity and Euclidean distance metrics.

Fig. 24: Quantitative results for test trajectories using AttentionExplainer, showing fidelity and Euclidean distance metrics.