Attention-Based Graph Evolution

Shuangfei Fan(✉) and Bert Huang

Virginia Tech, Blacksburg 24060, USA
{sophia23,bhuang}@vt.edu

Abstract. Based on the recent success of deep generative models on continuous data, various new methods are being developed to generate discrete data such as graphs. However, these approaches focus on unconditioned generation, which limits their control over the generating procedure to produce graphs in context, thus limiting the applicability to real-world settings. To address this gap, we introduce an attention-based graph evolution model (AGE). AGE is a conditional graph generator based on the neural attention mechanism that can not only model graph evolution in both space and time, but can also model the transformation between graphs from one state to another. We evaluate AGE on multiple conditional graph-generation tasks, and our results show that it can generate realistic graphs conditioned on source graphs, outperforming existing methods in terms of quality and generality.

Keywords: Conditional graph generation · Attention · Graph evolution

1 Introduction

As a fundamental topic in graph modeling, graph generation has a long history that began as early as the 1950s [6]. However, most traditional methods rely on prior knowledge of the graph topology and are limited in capability of learning generative properties from observations. To solve this problem, researchers have recently been exploring trainable deep models for graph generation based on the effectiveness of graph neural networks—e.g., graph convolutional networks [14]—which have been applied to various kinds of data describing, for example, molecular chemicals for drug design and scientific publications for predicting citations [25,31]. However, these approaches are unconditional generative models, which limits their control over the generating procedure and makes them unable to produce graphs in context. These limitations restrict the applicability of these approaches to real world settings where graphs transform from one state to another and evolve in dynamic network settings.

Modeling graph evolution is an important task that can be applied to various practical applications. A model of graph evolution would be a powerful tool for both predicting the future and the transformation of networks. For example, a marketer aiming to post an advertisement on an online social network may only have access to short-hop ego networks around users, but they need to know how the information would spread into the extended network beyond these ego networks. In disease control and prevention, when an infectious disease emerges and starts to spread, it is important to understand how it may spread beyond the visible network. Because graph data represents real-world phenomena that is changing or incompletely observed, there are many other examples of problems that could benefit from new tools for modeling graph evolution. Yet existing methods lack the flexibility of deep generative models or the ability to condition on previous graph states.

To provide this missing capability, we introduce an attention-based graph evolution model (AGE). AGE is a model for conditional graph generation based on the attention mechanism that allows consideration of global information with parallel computation across all graph nodes. AGE adopts the encoder-decoder structure, where the encoder tries to learn the representation of conditioned graphs using a self-attention mechanism, and the decoder tries to generate the representation of the target graphs using the correlation with the conditioned graphs and also with itself. The decoder can thus capture both global and local information. This graph-conditioned generation framework greatly enriches the potential applications for graph generation. AGE can be used to model not only graph evolution in space and in time, but also the transformation between graphs from one state to another. To evaluate how AGE performs on this problem setting, we perform experiments on datasets in various areas. The experiment results in terms of both the evaluation metrics, show that AGE can not only generate extremely realistic graphs, but also has the strong ability to model the evolution of graphs as a powerful conditioned graph generative model.

2 Related Work

Graph generation is one of the core topics in graph analysis. Many methods have been proposed to solve this problem, which can be traced back to at least 1959 when Erdös and Rényi [6] first introduced the Erdös-Rényi (E-R) model for generating random graphs. The model is based on the assumption that each pair of nodes are connected with a fixed pre-defined probability. However, this assumption is not realistic in most real world networks. To mimic the structure of real graphs, Albert and Barabási [2] proposed the preferential attachment model by further customizing the probability of each possible edge to be conditioned on current degrees of nodes. Separately, Airoldi et al. [1] proposed the mixed-membership stochastic block model (MMSB) to generate graphs that have a fixed number of communities based on a probability matrix to determine the possibility of a node pair from two communities been connected. This model is able to learn distributions from observed data, which makes it generate more useful random graphs based on basic assumptions. Other classical graph generative models include exponential random graph models (ERGMs) [21,26], the stochastic block model (SBM) [9], the Watts-Strogatz model [29], the Kronecker graph model [16], and many more. These older approaches have limited ability to learn about graph distributions from collections of graphs.

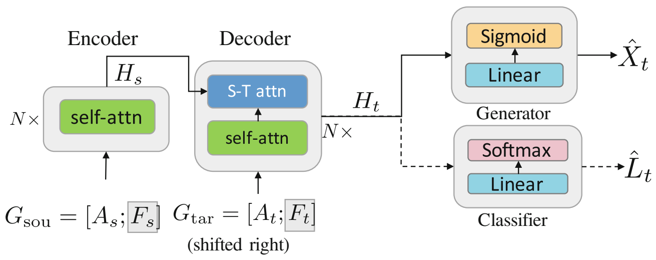

Fig. 1. The model architecture of AGE.

Recently, researchers also have proposed to use deep models to learn distributions for graph generation. These methods can be divided into two categories. Some of them are auto-regressive models, which generate the graph in a sequential manner. Examples of these are the DeepGMG model [18] and the GraphRNN model You et al. [31]. While some other methods are non-auto regressive models [22,25]. Among them, many models are based on generative adversarial networks (GANs) [10], which learn data distributions without explicitly defining a density function [3,5,7]. However, these deep models are either limited to generating small graphs with less than thirty nodes [18,25], or to generating specific types of graphs such as molecular graphs [5,30]. More importantly, the overarching drawback of all these deep generative models is that they are unconditioned, which severely limits their applicability to real-world tasks.

To further strengthen the power of graph generative models, Fan and Huang [7] proposed a conditioned model, which can generate graphs conditioned on discrete labels based on the conditional GAN frameworks [19,20]. However, this approach cannot be applied to circumstances where we want to generate graphs conditioned on another graph, which the motivating case for graph evolution and graph transformation. Also Jin et al. [12] proposed a model with the junction tree encoder-decoder framework for graph to graph transformation. However, they only target the task of molecular optimization. We largely expand the applications of conditioned graph generative model to various interesting problems for graph transformation, such as predicting the graph evolution in space (e.g., how ego-networks would look if expanded to a larger radius) and predicting graph evolution in time (forecasting the changes of dynamic graphs).

3 Attention-Based Graph Evolution Model

We define the prediction of graph evolution as taking an existing *source* graph with nodes (with or without label) as input and predicting, or generating, a transformed version of the graph, or *target* graph. The transformation can represent change over time in a dynamic graph, or expansion in space, such as how ego networks change as we expand out to more steps. Many powerful graph generation models are autoregressive, meaning they generate graphs by sequentially adding new nodes and evaluating relevant possible edges [18,31]. We also adopt this approach and further incorporate an attention-based transformer [27] to process a source graph. We use an attention mechanism instead of a graph convolutional network [14] because attention models can overcome depth limitations of GCNs. Therefore, they can learn more powerful embeddings based on global context.

We model the graph generation procedure as a sequential problem by adding new nodes one-at-a-time. Many other graph generative models such as GraphRNN [31] and DeepGMG [18] also use this same procedural structure; however, they all suffer from efficiency bottlenecks since the sequential procedure prohibits parallelization within instances during the training procedure. This drawback limits their applications on large graphs, especially for DeepGMG, which can only be applied to graphs with less than 30 nodes. To avoid these issues for large graphs with long sequences, we adopt the transformer framework, which instead processes the nodes ordered in a sequence in parallel while using the attention mechanism to incorporate information from all other nodes—even those far away in the sequence. By processing the nodes in parallel, we also significantly shorten the training time, making it much faster than other models.

The architecture of AGE is illustrated in Fig. 1. As in the standard transformer framework, AGE consists of two main components: an encoder E and a decoder D. The encoder learns hidden representations $\boldsymbol{H}_s$ of source graphs through a multi-head attention mechanism (where N is the number of identical layers, we set $N = 6$ in the experiments). The decoder, which is an autoregressive model, then sequentially generates one new node at a time, with possible edges connecting to existing nodes (i.e., the nodes in source graphs and the ones generated previously) and also learns a hidden representations of the target graph $\boldsymbol{H}_t$. In our model, a graph is represented as $G = (\boldsymbol{F}, \boldsymbol{A}, \boldsymbol{L})$ where $\boldsymbol{F}$ is the feature matrix of nodes in source graphs (if one is given), $\boldsymbol{A}$ is the adjacency matrix of source graphs, and $\boldsymbol{L}$ is the label matrix of the nodes in the graph (if labels are available). Among these three components, the adjacency matrix is essential. In some settings, we can leave out the features and labels if we do not have this information. The goal of AGE is therefore to learn a mapping from a source graph G_{sou} to a target graph G_{tar}.

3.1 Self Attention

In AGE, the encoder and the decoder each have their own self-attention block, which is designed to learn high-level node representations based on other nodes within the same graph. In the encoder, the representation of node i in source graphs is updated based on the following rule:

$$\boldsymbol{h}_i^{t+1} = \boldsymbol{h}_i^t + \sigma(\sum_{j=1}^{N_s} a_{i,j}^t \times \boldsymbol{W}_s^s \boldsymbol{h}_j^t), \tag{1}$$

where $a_{i,j}$ is the normalized weights the model learns between node v_i and v_j, $\boldsymbol{h}_i$ is the hidden node feature of node i, N_s is the number of nodes in the source graph, σ is a nonlinear activation and $\boldsymbol{W}_s^s$ is the linear transformation where the weights are learnable parameters separately instantiated for each attention step in the model. The edge weights between two nodes are computed based on the attention mechanism:

$$e_{i,j}^{t+1} = \text{Attention}(\boldsymbol{W}_s \boldsymbol{h}_j^{t+1}, \boldsymbol{W}'_s \boldsymbol{h}_i^{t+1}), \quad a_{i,j}^{t+1} = \frac{\exp(e_{i,j}^{t+1})}{\sum_{k=1}^{N_s} \exp(e_{k,j}^{t+1})}, \tag{2}$$

where $a_{i,j}$ is the normalized attention weight of $e_{i,j}$, which is the attention weight of edge from node i to node j, $\boldsymbol{W}_s$ and $\boldsymbol{W}'_s$ are linear transformations.

3.2 Source-Target Attention

To learn the correlations between the nodes in source graph and the ones to be generated by decoder, we apply a source-target (S-T) attention block after the self-attention operations. The representation of a predicted node j in generated graph is updated based on the learned embeddings of all nodes in the source graph using the following rule:

$$\boldsymbol{h}_j = \boldsymbol{h}_j + \sigma\left(\sum_{i=1}^{N_s} a_{i,j} \times \boldsymbol{W}_s^t \boldsymbol{h}_i\right), \tag{3}$$

where σ is a nonlinear activation function, $\boldsymbol{W}_s^t$ is a learnable linear transformation and $a_{i,j}$ is the normalized weights the model learned between node v_i and v_j. The edge weights between two nodes in different graphs are typically calculated in the same way as shown in Eq. 2.

3.3 Encoder

In AGE, the encoder takes in a source graph G_{sou} represented by its initial representations $G_{\text{sou}} = [\boldsymbol{A}_s; \boldsymbol{F}_s]$ (we can leave out $\boldsymbol{F}_s$ if it is not given) and maps it to a high-level embedding. Here $\boldsymbol{A}_s$ and $\boldsymbol{F}_s$ are the adjacency matrix and the feature matrix of the source graph, where the nodes are arranged in a breadth-first-search (BFS) ordering. We concatenate the feature matrix if we have one. When available, we can use features to generate new node features in addition to the graph structure. In our experiments, we focus on undirected, unweighted graphs where the adjacency matrix $\boldsymbol{A}$ is a symmetric binary matrix with each element represents the connectivity of a pair of nodes, but our approach can be easily extended to both directed and weighted graphs.

We use a fixed maximum number of nodes, N_s for the source graph and N_t for the target graph. AGE can learn about and generate structures with various sizes smaller than these maximums by ignoring isolated nodes in the generated graph. We also define a fixed minimum number of nodes for both source and target graphs to ensure that the input graph is not empty, and to ensure that there are some differences between the source and target graphs.

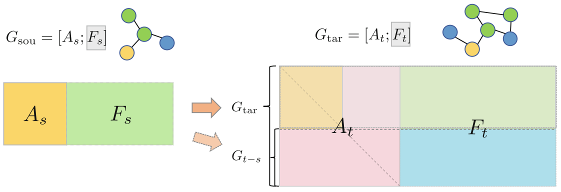

Fig. 2. Data construction for graph evolution in space. The graph and matrix on the left represents the input source graph, which contains a portion of the full target graph on the right. The full target graph contains the adjacency and feature matrices of the source graph in this setting.

3.4 Decoder

The decoder is composed of several stacked attention modules that alternate self-attention and source-target attention layers. The input for the decoder includes two parts: the target graphs G_{tar} and the learned embeddings $\boldsymbol{H}_s$ of the source graph (provided by the encoder). The target graph G_{tar} is represented by the shifted node representations (shifted to the right by one position): $G_{\text{tar}} = [\boldsymbol{A}_t; \boldsymbol{F}_t]$ (leaving out $\boldsymbol{F}_t$ if it is not given), with a start token and an end token filled at the beginning and appended to the end to ensure that the decoder predicts the next node based on the previously generated set. Like the source graph, the nodes in the target graph are also arranged in a breadth-first-search (BFS) ordering at training time, and the model is expected to learn to generate BFS orders.

Given $\boldsymbol{H}_s$, the autoregressive decoder generates an output sequence of nodes one at a time, where each step is also conditioned on the previously generated nodes. The decoder maps the embedding to the space of adjacency matrices and space of label matrices (if the data has label information) to reconstruct the generated graphs. We use a generator which is a combination of a linear transformation and the sigmoid activation function to map $\boldsymbol{H}_t$ to the adjacency matrix $\hat{\boldsymbol{A}}_t$ and we use a classifier which is a combination of a linear transformation and the softmax activation function to map $\boldsymbol{H}_t$ to the label vector $\hat{\boldsymbol{L}}_t$:

$$\hat{\boldsymbol{A}}_t = \text{sigmoid}(\boldsymbol{W}_g \boldsymbol{H}_t); \quad \hat{\boldsymbol{L}}_t = \text{softmax}(\boldsymbol{W}_c \boldsymbol{H}_t). \tag{4}$$

For the predicted adjacency matrix, we use the binary cross-entropy loss function to measure the differences:

$$L_{\text{adj}} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{A}_{ij} \log(\hat{\boldsymbol{A}}_{ij}) + (1 - \boldsymbol{A}_{ij}) \log(1 - (\hat{\boldsymbol{A}}_{ij})). \tag{5}$$

Moreover, if the data has the label information, we also added the loss on labels based on label smoothing using the KL divergence loss.

4 Experiments

In this section, we compare AGE with other graph generation methods on various conditioned graph generation problems to demonstrate its wide applicability.

In the following experiments, we extract 70% of the data as training set, 20% for the validation set and 10% for the test sets. We used six attention layers ($N = 6$) for both self-attention and source-target attention block and within each, we set the number of heads to eight.

Baselines. As we mentioned before, some other methods have been proposed to generate graphs using deep models. However, few of them can condition on existing graphs for general tasks. Therefore, we compare AGE against two categories of other relevant models. The first set consists of methods that can (or can be modified to) generate graphs conditionally, such as the Erdös-Rényi model (E-R) and the Barabási-Albert (B-A) model. These generative models iteratively grow a graph, so they can start from an existing graph. The second set of more recent methods are unconditional graph generation models, such as the mixed-membership stochastic block models (MMSB), DeepGMG and GraphRNN, which include state-of-the-art deep generative models. Notice that due to the computational complexity of the DeepGMG model, we only perform experiments with it on small graphs. In our experiments, we train these models directly on the target graphs without the source graphs.

Evaluation Metrics. We evaluate the generated graphs in two modes. First, we evaluate whether the distribution of generated target graphs is realistic, which captures how well the generative model captures variation in generated graphs. We compute the distances of the distributions of generated graphs and the target graphs using *maximum mean discrepancy* (MMD) [11], following the evaluation procedure used by You et al. [31]. We compute MMD for four graph statistics: degree distribution, clustering coefficient distribution, node-label distribution (if labels are unavailable, the metric "MMD_label" will be listed as N/A), and average orbit count statistics. A model that faithfully captures the conditional distribution over target graphs should have low MMD with the set of true target graphs. Secondly, we compute the similarity between the generated graph and the true target graph for each source graph. This metric evaluates the performance of conditional generation. We calculate the graph similarities using three graph kernels: the shortest path kernel [4] (GK_st), the graphlet sampling kernel [24] (GK_gs), and the SVM-θ kernel [13] (GK_svm). A good conditional graph generator should generate graphs with high similarity to the true target graphs.

4.1 Graph Evolution in Space

Our first evaluation setting considers the graph evolution problem in space. In real-world networks, graph data is collected by subsampling from larger graphs. Due to resource constraints, data collection may not gather as large subsamples as needed. A generative model that can conditionally add nodes in a manner consistent with how graphs grow as one expands the subsample could enable larger analyses of semi-synthetic networks.

Table 1. Comparison of AGE and other generative models on graph evolution in space using MMD evaluation metrics and graph kernel similarities.

	Cora_small							Citeseer						
	Distribution distance				Graph similarity			Distribution distance				Graph similarity		
	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}
E-R	0.33	0.53	0.11	N/A	0.77	0.74	0.93	0.66	0.62	0.21	N/A	0.72	0.74	0.96
B-A	0.35	0.40	0.22	N/A	0.71	0.50	0.53	0.14	0.29	0.14	N/A	0.80	0.78	0.91
MMSB	0.09	0.53	0.14	0.16	0.93	**0.85**	0.98	0.24	1.01	0.15	0.09	0.93	0.84	0.98
DeepGMG	0.37	0.54	0.06	N/A	0.88	0.79	0.90	–	–	–	–	–	–	–
GraphRNN	0.08	0.34	0.09	N/A	0.91	0.76	0.94	0.03	0.23	0.03	N/A	0.86	0.81	0.95
AGE	**0.01**	**0.04**	**0.01**	**0.01**	**0.94**	**0.85**	**0.99**	**0.01**	**0.01**	**0.02**	**0.01**	**0.94**	**0.85**	**0.99**

Datasets. We test this problem setting on citation networks. The problem is to predict the expansion of ego networks with farther-hop neighbors. We used the Cora and Citeseer datasets [23]. We evaluated our models with different graph sizes. For small datasets (Cora_small and Citeseer_small), we extract one-hop ($G_{\text{sou}} = G_1 = \{V_1, E_1\}$) and two-hop ($G_{\text{tar}} = G_2 = \{V_2, E_2\}$) ego networks with $5 \leq |V_1| \leq 20$ and $30 \leq |V_2| \leq 50$ as the source and target graphs. For the large datasets (Cora and Citeseer), we extract two-hop ($G_{\text{sou}} = G_2 = \{V_2, E_2\}$) and three-hop ($G_{\text{tar}} = G_3 = \{V_3, E_3\}$) ego networks with $10 \leq |V_2| \leq 50$ and $40 \leq |V_3| \leq 170$ as the source and target graphs. Data construction for this problem is illustrated in Fig. 2. The training data consists of graph pairs extracted from the datasets. The source graph G_{sou} is the i-hop ego network where the initial embeddings is constructed by concatenating the adjacency matrix $\boldsymbol{A}_s$ and the feature matrix $\boldsymbol{F}_s$ (if $\boldsymbol{F}_s$ is given). The target graphs G_{tar} are the $(i+1)$-hop ego networks of the same node v where the initial embeddings is constructed by concatenating the adjacency matrix $\boldsymbol{A}_t$ and the feature matrix $\boldsymbol{F}_t$.

Results are listed in Table 1. (In all tables, values are rounded to two decimal places.) The metrics indicate that AGE is a strong graph generator in both its ability to mimic graph distributions and match the target graphs. Considering the evaluation of the distance between the distributions of generated graphs and target graphs, AGE achieves the best scores. AGE scores less than 0.1 MMD on all cases, with at least a 30% decrease compared to the second best method, GraphRNN on two datasets with different graph sizes. This result corroborates that, as a graph generative model, AGE can generate realistic graphs that appear to be from the same distribution as the true target graphs. Moreover, considering how well generated graphs match the specific target graphs, we also calculate the graph similarities between the generated graphs and the target graphs. The kernel similarity scores are normalized, so they range from 0 to 1. The graphs AGE generates consistently have the best similarity scores.

4.2 Graph Evolution in Time

Many graph generation methods are designed for static graphs. However in practice, many networks are not static. Instead, they change and evolve over time, with the addition of new nodes and edges, such as in citation networks and collaboration networks, and also with the deletion of existing nodes and edges, such as in computer networks and social networks.

Table 2. Comparison of AGE and other generative models on graph evolution in time using MMD evaluation metrics and graph kernel similarities.

	Facebook-friend							Cit-HepPh						
	Distribution distance				Graph similarity			Distribution distance				Graph similarity		
	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}
E-R	0.54	1.25	0.32	–	0.50	0.55	0.98	0.43	1.15	0.27	–	0.55	0.56	0.93
B-A	0.49	1.08	0.34	–	0.78	0.85	0.78	0.43	0.65	0.16	–	0.71	0.85	0.94
MMSB	**0.09**	0.53	**0.14**	–	0.89	0.85	0.98	0.19	1.20	0.14	–	0.84	0.59	0.99
GraphRNN	0.17	0.18	0.21	–	0.76	0.64	0.95	**0.08**	0.81	0.08	–	0.86	0.69	0.92
AGE	**0.09**	**0.01**	0.19	–	**0.93**	**0.88**	**0.99**	0.10	**0.01**	**0.04**	–	**0.94**	**0.89**	**0.99**
	Bitcoin-OTC							Cit-HepTh_small						
	Distribution distance				Graph similarity			Distribution distance				Graph similarity		
	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}
E-R	0.63	1.12	0.21	N/A	0.57	0.43	0.98	0.33	0.81	0.22	–	0.64	0.24	0.93
B-A	0.40	0.46	0.14	N/A	0.68	0.90	0.95	0.37	0.71	0.28	–	0.63	0.57	0.86
MMSB	0.30	1.17	0.12	0.15	0.80	0.59	0.98	0.28	0.83	0.42	–	0.82	0.36	0.98
DeepGMG	–	–	–	–	–	–	–	0.12	0.68	0.20	–	0.93	0.56	0.92
GraphRNN	0.16	0.43	0.20	N/A	0.84	0.64	0.94	0.05	0.27	0.07	–	0.96	0.80	0.95
AGE	**0.08**	**0.04**	**0.10**	**0.01**	**0.97**	**0.92**	**0.99**	**0.04**	**0.01**	**0.04**	–	**0.99**	**0.94**	**0.99**

Datasets. For this task, we evaluate AGE on three datasets: the Facebook Friendship Networks [28], the Bitcoin Networks [15], and two citation networks in Physics: cit-HepPh and cit-HepTh [8]. We extract two-hop ($G_2 = \{V_2, E_2\}$) ego networks with $30 \leq |V_2| \leq 120$ (or $20 \leq |V_2| \leq 50$ for _small data) at time t as the source graphs and the two-hop ego networks of the same node at time $t+1$ as the target graphs. Here, we have $G_2^t \in G_2^{t+1}$, and the problem is to model how networks evolve (or grow) with actual time.

We compare AGE with other graph generative models and the results are shown in Table 2. The evaluation results show that AGE can accurately model the graph evolution or growth over time. We compute the distance between the distributions of generated graphs and target graphs, and, as before, AGE achieves the best scores among all the generative models regarding the realism of the generated graphs. Again, this is strong evidence that AGE can generate realistic graphs that appear to be from the same distribution of the target graphs. Considering the graph similarities between the generated graphs by all models and the target graphs, Table 2 shows that among all models, AGE is the only one that can reach similarity 0.9 for all three graph kernels, while the other methods cannot consistently score high across different kernels. This suggests some aspect of graph similarity is not satisfied by these other generation procedures. These results again demonstrate that AGE represents a significant step in our ability to model the evolution of graphs in time.

4.3 Graph Evolution in Time with Deletion

To evaluate the performance of AGE on modeling the evolution of graphs with deletion, study cases where the source graphs evolves with not only addition of new nodes and edges, but also allows the deletion of existing nodes and edges.

Table 3. Comparison of AGE and other generative models on graph evolution in time with deletion using MMD evaluation metrics and graph kernel similarities.

	Oregon						
	Distribution distance				Graph similarity		
	Degree	Clustering	Orbit	Label	GK_{st}	GK_{gs}	GK_{svm}
E-R	0.51	0.37	0.25	–	0.55	0.63	0.96
B-A	0.11	0.35	0.21	–	0.85	0.98	0.92
MMSB	0.54	0.39	0.29	–	0.71	0.45	0.93
GraphRNN	0.14	0.12	0.20	–	0.91	0.88	0.93
AGE	**0.01**	**0.01**	**0.01**	–	**0.99**	**0.99**	**0.99**

Datasets. We use the Computer Network dataset [17], which is a network describing peering information inferred from Oregon route-views with nine different timestamps in total. We extract two-hop ($G_2 = \{V_2, E_2\}$) ego networks with $30 \leq |V_2| \leq 120$ at the first and last timestamp, respectively, as the source and target graphs. In this experiment, we focus on the more difficult problem of modeling the evolution of graphs with deletion. The difference with the second experiment is that in this case, the condition $G_2^t \subseteq G_2^{t+1}$ does not hold anymore.

We compare AGE with other graph generative models, listing results in Table 3. The evaluation results show that, even for this more complex problem, AGE still maintains a high-level performance compared to the other generative models in terms of both the realism of generated graphs and the similarity to the target ones. Therefore, together with the second experiment, we find that AGE is not only able to learn graph evolution through growth, but also the more complex setting of volatile evolution.

5 Conclusion

In this work, we proposed attention-based graph evolution (AGE), a conditioned generative model for graphs based on the attention mechanism, which can model graph evolution in both space and time. AGE is capable of generating graphs conditioned on existing graphs. Our model can be useful for many applications in various domains, such as for predicting information propagation in social networks, disease control for healthcare, and traffic prediction in road networks. We model graph generation as a sequential problem, yet we are able to train AGE models in parallel by adopting the transformer framework. Our experimental results demonstrate that AGE is a powerful and efficient conditioned graph generative model, which outperforms all the other state-of-the-art deep generative models for graphs. In our several experiments on various datasets, AGE is to be able to adapt to various kinds of evolution or transformations between graphs, and it performs consistently well in terms of both the realism of its generated graphs and the similarity to ground-truth target graphs. Finally, AGE has a flexible structure that can be used to generate graphs with or without features and labels. This flexibility thus enables a wider range of applications by allowing it to model many forms of graph evolution.

References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. J. Mach. Learn. Res. **9**(Sep), 1981–2014 (2008)
2. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**(1), 47 (2002)
3. Bojchevski, A., Shchur, O., Zügner, D., Günnemann, S.: NetGAN: generating graphs via random walks. In: International Conference on Learning Representations (2018)
4. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining, pp. 8–pp, IEEE (2005)
5. De Cao, N., Kipf, T.: MolGAN: an implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973 (2018)
6. Erdös, P., Rényi, A.: On random graphs I. Publicationes Math. Debrecen **6**, 290–297 (1959)
7. Fan, S., Huang, B.: Labeled graph generative adversarial networks. arXiv preprint arXiv:1906.03220 (2019)
8. Gehrke, J., Ginsparg, P., Kleinberg, J.: Overview of the 2003 KDD cup. ACM SIGKDD Explor. Newsl. **5**(2), 149–151 (2003)
9. Goldenberg, A., Zheng, A.X., Fienberg, S.E., Airoldi, E.M.: A survey of statistical network models. Found. Trends Mach. Learn. **2**(2), 129–233 (2010)
10. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
11. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. J. Mach. Learn. Res. **13**(Mar), 723–773 (2012)
12. Jin, W., Yang, K., Barzilay, R., Jaakkola, T.: Learning multimodal graph-to-graph translation for molecular optimization. In: International Conference on Learning Representations (2019)
13. Johansson, F., Jethava, V., Dubhashi, D., Bhattacharyya, C.: Global graph kernels using geometric embeddings. In: Proceedings of the International Conference on Machine Learning (2014)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
15. Kumar, S., Hooi, B., Makhija, D., Kumar, M., Faloutsos, C., Subrahmanian, V.: REV2: fraudulent user prediction in rating platforms. In: Proceedings of the ACM International Conferene on Web Search and Data Mining, pp. 333–341 (2018)

16. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: an approach to modeling networks. J. Mach. Learn. Res. **11**(Feb), 985–1042 (2010)
17. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 177–187. ACM (2005)
18. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., Battaglia, P.: Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324 (2018)
19. Mirza, M., Osindero, S.: Conditional generative adversarial nets. ArXiv abs/1411.1784 (2014)
20. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the International Conference on Machine Learning, pp. 2642–2651 (2017)
21. Robins, G., Pattison, P., Kalish, Y., Lusher, D.: An introduction to exponential random graph (p*) models for social networks. Soc. Netw. **29**(2), 173–191 (2007)
22. Samanta, B., De, A., Ganguly, N., Gomez-Rodriguez, M.: Designing random graph models using variational autoencoders with applications to chemical design. arXiv preprint arXiv:1802.05283 (2018)
23. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
24. Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: Artificial Intelligence and Statistics, pp. 488–495 (2009)
25. Simonovsky, M., Komodakis, N.: GraphVAE: towards generation of small graphs using variational autoencoders. arXiv preprint arXiv:1802.03480 (2018)
26. Snijders, T.A., Pattison, P.E., Robins, G.L., Handcock, M.S.: New specifications for exponential random graph models. Sociol. Methodol. **36**(1), 99–153 (2006)
27. Vaswani, A., et al.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
28. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in Facebook. In: Proceedings of the Workshop on Online Social Networks, pp. 37–42 (2009)
29. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-world networks. Nature **393**(6684), 440 (1998)
30. You, J., Liu, B., Ying, R., Pande, V., Leskovec, J.: Graph convolutional policy network for goal-directed molecular graph generation. In: Advances in Neural Information Processing Systems (2018)
31. You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J.: GraphRNN: generating realistic graphs with deep auto-regressive models. In: International Conference on Machine Learning, pp. 5694–5703 (2018)