MPGVAE: improved generation of small organic molecules using message passing neural nets

MACHINE LEARNING Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
6 November 2020

REVISED
21 March 2021

ACCEPTED FOR PUBLICATION
7 April 2021

PUBLISHED
19 July 2021

PAPER

MPGVAE: improved generation of small organic molecules using message passing neural nets

Daniel Flam-Shepherd[1,2,*], Tony C Wu[3,*] and Alan Aspuru-Guzik[1,2,3,4]

[1] Department of Computer Science, University of Toronto, Toronto, Canada
[2] Vector Institute for Artificial Intelligence, Toronto, Canada
[3] Department of Chemistry, University of Toronto, Toronto, Canada
[4] Canadian Institute for Advanced Research (CIFAR) Senior Fellow, Toronto, Canada
[*] Authors to whom any correspondence should be addressed.

E-mail: danielfs@cs.toronto.edu **and** tonyc.wu@utoronto.ca

Keywords: deep generative models, organic molecule generation, graph neural networks

Abstract

Graph generation is an extremely important task, as graphs are found throughout different areas of science and engineering. In this work, we focus on the modern equivalent of the Erdos–Rényi random graph model: the graph variational autoencoder (GVAE) (Simonovsky and Komodakis 2018 *Int. Conf. on Artificial Neural Networks* pp 412–22). This model assumes edges and nodes are independent in order to generate entire graphs at a time using a multi-layer perceptron decoder. As a result of these assumptions, GVAE has difficulty matching the training distribution and relies on an expensive graph matching procedure. We improve this class of models by building a message passing neural network into GVAE's encoder and decoder. We demonstrate our model on the specific task of generating small organic molecules.

1. Introduction

In the past five years there has been rapid progress in the development of deep generative models for continuous data like images (Kingma and Welling 2014) and sequences like natural language (Sutskever *et al* 2011, Vaswani *et al* 2017, Devlin *et al* 2019). The two most prominent models developed are generative adversarial networks (Goodfellow *et al* 2014, Radford *et al* 2015, Chen *et al* 2016), flow based models (Dinh *et al* 2014, Kingma and Dhariwal 2018) and variational autoencoders (VAEs) (Kingma and Welling 2014), all learn a distribution parameterized by deep neural networks.

There has also been tremendous progress in deep generative models for combinatorial structures, particularly graphs. Graph generation is an important research area with significant applications in drug and material designs (Gómez-Bombarelli *et al* 2016, Zhavoronkov *et al* 2019). However, discovering new compounds with specific properties is an extremely challenging task because of the huge, unstructured and discrete nature of the search space. Hence, more effort needs to be directed toward building simple yet efficient models with the proper inductive biases where the model architecture choices match the structure of the data. We focus our efforts specifically on generating molecular graphs.

One of the first graph generative models is the Erdos–Rényi (ER) random graph model (Erdos and Rényi 1960) where each edge and node exists with independent probability. The modern approach from deep generative models in this class is graph variational autoencoder (GVAE) where the decoder outputs independent probabilities for edge and node features.

Another class of generative models of graphs are sequential models that construct graphs sequentially, node by node. When generating molecules, these models can be constrained to ensure they only generate valid molecules, but they must be trained one molecule at a time since the generative process depends on a sequence of probabilistic decisions. This also makes them more difficult to train since these sequences are typically long, and the model structure is constantly changing (Li *et al* 2018).

Models in the ER family such as GVAE, do not have this requirement and are significantly easier to train. However, because they do not consider edge correlations, it is much harder to match the training distribution with these models. This also makes them difficult if not impossible to constrain so that they only generate valid molecules. For example, Ma *et al* (2018) proposes a regularization framework for GVAEs to generate semantically valid graphs. They formulate penalty terms that address validity constraints. However, this requires a complicated framework of constraints per atom and edge. We seek to improve GVAE similarly but directly by using a more appropriate generative model.

We propose that more effort should be directed toward improving the underlying model design of GVAE before attempting to construct regularization frameworks. In this work, we focus on improving GVAE's basic structure: in particular we note that its multilayer perceptron decoder does not have the proper inductive bias for graph structured data. Therefore, we introduce a simple model for graph generation by building a message passing neural networks (MPNNs) (Gilmer *et al* 2017) into the encoder and decoder of a VAEß (MPGVAE). Our model is still simple to implement but is more adept at domain specific graph generation and can generate from the distribution of training molecules without graph matching. We demonstrate its ability on a few standard tasks in molecular generative modeling.

2. Background

In this paper we focus on undirected molecular graphs G with n nodes denoted $G=(\mathbf{A},\mathbf{E},\mathbf{X})$, where the adjacency matrix is $\mathbf{A}\in\{0,1\}^{n\times n}$ such that $\mathbf{A}_{uv}=1$ implies nodes u and v are connected. The edge feature tensor $\mathbf{E}\in\{0,1\}^{n\times n\times e}$ where $\mathbf{E}_{uvw}=1$ denotes that nodes u and v have edge type w with edge features $\mathbf{e}_{uv}\in\{0,1\}^{e}$ where e is the number of bond types. The node feature matrix $\mathbf{X}\in\mathbb{R}^{n\times f}$ stacks all node feature vectors $\mathbf{x}_v\in\mathbb{R}^f$ where f for number of different atoms.

2.1. Neural message passing

MPNNs operate on graphs G. First, a message passing phase runs for T propagation steps and is defined in terms of message functions M_t and node update functions U_t. During the message passing phase, hidden states $\mathbf{h}_v^t$ of each node in the graph are updated based on messages $\mathbf{m}_v^{t+1}$ according to

$$\mathbf{m}_v^{t+1}=\sum_{w\in\mathcal{N}_v}M_t(\mathbf{h}_v^t,\mathbf{h}_w^t,\mathbf{e}_{vw}) \tag{1}$$

$$\mathbf{h}_v^{t+1}=U_t(\mathbf{h}_v^t,\mathbf{m}_v^{t+1}). \tag{2}$$

Every node v receives an aggregate message from its neighbors $\mathcal{N}_v$, in this case, by simple summation. Then in the second phase we readout predictions $\mathbf{y}$ based on final node embeddings, after T propagation steps.

$$\mathbf{y}=\texttt{Readout}(\{\mathbf{h}_v^T\}_{v\in G}). \tag{3}$$

2.2. Variational auto-encoders for graphs

A GVAE learns a probability distribution from a set of training graphs $\mathcal{D}_G$ such that we can sample new graphs from it. To do this, a VAE learns a latent representation $\mathbf{z}$ of those graphs so that the generative model $p_\theta(G|\mathbf{z})$, which is defined by a neural network with parameters θ, can generate a graph G. Assuming the training data are independent, the objective is to maximize the log-evidence of the data

$$\mathbb{E}_{p(\mathcal{D}_G)}[\log p(G)]=\mathbb{E}_{p(\mathcal{D}_G)}[\log\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[p(G|\mathbf{z})]], \tag{4}$$

which is intractable but can be lower-bounded by introducing a variational approximation $q_\phi(\mathbf{z}|G)$. Another neural network called the inference network encodes a training graph G into a latent representation $\mathbf{z}$ and outputs the parameters of $q_\phi(\mathbf{z}|G)$

$$q(\mathbf{z}|G)=\mathcal{N}(\mathbf{z}|\mu(G),\sigma(G)). \tag{5}$$

The latent space z is low dimensional to make sure the model does not just memorize the training data. We maximize the lower bound with respect to the generator and inference network parameters:

$$\log p(G)\geqslant\mathbb{E}_{q_\phi(\mathbf{z}|G)}[p_\theta(G|\mathbf{z})]-\mathbb{D}_{\mathrm{KL}}[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \tag{6}$$

where $\mathbb{D}_{\mathrm{KL}}$ denotes the Kullback–Leibler divergence. The first term in (6) is the expected log-likelihood which ensures the generated graphs are similar to the training graphs, in this work it is a cross-entropy loss. The second term in (6) regularizes the latent space to ensure that we spread the density of the generative model around. The prior is $p(\mathbf{z})=\mathcal{N}(0,\mathbf{I})$.

3. Related work

3.1. Graph neural networks

The first neural network operating on graphs model was proposed by Scarselli *et al* (2008), and later improved upon by using gated recurrent neural networks (RNNs) (Li *et al* 2015). Around the same time, Duvenaud *et al* (2015) introduced a convolutional neural network for molecular graphs. These works fit into the general framework of neural message passing constructed by Gilmer *et al* (2017) with different message and node update functions.

3.2. Generating SMILES

Several works have explored training generative models on string based representations of molecules called SMILES (O'Boyle 2012). One of the first, CharacterVAE (Gómez-Bombarelli *et al* 2018) is based on a VAE with RNNs. The GrammarVAE (Kusner *et al* 2017) and SDVAE (Dai *et al* 2018) constrain the decoder in order to follow particular syntactic and semantic rules. Recently, these works were extended by Krenn *et al* (2019) to ensure that every string will be a valid molecule. These models can suffer from posterior collapse (Bowman *et al* 2016) where the approximate posterior collapses to the prior and the latent variables are effectively ignored.

Sequential models are another class of models for graph generation. The first of such models comes from Johnson (2016), which incrementally constructs a graph as a sequence of probabilistic decisions, in order to do some reasoning task. Using this framework, DeepGMG (Li *et al* 2018) built an autoregressive model for graphs conditioned on the full generation history. CGVAE (Liu *et al* 2018) improves upon DeepGMG by building a Gated Graph NN (Li *et al* 2015) into the encoder and decoder of a VAE, but it only conditions on the current partial graph during generation. Graph convolution policy network (You *et al* 2018a) uses a graph convolutional neural network (Kipf and Welling 2016) to sequentially generate molecules in goal-directed way through reinforcement learning. More generally GraphRNN (You *et al* 2018b) generates the adjacency matrix sequentially, one entry or one column at a time through a RNN. Another recent work, graph recurrent attention network (Liao *et al* 2019) generates graphs one block of nodes and associated edges at a time. In general, sequential models have some issues with stability and scaling to larger graphs (Li *et al* 2018).

3.3. Modern ER models

These are models which generate entire graphs at a time where each edge and node is independent; these include Graphvae (Simonovsky and Komodakis 2018) and MolGAN (De Cao and Kipf 2018) which avoids issues arising from node ordering that are associated with likelihood based methods by using an adversarial loss instead (Goodfellow *et al* 2014). Graphite (Grover *et al* 2018) is a generative model of graphs, parameterizing a variational auto-encoder with a graph neural network using a iterative graph refinement strategy in the decoder. This particular work is very similar to MPGVAE but uses a graph convolutional neural network (Kipf and Welling 2016) and includes latent variables per node. Our model can be seen as a domain specific version of graphite for molecular graphs. These models are the focus of this work, since they are very simple, improving them will be useful for molecular design.

4. Message passing GVAE

The MPGVAE consists of an encoder and decoder that both use a MPNN to learn a distribution of molecules. We use a variant of the MPNN from Gilmer *et al* (2017) that uses graph attention (Veličković *et al* 2017) as an aggregation method and the message function similar to interaction networks (Battaglia *et al* 2016). The node update function uses a GRUCell (Chung *et al* 2014). After propagation through message passing layers, we use the set2set model (Vinyals *et al* 2015) as the readout function to combine the node hidden features into a fixed-size graph level representation vector.

The encoder encodes a molecular graph into fixed dimensional latent representation and the decoder reconstructs the molecule from that latent representation. Based on the assumptions of the ER family, the decoder assume the probability distribution of graphs G factorizes as:

$$p(G|\mathbf{z}) = \prod_{v \in G} p(\mathbf{x}_v|\mathbf{z}) \prod_{u \in G} p(\mathbf{e}_{uv}|\mathbf{z}) \tag{7}$$

over nodes $u, v \in G$ (Graph G). For molecular graphs we have that both edges and node features are categorical with respective probabilities $\mathbf{p}_v, \mathbf{p}_{uv}$ which are output by the model's decoder.

$$\mathbf{x}_v \sim \mathtt{Cat}(\mathbf{p}_v) \text{and } \mathbf{e}_{uv} \sim \mathtt{Cat}(\mathbf{p}_{uv}) \tag{8}$$

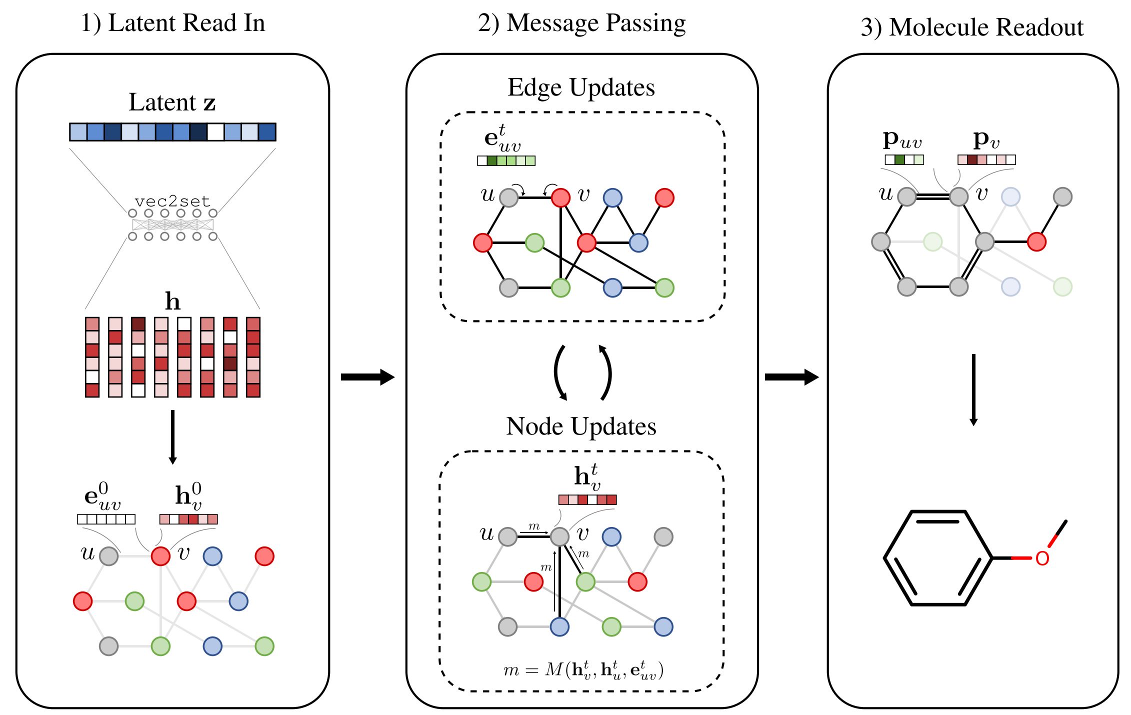

Figure 1. MPGVAE generating a single molecule from its latent representation. In column (1) we read in an unconnected graph from the fixed dimension latent space and generate an initial edge and node representation. In column (2) we perform message passing on that initial graph to create a final edge and node representation. In column (3) we use that final representation to read out a generated graph by creating some edge and node probabilities that can be used to sample a new molecule.

we also treat non-existent nodes as a category for flexible molecular sizes and non-connecting edges as a category (in all $e + 1$ edge types and $f + 1$ atom types).

The basic structure of the decoder which performs three main graph generation steps: (1) Graph read in (2) Message passing and (3) Graph readout. These three main steps that takes a latent vector and transforms it into a molecule are visualized in figure 1 and described further below.

(a) **Graph read in.** The first step, reads in the initial graph representation from the fixed dimensional latent space by projecting the latent representation with a linear layer then passing each vector through a RNN cell to construct an initial state for each node. Each edge is initialized with a zero vector.
(b) **Message passing.** Afterwards, using the initialized graph we perform message passing on both the node and edge representations. This allows us to create a representation we can use to read out a graph
(c) **Graph readout.** Lastly, using the final edge and node representation we transform the edge representation and predict independent edge and node probabilities.

GVAE with graph matching has complexity scaling with $\mathcal{O}(n^4)$ where n is number of nodes while MPGVAE is roughly $\mathcal{O}(n^2\ell^2)$ where ℓ is the dimension of the node level representation in the decoder's MPNN. MPGVAE has lower complexity but can better capture the training distribution.

Algorithm 1 defines a single generation step for the MPGVAE starting by encoding a training molecule then reconstructing it using the decoder. The next two subsections discuss the encoder and decoder which form the two main components of algorithm 1.

We construct the MPNN detailed below using components from a few models that achieve state of the art results in variety of geometric deep learning tasks (1) a message aggregator similar to Veličković *et al* (2017) (2) the message function from Battaglia *et al* (2016) (3) the graph level readout from Gilmer *et al* (2017) know as `Set2Vec` which is used to perform graph level aggregation after message passing in the encoder only.

4.1. The encoder

For our latent space, the variational posterior $q(\mathbf{z}|G) = \mathcal{N}(\mu, \sigma)$ and prior $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ are multivariate Gaussians with diagonal covariance matrices, in the latter, the identity matrix $\mathbf{I}$.

The encoder uses a MPNN (Gilmer *et al* 2017) which takes in a molecular graph and encodes it into a fixed dimensional representation that is mapped to the parameters of the variational approximation. We describe the model below:

Algorithm 1. Generation Step with MPGVAE.

```
1: Input $G \sim p(\mathcal{D}_G)$                    ▷ specify distribution of graphs
2: Initialize $\theta, \phi$
3: Sample $\mathbf{x}_v, \mathbf{e}_{uv} \sim p(\mathcal{D}_G)$
4: $\mathbf{h}_v^T = \texttt{MessagePassing}(\mathbf{x}_v, \mathbf{e}_{uv})$
5: $\mu, \log\sigma = \texttt{Set2Vec}(\{\mathbf{h}_v^T, \mathbf{x}_v\})$
6: $\mathbf{z} \sim q_\phi(\mathbf{z}|G) \Longrightarrow \mathbf{z} = \mu + \sigma \odot \epsilon, \ \ \epsilon \sim \mathcal{N}(0, \mathbf{I})$
7: $\{\mathbf{x}_v^{(0)}, \mathbf{e}_{uv}^{(0)}\} = \texttt{ReadIn}_\theta(\mathbf{z})$
8: $\{\mathbf{x}_v^{(T)}, \mathbf{e}_{uv}^{(T)}\} = \texttt{MessagePassing}_\theta(\mathbf{x}_v, \mathbf{e}_{uv})$
9: $\tilde{G} = \texttt{ReadOut}(\{\mathbf{x}_v^{(T)}, \mathbf{e}_{uv}^{(T)}\}_{u,v})$
```

4.1.1. Messages
We pass messages along edges where each message is updated using the current edge and node representation

$$\mathbf{m}_{uw}^t = \mathrm{Tanh}(\mathbf{W}_e \mathbf{e}_{uw}^t + \mathbf{W}_u \mathbf{h}_u^t + \mathbf{W}_w \mathbf{h}_w^t) \tag{9}$$

for edge $\mathbf{e}_{u,w}$ between nodes u, w where the $\mathbf{W}_e, \mathbf{W}_u, \mathbf{W}_w$ are weight matrices. This is a similar message function to the one found in interaction networks (Battaglia *et al* 2016).

4.1.2. Edge update
We use the message representation to directly update edges $\mathbf{e}_{uv}^{t+1} = \mathbf{m}_{uv}^t$ For all edges in the graph. The edges are initialized with the bond features.

4.1.3. Node update
For each node we construct a message by aggregating using a graph attention (Veličković *et al* 2017) where we first compute attention coefficients

$$a_{uv} = \exp(\mathbf{W}\mathbf{m}_{uv}^t) / \sum_{w \in \mathcal{N}_u} \exp(\mathbf{W}\mathbf{m}_{uw}^t). \tag{10}$$

Then aggregate over neighborhoods by summation:

$$\mathbf{m}_{uv}^{(t+1)} = \sum_{w \in \mathcal{N}_v} a_{uv} \mathbf{m}_{uw}^t. \tag{11}$$

Then each node is updated using the previous state and messages using a Gated Recurrent cell (Chung *et al* 2014, Li *et al* 2015) given by:

$$\mathbf{h}_v^{(t+1)} = \texttt{GRUCell}(\mathbf{h}_v^{(t)}, \mathbf{m}_{uv}^{(t+1)}). \tag{12}$$

4.1.4. Read out
After propagation through message passing layers, we use the set2set model (Vinyals *et al* 2015) as the readout function to combine the node hidden features into a fix-sized hidden graph level representation.

$$\mathbf{h}_G = \texttt{Set2Vec}(\{\mathbf{h}_v^{(T)}\}_{v \in G}). \tag{13}$$

Using this representation, the mean and log variance of the variational posterior are computed with linear layer $\ell(\mathbf{h}_G)$: which are used to sample a latent representation using the reparameterization trick (Kingma and Welling 2014)

$$\mathbf{z} = \mu + \sigma \odot \epsilon, \quad \mu, \log\sigma = \ell(\mathbf{h}_G) \tag{14}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

4.2. The decoder

4.2.1. Graph read in
First we read in the raw graph representation from the fixed dimensional latent space by projecting the latent representation to a high dimensional space with a linear layer with a sigmoid activation σ_z. Then we pass each transformed vector through a RNN cell to read in an initial state for each node. Each edge is initialized with a zero vector. This is described in algorithm box 2.

Algorithm 2. `ReadIn`: From an initial sample from the variational approximation we obtain an initial graph (with the max number of nodes—which is the range of u, v) to perform message passing on.

```
1: Input $\mathbf{z} \sim q_\phi(\mathbf{z}|G)$
2: $\mathbf{h}_z = \mathrm{sigmoid}(\mathbf{z})$
3: $\{\mathbf{x}_v^{(0)}\}_v = \mathtt{RNNCell}(\mathbf{h}_z)$
4: $\{\mathbf{e}_{uv}^{(0)} = 0\}_{u,v}$
```

Algorithm 3. `ReadOut`. u, v range over max number of nodes. Returns $G = (\{\tilde{\mathbf{x}}_v, \tilde{\mathbf{e}}_{uv}\})$.

```
1: Input$\{\mathbf{x}_v^{(T)}, \mathbf{e}_{uv}^{(T)}\}_v)$
2: $\{\mathbf{p}_v\}_v = \mathtt{Softmax}(\sigma_x(\mathbf{x}_v^{(T)}))$
3: $\{\mathbf{e}_{uv}\}_{uv} = (\mathbf{e}_{uv}^{(T)} + \mathbf{e}_{vu}^{(T)})/2$
4: $\{\mathbf{p}_{uv}\}_{uv} = \mathtt{Softmax}(\sigma_e(\mathbf{e}_{uv}))$
5: $\{\tilde{\mathbf{x}}_v \sim \mathtt{Cat}(\mathbf{p}_v)\}$ and $\{\tilde{\mathbf{e}}_{uv} \sim \mathtt{Cat}(\mathbf{p}_{uv})\}$
```

4.2.2. Message passing

Using the initialized graph we perform message passing using both the node and edge representations. We use a MPNN that is identical to the encoder's MPNN except does not perform any graph level aggregation (and there is no set2vec). After T propagation rounds we come to the final representation for each edge and node.

4.2.3. Graph readout

Algorithm 3 describes the readout phase of the generation process where using the final edge and node representation we transform each edge representation such that we obtain a symmetric edge tensor—simply by adding its transpose to itself and dividing by 2 thus making it symmetric. From this state, using both representations we can sample independent edge and node probabilities using single layer neural nets σ_x, σ_e with a softmax activation.

$$\mathtt{softmax}(\mathbf{x})_u = \exp(\mathbf{x}_u) / \sum_w \exp(\mathbf{x}_w). \tag{15}$$

5. Experiments

We focus our evaluation efforts on a few standard tasks in generative modeling of molecules:

- **Molecular generation quality.** We test the MPGVAE on the task of generating molecules when sampling from the prior distribution using a variety of established measures and plot samples from the model.
- **Matching the training data distribution.** We also test the MPGVAE on its ability to match the distributional statistics of the training data.
- **Conditional generation.** Lastly, we test the ability of the MPGVAE to generate molecules conditionally based on atom histograms.

5.1. Dataset

In all experiments, we used QM9 (Ramakrishnan *et al* 2015) a subset of the massive GDB-17 chemical database (Ruddigkeit *et al* 2012) with 133 885 organic compounds of up to nine heavy atoms (Carbon, Nitrogen, Oxygen and Fluorine).

5.2. Baselines

We consider two main baselines GraphVAE (Simonovsky and Komodakis 2018) and MolGAN (De Cao and Kipf 2018). We also compare with the Character VAE (CVAE) (Gómez-Bombarelli *et al* 2018) and GrammarVae (GraVAE) (Kusner *et al* 2017) on the first task.

5.3. Model architecture and training

To train the model we used the adam optimization algorithm (Kingma and Ba 2014), training on a v100 GPU. Training time ranges depending on the model architecture but for the best model is around 3–4 h. To optimize hyperparameters we used grid search and we found the best results using a batch size of 96, a

Table 1. Comparison of validity, uniqueness, novelty and VUN # between MPGVAE and baselines.

Model	Valid	Unique	Novel	# VUN
CVAE	0.10	0.68	0.90	612
GraVAE	0.60	0.09	0.81	437
GVAE	0.81	0.24	0.61	1185
MolGAN	0.98	0.10	0.94	921
MPGVAE	0.91	0.68	0.54	**3341**

learning rate of 10^{-4} trained for 1200 epochs. In every layer we search through $\{32, 64, 128, 256\}$ hidden units. We found that the model was able to achieve similar results for many sets of hyperparameters.

Both the encoder and decoder MPNN use four layers with $[32, 64, 64, 128]$ units in the encoder with a set2set model with an LSTM of 128 unit and a linear layer with 64 units. The latent space maps to the graph level representation using a RNN with 128 hidden units and then the decoder uses four GNN layers with $[64, 64, 32, 32]$ hidden units. The latent space uses 18 dimensions.

5.4. Molecular generation quality

5.4.1. Metrics

We generate $N = 10^4$ samples (from the prior) for the MPGVAE and baselines and assess them using the following statistics defined in Simonovsky and Komodakis (2018):

(1) Validity is the ratio between the number of valid and generated molecules N_{valid}/N. Valid means that rdkit can parse the smiles string. (2) Uniqueness is the ratio between the number of unique samples and valid samples $N_{\text{unique}}/N_{\text{valid}}$. (3) Novelty measures the ratio between the set of valid samples not in the training data and the total number of valid samples $N_{\text{novel}}/N_{\text{valid}}$. as well as (4) The number of valid, unique and novel molecules from the sample (# VUN).

We can see from table 1, specifically the bold entry in the # of VUN column—that MPGAVE has three times the # of VUN molecules compared to the next best baseline. It also generates eight times more # of VUN molecules than GraVAE. It is important to notice that MolGAN, GVAE and GraVAE have low uniqueness and cannot generate unique molecules—meaning they only learn to generate a few different kinds of molecules and therefore have a much lower # of VUN molecules. Furthermore, MPGVAE generates more valid and unique molecules than GVAE and is only slightly worse in generating novel molecules.

Figure 2 visualizes some random samples of molecules using the cheminformatics python package 'rdkit' (Landrum 2013) from the MPGVAE and the two closest baselines GVAE and MolGAN, along with samples from the training data QM9. we can see that GVAE generates a lot of molecules that have disconnected and isolated scaffolds, overall the molecules are not very similar in structure to the training molecules sampled. MolGAN also generates some molecules that are disconnected and have isolated scaffolds (the molecules consisting of two or more unconnected sub-graphs). MPGVAE does not generate any molecules that are disconnected scaffolds.

5.5. Comparing distributional statistics

In this section, we use two sets of continuous and discrete metrics quantifying molecular structure and properties to compare molecules sampled from the MPGVAE, the baselines and the training molecules.

5.5.1. Continuous statistics

Similar to Seff *et al* (2019) we leverage three commonly used quantities when assessing molecules: the quantitative estimate of drug-likeness score, the synthetic accessibility score, and the log octanol-water partition coefficient (logP). The metrics depend on many molecular features allowing for an overall comparison of distributional statistics.

5.5.2. Discrete statistics

We consider a second set of discrete distributional statistics to measure how well each model captures the training distribution. Following Liu *et al* (2018), we measure the average number of each atom type in the generated samples, and count the average number rings in each molecule.

5.5.3. Results and discussion

For each model and QM9, we randomly sample 10^4 molecules and calculate the continuous and discrete metrics for each sample. We compare the distribution of metrics that the MPGVAE and baselines generate with the distribution from the training molecules. In figure 3, we display Gaussian kernel density estimates (KDEs) of the continuous metrics calculated with the sampled molecules from the MPGVAE and the

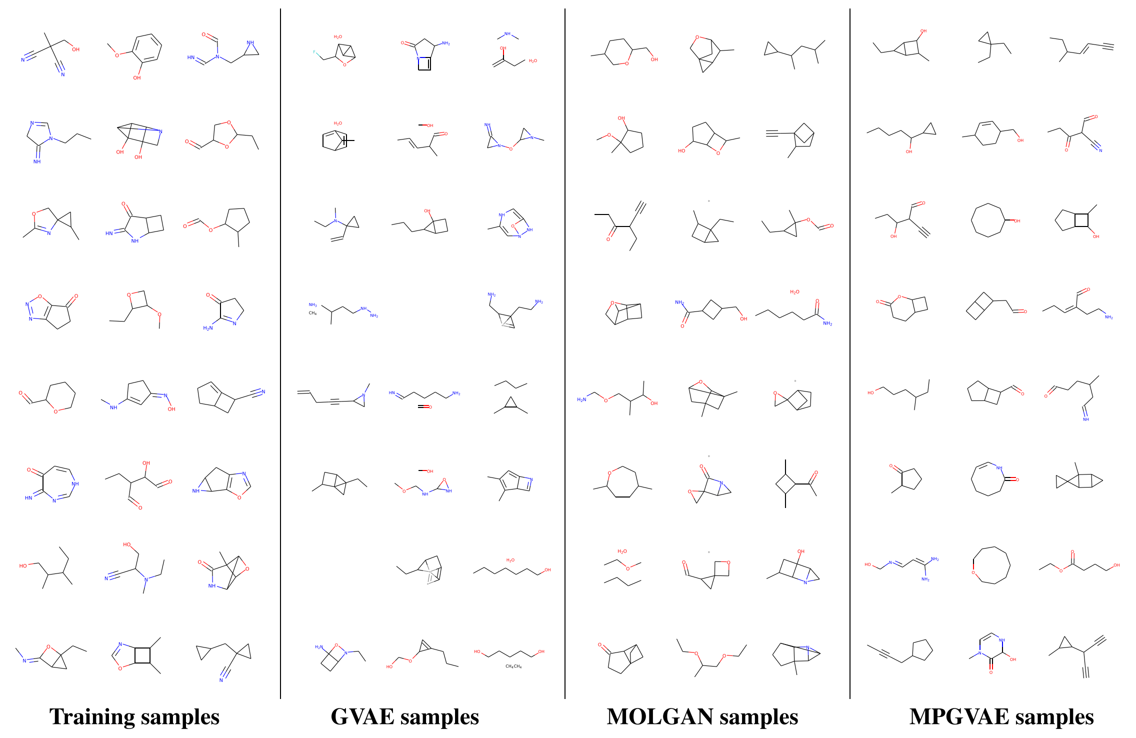

Figure 2. Molecular samples drawn from the latent space (Blue are Nitrogen atoms, red are oxygen, black as Carbon and Teal as Fluorine). Here we compare our MPGAVE to GVAE and MOLGAN samples. We also included a random training samples for representation comparisons. Empty spaces are shown if the molecules are invalid.

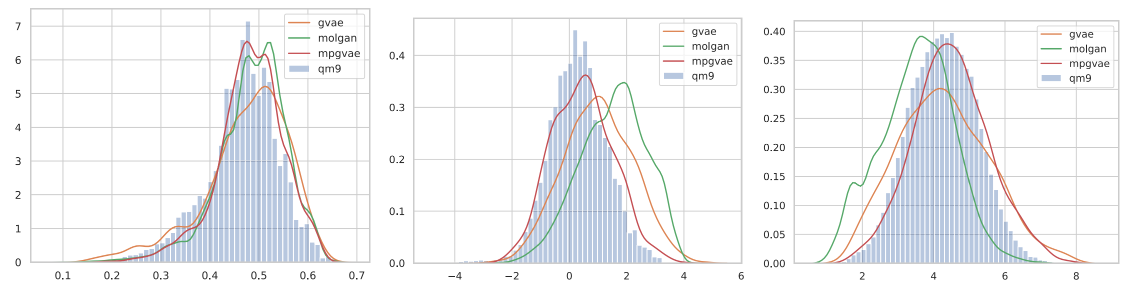

Figure 3. Distributions of QED (left), logp (middle), and SA (right) for sampled molecules and QM9.

baselines. A normalized histogram of the metrics calculated using molecules sampled from QM9 is also shown for visual comparison. It can be seen that in each plot, the KDE of the MPGVAE's samples closer resembles the histogram of the training molecules from QM9 than the baselines. In figure 4 we display a stacked bar plot of the average number of atoms and rings per molecule calculated using the samples. For both average atom and ring count, the stacked bar for the MPGVAE has a closer distribution to QM9's stacked bar than the two other baselines. Hence there is evidence to suggest that using a MPNN in the decoder improves GVAE's ability to generate molecules similar to its training data.

5.6. Conditional generation

For more control over the generated molecules we can condition both encoder and decoder on a label vector $\mathbf{y}$ of atom histograms associated with each input molecule G (Sohn *et al* 2015). The decoder $p(G|\mathbf{z},\mathbf{y})$ 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.

Again we draw 10^4 samples from the prior and as in Simonovsky and Komodakis (2018) compute the discrete point estimate of what is decoded argmax $p(G|\mathbf{z},\mathbf{y})$. We are interested in accuracy which is the ratio of chemically valid molecules with atom histograms equal to their label $\mathbf{y}$ over number of sampled molecules. The results are displayed table 2 and we report that MPGVAE has a substantial improvement in accuracy over GVAE, likely due to the better understanding it has of the training distribution.

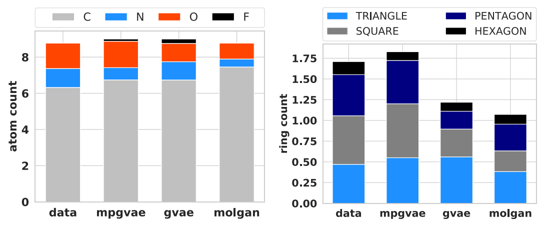

Figure 4. Stacked bar plots of average atom (left) and ring (right) count for 10^4 molecules sampled from MPGVAE, baselines and QM9. Average # of Carbon, Nitrogen, Oxygen and Fluorine atoms in the sampled molecules correspond to grey, blue, orange and black colors. Average # of triangle, square, pentagon and hexagon rings in the sampled molecules correspond to blue, grey, dark blue and black colors.

Table 2. Validity and accuracy of MPGVAE and baseline GAVE in the conditional generation task.

Model	Validity	Accuracy
GVAE	0.57	0.47
MPGVAE	0.89	0.67

6. Conclusion

The modern deep learning version of the ER graph model is the GVAE, because this model does not consider edge correlation it has limited ability as it cannot generate edges or nodes conditionally, based on other structures in the graph. However, since it is simple to implement and not difficult to train, it is still useful for the generation of small organic molecules.

This work focuses on an improved version of GVAE with a MPNN in both its encoder and decoder. We demonstrate in section 5, that this version performs better in a few standard molecular generation tasks.

A limitation of the baseline GVAE is that the model cannot easily differentiate between two graphs with different orderings of nodes as the matrix representation of graphs used is not invariant to permutations of nodes. GVAE attempts to account for this issue by using a graph matching algorithm in its objective. The MPGVAE does not use graph matching but achieves better performance than the GVAE.

Ultimately, more work must be done to improve GVAE's ability to generate larger drug like molecules as QM9 is a dataset of small and less useful molecules. Important areas for improvement and future work include making MPGVAE directly handle the node permutation invariance issue as well as finding a way to constrain MPGVAE to generate only valid molecules.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgment

Machine learning models are trained with the GPUs on Béluga, Calcul Québec. The training was enabled in part by ßßsupport provided by Calcul Québec (www.calculquebec.ca/) and Compute Canada (www.computecanada.ca). A A-G acknowledges generous support from the Canada 150 Research Chair Program, Tata Steel, Anders G Froseth, and the Office of Naval Research.

ORCID iDs

Daniel Flam-Shepherd https://orcid.org/0000-0002-9568-3451
Alan Aspuru-Guzik https://orcid.org/0000-0002-8277-4434

References

Battaglia P *et al* 2016 Interaction networks for learning about objects, relations and physics *Advances in Neural Information Processing Systems* pp 4502–10

Bowman S R, Vilnis L, Vinyals O, Dai A, Jozefowicz R and Bengio S 2016 Generating sentences from a continuous space *Proc. 20th SIGNLL Conf. on Computational Natural Language Learning*

Chen Xi, Duan Y, Houthooft R, Schulman J, Sutskever I and Abbeel P 2016 InfoGAN: interpretable representation learning by information maximizing generative adversarial nets *Neural Information Processing Systems (NIPS)*

Chung J, Gulcehre C, Cho K and Bengio Y 2014 Empirical evaluation of gated recurrent neural networks on sequence modeling (arXiv:1412.3555)

Dai H, Tian Y, Dai B, Skiena S and Song L 2018 Syntax-directed variational autoencoder for structured data (arXiv:1802.08786)

De Cao N and Kipf T 2018 MolGAN: an implicit generative model for small molecular graphs (arXiv:1805.11973)

Devlin J, Chang M-W, Lee K and Toutanova K 2019 Bert: pre-training of deep bidirectional transformers for language understanding *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies* vol 1 (Long and Short Papers) pp 4171–86

Dinh L, Krueger D and Bengio Y 2014 NICE: non-linear independent components estimation (arXiv:1410.8516)

Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A and Adams R P 2015 Convolutional networks on graphs for learning molecular fingerprints *Neural Information Processing Systems*

Erdos P and Rényi A 1960 On the evolution of random graphs *Publ. Math. Inst. Hung. Acad. Sci* **5** 17–60

Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 Neural message passing for quantum chemistry *Proc. 34th Int. Conf. on Machine Learning* vol 70 (JMLR.org) pp 1263–72

Gómez-Bombarelli R *et al* 2016 Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach *Nat. Mater.* **15** 1120–7

Gómez-Bombarelli R *et al* 2018 Automatic chemical design using a data-driven continuous representation of molecules *ACS Cent. Sci.* **4** 268–76

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 Generative adversarial nets *Advances in Neural Information Processing Systems* pp 2672–80

Grover A, Zweig A and Ermon S 2018 Graphite: iterative generative modeling of graphs (arXiv:1803.10459)

Johnson D D 2016 Learning graphical state transitions *3rd Int. Conf. on Learning Representations*

Kingma D P and Ba J 2014 Adam: a method for stochastic optimization *1st Int. Conf. on Learning Representations*

Kingma D P and Dhariwal P 2018 Glow: generative flow with invertible 1 × 1 convolutions *NeurIPS*

Kingma D P and Welling M 2014 Auto-encoding variational Bayes *1st Int. Conf. on Learning Representations*

Kipf T N and Welling M 2016 Semi-supervised classification with graph convolutional networks (arXiv:1609.02907)

Krenn M, Häse F, Nigam A, Friederich P and Aspuru-Guzik Aan 2019 SELFIES: a robust representation of semantically constrained graphs with an example application in chemistry (arXiv:1905.13741)

Kusner M J, Paige B and Hernández-Lobato J M 2017 Grammar variational autoencoder *Proc. 34th Int. Conf. on Machine Learning* vol 70 (JMLR.org) pp 1945–54

Landrum G 2013 *RDKit: A Software Suite for Cheminformatics, Computational Chemistry and Predictive Modeling* (New York: Academic)

Li Y, Tarlow D, Brockschmidt M and Zemel R 2015 Gated graph sequence neural networks (arXiv:1511.05493)

Li Y, Vinyals O, Dyer C, Pascanu R and Battaglia P 2018 Learning deep generative models of graphs (arXiv:1803.03324)

Liao R, Li Y, Song Y, Wang S, Hamilton W, Duvenaud D K, Urtasun R and Zemel R 2019 Efficient graph generation with graph recurrent attention networks *Advances in Neural Information Processing Systems* pp 4257–67

Liu Q, Allamanis M, Brockschmidt M and Gaunt A 2018 Constrained graph variational autoencoders for molecule design *Advances in Neural Information Processing Systems* pp 7795–804

Ma T, Chen J and Xiao C 2018 Constrained generation of semantically valid graphs via regularizing variational autoencoders *Advances in Neural Information Processing Systems* pp 7113–24

O'Boyle N M Towards a universal SMILES representation—a standard method to generate canonical SMILES based on the InChI 2012 *J. Cheminformatics* **4** 1–14

Radford A, Metz L and Chintala S 2015 Unsupervised representation learning with deep convolutional generative adversarial networks (arXiv:1511.06434)

Ramakrishnan R, Hartmann M, Tapavicza E and Von Lilienfeld O A 2015 Electronic spectra from TDDFT and machine learning in chemical space *J. Chem. Phys.* **143** 084111

Ruddigkeit L, Van Deursen R, Blum L C and Reymond J-L 2012 Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17 *J. Chem. Inf. Model.* **52** 2864–75

Scarselli F, Gori M, Tsoi A C, Hagenbuchner M and Monfardini G 2008 The graph neural network model *IEEE Trans. Neural Netw.* **20** 61–80

Seff A, Zhou W, Damani F, Doyle A and Adams R P 2019 Discrete object generation with reversible inductive construction *Advances in Neural Information Processing Systems* pp 10353–63

Simonovsky M and Komodakis N 2018 GraphVAE: towards generation of small graphs using variational autoencoders *Int. Conf. on Artificial Neural Networks* (Springer) pp 412–22

Sohn K, Lee H and Yan X 2015 Learning structured output representation using deep conditional generative models *Advances in Neural Information Processing Systems* pp 3483–91

Sutskever I, Martens J and Hinton G E 2011 Generating text with recurrent neural networks *Proc. 28th Int. Conf. on Machine Learning (ICML-11)* pp 1017–24

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L and Polosukhin I Attention is all you need *NIPS* p 2017

Veličković P, Cucurull G, Casanova A, Romero A, Lio P and Bengio Y 2017 Graph attention networks (arXiv:1710.10903)

Vinyals O, Bengio S and Kudlur M 2015 Order matters: sequence to sequence for sets (arXiv:1511.06391)

You J, Liu B, Ying Z, Pande V and Leskovec J 2018a Graph convolutional policy network for goal-directed molecular graph generation *Advances in Neural Information Processing Systems* pp 6410–21

You J, Ying R, Ren X, Hamilton W L and Leskovec J 2018b GraphRNN: generating realistic graphs with deep auto-regressive models (arXiv:1802.08773)

Zhavoronkov A *et al* 2019 Deep learning enables rapid identification of potent DDR1 kinase inhibitors *Nat. Biotechnol.* **37** 1038–40