GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders

Martin Simonovsky, Nikos Komodakis

GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders

Martin Simonovsky and Nikos Komodakis

Imagine & LIGM; Université Paris Est & École des Ponts; Champs sur Marne, France
{martin.simonovsky,nikos.komodakis}@enpc.fr

Abstract. Deep learning on graphs has become a popular research topic with many applications. However, past work has concentrated on learning graph embedding tasks, which is in contrast with advances in generative models for images and text. Is it possible to transfer this progress to the domain of graphs? We propose to sidestep hurdles associated with linearization of such discrete structures by having a decoder output a probabilistic fully-connected graph of a predefined maximum size directly at once. Our method is formulated as a variational autoencoder. We evaluate on the challenging task of molecule generation.

1 Introduction

Deep learning on graphs has very recently become a popular research topic [3]. Past work has concentrated on learning graph embedding tasks so far, *i.e.* encoding an input graph into a vector representation. This is in stark contrast with fast-paced advances in generative models for images and text, which have seen massive rise in quality of generated samples. Hence, it is an intriguing question how one can transfer this progress to the domain of graphs, *i.e.* their decoding from a vector representation. Moreover, the desire for such a method has been mentioned in the past [5].

However, learning to generate graphs is a difficult problem, as graphs are discrete non-linear structures. In this work, we propose a variational autoencoder [9] for probabilistic graphs of a predefined maximum size. In a probabilistic graph, the existence of nodes and edges, as well as their attributes, are modeled as independent random variables.

We demonstrate our method, coined GraphVAE, in cheminformatics on the task of molecule generation. Molecular datasets are a challenging but convenient testbed for generative models, as they easily allow for both qualitative and quantitative tests of decoded samples. While our method is applicable for generating smaller graphs only and its performance leaves space for improvement, we believe our work is an important initial step towards powerful and efficient graph decoders.

2 Related work

Graph Decoders in Deep Learning. Graph generation has been largely unexplored in deep learning. The closest work to ours is by Johnson [8], who incrementally constructs a probabilistic (multi)graph as a world representation according to a sequence of input sentences to answer a query. While our model also outputs a probabilistic graph, we do

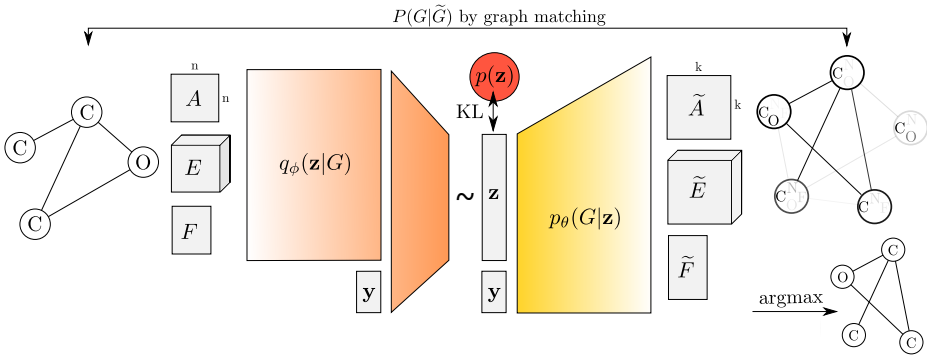

Fig. 1: Illustration of the proposed variational graph autoencoder. Starting from a discrete attributed graph $G = (A, E, F)$ on n nodes (*e.g.* a representation of propylene oxide with 3 carbons and 1 oxygen), stochastic graph encoder $q_\phi(\mathbf{z}|G)$ embeds the graph into continuous representation $\mathbf{z}$. Given a point in the latent space, our novel graph decoder $p_\theta(G|\mathbf{z})$ outputs a probabilistic fully-connected graph $\widetilde{G} = (\widetilde{A}, \widetilde{E}, \widetilde{F})$ on predefined $k \geq n$ nodes, from which discrete samples may be drawn. The process can be conditioned on label $\mathbf{y}$ for controlled sampling at test time. Reconstruction ability of the autoencoder is facilitated by approximate graph matching for aligning G with $\widetilde{G}$.

not assume having a prescribed order of construction transformations available and we formulate the learning problem as an autoencoder.

Xu *et al.* [23] learns to produce a scene graph from an input image. They construct a graph from a set of object proposals, provide initial embeddings to each node and edge, and use message passing to obtain a consistent prediction. In contrast, our method is a generative model which produces a probabilistic graph from a single opaque vector, without specifying the number of nodes or the structure explicitly.

Discrete Data Decoders. Text is the most common discrete representation. Generative models there are usually trained in a maximum likelihood fashion by teacher forcing [22], which avoids the need to backpropagate through output discretization but may lead to expose bias [1]. Recently, efforts have been made to overcome this problem by using Gumbel distribution [10] or reinforcement learning [24]. Our work also circumvents the non-differentiability problem, namely by formulating the loss on a probabilistic graph.

Molecule Decoders. Generative models may become promising for *de novo* design of molecules fulfilling certain criteria by being able to search for them over a continuous embedding space [14]. While molecules have an intuitive and richer representation as graphs, the field has had to resort to textual representations with fixed syntax, *e.g.* so-called SMILES strings, to exploit recent progress made in text generation with RNNs [14, 16, 5]. As their syntax is brittle, many invalid strings tend to be generated, which has been recently addressed by [11] by incorporating grammar rules into decoding. While encouraging, their approach does not guarantee semantic (chemical) validity, similarly as our method.

3 Method

Our method is formulated in the framework of variational autoencoders (VAE) [9]. The main idea is to output a probabilistic fully-connected graph and use a graph matching algorithm to align it to the ground truth. We briefly recapitulate VAE below and continue with introducing our novel graph decoder together with an appropriate objective.

3.1 Variational Autoencoder

Let $G = (A, E, F)$ be a graph specified with its adjacency matrix A, edge attribute tensor E, and node attribute matrix F. We wish to learn an encoder and a decoder to map between the space of graphs G and their continuous embedding $\mathbf{z} \in \mathbb{R}^c$, see Figure 1. In the probabilistic setting of a VAE, the encoder is defined by a variational posterior $q_\phi(\mathbf{z}|G)$ and the decoder by a generative distribution $p_\theta(G|\mathbf{z})$, where ϕ and θ are learned parameters. Furthermore, there is a prior distribution $p(\mathbf{z})$ imposed on the latent code representation as a regularization; we use a simplistic isotropic Gaussian prior $p(\mathbf{z}) = N(0, I)$. The whole model is trained by minimizing the upper bound on negative log-likelihood $-\log p_\theta(G)$ [9]:

$$\mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G|\mathbf{z})] + \mathrm{KL}[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \tag{1}$$

The first term of $\mathcal{L}$, the reconstruction loss, enforces high similarity of sampled generated graphs to the input graph G. The second term, KL-divergence, regularizes the code space to allow for sampling of $\mathbf{z}$ directly from $p(\mathbf{z})$ instead from $q_\phi(\mathbf{z}|G)$ later. While the regularization is independent on the input space, the reconstruction loss must be specifically designed for each input modality.

3.2 Probabilistic Graph Decoder

In a related task of text sequence generation, the currently dominant approach is character-wise or word-wise prediction [2]. However, graphs can have arbitrary connectivity and there is no clear way how to linearize their construction in a sequence of steps: Vinyals *et al.* [21] empirically found out that the linearization order matters when learning on sets. On the other hand, iterative construction of discrete structures during training without step-wise supervision involves discrete decisions, which are not differentiable and therefore problematic for back-propagation.

Fortunately, the task can become much simpler if we restrict the domain to the set of all graphs on maximum k nodes, where k is fairly small (in practice up to the order of tens). Under this assumption, handling dense graph representations is still computationally tractable. We propose to make the decoder output a probabilistic fully-connected graph $\widetilde{G} = (\widetilde{A}, \widetilde{E}, \widetilde{F})$ on k nodes at once. This effectively sidesteps both problems mentioned above.

In probabilistic graphs, the existence of nodes and edges is modeled as Bernoulli variables, whereas node and edge attributes are multinomial variables. While not discussed in this work, continuous attributes could be easily modeled as Gaussian variables represented by their mean and variance. We assume all variables to be independent.

Each tensor of the representation of $\widetilde{G}$ has thus a probabilistic interpretation. Specifically, the predicted adjacency matrix $\widetilde{A} \in [0,1]^{k \times k}$ contains both node probabilities $\widetilde{A}_{a,a}$ and edge probabilities $\widetilde{A}_{a,b}$ for nodes $a \neq b$. The edge attribute tensor $\widetilde{E} \in \mathbb{R}^{k \times k \times d_e}$ indicates class probabilities for edges and, similarly, the node attribute matrix $\widetilde{F} \in \mathbb{R}^{k \times d_n}$ contains class probabilities for nodes.

The decoder itself is deterministic. Its architecture is a simple multi-layer perceptron (MLP) with three outputs in its last layer. Sigmoid activation function is used to compute $\widetilde{A}$, whereas edge- and node-wise softmax is applied to obtain $\widetilde{E}$ and $\widetilde{F}$, respectively. At test time, we are often interested in a (discrete) point estimate of $\widetilde{G}$, which can be obtained by taking edge- and node-wise argmax in $\widetilde{A}$, $\widetilde{E}$, and $\widetilde{F}$. Note that this can result in a discrete graph on less than k nodes.

3.3 Reconstruction Loss

Given a particular instance of a discrete input graph G on $n \leq k$ nodes and its probabilistic reconstruction $\widetilde{G}$ on k nodes, evaluation of Equation 1 requires computation of likelihood $p_\theta(G|\mathbf{z}) = P(G|\widetilde{G})$.

Since no particular ordering of nodes is imposed in either $\widetilde{G}$ or G and matrix representation of graphs is not invariant to permutations of nodes, comparison of two graphs is hard. However, approximate graph matching described further in Subsection 3.4 can obtain a binary assignment matrix $X \in \{0,1\}^{k \times n}$, where $X_{a,i} = 1$ only if node $a \in \widetilde{G}$ is assigned to $i \in G$ and $X_{a,i} = 0$ otherwise.

Knowledge of X allows to map information between both graphs. Specifically, input adjacency matrix is mapped to the predicted graph as $A' = XAX^T$, whereas the predicted node attribute matrix and slices of edge attribute matrix are transferred to the input graph as $\widetilde{F}' = X^T\widetilde{F}$ and $\widetilde{E}'_{\cdot,\cdot,l} = X^T\widetilde{E}_{\cdot,\cdot,l}X$. The maximum likelihood estimates, *i.e.* cross-entropy, of respective variables are as follows:

$$\begin{aligned}
\log p(A'|\mathbf{z}) &= 1/k \sum_a A'_{a,a} \log \widetilde{A}_{a,a} + (1 - A'_{a,a}) \log(1 - \widetilde{A}_{a,a}) + \\
&\quad + 1/k(k-1) \sum_{a \neq b} A'_{a,b} \log \widetilde{A}_{a,b} + (1 - A'_{a,b}) \log(1 - \widetilde{A}_{a,b}) \\
\log p(F|\mathbf{z}) &= 1/n \sum_i \log F^T_{i,\cdot} \widetilde{F}'_{i,\cdot} \\
\log p(E|\mathbf{z}) &= 1/(||A||_1 - n) \sum_{i \neq j} \log E^T_{i,j,\cdot} \widetilde{E}'_{i,j,\cdot}
\end{aligned} \tag{2}$$

where we assumed that F and E are encoded in one-hot notation. The formulation considers existence of both matched and unmatched nodes and edges but attributes of only the matched ones. Furthermore, averaging over nodes and edges separately has shown beneficial in training as otherwise the edges dominate the likelihood. The overall reconstruction loss is a weighed sum of the previous terms:

$$-\log p(G|\mathbf{z}) = -\lambda_A \log p(A'|\mathbf{z}) - \lambda_F \log p(F|\mathbf{z}) - \lambda_E \log p(E|\mathbf{z}) \tag{3}$$

3.4 Graph Matching

The goal of (second-order) graph matching is to find correspondences $X \in \{0,1\}^{k \times n}$ between nodes of graphs G and $\widetilde{G}$ based on the similarities of their node pairs $S : (i,j) \times (a,b) \rightarrow \mathbb{R}^+$ for $i,j \in G$ and $a,b \in \widetilde{G}$. It can be expressed as integer quadratic programming problem of similarity maximization over X and is typically approximated by relaxation of X into continuous domain: $X^* \in [0,1]^{k \times n}$ [4]. For our use case, the similarity function is defined as follows:

$$\begin{aligned} S((i,j),(a,b)) = (E^T_{i,j,\cdot}\widetilde{E}_{a,b,\cdot})A_{i,j}\widetilde{A}_{a,b}\widetilde{A}_{a,a}\widetilde{A}_{b,b}[i \neq j \wedge a \neq b] + \\ + (F^T_{i,\cdot}\widetilde{F}_{a,\cdot})\widetilde{A}_{a,a}[i = j \wedge a = b] \end{aligned} \tag{4}$$

The first term evaluates similarity between edge pairs and the second term between node pairs, $[\cdot]$ being the Iverson bracket. Note that the scores consider both feature compatibility ($\widetilde{F}$ and $\widetilde{E}$) and existential compatibility ($\widetilde{A}$), which has empirically led to more stable assignments during training. To summarize the motivation behind both Equations 3 and 4, our method aims to find the best graph matching and then further improve on it by gradient descent on the loss. Given the stochastic way of training deep networks, we argue that solving the matching step only approximately is sufficient. This is conceptually similar to the approach for learning to output unordered sets [21], where the closest ordering of the training data is sought.

In practice, we are looking for a graph matching algorithm robust to noisy correspondences which can be easily implemented on GPU in batch mode. Max-pooling matching (MPM) by [4] is a simple but effective algorithm following the iterative scheme of power methods. It can be used in batch mode if similarity tensors are zero-padded, *i.e.* $S((i,j),(a,b)) = 0$ for $n < i,j \leq k$, and the amount of iterations is fixed.

Max-pooling matching outputs continuous assignment matrix X^*. Unfortunately, attempts to directly use X^* instead of X in Equation 3 performed badly, as did experiments with direct maximization of X^* or soft discretization with softmax or straight-through Gumbel softmax [7]. We therefore discretize X^* to X using Hungarian algorithm to obtain a strict one-on-one mapping. While this operation is non-differentiable, gradient can still flow to the decoder directly through the loss function and training convergence proceeds without problems. Note that this approach is often taken in works on object detection, *e.g.* [19], where a set of detections need to be matched to a set of ground truth bounding boxes and treated as fixed before computing a differentiable loss.

3.5 Further Details

Encoder. A feed forward network with edge-conditioned graph convolutions (ECC) [17] is used as encoder, although any other graph embedding method is applicable. As our edge attributes are categorical, a single linear layer for the filter generating network in ECC is sufficient. As usual in VAE, we formulate the encoder as probabilistic and enforce Gaussian distribution of $q_\phi(\mathbf{z}|G)$ by having the last encoder layer outputs $2c$ features interpreted as mean and variance, allowing to sample $\mathbf{z}_l \sim N(\mu_l(G), \sigma_l(G))$ for $l \in 1,..,c$ using the re-parameterization trick [9].

Disentangled Embedding. In practice, rather than random drawing of graphs, one often desires more control over generated graphs. In such case, we follow [18] and condition both encoder and decoder on label vector $\mathbf{y}$ associated with each input graph G. Decoder $p_\theta(G|\mathbf{z},\mathbf{y})$ is fed a concatenation of $\mathbf{z}$ and $\mathbf{y}$, while in encoder $q_\phi(\mathbf{z}|G,\mathbf{y})$, $\mathbf{y}$ is concatenated to every node's features just before the graph pooling layer. If the size of latent space c is small, the decoder is encouraged to exploit information in the label.

Limitations. The proposed model is expected to be useful only for generating small graphs. This is due to growth of GPU memory requirements and number of parameters ($O(k^2)$) as well as matching complexity ($O(k^4)$), with small decrease in quality for high values of k. In Section 4 we demonstrate results for up to $k = 38$. Nevertheless, for many applications even generation of small graphs is still very useful.

4 Evaluation

We demonstrate our method for the task of molecule generation by evaluating on two large public datasets of organic molecules, QM9 and ZINC.

4.1 Application in Cheminformatics

Quantitative evaluation of generative models of images and texts has been troublesome [20], as it very difficult to measure realness of generated samples in an automated and objective way. Thus, researchers frequently resort there to qualitative evaluation and embedding plots. However, qualitative evaluation of graphs can be very unintuitive for humans to judge unless the graphs are planar and fairly simple.

Fortunately, we found graph representation of molecules, as undirected graphs with atoms as nodes and bonds as edges, to be a convenient testbed for generative models. On one hand, generated graphs can be easily visualized in standardized structural diagrams. On the other hand, chemical validity of graphs, as well as many further properties a molecule can fulfill, can be checked using software packages (`SanitizeMol` in RDKit [12]) or simulations. This makes both qualitative and quantitative tests possible.

Chemical constraints on compatible types of bonds and atom valences make the space of valid graphs complicated and molecule generation challenging. In fact, a single addition or removal of edge or change in atom or bond type can make a molecule chemically invalid. Comparably, flipping a single pixel in MNIST-like number generation problem is of no issue.

To help the network in this application, we introduce three remedies. First, we make the decoder output symmetric $\widetilde{A}$ and $\widetilde{E}$ by predicting their (upper) triangular parts only, as undirected graphs are sufficient representation for molecules. Second, we use prior knowledge that molecules are connected and, at test time only, construct maximum spanning tree on the set of probable nodes $\{a : \widetilde{A}_{a,a} \geq 0.5\}$ in order to include its edges (a, b) in the discrete pointwise estimate of the graph even if $\widetilde{A}_{a,b} < 0.5$ originally. Third, we do not generate Hydrogen explicitly and let it be added as "padding" during chemical validity check.

4.2 QM9 Dataset

QM9 dataset [15] contains about 134k organic molecules of up to 9 heavy (non Hydrogen) atoms with 4 distinct atomic numbers and 4 bond types, we set $k = 9$, $d_e = 4$ and $d_n = 4$. We set aside 10k samples for testing and 10k for validation (model selection).

We compare our unconditional model to the character-based generator of Gómez-Bombarelli *et al.* [5] (CVAE) and the grammar-based generator of Kusner *et al.* [11] (GVAE). We used the code and architecture in [11] for both baselines, adapting the maximum input length to the smallest possible. In addition, we demonstrate a conditional generative model for an artificial task of generating molecules given a histogram of heavy atoms as 4-dimensional label $\mathbf{y}$, the success of which can be easily validated.

Setup. The encoder has two graph convolutional layers (32 and 64 channels) with identity connection, batchnorm, and ReLU; followed by the graph-level output formulation in Equation 7 in [13] with auxiliary networks being a single fully connected layer (FCL) with 128 output channels; finalized by a FCL outputting (μ, σ). The decoder has 3 FCLs (128, 256, and 512 channels) with batchnorm and ReLU; followed by parallel triplet of FCLs to output graph tensors. We set $c = 40$, $\lambda_A = \lambda_F = \lambda_E = 1$, batch size 32, 75 MPM iterations and train for 25 epochs with Adam with learning rate 1e-3 and β_1=0.5.

Embedding Visualization. To visually judge the quality and smoothness of the learned embedding $\mathbf{z}$ of our model, we may traverse it in two ways: along a slice and along a line. For the former, we randomly choose two c-dimensional orthonormal vectors and sample $\mathbf{z}$ in regular grid pattern over the induced 2D plane. Figure 2 shows a varied and fairly smooth mix of molecules (for unconditional model with $c = 40$ and within 5 units from the origin). For the latter, we randomly choose two molecules $G^{(1)}, G^{(2)}$ of the same label from test set and interpolate between their embeddings $\mu(G^{(1)}), \mu(G^{(2)})$. This also evaluates the encoder, and therefore benefits from low reconstruction error. In Figure 3 we can find both meaningful (1st, 2nd and 4th row) and less meaningful transitions, though many samples on the lines do not form chemically valid compounds.

Decoder Quality Metrics. The quality of a conditional decoder can be evaluated by the validity and variety of generated graphs. For a given label $\mathbf{y}^{(l)}$, we draw $n_s = 10^4$ samples $\mathbf{z}^{(l,s)} \sim p(\mathbf{z})$ and compute the discrete point estimate of their decodings $\hat{G}^{(l,s)} = \arg\max p_\theta(G|\mathbf{z}^{(l,s)}, \mathbf{y}^{(l)})$.

Let $V^{(l)}$ be the list of chemically valid molecules from $\hat{G}^{(l,s)}$ and $C^{(l)}$ be the list of chemically valid molecules with atom histograms equal to $\mathbf{y}^{(l)}$. We are interested in ratios $\mathrm{Valid}^{(l)} = |V^{(l)}|/n_s$ and $\mathrm{Accurate}^{(l)} = |C^{(l)}|/n_s$. Furthermore, let $\mathrm{Unique}^{(l)} = |\mathrm{set}(C^{(l)})|/|C^{(l)}|$ be the fraction of unique correct graphs and $\mathrm{Novel}^{(l)} = 1 - |\mathrm{set}(C^{(l)}) \cap \mathrm{QM9}|/|\mathrm{set}(C^{(l)})|$ the fraction of novel out-of-dataset graphs; we define $\mathrm{Unique}^{(l)} = 0$ and $\mathrm{Novel}^{(l)} = 0$ if $|C^{(l)}| = 0$. Finally, the introduced metrics are aggregated by frequencies of labels in QM9, *e.g.* $\mathrm{Valid} = \sum_l \mathrm{Valid}^{(l)} \mathrm{freq}(\mathbf{y}^{(l)})$. Unconditional decoders are evaluated by assuming there is just a single label, therefore $\mathrm{Valid} = \mathrm{Accurate}$.

In Table 1, we can see that on average 50% of generated molecules are chemically valid and, in the case of conditional models, about 40% have the correct label which the

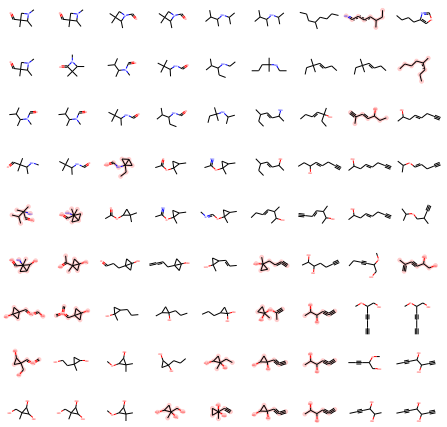

Fig. 2: Decodings over a random plane in $\mathbf{z}$-space. Chemically invalid graphs in red.

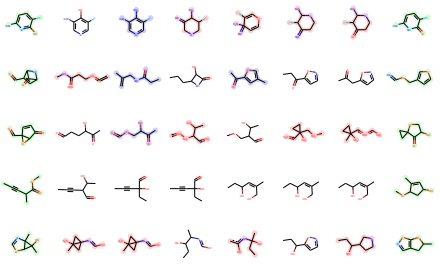

Fig. 3: Linear interpolation between row-wise pairs of randomly chosen molecules in $\mathbf{z}$-space in a conditional model. Color highlight legend: encoder inputs (green), chemically invalid graphs (red), valid graphs with wrong label (blue).

decoder was conditioned on. Larger embedding sizes c are less regularized, demonstrated by a higher number of Unique samples and by lower accuracy of the conditional model, as the decoder is forced less to rely on actual labels. The ratio of Valid samples shows less clear behavior, likely because the discrete performance is not directly optimized for. For all models, it is remarkable that about 60% of generated molecules are out of the dataset, *i.e.* the network has never seen them during training.

Looking at the baselines, CVAE can output only very few valid samples as expected, while GVAE generates the highest number of valid samples (60%) but of very low variance (less than 10%). Additionally, we investigate the importance of graph matching by using identity assignment X instead and thus learning to reproduce particular node permutations in the training set, which correspond to the canonical ordering of SMILES strings from RDKit. This ablated model (denoted as NoGM in Table 1) produces many valid samples of lower variety and, surprisingly, outperforms GVAE in this regard. In comparison, our model can achieve good performance in both metrics at the same time.

Likelihood. Besides the application-specific metric introduced above, we also report evidence lower bound (ELBO) commonly used in VAE literature, which corresponds to $-\mathcal{L}(\phi, \theta; G)$ in our notation. In Table 1, we state mean bounds over test set, using a single $\mathbf{z}$ sample per graph. We observe both reconstruction loss and KL-divergence decrease due to larger c providing more freedom. However, there seems to be no strong correlation between ELBO and Valid, which makes model selection somewhat difficult.

4.3 ZINC Dataset

ZINC dataset [6] contains about 250k drug-like organic molecules of up to 38 heavy atoms with 9 distinct atomic numbers and 4 bond types, we set $k = 38$, $d_e = 4$ and $d_n = 9$ and use the same split strategy as with QM9. We investigate the degree of

Table 1: Performance on conditional and unconditional QM9 models evaluated by mean test-time reconstruction log-likelihood ($\log p_\theta(G|\mathbf{z})$), mean test-time evidence lower bound (ELBO), and decoding quality metrics (Section 4.2). Baselines CVAE [5] and GVAE [11] are listed only for the embedding size with the highest Valid.

		$\log p_\theta(G\|\mathbf{z})$	ELBO	Valid	Accurate	Unique	Novel
Cond.	Ours $c = 20$	-0.578	-0.722	0.565	0.467	0.314	0.598
	Ours $c = 40$	-0.504	-0.617	0.511	0.416	0.484	0.635
	Ours $c = 60$	-0.492	-0.585	0.520	0.406	0.583	0.613
	Ours $c = 80$	-0.475	-0.557	0.458	0.353	0.666	0.661
Unconditional	Ours $c = 20$	-0.660	-0.916	0.485	0.485	0.457	0.575
	Ours $c = 40$	-0.537	-0.744	0.542	0.542	0.618	0.617
	Ours $c = 60$	-0.486	-0.656	0.517	0.517	0.695	0.570
	Ours $c = 80$	-0.482	-0.628	0.557	0.557	0.760	0.616
	NoGM $c = 80$	-2.388	-2.553	0.810	0.810	0.241	0.610
	CVAE $c = 60$	–	–	0.103	0.103	0.675	0.900
	GVAE $c = 20$	–	–	0.602	0.602	0.093	0.809

scalability of an unconditional generative model. The setup is equivalent as for QM9 but with a wider encoder (64, 128, 256 channels).

Our best model with $c = 40$ has archived $\text{Valid} = 0.135$, which is clearly worse than for QM9. For comparison, CVAE failed to generated any valid sample, while GVAE achieved $\text{Valid} = 0.357$ (models provided by [11], $c = 56$). We attribute such a low performance to a generally much higher chance of producing a chemically-relevant inconsistency (number of possible edges growing quadratically). To confirm the relationship between performance and graph size k, we kept only graphs not larger than $k = 20$ nodes, corresponding to 21% of ZINC, and obtained $\text{Valid} = 0.341$ (and $\text{Valid} = 0.185$ for $k = 30$ nodes, 92% of ZINC).

5 Conclusion

In this work we addressed the problem of generating graphs from a continuous embedding in the context of variational autoencoders. We evaluated our method on two molecular datasets of different maximum graph size. While we achieved to learn embedding of reasonable quality on small molecules, our decoder had a hard time capturing complex chemical interactions for larger molecules. Nevertheless, we believe our method is an important initial step towards more powerful decoders and will spark interest in the community.

References

1. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: NIPS. pp. 1171–1179 (2015)
2. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Józefowicz, R., Bengio, S.: Generating sentences from a continuous space. In: CoNLL. pp. 10–21 (2016)

3. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine 34(4), 18–42 (2017)
4. Cho, M., Sun, J., Duchenne, O., Ponce, J.: Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In: CVPR. pp. 2091–2098 (2014)
5. Gómez-Bombarelli, R., Duvenaud, D.K., Hernández-Lobato, J.M., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. CoRR abs/1610.02415 (2016)
6. Irwin, J.J., Sterling, T., Mysinger, M.M., Bolstad, E.S., Coleman, R.G.: ZINC: A free tool to discover chemistry for biology. Journal of Chemical Information and Modeling 52(7), 1757–1768 (2012)
7. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. CoRR abs/1611.01144 (2016)
8. Johnson, D.D.: Learning graphical state transitions. In: ICLR (2017)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. CoRR abs/1312.6114 (2013)
10. Kusner, M.J., Hernández-Lobato, J.M.: GANS for sequences of discrete elements with the gumbel-softmax distribution. CoRR abs/1611.04051 (2016)
11. Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder. In: ICML. pp. 1945–1954 (2017)
12. Landrum, G.: RDKit: Open-source cheminformatics, `http://www.rdkit.org`
13. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. CoRR abs/1511.05493 (2015)
14. Olivecrona, M., Blaschke, T., Engkvist, O., Chen, H.: Molecular de novo design through deep reinforcement learning. CoRR abs/1704.07555 (2017)
15. Ramakrishnan, R., Dral, P.O., Rupp, M., von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Scientific Data 1 (2014)
16. Segler, M.H.S., Kogej, T., Tyrchan, C., Waller, M.P.: Generating focussed molecule libraries for drug discovery with recurrent neural networks. CoRR abs/1701.01329 (2017)
17. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: CVPR (2017)
18. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. In: NIPS. pp. 3483–3491 (2015)
19. Stewart, R., Andriluka, M., Ng, A.Y.: End-to-end people detection in crowded scenes. In: CVPR. pp. 2325–2333 (2016)
20. Theis, L., van den Oord, A., Bethge, M.: A note on the evaluation of generative models. CoRR abs/1511.01844 (2015)
21. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391 (2015)
22. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1(2), 270–280 (1989)
23. Xu, D., Zhu, Y., Choy, C.B., Fei-Fei, L.: Scene graph generation by iterative message passing. In: CVPR (2017)
24. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: AAAI (2017)