Survey on Graph Autoencoders

Orestis Konstantaropoulos
Electical and Computer Engineering
National Technical University of Athens
orestiskonsta@gmail.com

***Abstract*—Graph-structured data is ubiquitous in natural and social sciences, and as deep learning excels at uncovering patterns in Euclidean data, the surge in applications involving graph-based data calls for integrating relational inductive biases into deep learning architectures. This survey delves into graph neural networks utilizing autoencoder architectures to tackle the challenges posed by complex graph data. The study categorizes existing methods into two main classifications based on their use for pure representation learning or graph generation. Within the representation learning category, four subcategories are identified based on model architectures and training strategies. Graph generative learning models are further classified based on their generation granularity level, distinguishing between simultaneous graph generation, substructure-based generation, and node-by-node graph generation. The study provides a comprehensive taxonomy and overview, systematically summarizing the distinctions and connections among diverse methods, shedding light on the evolving landscape of graph autoencoders.**

I. Introduction

Graph-structured data is ubiquitous throughout various domains in the natural and social sciences, from telecommunication networks to quantum chemistry. Although deep learning has demonstrated efficacy in uncovering hidden patterns in Euclidean data, a growing number of applications involve data represented in graph form. It is imperative to incorporate relational inductive biases into deep learning architectures to enable systems capable of learning, reasoning, and generalizing from such data.

In recent years, there has been a notable upswing in research focused on graph representation learning. This encompasses advancements in deep graph embeddings, extensions of convolutional neural networks tailored for graph-structured data, and approaches rooted in neural message-passing inspired by belief propagation. These developments in graph representation learning have yielded state-of-the-art outcomes across diverse domains, including recommender systems, question answering, and social network analysis. In chemistry, molecules are modeled as graphs, and their bioactivity needs to be identified for drug discovery. Similarly, in a citation network, papers are interconnected through citationships, necessitating the categorization of papers into distinct groups.

The intricate nature of graph data presents considerable challenges to current machine learning algorithms. Graphs exhibit irregularities, wherein a graph may feature a variable size of unordered nodes, and these nodes may possess different numbers of neighbors. Consequently, operations like convolutions, which are straightforward in the image domain, become challenging when applied to the graph domain. Moreover, a fundamental assumption in existing machine learning algorithms—that instances are independent of each other—becomes obsolete when dealing with graph data. In the graph context, each instance (node) is interconnected with others through diverse types of links, including citations, friendships, and interactions.

In this survey, we present an extensive examination of graph neural networks designed to address the challenges outlined above by utilizing autoencoder architectures. These approaches aim to acquire effective representations of graph structures through training on a set of graphs. Subsequently, these learned representations can be employed for downstream prediction tasks or the generation of novel graphs. Graph autoencoder architectures have gained significant efficacy and recognition, leading to a wealth of related literature comprising various papers and methodologies. Although there is considerable diversity in the adopted architectures and training strategies to the best of our knowledge, limited effort has been made to systematically summarize the distinctions and interconnections among these diverse methods.

In this study, we categorize existing methods into two primary classifications based on whether they are employed for pure representation learning and tasks such as link prediction or node classification, or for the generation of new graphs with specific attributes. Additionally, within the first category, we further divide the methods into four subcategories, considering their model architectures and training strategies. We commence by presenting historically initial approaches based on multilayer perceptrons, which served as a foundation for subsequent works and then delve into models grounded in spatial and spectral convolution, recursive structures, adversarial techniques, and self-supervised methods. Concerning graph generative learning models,

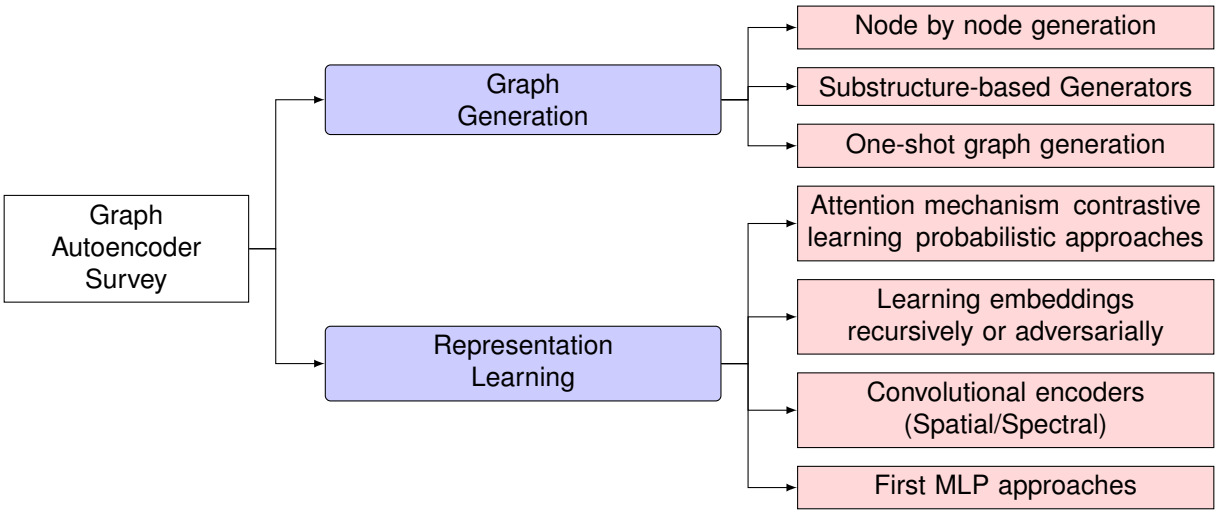

Fig. 1: Literature Survey Tree

we further classify them based on their generation granularity level. Some methodologies opt to generate entire graphs simultaneously, while others utilize substructures as building blocks or even generate graphs node-by-node. The exact taxonomy is shown in Fig.1.

II. Background, definitions

A. Autoencoders

1) Autoencoders for unsupervised learning: Autoencoders stand out as a prominent unsupervised learning technique wherein neural networks play a pivotal role in representation learning. The fundamental principle involves employing a neural architecture during the encoding phase, creating a bottleneck that compels the generation of a compressed knowledge representation of the initial input. In instances where inherent data structure exists, the autoencoder learns and subsequently utilizes this structure when directing the input through the network's bottleneck. In the decoding phase the initial input is reconstructed from the encoded in the latent space information. This dimensionality reduction can place semantically related examples near each other and highly improve performance on tasks such as classification or clustering.

2) Variational autoencoders: Variational Autoencoders are probabilistic generative models with similar architecture with AEs. In this setting the encoded latent variables are now stochastic and follow a posterior distribution in which an encoder model maps the input. The decoder now samples from this distribution conditioning on the latent variable Z and produces new realistic instances. In Fig. 2 a typical architecture for graph VAEs is shown. In more formal and mathematical detail, to build a VAE for graphs we must specify the following key components:

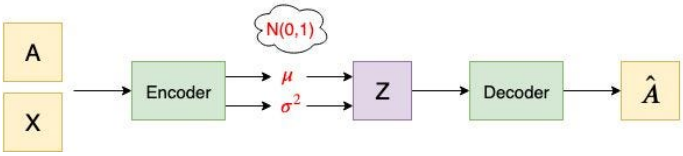

Fig. 2: Typical GVAE Architecture

1) A probabilistic encoder model q_ϕ. In the case of graphs, the probabilistic encoder model takes a graph G as input. From this input, q_ϕ then defines a distribution $q(Z|G)$ over latent representations. Generally, the latent conditional distribution is specified as $Z \sim N(\mu_\phi(G); \sigma_\phi(G)$, where μ_ϕ and σ_ϕ are neural networks that generate the mean and variance parameters for a normal distribution, from which latent embeddings Z are sampled.

2) A probabilistic decoder model p_θ. The decoder takes a latent representation Z as input and uses this input to specify a conditional distribution over graphs and a prior distribution $p(Z)$ over the latent space which is usually a normal distribution.

B. Definitions on Graphs

A graph is represented as $G = (V, E)$ where V is the set of vertices or nodes and E is the set of edges. Let $v_i \in V$ to denote a node and $e_{ij} = (v_i, v_j) \in E$ to denote an edge from v_i to v_j. The neighborhood of a node v is defined as $N(v) = \{u \in V | (v, u) \in E\}$. The adjacency matrix A is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. A graph may have node attributes X, where $X \in R^{n \times d}$ is a node feature matrix with $x_v \in R_d$ representing the feature vector of a node v. The Laplacian matrix of an undirected graph is defined

Model	Inputs	Architecture	Encoder	Decoder	Objective	Novelty	Year
DNGR [2]	A	SDNE	MLP	MLP	Reconstruct PPI matrix	Introduction of GAEs	2016
SDNE [26]	A	Stacked Autoencoder	MLP	MLP	Preserve node first-order and second-order proximity	Introduction of GAEs	2016
GSAGE [7]	A, X	GCNN	2 Graph CNN layers	-	Proximate nodes have closer embeddings	Leverage of node features and contrastive techniques	2017
DGI [23]	A, X	Generator-Discriminator	GCNN	-	Mutual info of nodes and graph summaries maximization	Applicable to inductive learning setups	2018
MGAE [25]	A, X	Stacked Single-layered Spectral CNN	-	-	Reconstruction of X	Introduction of dynamics by adding noise to features	2017
GALA [17]	A, X	Completely Symmetric Autoencoder	Laplacian smoothing	Laplacian sharpening	Reconstruction of X	Symmetric autoencoder	2019
AGE [3]	A, X	Spectral Convolution Filters	Adaptive smoothing filters	-	Reconstruction of X	Usage of optimal adaptive filters	2020
DRNE [22]	A	LSTM with node sequences as input	LSTM	-	Preserve pair-wise regular equivalence	MLP for embeddings regularization	2018
NetRA [27]	A, X	Adversarially Regularized LSTM	LSTM	-	Minimizes network locality-preserving loss and reconstruction error	Introduction of adversarial training in GAEs	2018
ARGA [16]	A, X	GCNN Encoder, MLP Discriminator	Spectral convolutional function	Sigmoid	Reconstruction of X adversarially	Introduction of adversarial training in GAEs	2018
GATE [18]	A, X	Transformer	Encoder layers with self-attention	Reverse encoder	Reconstruction of X, similarity of adjacent nodes representation	attention mechanism	2019
G2G [1]	A, X	Variational GAE	MLP	-	Order of KL-divergence between node distributions	Representing nodes as Gaussian distributions	2018
GMAE [8]	A, X	Architecture Agnostic	-	-	Cosine error	Self-supervision using MASK token	2022

TABLE I: GAES for Representation Learning

as $L = D - A$, where $D \in R^{n \times n}$ is a diagonal degree matrix with $D(i,i) = \sum_j A(i,j)$. Its eigendecomposition is denoted as $L = Q\Lambda Q^T$, where $\Lambda \in R^{n \times n}$ is a diagonal matrix of eigenvalues sorted in ascending order and $Q \in R^{n \times n}$ are the corresponding eigenvectors. The transition matrix is defined as $P = D^{-1}A$, where $P(i,j)$ represents the probability of a random walk starting from node v_i landing at node v_j. Finally, for deep learning models we use σ as the sigmoid activation function.

Symbol	Definition
$G = (V, E)$	A Graph
V, E	Sets of vertices, nodes
$vi \in V$	a specific node
$e_{ij} = (v_i, v_j) \in E$	A specific edge
$N(v) = \{u \in V \vert (v,u) \in E\}$	neighborhood of a node v
$X \in R^{n \times n}$	the adjacency matrix
$X \in R^{n \times d}$	node feature matrix
$L = D - A$	The Laplacian matrix
σ	the sigmoid activation function

TABLE II: Definitions on Graphs.

III. Graph autoencoders for representation learning

The main characteristics of the graph autoencoders used for learning useful representation are listed in Table I

A. First MLP approaches

Graph autoencoders generally aim to construct graph representations, lower dimensional vectors for each graph vertex that encapsulate the structure and features of the vertex neighborhood. The first approaches to capture the nonlinearity of graph information with deep architectures happened in 2016 by Cao [2] and Wang [26].

Drawing inspiration from word2vec [15], the Deep Neural Network for Graph Representations (DNGR) [2] introduces a novel approach for generating initial vertex representations through a random surfing method, taking into account the relative distances between nodes. The

formulation of the embedding for node i involves assessing the probability p_{ij} of transitioning from node i to node j within a specified number of steps. The representation for the i-th vertex should be constructed as:

$$r_i = \sum_{k=1}^{k=K} w(k) p_{ik}^*$$

where w is a decreasing function. The derived node representations are further utilized to construct a Positive Pointwise Mutual Information (PPMI) matrix, subsequently subjected to reconstruction using Stacked Denoising Autoencoders, as shown in Fig.3. Evaluation of the DNGR model extends to diverse tasks, including clustering and word similarity assessments and results outperform conventional Singular Value Decomposition (SVD) and random walk methods.

Concurrently, Structural Deep Network Embedding (SDNE) [26] proposes a stacked autoencoder architecture with the dual objective of jointly preserving first-order and second-order proximity among nodes. Specifically, the embeddings of neighboring nodes and nodes sharing analogous rows in the adjacency matrix are designed to exhibit similarity. The encoder, implemented as an MLP, processes rows from the adjacency matrix as input, aiming to minimize the divergence between latent representations of adjacent nodes, while the decoder endeavors to reconstruct the original row of the adjacency matrix. Formally. the encoder loss is:

$$L_{enc} = \sum_{(u,v)\in E} A_{u,v} ||enc(x_u) - enc(x_v)||^2$$

with the encoder being a multilayer perceptron and x_v a row of the adjacency matrix, while the decoder loss is defined as:

$$L_{dec} = \sum_{u\in V} A_{u,v} ||(dec(enc(x_u)) - x_v) \odot b_u||^2$$

where $b_{u,v} = 1$ if $A_{u,v} = 1$, else $b_{u,v} = \beta > 1$. The acquired representations preserve the local and global network structure and are robust to sparse networks thus, the results demonstrate the superior usefulness of the method in multi-label classification, reconstruction, link prediction and visualization.

B. Convolutional encoders

The initial methodologies solely considered the structural attributes of the graph, neglecting the individual feature information associated with each node. Moreover, these methods inherently exhibit transductive characteristics, rendering them incapable of extending predictions to previously unseen nodes, as their predictions are confined to a singular, immutable graph. In contrast, subsequent convolutional techniques incorporate node features, facilitating the acquisition of an embedding function within the spatial or spectral domain. This enables the model to generalize effectively to unseen nodes, while acquiring a nuanced understanding of the topological configuration within each node's neighborhood and the distribution of node features within said neighborhoods.

1) Spatial convolution methods: GraphSAGE [7], a generalization of Weisfeiler-Lehman (WL) isomorphism test, employs two graph convolutional layers to encode node features and learns how to effectively aggregate feature information from a node's local neighborhood. The model aggregates representations from a randomly sampled subset of nodes within a node's immediate vicinity, utilizing a differentiable function. The new vector is concatenated with the existing node representation and passes through a matrix operation followed by a nonlinear transformation. The aggregator can assume various forms, such as a simple mean function, max pooling or a learnable LSTM.

$$h_{N(u)}^k \leftarrow AGGREGATE_k(\{h_u^{k-1}, \forall u \in N(u)\})$$

$$h_u^k \leftarrow \sigma(W^k * CONCAT(h_u^{k-1}, h_{N(u)}^k))$$

The model parameters are iteratively optimized to ensure proximate nodes exhibit closely aligned embeddings, while disparate nodes possess highly distinct representations. This is accomplished by minimizing the following loss criterion which utilizes contrastive techniques.

$$J_{z_u} = -log(\sigma(z_u^T z_v)) - Q * \mathbb{E}_{u_n \sim P_n(u)}[log(\sigma(-z_u^T z_v))]$$

where v is a node that co-occurs near u on fixed-length random walk and P_n is a negative sampling distribution, and Q defines the number of negative samples. GraphSAGE attains competitive performance in classification tasks even when presented with previously unseen graphs.

The aforementioned algorithms rely on random-walk objectives which are known to prioritize proximity information at the expense of structural insights and there is ambiguity regarding their utility, given that convolutional encoders introduce an inductive bias that neighboring nodes exhibit similar representations. Deep Graph Infomax [23] does not rely on random walk objectives, and is readily applicable to both transductive and inductive learning setups. This methodology centers on maximizing the mutual information between patch representations and corresponding high-level graph summaries. These representations are generated through established graph convolutional architectures facilitating the exploration and preservation of similarities at the patch-level, including distant nodes with similar structural roles.

They initially create a corrupted graph from which they sample negative examples and obtain representations h_i for initial graph nodes and $\tilde{h_i}$ for corrupted

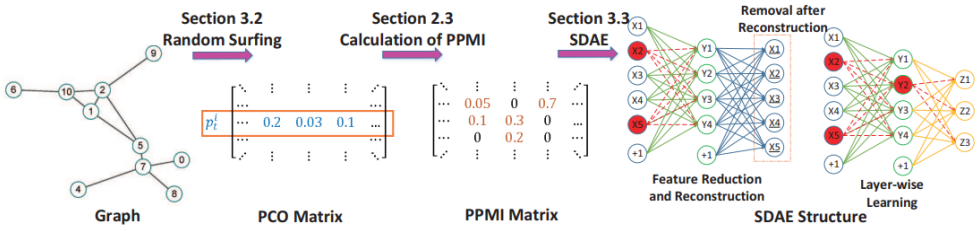

Fig. 3: Main components of DNGR [2]: random surfing, calculation of PPMI matrix and feature reduction by SDAE

graph nodes through a convolutional based encoder. Subsequently, the input graph is summarized by aggregating these representations to obtain a global graph representation s. Finally, they train a discriminator based on the following loss function in order to maximize the correspondence between a node representation and which graph the node belongs to.

$$\sum_{i=1}^{i=N} \mathbb{E}_{(X,A)}[log(D(h_i, s)] + \sum_{j=1}^{j=M} \mathbb{E}_{(\tilde{X},\tilde{A})}[log(1-D(\tilde{h_j}, s)]$$

Despite its experimental superiority over GraphSage, DGI has no guarantee to reconstruct the structural information of node neighborhoods. Recent works even demonstrate that maximizing such correspondence risks capturing only noisy information irrelevant to downstream tasks as the models could achieve mutual information maximization solely through such noisy information [21].

2) Spectral Convolution methods: MGAE [25] introduces a single layer graph convolutional autoencoder in the spectral domain as well as some sort of "dynamics". The single layer lost function is defined as: $||X - f(X, A)||^2$, with $f(X, A)$ signifying the spectral convolutional function $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW$. The dynamic element is introduced through the iterative addition of random noises to the content information, with the expectation that the autoencoder will effectively mitigate the noise. In order to obtain a deeper architecture multiple instances of such autoencoders are stacked. By applying spectral clustering algorithms to the outputs of the final layer, MGAE demonstrates superior performance compared to conventional methods in the realm of graph clustering. We demonstrate this framework in Fig, 4

However, since MGAE reconstructs the feature matrix of nodes without hidden layers, it cannot manipulate the dimension of the latent representation and performs a linear mapping. This is a distinct limitation in finding a latent representation that clearly reveals the structure of the graph. In contrast GALA [17] has a newly devised decoder that constructs a fully symmetric autoencoder form. This design facilitates the incorporation of the graph structure throughout the entire autoencoder processes. The encoder performing Laplacian smoothing aligns the latent representation of each node with those of its neighboring nodes, while the decoder performs Laplacian sharpening as the counterpart of Laplacian smoothing, a procedure that positions the reconstructed feature of each node farther away from the centroid of its neighbors. Empirical assessments on both network and image datasets demonstrated the validity of the proposed framework, showcasing superior performance in comparison to various graph-based clustering algorithms.

Nonetheless, conventional graph convolutional filters exhibit suboptimal low-pass filtering characteristics, as they are unable to effectively filter out noise within certain high-frequency intervals. Thus, they can not reach the best smoothing effect. In addition, recovering the feature matrix will force the model to remember high-frequency noises in features. In response to these challenges, Adaptive Graph Encoder (AGE) [3] consists of a well-designed non-parametric Laplacian smoothing filter to perform low-pass filtering, in order to attain a superior smoothing effect on the features. The AGE framework further incorporates an adaptive encoder that iteratively reinforces the filtered features, enhancing the quality of node embeddings. AGE first filters node feature matrix X with optimal laplacian filter. Subsequently, the model selects positive and negative samples by computing the cosine similarity of the filtered matrix X and proceeds to train the adaptive encoder employing a cross-entropy criterion.

C. Learning embeddings recursively or adversarially

The inherent sparsity of a graph contributes to a substantial imbalance between the number of positive and negative node pairs. To alleviate the data sparsity problem in learning network embedding, another line of work converts a graph into sequences through random permutations or random walks coupled with adversarial techniques. DRNE [22] learns node representations that can well preserve pair-wise regular equivalence and reflect several popular and typical node centralities. This is achieved by organizing neighbor embeddings based on

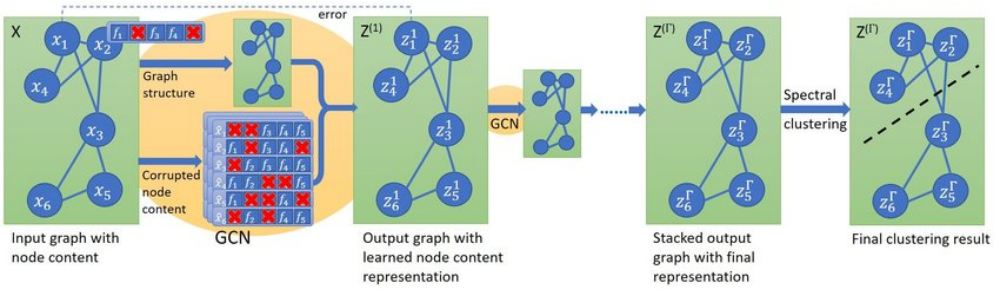

Fig. 4: Framework of Marginalized Graph Autoencoder (MGAE) [25] for graph clustering

node degree and subsequently feeding the resulting node sequence into an LSTM with layer normalization. The reconstruction error is defined as:

$$L = \sum_{u \in V} ||z_u - LSTM\{z_u | u \in N(u)\}$$

The acquired embeddings are then regularized by expecting an MLP to approximate the node degree.

Within a similar framework, NetRA [27] endeavors to acquire meaningful vertex representations through the training of an adversarially regularized LSTM encoder. The model concurrently minimizes both the network locality-preserving loss and the reconstruction error of an autoencoder, where an LSTM serves as the encoder, mapping input random-walk sequences to a fixed-length representation. NetRA also incorporates generative adversarial training to enhance discrete-space representation learning, effectively mitigating the overfitting problem associated with sparsely sampled walks. Specifically, in NetRA, the discriminator updates by contrasting samples from the latent space of the autoencoder with fake samples generated by the generator. The resulting representations demonstrate utility in downstream tasks, such as link prediction, network reconstruction and multi-class classification.

An additional framework that leverages an adversarial graph embedding paradigm for processing graph data is presented in ARGA [16]. In this work latent representations are compelled to align a prior distribution via an adversarial training scheme by minimizing the reconstruction errors of the graph structure while enforcing the latent codes to match a prior distribution. The encoder applies a spectral convolutional function to encode both the graph structure and node content into a cohesive representation, a gaussian distribution:

$$q(z_i|X, A) = N(z_i|\mu_i, diag(\sigma^2))$$

with μ and $log\sigma$ the output of spectral convolutional encoders. A sigmoid function over the inner product of node embeddings approximates the likelihood of link existence between nodes.

$$p(\hat{A}_{ij}|z_i, z_j) = \sigma(z_i^T z_j)$$

An MLP discriminator distinguishes whether a latent code is sampled from a prior distribution or from the graph encoder. The encoder and the discriminator are trained simultaneously in a zero sum game as shown in Fig. 5

D. Other popular techniques (Attention mechanism, self-supervised learning and a probabilistic approach)

In this section we present profoundly impactful independent techniques that rely on attention mechanisms, probabilistic methodologies and self supervised learning. GATE [18] an architecture able to reconstruct graph structured inputs, including both node attributes and the graph structure, introduces stacked encoder/decoder layers equipped with self-attention mechanisms. It can be applied to inductive learning and demonstrates competitive performance surpassing even the benchmarks set by supervised learning baselines.

Graph2Gauss [1] embeds each node as a Gaussian distribution instead of as a single point, allowing us to capture uncertainty about the representation through a deep mapping from the node attributes to the means and variances of the distribution. Subsequently, G2G ranks the Kullback-Leibler (KL) divergence of node distributions based on k-hop paths between nodes. It ensures that the relative order of KL-divergence between node representations aligns with the graph distance. Notably, this model can effectively generalize to previously unseen nodes.

A masked graph autoencoder GraphMAE [8] inspired by the recent success of self-supervised learning in natural language processing, introduces a masking strategy, showcasing the potential of generative self-supervised pre-training applied to graphs. GraphMAE first masks input node features with a mask token [MASK]. The graph with partially observed features is fed into the encoder to generate the encoded node representations. The graph, now containing partially observed features, undergoes encoding to produce the encoded node representations. During decoding, GraphMAE employs a re-masking strategy using a different token [DMASK] on the codes of selected nodes, followed by the utilization

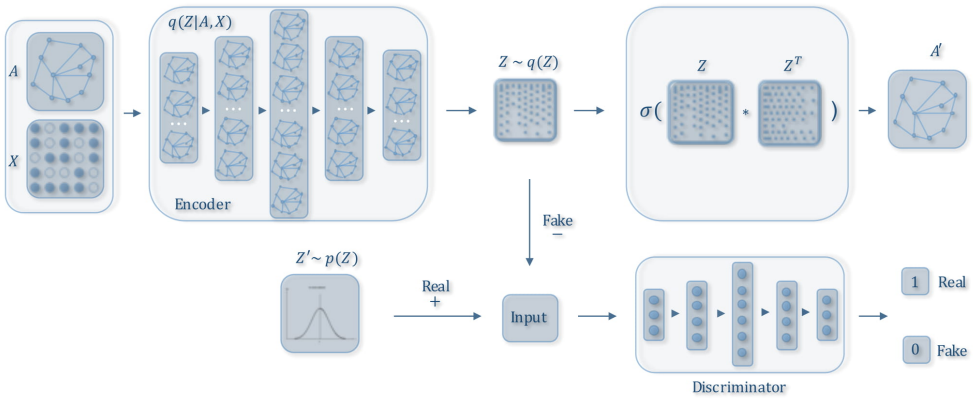

Fig. 5: Architecture of the adversarially regularized graph autoencoder (ARGA) [16] that incorporates adversarial training

of a Graph Neural Network (GNN), such as Graph Attention Network (GAT) or Graph Isomorphism Network (GIN), as the decoder. The output of the decoder is employed to reconstruct the input node features of the masked nodes, with a scaled cosine error serving as the optimization criterion. The authors conduct extensive experiments on a wide range of node and graph classification benchmarks, and the results demonstrate the effectiveness and generalizability of GraphMAE.

IV. Variational Graph Autoencoders for Graph Generation

Incorporating multiple graphs in their training set, Graph Autoencoders (GAEs) demonstrate the capacity to learn generative distributions by encoding them into hidden representations and subsequently decoding graph structures based on these representations. The predominant focus of GAEs for graph generation lies in addressing the molecular graph generation problem, which holds significant practical value in the realm of drug discovery. These methods typically approach the problem by proposing a new graph either in a sequential manner or globally. This section provides a review of approaches utilizing Variational Autoencoders (VAEs) to generate graph structures. A prevalent strategy in these methods involves initially encoding an input graph into a latent space using Graph Neural Networks (GNN), Graph Convolutional Networks (GCN), or their variants, followed by the generation of the graph from this latent space embedding.

To categorize the existing approaches based on their generation granularity level (i.e., adopting an all-at-once generation strategy, using valid substructures as building blocks, or generating graphs in a node-by-node fashion), we divide them into the following three subsections. Additionally, a final subsection is included, focusing on the optimization of the properties of the generated graphs.

A. One-shot graph generation

The Variational Graph Auto-Encoder framework was initially proposed by Kipf and Welling, 2016 [12]. The framework introduces stochastic latent variables z_i that can be sampled to generate the adjacency matrix of a new graph. These stochastic variables are modeled as Gaussian distributions with mean μ_i and variance σ_i^2, computed from a learnable GCN.

$$q(z_i) = N(\mu_i, diag(\sigma_i^2))$$

$$q(Z) = \Pi q(z_i),\ \mu_i = GCN(X, A),\ log\sigma_i = GCN(X, A)$$

The probability of two new nodes sharing a common edge is computed through a simple inner product, $p(A_{ij}) = \sigma(z_i^T * z_j)$ assuming independence between different edges. The learning process involves optimizing the following function L to maximize the likelihood of the graph while maintaining simplicity in the distribution of the latent variables.

$$L = \mathbb{E}_{q(X|Z,A)}[log(P(A|Z)] - KL[q(Z|X,A)||p(Z)]$$

where KL is the Kullback-Leibler divergence and $p(Z)$ an isotropical gaussian prior. This loss function prevails in the majority of works involving graph VAEs. However, this architecture has limitations, as it can only learn from a single graph, and the simplicity of the decoder constrains its capacity as an effective graph generator.

In the probabilistic setting of a VAE, the encoder is defined by a variational posterior $q(z|G)$ and the decoder by a generative distribution $p(G|z)$. There is a prior distribution $p(z)$ imposed on the latent code representation as a regularization, usually a simplistic isotropic Gaussian prior $p(z) = N(0; I)$. GraphVAE [20], extends the aforementioned setting to accommodate learning

from a dataset comprising multiple graphs. The decoder is now a deterministic MLP that computes the Adjacency and Feature Matrix of the generated graphs. In order to compute the posterior probability of the initial graph $p(G|z)$ they leverage an approximate graph matching algorithm for aligning G with the generated $\tilde{G}$, which computational complexity imposes limitations on the framework scalability, particularly for larger graph generation tasks.

An alternative methodology that can handle larger graphs is that of Graphite [6] a latent variable generative model for graphs. In this work, the encoding is straightforward and can use any existing graph neural network, whereas the decoding of these latent features to reconstruct the original graph is conducted through a multi-layer iterative procedure. The iterative process initiates with an initial reconstruction based on the inferred latent features, and progressively refines the reconstructed graph through a message-passing operation. Formally, given Z and X, Graphite iterates over the following two operations:

$$\tilde{A} = \frac{ZZ^T}{||Z||^2} + 1I^T,\ Z^* = GNN(\tilde{A}, [Z|X])$$

where an additional constant of I is added into the first operation to ensure non-negativity of entries. Similar to VGAE [12], Graphite derives the final distribution over graph parameters through an inner product step on Z and is constrained to learning from a single input graph.

A domain specific adaptation of Graphite tailored for handling more intricate constrained graphs, while simultaneously enhancing the foundational structure of GraphVAE, is MPGVAE [4]. In particular GraphVAE's MLP decoder lacks the appropriate inductive bias for graph structured data. In response, MPGVAE introduces a message-passing neural network into both the encoder and decoder components of a VAE. The model's workflow as demonstrated in Fig.6 unfolds as follows: a message-passing phase, comprising T propagation steps coupled with a graph attention mechanism, operates to allow each node v to receive an aggregate message from its neighbors $N(v)$. Subsequently, the encoder employs a set2set model [24] to parameterize the posterior distribution $q(z|G)$. For the decoding process, after sampling initial states of the nodes in the new graph, an RNN is employed, followed by an execution of the message-passing mechanism to generate the final graph. This augmentation results in competitive performance, particularly evident in the domain of molecular generative modeling.

RGVAE [14] directly addresses the challenge of generating semantically valid graphs by introducing a regularization framework for VAEs. In practical applications, instances arise where specific combinations of nodes and edges must adhere to predefined constraints for validity. For instance, in molecular graphs, the count of bonding-electron pairs should not surpass the valence of an atom in the generated graph. This framework focuses on the matrix representation and formulate penalty terms that address such validity constraints. It transforms a constrained optimization problem, that of optimizing L satisfying such constraints to a regularized, unconstrained one by adding inequality constraints to L, which forms a Lagrangian function. More precisely, RGVAE minimizes the loss function L, which is augmented by $\mu_i * \sum_i g_i(\theta, z)$ where g_i represents a validity constraint. These constraints are tailored to reflect molecular properties, ensuring both high validity and novelty in the generated molecules.

A noteworthy extension of the above methods is presented in D-VAE [28], introducing an asynchronous message passing scheme designed to encode computations on Directed Acyclic Graphs (DAGs). In contrast to the simultaneous message passing occurring at all nodes, D-VAE respects the computation dependencies, or the partial order, among nodes. Additionally, it has been demonstrated that D-VAE can injectively encode computations on DAGs. This implies that the model establishes a mapping from the discrete space to a continuous latent space, ensuring a unique embedding for each DAG computation in the latent space. This substantiates the feasibility of performing optimization in the latent space rather than the original design space, offering promising new directions to two important problems, neural architecture search and Bayesian network structure learning.

B. Substructure-based Generators

Several works employ valid chemical substructures as foundational elements to generate more plausible molecular graphs. The aim of these endeavors is to address challenges in drug discovery, specifically the identification of target molecules possessing desired chemical properties. JT-VAE [9] adopts such a strategy by extending the variational autoencoder framework, effectively mitigating the issue of invalidity in intermediate subgraphs. This model generates molecular graphs through a two-phase process. Initially, it constructs a tree-structured scaffold over chemical substructures, and subsequently combines them into a molecule using a graph message passing network. This methodology facilitates the incremental expansion of molecules while ensuring chemical validity at each step.

The model initiates the process by forming a junction tree of clusters, using it as the tree representation of the molecule with predefined constraints on the choice of cliques. Subsequently, the molecule is encoded into a two-part latent representation $z = [z_T; z_G]$, where z_T

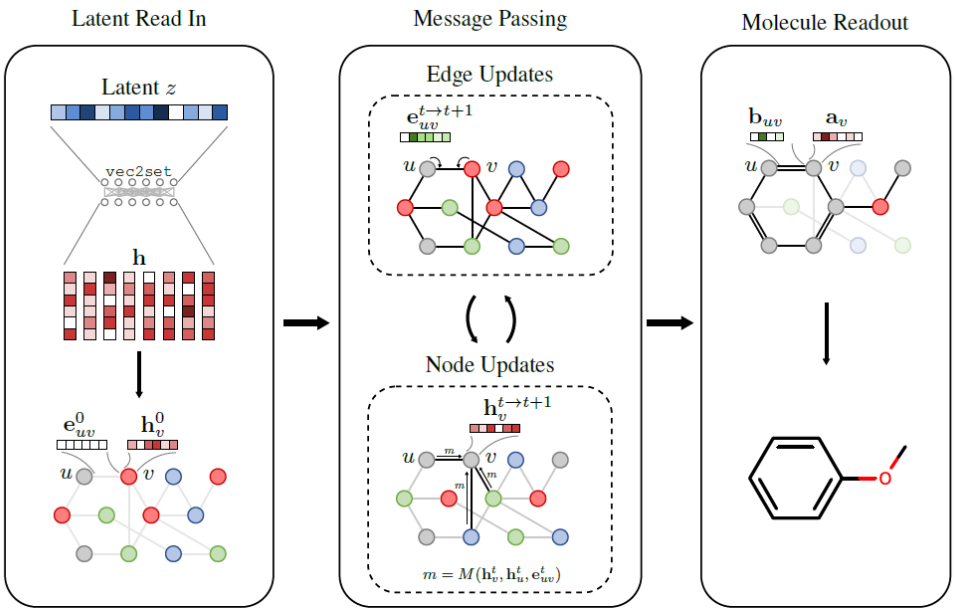

Fig. 6: The Generative Process for MPGVAE [4]

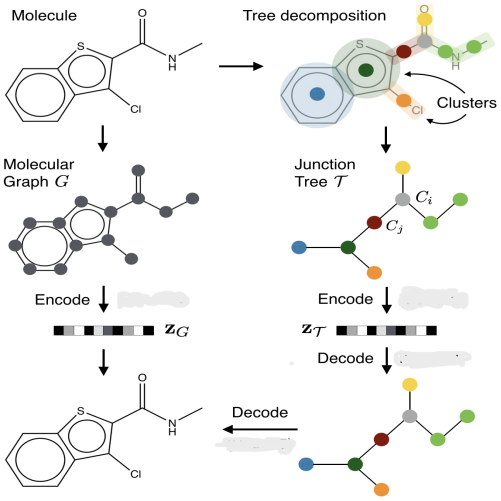

Fig. 7: Overview of Junction Tree VAE [9]

encodes the tree structure, capturing information about the clusters without fully specifying their interconnections, while z_G encodes the fine-grained connectivity of the graph. The decoder reproduces the junction tree using a tree decoder $p(T|z_T)$ followed by predicting the detailed connectivity between clusters in the junction tree using a graph decoder $p(G_j|z_G)$ to reconstruct the full molecular graph. An overview of the method is shown in Fig. 7. This approach demonstrates significant performance improvements in both molecule generation and optimization.

JT-VAE is specifically crafted to utilize small substructures as fundamental components, a design choice that results in diminished performance when tasked with generating larger molecules, such as polymers. Addressing this limitation, HierVAE [10] , extends this framework and proposes a motif based hierarchical graph encoder-decoder that employs significantly larger motifs as foundational building blocks. In the generation process, molecules are constructed incrementally by attaching motifs, whether large or small, to the evolving molecular structure. The hierarchical decoding process unfolds in a coarse-to-fine manner, involving three consecutive key predictions during each pass: the selection of a new motif, determining which part of it attaches, and identifying the points of contact with the existing molecule. These decisions exhibit high interdependence and are naturally modeled in an auto-regressive fashion. Empirical experiments conducted show that this model outperforms prior atom and substructure based methods particularly in the domain of polymers.

While previous tree representations effectively tackle the challenge of ensuring the validity of generated molecules, they are limited in their modeling scope, leaving certain molecular properties to be handled by neural networks. For example, these representations only specify fragment-level connections and fail to specify which atoms in the fragments to be connected. In MHG-VAE [11], a context-free molecular hypergraph grammar, MHG, is developed to address these limitations. MHG ensures the generation of valid molecules and can represent atom-level connections and stereochemistry information. This grammar is inferred from a set of

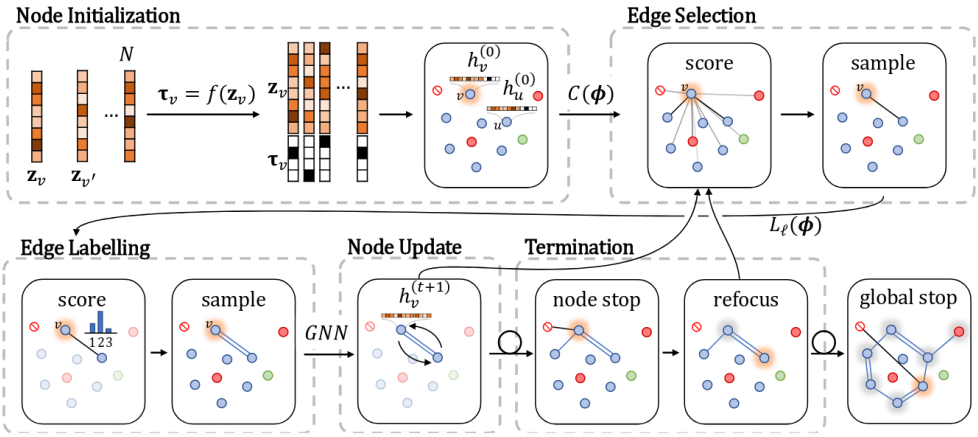

Fig. 8: Generative procedure in CGVAE [13]

molecules and is utilized to generate a parse tree as an intermediate representation of a molecule.The proposed decoder in MHGVAE comprises three components: (i) EncH encodes a molecular graph into a molecular hypergraph, (ii) EncG represents the molecular hypergraph as a parse tree using MHG (iii) EncN encodes the previously generated parse tree into the latent continuous space using a seq2seq GVAE. The model further employs Bayesian optimization to optimize molecule properties in the latent space, ultimately utilizing a decoder that functions as an inversion of the encoder.

C. Node by node generation

In addition to the aforementioned methods, certain other autoencoder-based Graph Generators adopt graph nodes as fundamental building blocks, employing decoders with autoregressive generation strategies. In contrast to approaches that disregard correlations and model the existence and label of each edge with independent random variables, as discussed in Section 4.1, CGVAE [13] factors the distribution into a sequence of discrete decisions within a graph construction trace.

CGVAE designs a generative model where the learned component is conditioned solely on the current state of generation, and the graph is generated sequentially. The encoder initially samples a latent vector z_v for each node v in an input graph from a normal distribution parameterized using gated Graph Neural Networks. Subsequently, the decoder commences from these vectors and sequentially generates a graph node-by-node with the assistance of two decision functions: ”focus” and ”expand.” The former determines the next node to be added to the graph, while the latter iteratively selects edges to add from the currently focused node until a specified stop criterion is met(Fig. 8). The representation of each node is then updated using a standard gated GNN mechanism. During training, the optimization of generated graphs concerning some numerical property is achieved through gradient ascent in the continuous latent space using a differentiable gated regression model, expressed as:

$$R(z_u) = \sum_u \sigma(g_1(z_u) * g_2(z_u)$$

where g_1 and g_2 are neural networks. An L_2 distance loss between labeled Q and $R(z_u)$ shapes the latent space, enabling optimization for the property Q within it.

NeVAE [19] introduces a permutation invariant encoder that learns to aggregate information from a different number of hops away from a given atom and then map this aggregate information into a continuous latent space with variational inference. It also proposes a probabilistic decoder that uses a mask in the edge distribution definition, to prevent the generation of undesirable edges. The decoder samples a latent vector z_v per node v from a normal distribution and utilizes a neural classifier to determine node features, such as the atom type. Subsequently, the total number of graph edges is sampled from a Poisson distribution, parameterized by another neural network conditioned on all latent vectors Z. The decoder then samples graph edges one by one from a softmax distribution among all potential edges not generated up to that point. Similar to CGVAE [13] , a set of binary masks is employed to ensure local structural and functional properties during the edge generation process. Lastly, the edge type is determined by sampling from another softmax distribution, incorporating different binary masks. These masks are updated each time the decoder generates a new edge.

D. Bayesian Optimization in latent space

As was discussed in previous sections the graph variational autoencoder architecture is capable of encoding continuous representations of molecules in continuous

latent space, which is usually leveraged to find molecules that maximize a design metric with gradient-based optimization. In [5], the Automatic Chemical Design model is reformulated as a constrained Bayesian optimization problem formally stated as:

$$max_Z f(Z) \; s.t. \; Pr(C(Z)) \geq 1 - \delta$$

where $f(z)$ is a black-box objective function, $Pr(C(z))$ denotes the probability that a Boolean constraint $C(z)$ is satisfied and δ is some user-specified minimum confidence that the constraint is satisfied. Through thorough experiments they offer compelling evidence that this reformulation can result in tangible enhancements in both the validity and quality of the generated molecules. Moreover, they propose that this approach serves as an effective solution for mitigating the training set mismatch pathology inherent in the unconstrained method for molecule generation.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv: Machine Learning*, 2017.

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1145–1152. AAAI Press, 2016.

[3] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20. ACM, August 2020.

[4] Daniel Flam-Shepherd, Tony C Wu, and Alán Aspuru-Guzik. Graph deconvolutional generation. *ArXiv*, abs/2002.07087, 2020.

[5] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chem. Sci.*, 11:577–586, 2020.

[6] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs, 2019.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[8] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, C. Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[9] Wengong Jin, Regina Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ArXiv*, abs/1802.04364, 2018.

[10] Wengong Jin, Regina Barzilay, and T. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*, 2020.

[11] Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. *ArXiv*, abs/1809.02745, 2018.

[12] Thomas Kipf and Max Welling. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016.

[13] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. In *Neural Information Processing Systems*, 2018.

[14] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *ArXiv*, abs/1809.02630, 2018.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[16] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder. *ArXiv*, abs/1802.04407, 2018.

[17] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning, 2019.

[18] Amin Salehi and Hasan Davulcu. Graph attention auto-encoders. *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 989–996, 2019.

[19] Bidisha Samanta, Abir De, Gourhari Jana, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs, 2019.

[20] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, 2018.

[21] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning, 2020.

[22] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. Deep recursive network embedding with regular equivalence. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[23] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax, 2018.

[24] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets, 2016.

[25] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 889–898, New York, NY, USA, 2017. Association for Computing Machinery.

[26] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1225–1234, New York, NY, USA, 2016. Association for Computing Machinery.

[27] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[28] Muhan Zhang, Shali Jiang, Zhicheng Cui, R. Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. In *Neural Information Processing Systems*, 2019.