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

Maksym Druchok* [1,2], **Dzvenymyra Yarish** [1,3], **Oleksandr Gurbych** [1] **and Mykola Maksymenko** [1]

[1] *SoftServe, 2d Sadova Str., 79021 Lviv, Ukraine*
[2] *Institute for Condensed Matter Physics, 1 Svientsitskii Str., 79011 Lviv, Ukraine,*
[3] *Ukrainian Catholic University, 17 Svientsitskii Str., 79011 Lviv, Ukraine*

Correspondence*:
Maksym Druchok
mdruc@softserveinc.com

ABSTRACT

Efficient design and screening of the novel molecules is a major challenge in drug and material design. This report focuses on a multi-stage pipeline in which several deep neural network (DNN) 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 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 pipeline allows generating novel structures with control of SAS (Synthetic Accessibility Score) and a series of ADME metrics that assess the drug-likeliness.

Keywords: molecular design, machine learning, deep neural networks, autoencoder, ADME

1 INTRODUCTION

Designing new materials is often a challenging, resourceful, and time-consuming endeavor (1; 2; 3). In particular, drug development is a stepwise, iterative process involving certain necessary stages: discovery and research, development, regulatory approval, only then followed by production and marketing. Typically, it takes about 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 discovered just a tiny piece of an exponential set of potential compounds: it is estimated that only about 10^8 of small drug-like molecules (6) 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 sufficiently widen the areas of potential impact, including applications to scientific problems (10; 11).

In this respect, chemical and pharmaceutical industries reveal a substantial interest in machine learning approaches aiding on the increase in efficiency of novel materials design (12; 13; 14). As a new milestone reached, it is worth mentioning a very recent study by Zhavoronkov *et al.* (15) reporting an AI-driven development of inhibitor candidates in just **21** days, which unprecedentedly shortens the pre-clinical phase.

1.1 Machine Learning 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 (17). Here, recent advances in ML allow to efficiently tackle multiple problems from the approximation of complex functions or quantum wave-functions to the prediction of properties, phase transitions and temporal dynamics (18; 19; 20; 21; 22; 23; 24; 25).

At atomic level DNNs are used to approximate quantum mechanical computations such as atomic/molecular parameters in all-particle microscopic simulations (18; 26), decomposing cluster energies or predict next simulation step instead of CPU-costly traditional routines, hence, speeding up the sampling (27; 28; 29; 30; 31; 32). Recently, a few simulation packages already incorporated these approaches into their computational tools (33; 34; 35; 36; 37; 38).

On a physical chemistry level DNNs help to 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, like solubility in water or organic solvents, melting point, solvation energies, etc. (39; 40; 41). The latter ability to use machine learning to link structure to a property is interesting from biophysics and physiology perspectives to address drug-likeliness of a particular structure. In this case the focus is on a set of ADME (Absorption, Distribution, Metabolism, Excretion) or ADMET (if Toxicity is also considered) properties, predicting binding affinity, inhibition activity, biodegradability, and toxicity of compounds (42; 43; 44; 45; 46; 47; 48; 49; 50; 51). A kind of "golden standard" in ADME screening is a so-called *the rule of five*, suggested by a study of Lipinski *et al.* (52), or its close variations (53; 54; 55; 56), allowing pharma-industry considerably shrink the screening pool of drug-candidates at the pre-clinical stage.

In ML methods dealing with molecular structures, the crucial step is the efficient representation of the structural data. The above approaches use different variations of input data: graphs, fingerprints, descriptors, mixed fingerprint-descriptor models (see, for example, Ref. (46)), one-line notations (like 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 NLP methods to chemistry-related problems, including the generation of new compounds (57; 58). In Refs. (59; 60) SMILES strings are converted into 2D images and then fed into DNNs. A similar approach to use molecular 2D images as inputs is also exploited in Ref. (61).

Methods based on graph neural networks (see, for example, recent reviews (62; 63) 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 (64). In particular, graph convolution networks are under consideration in Refs. (64; 65), where authors use molecular graphs to predict solubility, toxicity and other properties of compounds. Ref. (66) unites graph representations with adversarial training and reinforcement learning to guide the process of molecular design towards the outputs with desired properties. In Ref. (67) authors utilize graph neural networks for protein interface predictions, the Ref. (68) suggests a combined approach of graph and relation networks for breast cancer classification problem.

The study by Zitnik *et al.* (69) 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. (70).

Recent advances of Deep Learning to 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 (71; 72). In the context of molecular data, this opens a route for the synthesis of novel structures with given properties (see, for example, a review by Jørgensen *et al* (73)). In (61; 74) 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 in order to improve the quality of continuous representations, reduce reconstruction errors and assess the performance of different models (73; 75; 76). A study introducing generative adversarial autoencoder is reported in Ref. (77) – 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 (78). A GAN-based model by Guimaraes *et al.* (79) adversarially learns to output SMILES strings optimizing them towards chemical metrics.

Note, that despite the continuous nature of the autoencoder latent space and infinite possibilities of choosing arbitrary latent vectors, not all the 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. Hence, an additional post-processing and analysis over the decoded strings are needed (74).

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 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 pipeline covers substantial stages of early stage drug discovery. From proposing new structure to ADMET filtering and confirming properties in numerical simulation (See Fig. 1).

First, we use a common Autoencoder architecture allowing to map discrete SMILES strings into continuous latent space (80; 81). We show that the size of the dataset plays a crucial role in achieving higher generalization and reconstruction quality. Since 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 Attention-based Sequence-to-Sequence model to correct errors in generated notations.

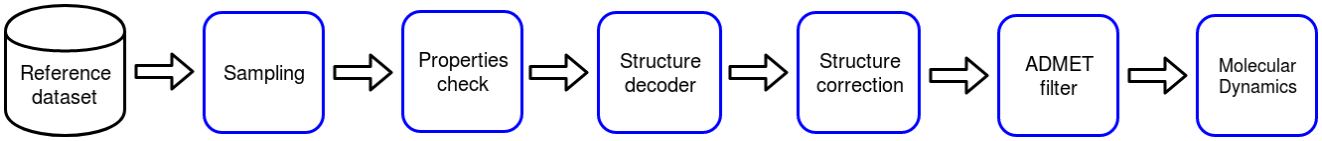


Figure 1. The proposed system covers multiple steps of early-stage drug discovery. Datapoints are generated directly in continuous space with regards to the reference dataset with immediate assessment of properties. Once the SMILE is decoded and corrected it passes ADMET filter and finally basis property is confirmed in molecular dynamics simulation.

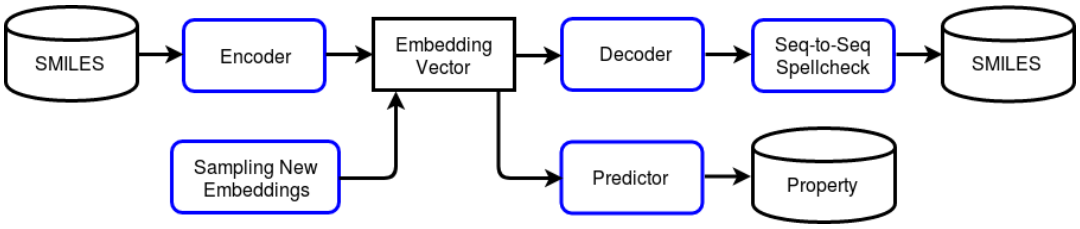


Figure 2. The system architecture consists of the generative part (encoder and decoder), error correction model (Seq-to-Seq), set of predictors or properties and the sampler of new datapoints in the embedding space.

To assess the quality of resulting structures we introduce 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 assessment by performing a computer simulation on all-atom level. This simulation modelled a set of aqueous solutions of one of newly generated compound 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 solubility prediction. The general pipeline is shown in Fig. 1. Below we will discuss the pipeline in more detail.

1.3 Plan of the report

The remaining part of the report is organized as follows: in Section 2 we discuss the model architecture, datasets used, and the operating pipeline. Section 2.5 presents the techniques engaged for the error-correction in decoded structures. Then, in Section 3 we gathered the main results of the study. Finally, the remarks and discussion conclude the report in Section 4.

2 MATERIALS AND METHODS

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

The predictors 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.

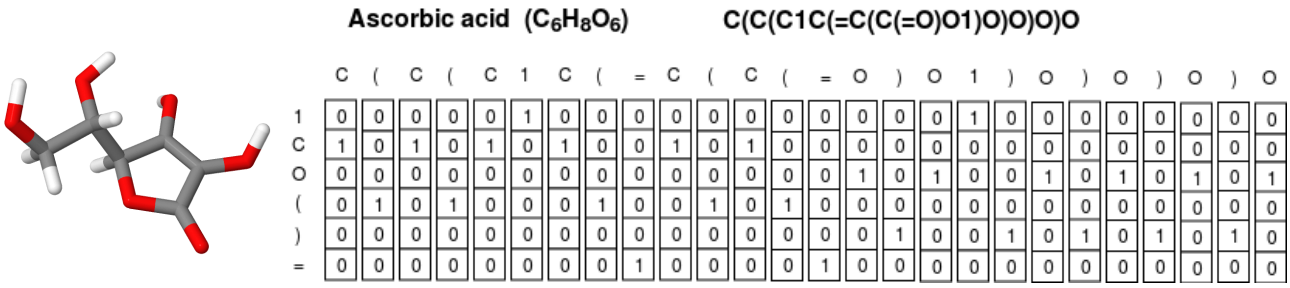


Figure 3. Ascorbic acid as molecular formula, SMILES string, one-hot matrix, and a spatial structure of a molecule represented by sticks. Carbons in the 3D representation are shown in gray, oxygens – in red, hydrogens – in white.

2.1 Molecular representations

There exist various chemical representations helping encode spatial molecular structure by compact one-line notations; most popular among them are SMILES and InChi. Ref. (74) demonstrated that InChi representations had shown a worse performance, than SMILES-based ones. The SMILES form carries all the necessary information to calculate most of the ADME metrics (H-bond donors, acceptors, molecular weight), except lipophilicity logP and aqueous solubililty logS. A SMILES string itself can not 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 a length of a SMILES string. The example of ascorbic acid is illustrated in Fig. 3, where a spatial representation for this molecule 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 eMolecules database (82) – a set of commercially distributed chemicals – as a large list of valid SMILES strings to train Autoencoder.

For the property prediction we focus on solubility and compiled our dataset from a series of open sets published by Huuskonen (83), Hou *et al.* (84), Delaney (85), and Mitchell (86). 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 of not longer than 60 elements. All solubilities are presented in the form of the so-called log solubility – logS, which is the 10-based logarithm of the solubility measured in mol/L.

It is important to keep in mind, that the ADMET prediction is a difficult task, since datasets usually combine the data generated 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. (87; 88).

2.3 Model design

We decided to utilize a fully deterministic Autoencoder (see APPENDIX and Fig. 11 therein for the detailed architectures). The Encoder, first part of the Autoencoder, consists of four convolution layers, followed by a fully connected (FC) layer. The output of the Encoder is considered as a latent SMILES representation. The Decoder, second part of the Autoencoder, is aimed to decode latent representations back to original SMILES strings. It, symmetrically, consists of a 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 Fig. 12 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 Autoencoder and Predictor, adjusting only the learning rate within the range of $10^{-3} \div 10^{-5}$. In the case of the Autoencoder 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 Autoencoder 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. It is important to note, that the above scheme allowed us to build other predictors on top of the existing latent space and predict other properties along with the 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 to operate 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 using "smarter" linear combinations of vectors belonging to existing (correct) structures.

Indeed, the random selection of vectors yields almost 100% faulty results. Therefore, we used the SMOTE (Synthetic Minority Oversampling Technique) approach (89) adopting its implementation from the Imblearn library (90). In particular, new samples are generated with the algorithm, which selects similar instances (using a distance measure) and perturbs an instance one attribute at a time by a random delta within the difference to the neighboring instances.

For the generation purposes we randomly selected a subset of ≈190.000 SMILES entities from the eMolecules set (82), converted them into corresponding latent vectors and ran the SMOTE algorithm to produce extra ≈95.000 latent vectors. Such a strategy had shown a good outcome and below we discuss these results in detail.

After the latent vectors for new structures are oversampled from the continuous space, they should be decoded into SMILES strings. The Decoder outputs are in the one-hot encoding form and are converted to SMILES strings, however, an additional post-processing 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 RDKit library (91) and drop the incorrect ones. However, to increase the sampling rate we introduced an error correction scheme, which is discussed in Section 2.5. Further, the correct and corrected strings are additionally filtered to avoid overlaps with datasets of known compounds.

2.5 Error correction for generated SMILES strings

In the experiments by Gómez-Bombarelli *et al.* (74) from 70% to less than 1% of the SMILES strings generated by the Autoencoder correspond to valid molecules. Therefore, one would have to overgenerate in order 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 Autoencoder output.

A successful attempt to tackle this issue was made in (58), where the regular 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' (58)). Ref. (92) extended the idea of SMILES grammar by enriching it with attributes which add some context awareness to the model. Another on-going research proposes to produce more correct SMILES strings by actively learning the valid sequences (93).

The problem of incorrect Autoencoder outputs can be decomposed into two subproblems. First, the latent space learned by the Autoencoder is too sparse and heterogeneous, so that the latent vector chosen for decoding by either of the methods mentioned in Section 2.4 already represents an invalid SMILES string. It could be solved by enforcing certain restrictions on the latent space, for example by jointly training a classifier network, classifying 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 completely different molecule, while one misplaced probability in the output from the final layer of Decoder would not affect the loss function considerably. We chose to focus on the latter part which by its essence resembles of a spell-checking task in NLP. Since 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), 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 post-processing to the Autoencoder 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 (94; 95). In order to exploit the sequential nature of strings both Encoder and Decoder are build of LSTM (Long Short-Term Memory) (96) cells with the hidden state of size 512. The Encoder transforms the input SMILES string X into a sequence of hidden states $(h_1, h_2, ..., h_{|X|})$, while the Decoder generates one symbol from SMILES characters dictionary (see Section 2.1) 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 $a, b = argmin(Y - b(a(X))^2$, 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 $h_{1:|X|}$:

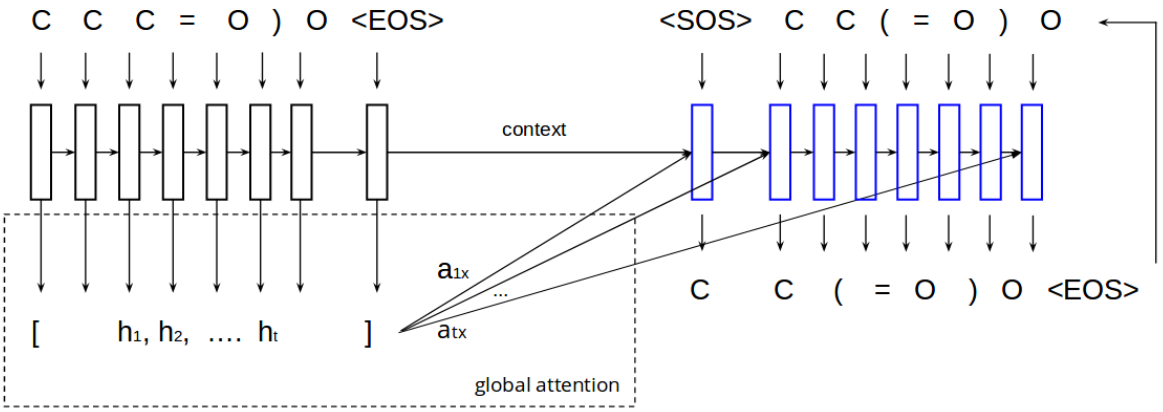


Figure 4. The architecture of the Attention-based Sequence-to-Sequence error correction model. The unrolled Encoder is pictured with black rectangles, the unrolled Decoder – with blue.

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

whose weights a_t are found by an attention mechanism:

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

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 NLP practices, symbols from SMILES strings are converted into embeddings (97), 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 (98), 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. Fig. 4 shows the schematic architecture and training procedure of the error correction model.

It may seem unnecessary to train a separate network for correcting the AE output. However, not only the model actually increases the number of valid SMILES strings, but also provides noteworthy insights into working with SMILES as with the specific language and offers some interpretability in the form of attention maps. Besides, comparing the performance of hidden vectors learned by the AE and by the error correction model on different tasks could grant us deeper understanding of strengths and weaknesses of each type of models.

2.7 Error correction data

Since the initial intention was to correct errors, made at the decoding stage, we collected all produced by the Autoencoder SMILES strings, which were tagged by RDKit library as invalid, and the corresponding original valid strings. That way we obtained the dataset for the AE error correction, consisting of ≈300.000 SMILES pairs. Obviously, the dataset formed in the way described above is biased and incomplete and

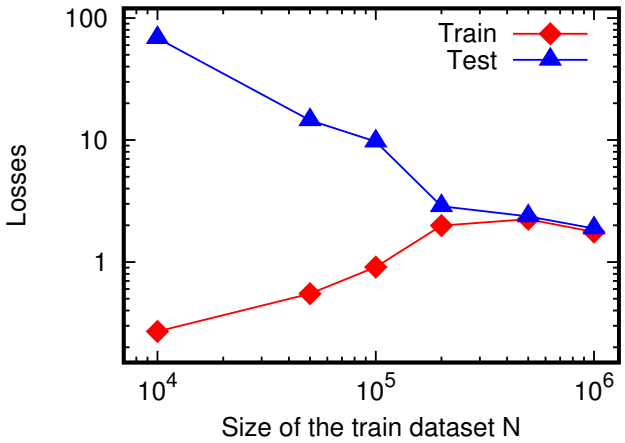


Figure 5. The Autoencoder loss dependence on the train dataset size.

the model trained on it would make very little use for any other SMILES-related task. Inspired by the successful application of denoising autoencoder (99) 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 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.

To sum up, we developed two models, identical in every aspect except for the nature of errors in SMILES, on which they were trained. For the sake of brevity, further on we will refer to those two models as AE noise model and random noise model.

3 RESULTS

3.1 Autoencoder 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 ≈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 Fig. 5.

The values for *500k* and *1M* cases show a relatively small difference in train-test gap. Hence we further decided to utilize a train dataset with 500,000 entities. Once the Autoencoder is trained it constitutes the latent space and we proceed with the generation 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 ≈15.000 SMILES pairs): valid SMILES strings – to check how accurate the model reconstructs the input strings; SMILES strings with AE noise and SMILES strings with

Table 1. Performance comparison of error correction models.

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

random noise – to test the model's ability to correct errors. The results are expressed by two numbers:
1. how many of 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 of outputted by the model strings are valid as verified by 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 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 only, since those language models are too different. As a result, the model outputs are hanging somewhere in between.

3.3 Statistics on generated SMILES strings

There were 95446 SMILES strings generated by the AE and SMOTE (as described in the subsection 2.4). Not all of them were grammatically correct. 39.7% (37890) of the generated strings were discarded during the full chemical parsing using 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 (as described in the subsection 2.7). Out of 37890 discarded strings, 12040 (31.8%) were corrected successfully.

One of the important questions we wanted to answer concerned the novelty of the generated and corrected molecules. Almost all corrected SMILES strings were unique (99,8% or 12024 out of 12040). Out of 57556 valid molecules generated by the AE, 89.8% (51720) 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 17860 unique novel molecules in total – 5836 from the AE (*generated* dataset) and 12024 from the error correction models (*corrected* dataset).

Unique generated and corrected compounds were compared to known ones in ChEMBL (100) and PubChem (101) 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 generated with SMOTE technique. However, taking into consideration the fact that out of all correctly generated SMILES strings 10.2%(5836) were unique, 12040 unique corrected strings accounted for **67%** for all newly generated

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

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.

It is also 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

Scaffold analysis. Generated dataset (5836 structures) contained 3945 unique scaffolds (the scaffold is a part of the molecule after removal of non-ring substituent, for molecules without rings it is the largest chain). For comparison, the source dataset of 189936 molecules used for SMOTE sampling procedure contained 58229 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). The data illustrate that the protocol generates novel molecules and corrects erroneous ones within a similar distribution and does not mirror the sampling set too closely.

Substructure features comparison. We calculated the following substructural features for 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 Sequence-to-Sequence model, show the distribution close to the ones of the source and generated datasets, although having some specific differences like 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.

Common functional groups. An algorithm to identify functional groups in organic molecules (102) 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 SMILES arbitrary target specification (SMARTS). SMARTS is a language for specifying substructural patterns in molecules (103). In particular, Fig. 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

corrected by the LSTM have very similar distributions of functional groups, again confirming that both novel and corrected datasets propagate over the same chemical space.

Common molecular descriptors comparison. Properties of generated, corrected and source data molecules were compared using common molecular descriptors (molecular weight, atom-based logP (104) 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. logP is the most commonly used measure of lipophilicity. This is the partition coefficient of a molecule between an aqueous and lipophilic phases, usually octanol and water. Results are shown in Fig. 7. It is clearly visible that all datasets have similar distributions of all the descriptors.

Synthetic accessibility score (SAS) as a number between 1 (easy to make) and 10 (very difficult to make) was calculated as described in (105). 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 Fig. 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.

3.5 Solubility prediction

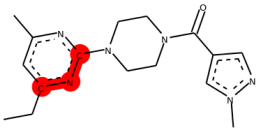
The quality of Predictor's solubility reconstruction is visualized in Fig. 8, reflecting how good the experimental values from train and test datasets are reproduced. Obviously, the perfect reconstruction is achieved if the predicted logS values fit the function $y = x$ (shown by the red lines). The solubility reconstruction within the train dataset is close to ideal, as shown on the left plot of the Fig. 8, and the corresponding coefficient of determination R^2 (in this case it coincides with the square of the Pearson correlation coefficient) for logS is equal to 0.99. The reconstruction within the test dataset (right plot) shows a certain degree of scatter around the ideal case with the coefficient of determination $R^2 = 0.86$. The thickness of the data cloud around $y = x$ roughly estimates the quality of prediction.

The difference between predicted and experimental logS values usually does not exceed unity (see the bottom plot in Fig. 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. On the other hand, t-sne and pca analysis have shown that close SMILES representations might demonstrate a highly scattered solubilities. Nevertheless, often even a rough estimate can be helpful to categorize a compound with respect to a degree of its solubility.

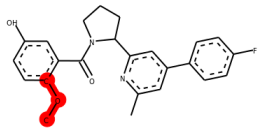
Similar plots comparing experimental vs. predicted logS are also reported in Refs. (46; 88). In particular Ref. (46) provides a set of coefficients of determination for different regression approaches. Our estimation of $R^2 = 0.86$ is slightly higher than the numbers provided in Ref. (46), however, 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

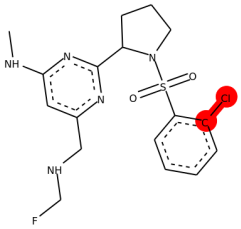
As an additional solubility prediction test we performed a short series of all-atom computer simulations. According to the IUPAC definition, solubility is the composition of a saturated solution expressed as a proportion of a solute in a solvent. In other words, the solubility of a solute defines its maximum concentration that can be dissolved in a solvent, otherwise a solute starts to aggregate and precipitates. 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, for example,



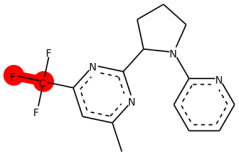
cnc 13.2% 12.9% 11.9%



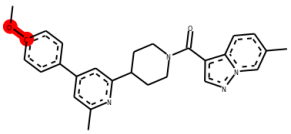
cOC 9.4% 7.0% 8.1%



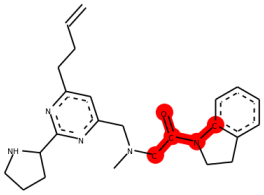
cCl 5.4% 5.4% 5.7%



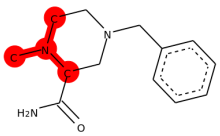
CF 3.6% 3.9% 3.5%



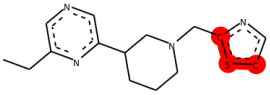
cO 1.5% 3.3% 2.0%



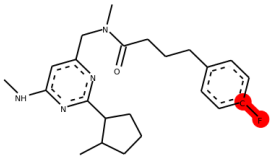
cNC(C)=O 3.3% 3.3% 3.0%



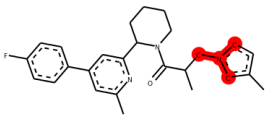
CN(C)C 2.9% 2.8% 2.8%



csc 3.2% 2.7% 2.8%



cF 2.7% 2.5% 2.6%



cn(c)C 2.8% 2.3% 2.2%

Figure 6. The most common substructures in datasets. Legend format: 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.

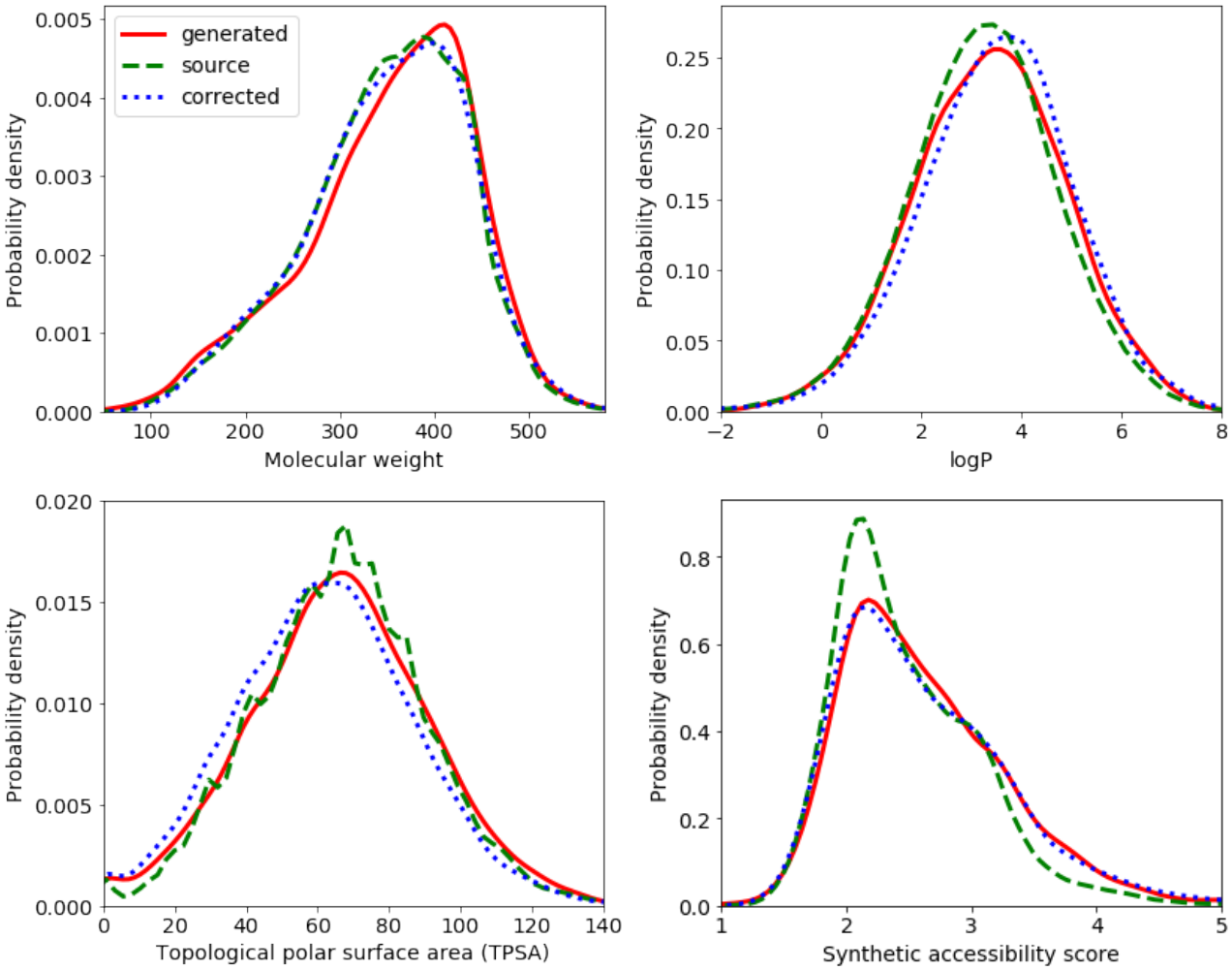


Figure 7. Comparison of probability density functions of molecular descriptors (molecular weight, logP, TPSA) and synthetic accessibility score for the source dataset, novel generated and corrected molecules.

Refs. (106; 107; 108)). 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)C(CC)CO (shown in Fig. 9). According the Predictor its log-solubility logS is -0.26, corresponding to the concentration of ≈ 0.55 mol/L.

For this purpose we "cooked" five all-atom mixtures consisting of solvent molecules H_2O and molecules of the solute $C_7H_{16}O_3$. The water was presented within the SPC/E model (109), while the solute molecules were constructed within the OPLS-AA forcefield (110). The Lennard-Jones parameters for unlike sites were calculated 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.

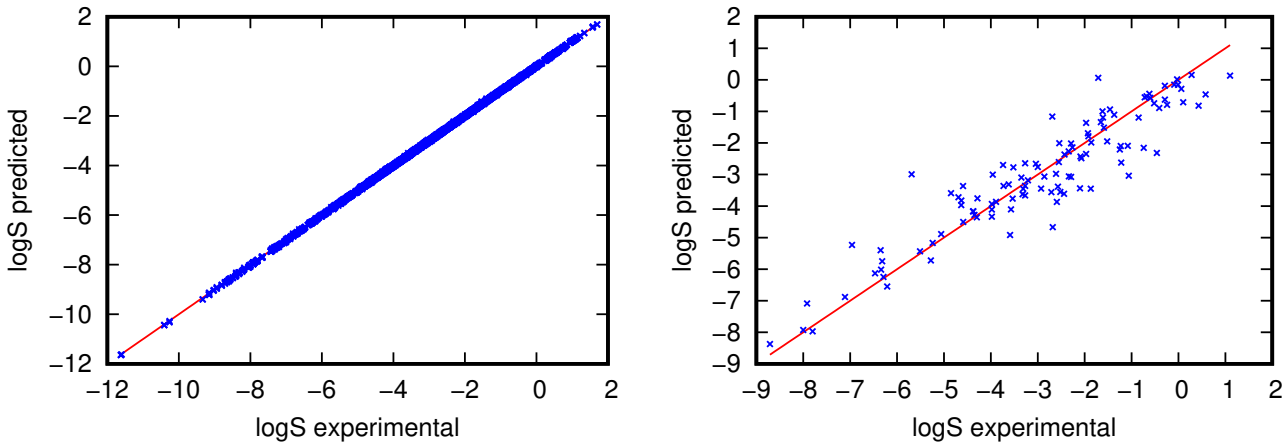


Figure 8. logS as reproduced on the train (left) and test (right) solubility datasets. The line $y = x$ marking the "ideal reconstruction" is plotted to guide the eye. The train dataset is reproduced perfectly, while the test one shows a certain degree of scatter.

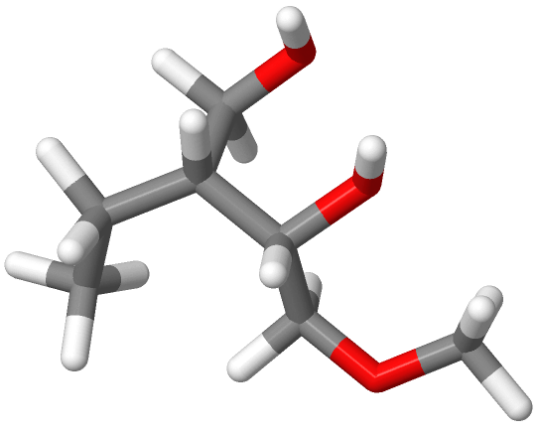


Figure 9. Spatial structure of a molecule COCC(O)C(CC)CO represented by sticks. Carbons are shown in gray, oxygens – in red, hydrogens – in white.

Table 3. The compositions and corresponding concentrations of the simulated systems.

number of molecules		solute concentration
H_2O	$C_7H_{16}O_3$	[mol/L]
20000	10	0.03
10000	20	0.11
10000	100	0.5
solubility limit		0.55
10000	400	1.7
10000	2000	4.2

The molecular dynamics simulations were performed with the use of DL_POLY package (111). In all the 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 (112). 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.

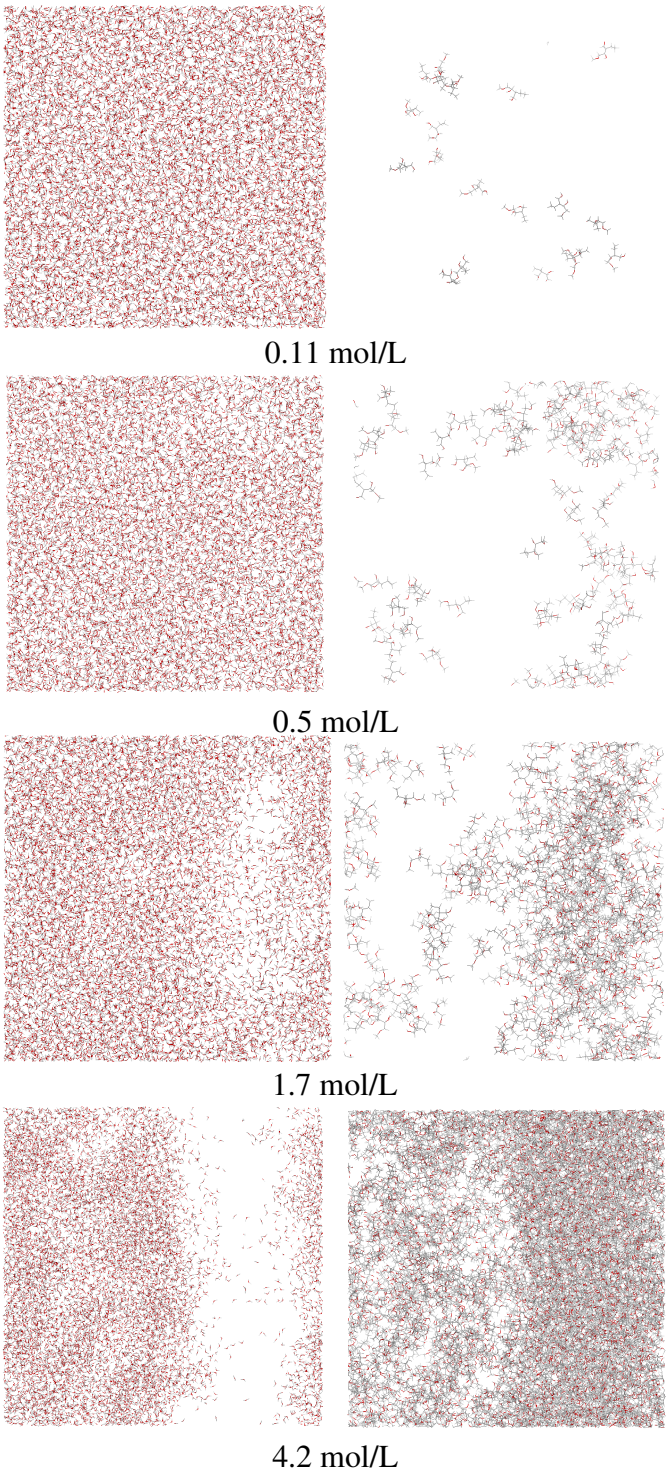


Figure 10. The snapshots of the simulation box, showing water H_2O (left) and $C_7H_{16}O_3$ (right) molecules as wireframes. Carbons are shown in gray, oxygens – in red, hydrogens – in white. The solute concentrations are 0.11/0.5/1.7/4.2 mol/L from the top to the bottom as denoted below each row. One can see the aggregation of solute molecules at the solute concentration of 1.7 mol/L and the areas not occupied by solvent or solute start to appear. This agrees with our solute-solvent separation prediction.

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 concentration of the

solute gradually increases over the 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 two snapshots, separately showing the simulation unit cell with only H_2O or $C_7H_{16}O_3$ molecules. These snapshots except the case of the lowest concentration of the solute (0.03 mol/L) are shown in Fig. 10. The snapshots in left column show H_2O molecules, right snapshot – $C_7H_{16}O_3$ molecules.

In the case of 0.11 mol/L the solvent (top left snapshot in Fig. 10) and solute (top right 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 for the lowest concentration of 0.03 mol/L, since they resemble the 0.11 mol/L case.

In the second top row of Fig. 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, left figure shows H_2O, right figure – $C_7H_{16}O_3$ molecules. One can notice that the solute is well distributed over the box and only begins to aggregate.

In the second last row we show the system with the solute concentration of 1.7 mol/L. This concentration is above the predicted solubility limit and clearly 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 bottom row of Fig. 10. One can clearly see the areas not occupied by H2O, as well as the areas not occupied by $C_7H_{16}O_3$. The actual composition is way above the solubility limit so the separation here is well manifested.

3.7 Druglikeliness

Druglikeness reflects the ability of an organism to pass a particular compound along the ADME chain and can be assessed for any molecule, however, it does not evaluate its pharmacological effect. Let us now check the above compound COCC(O)C(CC)CO for a druglikeness. In the Introduction section we have mentioned a bunch of bioavailability filters (52; 53; 54; 55; 56). All of them are based on simple metrics, like 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. Parameterized values for polar surface area, molar refractivity, logP one can obtain via RDKit library, solubility in water – by neural network, and the rest of metrics are directly defined from molecular structure. We present these values in Table 4

The above-mentioned filter rules can be found in the original papers, therefore, we will just briefly summarize the druglikeness scores against them. So, the filters of Lipinski (52), Egan (54), and Veber (56) report no violations for the compound and claim it to be druglike. The filter by Ghose (53) does not approve the compound for two reasons: i) its molecular weight of 148.2 g/mol does 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 filter by Ghose has two violations, but the failing values are just slightly below the qualifying ones. The properties predicted here resonate with ADME *bioavailability radar*, proposed by Daina *et al.* (113) in their SwissADME tool.

Table 4. ADME 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 sp3 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

4 DISCUSSION

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 (61; 74). Trained end-to-end such models result in the efficient structure of the chemical space where molecules with different properties could effectively 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 the 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 standard Autoencoder network that is 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 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. With this 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 both good generation and prediction quality for novel structures. We show that the error correction block allows increasing by 67% the number of unique and valid molecular structures in the output of basic autoencoder model. The prediction model trained directly on the embeddings of the reference dataset results in the state of the art estimation of solubility, $R^2 = 0.86$ for a coefficient of determination.

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 an 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 against various ADMET filters to assess their drug-likeness.

REFERENCES

[1]M. Dickson, J.P. Gagnon, Nat. Rev. Drug Discov., 3 (2004) 417–429. DOI: 10.1038/nrd1382

[2]A. Jahan, M.Y. Ismail, S.M. Sapuan, F. Mustapha, Mater. Des., 31 (2010) 696-705. DOI: 10.1016/j.matdes.2009.08.013

[3]A. Schuhmacher, O. Gassmann, M. Hinder, J. Transl. Med., 14 (2016) 105. DOI: 10.1186/s12967-016-0838-4

[4]J.C. Babiarz, Chapter 1. Overview of FDA and drug development. In D.J. Pisano, D.S. Mantus (Eds.) FDA Regulatory affairs. A guide for prescription drugs, medical devices, and biologics (2nd ed). New York: Informa Healthcare 2008.

[5]E. Petrova, Chapter 2. Innovation in the Pharmaceutical Industry: The Process of Drug Discovery and Development. In M. Ding *et al.* (Eds.) Innovation and Marketing in the Pharmaceutical Industry, International Series in Quantitative Marketing 20, New York: Springer Science+Business Media 2014. DOI: 10.1007/978-1-4614-7801-0_2

[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, PubChem Substance and Compound databases. Nucleic Acids Res. 44(D1) (2016) D1202-13. DOI: 10.1093/nar/gkv951

[7]P. Kirkpatrick, C. Ellis, Nature, 432 (2004) 823. DOI: 10.1038/432823a

[8]M. Webb, J. Van Reenen, C. Jones, N. Bloom, 2017 Meeting Papers, Society for Economic Dynamics, 566 (2017).

[9]Y. LeCunn, Y. Bengio, G. Hinton, Nature, 521 (2015) 436-444. DOI: 10.1038/nature14539

[10]G.B. Goh, N.O. Hodas, A. Vishnu, J. Comput. Chem., 38 (2017) 1291-1307. DOI: 10.1002/jcc.24764

[11]R. Miotto, F. Wang, S. Wang, X. Jiang, J.T. Dudley, Brief. Bioinform., 19(6) (2018) 1236-1246. DOI: 10.1093/bib/bbx044.

[12]G. Schneider, Nat. Rev. Drug Discov., 17 (2018) 97-113. DOI: 10.1038/nrd.2017.232

[13]J. Boström, D.G. Brown, R.J. Young, G.M. Keserü, Nat. Rev. Drug Discov., 17 (2018) 709-727. DOI: 10.1038/nrd.2018.116

[14]K.T. Butler, D.W. Davies, H. Cartwright, O. Isayev, A. Walsh, Nature, 559 (2018) 547-555. DOI: 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. Zholus, 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., 37 (2019) 1038-1040. DOI: 10.1038/s41587-019-0224-x

[16]M.O. Steinhauser, S. Hiermaier, Int. J. Mol. Sci., 10(12) (2009) 5135-5216. DOI: 10.3390/ijms10125135

[17]W. Hergert, M. Däne, A. Ernst (Eds.), Computational Materials Science. From Basic Principles to Material Properties. Berlin, Heidelberg: Springer 2004. DOI: 10.1007/b11279

[18]J. Behler, Neural network potential-energy surfaces for atomistic simulations. In M. Springborg (Ed.) Chemical Modelling: Applications and Theory Volume 7, The Royal Society of Chemistry 2010, 1–41. DOI: 10.1039/9781849730884-00001

[19]S.A. Ghasemi, A. Hofstetter, S. Saha, S. Goedecker, Phys. Rev. B, 92 (2015) 045131. DOI: 10.1103/PhysRevB.92.045131

[20]K.T. Schütt, F. Arbabzadah, S. Chmiela, K.R. Müller, A. Tkatchenko, Nat. Comm., 8 (2017) 13890. DOI: 10.1038/ncomms13890

[21]J. Carrasquilla, R.G. Melko, Nat. Phys. 13 (2017) 431-434. DOI: 10.1038/nphys4035

[22]T. Xie, J.C. Grossman, Phys. Rev. Lett. 120 (2018) 145301. DOI: 10.1103/PhysRevLett.120.145301

[23]K. Ryan, J. Lengyel, M. Shatruk, J. Am. Chem. Soc., 140(32) (2018) 10158-10168. DOI: 10.1021/jacs.8b03913

[24]S. Amabilino, L.A. Bratholm, S.J. Bennie, A.C. Vaucher, M. Reiher, D.R. Glowacki, J. Phys. Chem. A, 123(20) (2019) 4486-4499. DOI: 10.1021/acs.jpca.9b01006

[25]F.E. Bock, R.C. Aydin, C.J. Cyron, N. Huber, S.R. Kalidindi, B. Klusemann, Front. Mater. 6:110 (2019). DOI: 10.3389/fmats.2019.00110

[26]M. Haghighatlari, J. Hachmann, Curr. Opin. Chem. Eng., 23 (2019) 51-57. DOI: 10.1016/j.coche.2019.02.009

[27]S. Chiriki, S.S. Bulusu, Chem. Phys. Lett., 652 (2016) 130–135. DOI: 10.1016/j.cplett.2016.04.013

[28]L. Shen, W. Yang, J. Chem. Theory Comput., 14 (2018) 1442-1455. DOI: 10.1021/acs.jctc.7b01195

[29]S. Jindal, S.S. Bulusu, J. Chem. Phys. 149 (2018) 194101. DOI: 10.1063/1.5043247

[30]R. Kondor, arXiv:1810.06204 [physics.chem-ph] (2018).

[31]K.T. Schütt, H.E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, J. Chem. Phys. 148 (2018) 241722. DOI: 10.1063/1.5019779

[32]A. Pérez, G. Martínez-Rosell, G. De Fabritiis, Curr. Opin. Struct. Biol. 49 (2018) 139-144. DOI: 10.1016/j.sbi.2018.02.004

[33]J. Herr, K. Yao, R. McIntyre, D.W. Toth, J. Parkhill, J. Chem. Phys. 148 (2018) 241710. DOI: 10.1063/1.5020067

[34]K. Yao, J.E. Herr, D.W. Toth, R. MckIntyre, J. Parkhill, Chem. Sci., 9 (2018) 2261-2269. DOI: 10.1039/C7SC04934J

[35]H. Wang, L. Zhang, J. Han, W. E, Comput. Phys. Commun., 228 (2018) 178-184. DOI: 10.1016/j.cpc.2018.03.016

[36]L. Zhang, H. Wang, J. Han, R. Car, W. E, Phys. Rev. Lett., 120 (2018) 143001. DOI: 10.1103/PhysRevLett.120.143001

[37]L. Zhang, H. Wang, W. E, J. Chem. Phys., 149 (2018) 154107. DOI: 10.1063/1.5042714

[38]L. Zhang, J. Han, H. Wang, R. Car, W. E, J. Chem. Phys., 149 (2018) 034101 (2018). DOI: 10.1063/1.5027645

[39]A. Lusci, G. Pollastri, P. Baldi, J. Chem. Inf. Model., 53 (2013) 1563-1575. DOI: 10.1021/ci400187y

[40]G.E. Dahl, N. Jaitly, R. Salakhutdinov, arXiv:1406.1231 [stat.ML] (2014).

[41]E.O. Pyzer-Knapp, K. Li, A. Aspuru-Guzik, Adv. Funct. Mater., 25 (2015) 6495-6502. DOI: 10.1002/adfm.201501919

[42]B. Alipanahi, A. Delong, M.T. Weirauch, B.J. Frey, Nature Biotechnol., 33 (2015) 831–838. DOI: 10.1038/nbt.3300

[43]I. Wallach, M. Dzamba, A. Heifets, arXiv:1510.02855 [cs.LG] (2015).

[44]A. Mayr, G. Klambauer, T. Unterthiner, S. Hochreiter, Front. Environ. Sci. 3:80 (2016). DOI: 10.3389/fenvs.2015.00080

[45]E.J. Bjerrum, arXiv:1703.07076 [cs.LG] (2017).

[46]A.K.Sharma, G.N. Srivastava, A. Roy,V.K.Sharma, Front. Pharmacol., 8:880 (2017). DOI: 10.3389/fphar.2017.00880

[47]S. Kearnes, B. Goldman, V. Pande, arXiv:1606.08793 [stat.ML] (2016).

[48]J.Jiménez, M. Škalič, G. Martínez-Rosell, G. De Fabritiis, J. Chem. Inf. Model., 58 (2018) 287-296. DOI: 10.1021/acs.jcim.7b00650

[49]G.B. Goh, N.O. Hodas, C. Siegel, A. Vishnu, arXiv:1712.02034 [stat.ML] (2017).

[50]G.B. Goh, C. Siegel, A. Vishnu, N.O. Hodas, arXiv:1712.02734 [stat.ML] (2017).

[51]N. Ståhl, G. Falkman, A. Karlsson, G. Mathiason, J. Boström, J. Integr. Bioinform., 20180065 (2018) 1613-4516. DOI: 10.1515/jib-2018-0065

[52]C.A. Lipinski, F. Lombardo, B.W. Dominy, P.J. Feeney, Adv. Drug Deliv. Rev. 46 (2001) 3-26. DOI: 10.1016/S0169-409X(96)00423-1

[53]A.K. Ghose, V.N. Viswanadhan, J.J. Wendoloski, J. Comb. Chem. 1 (1999) 55-68. DOI: 10.1021/cc9800071

[54]W.J. Egan, K.M. Merz, J.J. Baldwin, J. Med. Chem. 43 (2000) 3867-3877. DOI: 10.1021/jm000292e

[55]I. Muegge, S.L. Heald, D. Brittelli, J. Med. Chem. 44 (2001) 1841-1846. DOI: 10.1021/jm015507e

[56]D.F. Veber, S.R. Johnson, H.Y. Cheng, B.R. Smith, K.W. Ward, K.D. Kopple, J. Med. Chem. 45 (2002) 2615-2623. DOI: 10.1021/jm020017n

[57]M.H. Segler, T. Kogej, C. Tyrchan, M.P. Waller, ACS Cent. Sci. 4(1) (2018) 120-131. DOI: 10.1021/acscentsci.7b00512

[58]M.J. Kusner, B. Paige, J.M. Hernández-Lobato, Grammar Variational Autoencoder. In D. Precup, Y.W. Teh (Eds.) ICML'17 Proceedings of the 34th International Conference on Machine Learning – 70, Sydney, NSW 2017, 1945-1954.

[59]G.B. Goh, C. Siegel, A. Vishnu, N.O. Hodas, N. Baker, arXiv:1710.02238 [stat.ML] (2017).

[60]G.B. Goh, K. Sakloth, C. Siegel, A. Vishnu, J. Pfaendtner, arXiv:1808.04456 [cs.LG] (2018).

[61]D. Kuzminykh, D. Polykovskiy, A. Kadurin, A. Zhebrak, I. Baskov, S. Nikolenko, R. Shayakhmetov, A. Zhavoronkov, Mol. Pharmaceutics, 15 (2018) 4378-4385. DOI: 10.1021/acs.molpharmaceut.7b01134

[62]Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, arXiv:1901.00596v2 [cs.LG] (2019).

[63]J.Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, arXiv:1812.08434v3 [cs.LG] (2019).

[64]S. Kearnes, K. McCloskey, M. Berndl V. Pande, P. Riley, J. Comput. Aided Mol. Des. 30(8) (2016) 595-608. DOI: 10.1007/s10822-016-9938-8

[65]D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional Networks on Graphs for Learning Molecular Fingerprints. In C. Cortes *et al.* (Eds.) Advances in Neural Information Processing Systems 28 (NIPS 2015) Curran Associates, Inc. 2015, 2224-2232.

[66]J. You, B. Liu, R. Ying, V. Pande, J. Leskovec, arXiv:1806.02473v3 [cs.LG] (2019).

[67]A. Fout, J. Byrd, B. Shariat, A. Ben-Hur, Protein interface prediction using graph convolutional networks. In I. Guyon *et al.* (Eds.) Advances in Neural Information Processing Systems 30 (NIPS 2017) Curran Associates, Inc. 2017, 6530-6539.

[68]S. Rhee, S. Seo, S. Kim, arXiv:1711.05859v3 [cs.CV] (2018).

[69]M. Zitnik, M. Agrawal, J. Leskovec, arXiv:1802.00543v2 [cs.LG] (2018).

[70]N. De Cao, T. Kipf, arXiv:1805.11973v1 [stat.ML] (2018).

[71]I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets. In Z. Ghahramani *et al.* (Eds.) Advances in Neural Information Processing Systems 27 (NIPS 2014) Curran Associates, Inc. 2014, 2672-2680.

[72]A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A.A. Bharath, IEEE-SPM 35(1) (2018) 53-65. DOI: 10.1109/MSP.2017.2765202

[73]P.B. Jørgensen, M.N. Schmidt, O. Winther, Mol. Inf., 37 (2018) 1700133. DOI: 10.1002/minf.201700133

[74]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., 4(2) (2018) 268-276. DOI: 10.1021/acscentsci.7b00572

[75]A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, A. Zhavoronkov, Mol. Pharmaceutics, 14 (2017) 3098-3104. DOI: 10.1021/acs.molpharmaceut.7b00346

[76]E. Putin,A. Asadulaev, Q. Vanhaelen, Y. Ivanenkov, A.V. Aladinskaya, A. Aliper, A. Zhavoronkov, Mol. Pharmaceutics, 15 (2018) 4386-4397. DOI: 10.1021/acs.molpharmaceut.7b01137

[77]T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath, H. Chen, Mol. Inf., 37 (2018) 1700123. DOI: 10.1002/minf.201700123

[78]E.J. Bjerrum, B. Sattarov, Biomolecules, 8 (2018) 131. DOI:10.3390/biom8040131

[79]G. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P.L.C. Farias, A. Aspuru-Guzik, arXiv:1705.10843v3 [stat.ML] (2018).

[80]G.E. Hinton, R.S. Zemel, Autoencoders, minimum description length and Helmholtz free energy. In J.D. Cowan, G. Tesauro, J. Alspector (Eds.) Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS 1993) Morgan Kaufmann Publishers Inc. 1993, 3-10.

[81]D.P. Kingma, M. Welling, arXiv:1312.6114v10 [stat.ML] (2014).

[82]https://www.emolecules.com/info/plus/download-database

[83]J. Huuskonen, J. Chem. Inf. Comput. Sci., 40 (2000) 773-777. DOI: 10.1021/ci9901338

[84]T. Hou, K. Xia, W. Zhang, X. Xu, J. Chem. Inf. Comput. Sci., 44 (2004) 44 266-275. DOI: 10.1021/ci034184n

[85]J.S. Delaney, J. Chem. Inf. Comput. Sci., 44 (2004) 1000-1005. DOI: 10.1021/ci034243x

[86]DLS-100 Solubility Dataset. DOI: 10.17630/3a3a5abc-8458-4924-8e6c-b804347605e8

[87]A. Llinàs, R.C. Glen, J.M. Goodman, J. Chem. Inf. Model., 48 (2008) 1289-1303. DOI: 10.1021/ci800058v

[88]A.J. Hopfinger, E.X. Esposito, A. Llinàs, R.C. Glen, J.M. Goodman, J. Chem. Inf. Model., 49 (2009) 1-5. DOI: 10.1021/ci800436c

[89]N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P.Kegelmeyer, J. Artif. Int. Res., 16 (2002) 321-357. DOI: 10.1613/jair.953

[90]G. Lemaître, F. Nogueira, C.K. Aridas, J. Mach. Learn. Res., 18 (2017) 1-5.

[91]RDKit: Open-source cheminformatics; http://www.rdkit.org

[92]H. Dai, Y. Tian, B. Dai, S. Skiena, L. Song, arXiv:1802.08786v1 [cs.LG] (2018).

[93]D. Janz, J.V. Westhuizen, J.M. Hernández-Lobato, arXiv:1708.04465v1 [stat.ML] (2017).

[94]I. Sutskever, O. Vinyals, Q.V. Le, Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani *et al.* (Eds.) Advances in Neural Information Processing Systems 27 (NIPS 2014) Curran Associates, Inc. 2014, 3104-3112.

[95]D. Bahdanau, K. Cho, Y. Bengio, arXiv:1409.0473 [cs.CL] (2014).

[96]S. Hochreiter, J. Schmidhuber, Neural Comput., 9(8) (1997) 1735-1780. DOI: 10.1162/neco.1997.9.8.1735

[97]T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality. In C.J.C. Burges *et al.* Advances in Neural Information Processing Systems 26 (NIPS 2013) Curran Associates, Inc. 2013, 3111–3119.

[98]A. Lamb, A. Goyal, S. Zhang, A.C. Courville, Y. Bengio, Professor Forcing: A New Algorithm for Training Recurrent Networks. In D.D. Lee *et al.* (Eds.) Advances in Neural Information Processing Systems 29 (NIPS 2016) Curran Associates, Inc. 2016, 4601–4609.

[99]P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning 2008, 1096-1103. DOI: 10.1145/1390156.1390294

[100]https://www.ebi.ac.uk/chembl/

[101]https://pubchem.ncbi.nlm.nih.gov/

[102]P. Ertl, J. Cheminformatics 9 (2017) 36. DOI: 10.1186/s13321-017-0225-z

[103]SMARTS Theory Manual, Daylight Chemical Information Systems. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html

[104]A. Wildman, G.M. Crippen, J. Chem. Inf. Comput. Sci., 395 (1999) 868-873. DOI: 10.1021/ci990307l

[105]P. Ertl, A. Schuffenhauer, J. Cheminformatics 1 (2009) 8. DOI: 10.1186/1758-2946-1-8

[106]L. Li, T. Totton, D. Frenkel, J. Chem. Phys., 146 (2017) 214110. DOI: 10.1063/1.4983754

[107]R. Gillet, A. Fierro, L.M. Valenzuela, J.R. Pérez-Correa, Fluid Phase Equilib. 472 (2018) 85-93. DOI: 10.1016/j.fluid.2018.05.013

[108]G. Duarte Ramos Matos, D.L. Mobley, F1000Research, 7 (2019) 686. DOI: 10.12688/f1000research.14960.2

[109]H.J.C. Berendsen, J.R. Grigera, T.P. Straatsma, J. Phys. Chem. 91 (1987) 6269-6271. DOI: 10.1021/j100308a038

[110]W.L. Jorgensen, D.S. Maxwell, J. Tirado-Rives, J. Am. Chem. Soc. 118 (1996) 11225–11236. DOI: 10.1021/ja9621760

[111]W. Smith, T. Forester, I. Todorov, "The DL POLY Classic User Manual", STFC Daresbury Laboratory, 2010, available from http://www.ccp5.ac.uk/DL_POLY_C/

[112]S. Melchionna, G. Ciccotti, B.L. Holian, Molec. Phys. 78 (1993) 533–544. DOI: 10.1080/00268979300100371

[113]A. Daina, O. Michielin, V. Zoete, Sci. Rep. 7 (2017) 42717. DOI: 10.1038/srep42717

APPENDIX: ARCHITECTURES OF AUTOENCODER AND PREDICTOR

The Autoencoder consists of two sub-networks – Encoder and Decoder (Fig. 11). Both the Encoder and Decoder utilize convolution and fully-connected type layers. The dimensions of the layers are chosen to maintain a constant number of neurons per layer, allowing to apply a residual technique, when a layer output is mixed with outputs of subsequent layers by skip connections, helping 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. The Predictor's architecture consists of four residual blocks and four FC layers. As seen in Fig. 12 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

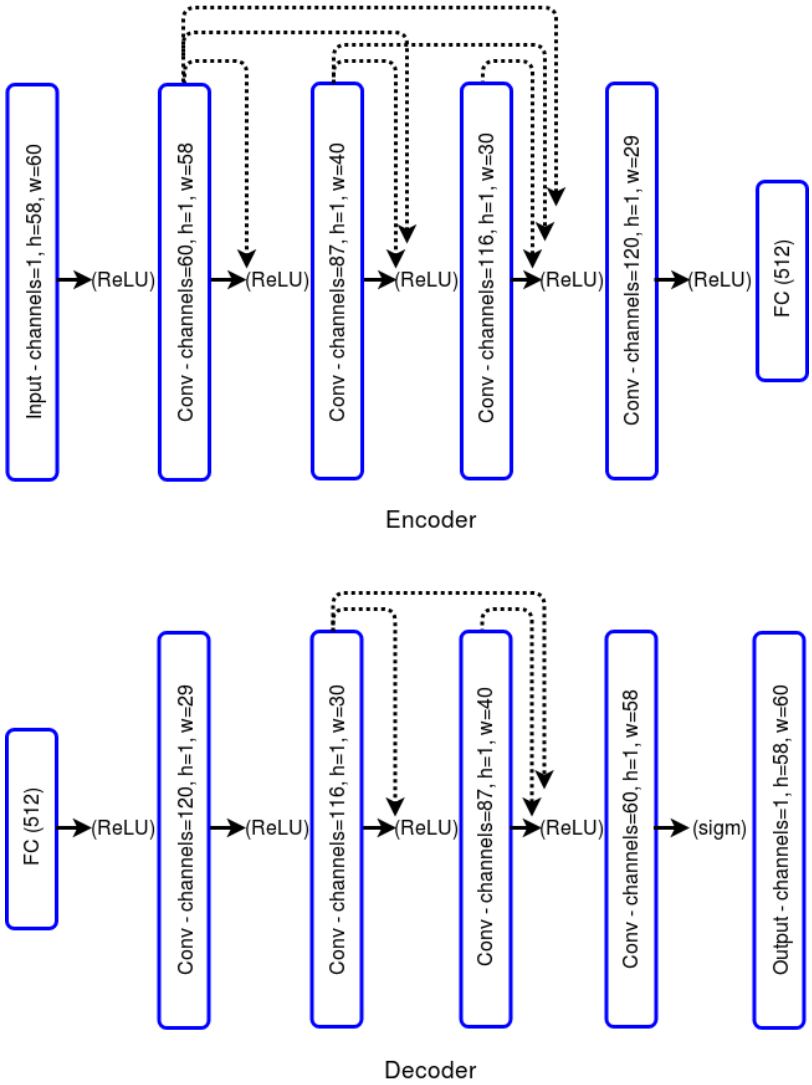


Figure 11. The architecture of the Autoencoder. The upper chart presents the Encoder: differentiable connections between layers are denoted by black solid arrows, while the dotted ones denote non-weighted addition with the appropriate reshape, followed by the activation function of the ReLU or sigmoid 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.

FC outputs (denoted by dotted arrows). The Predictor outputs a single value, being treated as a solubility prediction.

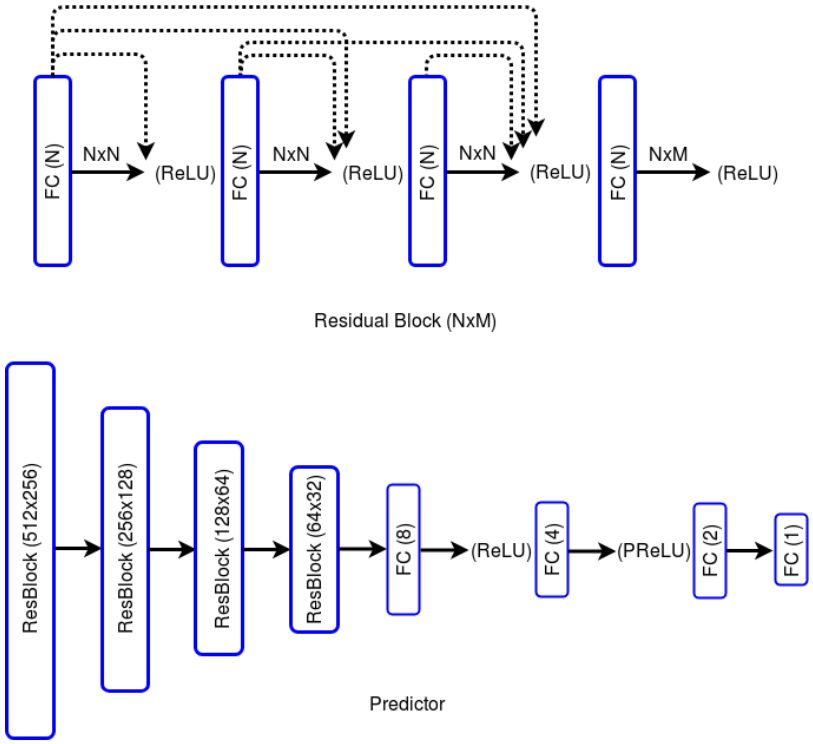


Figure 12. The architecture of the Predictor. The upper chart shows a residual block: differentiable connections between FC layers are denoted by solid arrows, the dotted arrows denote non-weighted addition, then followed by activation function of the ReLU type. The bottom chart presents the general scheme of the Predictor: four residual blocks are followed by four fully connected layers.