Efficient graph representation framework for chemical molecule similarity tasks

Jiaji Ma
Department of Computer Science
University of Virginia
Charlottesville, VA, USA
yjk8jd@virginia.edu

Seung-Hwan Lim
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA
lims1@ornl.gov

***Abstract*—Graph data has emerged in numerous scientific domains and machine learning techniques have been widely used for analysis and learning of diverse data for prediction and decision. Machine learning techniques can readily address complex problems by leveraging their structural information. But graphs cannot be directly used for existing machine learning algorithms unless encoded as vectors. The problem of efficient representation of graphs is a substantial challenge in graph machine learning. In this paper, we propose a novel two-stage framework for the representation of chemical molecule graphs based on the strengths of Graph Isomorphism Networks (GINs) and Siamese autoencoders. In the first stage, the GIN model is constructed and trained using the structural information of chemical molecule graphs. Node attributes, edge attributes, and edge indices are used as input data, while graph attributes are used as labels. The GIN model effectively captures the structural characteristics of graphs and can accurately predict graph attributes, i.e., molecular properties. It also generates Graph Embeddings, represented as vectors that encode the structural information of graphs. In the second stage, Graph Embedding vectors are further optimized for downstream similarity tasks while preserving the graph structural information. The Siamese autoencoder is constructed and trained, which reduces the dimensionality of the Graph Embedding vectors, while maximizing the preservation of structural information in the original high-dimensional vectors. The resulting low-dimensional Graph Embeddings can be effectively utilized for tasks such as approximate nearest neighbor search. The experimental results demonstrate the effectiveness of our proposed framework in accurately predicting graph similarity.**

***Index Terms*—Graph representation learning, Autoencoder, Similarity Learning, Graph Neural Network**

I. Introduction

In recent years, machine learning methodologies have gained remarkable success in various domains, particularly when applied to text and image data. There have been significant advancements in fields such as computer vision and natural language processing, fueled by the power of machine learning techniques [1], [2]. However, as the complexity and diversity of real-world data continue to expand, a growing realization has emerged that graphs, as a versatile and expressive data structure, have tremendous potential for capturing and understanding intricate relationships among entities [3].

Graphs have proven to be highly effective structures for representing and analyzing real-world data in numerous domains, such as social networks and biological networks. The effectiveness of graphs in these domains can be attributed to their ability to encode both structural and semantic information. By representing entities as nodes and relationships as edges, graphs provide a powerful framework for capturing the underlying patterns, dependencies, and inter-dependencies within complex systems. This flexibility and ability to capture and model arbitrary relationships between arbitrary entities allow graphs to go beyond the limitations of traditional data structures, providing a more comprehensive and holistic understanding of complex systems and real-world data.

One of the fundamental problems is retrieving a set of similar graphs from a database given a query [4]. Many different similarity metrics have been proposed, such as the Graph Edit Distance [5] and Maximum Common Subgraph [6] that are based only on the graph structure without any real-world context. Models have been proposed to approximate the true values of these NP-complete problems [4]. But in many fields, it is hard to adapt these methods to new tasks since these metrics do not necessarily have any meaningful interpretations.

Efficient graph representation is particularly valuable in real-world applications. Such as virtual screening chemical compounds for drug discovery, which is an *in silico* method that enables researchers to avoid the time and cost of experimentally testing infinite possibilities of compounds and reduces the number of candidate molecules [7]. The ability to find similar molecules to a given compound allows for the exploration of the chemical space and aids in the identification of potential drug candidates with desirable properties [8]. Similarity-based approaches enable the transfer of knowledge from known molecules to new or unexplored molecules, facilitating the discovery of novel chemical entities [9]. They also enable the exploration of alternative chemical scaffolds that may exhibit improved drug-like properties or reduced side effects. By efficiently representing molecular graph structures

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

as vectors, we can employ various similarity search algorithms to identify similar molecules.

The integration of graph-based methodologies with machine learning techniques for graph representation learning has gained significant attention [10]. There have been methods developed for graph isomorphism testing without mapping functions [11] and thus cannot be applied for general similarity learning. For similarity learning models to be useful for querying in downstream tasks, the graphs have to be encoded into vector representations. Graph embeddings enable the transformation of graph data into vector representations, thereby bridging the gap between graph structures and traditional vector-based machine learning models [12], [13]. Using graph embeddings, a wide array of established machine learning techniques can be employed to tackle diverse tasks, extending beyond the limited applications of graph data in its original form. This enhanced flexibility allows for more comprehensive analysis, prediction, and decision-making in real-world scenarios. With the emergence of deep learning on graph data, Graph Neural Networks (GNNs) have become a powerful tool to encode graphs into embedding vectors [14]–[17]. Compared to traditional graph embedding methods, GNNs address tasks in an end-to-end manner [18] and can better leverage graph features for specific learning tasks. GNN models have been proposed to solve problems in multiple domains, such as brain networks [19] and computer security [20].

In the field of biomedicine, due to the nature of the graph data as chemical molecules, there have been cheminformatics tools for mapping the chemical space long before machine learning, called molecular fingerprinting. Specifically, Morgan fingerprinting [21] is one of the most widely used featurization methods for chemical molecules. The algorithm iteratively encodes circular substructures of a molecule as identifiers, hashes them, and folds them to bit positions to generate a bit string. Since fingerprinting methods are optimized specifically for chemical molecules, fingerprinting has achieved good performance when used as input representations in deep neural networks [22], but in many cases fingerprinting methods are not able to offer ideal performance due to the length of the resulting embedding vectors.

In this paper we propose a two-stage framework to generate efficient vector representations for molecular graphs, which combines the power of the Graph Isomorphism Network and the Siamese autoencoder. We aim to enhance the accuracy and efficiency of similarity search for chemical molecules. Our framework provides a robust representation of molecules as graphs that capture essential structural and chemical information. Using machine learning techniques, we can effectively compare and rank molecules on their similarity, enabling more targeted and efficient drug discovery processes. Our research aims to advance the field of graph representation for similarity search, providing a powerful tool for the exploration and discovery of new compounds with desired properties. Our results demonstrate the effectiveness of the proposed framework in accurately predicting graph similarity while preserving the essential structural information of the graphs in Graph Embeddings. Our findings provide valuable insights into the application of machine learning to graph data, specifically chemical molecule analysis.

The rest of this paper is organized as follows. Section 2 introduces the methodology and technical details of our framework. Section 3 presents the experimental results. Section 4 summarizes this paper and discusses possible research directions.

II. Methodology

In this section, we introduce the overall framework to produce an efficient vector representation for any given chemical molecule, as illustrated in Figure 1. The overall pipeline in our proposed framework consists of two stages. In the first stage, we use the structural information of molecule graphs to train Graph Neural Network models for predicting individual graph attributes. Then we use these trained models to calculate the embeddings of the molecules for each graph attribute. In the second stage, we use the output vectors from the first stage to train Siamese autoencoders for preserving the information in the vectors and the relative similarity between vectors. Then we use the trained autoencoders to obtain the corresponding low-dimensional Graph Embedding vectors from the high-dimensional embedding vectors from the previous stage.

A. Graph Isomorphism Network (GIN)

Each chemical molecule can be represented as an undirected weighted graph denoted as G = (V, E). The set of nodes in the graph, V, corresponds to the atoms present in the molecule. The set of edges, E, represents the bonds between the atoms, capturing the connectivity information of the molecular structure. The graph includes intrinsic properties associated with the molecule itself in nodes, edges, and the overall graph, providing valuable data for analysis. Specifically, each node (atom) in the graph is associated with 11 attributes that characterize various atomic properties. Similarly, each edge (bond) carries 4 attributes that describe bond-specific properties. At the graph level, 19 attributes are provided that represent different molecular properties as shown in Table I. These descriptors encompass a diverse set of features, and provide relevant information that captures global aspects of the chemical molecules.

By leveraging the properties encoded within the dataset, our GIN exploits the rich information present in the graphs. This comprehensive representation enables the model to effectively capture the intricate relationships and interactions between atoms, bonds, and the overall molecular structure.

We designed our framework for similarity tasks, but currently, for how similar any two given chemical molecule graphs are, there is no canonical definition and no standard way for measuring or calculating the ground truth. So, based on a central premise of medicinal chemistry that structurally similar molecules tend to exhibit similar properties [23], we

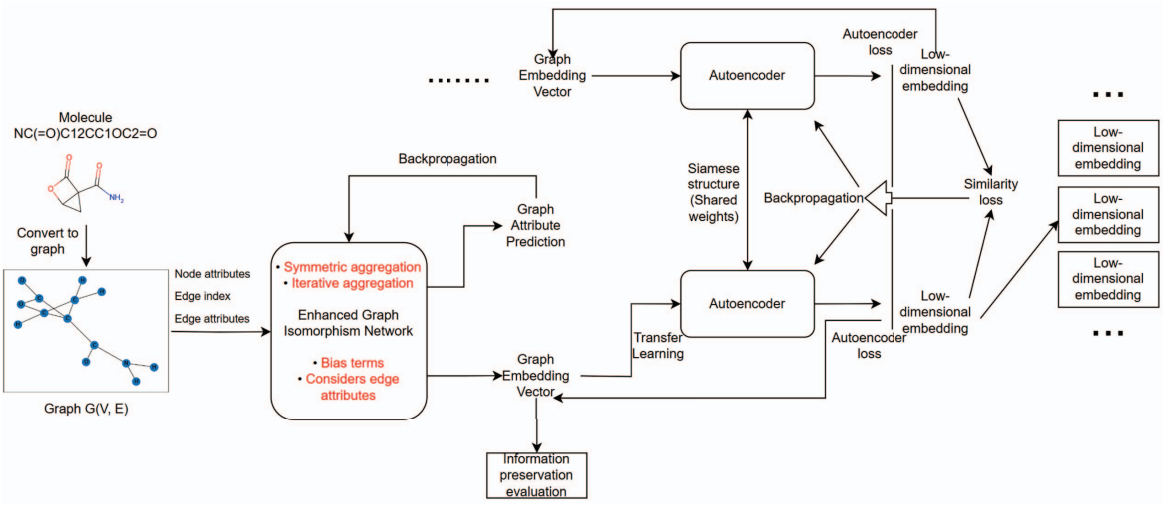

Fig. 1. Sequence diagram of overall 2-stage framework

TABLE I
GRAPH ATTRIBUTES IN QM9 DATASET

Attribute index	Molecular property	Unit
0	Dipole moment μ	D
1	Isotropic polarizability α	${a_0}^3$
2	Highest occupied molecular orbital energy ϵ_{HOMO}	eV
3	Lowest unoccupied molecular orbital energy ϵ_{LUMO}	eV
4	Gap between ϵ_{HOMO} and ϵ_{LUMO}	eV
5	Electronic spatial extent $\langle R^2 \rangle$	
6	Zero point vibrational energy ZPVE	eV
7	Internal energy at 0K U_0	eV
8	Internal energy at 298.15K U	eV
9	Enthalpy at 298.15K H	eV
10	Free energy at 298.15K G	eV
11	Heat capavity at 298.15K c_{v}	$\frac{\text{cal}}{\text{mol K}}$
12	Atomization energy at 0K U_0^{ATOM}	eV
13	Atomization energy at 298.15K U^{ATOM}	eV
14	Atomization enthalpy at 298.15K H^{ATOM}	eV
15	Atomization free energy at 298.15K G^{ATOM}	eV
16	Rotational constant A	GHz
17	Rotational constant B	GHz
18	Rotational constant C	GHz

use the graph attributes, which represent molecular properties, as the labels for each graph.

Our GIN model has the following stages: (1) Node embedding, which encodes the features and structural properties of each node into a vector; (2) Graph embedding, during which graph level representations are obtained from each set of node embeddings using global pooling, then graph level embeddings from each level of abstraction is concatenated to form a single high-dimensional embedding vector; and (3) the graph attribute computation stage, which reduces the high-dimensional vectors into one final attribute value, which is compared with the ground-truth value to update model parameters.

The model formulated as a sequence of graph convolution operations can be represented as the following expression.

$$h^{l+1} = \sigma(Ah^l W^l + b^l) \tag{1}$$

Where h^l represents the node embeddings in layer l, A denotes the graph adjacency matrix, W^l is the weight matrix for layer l, and b^l represents the bias vector at layer l. The activation function σ is applied element-wise, introducing non-linearity to the model.

1) Stage 1: Node embedding: We choose GINE [24] as the convolutional layer, which is based on the Graph Isomorphism Network aggregation method and has the following aggregation function:

$$\mathbf{x}_i' = h_\Theta \left((1+\epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \text{ReLU}(\mathbf{x}_j + \mathbf{e}_{j,i}) \right) \tag{2}$$

Where $\mathbf{x}_i$ is the input representation of node i, $\mathbf{x}_i'$ is the updated representation of node i, $\mathcal{N}(i)$ is the set of neighboring nodes of node i, $\mathbf{e}_{j,i}$ is the edge feature vector between nodes i and j, ReLU is the rectified linear unit activation function, ϵ is a learnable parameter that controls the scaling of the input representation, and h_Θ is a non-linear activation function applied to the aggregated representation.

At the core of obtaining node embeddings is the aggregation function, which computes the updated representation of each node by combining its own input representation with the representations of its neighboring nodes and edges [25]. Notably, the edge level attributes are incorporated into the update process, the input representation of each neighboring node j is first transformed by adding the corresponding edge feature vector $\mathbf{e}_{j,i}$ and applying the rectified linear unit activation function. The resulting representations are then summed and scaled by a learnable parameter ϵ, before being added to the scaled input representation of the node i. The aggregated representation is then passed through a non-linear activation function h_Θ to obtain the final updated representation. This function is learned during training and is representation-invariant, which improves its ability to generalize to unseen data.

Our model consists of five GINE convolutional layers to update node embeddings, with each layer capturing the essential structural and semantic information at a different level of abstraction through a nonlinear activation function. The forward pass involves successive updates of node embeddings through each convolutional layer, this results in five sets of node embeddings, corresponding to different levels of neighborhood information, ranging from local neighborhood to global graph characteristics. The convolutional layers allow the model to aggregate and propagate information from neighboring nodes, allowing it to discern local structural patterns and capture local context.

To further enhance the expressiveness and discriminative capabilities of our model, we include a bias term in the forward pass. The bias term is a learnable parameter that is added to each layer's aggregated representations of neighboring nodes and edges after passing them through a neural network. The neural network then applies a nonlinear activation function to the sum of the aggregated representations and the bias term. This allows the model to learn a shift in the activation function, which can be useful for capturing complex nonlinearities in the data. By customizing the influence of each layer, we can improve the model's ability to capture the local structure of the graph and learn more expressive node embeddings, since the model can selectively emphasize or attenuate specific graph features during the aggregation process. This adaptability allows the model to focus on the most relevant graph properties, leading to improved predictive performance and better generalization.

2) Graph Representation: To obtain the graph-level representation, we perform global add pooling operations. They compute the sum of the node embeddings across the set of nodes, to obtain a single vector representation that captures the aggregate features of the graph for each level of abstraction. This operation is applied independently to each feature dimension, resulting in sets of feature vectors that capture different aspects of the graph. These pooled representations provide a concise summary of the graph's structural and semantic properties, enabling higher-level analysis and inference.

The global add pooling mechanism is a powerful tool for capturing the aggregate features of the graph, which helps effectively capture the overall characteristics and patterns exhibited by the graph as a whole, facilitating the capture of more holistic and higher-level graph properties. Notably, it is permutation invariant, meaning that the order of the nodes does not affect the resulting pooled representation. This property makes the pooling operation robust to variations in graph structure and node ordering, leading to more consistent and reliable graph-level representations. And it offers flexibility when graphs in the dataset have varying sizes since the representation it produces has a fixed size.

To combine and integrate graph embeddings from different levels, the resulting set of feature vectors is concatenated into a single high-dimensional vector with a length of 64, which captures the structure of the graph and encapsulates a rich and expressive representation of the global context of the entire graph, consolidating information from various levels of abstraction.

3) Graph attribute computation: The final stage of our model involves computing the graph attributes based on the concatenated graph embedding. To achieve this, we employ a series of fully connected layers, each consisting of a linear layer and with rectified linear unit (ReLU) activations. These layers enable the model to learn complex mappings from the high-dimensional concatenated embedding space to the desired attribute space. We use three linear layers to gradually reduce the dimensionality of the concatenated graph embedding while capturing relevant features. The activation function after each linear layer introduces non-linearity into the attribute computation process and allows the model to learn complex relationships between the graph embeddings and the target attributes. Finally, we employ a fourth linear layer to map the reduced-dimensional embedding to a single attribute value. The output of this layer represents the predicted value for the target attribute. By comparing this predicted value with the ground truth value, we can calculate the loss and update the model parameters during the training process. During training, we employ the Adam optimizer with a learning rate of 0.01 and weight decay of 0.01. The loss function used is the L1 loss, which measures the absolute difference between the predicted and ground truth attribute values. The model is trained for 100 epochs, and in each epoch, the training loss and error rate, as well as the validation loss and error rate are computed and monitored. The model with the lowest validation loss is selected as the final model. After training, we evaluate the performance of the trained model using the test dataset. We compute the L1 loss and error rate on the test data to assess the model's predictive accuracy and generalization ability.

B. Siamese Autoencoder

After obtaining the complete, high-dimensional Graph Embedding vectors from our GIN model, optimizing them to be more suitable to use for downstream tasks is an important issue. Since superficially high-dimensional and complex phenomena can actually be dominated by a small amount of simple variables in most situations [26], this can be done using some learned projection method that maps the vectors in high-dimensional feature space to low-dimensional feature space. We choose Siamese autoencoders to be our dimensionality reduction technique, to maximize preservation of information in the original high-dimensional embedding vectors while reducing the dimensionality of the vectors, which optimizes them for tasks requiring vector input.

1) Autoencoders: Autoencoders serve as the foundational component of the Siamese autoencoder stage of our framework. They are neural network architectures designed for unsupervised representation learning. An autoencoder consists of three layers, separated by an encoder and a decoder, which work together to learn compressed and meaningful representations of the input data, through the minimization of reconstruction error by adjusting model parameters.

The encoder takes an input sample and maps it to a lower-dimensional latent space representation with length 10, this is also known as a bottleneck layer or code, and is followed by a Rectified Linear Unit (ReLU) activation function, which introduces non-linearity and captures complex patterns in the data. The latent representation aims to capture the most salient features and patterns in the data. The decoder, on the other hand, reconstructs the original input sample from the latent representation, aiming to minimize the reconstruction error.

In the process of dimensionality reduction, discarding and simplifying some dimensions inevitably leads to loss of information, so the autoencoder is trained to keep the most important characteristics as much as possible. During training, the autoencoder learns to minimize the discrepancy between the reconstructed output and the original input by optimizing a L1 loss function, which measures the difference between the input and its reconstruction. By iteratively adjusting the weights and biases of the encoder and decoder, the autoencoder gradually improves its ability to compress and reconstruct the input data.

2) Siamese structure: Siamese networks consist of two identical neural networks with shared weights that take separate inputs. The neural networks are eventually merged into a single layer by applying a discriminative function to the outputs of individual neural networks [27]. This makes them ideal for similarity-related tasks, as they are designed to compare two inputs.

We design a Siamese autoencoder architecture, as an extension of the traditional autoencoder, specifically designed for similarity tasks, as it includes a pair of identical autoencoders sharing weights. Each autoencoder consists of an encoder and a decoder, which utilize fully connected layers to process the input vectors. During the forward pass, two input vectors are fed into autoencoders separately, resulting in two sets of encoded representations. These encoded representations capture the essential molecular features in a lower-dimensional space. The decoder then reconstructs the original molecules from the encoded representations, producing two sets of decoded molecules.

By utilizing this Siamese architecture, we leverage the shared weights to learn representations that effectively capture the similarity relationships between the input samples. During training, the Siamese autoencoder optimizes two main objectives simultaneously: accurate reconstruction and preservation of similarity. To achieve this, the Siamese autoencoder combines the autoencoder loss, which quantifies the discrepancy between the input samples and their reconstructions, with a similarity preservation loss. The similarity preservation loss compares the pairwise distances or similarities between the original input samples with those derived from the decoder outputs. By minimizing the difference between the original distances and the distances computed from the decoder outputs, the Siamese autoencoder ensures that the latent space representations maintain not only the high-dimensional vector information, but also the inherent pairwise relationships.

In our training process, we use the Adam optimizer with a learning rate of 0.01 and a weight decay of 0.1. The L1 loss function is applied for both the autoencoder and the Siamese loss components. The model is trained over 100 epochs using mini batches of data to update the parameters and optimize its performance. The Siamese autoencoder's architecture leverages the power of unsupervised representation learning while incorporating similarity preservation, which enables more effective and accurate handling of similarity-related tasks.

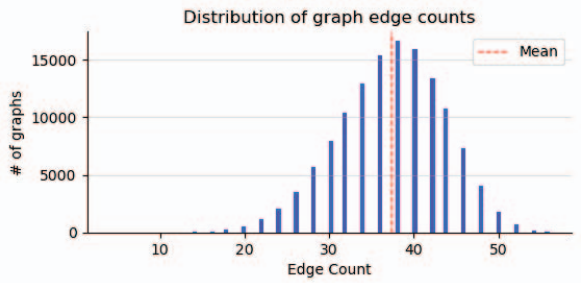

Fig. 2. Distribution of the number of edges in each graph in the entire QM9 dataset

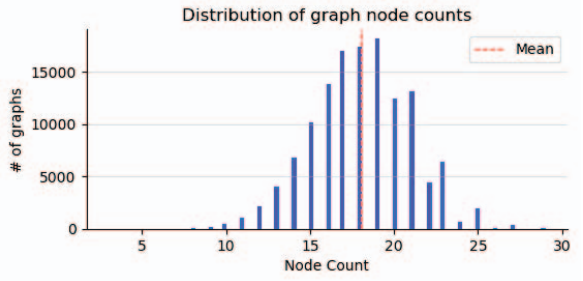

Fig. 3. Distribution of the number of nodes in each graph in the entire QM9 dataset

III. Results

A. Dataset

The chemical molecule dataset QM9 [28] is used for the experiments. It is a subset collection from the GDB-17 database [29] and consists of 130,831 stable organic molecules. All the molecules are made up of H, C, N, O, F elements, have up to nine non-hydrogen heavy atoms and are modeled using density functional theory. It has been used in several existing works on graph machine learning [30]–[32]. We randomly select 1000 graphs from the dataset and randomly split 60%, 20% and 20% as the training set, validation set, and testing set.

As illustrated in Figures 2, 3, and 4, the graph sizes are close to normally distributed, while most graph attributes have distributions without much discernible pattern, and for some attributes, there are some obvious outliers that deviate a lot from other values.

B. Baseline method

Our baseline approach is molecular fingerprinting, in which a kernel is applied that extracts features from the molecule. The features are hashed and then used to calculate a bit vector. Specifically, we chose one of the most widely used methods, the Morgan fingerprint [21], also known as the extended-connectivity fingerprint ECFP4, which is optimized

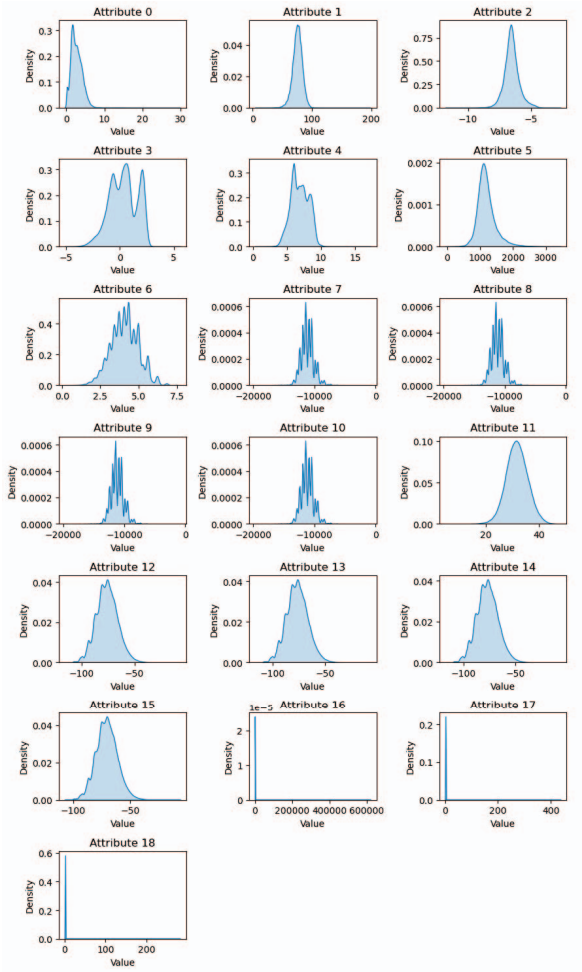

Fig. 4. Distribution of the value of each graph attribute across all graphs in the QM9 dataset

to compare the similarity between molecules. It takes into account the neighborhood of each atom and perceive the presence of specific circular substructures around each atom in a molecule, which are predictive of the biological activities. It is one of the best performing molecular fingerprinting methods for target prediction tasks. By default, it produces vectors of length 2048.

C. Evaluation metrics

For both the baseline method and after each stage of our framework, we run three different benchmarks to evaluate different aspects of the performance of our models. First, we use the Graph Embedding vectors as input to a simple Forward Feeding Neural Network that is trained to predict the original graph attribute; this measures how well the structural information (which was used to obtain the embeddings) is preserved. Second, we use Uniform Manifold Approximation and Projection (UMAP) [33] to project the Graph Embedding vectors onto a 2-dimensional space. This allows us to examine how well the distribution and vector distances reflect the ground truth value for attribute similarities between the graphs.

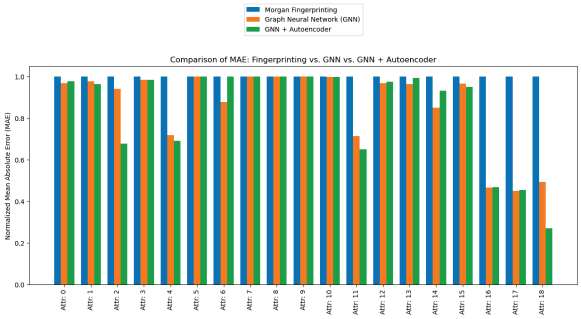

Fig. 5. Attribute prediction MAE comparison between embeddings obtained after each stage against Morgan Fingerprinting embeddings

And finally, we use the Graph Embedding Vectors as input to perform Approximate Nearest Neighbor Search using Faiss [34]. This measures the downstream task performance from multiple aspects for each set of Graph Embedding vectors.

D. Experiment results

Our framework was implemented using PyTorch Geometric (PyG) library for all model and tensor related computations in Python.

Since all the Graph Embedding vectors were obtained using the graph structural information as input, we are able to measure how well the they retained the ability of the original graph to predict graph attributes. For each attribute, 1000 previously unseen graphs are randomly selected and split 60% training set, 20% validation set and 20% testing set; these graphs are used to train the Feed Forward Neural Network (FNN). The FNN has 4 fully connected layers, each separated by a non-linear activation function and a dropout layer. Three FNNs were trained separately using fingerprinting vectors, vectors after our first stage, and vectors after the second stage, respectively. Another 1000 unseen graphs were randomly selected and run on each trained FNN. Then the Mean Absolute Error (MAE) values between the ground truth graph attribute values and the FNN prediction values were calculated for each of the three sets of vector inputs. Figure 5 shows the comparison of MAE (after normalizing by setting the values of the fingerprinting method as reference points) between the three embedding methods. For all graph attributes, the Graph Embedding vectors after both stages of the framework consistently provide at least similar or better performance as the fingerprinting embedding vectors despite having far fewer dimensions. Notably, for attributes 2, 4, 11, 16, 17 and 18, our embedding methods offer far superior performance.

The preservation of similarity is illustrated in Figure 6. Using UMAP, Graph Embedding vectors obtained using each method are mapped onto a 2-dimensional space to make their relative position and distance more intuitive and comprehensible. 500 embeddings were randomly selected, in which a random embedding is designated as the query entry. The ground truth for the original graph attribute similarities between the query entry and all other embeddings is calculated and mapped to a continuous color scale. The scatter plots of these randomly selected embeddings demonstrate that after

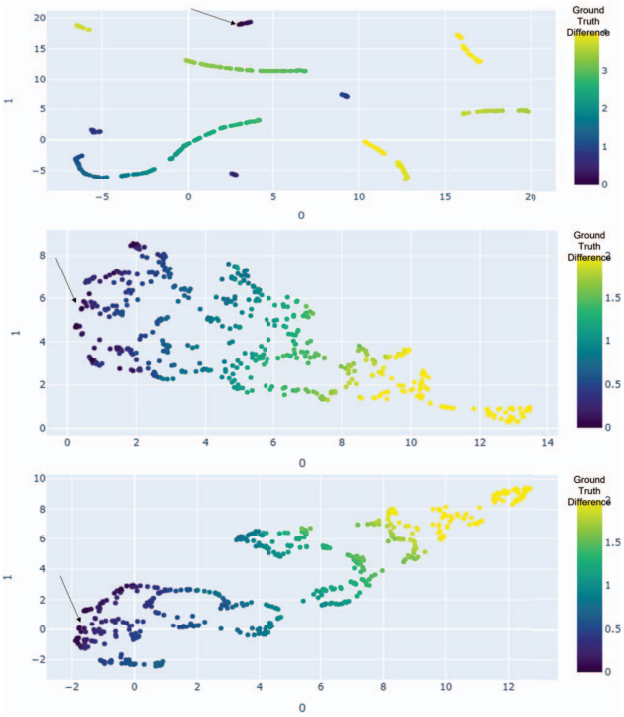

Fig. 6. Embedding vectors mapped to 2-d using UMAP distance scatterplot (colored by ground truth differences)

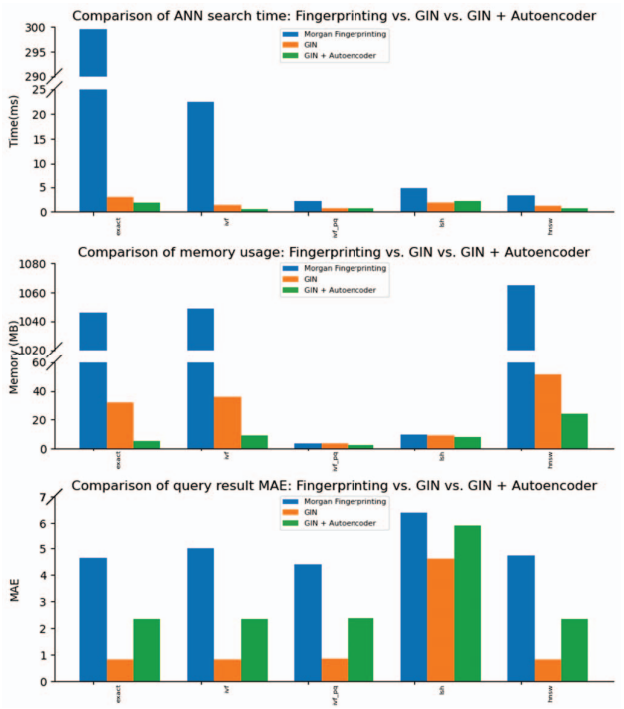

Fig. 7. Time, memory, and MAE performance in Faiss Approximate Nearest Neighbor search using different optimizations

each stage of our framework, the original graph similarity is still mostly preserved, as graphs with higher values of ground truth difference are mapped as points further away, generally, and vice versa. And embedding vectors obtained using our models achieve much better similarity preservation compared to fingerprinting vectors. These scatterplots can be done individually for each graph attribute. Figure 6 is an example using attribute 0, other attributes yield similar results.

Figure 4 shows the downstream task performance comparison of the three different embedding methods, including time, indexing memory, and MAE between the query results and ground truth. It illustrates the differences across different Approximate Nearest Neighbor (ANN) search optimization methods, including Exact Search, Inverted File Indexing (IVF), IVF + Product Quantization (PQ), Locality Sensitive Hashing (LSH), and Hierarchical Navigable Small World (HNSW). Here wer use attribute 0 as an example in Figure 7, similar differences can be observed for other graph attributes as well. The low-dimensional embedding vectors obtained from our models achieve better time and memory performance than molecular fingerprinting methods. This was to be expected due to the differences in dimensionality of the vectors. It is worth noting that our embedding methods also achieve superior MAE performance in terms of original graph attributes, meaning that the query results have higher quality (lower distance from query item) since the vectors preserved the graph attribute similarities better. Memory usage differences are especially great when compression techniques are not used. For example, when using Hierarchical Navigable Small World (HNSW), compared to fingerprinting vectors, the vectors after stage 1 of our model only require 4.82% of the memory, and the vectors after stage 2 of our model only requires 2.28% of the memory. Furthermore, our embedding vectors were able to greatly reduce query time. Specifically, embeddings obtained from the autoencoders only require 0.64%, 2.60%, 29.88%, 46.00% and 20.83% of the time when using fingerprinting vectors, under different optimization techniques. We observe that the relative query result quality advantages of our embeddings are the least obvious when utilizing Locality Sensitive Hashing (LSH). Specifically, they were only able to reduce the MAE by 27.9% and 7.6%, while under Inverted File Index (IVF), our embedding methods reduce the MAE by 83.7% and 53.1%, in addition to reducing the search time by 94.2% and 97.4%. Figure 7 illustrates the effectiveness of Faiss Approximate Nearest Neighbor Search. Across all three embedding methods, better query time and indexing memory usage can be achieved with little to almost no MAE loss. It is worth noting that GNN vectors achieve universally better performance than fingerprinting vectors, but further reducing the vector dimensionality through the autoencoders involves a trade-off, especially between memory and MAE.

IV. Discussion and Conclusion

This study contributes to existing knowledge about graph representation and provides valuable insights into the complex nature of chemical molecule data and proposes an efficient approach to obtain Graph Embeddings in vector format optimized for chemical molecule similarity tasks. Through its incorporation of graph convolutional layers, global pooling, and edge features, as well as the introduction of biases

for enhanced discriminative power, our GIN model offers a more comprehensive and expressive representation of graph-structured data, capturing intricate relationships and attributes with greater accuracy and efficiency. And the Siamese autoencoder achieves dimensionality reduction while preserving essential information and learning similarity between pairs of graphs. Compared to traditional methods in the chemistry field such as molecular fingerprinting, our approach is able to greatly reduce the dimensionality of the embeddings, making them more manageable and computationally efficient. This dimensionality reduction also helps to improve the interpretability and scalability of the embeddings for downstream tasks. The results highlight the effectiveness of the combination of comprehensive modeling capabilities of our GIN model and the dimensionality reduction capabilities of the Siamese autoencoders, which allows our framework to effectively produce efficient graph representations for similarity tasks.

The current study focuses on static molecule structure graphs. Future work can benefit from spatial-temporal chemical molecule data that better reflect how molecules transform over time. For more realistic molecule behavior, molecule isomerization can be modeled using dynamic graphs in the form of time series data. Moreover, this framework is limited to individual graph attributes, multi-tasking models can be explored in the future. Our proposed framework and techniques can also be explored in various other domains that require analysis and prediction of complex graph-structured data.

REFERENCES

[1] S. Xu *et al.*, "Computer vision techniques in construction: a critical review," *Archives of Computational Methods in Engineering*, vol. 28, pp. 3383–3397, 2021.

[2] D. H. Maulud, S. R. Zeebaree, K. Jacksi, M. A. M. Sadeeq, and K. H. Sharif, "State of art for semantic analysis of natural language processing," *Qubahan academic journal*, vol. 1, no. 2, pp. 21–28, 2021.

[3] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[4] Y. Bai *et al.*, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 384–392.

[5] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," *Pattern Recognition Letters*, vol. 1, no. 4, pp. 245–253, 1983.

[6] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern recognition letters*, vol. 19, no. 3-4, pp. 255–259, 1998.

[7] A. Cereto-Massagué *et al.*, "Molecular fingerprint similarity search in virtual screening," *Methods*, vol. 71, pp. 58–63, 2015.

[8] M. A. Skinnider, C. A. Dejong, B. C. Franczak, P. D. McNicholas, and N. A. Magarvey, "Comparative analysis of chemical similarity methods for modular natural products with a hypothetical structure enumeration algorithm," *Journal of Cheminformatics*, vol. 9, pp. 1–15, 2017.

[9] Y. Yang, M. Liu, and J. R. Kitchin, "Neural network embeddings based similarity search method for atomistic systems," *Digital Discovery*, vol. 1, no. 5, pp. 636–644, 2022.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[12] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[13] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE transactions on knowledge and data engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[14] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[15] P. Velickovic *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[16] K. Xu *et al.*, "Representation learning on graphs with jumping knowledge networks," in *International conference on machine learning* PMLR, 2018, pp. 5453–5462.

[17] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[18] G. Ma, N. K. Ahmed, T. L. Willke, and P. S. Yu, "Deep graph similarity learning: A survey," *Data Mining and Knowledge Discovery*, vol. 35, pp. 688–725, 2021.

[19] G. Ma *et al.*, "Deep graph similarity learning for brain data analysis," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2743–2751.

[20] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *International conference on machine learning*. PMLR, 2019, pp. 3835–3845.

[21] H. L. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service." *Journal of chemical documentation*, vol. 5, no. 2, pp. 107–113, 1965.

[22] T. Unterthiner *et al.*, "Deep learning as an opportunity in virtual screening," in *Proceedings of the deep learning workshop at NIPS* vol. 27, 2014, pp. 1–9.

[23] I. Muegge and P. Mukherjee, "An overview of molecular fingerprint similarity search in virtual screening," *Expert opinion on drug discovery* vol. 11, no. 2, pp. 137–148, 2016.

[24] W. Hu *et al.*, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations (ICLR)*, 2020.

[25] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[26] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[27] L. V. Utkin, V. S. Zaborovsky, A. A. Lukashin, S. G. Popov, and A. V. Podolskaja, "A siamese autoencoder preserving distances for anomaly detection in multi-robot systems," in *2017 international conference on control, artificial intelligence, robotics & optimization (ICCAIRO)* IEEE, 2017, pp. 39–44.

[28] Z. Wu *et al.*, "Moleculenet: a benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[29] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *Journal of chemical information and modeling*, vol. 52, no. 11, pp. 2864–2875, 2012.

[30] G. A. Pinheiro *et al.*, "Machine learning prediction of nine molecular properties based on the smiles representation of the qm9 quantum-chemistry dataset," *The Journal of Physical Chemistry A*, vol. 124, no. 47, pp. 9854–9866, 2020.

[31] O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Exploring chemical compound space with quantum-based machine learning," *Nature Reviews Chemistry*, vol. 4, no. 7, pp. 347–358, 2020.

[32] S. Liu, M. F. Demirel, and Y. Liang, "N-gram graph: Simple unsupervised representation for graphs, with applications to molecules," *Advances in neural information processing systems*, vol. 32, 2019.

[33] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *stat*, vol. 1050, p. 18, 2020.

[34] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.