


Toward efficient generation, correction, and properties control of unique drug-like structures

Maksym Druchok^{1,2}  | Dzvenymyra Yarish¹ | Oleksandr Gurbych¹ | Mykola Maksymenko¹

¹SoftServe, Inc, Lviv, Ukraine

²Institute for Condensed Matter Physics, Lviv, Ukraine

Correspondence

Maksym Druchok, SoftServe, Inc, 2d Sadova Str., Lviv 79021, Ukraine.

Email: maksym@icomp.lviv.ua

Abstract

Efficient design and screening of the novel molecules is a major challenge in drug and material design. This paper focuses on a multi-stage pipeline, in which several deep neural network models are combined to map discrete molecular representations into continuous vector space to later generate from it new molecular structures with desired properties. Here, the Attention-based Sequence-to-Sequence model is added to “spellcheck” and correct generated structures, while the oversampling in the continuous space allows generating candidate structures with desired distribution for properties and molecular descriptors, even for a small reference datasets. We further use computer simulation to validate the desired properties in the numerical experiment. With the focus on the drug design, such a pipeline allows generating novel structures with a control of Synthetic Accessibility Score and a series of metrics that assess the drug-likeness. Our code is available at <https://github.com/SoftServeInc/novel-molecule-generation>.

KEYWORDS

molecular design, machine learning, deep neural networks, autoencoder

1 | INTRODUCTION

Designing new materials is often a challenging, resourceful, and time-consuming endeavor.^{1–3} In particular, drug development is a step-wise, iterative process involving certain necessary stages: discovery and research, development, regulatory approval, only then followed by production and marketing. Typically, it takes from 10 to 15 years from initial idea to a final approval and commercial distribution.^{4,5} Such a timeline is mostly defined by difficulties with finding and evaluating appropriate drug candidates that will successfully pass clinical studies. The material science explored just a tiny piece of a space of potential compounds: it is estimated that only about 10^8 of small drug-like molecules⁶ have been synthesized so far out of more than 10^{60} possible ones.⁷ Today, synthesis of new candidates requires increased investments in R&D studies,⁸ mainly due to increased complexity of the search in this exponential space.

Lately, we witnessed the number of successful applications of machine learning (ML) in many domains.⁹ While most of the interest

comes from advances in deep neural networks (DNNs) for speech recognition and computer vision, the universality of these methods in learning the representations of data allows sufficient widening of the areas of potential impact, including applications to scientific problems.^{10,11}

In this respect, chemical and pharmaceutical industries reveal a substantial interest in ML approaches aiding on the increase in efficiency of novel materials design.^{12–14} As a new milestone reached, it is worth mentioning a recent study by Zhavoronkov et al¹⁵ reporting an ML-driven development of inhibitor candidates in just 21 days, which unprecedentedly shortens the preclinical phase.

1.1 | ML for material design

The material science operates on different levels of abstraction, from the atomic scale to more coarse-grained ones, such as classical

molecular or thermodynamic scales.¹⁶ This implies a different set of computational and theoretical tools being used at each of them. Here, recent advances in ML allow tackling multiple problems from the approximation of complex functions or quantum wave-functions to the prediction of properties, phase transitions, and temporal dynamics.^{17–24}

At the atomic level, DNNs are used to approximate quantum mechanical computations, such as atomic/molecular parameters in all-particle microscopic simulations,^{17,25} decomposing cluster energies or predict next simulation step instead of CPU-costly traditional routines, hence, speeding up the sampling.^{26–31} Recently, a few simulation packages already incorporated these approaches into their computational tools.^{32–37}

At the physical chemistry level, DNNs deal with the generalization of more coarse-grained features.¹⁰ Quantitative structure-activity/property relationship (QSAR) models relate structure descriptors of compounds with their physical/chemical properties, such as solubility in water or organic solvents, melting point, solvation energies, and so on.^{38–40} The latter ability to use ML for linking structure with a property is interesting from biophysics and physiology perspectives to address drug-likeness and ADME (absorption, distribution, metabolism, excretion) or ADMET (if Toxicity is also considered) scores of a particular compound. In this case, the focus is on a set of properties predicting binding affinity, biodegradability, and toxicity of compounds.^{41–50} A kind of “golden standard” in drug-likeness screening is a so-called *the rule of five*, suggested by a study of Lipinski et al,⁵¹ or its close variations,^{52–55} allowing pharma-industry considerably shrink a screening pool of drug-candidates at the discovery stage.

In ML methods dealing with molecular structures, the crucial step is the efficient representation of the structural data. The above-mentioned approaches use different variations of input data: graphs, fingerprints, descriptors, mixed fingerprint-descriptor models (see, e.g., ref. [45]), one-line notations (such as SMILES—simplified molecular input line entry specification). Treating the latter string representations of molecules as a stand-alone language can unlock a substantial number of possible applications of natural language processing methods to chemistry-related problems, including the generation of new compounds.^{56,57} In refs. [58, 59], SMILES strings are converted into 2D structure images, and then fed into DNNs. A similar approach to use 2D structure images as inputs is also exploited in ref. [60].

Methods based on graph neural networks (see, e.g., recent reviews^{61,62} and the references therein) can directly exploit the graph representation of molecules. Such representation is a natural choice for studies of molecular structures, interactions, and synthesis.⁶³ In particular, graph convolution networks are under consideration in refs. [63, 64], where molecular graphs are used to predict solubility, toxicity, and other properties of compounds. Ref. [65] unites graph representations with adversarial training and reinforcement learning to guide the process of molecular design toward the outputs with desired properties. Ref. [66] uses graph neural networks for protein interface predictions. The study by Zitnik et al⁶⁷ engages graph convolutional networks to model polypharmacy side effects by examining

multirelational drug-drug links. A generative network MolGAN for molecular graphs is suggested in ref. [68].

Recent advances of Deep Learning to a large extent concern various applications of generative adversarial networks (GANs) and other examples of deep generative models able to generate or reconstruct data from a given distribution.^{69,70} In the context of molecular data, this opens a route for the synthesis of novel structures with given properties (see, e.g., a review by Jørgensen et al⁷¹). In refs. [60, 72], the autoencoder (AE) or variational autoencoder (VAE) architectures are used to map discrete SMILES strings into a continuous space. A scan in such space allows generating unseen structures that can be decoded back to SMILES strings. In recent studies, a number of other deep generative models were explored to improve the quality of continuous representations, reduce reconstruction errors, and assess the performance of different models.^{71,73,74} A study introducing generative adversarial AE is reported in ref. [75]—the authors tested different architectures for structure generation and inverse QSAR mapping (sampling new structures with applied activity constraints). The detailed discussion on the problem of “chemical space” and reconstruction from embeddings is presented in a recent study of Bjerrum and Sattarov.⁷⁶ A GAN-based model by Guimaraes et al.⁷⁷ adversarially learns to output SMILES strings, optimizing them toward chemical metrics.

Note that, despite the continuous nature of the AE latent space and infinite possibilities of choosing arbitrary latent vectors, not all vectors correspond to proper SMILES strings. Some of these vectors might decode to chemically incorrect SMILES strings, while others (even “grammatically” correct) may correspond to unstable compounds. A successful attempt to tackle this issue was made in ref. [57], where the plain VAE is replaced with the Grammar VAE. The main idea is to convert SMILES strings into parse trees from the predefined context-free grammar and train the model on them. While this setting substantially increases the number of valid SMILES strings in the output, it does not spot the errors, which could not be identified without context (i.e., when a ringbond is not opened and closed by the same digit, starting with “1”—like in benzene “c1ccccc1”⁵⁷). Ref. [78] extended the idea of SMILES grammar by enriching it with attributes which add some context awareness to the model.

1.2 | Focus of this study

In the above subsection, we reviewed a current state of ML-assisted approaches in cheminformatics, making an accent on the importance of latent space representation, the issues with incorrect structures, and property control of synthesized compounds, which are decoded from the points in the latent space. These issues may become particularly crucial when the reference datasets are small and the end-to-end learning of the full pipeline is not possible.

This paper presents a practical multi-stage pipeline in which several models are trained on specific tasks and large open datasets. We introduce a number of pre- and postprocessing steps to enhance the quality and success rate of generated molecules. Importantly, for the generation task, this pipeline allows working with comparably smaller

reference datasets. To this end, such a pipeline covers substantial part of the early-stage drug discovery, from proposing a new structure to predicting physical-chemical properties and confirming them in numerical simulation (see Figure 1).

First, we use a common AE architecture allowing to map discrete SMILES strings into continuous latent space.^{79,80} We show that the size of the dataset plays a crucial role in achieving higher generalization and reconstruction quality. As datasets with the labeled properties may often be of limited size, instead of end-to-end learning, we train separate models on a number of tasks: reconstruction of structures, error correction, and prediction of properties.

To increase the number of valid generated structures, we introduce an oversampling step allowing to generate new points in the latent space based on the limited number of the reference ones. An additional step includes the attention-based sequence-to-sequence model to correct errors in generated notations.

To assess the quality of resulting structures, we introduce a post-processing step allowing to compute properties of the generated molecules directly from the structure. These extra steps provide a comparative analysis over scaffolds, fingerprints, descriptors, and functional groups to assess the quality of the generated molecules. We confirm that generated unique SMILES have a similar distribution of structural features and molecular descriptors.

We also went further with such an assessment by performing a computer simulation at all-atom level. This simulation modeled a set of aqueous solutions of one of newly generated compounds at different concentrations. This part of the study is intended to be a computer experiment instead of the real one and serves as a sanity check of the aqueous solubility prediction. The general pipeline is shown in Figure 1. Below, we discuss the pipeline in more detail.

2 | METHODOLOGY

In the core of our architecture is the idea of having a continuous space to represent small organic molecules. To create such mapping, we use the AE network, consisting of two subnets—encoder and decoder, as shown in Figure 2. The mapped embedding space allows one to continuously change representation vector in order to sample new candidate molecular structures.

The predictor models take embedding vector as input to predict a set of metrics (further we discuss a prediction of aqueous solubility). Finally, we add an attention-based sequence-to-sequence model with long short-term memory cells to “spell-check” possible errors in the generated notations.

2.1 | Molecular representations

There exist various chemical representations helping encode spatial molecular structures by compact one-line notations; the most popular among them are SMILES. The SMILES form carries all the necessary information to calculate most of the drug-likeness metrics (H-bond donors, acceptors, molecular weight, etc.), except lipophilicity logP and aqueous solubility logS. A SMILES string itself cannot be fed into neural network and needs to be expressed in a numerical form. It is convenient to represent categorical data (like SMILES elements) via so-called one-hot encoding which in the case of SMILES is a matrix (N by M) of 0's and 1's. N is a size of dictionary of valid SMILES elements (like C, c, @, O, brackets, etc.), while M runs over the characters of the SMILES string. One such example is illustrated in Figure 3, where a spatial representation of ascorbic acid (also well known as

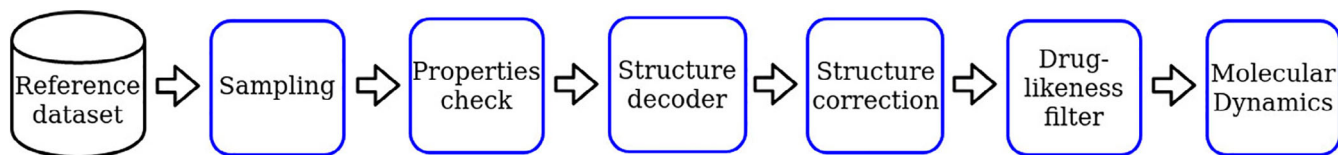


FIGURE 1 The proposed system covers multiple steps of the early-stage drug discovery. Datapoints are generated directly in continuous space with regards to the reference dataset with immediate assessment of properties. The SMILES is decoded, corrected, and screened by a drug-likeness filter and molecular dynamics simulation

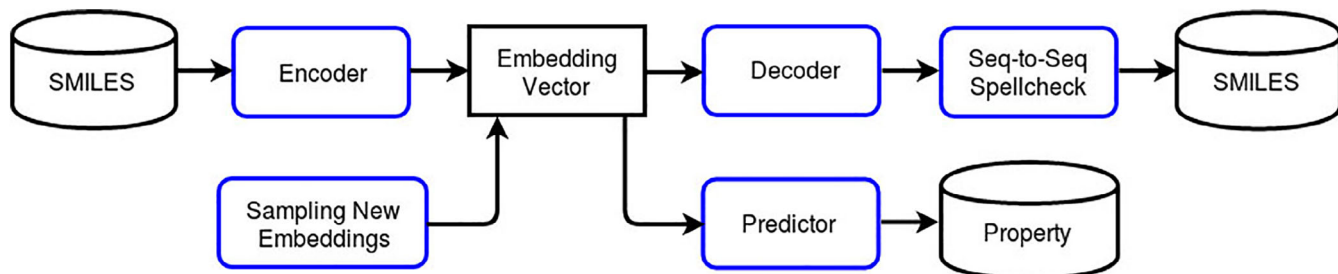


FIGURE 2 The system architecture consists of the generative part (encoder and decoder), the error correction model (Seq-to-Seq), a set of properties predictors, and the sampler of new datapoints in the embedding space

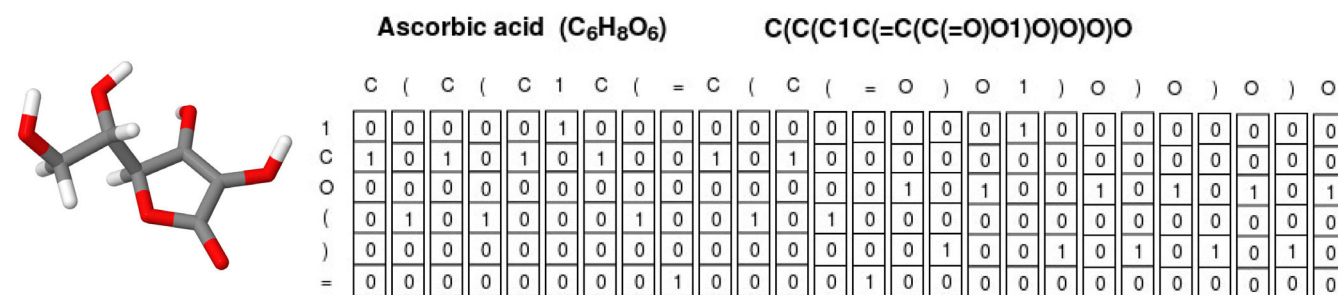


FIGURE 3 Ascorbic acid as molecular formula, SMILES string, one-hot matrix, and a spatial structure of a molecule represented by sticks. Carbons in the 3D representation are shown in gray, oxygens—in red, hydrogens—in white

vitamin C) along with its molecular formula, SMILES string, and one-hot matrix are shown. We consider the SMILES strings not longer than 60 elements applying zero-padding for the shorter ones. We also limited the dictionary size to only the elements encountered within the train and evaluation datasets (found 58 unique elements).

2.2 | What datasets are available

In this research, we use the eMolecules database⁸¹—a set of commercially distributed chemicals—as a large list of valid SMILES strings to train AE.

For the property prediction, we focus on the aqueous solubility and compiled our dataset from a series of open sets published by Huuskonen,⁸² Hou et al.,⁸³ Delaney,⁸⁴ and Mitchell.⁸⁵ Additionally, the overall solubility dataset was extended by transforming SMILES strings to a canonical form, which in total yielded ≈4300 solubility labels for SMILES strings not longer than 60 elements. All solubilities are presented in the form of the so-called log solubility—logS, which is the decimal logarithm of a maximal soluble concentration of solute in water, expressed in mol/L.

Public datasets often combine the data obtained in different laboratories by different techniques. This might sufficiently limit the quality of the data as a good training set. Regarding the solubility prediction case, it is instructive to review a solubility challenge and discussion of its results in refs. [86, 87].

2.3 | Model design

We decided to use a fully deterministic AE. The design of the AE was shaped during a series of grid-search experiments. The AE architecture was tuned by changing the number of layers (from 4 to 12) and their types (fully connected (FC) and convolutional). The number of neurons in each FC layer varied from 2 (at the bottleneck layer uniting encoder and decoder parts) to 3500 at the encoder/decoder hidden layers. The number of convolutional layers varied from 2 to 10 with different number and shapes of filters as well. The addition of convolutional layers improved the performance of the AE in comparison to the one based on purely FC layers. The experiments with activation

functions (sigmoid, ReLU, PReLU) favored ReLU, except the sigmoid activation at the output layer. We also tested the application of dropout, however, the optimal performance was achieved without it. Further, we discuss the final architecture of the AE (see Appendix and Figure A1 therein for the detailed architecture). The encoder, first part of the AE, consists of four convolution layers, followed by a FC layer. The output of the encoder is considered as a latent SMILES representation. The decoder, second part of the AE, is aimed to decode latent representations back to original SMILES strings. It, symmetrically, consists of an FC layer and four convolutional ones.

The predictor takes the latent representation as an input and is trained to match it against the aqueous solubility dataset. The predictor consists of four residual blocks and four FC layers as shown in Figure A2 in Appendix. The predictor's layers gradually decrease their size and the last one outputs a single number, which is considered as a solubility prediction.

We used the ADAM optimizer for training both the AE and predictor, adjusting only the learning rate within the range of $10^{-3} \div 10^{-5}$. In the case of the AE, optimization was aimed to minimize the loss function in the binary cross entropy form. The predictor's loss function is formulated as a mean square error. The AE is considered to be properly designed and trained if its inputs and outputs mutually coincide on a high rate, while the predictor's aim is to reproduce solubility. Notably, the above scheme allowed us to build other predictors on top of the existing latent space and predict other properties in parallel to the aqueous solubility.

2.4 | Oversampling in the continuous space for the new structures

The particular aim of this study is in the generation of new compounds. The above architecture allows operating with discrete SMILES notations in a continuous latent space giving infinite possibilities of choosing arbitrary vectors in this space. This may lead to either correct chemical structures, or structures that are chemically incorrect or difficult to synthesize. Keeping this in mind, we approach the problem not by a simple random selection of latent vectors, but by using “smarter” linear combinations of vectors belonging to existing (correct) structures.

Indeed, the random selection of vectors yields almost 100% faulty result. Therefore, we used the SMOTE (synthetic minority oversampling technique) approach⁸⁸ adopting its implementation from the Imblearn library.⁸⁹ SMOTE generates new synthetic data by selecting pairs of original samples close in the feature space and picking random points along the lines between these paired samples.

After the latent vectors for new structures are sampled from the continuous space, they need to be decoded to SMILES strings. The decoder outputs are in the one-hot encoding form and are converted to SMILES strings, however, an additional postprocessing is needed over the newly generated strings: many of them might not be chemically correct. As a harsh recipe one can test SMILES strings against the RDKit library⁹⁰ and drop the incorrect ones. However, to increase the sampling rate, we introduced an error correction scheme, which is discussed next.

2.5 | Error correction for generated SMILES strings

In the experiments by Gómez-Bombarelli et al.,⁷² from 70% to less than 1% of the generated SMILES strings correspond to valid molecules. Therefore, one would have to overgenerate, to obtain a certain number of valid new molecules. This becomes crucial when a limited number of molecules with desired properties are at one's disposal to use as the base for SMOTE. Thus, to efficiently exploit the earlier described pipeline, we need to find a way to increase the number of valid SMILES strings in the AE output.

The problem of incorrect Autoencoder outputs can be decomposed into two subproblems. First, the latent space learned by the AE is too sparse and heterogeneous, so that the latent vector chosen for direct decoding most likely represents an invalid SMILES string. It could be solved by enforcing certain restrictions on the latent space, for example, by training an additional neural network to classify latent vectors into those which would decode into valid molecules and those which would not. Second subproblem roots in the brittle nature of string representations—one incorrect symbol could lead to a completely different molecule, while one misplaced probability in the output from the final layer of the decoder would not affect the loss function considerably. We chose to focus on the latter part which by its essence resembles a spell-checking task in natural language processing. The conventional spelling correction algorithm is not applicable in the case of newly generated SMILES strings—there is neither a full vocabulary of all correct strings nor a table with the most common errors in generated SMILES strings along with probabilities, which are required for the conventional spell checking algorithm. Thus, we decided to turn to ML and train a neural network which would be capable of syntactic error corrections in SMILES strings and could be used as a postprocessing to the AE outputs.

2.6 | Error correction model

Error correction in SMILES strings can be framed as a standard sequence-to-sequence learning problem, which is traditionally solved

with attention-based encoder-decoder models.^{91,92} To exploit the sequential nature of strings, both the encoder and decoder are built of LSTM (long short-term memory)⁹³ cells with the hidden state of size 512. The encoder transforms the input SMILES string X into a sequence of hidden states $(h_1, h_2, \dots, h_{|X|})$, while the decoder generates one symbol from SMILES characters dictionary at a time in the output SMILES string \hat{Y} . Formally, the model learns transitions

$$a : X \Rightarrow F^{512}, b : F^{512} \Rightarrow \hat{Y}$$

such that $a, b = \operatorname{argmin}(Y - b(a(X)))^2$, where Y is a target SMILES string. The choice of \hat{y}_t is conditioned on the previous symbol \hat{y}_{t-1} and on the compressed X in the form of dynamically created context vector c_t . It is computed as a weighted sum of the encoder's hidden states h_i :

$$c_t = \sum_{i=1}^{|X|} a_{ti} h_i$$

whose weights a_t are found by an attention mechanism:

$$a_{ti} = \operatorname{softmax}(e_{ti}), e_t = A(\hat{y}_{t-1}, s_{t-1}),$$

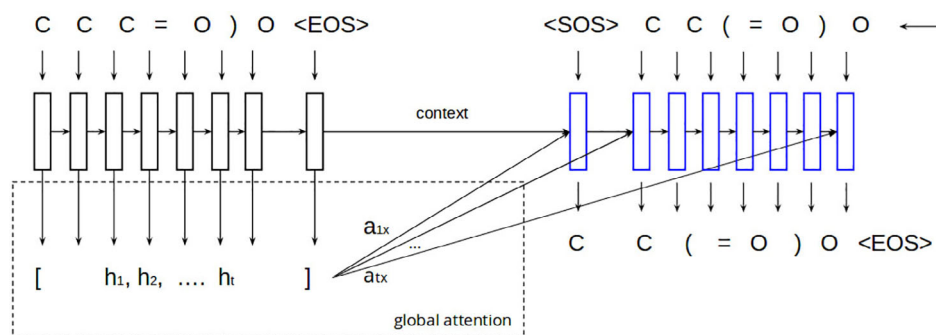
where A is an alignment model, implemented as feed-forward neural network of one fully-connected layer, and s_{t-1} is the previous hidden state of the decoder. A is jointly trained with the rest of the model.

Following the best natural language processing practices, symbols from SMILES strings are converted into embeddings,⁹⁴ learned by the trainable embedding layer in the model. The model is trained by minimizing the negative log-likelihood between the predicted SMILES string \hat{Y} and the (correct) target SMILES string Y . The training process runs for 70 epochs, making use of teacher forcing,⁹⁵ which suggests using the real target outputs as each next input during training. The alternative is using the decoder's own guess as the next input. Those two approaches are alternated during the training. Figure 4 shows the schematic architecture and training procedure of the error correction model.

2.7 | Error correction data

Since the initial intention was to correct errors made at the decoding stage, we collected all SMILES strings produced by the AE and tagged by the RDKit library as invalid, and the corresponding original valid strings. This gave us the dataset for the AE error correction, consisting of $\approx 300,000$ SMILES pairs. Obviously, the dataset formed in the way described above is biased and incomplete, and the model trained on it would make little use for other SMILES-related tasks. Inspired by the successful application of denoising AE⁹⁶ to feature extraction, we also experimented with SMILES strings with added random noise. By random noise we mean operations of replacement, deletion and insertion of random symbols from the dictionary in the input SMILES string following a predefined distribution. In that case, the model's input is a

FIGURE 4 The architecture of the attention-based sequence-to-sequence error correction model. The unrolled encoder is pictured with black rectangles, the unrolled decoder – with blue



SMILES string corrupted with random noise and the model has to learn how to convert them to the original valid string. That approach encouraged the model to construct more robust hidden representations of molecules and learn a SMILES language model, which would be closer to the real one. Thus, filling the gaps in the AE knowledge of the SMILES language model and correcting errors, caused by them.

We prepared two datasets with different nature of errors in SMILES strings: first, with random noise, the second one with AE generated errors. Two attention LSTM models were trained on these datasets correspondingly. Further on, for the sake of brevity, we will refer to those two models as random noise model and AE noise model.

2.8 | Molecular dynamics details

Besides the quality assessment of the predictor on the solubility task, we performed a series of molecular dynamics simulations for one of newly generated compounds—COCC(O)C(CC)CO. This part of the study is intended to test its predicted logS value via the computer experiment. The choice of the test compound is motivated by two arguments. First, its chemical structure is relatively simple, thus, one can expect that typical interaction forcefields describe it appropriately. Second, a moderate solubility of this compound allows us to simulate both concentration ranges (below and above the solubility limit) without extensive CPU-loads.

We prepared five mixtures of water molecules and molecules of the chosen compound at different concentrations. The water was represented within the SPC/E model,⁹⁷ while the solute molecules were constructed within the OPLS-AA forcefield.⁹⁸ The Lennard–Jones parameters for unlike sites were calculated by using the Lorentz–Berthelot mixing rules. All particles were allowed to move freely across the cubic unit cell with the periodic boundary conditions applied. The cell sizes $L_x = L_y = L_z$ varied within the range of 67–93 Å, depending on the solution composition.

The molecular dynamics simulations were performed with the use of the DL_POLY package.⁹⁹ In all simulations, we used the NPT ensemble with the pressure of 1 bar and temperature of 298 K controlled by the Nose–Hoover barostat and thermostat in the Melchionna's implementation.¹⁰⁰ The long-range Coulomb interactions were treated within the smooth particle mesh Ewald technique, while for the short-range interactions, the cut-off distance of 9 Å was

introduced. The equations of motion were integrated within the standard leapfrog scheme with a time step of 0.002 ps.

3 | RESULTS

3.1 | AE loss convergence

First, we ran a few tests to study how the AE convergence depends on the size of the train dataset. For this purpose, we prepared train datasets consisting of $\approx 10,000$, 50,000, 100,000, 200,000, 500,000, and 1,000,000 SMILES strings. The size of the test dataset of 20,000 entities remained unchanged over this benchmark. The summary plot with the dependence of the losses on the train dataset size is shown in Figure 5. The values for 500 k and 1 M cases show a relatively small difference in the train-test gap. Hence, we further decided to use a train dataset with 500 k entities. After the AE is trained, it constitutes the latent space and we can proceed with the generation and error-correction of new SMILES strings.

3.2 | Error correction evaluation

The error correction models were evaluated on three types of input data (all taken from the previously unseen by models test sets with the size of $\approx 15,000$ SMILES pairs): valid SMILES strings—to check how

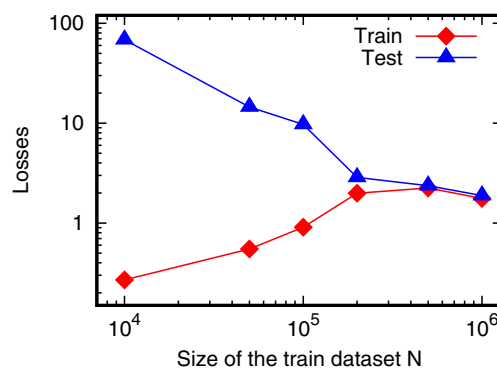


FIGURE 5 The autoencoder loss dependence on the train dataset size

accurate the model reconstructs the input strings; SMILES strings with AE noise and SMILES strings with random noise—to test the model's ability to correct errors. The results are expressed by two numbers:

1. How many output SMILES strings coincide with inputs in case of the reconstruction task or with original uncorrupted strings in case of error correction task.
2. How many strings outputted by the model are valid as verified by the RDKit library.

The performance of models is summarized in Table 1.

The model trained on AE noise seemingly learned the types of AE errors and the easiest way to correct them—by building direct mappings, not determining the underlying rules which tell how to construct valid strings. AE errors are deeply incorporated into the SMILES' structure itself and in dependencies between the symbols, therefore, the underlying language model is much skewed. The model trained on AE errors only is not capable of grasping the real language model from the valid SMILES strings, because the two language models are too different.

3.3 | Statistics on generated SMILES strings

For the generation purposes, we randomly selected a subset of $\approx 190,000$ SMILES entities from the eMolecules set,⁸¹ converted them into the corresponding latent vectors, and ran the SMOTE algorithm to produce 95,446 new latent vectors. Not all of the generated vectors corresponded to grammatically correct SMILES strings. Then, 39.7% (37890) of them were discarded by chemical parsing with RDKit due to unrealistic aromatic systems or incorrect atomic valences. 60.3% (57556) of the generated strings led to correct molecules.

Chemically incorrect generated SMILES strings were then revised and fixed by the error correction models. Out of 37,890 discarded strings, 12,040 (31.8%) were corrected successfully.

One of the important questions here was related to the novelty of the generated and corrected molecules. Almost all corrected SMILES strings were unique (99.8% or 12,024 out of 12,040). Out of 57,556 valid molecules generated by the AE, 89.8% (51,720) were identical to molecules in the source dataset. They were excluded from the target data analysis as we are interested only in a novel chemical matter from the same distribution. As a result, there were 17,860 unique novel molecules in total—5836 from the AE (*generated* dataset) and 12024 from the error correction models (*corrected* dataset).

TABLE 1 Performance comparison of error correction models

Task	Random noise model		AE noise model	
	Same as target [% of SMILES strings in the model output]	Valid	Same as target	Valid
Reconstruction	87	93	37	58
Random errors	66	83	16	36
AE errors	14	44	43	68

Unique generated and corrected compounds were searched within the ChEMBL¹⁰¹ and PubChem¹⁰² databases to verify their novelty. About 95% of generated and 98% of corrected structures were not found in ChEMBL. The search in PubChem yielded a lower but still solid fraction of novel structures: 72% of generated and 81% of corrected SMILES strings were not encountered in this database. These numbers demonstrate a high level of uniqueness of the compounds obtained.

Overall, based on the above numbers, the application of two error correction models (trained on AE noise and on random noise) increased by 20% the number of valid SMILES strings initially generated with the SMOTE technique. Taking into consideration the fact that out of all correctly generated SMILES strings 10.2%(5836) were unique, 12,040 unique corrected strings accounted for 67% for all newly generated samples. So error-correction models do not simply transform all invalid SMILES strings to the closest valid ones, which were seen by them during training.

Worth mentioning that two models do not always correct the same SMILES strings. Therefore, we can justify the use of two models for error correction, where the first knows how to fix very specific types of errors while the second is capable of more substantial and “creative” changes in invalid strings.

3.4 | Structural similarity and synthetic accessibility

3.4.1 | Scaffold analysis

The set of 5836 newly generated structures contained 3945 unique scaffolds (a scaffold is a part of a molecule after removal of nonring substituent, it is the largest chain for molecules without rings). For comparison, the source dataset of 189,936 molecules used for the SMOTE sampling procedure contained 58,229 scaffolds. The overlap between the generated and the source datasets was 2558 scaffolds (64.8% of generated scaffolds); the overlap between generated and corrected sets was 742 scaffolds (8.8% of corrected or 18.8% of generated scaffolds); the overlap between corrected and source datasets was 2594 scaffolds (30.8% of corrected scaffolds). These numbers illustrate that the protocol generates novel molecules and corrects erroneous ones within a similar distribution and does not mirror the sampling set too closely.

3.4.2 | Substructure features comparison

We calculated the following substructural features for the three datasets: number of rings in a molecule, presence of spiro rings, heterocycles, some biogenic elements, and halogens. The strings generated by the AE have the features distribution similar to those of the source class, although not exactly the same. Also, the strings, corrected by the Sequence-to-Sequence model, show the distribution close to the ones of the source and generated datasets, although having some specific differences, such as higher amount of structures

with one and two rings and lower amount of structures with multiple rings and heterocycles. Substructure features in dataset percentages are compared in Table 2.

TABLE 2 Comparison of substructure features between source, generated, and corrected class molecules

Feature	Source, %	Generated, %	Corrected, %
No rings	1.66	2.65	2.06
1 ring	7.74	7.88	9.04
2 rings	22.35	19.93	24.56
3 rings	32.33	31.6	31.75
4 rings	28.08	29.49	26.27
>4 rings	7.84	8.45	6.31
Spiro rings	2.01	1.99	1.94
Heterocycles	80.4	77.9	73.62
No N,O,S	0.48	0.98	1.0
Has N	94.06	92.37	91.74
Has O	93.25	92.3	91.51
Has S	39.25	37.56	35.95
Halogen	38.57	37.32	39.26

3.4.3 | Common functional groups

An algorithm to identify functional groups in organic molecules¹⁰³ was used for comparison of the most common functional groups in datasets. Functional groups were ranged from the most to least frequent ones and represented as the SMILES Arbitrary Target Specification (SMARTS). SMARTS is a language for specifying substructural patterns in molecules.¹⁰⁴ In particular, Figure 6 shows the most common substructures in the three datasets: the source, the novel generated, and the corrected SMILES dataset. Substructures are highlighted in red and presented as parts of novel generated SMILES strings. These examples are arranged by descending percentages within the dataset of novel generated molecules. These results indicate that initial strings of the source dataset, generated by the AE and the ones after the error-correction, have similar distributions of functional groups, again confirming that both novel and corrected datasets propagate over the same chemical space.

3.4.4 | Common molecular descriptors comparison

Properties of generated, corrected, and source data molecules were compared, using common molecular descriptors (molecular weight,

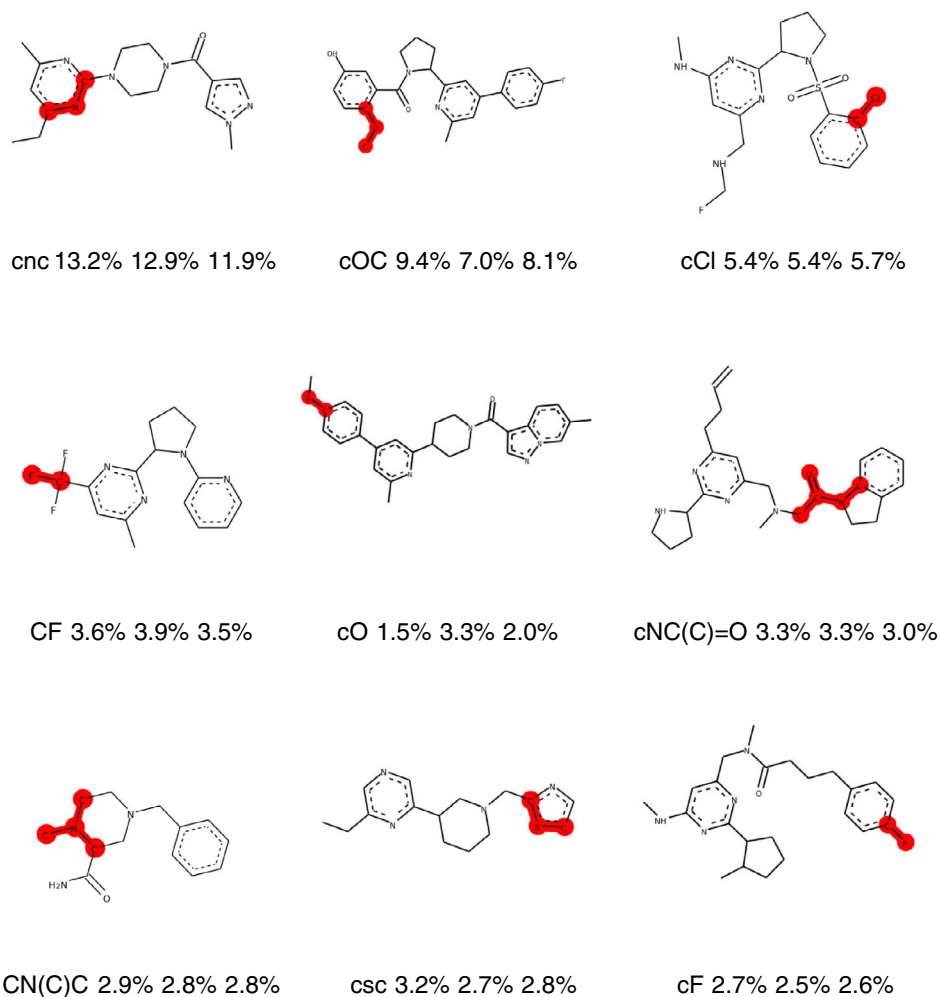


FIGURE 6 The most common substructures in datasets. The legend format: the substructure SMARTS code, percentage of occurrences in the source SMILES dataset, percentage of occurrences in the novel generated SMILES dataset, percentage of occurrences in the corrected SMILES dataset. Substructures are highlighted in red and presented as parts of novel generated SMILES strings

atom-based $\log P$,¹⁰⁵ and topological surface area). Topological polar surface area (TPSA) of a molecule is defined as the surface sum over all polar atoms including their attached hydrogen atoms. $\log P$ is the most commonly used measure of lipophilicity. This is the partition coefficient of solute between an aqueous and lipophilic phases, usually octanol and water. Results are shown in Figure 7. One can see that the three compared datasets demonstrate similar distributions.

Synthetic accessibility score (SAS) as a number between 1 (easy to make) and 10 (very difficult to make) was calculated according to ref. [106]. This method uses historical synthetic knowledge obtained by analyzing information from millions of already synthesized chemicals and considers chemical structure complexity. The score may be used to rank molecules generated by theoretical approaches for an organic synthesis. Resulting distributions are presented in Figure 7, bottom right plot. Mean values of SAS of source, novel generated, and corrected molecules are 2.5, 2.6, and 2.5 respectively, indicating that all they are relatively easy to synthesize on average.

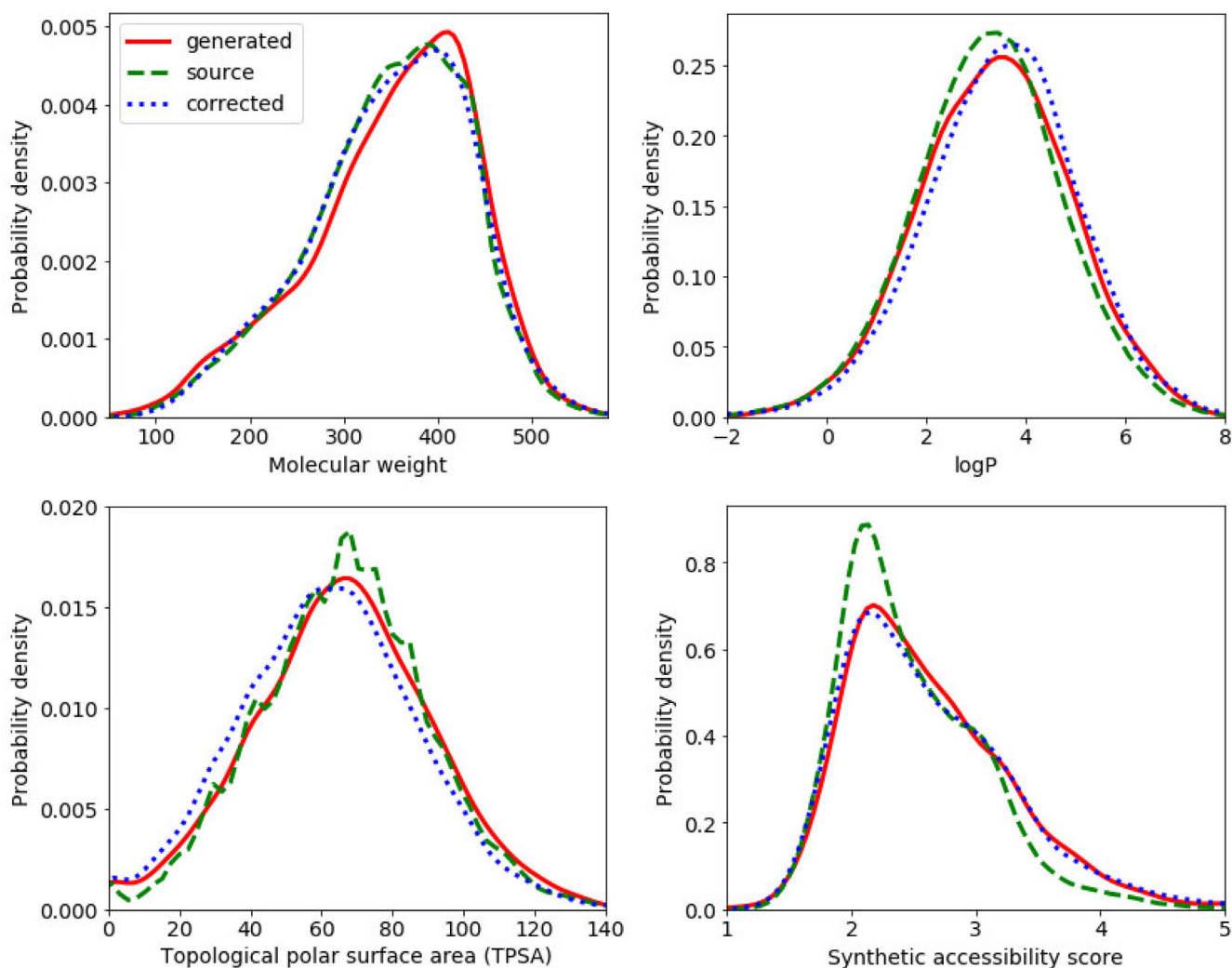


FIGURE 7 Comparison of probability density functions of molecular descriptors (molecular weight, $\log P$, TPSA) and synthetic accessibility score for the source dataset, novel generated and corrected molecules

3.5 | Solubility prediction

During the training of the predictor, we used an early stopping procedure to prevent overfitting. The quality of predictor's solubility reconstruction is visualized in Figure 8, reflecting how good the experimental values from train and test datasets are reproduced. Obviously, the perfect reconstruction is achieved if the predicted $\log S$ values fit the function $y = x$ (shown by the red lines). The solubility reconstruction within the train dataset is shown on the left plot of the Figure 8, and the corresponding coefficient of determination R^2 for $\log S$ is equal to 0.97. The reconstruction within the test dataset (right plot) shows a certain degree of scatter around the ideal case with the coefficient of determination $R^2 = 0.84$.

The difference between predicted and experimental $\log S$ values usually does not exceed unity (see the bottom plot in Figure 8), which in plain concentration units corresponds to an error of one order. Partly, this is due to a relatively small solubility dataset of only 4300 values. Nevertheless, even a rough estimate can be often helpful to categorize a compound with respect to a degree of its solubility.

Similar plots comparing experimental vs. predicted logS are also reported in refs. [45, 87]. In particular, ref. [45] provides a set of coefficients of determination for different regression approaches. Our estimation of $R^2 = 0.84$ is slightly higher than the numbers provided in ref. [45], however, we want to stress here that the direct comparison is appropriate only when the datasets are the same in both cases.

3.6 | Molecular dynamics simulation as a solubility test

We also performed a short series of all-atom computer simulations as an additional test. The solubility in computer simulations is usually measured via free energies calculations either by the thermodynamic integration or averaging over the path between thermodynamic states (see, e.g., refs. [107–109]). However, these approaches require intense simulations for numerous intermediate states and, therefore, are CPU-consuming. So, we focused on a more qualitative task aiming to check whether a particular solvent-solute composition reveals any separation trends with respect to a predicted solubility limit. Therefore, this part is rather intended to illustrate the change in the solute aggregation trend, not to find the exact solubility limit.

As a test case, we picked one of newly generated compounds, namely COCC(O)(CC)CO (shown in Figure 9). According to the predictor, its log-solubility logS is -0.26 , which corresponds to the concentration of ≈ 0.55 mol/L.

For this purpose, we “cooked” five all-atom mixtures consisting of solvent molecules H₂O and molecules of the solute C₇H₁₆O₃. The simulated mixtures differed by the solvent-solute compositions. Namely, the compositions and corresponding solute concentrations are collected in Table 3. One can see that the solute concentration gradually increases over the list of compositions. For the low-concentration compositions (below the solubility limit of 0.55 mol/L), we expect a “no-separation” regime, contrary to a “separation” regime for the cases with concentrations above the solubility limit. The compositions with moderate concentrations close to the threshold logS might show a regime transition.

Below, we discuss the instantaneous configurations of the above-defined systems. Each of the results is presented as a pair of snapshots, solely showing the simulation unit cell with only H₂O or C₇H₁₆O₃ molecules. The snapshots in the top row of Figure 10 show H₂O molecules, bottom row—C₇H₁₆O₃ molecules.

In the case of 0.11 mol/L, the solvent (top left snapshot in Figure 10) and solute (bottom most left snapshot) molecules are fully distributed over the simulation box, revealing no aggregation of the solute. For the sake of conciseness, we did not show the snapshots

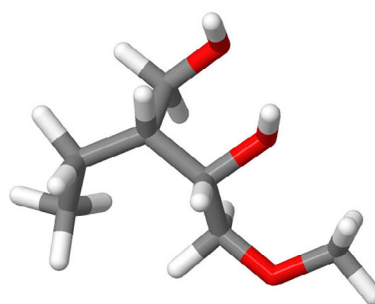


FIGURE 9 Spatial structure of a molecule COCC(O)(CC)CO represented by sticks. Carbons are shown in gray, oxygens—in red, hydrogens—in white

TABLE 3 The compositions and corresponding concentrations of the simulated systems

Number of molecules		Solute concentration
H ₂ O	C ₇ H ₁₆ O ₃	(mol/L)
20,000	10	0.03
10,000	20	0.11
10,000	100	0.5
Solubility limit		0.55
10,000	400	1.7
10,000	2000	4.2

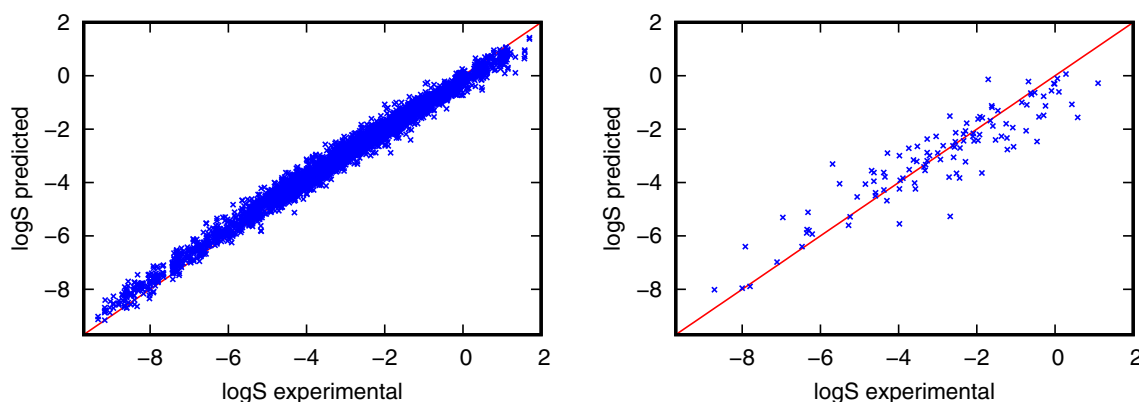


FIGURE 8 logS as reproduced on the train (left) and test (right) solubility datasets. The line $y = x$ marking the “ideal reconstruction” is plotted to guide the eye

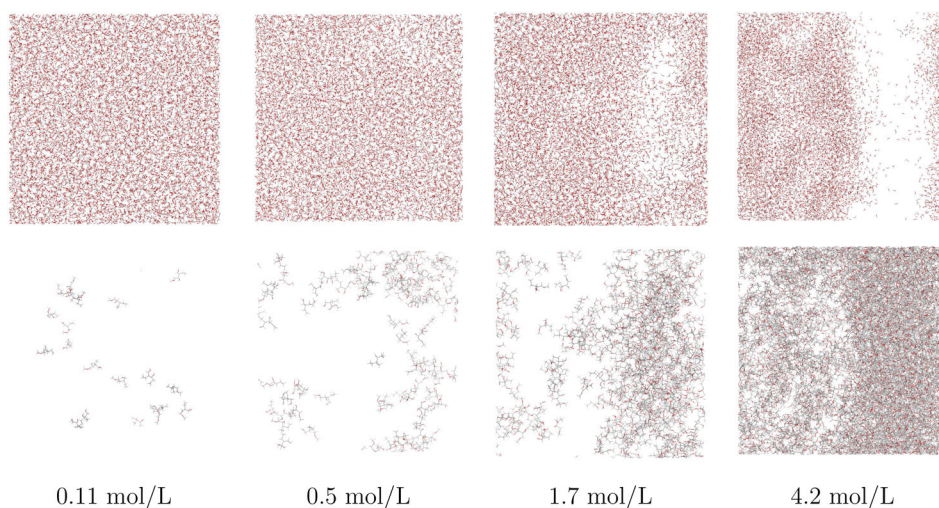


FIGURE 10 The snapshots of the simulation box, showing water H_2O (top row) and $\text{C}_7\text{H}_{16}\text{O}_3$ (bottom row) molecules as wireframes. Carbons are shown in gray, oxygens—in red, hydrogens—in white. The solute concentrations are denoted below each column

for the lowest concentration of 0.03 mol/L, because they resemble the 0.11 mol/L case.

In the second left column of Figure 10, we show the system with the solute concentration of 0.5 mol/L. This concentration is slightly below the predicted solubility limit of 0.55 mol/L. Again, top figure shows H_2O , bottom figure— $\text{C}_7\text{H}_{16}\text{O}_3$ molecules. One can notice that the solute is well distributed over the box and only begins to aggregate.

In the third column, we show the system with the solute concentration of 1.7 mol/L. This concentration is above the predicted solubility limit and reveals a separation trend, which agrees with our solubility estimation.

The case of the highest concentration of the solute (4.2 mol/L) is shown in the most right column of Figure 10. One can clearly see the areas not occupied by H_2O , as well as the areas not occupied by $\text{C}_7\text{H}_{16}\text{O}_3$. The actual composition is way above the solubility limit so the separation is well manifested.

3.7 | Drug-likeness

Let us now check the above compound COCC(O)C(CC)CO for the drug-likeness. Drug-likeness is a qualitative concept used in drug design. It can be assessed for any molecule, but it does not evaluate its pharmacological effect. In Section 1, we have mentioned a number of filters.^{51–55} All of them are based on simple metrics: the numbers of H-bond donors and acceptors, molecular weight, polar surface area, molar refractivity, solubility in water logS and octanol logP, fraction of sp^3 hybridized carbons, and number of rotatable bonds. The polar surface area, molar refractivity, and logP can be calculated via the RDKit library, for logS, we can use our solubility predictor, while the rest of metrics are directly defined from a molecular structure. We list these values in Table 4. The above-mentioned filter rules can be found in the original papers, therefore, we just briefly summarize the corresponding drug-likeness scores. So, the filters of Lipinski,⁵¹ Egan,⁵³ and Veber⁵⁵ report no violations for the compound and claim it to be drug-like. The filter by Ghose⁵² does not approve the compound for two reasons: (i) its molecular weight of 148.2 g/mol does

TABLE 4 Drug-likeness metrics for the compound COCC(O)C(CC)CO

Molecular weight	148.20 g/mol
Number of H-bond acceptors	3
Number of H-bond donors	2
Number of heteroatoms	2
Fraction of sp^3 hybridized carbons	1.00
Number of rotatable bonds	5
Number of rings	0
Topological polar surface area	49.69 Å ²
Molar refractivity	38.75
logS	−0.26
logP	0.01

not fit the range of [160:480] and (ii) molar refractivity of 38.75 is beyond the range of [40:130]. To sum up, three filters approved the compound, the Ghose filter flags two violations, but the failing values are just slightly below the qualifying limits. The properties predicted here resonate with the Bioavailability radar, proposed by Daina et al.¹¹⁰ in their SwissADME tool.

4 | CONCLUSIONS

The use of DNN-pipeline to map the discrete molecular representation (i.e., SMILES strings) into continuous vector space proved to be efficient step for predicting properties from compressed structure representation, generation of novel structures via varying the representation vector, or clustering of known ones directly in the vector space.^{60,72} Trained end-to-end such models result in the efficient structure of the chemical space where molecules with different properties belong to different parts of it.

In practice, a known set of small molecules is just a sparse fraction of exponential space of possible ones.⁷ As a result, a random vector in a corresponding continuous space may not necessarily represent a

valid chemical structure. This becomes even more severe in case of end-to-end training on limited training sets of structures along with given properties. Hence, efficient sampling in a continuous space and correction of resulting structures becomes as important as a search for an optimal mapping from discrete to continuous representation.

In this report, we focus on designing a multi-stage pipeline of several neural networks to generate novel molecular structures with desired properties and a testing pipeline to check the chemical validity of generated structures. Rather than refining a single end-to-end model, our pipeline works with a number of simple and readily available architectures that complement each other and could be trained separately on generation, prediction, and error correction tasks. Here, the generative block uses the AE network that is designed and trained to map generic discrete SMILES into a continuous vector space and reconstruct structures from the points in that space. Error correction block is built upon the attention-based sequence to sequence model that is trained independently on a set of corrupted SMILES strings. Finally, properties prediction is done via models trained on continuous representations of a reference datasets. The generation of new structures is done via an oversampling in the embedding space of a reference dataset that contains molecules with particular properties of interest and may be limited in size.

We test our pipeline in a number of experiments and show that it leads to good generation and prediction qualities for novel structures. We show that the error correction block allows increasing by 67% the number of valid and unique molecular structures in the output of basic AE model. The prediction model trained directly on the embeddings of the reference dataset results in the state of the art estimation of aqueous solubility.

In our analysis of structural similarity (i.e., distributions of scaffolds, substructure features, functional groups, and molecular descriptors) and synthetic accessibility of reference and generated structures, we showcase that unique structures span the same part of the chemical space. Finally, as a sanity check, we performed a molecular dynamics simulation to confirm our predictions for the aqueous solubility of generated molecules.

To summarize, our pipeline allows generating new valid chemical structures in seconds, predict their properties without running laboratory tests, and examine these compounds with various drug-likeness filters.

ORCID

Maksym Druchok  <https://orcid.org/0000-0003-3094-6414>

REFERENCES

- [1] M. Dickson, J. P. Gagnon, *Nat. Rev. Drug Discov.* **2004**, *3*, 417. <https://doi.org/10.1038/nrd1382>.
- [2] A. Jahan, M. Y. Ismail, S. M. Sapuan, F. Mustapha, *Mater. Des.* **2010**, *31*, 696. <https://doi.org/10.1016/j.matdes.2009.08.013>.
- [3] A. Schuhmacher, O. Gassmann, M. Hinder, *J. Transl. Med.* **2016**, *14*, 105. <https://doi.org/10.1186/s12967-016-0838-4>.
- [4] J. C. Babiarz, in *FDA Regulatory affairs. A guide for prescription drugs, medical devices, and biologics*, 2nd ed. (Eds: D. J. Pisano, D. S. Mantus), Informa Healthcare, New York **2008**; Chapter 1, p. 34.
- [5] E. Petrova, *Innovation and Marketing in the Pharmaceutical Industry*; (Eds: M. Ding, et al), International Series in Quantitative Marketing 20, Springer-Verlag, New York **2014**. DOI: <https://doi.org/10.1007/978-1-4614-7801-0>
- [6] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, S. H. Bryant, *Nucleic Acids Res.* **2016**, *44*(D1), D1202. <https://doi.org/10.1093/nar/gkv951>.
- [7] P. Kirkpatrick, C. Ellis, *Nature* **2004**, *432*, 823. <https://doi.org/10.1038/432823a>.
- [8] N. Bloom, C. I. Jones, J. Van Reenen, M. Webb, *Am Econ Rev* **2020**, *110*(4), 1104. <https://doi.org/10.3386/w23782>.
- [9] Y. LeCun, Y. Bengio, G. Hinton, *Nature* **2015**, *521*, 436. <https://doi.org/10.1038/nature14539>.
- [10] G. B. Goh, N. O. Hodas, A. Vishnu, *J. Comput. Chem.* **2017**, *38*, 1291. <https://doi.org/10.1002/jcc.24764>.
- [11] R. Miotto, F. Wang, S. Wang, X. Jiang, J. T. Dudley, *Brief. Bioinform.* **2018**, *19*(6), 1236. <https://doi.org/10.1093/bib/bbx044>.
- [12] G. Schneider, *Nat. Rev. Drug Discov.* **2018**, *17*, 97. <https://doi.org/10.1038/nrd.2017.232>.
- [13] J. Boström, D. G. Brown, R. J. Young, G. M. Keserü, *Nat. Rev. Drug Discov.* **2018**, *17*, 709. <https://doi.org/10.1038/nrd.2018.116>.
- [14] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, A. Walsh, *Nature* **2018**, *559*, 547. <https://doi.org/10.1038/s41586-018-0337-2>.
- [15] A. Zhavoronkov, Y. A. Ivanenkov, A. Aliper, M. S. Veselov, V. A. Aladinskiy, A. V. Aladinskaya, V. A. Terentiev, D. A. Polykovskiy, M. D. Kuznetsov, A. Asadulaev, Y. Volkov, A. Zhulus, R. R. Shayakhmetov, A. Zhebrak, L. I. Minaeva, B. A. Zagribelnyy, L. H. Lee, R. Soll, D. Madge, L. Xing, T. Guo, A. Aspuru-Guzik, *Nat. Biotechnol.* **2019**, *37*, 1038. <https://doi.org/10.1038/s41587-019-0224-x>.
- [16] M. O. Steinhauser, S. Hiermaier, *Int. J. Mol. Sci.* **2009**, *10*(12), 5135. <https://doi.org/10.3390/ijms10125135>.
- [17] J. Behler, in *Chemical Modelling: Applications and Theory*, Vol. 7 (Ed: M. Springborg), The Royal Society of Chemistry, New York **2010**, p. 1. <https://doi.org/10.1039/9781849730884-00001>.
- [18] S. A. Ghasemi, A. Hofstetter, S. Saha, S. Goedecker, *Phys. Rev. B* **2015**, *045131*, 92. <https://doi.org/10.1103/PhysRevB.92.045131>.
- [19] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, A. Tkatchenko, *Nat. Comm.* **2017**, *8*, 13890. <https://doi.org/10.1038/ncomms13890>.
- [20] J. Carrasquilla, R. G. Melko, *Nat. Phys.* **2017**, *13*, 431. <https://doi.org/10.1038/nphys4035>.
- [21] T. Xie, J. C. Grossman, *Phys. Rev. Lett.* **2018**, *145301*, 120. <https://doi.org/10.1103/PhysRevLett.120.145301>.
- [22] K. Ryan, J. Lengyel, M. Shatruk, *J. Am. Chem. Soc.* **2018**, *140*(32), 10158. <https://doi.org/10.1021/jacs.8b03913>.
- [23] S. Amabilino, L. A. Bratholm, S. J. Bennie, A. C. Vaucher, M. Reiher, D. R. Glowacki, *J. Phys. Chem. A* **2019**, *123*(20), 4486. <https://doi.org/10.1021/acs.jpca.9b01006>.
- [24] F. E. Bock, R. C. Aydin, C. J. Cyron, N. Huber, S. R. Kalidindi, B. Klusemann, *Front. Mater.* **2019**, *6*, 110. <https://doi.org/10.3389/fmats.2019.00110>.
- [25] M. Haghghatdari, J. Hachmann, *Curr. Opin. Chem. Eng.* **2019**, *23*, 51. <https://doi.org/10.1016/j.coche.2019.02.009>.
- [26] S. Chiriki, S. S. Bulusu, *Chem. Phys. Lett.* **2016**, *652*, 130. <https://doi.org/10.1016/j.cplett.2016.04.013>.
- [27] L. Shen, W. Yang, *J. Chem. Theory Comput.* **2018**, *14*, 1442. <https://doi.org/10.1021/acs.jctc.7b01195>.
- [28] S. Jindal, S. S. Bulusu, *J. Chem. Phys.* **2018**, *194101*, 149. <https://doi.org/10.1063/1.5043247>.
- [29] R. Kondor, A Transferable Artificial Neural Network Model for Atomic Forces in Nanoparticles. *arXiv:1810.06204* **2018**.
- [30] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, *J. Chem. Phys.* **2018**, *148*, 241722. <https://doi.org/10.1063/1.5019779>.
- [31] A. Pérez, G. Martínez-Rosell, G. De Fabritiis, *Curr. Opin. Struct. Biol.* **2018**, *49*, 139. <https://doi.org/10.1016/j.sbi.2018.02.004>.
- [32] J. Herr, K. Yao, K. McIntyre, D. W. Toth, J. Parkhill, *J. Chem. Phys.* **2018**, *148*, 241710. <https://doi.org/10.1063/1.5020067>.
- [33] K. Yao, J. E. Herr, D. W. Toth, R. MckIntyre, J. Parkhill, *Chem. Sci.* **2018**, *9*, 2261. <https://doi.org/10.1039/C7SC04934J>.

- [34] H. Wang, L. Zhang, J. Han, E. Weinan, *Comput. Phys. Commun.* **2018**, 228, 178. <https://doi.org/10.1016/j.cpc.2018.03.016>.
- [35] L. Zhang, H. Wang, J. Han, R. Car, E. Weinan, *Phys. Rev. Lett.* **2018**, 120, 143001. <https://doi.org/10.1103/PhysRevLett.120.143001>.
- [36] L. Zhang, H. Wang, E. Weinan, *J. Chem. Phys.* **2018**, 149, 154107. <https://doi.org/10.1063/1.5042714>.
- [37] L. Zhang, J. Han, H. Wang, R. Car, E. Weinan, *J. Chem. Phys.* **2018**, 149, 034101. <https://doi.org/10.1063/1.5027645>.
- [38] A. Lusci, G. Pollastri, P. Baldi, *J. Chem. Inf. Model.* **2013**, 53, 1563. <https://doi.org/10.1021/ci400187y>.
- [39] G.E. Dahl, N. Jaitly, R. Salakhutdinov. Multi-task Neural Networks for QSAR Predictions. *arXiv:1406.1231* **2014**.
- [40] E. O. Pyzer-Knapp, K. Li, A. Aspuru-Guzik, *Adv. Funct. Mater.* **2015**, 25, 6495. <https://doi.org/10.1002/adfm.201501919>.
- [41] B. Alipanahi, A. Delong, M. T. Weirauch, B. J. Frey, *Nature Biotechnol.* **2015**, 33, 831. <https://doi.org/10.1038/nbt.3300>.
- [42] I. Wallach, M. Dzamba, A. Heifets. AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-Based Drug Discovery. *arXiv:1510.02855* **2015**.
- [43] A. Mayr, G. Klambauer, T. Unterthiner, S. Hochreiter, *Front. Environ. Sci.* **2016**, 3, 80. <https://doi.org/10.3389/fenvs.2015.00080>.
- [44] E.J. Bjerrum. SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules. *arXiv:1703.07076* **2017**.
- [45] A. K. Sharma, G. N. Srivastava, A. Roy, V. K. Sharma, *Front. Pharmacol.* **2017**, 8, 880. <https://doi.org/10.3389/fphar.2017.00880>.
- [46] S. Kearnes, B. Goldman, V. Pande. Modeling Industrial ADMET Data with Multitask Networks. *arXiv:1606.08793* **2017**.
- [47] J. Jiménez, M. Škalič, G. Martínez-Rosell, G. De Fabritiis, *J. Chem. Inf. Model.* **2018**, 58, 287. <https://doi.org/10.1021/acs.jcim.7b00650>.
- [48] G.B. Goh, N.O. Hodas, C. Siegel, A. Vishnu. SMILES2Vec: An Interpretable General-Purpose Deep Neural Network for Predicting Chemical Properties. *arXiv:1712.02034* **2018**.
- [49] G.B. Goh, C. Siegel, A. Vishnu, N.O. Hodas. Using Rule-based Labels for Weak Supervised Learning: A Chemnet for Transferable Chemical Property Prediction. *arXiv:1712.02734* **2018**.
- [50] N. Ståhl, G. Falkman, A. Karlsson, G. Mathiason, J. Boström, *J. Integr. Bioinform.* **2018**, 20180065, 1613. <https://doi.org/10.1515/jib-2018-0065>.
- [51] C. A. Lipinski, F. Lombardo, B. W. Dominy, P. J. Feeney, *Adv. Drug Deliv. Rev.* **2018**, 46, 3. [https://doi.org/10.1016/S0169-409X\(96\)00423-1](https://doi.org/10.1016/S0169-409X(96)00423-1).
- [52] A. K. Ghose, V. N. Viswanadhan, J. J. Wendoloski, *J. Comb. Chem.* **1999**, 1, 55. <https://doi.org/10.1021/cc9800071>.
- [53] W. J. Egan, K. M. Merz, J. J. Baldwin, *J. Med. Chem.* **2000**, 43, 3867. <https://doi.org/10.1021/jm000292e>.
- [54] I. Muegge, S. L. Heald, D. Brittelli, *J. Med. Chem.* **2001**, 44, 1841. <https://doi.org/10.1021/jm015507e>.
- [55] D. F. Veber, S. R. Johnson, H. Y. Cheng, B. R. Smith, K. W. Ward, K. D. Kopple, *J. Med. Chem.* **2002**, 45, 2615. <https://doi.org/10.1021/jm020017n>.
- [56] M. H. Segler, T. Kogej, C. Tyrchan, M. P. Waller, *ACS Cent. Sci.* **2018**, 4(1), 120. <https://doi.org/10.1021/acscentsci.7b00512>.
- [57] M.J. Kusner, B. Paige, J.M. Hernández-Lobato. Grammar Variational Autoencoder. *arXiv:1703.01925v1* **2017**.
- [58] G.B. Goh, C. Siegel, A. Vishnu, N.O. Hodas, Baker, N. How Much Chemistry Does a Deep Neural Network Need to Know To Make Accurate Predictions? *arXiv:1710.02238* **2018**.
- [59] G.B. Goh, K. Sakloth, C. Siegel, A. Vishnu, J. Pfaendtner. Multimodal Deep Neural Networks Using Both Engineered and Learned Representations for Biodegradability Prediction. *arXiv:1808.04456* **2018**.
- [60] D. Kuzminykh, D. Polykovskiy, A. Kadurin, A. Zhebrak, I. Baskov, S. Nikolenko, R. Shayakhmetov, A. Zhavoronkov, *Mol. Pharmaceutics* **2018**, 15, 4378. <https://doi.org/10.1021/acs.molpharmaceut.7b01134>.
- [61] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu. A Comprehensive Survey on Graph Neural Networks. *arXiv:1901.00596v2* **2019**.
- [62] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434v3* **2019**.
- [63] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, P. Riley, *J. Comput. Aided Mol. Des.* **2016**, 30(8), 595. <https://doi.org/10.1007/s10822-016-9938-8>.
- [64] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams. *Adv. Neural Inform. Process. Syst.* 28 (NIPS 2015), Montreal, Canada, Dec 7–12, 2015; (Eds: C. Cortes et al), Curran Associates, Inc., Red Hook, NY **2016**, 2224.
- [65] J. You, B. Liu, R. Ying, V. Pande, J. Leskovec. Graph Convolutional Policy Network for Goal-directed Molecular Graph Generation. *arXiv:1806.02473v3* **2019**.
- [66] A. Fout, J. Byrd, B. Shariat, A. Ben-Hur. *Adv. Neural Inform. Process. Syst.* 30 (NIPS 2017), Long Beach, CA, USA, Dec 4–9, 2017; (Eds: I. Guyon et al), Curran Associates, Inc., Red Hook, NY **2018**, 6530.
- [67] M. Zitnik, M. Agrawal, J. Leskovec. Modeling Polypharmacy Side Effects With Graph Convolutional Networks. *arXiv:1802.00543v2* **2018**.
- [68] N. De Cao, T. Kipf. MolGAN: An Implicit Generative Model for Small Molecular Graphs. *arXiv:1805.11973v1* **2018**.
- [69] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. *Adv. Neural Inform. Process. Syst.* 27 (NIPS 2014), Montreal, Canada, Dec 8–13, 2014; (Eds: Z. Ghahramani et al.); Curran Associates, Inc., Red Hook, NY, **2015**, 2672.
- [70] A. Creswell, T. White, V. Dumoulin, K. Arulkumar, B. Sengupta, A. A. Bharath, *IEEE Signal Process. Mag.* **2018**, 35(1), 53. <https://doi.org/10.1109/MSP.2017.2765202>.
- [71] P. B. Jørgensen, M. N. Schmidt, O. Winther, *Mol. Inf.* **2018**, 1700133, 37. <https://doi.org/10.1002/minf.201700133>.
- [72] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, *ACS Cent. Sci.* **2018**, 4(2), 268. <https://doi.org/10.1021/acscentsci.7b00572>.
- [73] A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, A. Zhavoronkov, *Mol. Pharmaceutics* **2017**, 14, 3098. <https://doi.org/10.1021/acs.molpharmaceut.7b00346>.
- [74] E. Putin, A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A. V. Aladinskaya, A. Aliper, A. Zhavoronkov, *Mol. Pharmaceutics* **2018**, 15, 4386. <https://doi.org/10.1021/acs.molpharmaceut.7b01137>.
- [75] T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath, H. Chen, *Mol. Inf.* **2018**, 1700123, 37. <https://doi.org/10.1002/minf.201700123>.
- [76] E. J. Bjerrum, B. Sattarov, *Biomolecules* **2018**, 8, 131. <https://doi.org/10.3390/biom8040131>.
- [77] G. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P.L.C. Farias, A. Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv:1705.10843v3* **2018**.
- [78] H. Dai, Y. Tian, B. Dai, S. Skiena, L. Song. Syntax-Directed Variational Autoencoder for Structured Data. *arXiv:1802.08786v1* **2018**.
- [79] G.E. Hinton, R.S. Zemel. Proc. 6th Int. Conf. Neural Inform. Process. Syst. (NIPS 1993), Denver, CO, USA, Nov 30–Dec 2, 1993; (Eds: J.D. Cowan, et al), Morgan Kaufmann Publishers Inc, San Francisco, CA, **1994**, 3.
- [80] D.P. Kingma, M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114v10* **2014**.
- [81] <https://www.emolecules.com/info/plus/download-database>
- [82] J. Huuskonen, *J. Chem. Inf. Comput. Sci.* **2000**, 40, 773. <https://doi.org/10.1021/ci9901338>.
- [83] T. Hou, K. Xia, W. Zhang, X. Xu, *J. Chem. Inf. Comput. Sci.* **2004**, 44, 266. <https://doi.org/10.1021/ci034184n>.
- [84] J. S. Delaney, *J. Chem. Inf. Comput. Sci.* **2004**, 44, 1000. <https://doi.org/10.1021/ci034243x>.
- [85] DLS-100 Solubility Dataset. DOI: <https://doi.org/10.17630/3a3a5abc-8458-4924-8e6c-b804347605e8>

- [86] A. Llinàs, R. C. Glen, J. M. Goodman, *J. Chem. Inf. Model.* **2008**, 48, 1289. <https://doi.org/10.1021/ci800058v>.
- [87] A. J. Hopfinger, E. X. Esposito, A. Llinàs, R. C. Glen, J. M. Goodman, *J. Chem. Inf. Model.* **2009**, 49, 1. <https://doi.org/10.1021/ci800436c>.
- [88] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, *J. Artif. Int. Res.* **2002**, 16, 321. <https://doi.org/10.1613/jair.953>.
- [89] G. Lemaître, F. Nogueira, C. K. Aridas, *J. Mach. Learn. Res.* **2017**, 18, 1.
- [90] RDKit: Open-source cheminformatics; <http://www.rdkit.org>
- [91] I. Sutskever, O. Vinyals, Q.V. Le. *Adv. Neural Inform. Process. Syst.* 27 (NIPS 2014), Montreal, Canada, Dec 8–13, 2014; (Eds: Z. Ghahramani, et al.), Curran Associates, Inc., Red Hook, NY, **2015**, 3104.
- [92] D. Bahdanau, K. Cho, Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv:1409.0473 **2014**.
- [93] S. Hochreiter, J. Schmidhuber, *Neural Comput.* **1997**, 9(8), 1735. <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [94] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean. *Adv. Neural Inform. Process. Syst.* 26 (NIPS 2013), Lake Tahoe, NV, USA, Dec 5–10, 2013; (Eds: C.J.C. Burges, et al.), Curran Associates, Inc, Red Hook, NY **2014**, 3111.
- [95] A. Lamb, A. Goyal, S. Zhang, A.C. Courville, Y. Bengio. *Adv. Neural Inform. Process. Syst.* 29 (NIPS 2016), Barcelona, Spain, Dec 5–10, 2016; (Eds: D.D. Lee, et al), Curran Associates, Inc, Red Hook, NY **2017**, 4601.
- [96] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol. *Proc. 25th Int. Conf. Machine Learning*, Helsinki, Finland, July 5–9, 2008; (Eds: A. McCallum, S. Roweis). Omnipress: Madison, WI, USA, **2008**, 1096. DOI: <https://doi.org/10.1145/1390156.1390294>
- [97] H. J. C. Berendsen, J. R. Grigera, T. P. Straatsma, *J. Phys. Chem.* **1987**, 91, 6269. <https://doi.org/10.1021/j100308a038>.
- [98] W. L. Jorgensen, D. S. Maxwell, J. Tirado-Rives, *J. Am. Chem. Soc.* **1996**, 118, 11225. <https://doi.org/10.1021/ja9621760>.
- [99] W. Smith, T. Forester, I. Todorov. "The DL POLY Classic User Manual", STFC Daresbury Laboratory, http://www.ccp5.ac.uk/DL_POLY_C/, **2010**
- [100] S. Melchionna, G. Ciccotti, B. L. Holian, *Molec. Phys.* **1993**, 78, 533. <https://doi.org/10.1080/00268979300100371>.
- [101] <https://www.ebi.ac.uk/chembl/>
- [102] <https://pubchem.ncbi.nlm.nih.gov/>
- [103] P. Ertl, *J. Cheminformatics* **2017**, 9, 36. <https://doi.org/10.1186/s13321-017-0225-z>.
- [104] SMARTS Theory Manual, Daylight Chemical Information Systems. <https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>
- [105] A. Wildman, G. M. Crippen, *J. Chem. Inf. Comput. Sci.* **1999**, 395, 868. <https://doi.org/10.1021/ci990307l>.
- [106] P. Ertl, A. Schuffenhauer, *J. Cheminformatics* **2009**, 1, 8. <https://doi.org/10.1186/1758-2946-1-8>.
- [107] L. Li, T. Totton, D. Frenkel, *J. Chem. Phys.* **2017**, 214110, 146. <https://doi.org/10.1063/1.4983754>.
- [108] R. Gillet, A. Fierro, L. M. Valenzuela, J. R. Pérez-Correa, *Fluid Phase Equilib.* **2018**, 472, 85. <https://doi.org/10.1016/j.fluid.2018.05.013>.
- [109] G. Duarte Ramos Matos, D. L. Mobley, *F1000Research* **2019**, 7, 686. <https://doi.org/10.12688/f1000research.14960.2>.
- [110] A. Daina, O. Michielin, V. Zoete, *Sci. Rep.* **2017**, 7, 42717. <https://doi.org/10.1038/srep42717>.

How to cite this article: Druchok M, Yarish D, Gurbych O, Maksymenko M. Toward efficient generation, correction, and properties control of unique drug-like structures. *J Comput Chem.* 2021;42:746–760. <https://doi.org/10.1002/jcc.26494>

APPENDIX: ARCHITECTURES OF THE AUTOENCODER AND SOLUBILITY PREDICTOR

The autoencoder consists of two sub-networks—encoder and decoder (Figure A1). Both the encoder and decoder use convolution and fully-connected type layers. The dimensions of the layers are chosen to maintain a constant number of neurons per layer that allows applying a residual technique when a layer output is mixed with outputs of subsequent layers by skip connections and helps to improve the network convergence. The solid arrows denote the regular weighted connections, the dotted ones show the skip connections. The outputs after each layer are passed through the ReLU activation functions. The output of the Encoder is a vector that can be considered as a SMILES latent representation. This vector is further fed into the decoder, which then outputs the SMILES string.

The latent vectors are also fed into the structure-to-property predictor. In our case we trained the predictor for an aqueous solubility.

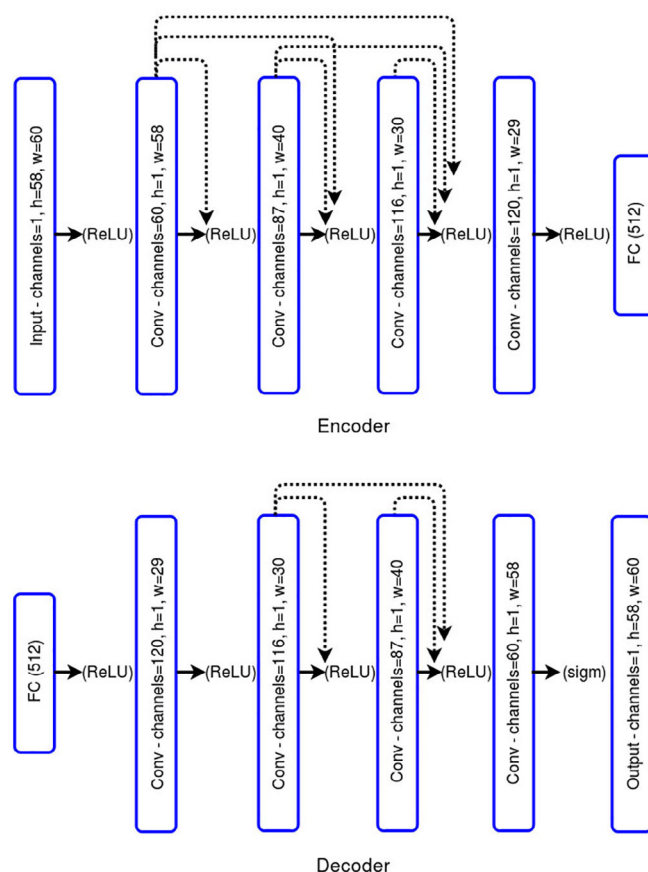


FIGURE A1 The architecture of the autoencoder. The upper chart presents the encoder: differentiable connections between layers are denoted by black solid arrows, while the dotted ones denote nonweighted addition with the appropriate reshape, followed by the activation function of the ReLU or sigmoid type. Each convolution layer is characterized by height (h), width (w), and number of channels. The bottom chart shows the decoder scheme with the same notation and color conventions

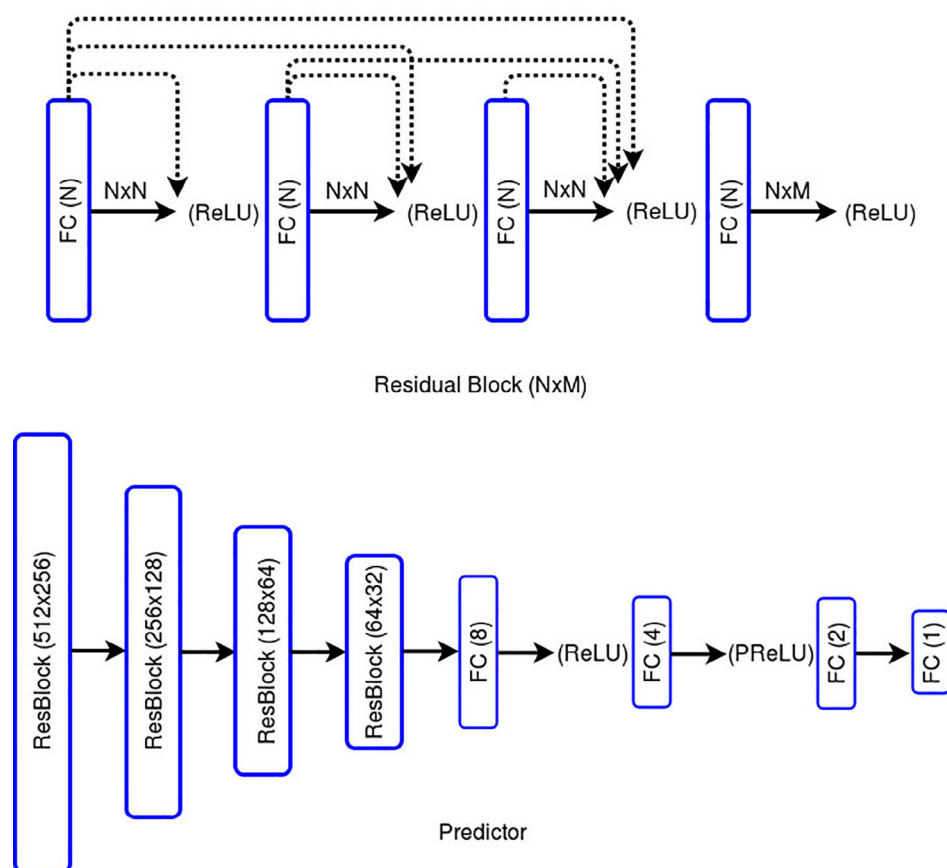


FIGURE A2 The architecture of the predictor. The upper chart shows a residual block: differentiable connections between FC layers are denoted by solid arrows, the dotted arrows denote nonweighted addition, then followed by activation function of the ReLU type. The bottom chart presents the general scheme of the predictor: four residual blocks are followed by four fully connected layers

The predictor's architecture consists of four residual blocks and four FC layers. As seen in Figure A2, the residual blocks unite four FC layers each. The regular weighted connections are denoted by solid arrows.

In contrast to the plain FC network, the intermediate vectors are added to subsequent FC outputs (denoted by dotted arrows). The predictor outputs a single value, being treated as a solubility prediction.