HyFactor: Hydrogen-count labelled graph-based defactorization Autoencoder

Tagir Akhmetshin[1, 2], Arkadii Lin[1], Daniyar Mazitov[2], Evgenii Ziaikin[2], Timur Madzhidov[2]*, and Alexandre Varnek[1]*

[1] Laboratory of Chemoinformatics, UMR 7140 CNRS, University of Strasbourg, 4, Blaise Pascal str., 67081 Strasbourg, France

[2] Laboratory of Chemoinformatics and Molecular Modeling, Butlerov Institute of Chemistry, Kazan Federal University, 18, Kremlyovskaya str., 420008 Kazan, Russia

*e-mail: Timur.Madzhidov@kpfu.ru, varnek@unistra.fr

Abstract

Graph-based architectures are becoming increasingly popular as a tool for structure generation. Here, we introduce a novel open-source architecture HyFactor which is inspired by previously reported DEFactor architecture and based on the hydrogen labeled graphs. Since the original DEFactor code was not available, its new implementation (ReFactor) was prepared in this work for the benchmarking purpose. HyFactor demonstrates its high performance on the ZINC 250K MOSES and ChEMBL data set and in molecular generation tasks, it is considerably more effective than ReFactor. The code of HyFactor and all models obtained in this study are publicly available from our GitHub repository: https://github.com/Laboratoire-de-Chemoinformatique/hyfactor

Keywords

Molecular design, Autoencoders, Deep learning, Generative models, Chemical databases

Introduction

Nowadays, Deep Neural Networks (DNNs) play an important and significant role in drug and materials discovery, being used for properties prediction [1], de novo design [2], and computer-aided retrosynthesis [3]. One of the most widely used DNN architectures is Autoencoder (AE)[4] due to its ability to encode chemical structures in their latent representation as well as generate new compounds by decoding sampled latent vectors using a decoder subnetwork.

For molecular generation, the majority of AEs are based on the Simplified Molecular Input Line-Entry System (SMILES). [5] This chemical language allows modellers to employ all the power of Natural Language Processing (NLP) techniques to solve chemical problems. Although SMILES seems suitable for de novo design tasks, the latent representation of text strings may not reflect chemical similarity relationships between underlying molecules.

Graph-based AE (GAE) architectures [6] offer a valuable alternative to SMILES-based architectures. Within these approaches, a molecule is represented as a graph in which nodes and edges encode, respectively, atoms and chemical bonds. GAEs have three fundamental advantages over SMILES-based architectures. First, no specific order of graph traversal is required, which solves the problem of a canonical order often imposed for SMILES strings. Second, a graph object does not obey specific grammar rules, such as opening and closing brackets, cycles numbering, etc., which seriously limits the generation ability of networks (although a recently proposed SELFIES language [7] may partially improve that). Finally, GAEs always sample graph objects, which, in turn, allows a meaningful errors analysis in terms of chemistry (i.e., detecting graph disconnectivity and valence errors).

A molecular graph can be decomposed to a vector of atoms and bonds matrix, and modelled further using Graph Convolutional Networks (GCN)[8]. Once a latent vector is obtained, it can be decoded using various decoding techniques: single-shot and iterative generation. Single-shot decoders generate atoms and bonds in a graph in a single pass.[6, 9] Their training is fast, but the simultaneous generation of atom vectors and bond matrices is technically challenging.[10] Iterative decoders, in contrast, create atoms and bonds sequentially one by one, so they can stop when a molecule is reconstructed.[11] The first class generates atoms' vectors, from which atom types and connectivity are extracted separately. One example of such iterative generation is an autoregressive concept, when the generation of the next atom is based on previously created atoms.[12, 13] Another popular approach uses a recurrence-based generation, where the next atom is generated from a hidden (or difference) vector, which updates every step.[14] The second class of decoders uses a Markov decision process. They apply several actions, such as "creation of atoms" and "creation of bonds" to a given subgraph of a molecule to obtain an updated subgraph until the molecule is reconstructed.[11, 15] However, interative generation requires much more complex and slow network architecture than single-shot Autoencoders.[9] These and other limitations (e.g., high memory and time costs) prevent GAE architectures to become widely used.

Recently, DEFactor GAE architecture was published by R. Assouel et al. [13] uses reccurent iterative decoder. The encoder in DEFactor is a multi-layer GCN, whereas the decoder combines the Long-Short Term Memory (LSTM) [16] cell for atomic vectors generation with a new adjacency matrix defactorization procedure (see Method section for details). This approach allows one to model molecular graphs much more efficiently. Although the results of the benchmarking study reported in the original publication look promising, many important details, such as the number of convolutional layers in the encoder or the batch size, needed for re-implementation of this approach are missing. Therefore, we made an attempt to re-implement and further improve the DEFactor architecture. The improved version of DEFactor is referred here as ReFactor (see the details in the Method chapter). It was tested in ZINC250K and ChEMBL v.27 data sets as well as MOSES benchmarking, and its efficiency was shown to be comparable to that of other GAE architectures.

Moreover, we propose a methodological improvement that reduces the amount of GPU memory required to store the model as well as training time without loss of accuracy. The GCN employed by the encoder subnetwork in DEFactor computes the neighbours' messages within each bond-type specific channel. As a consequence, it is necessary to store three or four bond-type-specific adjacency matrices and specific trainable weight matrices (depending on whether an aromatic bond is considered an alternate of single and double bonds or a separate bond type). This takes a lot of memory and adds numerous mathematical operations to the corresponding computational graph in DEFactor. Here, we propose to use the number of hydrogens attached to each heavy atom instead of bond order. In fact, based on the number of hydrogens at every atom and the graph adjacency matrix, molecular structure can be reproduced. Also, it solves the problem of the standard representation of functional groups and aromaticity. A similar approach is used in the InChI molecule representation system [17]. In this case, a molecular graph can be represented by three objects: 1) a vector of atom types, 2) a vector of the numbers of attached hydrogens, and 3) a single binary adjacency matrix. By simplifying the encoder GCN, we can reduce the number of trainable weights by 50% (see Methods section). Moreover, by using hydrogen atom numbers instead of bond orders, functional groups standardization and aromatization steps can be omitted. This can enhance the network's generation ability. For example, the addition of an aromatic bond to an open-chain fragment could easily introduce an error, and, in the opposite, the probability to generate an aromatic ring in a Kekule form is low if the model is trained to alternate single and double bonds in rings.

Based on the above mentioned inventiones, we introduce a new hydrogen-count labelled graph-based GAE architecture named HyFactor. Within this architecture, a DNN is combined with a set of heuristic rules on allowed atoms' valences needed to do a conversion of a regular molecular graph to a hydrogen-count labeled graph (a graph where a certain number of hydrogens is assigned to each heavy atom) and back. It was compared to DEFactor and ReFactor architectures using ZINC250K, ChEMBL v.27 and MOSES benchmarking data sets in the tasks of reconstruction and generation. It has been shown that ReFactor and HyFactor possess similar reconstruction rates, whereas HyFactor trains faster and requires less GPU memory.

Therefore, in this work we present the two new graph-based AutoEncoder architectures that can be used for molecule encoding and generation tasks. In order to accelerate development in this area and create sustainable and reproducible algorithms, we are contributing proposed architectures to open-source. Additionaly, we provide weights of both architectures as well as tools to create and decode a new molecular representation, the hydrogen-count labeled graph. The code and all models are publicly available in our GitHub repository: https://github.com/Laboratoire-de-Chemoinformatique/hyfactor

Methods

Data and curation

In this work, three data sources were used: a subset of ZINC database called ZINC250K [4], MOSES (v. 1.0) [18] data set, and ChEMBL database (v. 27) [19]. All sets were standardized using the following procedures: 1) dearomatization, 2) isotopes removal, 3) stereo marks removal, 4) explicit hydrogens removal, 5) small fragments removal, 6) solvents removal, 7) salts strip, 8) neutralization of charges, 9) functional groups transformation, 10) selection of canonical tautomer form of molecule, 11) aromatization, 12) duplicates removal, 13) dearomatization. All stages of the standardization were done by ChemAxon JChem's utilities [20]. Once structures were standardized, the order of atoms was defined from the canonical SMILES string produced by ChemAxon JChem.

About 1.7K structures were removed from the ZINC250K data set as the result of the cleaning procedure. The cleaned set was split into training, validation (tuning), and test sets according to the previous article [21]. The test set was taken as 5K predefined molecules, and the remaining structures were randomly split into training and validation sets in ratio 9:1. Note that several duplicates were found in ZINC250K (see Table 1), due to the presence of stereoisomers in the data set. This may cause some overestimation of model performance for architectures reported earlier.

Table 1. Results of ZINC 250K data set standardization.

Filters	Training and validation sets	Test set
Number of molecules	244 455	5 000
Duplicates within set	1612	1
Duplicates between training and test sets	73	
Remaining molecules	**242 770**	**4 999**

The ChEMBL database was standardized by the same workflow as the ZINC250K data set. The initial database consisted of 1.9M molecules and it was reduced to 1.6M of standardized structures. The prepared data set was additionally analyzed in terms of the frequency of atom types (Figure S1). 60 unique atomic types (including information on atomic charge) were found in ChEMBL, and only 15 (C, O, N, S, F, Cl, N^+, O^-, Br, P, I, N^-, B, Si, and Se) have been chosen according to the threshold of 1000. Next, compounds with less than 5 and more than 50 heavy atoms have been removed due to their under-representation. The filtrated ChEMBL data set was then split into a training set (80% of data or 1.3M molecules) and a validation set (20% of data or 327K molecules). It was not divided into a test set because the amount of data in the validation set was sufficient to reliably assess the performance of the model.

The MOSES data set was analyzed with the proposed standardization procedure; however, no mistakes were found. Therefore, it was used as it is. The original "training" set was split into training and validation sets in the 4:1 ratio.

All chemical structures from each data set have been kekulized in order to avoid a need to introduce an additional *aromatic* bond type for ReFactor architecture. The latter would increase the size of the graph-based architecture, slow down the calculations and decrease the number of valid structures in sampling.

After standardization of the data sets, they were analyzed in terms of heavy atoms count distribution (Figure 1).

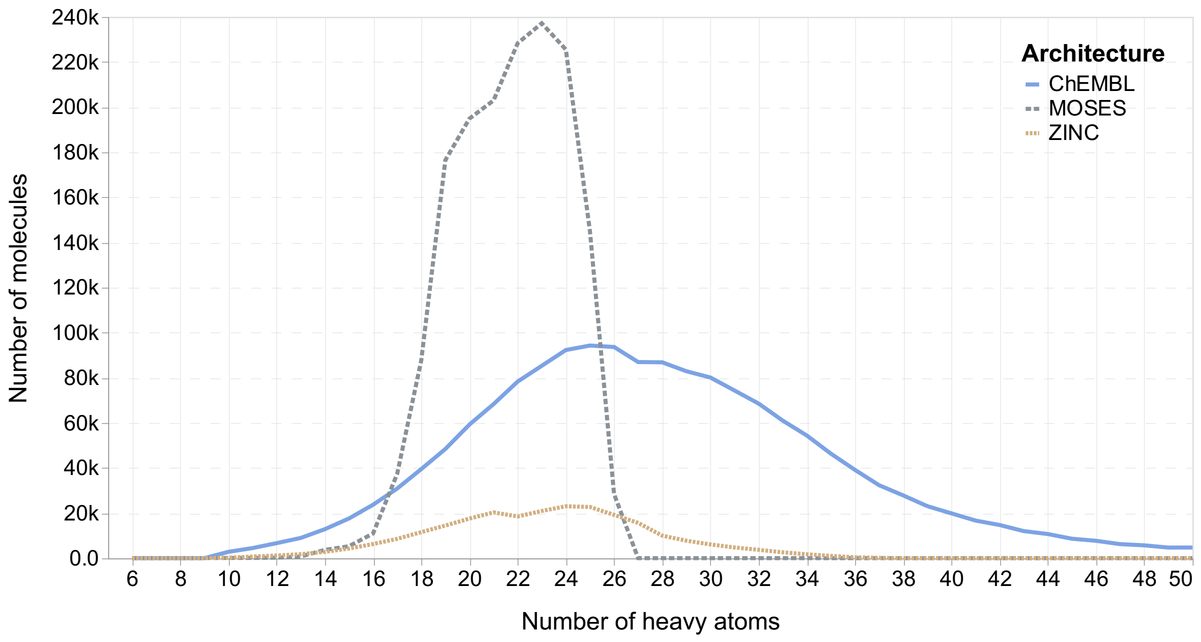

Figure 1. Heavy atoms count distribution in studied data sets

Both the ChEMBL and ZINC250K data sets have a similar distribution of heavy atom count. However, the MOSES data set differs from both, and most of the structures lie in the range from 16 to 26 heavy atoms.

Hydrogen-count labelled graph

We propose to use a new molecular graph-based representation – Hydrogen-count labelled graph (HLG) (see Figure 2). In this graph, only the connections between atoms and the count of hydrogens connected to the atoms are taken into account. Also, the formal charge of an atom is used as a vertex label. This representation has already been tested with graph convolution networks on quantitative structure-property relationship (QSPR) tasks.[22] In the latter work, the performance of HLG was similar to other representations, but it has not been compared to the classical molecular graph representation.

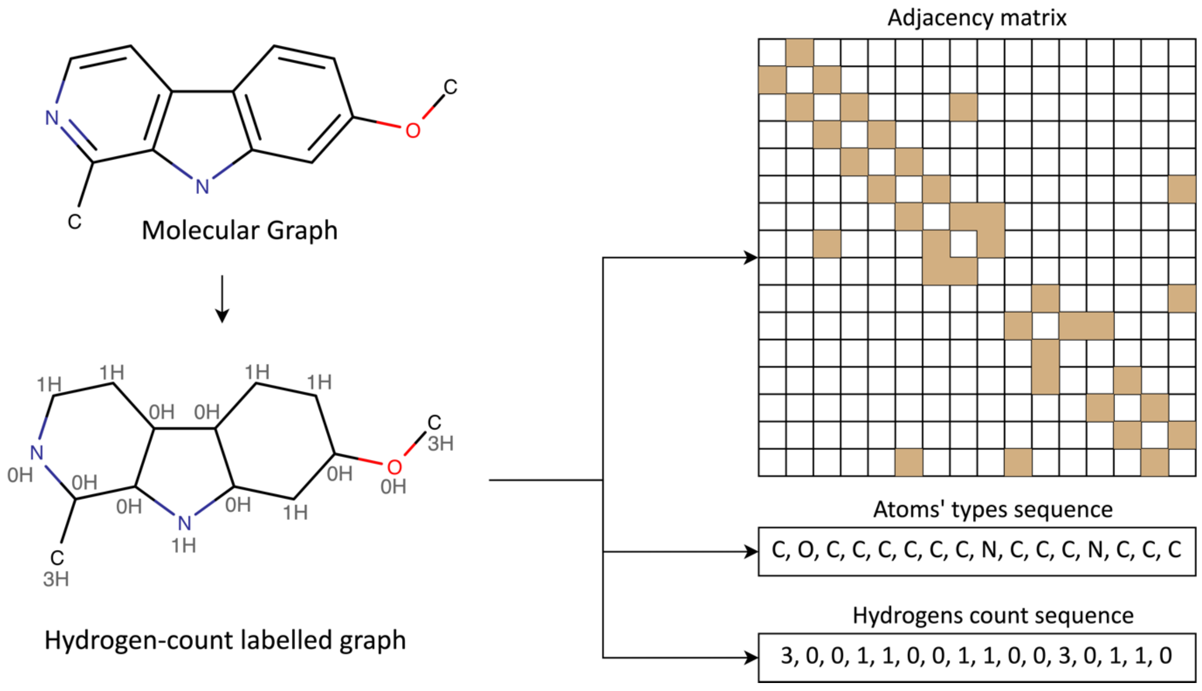

Figure 2. Hydrogen-count labeled graph representation. Here the molecular graph (hydrogens are hidden) is converted into a graph with no edge features, while the nodes have two features - types of atoms and number of hydrogens.

We convert the molecular graph to the HLG, and then to three numerical representations: adjacency matrix, atomic types which includes atom symbol and charge, and hydrogens count vectors (Figure 2). The conversion algorithm from molecular graph to HLG and backward is implemented with CGRtools toolkit.[23]

DEFactor, ReFactor, HyFactor

In this work, we propose *ReFactor* architecture which is a slight modification of *DEFactor* architecture [13] published by Assouel et al. earlier. We chose the DEFactor architecture because its concept of decoding incorporates the ideas of iterative decoders, making the generation procedure flexible and optimal.

All three architectures (DEFactor, ReFactor, and HyFactor) are depicted in Figure 3. The encoder from the DEFactor architecture uses one-hot embeddings to represent atoms in a molecular graph, and consists of several layers of edge-specific graph convolution networks [8, 24] that can be expressed as follows:

$$\mathbf{H}^{l+1} = \mathrm{ReLU}\left[\sum_{b}\left(\mathbf{D}_b^{-\frac{1}{2}}\mathbf{E}_b\mathbf{D}_b^{-\frac{1}{2}}\mathbf{H}^l\mathbf{W}_b^l\right) + \mathbf{H}^l\mathbf{W}_{\mathrm{self}}^l\right] \quad (1),$$

where $\mathbf{H}^l$ is atoms' vectors after the lth graph convolution layer, $\mathbf{E}_b$ is a bond-type specific adjacency matrix, $\mathbf{D}_b$ is the corresponding bond-type specific diagonal degree matrix, $\mathbf{W}_b$ and

$\mathbf{W}_{\mathrm{self}}$ are trainable matrices of weights for every bond type *b* and weights for self-channel, respectively, and ReLU is the rectifier activation function. The aggregation of atomic vectors is done by a Long short-term memory (LSTM) [16] unit followed by a one-layer perceptron (see Figure 3a) giving molecular latent vector.

In the decoder, the molecular latent vector is unpacked into a set of atomic embeddings $\widetilde{\mathbf{H}}$, where each h_i is predicted using LSTM layer. Thus, the entire matrix of atoms' embeddings is restored. Next, it passes through two subunits in parallel, where the first one is represented by a Multi-layer Perceptron (MLP) with softmax output function that returns predictions on atoms' types $\widetilde{\mathrm{A}}$, and the second one is the multichannel defactorization procedure [25] needed to reconstruct the bond matrix. The latter is described by the authors as

$$\widetilde{\mathbf{E}}_{\mathrm{b}} = \sigma(\widetilde{\mathbf{H}}\mathbf{W}_{\mathrm{b}}\widetilde{\mathbf{H}}^{\mathrm{T}} + \mathrm{bias}) \qquad (2),$$

where $\widetilde{\mathbf{E}}_{\mathrm{b}}$ is the reconstructed adjacency matrix for bond type *b*, $\widetilde{\mathbf{H}}$ is the matrix of the recovered atoms' embeddings h_i, $\mathbf{W}_{\mathrm{b}}$ is a diagonal matrix of weights for the bond type *b*, and σ is a sigmoid output function. A certain probability is returned for each bond type between each pair of atoms, and the bond type with the highest probability is selected for reconstruction. A three-steps procedure including teacher forcing has been used to speed up the training of DEFactor. Within each step, trainable weights of a certain part of the network were frozen, and then relaxed at the next step.

The loss function used in DEFactor was a combination of categorical cross-entropy (CCE) for atoms predictions (3), and binary cross-entropy (BCE) for bonds predictions (4):

$$\mathrm{L}_{\mathrm{atoms}} = -\frac{1}{\mathrm{n}}\sum \mathrm{A} * \log(\widetilde{\mathrm{A}}) \qquad (3)$$

$$\mathrm{L}_{\mathrm{bonds}} = -\frac{1}{\mathrm{n}^2}\sum_{b}^{4}\left[\mathrm{E}_{\mathrm{b}} * \log(\widetilde{\mathrm{E}}_{\mathrm{b}}) + (1 - \mathrm{E}_{\mathrm{b}}) * \log(\widetilde{\mathrm{E}}_{\mathrm{b}})\right] \qquad (4),$$

where A is a one-hot matrix for atoms types, Ã is a predicted atoms types probability matrix, and n is the number of atoms in a molecule, E_{b} is the real adjacency matrix for bond type *b*.

The ideas behind this architecture seemed promising, although the architecture was poorly described in the paper by Assouel et al [13]. For instance, the number of GCN layers as well as the dimensionality of the atoms' embedding matrix was not mentioned. Thus, we reimplemented DEFactor architecture as much as possible and introduced some changes that improved its performance.

Unless otherwise specified, the dimensionality of the atom embedding vectors, as well as all internal and latent vectors, was the same. Dimensionality of latent vectors refers to these dimensions. Parameters for each experiment are specified in Table S1 of Supporting Information. All layers were taken with standard parameters of Tensorflow package v. 2.6. if not additionaly mentioned.

To stabilize the learning process, the GCN was upgraded with Layer Normalization (LN) layer among atoms features (parameter "axis=-2") and masking of imaginary atoms (padding):

$$\mathbf{H}^{l+1} = \mathrm{Mask} * \mathrm{ReLU}\left[\sum_{b}\left(\mathbf{D}_b^{-\frac{1}{2}}\mathbf{E}_b\mathbf{D}_b^{-\frac{1}{2}} * LN(\mathbf{H}^l\mathbf{W}_b^l)\right) + LN(\mathbf{H}^l\mathbf{W}_{\mathrm{self}}^l)\right] \quad (5)$$

In each experiment, the number of CGN layers was fixed at five. The dimensionality of the input and output vectors was the same.

For atomic vector aggregation, LSTM units in DEFactor were replaced with bidirectional Gated Recurrent Units [26] (GRU; see Figure 3b). The output of GRU was passed to the dense, batch normalisation (BN) and ReLU activation layers to obtain the molecular latent vector. A teacher-forcing technique published in the original article was not applied since it had not led to performance improvement.

In the decoder, atom vectors were generated by sequentially connected two GRU layers and a dense layer in between, headed by a RepeatVector layer (see Figure 3b). In such case, the input molecular latent vector was repeated N times (i.e., according to the maximal molecular graph size), passed through the first GRU and intermediate atom vectors were returned. These intermediate vectors were then concatenated with the repeated molecular vectors. They passed first through a perceptron layer with a ReLU activation function and then through the second GRU layer. Further, the hidden vectors of the second GRU were used as retrieved atoms' embeddings. For the atoms reconstruction, the retrieved atoms' embeddings were passed to two dense layers with output dimensions equal to the number of atom types. Activation of the first layer was ReLU, and activation of the second - Softmax function. During the bonds reconstruction step, the atoms' embeddings were forwarded to a dense layer with output dimension of the latent vector divided by eight and ReLU activation and to the defactorization layer.

These and other minor changes allowed us to handle molecules containing up to 50 heavy atoms with the ReFactor architecture (see Results and Discussion section).

The HyFactor architecture was based on ReFactor. The main changes concern the steps of graph convolution and graph reconstruction from the atoms' vectors (see Figure 3c). First, the HLG was transformed to the atoms' (dimension of 64) and hydrogens' (dimension of 4) embeddings. These embeddings were concatenated and passed through the dense layer with ReLU activation function. The graph convolution network was similar to that in ReFactor:

$$\mathbf{H}^{l+1} = \mathrm{Mask} * \mathrm{ReLU}\left[\mathbf{D}^{-\frac{1}{2}}\mathbf{E}\mathbf{D}^{-\frac{1}{2}} * \mathrm{LN}(\mathbf{H}^l\mathbf{W}_{\mathrm{neighbors}}^l) + \mathrm{LN}(\mathbf{H}^l\mathbf{W}_{\mathrm{self}}^l)\right] \quad (6)$$

where **E** and **D** are adjacency and degree matrices of HLG, the other designations are the same as in eqn. 1 and 2. From the eqn. 6 it can be seen, that the number of the training parameters decreased in two times.

With the graph reconstruction task, the difference from ReFactor was that the number of hydrogens and atom types were predicted using a softmax function after two dense layers with a ReLU activation in between. In the experiments we considered that the maximal number of hygrogens around atom is 3. The adjacency matrix was reconstructed using defactorization (eqn. 2) ignoring bond types (only one trainable diagonal $\mathbf{W}_b$ matrix).

The initialization of weights for each layer was done with HE normal initialization [27] and the training of the architecture was done with AdaBelief optimizer [28] with standard parameters. In order to maintain training stability, exponential learning decay was applied.

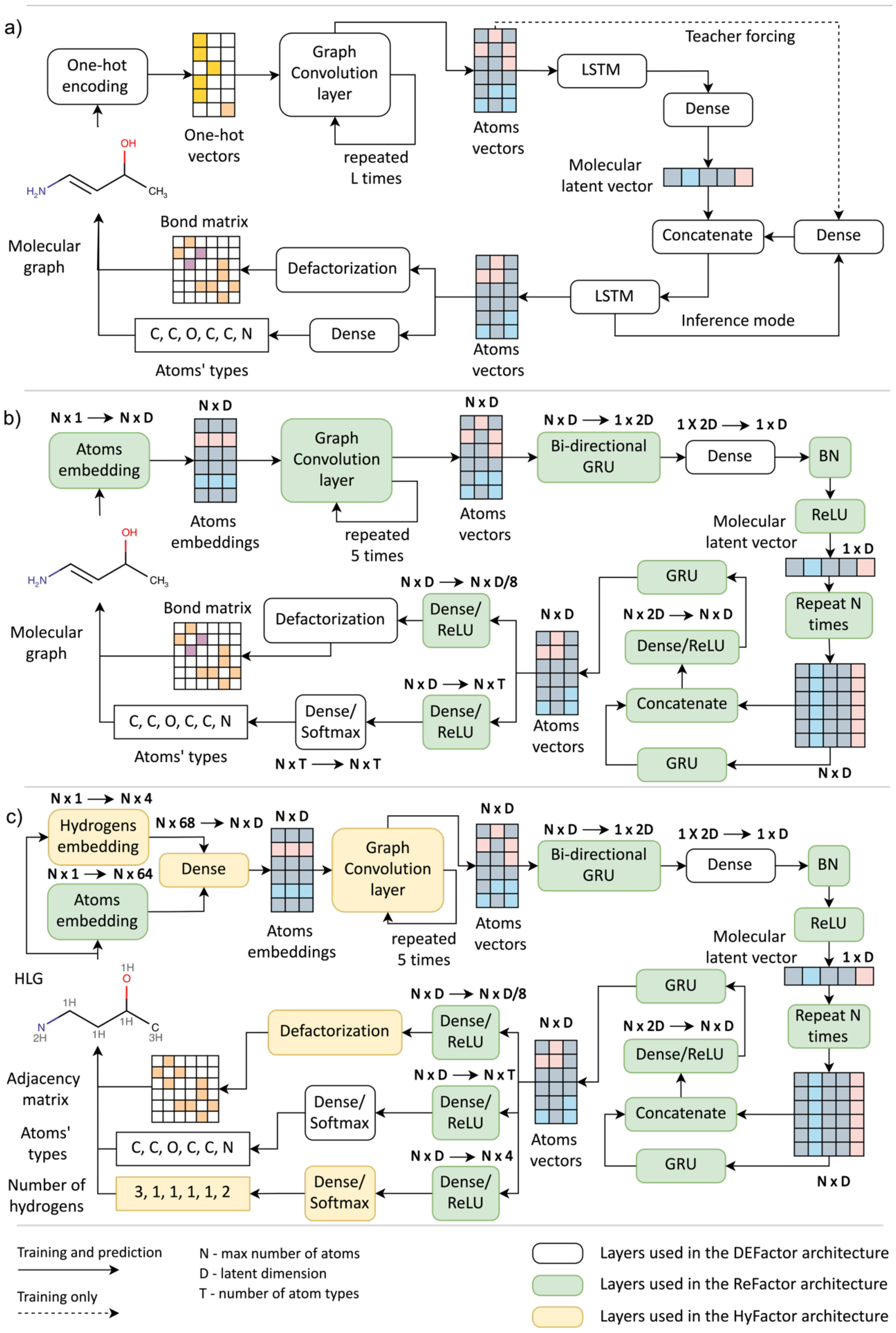

Figure 3. Architectures of different Autoencoders considered in this work: (a) DEFactor [13], (b) ReFactor and (c) HyFactor. BN refers to Batch Normalization layer and GRU – Gated Reccurent Unit. Parameters for each experiment are specified in Table S1 of Supporting Information.

Sampling procedure

Since in current architectures no special restrictions on latent space distribution were posed, for the molecule generation procedure the latent vectors were sampled in the vicinity of the known molecules. In this case, few molecules from the training set were selected and their latent vectors were used as seeds in the generation process as it was done in [29] (Figure 4). During generation, the latent vectors were multiplied by noise vectors (i.e., random numbers generated from a probability density function of a known random distribution), followed by their decoding to a molecular graph.

For the experiments, the probability density function of lognormal distribution was chosen. To effectively and systematically explore chemical space around the given seed, the “mean” parameter was set to 0 and the “standard deviation” was variable. Therefore, as the "standard deviation" parameter would rise, the probability of generating more dissimilar structures should increase.

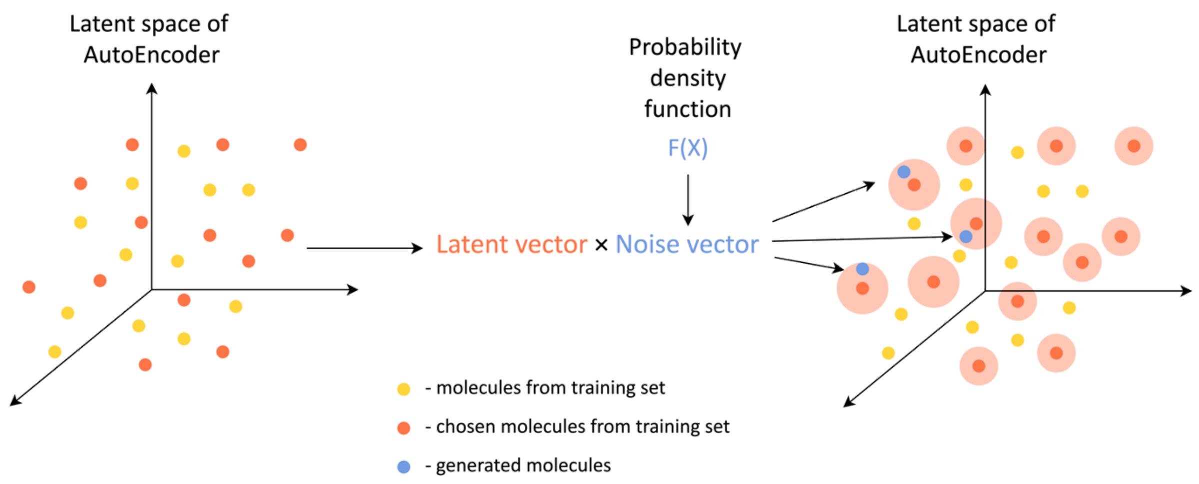
Latent space of AutoEncoder
Probability density function
F(X)
Latent vector × Noise vector
Latent space of AutoEncoder
- molecules from training set
- chosen molecules from training set
- generated molecules

Figure 4. Sampling strategy for new structures in the Autoencoder latent space.

Results and Discussion

Reconstruction rate of different graph Autoencoders

The performance of Autoencoders’ model is measured by the reconstruction rate representing a percentage of correctly reconstructed compounds out of the considered data set. Reconstruction rate values obtainedfor several SMILES-based and graph-based Autoencoders on the ZINC 250K reference set are reported in Table 2.

Table 2. ZINC 250K results

Architecture name	Standardization	Molecular representation	Reconstruction rate, %		
			Training set	Validation set	Test set
TSGCD[10]	-	Molecular graph	-	-	90.5
DEFactor[13]	-	Molecular graph	-	-	89.8
JTVAE[14]	-	Molecular graph*	-	-	76.7

Re-balanced VAE[30]	-	SMILES	-	-	92.7
All SMILES[31]	-	SMILES	-	-	87.6
SDVAE[32]	-	SMILES	-	-	76.2
ReFactor	no	Molecular graph	99.5	90.8	90.7
ReFactor	yes	Molecular graph	99.7	90.0	90.0
HyFactor	no	HLG	99.3	89.3	89.0
HyFactor	yes	HLG	99.2	89.8	88.4

* JTVAE uses hierarchical fragments instead of atoms to reconstruct the molecule

In comparison to graph-based architectures, SMILES-based architectures show similar performance. The leading graph-based architecture (TSGCD) has a reconstruction rate that is only 2% lower than the best SMILES-based Autoencoder (Re-balanced VAE). The DEFactor architecture also shows high performance and is only 1% lower than the leading graph-based Autoencoder. Although it was found that the protocols for data set standardization were not sufficiently addressed in the publications on all the architectures presented. Thus, results for the ReFactor and HyFactor architectures are presented for both non-standardized and standardized data sets. In addition, results for training and validation sets for model overfitting analysis are given.

The reconstruction rate for ReFactor, trained and tested on the non-standardized data set, is slightly higher than for the standardized data set. This fact supports our assumption that the performance of Autoencoder may be overestimated since training and test sets partially overlap in the non-standardised data set. Comparing the results of ReFactor with the original DEFactor, we see that on both standardized and non-standardized data sets our implementation outperforms the original architecture.

HyFactor architecture has almost the same reconstruction rate on both sets as ReFactor. Hence, it can be concluded that the molecular representation used in HyFactor does not change the performance of the model while reducing the number of neural network parameters (from 12M to 10M).

Generation ability of graph-based Autoencoders

The Autoencoders are a class of generative neural networks, which are widely used to generate novel chemical structures. Thus, here we discuss the ability of proposed graph-based Autoencoders to generate molecules. For this purpose, MOSES benchmarking was used [18]. Original DEFactor architecture could not be benchmarked since the source code is not available and corresponding information is not given in original publication.

The metrics included in the MOSES package are adopted for the distribution learning objective (see the supporting information for details). Even though the proposed generation method does not fit into the category of such methods, most of the metrics are applicable to sampled structures. Also worth noting is that distribution learning models generate various structures, while the proposed approach generates molecules around seeds. Therefore, it was not possible to select a subset of ten thousand structures for calculation of uniqueness, so it was calculated on the entire set.

For the generation tests ReFactor and HyFactor architectures were trained on 80% of the original MOSES training set. Both architectures achieved more than 99% of the reconstruction rate on the remaining 20% of the data used as validation set. Then, 10K compounds were randomly selected as seeds for sampling new structures. For each seed compound, 10 virtual structures were generated, so the total number of structures obtained should be 100K. The generation was based on log-normal distribution with a mean equal to zero and a variable standard deviation from 0.2 to 1.0 with a step of 0.2.

A preliminary review of the results revealed that MOSES does not correctly handle a molecular graph consisting of several components (mixture of compounds) and some valence errors during metrics calculation. At the same time, they were generated quite frequently by the proposed graph architectures. To verify the results of benchmarking study, we performed the validity analysis including additional valence checks and disconnected graph check. The validity checks were implemented using the CGRtools package [23]. The results of the analysis of standard deviation parameter influence on validity, uniqueness and novelty of the generated structures is given in Table 3.

Table 3. MOSES metrics calculated for the 100K structures generated with different standard deviations (STD)

STD	Validity		Uniqueness		Novelty	
	Original	Modified*	Original	Modified*	Original	Modified*
ReFactor						
0,2	0,997	0,996	0,656	0,105	0,074	0,042
0,4	0,921	0,886	0,748	0,265	0,634	0,578
0,6	0,698	0,503	0,948	0,730	0,875	0,814
0,8	0,540	0,149	0,998	0,971	0,978	0,855
1	0,516	0,022	0,961	1,000	0,999	0,983
HyFactor						
0,2	1,000	0,9880	0,661	0,193	0,117	0,006
0,4	0,989	0,7290	0,795	0,267	0,729	0,129
0,6	0,940	0,1820	0,980	0,634	0,923	0,288
0,8	0,886	0,0120	1,000	0,966	0,993	0,491
1	0,822	0,0003	0,981	1,000	1,000	0,912

* With additional valence and disconnected structures checks and uniqueness computed on the whole generated set

Observing the results, it is evident that the percentage of new molecules and their uniqueness grow with increasing standard deviation for both models, but the validity drops. Nonetheless, when standard deviation is high, structures returned after validity check are characterized by high novelty and uniqueness. The probable reason is the discontinuity of the latent space that has lots of voids. It is quite common for regular Autoencoders without special regularization on latent space, like in variational Autoencoders [33], or application of latent vectors for additional task solving [34].

Another observation is that the structures generated by ReFactor are more valid and novel than HyFactor's molecules. It has been suggested that during systematic sampling with HyFactor, it is easier to fall into a void, where an invalid structure is present. Yet, both architectures achieved high reconstruction rates, so they should have similar generative capabilities. Thus, if the sampled method is changed, it is possible to achieve better results for both HyFactor and ReFactor. We leave this issue for the future work.

After analyzing the generation with variable standard deviation, it was decided to compare the molecules generated for an STD equal to 0.4 with the results for known Autoencoders. The results of MOSES benchmarking are given in Table 4. In the table, the most important metrics that can help us to analyze the generation ability of the proposed architectures are presented. The results for the all other metrics available in the MOSES package are given in Table S2.

Table 4. Results of MOSES benchmarking compared to the known Autoencoders. The similarity metrics are given in comparison with the MOSES test set. The presented Autoencoders are SMILES-based, except JTVAE.

Model	Valid	Unique 10k	Novelty	FCD	SNN	Scaf	IntDiv
Combinatorial (BRICS frag)	*1.000*	*0.991*	*0.988*	*4.238*	*0.451*	*0.445*	*0.873*
AAE	0.937	0.997	0.793	0.556	0.608	0.902	0.856
VAE	0.977	0.998	0.695	0.099	0.626	0.939	0.856
JTVAE	1.000	1.000	0.914	0.395	0.548	0.896	0.855
ReFactor 0.4 std*	0,886	0,265**	0,578	1.743	0,547	0,847	0,868
HyFactor 0.4 std*	0,729	0,267**	0,129	0.365	0,614	0,862	0,860

* All metrics are calculated if valence errors and disconnected structures are excluded.
** Uniqueness was calculated on the whole generated data set after filtration by validity.

As graphs are discrete objects, it is expected that the latent space of a graph-based Autoencoder should also be discrete. In other words, invalid molecular graphs (for example, disconnected structures or structures with valence mistakes) are present in the space between the valid molecules. With the proposed sampling method, it is not possible to control the validity of the generated molecules, thus validity is lower than for other architectures (Table 4). The uniqueness and novelty is also lower than for other architectures, as we generate molecules in the vinicity of seed molecules. Therefore, validity, uniqueness, and novelty can be increased by updating the sampling method, which is outside of scope of this study.

According to Table 4, scaffold similarity (Scaf) is low for each proposed architecture, so more scaffolds were generated than for other Autoencoders. This is an advantage of both architectures, as they can perfrom scaffold hopping, which is crucial during search of new drug-like molecules. Moreover, HyFactor and ReFactor generate molecules with high internal diversity (IntDiv), indicating a broader variety of generated structures. These properties make them powerful tools for discovering new analogs of molecules that can be used for the creation of focused libraries.

Another observation is that generated samples from ReFactor have the biggest Fréchet Chemnet Distance (FCD) and the smallest Similarity to a Nearest Neighbour (SNN). These values indicate that ReFactor's structures are more dissimilar than any other architecture. Hence, ReFactor can be helpful in the search for a novel scaffold or pattern task, where only a few molecules are known.

Through the proposed generation approach, it is possible to examine how molecules are arranged in the latent space. It is expected that molecules generated close to the seed should have a similar chemical structure, and vice versa, if molecules are obtained far away from the seed, they should be more dissimilar. Hence, it was decided to take one seed structure and use both models to generate several molecules with a range of standard deviation from 0.3 to 1.0 with the step of 0.02. At each step, 10 molecules were generated followed by the validity and uniqueness check. These simulations resulted in 10 and 40 new generated structures in average from each structure for HyFactor and ReFactor, respectively. Several generated structures are shown in Figure 5.

Before analysis all molecules were aromatized with ChemAxon toolbox. To measure similarity of new molecules, Tanimoto metric was applied. The similarity was calculated on atom-centred ISIDA fragments [35] was based on sequences of atoms and bonds of size from 2 to 4 atoms. Additionally, bonds within a cycle in a fragment were marked with a special symbol.

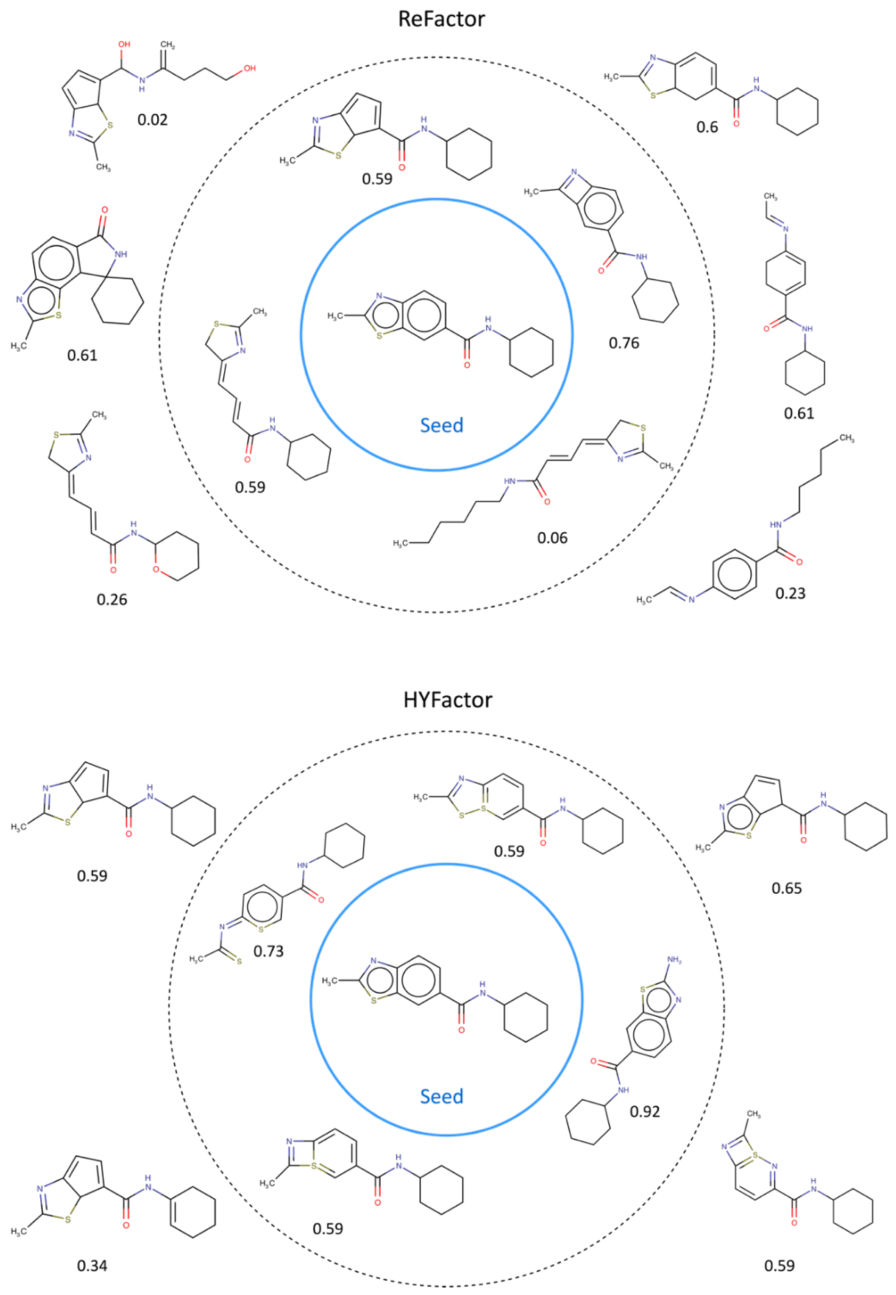

Figure 5. Example of structures generated with ReFactor and HyFactor trained on the MOSES set. Molecules generated with standard deviation < 0.6 (> 0.6) lie within (outside) the dashed circle. Each number corresponds to pairwise Tanimoto similarity of a given generated structure with respect to the seed assessed with the ISIDA descriptors.

From Figure 5 it can be seen, that the sampled structures are similar to the seed molecule. It further proves that even if the latent spaces of the proposed Autoencoders are discrete, the arrangement of molecules in them is reasonable. However, in some cases there are some structures dissimilar to the seed molecule that can be generated with a small standard deviation, and vice versa, rather similar structures can be found in the list of molecules generated with a large standard deviation.

Application on ChEMBL data set

The Achilles' heel of graph-based Autoencoders is the reconstruction of molecules with a large number of atoms. Indeed, as the size of a molecule grows, the probability of error in the type of atom or bond increases. Therefore, in the next phase of experiments, the reconstruction ability of Autoencoders was studied on the ChEMBL data set containing molecules bearing up to 50 heavy atoms. The results of training are given in Table 5 and the specifications on training parameters can be found in Table S1.

Table 5. Training results on the ChEMBL data set.

Architecture	Batch	Vector length	Number of training parameters, M		Time per epoch*, min	GPU Memory*, MB	Reconstruction rate, %	
			Encoder	Decoder			Training set	Validation set
ReFactor	1024	1024	35.7	14.8	~ 24.3	~ 22 845	99.8	95.2
HyFactor	1024	1024	25.3	15.1	~ 16.5	~ 16 755	99.7	95.0

* - measured in "mixed precision" mode, available in TensorFlow package. In this mode 16-bit floating-point type is used where it's possible, otherwise 32-bit floating-point type is applied.

According to Table 5, HyFactor uses 20% fewer training parameters than ReFactor in order to achieve a similar reconstruction rate, and, thus, its training is 33% faster than ReFactor. Although the overall reconstruction rate is high enough, the reconstruction error sharply increases for molecules containing more than 35 atoms and it reaches almost 30% for ReFactor and HyFactor for molecules containing 50 atoms (Figure 6a). The latter can be explained by small number of heavy molecules present in the training set (see Figure 1).

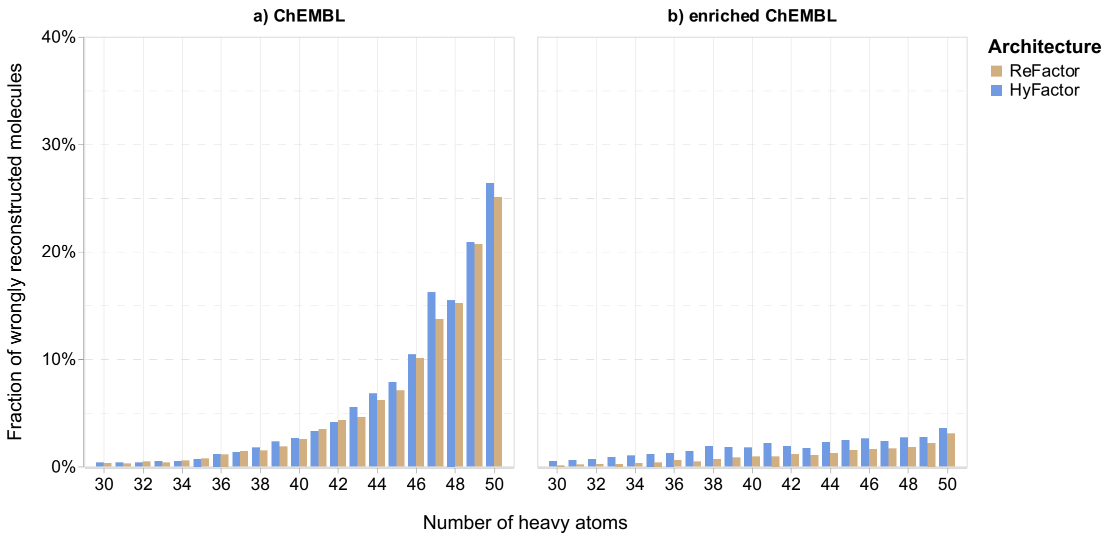

Figure 6. Distributions of errors in molecules from the **a)** ChEMBL and **b)** E_ChEMBL validation sets and for ReFactor (gold) and HyFactor (blue) architectures trained on **a)** ChEMBL and **b)** E_ChEMBL training sets as a function of molecular size.

In order to confirm these suggestions, 460K virtual structures containing >35 atoms have been generated using the SynthI tool [36] and then added to the ChEMBL set. The enriched ChEMBL set (from now on referred to as E_ChEMBL) was then divided into training and validation sets in the ratio 4:1 (1.6M and 400K of molecules) respectively. Distribution of molecular size in the obtained data set is given in Figure S3 of Supporting Materials. Both architectures trained on the E_ChEMBL training set achieved reconstruction rates above 95% measured on the E_ChEMBL validation set. As a result, the enrichment of the ChEMBL data set allowed to significantly decrease the reconstruction error for heavy molecules (Figure 6b).

By adding new molecules, one can expect the model to reconstruct molecules from the original ChEMBL set more accurately. The reconstruction rate of molecules from the original ChEMBL validation set was measured with Autoencoders trained on the E_ChEMBL training set to confirm this assumption. The results can be seen in Figure 7.

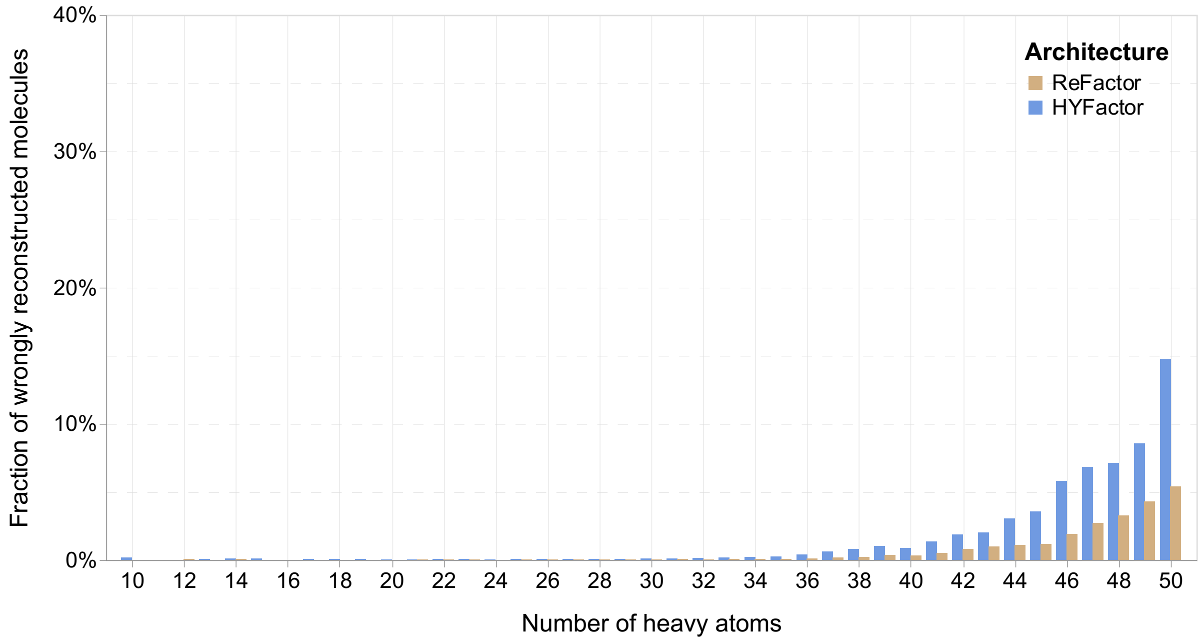

Figure 7. Distributions of errors in molecules from the ChEMBL validation set and for the models trained on the E_ChEMBL training set as a function of molecular size.

Comparing to Figure 6a, the decrease in reconstruction error of molecules bigger than 35 heavy atoms is dramatic both for ReFactor and HyFactor. The HyFactor has bigger errors in reconstruction, but the overall error is the same as for ReFactor (Figure 6b). This fact supports the assumption that the added molecules are helpful in learning ones from the original set. In conclusion, such combinatorial oversampling can be useful for generating novel structures and training robust models.

Conclusions

In this paper two graph-based Autoencoder architectures for molecule encoding and generation are presented. One of the architectures, named ReFactor, is an updated version of the previously published DEFactor architecture. Another Autoencoder, HyFactor, inherits approaches of ReFactor and is based on a new representation of the molecule, the hydrogen-count labelled graph.

Both architectures showed the highest performance in reconstruction ability on the ZINC250K dataset comparing to existing SMILES- and graph-based Autoencoders. The standardization of ZINC250K dataset allowed for a more accurate assessment of graph-based model performance.

Moreover, HyFactor and ReFactor were successfully tested in learning of molecules with up to 50 heavy atoms from the ChEMBL data set. While the HyFactor architecture achieved the same reconstruction rate as ReFactor, it was more effective in terms of number of network parameters and training time. The analysis of the results showed that the number of molecules larger than 35 heavy atoms in the ChEMBL dataset was insufficient for the models to learn them. To solve this problem, virtual molecules were added which improved the learning of large molecules.

Trained on MOSES dataset, both Autoencoders were used to systematically explore the chemical space around given molecules. From the calculation of MOSES metrics on the generated structures it was concluded that both architectures were able to generate diverse structures with various scaffolds. Although the validity and uniqueness of the generated structures was low, the autoecnoders outperformed previous architectures in terms of similarity metrics as well. To improve the validity of the generated structures, one of the future research directions might be the use of generative neural networks for latent vectors sampling.

In conclusion, although these architecutres have some shortcomings, their potential is very high. By providing the open-source code, we hope that our developments will accelerate research in this area.

Acknowledgments

Tagir Akhmetshin thanks the Region Grand Est for the PhD fellowship. Yuliana Zabolotna, postdoctoral researcher of the Chemoinformatics Laboratory in Strasbourg, is acknowledged for the help with combinatorial generation of molecules using SyntI tool.

References

1. Varnek A, Baskin I (2012) Machine learning methods for property prediction in chemoinformatics: Quo Vadis? J Chem Inf Model 52:1413–1437 . https://doi.org/10.1021/ci200409x
2. Button A, Merk D, Hiss JA, Schneider G (2019) Automated de novo molecular design by hybrid machine intelligence and rule-driven chemical synthesis. Nat Mach Intell 1:307–315 . https://doi.org/10.1038/s42256-019-0067-7
3. Segler MHS, Preuss M, Waller MP (2018) Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555:604–610 . https://doi.org/10.1038/nature25978
4. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent Sci 4:268–276 . https://doi.org/10.1021/acscentsci.7b00572
5. Baskin II (2020) The power of deep learning to ligand-based novel drug discovery. Expert Opin Drug Discov 15:755–764 . https://doi.org/10.1080/17460441.2020.1745183
6. Simonovsky M, Komodakis N (2018) GraphVAE: Towards generation of small graphs using variational autoencoders. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 11139 LNCS:412–422 .

https://doi.org/10.1007/978-3-030-01418-6_41

7. Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A (2020) Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. Mach Learn Sci Technol 1:045024 . https://doi.org/10.1088/2632-2153/aba947

8. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. 5th Int Conf Learn Represent ICLR 2017 - Conf Track Proc 1–14

9. Samanta B, De A, Jana G, Chattaraj PK, Ganguly N, Rodriguez MG (2019) NEVAE: A deep generative model for molecular graphs. 33rd AAAI Conf Artif Intell AAAI 2019, 31st Innov Appl Artif Intell Conf IAAI 2019 9th AAAI Symp Educ Adv Artif Intell EAAI 2019 33:1110–1117 . https://doi.org/10.1609/aaai.v33i01.33011110

10. Bresson X, Laurent T (2019) A Two-Step Graph Convolutional Decoder for Molecule Generation. arXiv

11. Zhou Z, Kearnes S, Li L, Zare RN, Riley P (2019) Optimization of Molecules via Deep Reinforcement Learning. Sci Rep 9: . https://doi.org/10.1038/s41598-019-47148-x

12. Jin W, Barzilay R, Jaakkola T (2019) Hierarchical Graph-to-Graph Translation for Molecules. 1–14

13. Assouel R, Segler MH, Ahmed M, Saffari A, Bengio Y (2018) DEFactor: Differentiable edge factorization-based probabilistic graph generation. arXiv 1–14

14. Jin W, Yang K, Barzilay R, Jaakkola T (2018) Learning multimodal graph-to-graph translation for molecular optimization. arXiv 1–13

15. Kearnes S, Li L, Riley P (2019) Decoding molecular graph embeddings with reinforcement learning. arXiv

16. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Comput 9:1735–1780 . https://doi.org/10.1162/neco.1997.9.8.1735

17. Heller SR, McNaught A, Pletnev I, Stein S, Tchekhovskoi D (2015) InChI, the IUPAC International Chemical Identifier. Journal of Cheminformatics

18. Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, Golovanov S, Tatanov O, Belyaev S, Kurbanov R, Artamonov A, Aladinskiy V, Veselov M, Kadurin A, Johansson S, Chen H, Nikolenko S, Aspuru-Guzik A, Zhavoronkov A (2020) Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. Front Pharmacol 11:1–10 . https://doi.org/10.3389/fphar.2020.565644

19. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B, Overington JP (2012) ChEMBL: A large-scale bioactivity database for drug discovery. Nucleic Acids Res 40:1100–1107 . https://doi.org/10.1093/nar/gkr777

20. ChemAxon (2020) JChem

21. Kusner MJ, Paige B, Hemández-Lobato JM (2017) Grammar variational autoencoder. 34th Int Conf Mach Learn ICML 2017 4:3072–3084

22. Pocha A, Danel T, Maziarka Ł (2020) Comparison of Atom Representations in Graph Neural Networks for Molecular Property Prediction

23. Nugmanov RI, Mukhametgaleev RN, Akhmetshin T, Gimadiev TR, Afonina VA, Madzhidov TI, Varnek A (2019) CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing. J Chem Inf Model 59:2516–2521 .

https://doi.org/10.1021/acs.jcim.9b00102

24. Simonovsky M, Komodakis N (2017) Dynamic edge-conditioned filters in convolutional neural networks on graphs. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua:29–38 . https://doi.org/10.1109/CVPR.2017.11

25. Zitnik M, Agrawal M, Leskovec J (2018) Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics 34:i457–i466 . https://doi.org/10.1093/bioinformatics/bty294

26. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP 2014 - 2014 Conf Empir Methods Nat Lang Process Proc Conf 1724–1734 . https://doi.org/10.3115/v1/d14-1179

27. He K, Zhang X, Ren S, Sun J (2015) Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, pp 1026–1034

28. Zhuang J, Tang T, Ding Y, Tatikonda S, Dvornek N, Papademetris X, Duncan JS (2020) AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. arXiv 1–29

29. Sattarov B, Baskin II, Horvath D, Marcou G, Bjerrum EJ, Varnek A (2019) De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. J Chem Inf Model 59:1182–1196 . https://doi.org/10.1021/acs.jcim.8b00751

30. Yan C, Wang S, Yang J, Xu T, Huang J (2020) Re-balancing Variational Autoencoder Loss for Molecule Sequence Generation. Proc 11th ACM Int Conf Bioinformatics, Comput Biol Heal Informatics, BCB 2020. https://doi.org/10.1145/3388440.3412458

31. Alperstein Z, Cherkasov A, Rolfe JT (2019) All SMILES variational autoencoder. arXiv

32. Dai H, Tian Y, Dai B, Skiena S, Song L (2018) Syntax-directed variational autoencoder for structured data. arXiv 1–17

33. Kingma DP, Welling M (2014) Auto-encoding variational bayes. 2nd Int Conf Learn Represent ICLR 2014 - Conf Track Proc 1–14

34. Winter R, Montanari F, Noé F, Clevert D-A (2019) Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. Chem Sci 10:1692–1701 . https://doi.org/10.1039/C8SC04175J

35. Varnek A, Fourches D, Horvath D, Klimchuk O, Gaudin C, Vayer P, Solov'ev V, Hoonakker F, Tetko I, Marcou G (2008) ISIDA - Platform for Virtual Screening Based on Fragment and Pharmacophoric Descriptors. Curr Comput Aided-Drug Des 4:191–198 . https://doi.org/10.2174/157340908785747465

36. Zabolotna Y, Volochnyuk DM, Ryabukhin S V, Gavrylenko K (2021) SynthI : a new open-source tool for synthon-based library design. 1–16

37. VanderPlas J, Granger B, Heer J, Moritz D, Wongsuphasawat K, Satyanarayan A, Lees E, Timofeev I, Welsh B, Sievert S (2018) Altair: Interactive Statistical Visualizations for Python. J Open Source Softw 3:1057 . https://doi.org/10.21105/joss.01057

38. Satyanarayan A, Moritz D, Wongsuphasawat K, Heer J (2017) Vega-Lite: A Grammar of Interactive Graphics. IEEE Trans Vis Comput Graph 23:341–350 . https://doi.org/10.1109/TVCG.2016.2599030

Supplementary information

I. ChEMBL data set analysis

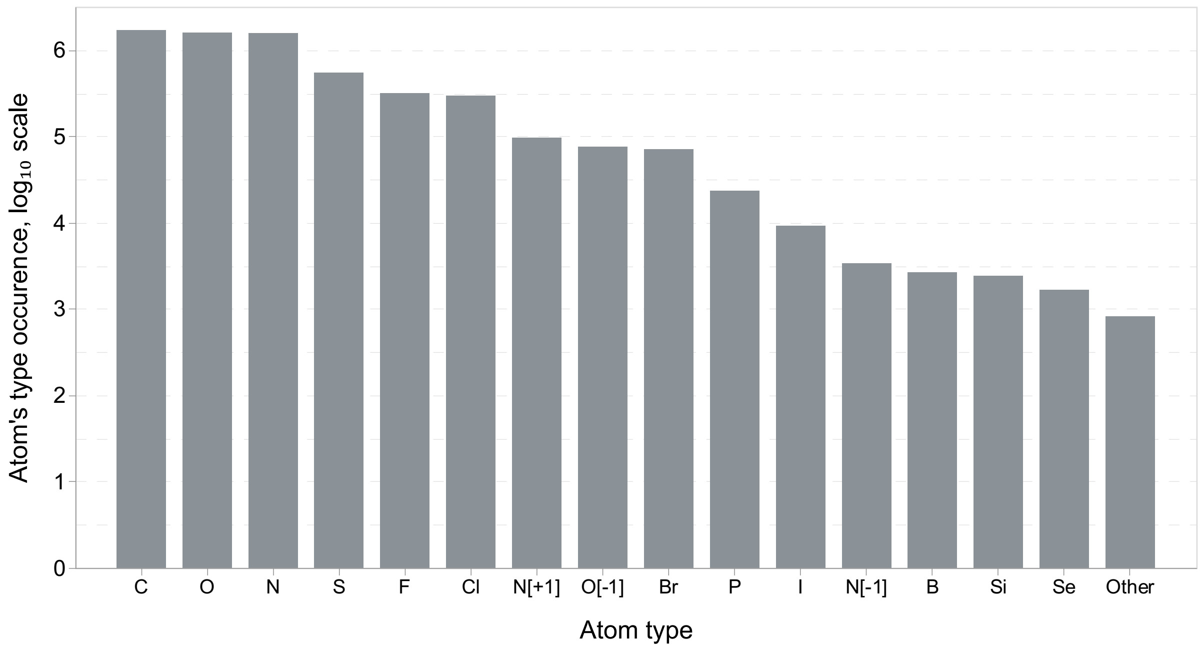

Figure S1. Distribution of molecules from standardised ChEMBL v. 27 as a function of atoms' types presence.

II. Training parameters

Table S1. Training parameters for each data set. The parameters are equal for both ReFactor and Hyfactor architectures.

Dataset	Latent dimension	Batch	Initital learning rate	Max number of atoms	Number of epochs	Number of atom types	Number of training parameters	
							ReFactor	HyFactor
ZINC 250K	512	256	0.001	39	100	15	12M	10M
MOSES	512	1024	0.001	28	100	7	12M	10M
ChEMBL & Enriched ChEMBL	1024	1024	0.0008	50	150	15	50M	40M

All calculations were made with NVIDIA QUADRO RTX 6000 GPU with CUDA drivers 11.2. The version of Tensorflow was 2.6. All preprocessing, including transformations of molecular graphs to matrix representation, was done with CGRtools version of 4.1.33. The charts were done with Altair Python package [37, 38].

III. MOSES benchmarking

The MOSES benchmarking is a distribution learning bencmarking. The main goal is to create generative architecture, that will approximate distribution of real drug-like molecules over known distribution. In the best case, the model should generate valid, unique and novel structures, while the distribution of the generated molecules should be almost the same as real ones.

To measure the quality of learned distribution, several metrics are considered in this tool:

- Validity – a fraction of valid molecules compared to generated ones;
- Unique 10K – a fraction of 10K first unique molecules from the valid ones;
- Novelty - a fraction of novel generated molecules from unique ones;
- Frag (fragment similarity) – a cosine similarity based on the occurrence of BRICS fragments in compared sets. Higher value means that both sets have similar fragments;
- Scaf (scaffold similarity) – a cosine similarity based on the occurrence of Bemis-Murcko scaffolds in compared sets. Higher value means that both sets have similar scaffolds;
- SNN (Similarity to a Nearest Neighbor) – an average Tanimoto similarity calculated on Morgan fingerprints between a molecule from the generated set and its nearest neighbor from the reference set;
- FCD (Fréchet ChemNet Distance) – a Wasserstein-2 distance computed on vectors produced by the last layer of ChemNet neural network between generated and reference sets. The lower values indicate that structures from compared sets should have similar biological properties;
- IntDiv (Internal Diversity) – an average Tanimoto distance between molecules in the generated set based on Morgan fingerprints;
- Filters – a fraction of molecules that pass MCFs (Medicinal Chemistry Filters) and PAINS (Pan-Assay Interfering compounds) medicinal filters.

The data used in MOSES benchmarking based on ZINC Clean Leads collection. After standardizing and filtering, 1.9M molecules with non-charged atom types such as C, N, S, O, F, Cl, Br remained. The cleaned set was split into training, test and scaffold test (TestSF) sets in ratio 9:1:1. In the scaffold test set, there were molecules with unique Bemis-Murcko scaffolds that were not present in the training and test sets.

Table S2. MOSES benchmarking results

Architecture	Valid	Unique	FCD		SNN		Frag		Scaf		IntDiv	Filters	Novelty
			Test	TestSF	Test	TestSF	Test	TestSF	Test	TestSF			
ReFactor 0.4 std	0,886	0,265	1.743	2.380	0.547	0.511	0.996	0.993	0.847	0.041	0.869	0.850	0,578
HyFactor 0.4 std	0,729	0,267	0.365	0.871	0.614	0.566	0.999	0.998	0.862	0.003	0.860	0.959	0,129

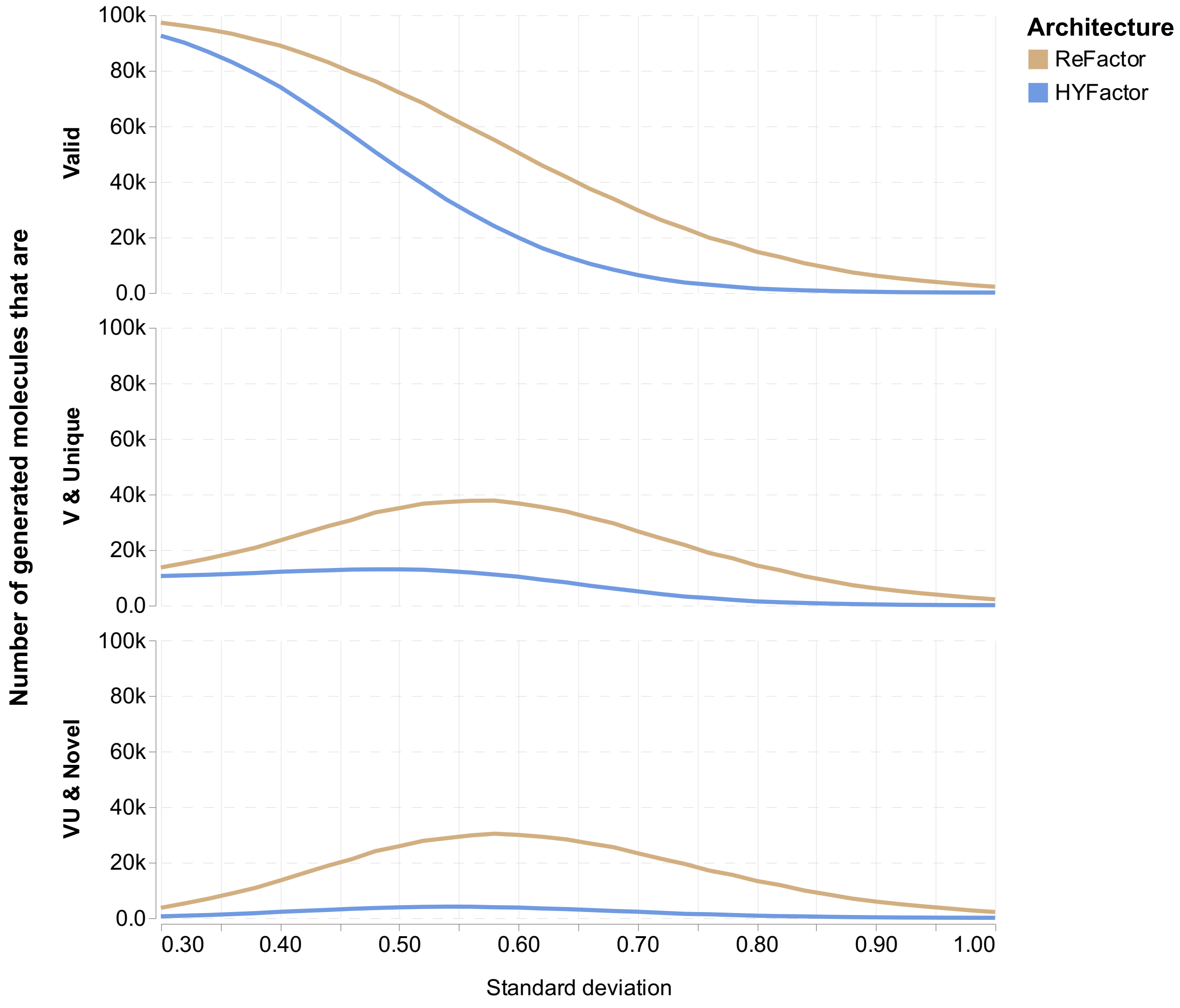

Figure S2. Results of generation with variable standard deviation in range from the 0.3 to 1.0 with the step of 0.02. V corresponds to Valid, U corresponds to Unique.

IV. Enriched ChEMBL experiments

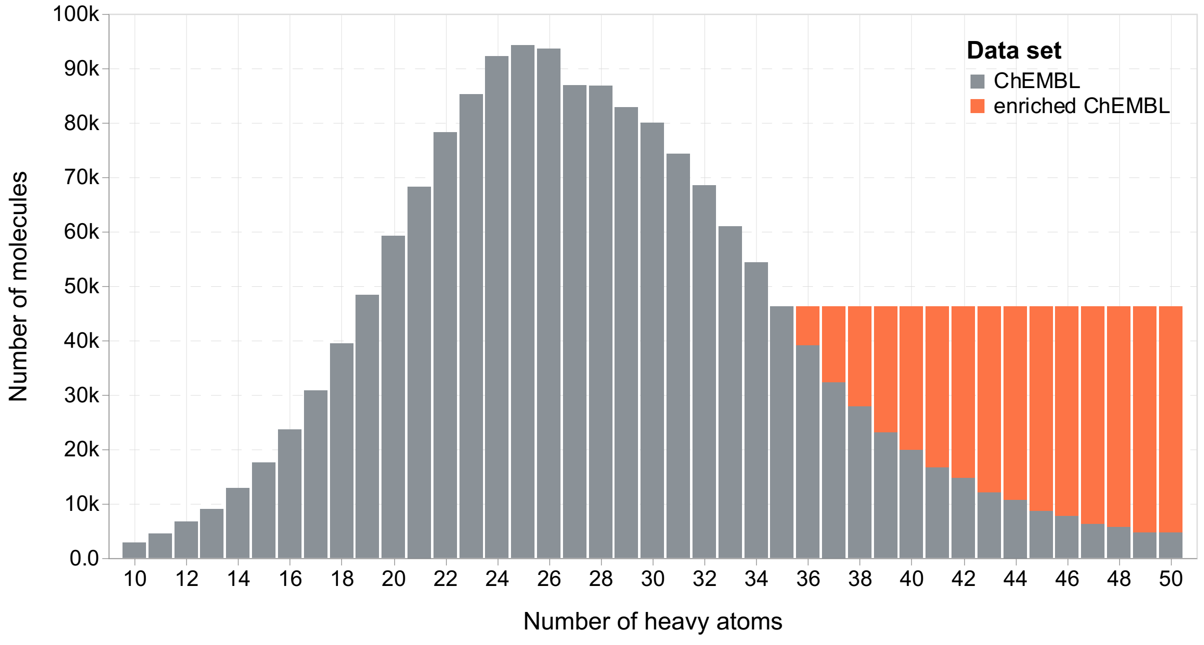

Figure S3. Heavy atoms count distribution in the ChEMBL (gray) and enriched ChEMBL (red) data sets.

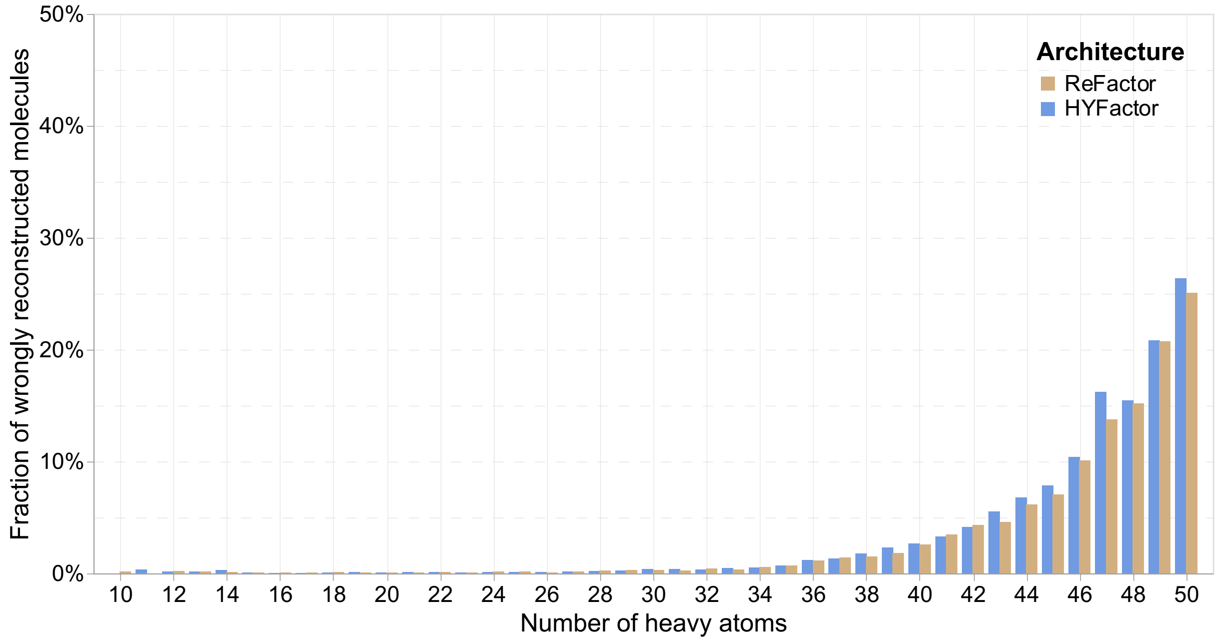

Figure S4. Distributions of errors in molecules from the ChEMBL validation set for ReFactor (gold) and HyFactor (blue) architectures as a function of molecular size.

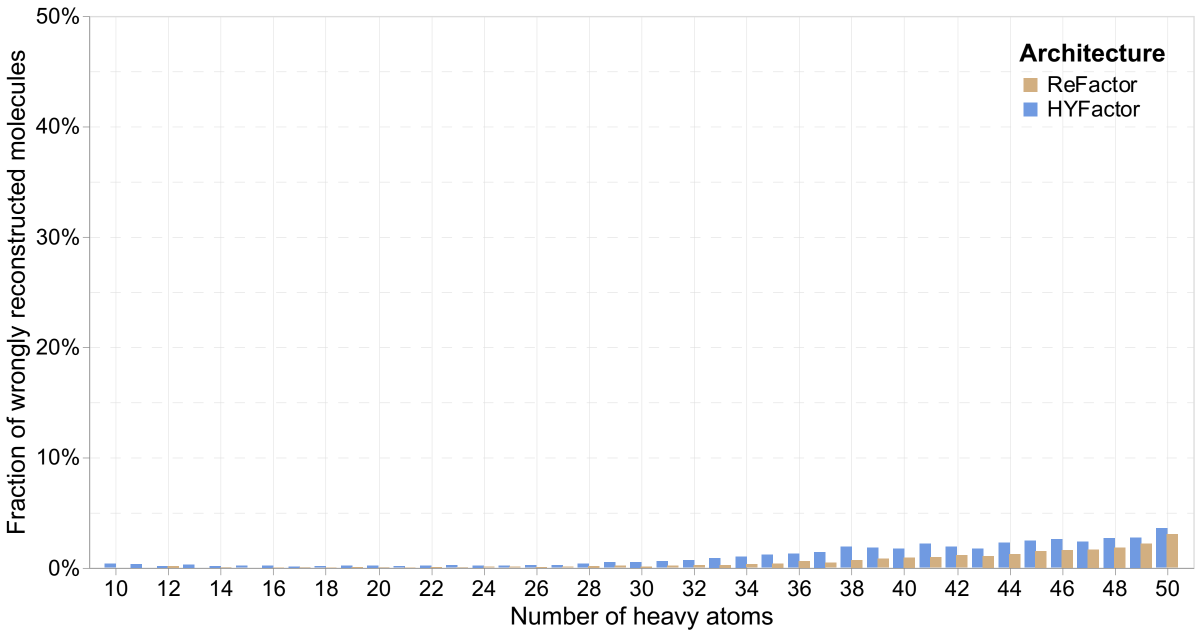

Figure S5. Distributions of errors in molecules from the ChEMBL validation set for ReFactor (gold) and HyFactor (blue) architectures as a function of molecular size.