GRAPH ENERGY-BASED MODEL FOR MOLECULAR GRAPH GENERATION

Ryuichiro Hataya[⋆,†], **Hideki Nakayama**[⋆], **& Kazuki Yoshizoe**[†]
[⋆] The University of Tokyo
[†] RIKEN AIP
hataya@nlab.ci.i.u-tokyo.ac.jp

ABSTRACT

We present Graph Energy-based Model (GEM), an energy-based model for molecular graph generation. GEM uses dequantization and gradient symmetrization to incorporate generation by stochastic gradient Langevin dynamics for graph representation that is discrete and includes symmetric constraint. Experimental results show that GEM can comparably design compounds as other deep generative approaches.

1 INTRODUCTION

Discovering novel molecules is important but costly and time-consuming. Machine learning-based novel molecule generation approaches are expected to remedy this problem. Specifically, recent approaches use deep generative models, such as GANs (De Cao & Kipf (2018); Maziarka et al. (2020)), VAEs (Simonovsky & Komodakis (2018); Jin et al. (2018)) and normalizing flows (Kaushalya et al. (2019); Zang & Wang (2020)), to produce graph representation of compounds.

This paper introduces Graph Energy-based Model (GEM), which uses another generative model, namely energy-based model (EBM) for molecular graph generation. We empirically demonstrate that GEM can generate novel molecules as other approaches. Additionally, its generation in the input space enables preserving specified substructures, which distinguishes GEM from different deep generative approaches.

2 GRAPH ENERGY-BASED MODELS

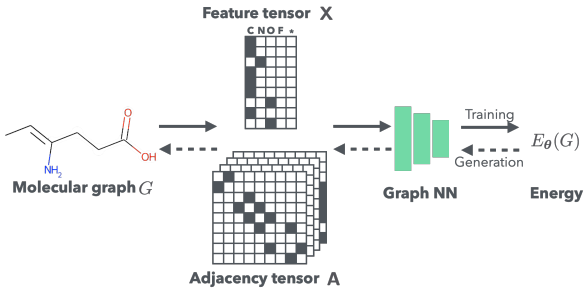

Figure 1: During training, GEM learns to assign lower energy to molecules in a dataset and higher energy to generated ones including invalid graphs. For property-targeted generation, molecules with desired properties are expected to have lower energy. GEM generates molecular graphs with lower energy using MCMC, which are expected to be valid molecules.

2.1 NOTATIONS

A molecular graph G can be represented as an undirected graph depicted by a pair of tensors: a feature tensor $\mathbf{X} \in \{0,1\}^{N \times \#\mathcal{M}}$ and an adjacency tensor $\mathbf{A} \in \{0,1\}^{N \times N \times \#\mathcal{B}}$. The

feature tensor $\mathbf{X}$ represents atoms in the molecule, and the adjacency tensor represents bonds among them. N is the maximum number of atoms in molecules in a dataset, $\mathcal{M}$ is a set of considered atoms, *e.g.*, $\mathcal{M} = \{\mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{F}, \text{virtual node}\}$, and $\mathcal{B}$ is the set of bond types, *i.e.*, $\mathcal{B} = \{\text{single}, \text{double}, \text{triple}, \text{virtual bond}\}$. "virtual node" and "virtual bond" are used for padding in case the number of atoms in a given molecule is smaller than N.

For each triplet of $(N, \mathcal{M}, \mathcal{B})$, there is a set of valid molecular graphs $\mathcal{G} = \mathcal{G}_{(N,\mathcal{M},\mathcal{B})}$. Validity includes the symmetry of adjacency tensor slices: $\mathbf{A}_{:,:,b}$ is a symmetric matrix for $b = 1, 2, \ldots, \#\mathcal{B}$. Practically, we use datasets $\mathcal{D} \subset \mathcal{G}$.

2.2 Generating Graphs by EBMs

We propose to generate a novel molecule by using an energy function $E_{\boldsymbol{\theta}} : \mathcal{G} \to \mathbb{R}$, parameterized by a real vector $\boldsymbol{\theta}$. Specifically, we use a graph neural network to represent this parameterized function. The energy function is expected to assign smaller values to valid molecules and higher values to invalid ones. This energy function determines a Boltzmann-Gibbs distribution $p_{\boldsymbol{\theta}}(G) = \frac{\exp(-E_{\boldsymbol{\theta}}(G))}{\sum_{G' \sim \mathcal{G}} \exp(-E_{\boldsymbol{\theta}}(G))}$, from which molecules are expected to be sampled with a high probability. If graphs are continuous, we can sample graphs from this distribution by using stochastic gradient Langevin dynamics (SGLD, Welling & Teh (2011)):

$$\mathbf{X}^{(t+1)} = \mathbf{X}^{(t)} + \frac{\alpha_t}{2} g_{\mathbf{X}}(\mathbf{X}^{(t)}, \mathbf{A}^{(t)}) + \sqrt{\alpha_t}\boldsymbol{\epsilon}_{\mathbf{X}}, \mathbf{A}^{(t+1)} \quad = \mathbf{A}^{(t)} + \frac{\alpha_t}{2} g_{\mathbf{A}}(\mathbf{X}^{(t)}, \mathbf{A}^{(t)}) + \sqrt{\alpha_t}\boldsymbol{\epsilon}_{\mathbf{A}}, \quad (1)$$

where $\alpha_t \in \mathbb{R}^+$ is a step size, $g_{\mathbf{X}} = \nabla_{\mathbf{X}} E_{\boldsymbol{\theta}}$ and $g_{\mathbf{A}} = \nabla_{\mathbf{A}} E_{\boldsymbol{\theta}}$ are score functions, and $\boldsymbol{\epsilon}_{\mathbf{X}}$ and $\boldsymbol{\epsilon}_{\mathbf{A}}$ are standard normals. This generation (Equation (1)) can also be achieved by directly estimating $g_{\mathbf{X}}$ and $g_{\mathbf{A}}$ as (Niu et al. (2020)). $\mathbf{X}^{(0)}$ and $\mathbf{A}^{(0)}$ are sampled from a uniform distribution on $[0, 1]$. The distribution of $G^{(\infty)} = (\mathbf{X}^{(\infty)}, \mathbf{A}^{(\infty)})$ is asymptotically equal to $p_{\boldsymbol{\theta}}(G)$, and we assume that this property can be approximated with finite steps with a small constant state size, *i.e.*, $\alpha_t = \alpha$, following the literature.

Actually, simply applying Equation (1) does not work in our case, because they do not consider the following requirements: 1. $\mathbf{X}$ and $\mathbf{A}$ are discrete, and 2. slices of $\mathbf{A}$ is symmetric. To fix the first issue, we relax the domains of $\mathbf{X}$ and $\mathbf{A}$ to be $(0, 1)^{N \times \#\mathcal{M}}$ and $(0, 1)^{N \times N \times \#\mathcal{B}}$. For discrete tensors from datasets, we modify them by using *dequantization* and applying softmax function along the last axes. Dequantization is a technique used in Kaushalya et al. (2019), which adds random values to the tensor elements $\mathbf{X} \leftarrow \mathbf{X} + c\mathbf{U}_{\mathbf{X}}$, $\mathbf{A} \leftarrow \mathbf{A} + c\mathbf{U}_{\mathbf{A}}$, where $c \in (0, 1)$ is a scaling parameter, and $\mathbf{U}_{\mathbf{X}}, \mathbf{U}_{\mathbf{A}}$ are uniform noise on $(0, 1)$. We set $c = 0.9$ in the experiments.

To avoid sampled adjacency tensors being asymmetric, we sample $\mathbf{A}^{(0)}$ and $\boldsymbol{\epsilon}_{\mathbf{A}}$ from symmetric distributions, where $(\mathbf{A}^{(0)})_{i,j,b} = (\mathbf{A}^{(0)})_{j,i,b}$, $(\boldsymbol{\epsilon}_{\mathbf{A}})_{i,j,b} = (\boldsymbol{\epsilon}_{\mathbf{A}})_{j,i,b}$, for $i, j \in \{1, 2, \ldots, N\}$ and $b \in \{1, 2 \ldots, \#\mathcal{B}\}$. Additionally, the score function $g_{\mathbf{A}}$ needs to be symmetric, which we will describe in the next section.

2.3 Symmetrize Gradient of Adjacency Tensor

We use a neural network based on Relational GCN (RGCN, (Schlichtkrull et al. (2018))) as an energy function. RGCN is a graph convolutional neural network for graphs with multiple edge types. For each graph $G = (\mathbf{X}, \mathbf{A})$, the lth RGCN layer processes node representation $\boldsymbol{H}_l \in \mathbb{R}^{N \times C}$ as

$$\boldsymbol{H}_{l+1} = \sigma\left(\boldsymbol{H}_l W_l^{(0)} + \sum_{b=1}^{\#\mathcal{B}} \mathbf{A}_{:,:,b} \boldsymbol{H}_l \boldsymbol{W}_l^{(b)}\right), \quad (2)$$

where $\boldsymbol{H}_0 = \mathbf{X}$, $\boldsymbol{W}_l^{(0)}, \boldsymbol{W}_l^{(b)} \in \mathbb{R}^{C \times D}$ are learnable parameters, σ is a nonlinear activation function, and C, D are input and output feature dimensions. After several RGCN layers, a graph-level

representation is obtained by the aggregation of (Li et al. (2016)). This representation is transformed into a scalar value $E_{\boldsymbol{\theta}}(G)$ by a multi layer perceptron.

Crucially, with this energy function, the score function $g_{\mathbf{A}}$ is asymmetric. Indeed, by focusing on the first layer of RGCN layers and ignoring the nonlinear activation for simplicity, we obtain a Jacobian tensor of $\dfrac{\partial \boldsymbol{H}_1}{\partial(\mathbf{A})_{i,j,b}} = \dfrac{\partial \mathbf{A}_{:,:,b}\mathbf{X}\boldsymbol{W}_b^{(1)}}{\partial(\mathbf{A})_{i,j,b}} = \boldsymbol{J}^{(i,j)}\mathbf{X}\boldsymbol{W}_b^{(1)}$, where $\boldsymbol{J}^{(i,j)}$ denotes a single entry matrix of 1 at (i,j) and 0 elsewhere (Petersen & Pedersen (2006)). This gradient is not symmetric for each b. To remedy this, we modify Equation (2) as

$$\boldsymbol{H}_{l+1} = \sigma\left(\boldsymbol{H}_l\boldsymbol{W}_l^{(0)} + \sum_{b=1}^{\#\mathcal{B}} \frac{1}{2}(\mathbf{A}_{:,:,b} + \mathbf{A}_{:,:,b}^{\top})\boldsymbol{H}_l\boldsymbol{W}_l^{(b)}\right). \tag{3}$$

Though this modification does not change the output because each $\mathbf{A}_{:,:,b}$ is symmetric by definition, now the Jacobian tensor is also symmetrized as $\dfrac{\partial \boldsymbol{H}_1}{\partial(\mathbf{A})_{i,j,b}} = \dfrac{1}{2}(\boldsymbol{J}^{(i,j)} + \boldsymbol{J}^{(j,i)})\mathbf{X}\boldsymbol{W}_b^{(1)}$, from which we can deduce $\dfrac{\partial E_{\boldsymbol{\theta}}}{\partial(\mathbf{A})_{i,j,b}} = \dfrac{\partial E_{\boldsymbol{\theta}}}{\partial(\mathbf{A})_{j,i,b}}$, the symmetry of the score function $g_{\mathbf{A}}$. Practically, the modification of Equation (3) can be separately done before the forward pass of the model, which means the actual modification to the off-the-shelf models is minimum. In the experiments, we use the abovementioned RGCN variant, which is also used in other graph-based molecular generation methods (De Cao & Kipf (2018); Kaushalya et al. (2019)).

2.4 Training of GEM

To optimize the energy function $E_{\boldsymbol{\theta}}$, we can use stochastic gradient of $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\mathcal{D}}\left[\log p_{\boldsymbol{\theta}}(G)\right] = \mathbb{E}_{\mathcal{D}}\left[\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(G)\right] - \mathbb{E}_{p_{\boldsymbol{\theta}}(G')}\left[\nabla_{\boldsymbol{\theta}}E_{\boldsymbol{\theta}}(G')\right]$. At the LHS's second term, samples from the model $G' \sim p_{\boldsymbol{\theta}}(G')$ are used. As discussed in Section 2.2, we use a finite step of SGLD to approximate this sampling, resulting in diverged samples from the model distribution. To remedy this problem, we use the persistent contrastive divergence (PCD, Tieleman (2008)), which reuses the past generated samples. Additionally, we penalized $\{E_{\boldsymbol{\theta}}(G)\}^2$ (Du & Mordatch, 2019).

2.5 Generation by GEM

GEM generates molecular graphs using SGLD (Equation (1)), which adds noise to graph representation, and thus, sometimes collapses its validity. To remedy this issue, we apply validity correction (Zang & Wang (2020)) to feature and adjacency tensors after generation steps.

One of the most appealing ability of GEM is substructure preserving generation. Because GEM samples molecular graphs in the input space by SGLD, this ability is achieved by updating parts of graph representation (see also Figure 2 and Appendix A).

3 Experiments

3.1 Experimental Settings

We used QM9 (Wu et al. (2018)) and ZINC-250k (Irwin & Shoichet (2015)) as datasets $\mathcal{D}$. QM9 and ZINC-250k contain 1.3×10^5 and 2.5×10^5 molecules, respectively. Following the preprocessing protocols in Kaushalya et al. (2019), we kekulize each molecule in each dataset and ignore hydrogens as the SMILES format. As a result, the maximum number of atoms in a molecule N is 9 for QM9 9 and 38 for ZINC-250k. The number of atom types $\#\mathcal{M}$ including the virtual node is 5 for QM9 and 10 for ZINC-250k. The number of bond types $\#\mathcal{B}$ is 4, namely $\mathcal{B} = \{\text{single}, \text{double}, \text{triple}, \text{virtual bond}\}$, for both datasets. We also followed the data split of Kaushalya et al. (2019).

Each input feature tensor is embedded in 16-dimensional space and processed by a two-layer RGCN of 128 hidden dimensions. Its output is aggregated in a 256-dimensional space and converted to

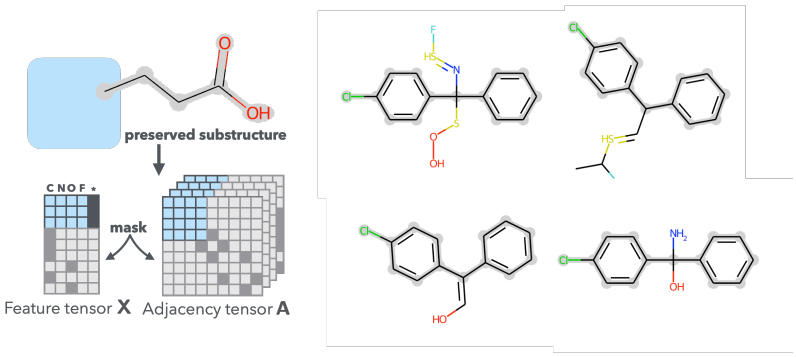

Figure 2: **Left:** GEM can generate molecular graphs while preserving specified substructures (highlighted by light gray) by applying masks to the corresponding parts and only updating the rest (highlighted by light blue). **Right:** Examples of substructure generation while fixing 4-Chlorodiphenylmethane `c1ccc(cc1)Cc2ccc(cc2)Cl` highlighted by light gray.

scalar energy by an MLP of $(1024, 512)$ hidden units. The hyperbolic tangent function is used as an activation function, and the sigmoid function $\varsigma(x) = \{1+\exp(-x)\}^{-1}$ is applied to the final output that restricts the range to $[0, 1]$. Please refer to Appendix B for details of training and generation.

3.2 Results

We present validity, novelty, and uniqueness of molecular graphs generated from 1,000 random initial states in Table 1 compared with baselines of MoFlow (Zang & Wang (2020)), GraphNVP (Kaushalya et al. (2019)), MolGAN (De Cao & Kipf (2018)), and RVAE (Ma et al. (2018)). GEM shows comparable performance with other methods using other deep generative methods on both datasets.

Additionally, samples of substructure preserving generation are presented in Figure 2. GEM can exactly fix specified substructures, which distinguishes our approach from different generative methods. Other samples are presented in Figure 3 of Appendix A.

Table 1: The results of non substructure-preserving molecular graph generation. Baseline results are borrowed from the original papers. For GEM, average and standard deviation of three runs are reported. * indicates the use of validity correction (Zang & Wang (2020)).

Dataset	QM9			ZINC-250k		
Method	Validity	Novelty	Uniqueness	Validity	Novelty	Uniqueness
GEM *	100 ± 0.0	99.1 ± 0.2	85.9 ± 0.6	100 ± 0.0	100 ± 0.0	100 ± 0.0
MoFlow*	100 ± 0.0	98.0 ± 0.1	99.2 ± 0.1	100 ± 0.0	100 ± 0.0	100 ± 0.0
GraphNVP	83.1 ± 0.5	58.2 ± 1.9	99.2 ± 0.3	42.6 ± 1.6	100 ± 0.0	94.8 ± 0.6
MolGAN	98.1	94.2	10.4	N/A	N/A	N/A
RVAE	96.6	97.5	N/A	34.9	100	N/A

4 Conclusion

In this paper, we have proposed GEM, an EBM for molecular graphs. Dequantization and gradient symmetrization have been introduced to generate discrete and symmetric representations of graphs in continuous space. We empirically demonstrate the effectiveness of GEM and its unique ability, substructure preserving generation. We hope energy-based molecular generation, including GEM, opens a new direction of de novo design.

REFERENCES

Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. In *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy-based models. In *NeurIPS*, 2019.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv:2012.01316*, 2020.

Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *ICLR*, 2020.

John J Irwin and Brian K Shoichet. ZINC – a free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, 45, 2015.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.

Madhawa Kaushalya, Ishiguro Katushiko, Nakago Kosuke, and Abe Motoki. Graphnvp: An invertible flow model for generating molecular graphs. In *NeurIPS*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS*, 2018.

Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and MichałWarchoł. Mol-CycleGAN: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(2), 2020.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019.

Kaare Petersen and Michael Pedersen. The matrix cookbook. 2006.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 2018.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. *arXiv:1802.03480*, 2018.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *ICML*, 2008.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9:513–530, 2018.

Chengxi Zang and Fei Wang. Moflow: An invertible flow model for generating molecular graphs. In *KDD*, 2020.

A SUBSTRUCTURE PRESERVING GENERATION

To fix substructures, we apply masks to both feature and adjacency tensors. Suppose the number of atoms in a given substructure is $S < N$, where N is the maximum number of atoms in molecules of a dataset. Because `GEM` is permutation invariant to an input representation, we can re-index atoms in the substructure to $1, 2, \ldots, S$ such that the Sth atom to be connected with the rest part, without loss of generality. Then, we use a mask to update only a part of the feature tensor corresponding to $S + 1, S + 2, \ldots, N$th atoms and fix the atoms in the substructure. Similarly, we only update connections among $S, S + 1, \ldots, N$th atoms and fix the connections among the rests by masking the adjacency tensor. This masking can be extended to appending the rest parts to multiple atoms.

B EXPERIMENTS

B.1 DETAILS OF TRAINING AND GENERATION

We used PyTorch v1.7 (Paszke et al. (2019)) for model implementation, chainer-chemistry v0.7 [1] for data preprocessing, and RDKit v2020.09 [2] for handling molecule information.

We trained `GEM` using Adam (Kingma & Ba (2015)) with a learning rate of 1.0×10^{-4} for 30 epochs. For SGLD, we set a step size α to 1.0×10^{-4} and the number of steps to 40. Following Grathwohl et al. (2020); Du et al. (2020), we set the buffer size of PCD to 10^4 and the reinitialization probability ρ to 5.0×10^{-2}, and reduced the effect of additive noise by multiplying 0.1 to the standard deviation as common practice. For SGLD, we used an exponential moving average of the model with a decay rate of 1.0×10^{-3} for the stability.

To generate molecular graphs, we used SGLD of step size of 1.0×10^{-1} for `QM9` and 1.0×10^{-2} for `ZINC-250k`, and the number of steps of 10^3. Adding noise in Equation (1) sometimes turns once generated valid molecular graphs into invalid ones. Therefore, we record all valid graphs generated at each step. We discarded the graphs generated during the first 100 steps to reduce the effects of initial states.

[1] `https://github.com/chainer/chainer-chemistry`

[2] `https://www.rdkit.org`

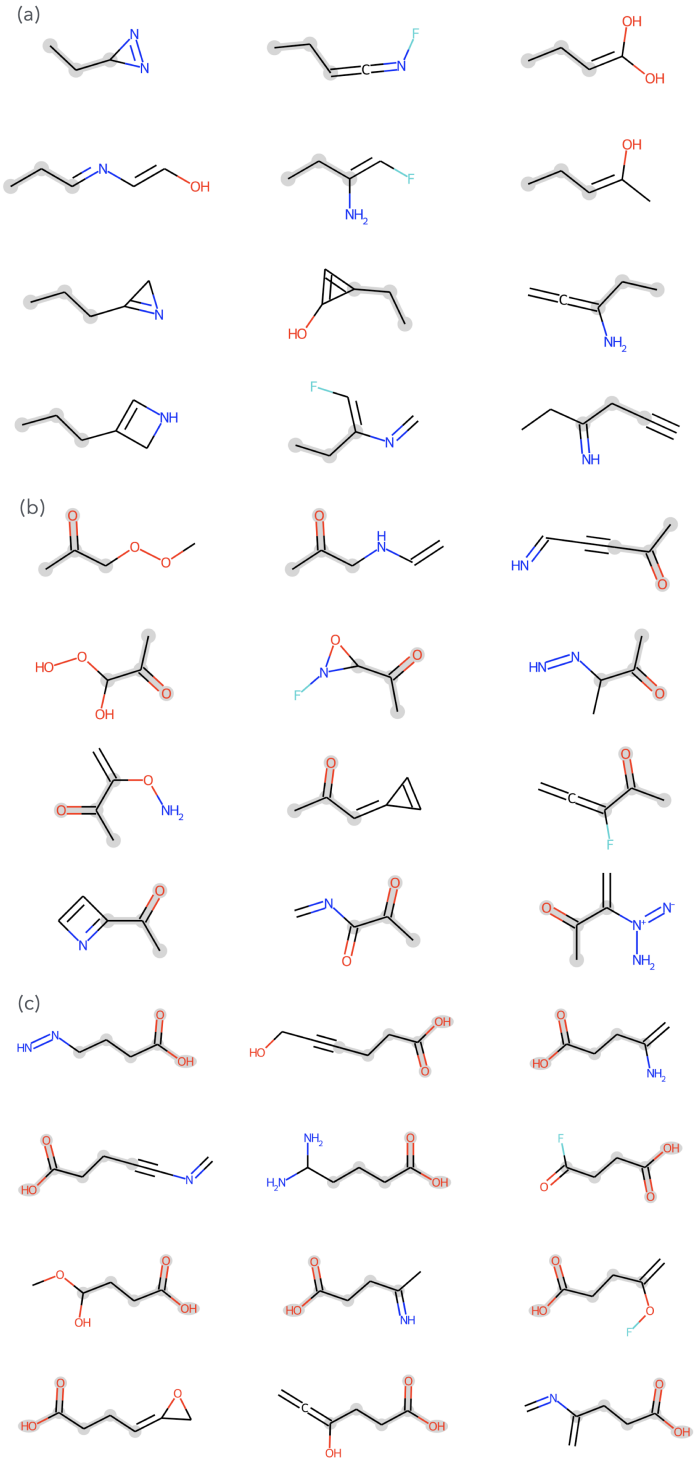

Figure 3: Randomly selected molecules of substructure preserving generation. Conditioned substructures are (a) propane `CCC`, (b) acetone `CC(=O)C`, and (c) butanoic acid `CCCC(=O)O`, which are highlighted by light gray.