Molecule Edit Graph Attention Network: Modeling Chemical Reactions as Sequences of Graph Edits

Mikołaj Sacha [1] **Mikołaj Błaż** [1] **Piotr Byrski** [1] **Paweł Włodarczyk-Pruszyński** [1] **Stanisław Jastrzębski** [1 2]

Abstract

One of the key challenges in automated chemical synthesis planning is to propose diverse and reliable reactions. A common approach is to generate reactions using *reaction templates*, which represent a reaction as a fixed graph transformation. This enables accurate and interpretable predictions but can suffer from limited diversity. On the other hand, *template-free* methods increase diversity but can be prone to making trivial mistakes. Inspired by the efficacy of reaction templates, we propose Molecule Edit Graph Attention Network (MEGAN), a template-free model that encodes reaction as a sequence of graph edits. Our model achieves state-of-the-art results on a standard retrosynthesis benchmark without any manual rule encoding.

1. Introduction

Chemical synthesis planning is a demanding task due to the substantial size and complexity of the reaction space. Computer-aided methods are a promising approach to automating this process (Corey & Wipke, 1969; Segler et al., 2018; Coley et al., 2018a; Lee et al., 2019).

Generating reaction candidates is an essential part of automated synthesis planning. The accuracy with which we are able to predict reaction outcomes remains a key roadblock in wider applicability of computer-aided methods (Davies, 2019).

Many of the recent approaches to reaction generation use *reaction templates*, which are encoded graph transformation rules that enable generating reactions by applying them to the input molecule. Such methods are highly interpretable and achieve strong performance (Dai et al., 2020). However, template-based methods have some disadvantages. Perhaps the most pressing one is that due to the computational limits they require modeling reactions using a relatively small number of templates. This necessarily limits the size of the chemical reaction space accessible by such methods.

The shortcomings of template-based methods have been addressed with the development of template-free models for reaction generation. In particular, modeling reaction generation as a machine translation task has lead to promising results (Liu et al., 2017). This and related methods are still however often outperformed by template-based methods (Dai et al., 2020), and have been also argued to be prone to making trivial mistakes (Molga et al., 2019).

Arguably, a more natural approach to reaction generation is to represent reaction as a sequence of graph edits. Bradshaw et al. (2018) models reaction as a sequence of bond removals and additions. However, their approach is limited to a certain subset of the chemical reaction space (reactions with linear chain topology) and forward synthesis (predicting product of a reaction). Do et al. (2019) models reaction as a set of operation on atom pairs. Similarly, their method cannot be readily applied to retrosynthesis (predicting substrates based on the product) due to the lack of support for atom addition.

In this work, we present the Molecule Edit Graph Attention Network (MEGAN). We propose an encoder-decoder model that generates a reaction as a sequence of graph edits. We include atom addition and bond removal in the action space to apply the model to the retrosynthesis task, where we achieve state-of-the-art results. Specifically, our main contributions are as follows:

- We propose a novel graph encoder-decoder architecture for reaction generation.
- We achieve state-of-the-art results in retrosynthesis.
- We demonstrate the feasibility of representing reaction as a sequence of graph edits for retrosynthesis, which is a promising step towards more interpretable neural models for reaction generation.

[1]Molecule.one [2]Department of Computer Science, New York Univesity, New York, United States. Correspondence to: Mikołaj Sacha <mikolajsacha@gmail.com>, Stanisław Jastrzębski <staszek.jastrzebski@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

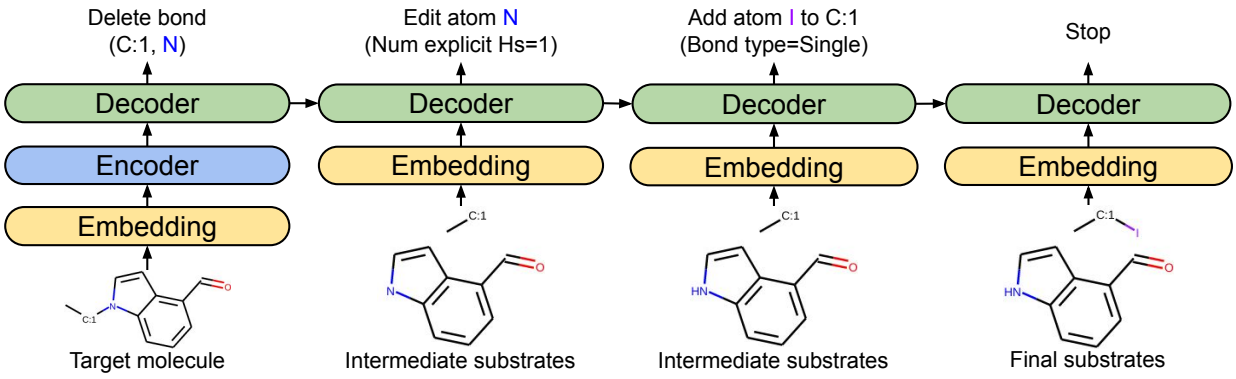

Figure 1. Retrosynthesis prediction generated by MEGAN. The model modifies the target molecule by sequentially executing actions on the molecular graph until it decides to stop.

2. Molecule Edit Graph Attention Network

Graph Convolutional Networks (GCN) have been successfully applied to a variety of tasks in computational chemistry (Duvenaud et al., 2015; Xie & Grossman, 2018; You et al., 2018a; Coley et al., 2018b). Our Molecule Edit Graph Attention Network (MEGAN) is an encoder-decoder architecture based on GCN that is able to predict a sequential series of actions on atoms and bonds of a chemical compound. The key innovation is to combine an effective architecture and training procedure, which enables us to extend ideas from Bradshaw et al. (2018); Do et al. (2019) to retrosynthesis and achieve state of the art performance. We begin by describing the input and the output representation. Next, we introduce the overall architecture.

2.1. Input and output representation

Reaction as a sequence of edits We reformulate retrosynthesis as predicting a series of actions on the graph of the product molecule that produce the reactants. We define the following graph actions:

- Edit atom properties *(EditAtom)*
- Edit bond between two atoms *(EditBond)*
- Add new atom to the graph *(AddAtom)*
- Add new benzene ring to the graph *(AddBenzene)*
- Stop generation *(Stop)*

EditAtom changes properties of atoms, such as the formal charge, chirality or aromaticity. *EditBond* adds, edits or deletes a bond between two atoms. *AddAtom* adds a new atom of a specified type as a neighbor of another atom already existing in the graph, with a specified bond type. *AddBenzene* optimizes atom addition by appending a complete benzene ring to a selected carbon atom. *Stop* action indicates the end of the generation process. We use atom mapping information to define an order on actions, which we describe in the experimental section. We describe the possible actions in details in the Supplement.

Molecule representation MEGAN takes as input a molecular graph, which is represented by labeled node vectors and a labeled adjacency matrix. More specifically, the input consists of a matrix of features $H^{OH} \in {\mathbb{Z}_{\geq 0}}^{n \times h_{OH}}$ and an adjacency matrix $A^{OH} \in {\mathbb{Z}_{\geq 0}}^{n \times n \times a_{OH}}$, where n is the number of nodes in a graph and h_{OH} and a_{OH} are sizes of concatenated one-hot vectors of atom and bond features. Hydrogen atoms are removed from the graph, as for heavy atoms the number of neighboring hydrogen atoms can be deduced implicitly or marked explicitly by a special atom feature if needed. We selected a minimal set of atom and bond features that allows for exact reconstruction of SMILES of all products and reactants from the development set. We describe the featurization in details in the Supplement.

2.2. Model architecture

Input embedding Input features are embedded using linear layers $f_{emb} \colon \mathbb{R}^{h_{OH}} \to \mathbb{R}^{h}$ and $g_{emb} \colon \mathbb{R}^{a_{OH}} \to \mathbb{R}^{a}$:

$$\mathbb{R}^{n \times h} \ni H^0 = f_{emb}(H^{OH}) \tag{1}$$

$$\mathbb{R}^{n \times n \times a} \ni A = g_{emb}(A^{OH}) \tag{2}$$

GCN-att layer The basic building block of the network is an attention-based GCN layer named *GCN-att*. We enhance the GCN layer from Veličković et al. (2017) by adding bond features as input information for computing the attention values. Let $H^t \in \mathbb{R}^{n \times h}$ denote input node features for the t-th GCN-att layer and $N(i) \subset \mathbb{Z}_{\geq 0}$ denote set of indices of neighbors of node at index i (where $i \in N(i)$). We calculate new node features $H^{t+1} \in \mathbb{R}^{n \times h}$ as follows:

$$\mathbb{R}^d \ni H_i^{t'} = \sigma_r(f_{att}^t(H_i^t)) \tag{3}$$

$$\mathbb{R}^{2d+a} \ni B_{ij}^t = H_i^{t'} \parallel H_j^{t'} \parallel A_{i,j} \tag{4}$$

$$\mathbb{R}^K \ni C_{ij}^t = f_{att'}^t(B_{ij}^t) \tag{5}$$

$$\mathbb{R}^h \ni G_{ik}^t = \sum_{j \in N(i)} \frac{\exp C_{ijk}^t}{\sum_{l \in N(i)} \exp C_{ilk}^t} H_j^t \tag{6}$$

$$\mathbb{R}^h \ni H_i^{t+1} = \mathop{\Vert}_{1 \le k \le K} \sigma_r(f_k^t(G_i^t)) \tag{7}$$

where σ_r denotes *relu* activation function, $\parallel$ indicates vector concatenation, $K \in \mathbb{N}_+$ is the number of attention heads and $f_{att}\colon \mathbb{R}^h \to \mathbb{R}^d$, $f_{att'}\colon \mathbb{R}^{2d+a} \to \mathbb{R}^K$ and $f\colon \mathbb{R}^h \to \mathbb{R}^{h/K}$ are standard linear layers. Numbers h, a, d and K are hyperparameters of the model. We require that h is divisible by K. We use the same hyperparameter values for all GCN-att layers in the model, but do not share their weights.

Supernode A single pass through a GCN-att layer transfers information only between neighboring atoms. This can potentially hinder the ability to learn graph-level features, such as coexistence of functional groups in different parts of a compound. To mitigate this, we introduce an additional node named *supernode* (Li et al., 2017), which is connected to all atoms in the graph with a special SUPERNODE bond type. Supernode is particularly useful for passing information between connected components of a graph after it was split by deleting a bond.

Overall architecture The model consists of two parts: the *encoder*, which is invoked only once per reaction generation and the *decoder*, which is sequentially invoked to generate actions (Figure 1). At each generation step, features are embedded using layers f_{emb} and g_{emb}. The encoder has N_e stacked GCN-att layers, which are invoked at the first generation step after the embedding layers. The decoder has N_d stacked GCN-att layers followed by final linear layers that output action probabilites for atom and bond actions as follows. Let $m = N_e + N_d$ for the first generation step and $m = N_d$ for the other generation steps. The logits L^a and L^b for atom actions Act_a and bond actions Act_b are calculated as follows:

$$\mathbb{R}^d \ni F_i = \sigma_r(g_{atom}(H_i^m)) \tag{8}$$

$$\mathbb{R}^{|Act_a|} \ni L_i^a = g'_{atom}(F_i) \tag{9}$$

$$\mathbb{R}^d \ni J_i = \sigma_r(g_{bond}(H_i^m)) \tag{10}$$

$$\mathbb{R}^{d+a} \ni J'_{ij} = \sigma_r(J_i + J_j \parallel A_{i,j}) \tag{11}$$

$$\mathbb{R}^{|Act_b|} \ni L_{ij}^b = g'_{bond}(J'_{ij}) \tag{12}$$

We reuse the hyperparameter value d for simplicity. To acquire the final action probabilties, we apply softmax activation function to concatenated vectors of logits of all possible atom actions Act_a and possible bond actions Act_b. To decide which actions are legal, we use the following rules:

- *Stop* action can be predicted only by the supernode
- All other atom actions can be predicted by all nodes except the supernode.
- Bond actions can be predicted for indices i and j, where $i < j$ and nodes at i and j are atoms

Retaining generation state Finally, we want our model to be *stateful*, that is to be able to take advantage of the information about the previous generation steps to predict the next action. We achieve this by storing the output H_s^m of the last GCN-att layer at the generation step s and merging it with the embedded input H_{s+1}^0 at step $s+1$: $\mathbb{R}^h \ni H_{s+1}^0 := \max(H_{s+1}^0, H_s^m)$, where $\max$ is the piecewise maximum of vectors. In H_s^m, we zero-pad features for any node that was added to the graph at step s.

3. Experiments

We run experiments on the standard retrosynthesis benchmark USPTO-50k (Lowe, 2012; Schneider et al., 2016). We begin by introducing the experimental setting.

3.1. Experimental setting

Tasks We evaluate the models on retrosynthesis. The goal in retrosynthesis is to predict the set of reactants based on the product of a reaction. The accuracy is measured by comparing the SMILES (Weininger, 1988) representation of the generated reactants to the SMILES representation of the ground truth reactants. Before the comparison, we remove any mapping information from the SMILES strings and canonicalize them using RdKit (Landrum). We use top K accuracy computed on the reactions from the test set as the main evaluation metric.

Data We use the USPTO-50k data set of approximately 50000 reactions, which was collected by Lowe (2012) and classified into 10 reaction types by Schneider et al. (2016). We use the same processed version of the data set as Coley et al. (2017), where each reaction consists of a single product molecule and a set of one or more reactants, with corresponding atoms between the reactants and the product mapped. Following other studies, we assign each reaction randomly to one of the training/validation/test sets with respective probabilities of 80%/10%/10%.

Gradient-based training of MEGAN In contrast to Do et al. (2019) who use reinforcement learning to train their model, we back-propagate directly through the maximum

likelihood objective to train MEGAN. This is nontrivial, as computing the gradient of the likelihood objective requires defining a fixed ordering of actions (You et al., 2018b). To solve this issue, You et al. (2018b) enumerates atoms using breadth-first search. We adapt a similar idea to reaction generation. We use the mapping provided in the USPTO-50k data set, which describes atom correspondence between the product and the substrates, to predetermine an ordering of actions. This provides supervision for each generation step and thus enables us to compute the gradient. We provide the remaining details in the Supplement.

Other training and evaluation details of MEGAN For training, we use batch size of 4 reactions. We use Adam (Kingma & Ba, 2014) with the initial learning rate of 0.0001. We use warm-up, increasing the learning rate from 0 to 0.0001 over the first 10000 training steps. For efficiency, we compute the validation loss on a subset of 2500 validation samples after each 10000 training samples. We multiply the learning rate by 0.1 if the estimated validation loss has not decreased for more than 4 such validation checks. We stop the training after the estimated validation loss has not decreased for more than 8 validation checks. We adapt the hyperparameters based on the validation loss. The final hyperparameter values are described in the Supplement.

Following other studies, we run two variants of training: one with unknown reaction type and one for which reaction type is given as a prior by an additional embedding layer. For both runs, we use the same model architecture and the same training setup. The training takes approximately 8 hours on a single Nvidia Tesla V100 GPU for both variants.

We use beam search (Graves, 2012) on output probabilities of actions to generate multiple ranked candidates for each product. We set the maximum number of steps to 16 and the beam width to 50, as it is the largest K for which accuracy was reported for the baseline models. Running predictions with the beam width of 50 on the test set of USPTO-50k (5155 test products) takes about 45 minutes on a single Nvidia Tesla V100 GPU.

Baselines We compare performance of MEGAN with several template-free and template-based models, including current state-of-the-art methods. Seq2seq (Liu et al., 2017) and Transformer (Karpov et al., 2019) are both template-free methods based on machine translation models applied on SMILES strings. G2G (Shi et al., 2020) (concurrent work) is also a template-free model based on modifying molecular graphs, with a separate module for predicting reaction center. Retrosim (Coley et al., 2017) uses reaction fingerprint to select template based on similar reactions in the data set. Neuralsym (Segler & Waller, 2017) uses a multi-linear percceptron to rank templates. GLN (Dai et al., 2020) employs a graph model that assesses when rules from templates should be applied.

Table 1. Top-k test accuracy on the USPTO-50k data set. Results of other methods taken from Dai et al. (2020) and Shi et al. (2020)

METHODS	TOP-K ACCURACY %					
	1	3	5	10	20	50
REACTION TYPE UNKNOWN						
TRANS	37.9	57.3	62.7	/	/	/
RETROSIM	37.3	54.7	63.3	74.1	82.0	85.3
NEURALSYM	44.4	65.3	72.4	78.9	82.2	83.1
G2GS	48.9	67.6	72.5	75.5	/	/
GLN	**52.5**	69.0	75.6	83.7	89.0	92.4
MEGAN	47.6	**71.2**	**79.5**	**87.0**	**91.4**	**94.2**
REACTION TYPE GIVEN AS PRIOR						
SEQ2SEQ	37.4	52.4	57.0	61.7	65.9	70.7
RETROSIM	52.9	73.8	81.2	88.1	91.8	92.9
NEURALSYM	55.3	76.0	81.4	85.1	86.5	86.9
G2GS	61.0	81.3	86.0	88.7	/	/
GLN	**64.2**	79.1	85.2	90.0	92.3	93.2
MEGAN	61.6	**83.7**	**89.2**	**93.2**	**95.3**	**96.6**

3.2. Results

Table 1 reports results on the USPTO-50k benchmark in a variant with and without reaction type information. For the top ranked prediction ($K = 1$), MEGAN is surpassed only by GLN when reaction type is given and by GLN and G2Gs when reaction type is unknown. For $K > 1$, our model achieves state-of-the-art accuracy, outperforming all baselines in both settings.

We hypothesize that the advantage of MEGAN for $K > 1$ stems largely from the fact that MEGAN generates reaction as a sequence of edits. This might help to efficiently search through different plausible reaction centers, hence covering a more diverse subset of the reaction space. It can also enable MEGAN to achieve high coverage of the reaction space, which is indicated by Top 50 accuracy of 94.2% when reaction type is unknown and 96.6% when reaction type is provided.

4. Conclusions

In this work, we presented the Molecule Edit Graph Attention Network (MEGAN), a template-free model that achieves state-of-the-art performance on retrosynthesis. The key idea is to generate reaction as a sequence of edits. We hypothesize that this enables generating more diverse predictions by a more efficient search through the space of plausible reactions.

To train our model efficiently, we used heuristically computed reaction mapping to define an order of actions. An interesting topic for the future would be to combine gradient-based training with reinforcement learning to reduce the reliance of MEGAN on mapping.

References

Bradshaw, J., Kusner, M. J., Paige, B., Segler, M. H. S., and Hernández-Lobato, J. M. A generative model for electron paths, 2018. URL https://arxiv.org/abs/1805.10970.

Coley, C. W., Rogers, L., Green, W. H., and Jensen, K. F. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 2017. doi: 10.1021/acscentsci.7b00355. URL https://doi.org/10.1021/acscentsci.7b00355. PMID: 29296663.

Coley, C. W., Green, W. H., and Jensen, K. F. Machine learning in computer-aided synthesis planning. *Accounts of Chemical Research*, 51(5):1281–1289, 2018a. doi: 10.1021/acs.accounts.8b00087. URL https://doi.org/10.1021/acs.accounts.8b00087. PMID: 29715002.

Coley, C. W., Jin, W., Rogers, L., Jamison, T. F., S Jaakkola, T., Green, W. H., Barzilay, R., and Jensen, K. F. A graph-convolutional neural network model for the prediction of chemical reactivity, Oct 2018b. URL https://chemrxiv.org/articles/A_Graph-Convolutional_Neural_Network_Model_for_the_Prediction_of_Chemical_Reactivity/7163189/1.

Corey, E. J. and Wipke, W. T. Computer-assisted design of complex organic syntheses. *Science*, 166(3902):178–192, 1969. ISSN 0036-8075. doi: 10.1126/science.166.3902.178. URL https://science.sciencemag.org/content/166/3902/178.

Dai, H., Li, C., Coley, C. W., Dai, B., and Song, L. Retrosynthesis prediction with conditional graph logic network, 2020.

Davies, I. W. The digitization of organic synthesis. *Nature*, 570(7760):175–181, 2019. doi: 10.1038/s41586-019-1288-y. URL https://doi.org/10.1038/s41586-019-1288-y.

Do, K., Tran, T., and Venkatesh, S. Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pp. 750–760, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330958. URL https://doi.org/10.1145/3292500.3330958.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2224–2232. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf.

Graves, A. Sequence transduction with recurrent neural networks, 2012.

Karpov, P., Godin, G., and Tetko, I. *A Transformer Model for Retrosynthesis*, pp. 817–830. 09 2019. ISBN 978-3-030-30492-8. doi: 10.1007/978-3-030-30493-5_78.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Landrum, G. Rdkit: Open-source cheminformatics. URL http://www.rdkit.org.

Lee, A. A., Yang, Q., Sresht, V., Bolgar, P., Hou, X., Klug-McLeod, J. L., and Butler, C. R. Molecular transformer unifies reaction prediction and retrosynthesis across pharma chemical space. *Chem. Commun.*, 55:12152–12155, 2019. doi: 10.1039/C9CC05122H. URL http://dx.doi.org/10.1039/C9CC05122H.

Li, J., Cai, D., and He, X. Learning graph-level representation for drug discovery, 2017.

Liu, B., Ramsundar, B., Kawthekar, P., Shi, J., Gomes, J., Nguyen, Q. L., Ho, S., Sloane, J., Wender, P., and Pande, V. Retrosynthetic reaction prediction using neural sequence-to-sequence models, 2017.

Lowe, D. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 10 2012. URL https://www.repository.cam.ac.uk/handle/1810/244727.

Molga, K., Gajewska, E. P., Szymkuć, S., and Grzybowski, B. A. The logic of translating chemical knowledge into machine-processable forms: a modern playground for physical-organic chemistry. *React. Chem. Eng.*, 4:1506–1521, 2019. doi: 10.1039/C9RE00076C. URL http://dx.doi.org/10.1039/C9RE00076C.

Schneider, N., Stiefl, N., and Landrum, G. A. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling*, 56(12):2336–2346, 2016. doi: 10.1021/acs.jcim.6b00564. URL https://doi.org/10.1021/acs.jcim.6b00564. PMID: 28024398.

Segler, M. H. S. and Waller, M. P. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry – A European Journal*, 23(25):5966–5971, 2017. doi: 10.1002/chem.201605499. URL https://chemistry-europe.onlinelibrary.wiley.com/doi/abs/10.1002/chem.201605499.

Segler, M. H. S., Preuss, M., and Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018. doi: 10.1038/nature25978. URL https://doi.org/10.1038/nature25978.

Shi, C., Xu, M., Guo, H., Zhang, M., and Tang, J. A graph to graphs framework for retrosynthesis prediction, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2017.

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988. doi: 10.1021/ci00057a005. URL https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

Xie, T. and Grossman, J. C. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120:145301, Apr 2018. doi: 10.1103/PhysRevLett.120.145301. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.145301.

You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6410–6421. Curran Associates, Inc., 2018a. URL http://papers.nips.cc/paper/7877-graph-convolutional-policy-network-for-goal-directed-molecular-graph-generation.pdf.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. GraphRNN: Generating realistic graphs with deep auto-regressive models. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5708–5717, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/you18a.html.

5. Supplementary Material

5.1. Featurization

We featurize atoms and bonds with one-hot encoded vectors of features calculated using RdKit (Landrum). We select features that allow for correct reconstruction of all products and substrates from the USPTO-50k development set. In Tables 2 and 3 we present all used features and their possible values. We concatenate one-hot feature vectors to gain the final input representation of atoms and bonds $H^{OH} \in {\mathbb{Z}_{\geq 0}}^{n \times h_{OH}}$ and $A^{OH} \in {\mathbb{Z}_{\geq 0}}^{n \times n \times a_{OH}}$. During evaluation, if an unknown feature value is seen (for instance, BOND TYPE=QUADRUPLE), we set the one-hot vector for this feature to zeros. For supernodes, all one-hot feature vectors are set to zeros, apart from vectors for the features IS SUPERNODE and BOND TYPE. We connect each atom with itself with a special bond of type SELF. For non-neighboring atoms at i and j we set $A^{OH}_{ij} = \vec{0}$.

For both atoms and bonds, we add a special IS EDITED feature that marks all bonds and atoms that have been modified by actions. This aims to help the decoder to focus on these atoms and bonds, as they are the most probable candidates for the next generation steps.

5.2. Graph edit actions

In Table 6 we show all possible graph actions. The actions were found during the generation of the training samples, which we describe in the next paragraph. The actions have different sets of parameters, depending on the action type.

For *EditAtom*, the action parameters are the atom properties that are changed by the action. Note that a single *EditAtom* action sets all these properties to the specified values. For instance, action number 1, when executed on an atom, sets its formal charge to 0, chiral tag to None, number of explicit Hydrogen atoms to 1 and marks it as aromatic. *EditBond* acts similarly to *EditAtom* but edits properties of a bond instead of an atom. *AddAtom* adds a new atom with specified features, connected to an existing atom with a bond with specified features. *AddBenzene* appends a benzene ring to a specified carbon atom and has no parameters. *Stop* terminates reaction generation and also has no parameters.

BondEdit actions are bond actions, that is they are predicted for a pair of atoms. All other types of actions are atom actions and are predicted for a single atom in the graph.

5.3. Generating training samples

For each reaction from the development set, we generate training samples by finding actions that lead from the target molecule to the reactants. At first, we remap the reaction to a canonical form so that the atom map numbers correspond to to the order of atoms in the canonical SMILES of the compounds. We construct the target graph T and the reactants graph R, using features from Tables 2 and 3. We also initialize the stack of edited atoms $S = \{\}$, which is used for drawing candidate atoms for actions, prioritizing depth-first generation. At each step, we execute the first possible action for the concatenated lists of candidate atoms $S \parallel M(T)$, where $M(T)$ is the list of all atoms from the target ordered by their mapping numbers. After each step, we push the modified atoms to the stack S. If there is more than one possible action to perform on a selected atom i or atom pair i, j, we use the following priority of actions Π:

- Action that deletes the existing bond between i, j
- Any action that adds a bond between i, j
- Any action that edits the existing bond between i, j
- Any action that edits the existing atom i
- Any action that adds a benzene ring to i
- Any action that adds an atom to i

where bond deletion has the highest priority. The aim is to prioritize actions that are usually the hardest (such as deleting a bond, which usually means finding the reaction center) or logically follow each other (editing a neighboring bond and/or atom often follows bond deletion or addition). The training reaction generation is described in Algorithm 1.

5.4. Hyperparameter search

Table 4 shows the hyperparameter values used for the final models. We found them heuristically by observing the validation loss when training with unknown reaction type. We tried increasing N_e to 8, as well as K to 12 and h to 1440 but we did not see a decrease in validation error. We also wanted the decoder to be as small as possible, as it is invoked multiple times during the reaction generation. We found that $N_d = 2$ is sufficient to achieve the best accuracy and increasing it lead to faster overfitting. The total number of learnable parameters in the model is approximately 9.8 million.

Table 2. Features of atoms

NAME	POSSIBLE VALUES	DIM
IS SUPERNODE	YES, NO	2
ATOMIC NUMBER	5, 6, 7, 8, 9, 12, 14, 15, 16, 17, 29, 30, 34, 35, 50, 53	16
FORMAL CHARGE	-1, 0, 1	3
CHIRAL TAG	NONE, @, @@	3
NUMBER OF EXPLICIT HS	0, 1, 2, 4	4
IS AROMATIC	YES, NO	2
IS EDITED	YES, NO	2
TOTAL		32

Table 3. Features of bonds

NAME	POSSIBLE VALUES	DIM
BOND TYPE	SUPERNODE, SELF, SINGLE, DOUBLE, TRIPLE, AROMATIC	6
BOND STEREO	NONE, Z, E	3
IS EDITED	YES, NO	2
TOTAL		11

Table 4. Final model hyperparameter values

h	a	d	K	N_e	N_d	TOTAL PARAMS
1024	128	128	8	6	2	~9.8MIL

Table 5. USPTO-50k data set information

# TRAIN	# VALID	# TEST	# TOTAL
39662	5199	5155	50016

Algorithm 1 Generating training samples for a reaction

Input: target graph T, reactants graph R
Initialize $S \coloneqq \{\}$ (empty list)
Let $M(T) = \{$all atoms from T ordered by mapping$\}$
`marker:`
while $T \neq R$ **do**
  for i in $S \parallel M(T)$ **do**
    for P in Π **do**
      for a in P **do**
        if a is an atom action **then**
          if applying a to i leads to R **then**
            generate training sample (T, a)
            apply action a to atom i in T
            $S \coloneqq \{i\} \parallel S$
            go to `marker`
          end if
        else if a is a bond action **then**
          for j in $S \parallel M(T)$ **do**
            if applying a to i, j leads to R **then**
              generate training sample (T, a)
              apply action a to atoms i, j in T
              $S \coloneqq \{j, i\} \parallel S$
              go to `marker`
            end if
          end for
        end if
      end for
    end for
  end for
end while

Table 6. All graph actions found on the USPTO-50k development set

	Formal charge	Chiral tag	Num of explicit Hs	Is aromatic	
EditAtom	0	None	1	Yes	1
	+1	None	0	No	2
	0	@@	1	No	3
	0	None	0	No	4
	0	@	1	No	5
	-1	None	0	No	6
	0	None	0	Yes	7
	0	@	0	No	8
	0	@@	0	No	9
	-1	None	0	Yes	10
	0	None	1	No	11

	Bond type	Bond stereo	
EditBond	None (delete bond)	/	12
	Single	None	13
	Triple	None	14
	Double	None	15
	Double	Z	16
	Double	E	17
	Aromatic	None	18

	Atomic num	Formal charge	Chiral tag	Number of exp Hs	Is aromatic	Bond type	Bond stereo	
AddAtom	35	0	None	0	No	Single	None	19
	6	0	None	0	No	Single	None	20
	8	0	None	0	No	Double	None	21
	6	0	None	0	Yes	Single	None	22
	8	0	None	0	No	Single	None	23
	17	0	None	0	No	Single	None	24
	7	0	None	0	No	Single	None	25
	16	0	None	0	No	Single	None	26
	53	0	None	0	No	Single	None	27
	8	–1	None	0	No	Single	None	28
	5	0	None	0	No	Single	None	29
	9	0	None	0	No	Single	None	30
	15	+1	None	0	No	Single	None	31
	7	+1	None	0	No	Single	None	32
	50	0	None	0	No	Single	None	33
	14	0	None	0	No	Single	None	34
	7	+1	None	0	No	Double	None	35
	7	–1	None	0	No	Double	None	36
	29	0	None	0	No	Single	None	37
	16	+1	None	0	No	Single	None	38
	12	+1	None	0	No	Single	None	39
	15	0	None	0	No	Single	None	40
	6	0	@@	1	No	Single	None	41
	30	+1	None	0	No	Single	None	42
	30	0	None	0	No	Single	None	43
	6	0	None	0	No	Double	None	44
	15	0	None	0	No	Double	None	45
	16	0	None	0	No	Double	None	46
	7	0	None	0	No	Double	None	47
	12	0	None	0	No	Single	None	48
	6	0	@@	0	No	Single	None	49
	6	0	@	1	No	Single	None	50
	6	0	None	0	No	Double	E	51
	6	0	None	0	Yes	Aromatic	None	52
AddBenzene				/				53
Stop				/				54