RESEARCH

Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks

Marwin HS Segler[1*], Thierry Kogej[2], Christian Tyrchan[3] and Mark P Waller[4]

Abstract

In *de novo* drug design, computational strategies are used to generate novel molecules with good affinity to the desired biological target. In this work, we show that recurrent neural networks can be trained as generative models for molecular structures, similar to statistical language models in natural language processing. We demonstrate that the properties of the generated molecules correlate very well with the properties of the molecules used to train the model. In order to enrich libraries with molecules active towards a given biological target, we propose to fine-tune the model with small sets of molecules, which are known to be active against that target.

Against Staphylococcus aureus, the model reproduced 14% of 6051 hold-out test molecules that medicinal chemists designed, whereas against Plasmodium falciparum (Malaria) it reproduced 28% of 1240 test molecules. When coupled with a scoring function, our model can perform the complete *de novo* drug design cycle to generate large sets of novel molecules for drug discovery.

Keywords: computer-assisted drug design; recurrent neural networks

1 Introduction

Chemistry is the language of nature. Chemists speak it fluently and have made their discipline one of the true contributors to human well-being, which has *"change[d] the way you live and die"*.[1] This is particularly true for medicinal chemistry. However, creating novel drugs is an extraordinarily hard and complex problem.[2] One of the many challenges in drug design is the sheer size of the search space for novel molecules. It has been estimated that 10^{60} drug-like molecules could possibly be synthetically accessible.[3] Chemists have to select and examine molecules from this large space to find molecules that are active towards a biological target. Active means for example that a molecule binds to a biomolecule, which causes an effect in the living organism, or inhibits replication of bacteria. Modern high-throughput screening techniques allow to test molecules in the order of 10^6 in the lab.[4] However, larger experiments will get prohibitively expensive. Given this practical limitation of *in vitro* experiments, it is desirable to have computational tools to narrow down the enormous search space. *Virtual screening* is a commonly used strategy to search for promising molecules amongst millions of existing or billions of virtual molecules.[5] Searching can be carried out using similarity-based metrics, which provides a quantifiable numerical indicator of closeness between molecules. In contrast, in *de-novo* drug design, one aims to directly create novel molecules that are active towards the desired biological target.[6, 7] Here, like in any molecular design task, the computer has to

i create molecules,
ii score and filter them, and
iii search for better molecules, building on the knowledge gained in the previous steps.

Task i, the generation of novel molecules, is usually solved with one of two different protocols.[7] One strategy is to build molecules from predefined groups of atoms or fragments. Unfortunately, these approaches often lead to molecules that are very hard to synthesise.[8] Therefore, another established approach is to conduct virtual chemical reactions based on expert coded rules, with the hope that these reactions could then also be applied in practice to make the molecules in the laboratory.[9] These systems give reasonable drug-like molecules, and are considered as "the solution" to the structure generation problem.[2] We generally share this view. However, we have recently shown that the predicted reactions from these rule-based expert systems can sometimes fail.[10] Also, fo-

*Correspondence: marwin.segler@uni-muenster.de
[1]Institute of Organic Chemistry & Center for Multiscale Theory and Computation, Westfälische Wilhelms-Universität, Münster, Germany
Full list of author information is available at the end of the article

cussing on a small set of robust reactions can unnecessarily restrict the possibly accessible chemical space.

Task ii, scoring molecules and filtering out undesired structures, can be solved with substructure filters for undesirable reactive groups in conjunction with established approaches such as docking[11] or machine-learning (ML) approaches.[7, 12, 13] The ML approaches are split into two branches: Target prediction classifies molecules into active and inactive, and quantitative structure-activity relationships (QSAR) seek to quantitatively predict a real-valued measure for the effectiveness of a substance (as a regression problem). As molecular descriptors, Signature Fingerprints, Extended-Connectivity (ECFP) and atom pair (APFP) fingerprints and their fuzzy variants are the *de-facto* standard today.[14–16] Convolutional Networks on Graphs are a recent addition to the field of molecular descriptors.[17, 18] Random Forests and Neural Networks are currently the most widely used machine learning models for target prediction.[19–31]

This leads to task iii, the search for molecules with the right binding affinity combined with optimal molecular properties. In earlier work, this was performed (among others) with classical global optimisation techniques, for example genetic algorithms or ant-colony optimisation.[7, 32] Furthermore, *de novo* design is related to inverse QSAR.[33–36] While in *de novo* design design, a *regular* QSAR mapping $X \rightarrow y$ from molecular descriptor space X to properties y is used as the scoring function for the global optimizer, in *inverse* QSAR one aims to find an explicit inverse mapping $y \rightarrow X$, and then maps back from optimal points in descriptor space X to valid molecules. However, this is not well defined, because molecules are inherently discrete. Several protocols have been developed to address this, for example enumerating all structures within the constraints of hyper-rectangles in the descriptor space.[33–38] Gómez-Bombarelli *et al.* proposed to learn continuous representations of molecules with variational auto-encoders, based on the model by Bowman *et al.*,[39] and to perform Bayesian optimisation in this vector space to optimise molecular properties.[40] Nevertheless, this approach was not applied to create active drug molecules, and did not succeed in optimising more complex molecular properties, such as emission color and delayed fluorescence decay rate (k_{TADF}).[40]

In this work, we suggest a novel, completely data-driven *de novo* drug design approach. It relies only on a generative model for molecular structures, based on a recurrent neural network, that is trained on large sets of molecules. Generative models learn a probability distribution over the training examples; sampling from this distribution generates new examples similar to the training data. Intuitively, a generative model for molecules trained on drug molecules would "know" how valid and reasonable drug-like molecules look like, and could be used to generate more drug-like molecules. However, for molecules, these models have been studied rarely, and rigorously only with traditional models such as Gaussian mixture models (GMM).[37, 41] Recently, recurrent neural networks (RNNs) have emerged as powerful generative models in very different domains, such as natural language processing,[42] speech,[43] images,[44] video,[45] formal languages,[46] computer code generation,[47] and music scores.[48] In this work, we highlight the analogy of language and chemistry, and show that RNNs can also generate reasonable molecules. Furthermore, we demonstrate that RNNs can also transfer their learned knowledge from large molecule sets to directly produce novel molecules that are biologically active by retraining the models on small sets of already known actives. We test our models by reproducing hold-out test sets of known biologically active molecules.

2 Methods

2.1 Representing Molecules

To connect chemistry with language, it is important to understand how molecules are represented. Usually, they are modeled by molecular graphs, also called Lewis structures in chemistry. In molecular graphs, atoms are labeled nodes. The edges are the bonds between atoms, which are labeled with the bond order (e.g. single, double or triple). One could therefore envision having a model that reads and outputs graphs. Several common chemistry formats store molecules in such a manner. However, in models for natural language processing, the input and output of the model are usually sequences of single letters, strings or words. We therefore employ the SMILES format, which encodes molecular graphs compactly as human-readable strings. SMILES is a formal grammar which describes molecules with an alphabet of characters, for example `c` and `C` for aromatic and aliphatic carbon atoms, `O` for oxygen, `-`, `=` and `#` for single, double and triple bonds (see Figure 1).[49] To indicate rings, a number is introduced at the two atoms where the ring is closed. For example, benzene in aromatic SMILES notation would be `c1ccccc1`. Side chains are denoted by round brackets. To generate valid SMILES, the generative model would have to learn the SMILES grammar, which includes keeping track of rings and brackets to eventually close them. In morphine, a complex natural product, the number of steps between the first `1` and the second `1`, indicating a ring, is 32. Having established a link between molecules and (formal) language, we can now discuss language models.

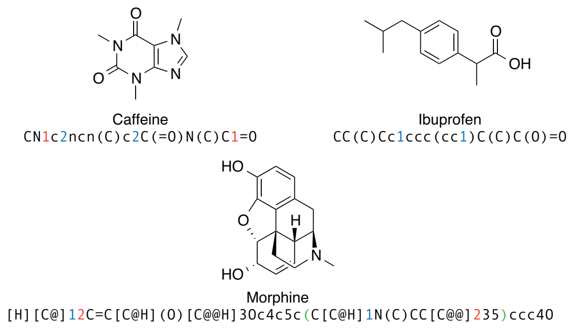
Caffeine
CN1c2ncn(C)c2C(=O)N(C)C1=O
Ibuprofen
CC(C)Cc1ccc(cc1)C(C)C(O)=O
Morphine
[H][C@]12C=C[C@H](O)[C@@H]3Oc4c5c(C[C@H]1N(C)CC[C@@]235)ccc4O

Figure 1 Examples of molecules and their SMILES representation. To correctly create smiles, the model has to learn long term dependencies, for example to close rings (indicated by numbers) and brackets.

2.2 Language Models and Recurrent Neural Networks

Given a sequence of words $(w_1, ..., w_i)$, language models predict the distribution of the $(i+1)$th word w_{i+1}.[50] For example, if a language model receives the sequence `"Chemistry is"`, it would assign different probabilities to possible next words. `"fascinating"`, `"important"`, or `"challenging"` would receive high probabilities, while `"runs"` or `"potato"` would receive very low probabilities. Language models can both capture the grammatical correctness ("runs" in this sentence is wrong) and the meaning ("potato" does not make sense). Language models are implemented for example in message autocorrection in many modern smartphones. Interestingly, language models do not have to use words. They can also be based on characters or letters.[50] In that case, when receiving the sequence of characters `chemistr`, it would assign a high probability to `y`, but a low probability to `q`. To model molecules instead of language, we simply swap words or letters with atoms, or, more practically, characters in the SMILES alphabet, which form a (formal) language. For example, if the model receives the sequence `c1ccccc`, there is a high probability that the next symbol would be a `"1"`, which closes the ring, and yields benzene.

More formally, to a sequence S of symbols s_i at steps $t_i \in T$, the language model assigns a probability of

$$P_\theta(S) = P_\theta(s_1) \cdot \prod_{t=2}^{T} P_\theta(s_t | s_{t-1}, ..., s_1) \tag{1}$$

where the parameters θ are learned from the training set.[50] In this work, we use a recurrent neural network (RNN) to estimate the probabilities of Equation 1. In contrast to regular feedforward neural networks, RNNs maintain state, which is needed to keep track of the symbols seen earlier in the sequence. In abstract terms, an RNN takes a sequence of input vectors $\mathbf{x}_{1:n} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ and an initial state vector $\mathbf{h}_0$, and returns a sequence of state vectors $\mathbf{h}_{1:n} = (\mathbf{h}_1, ..., \mathbf{h}_n)$ and a sequence of output vectors $\mathbf{y}_{1:n} = (\mathbf{y}_1, ..., \mathbf{y}_n)$. The RNN consists of a recursively defined function R, which takes a state vector $\mathbf{h}_i$ and input vector $\mathbf{x}_{i+1}$ and returns a new state vector $\mathbf{h}_{i+1}$. Another function O maps a state vector $\mathbf{h}_i$ to an output vector $\mathbf{y}_i$.[50]

$$\mathrm{RNN}(\mathbf{h}_0, \mathbf{x}_{1:n}) = \mathbf{h}_{1:n}, \mathbf{y}_{1:n} \tag{2}$$
$$\mathbf{h}_i = R(\mathbf{h}_{i-1}, \mathbf{x}_i) \tag{3}$$
$$\mathbf{y}_i = O(\mathbf{h}_i) \tag{4}$$

The state vector $\mathbf{h}_i$ stores a representation of the information about all symbols seen in the sequence so far. As an alternative to the recursive definition, the recurrent network can also be *unrolled* for finite sequences (see Figure 2). An unrolled RNN can be seen as a very deep neural network, in which the parameters θ are shared among the layers, and the hidden state $\mathbf{h}_t$ is passed as an additional input to the next layer. Training the unrolled RNN to fit the parameters θ can then simply be done by using backpropagation to compute the gradients with respect to the loss function, which is categorical cross-entropy in this work.[50]

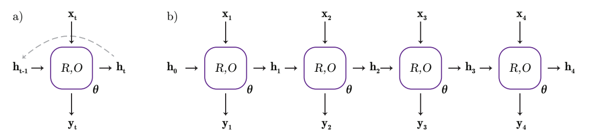

Figure 2 a) Recursively defined RNN b) The same RNN, unrolled. The parameters θ (the weight matrices of the neural network) are shared over all time steps.

As the specific RNN function, in this work, we use the Long Short Term Memory (LSTM), which was introduced by Hochreiter and Schmidhuber.[51] It has been used successfully in many natural language processing tasks,[42] for example in Google's Neural Machine Translation system.[52] For excellent in-depth discussions of the LSTM, we refer to the articles by Goldberg,[50] Graves,[53] Olah,[54] and Greff *et al.*[55]

To encode the SMILES symbols as input vectors $\mathbf{x}_t$, we employ the "one-hot" representation.[53] This means if there are K symbols, and k is the symbol to be input at step t, then we can construct an input vector $\mathbf{x}_t$ with length K, whose entries are all zero except the k-th entry, which is one. If we assume a very restricted set of symbols {`c`, `1`, `\n`}, input `c` would correspond to $\mathbf{x}_t = (1, 0, 0)$, `1` to $\mathbf{x}_t = (0, 1, 0)$ and `\n` to $\mathbf{x}_t = (0, 0, 1)$.

The probability distribution $P_\theta(s_{t+1}|s_t, ..., s_1)$ of the next symbol given the already seen sequence is thus

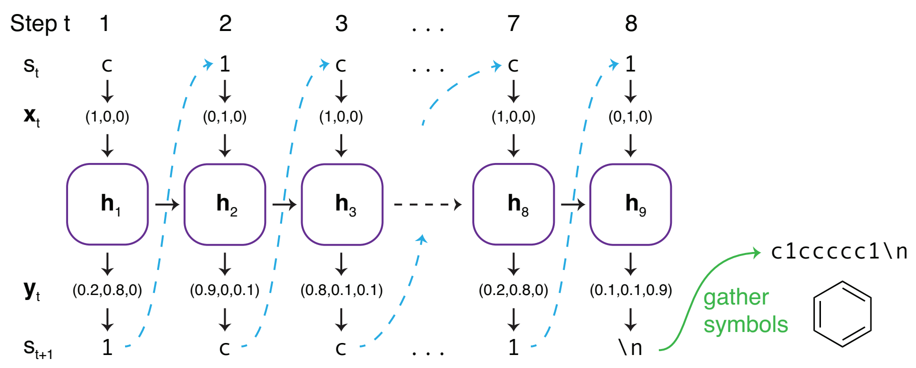

Figure 3 The Symbol Generation and Sampling Process. We start with a random seed symbol $\mathbf{s}_1$, here c, which gets converted into a one-hot vector $\mathbf{x}_1$ and input into the model. The model then updates its internal state $\mathbf{h}_0$ to $\mathbf{h}_1$ and outputs $\mathbf{y}_1$, which is the probability distribution over the next symbols. Here, sampling yields $\mathbf{s}_2$ =1. Converting $\mathbf{s}_2$ to $\mathbf{x}_2$, and feeding it to the model leads to updated hidden state $\mathbf{h}_2$ and output $\mathbf{y}_2$, from which can sample again. This iterative symbol-by-symbol procedure can be continued as long as desired. In this example, we stop it after observing an EOL (\n) symbol, and obtain the SMILES for benzene. The hidden state $\mathbf{h}_i$ allows the model to keep track of opened brackets and rings, to ensure that they will be closed again later.

a multinomial distribution, which is estimated using the output vector $\mathbf{y}_t$ of the recurrent neural network at time step t by

$$P_\theta(s_{t+1} = k|s_t, ..., s_1) = \frac{\exp(y_t^k)}{\sum_{k'=1}^{K} \exp(y_t^{k'})} \tag{5}$$

where y_t^k corresponds to the k-th element of vector $\mathbf{y}_t$.[53] Sampling from this distribution would then allow generating novel molecules: After sampling a SMILES symbol s_{t+1} for the next time step $t+1$, we can construct a new input vector $\mathbf{x}_{t+1}$, which is fed into the model, and via $\mathbf{y}_{t+1}$ and Equation 5 yields $P_\theta(s_{t+2}|s_{t+1}, ..., s_1)$. Sampling from the latter generates s_{t+2}, which serves again also as the model's input for the next step (see Figure 4). This symbol-by-symbol sampling procedure is repeated until the desired number of characters has been generated.[53]

To indicate that a molecule is "completed", each molecule in our training data finishes with an “end of line” (EOL) symbol, in our case the single character \n (which means the training data is just a simple SMILES file). Thus, when the system outputs an EOL, a generated molecule is finished. However, we simply continue sampling, thus generating a regular SMILES file that contains one molecule per line.

In this work, we used a network with three stacked LSTM layers, using the keras library.[56] The model was trained with back propagation through time,[53] using the ADAM optimizer at standard settings.[57] To mitigate the problem of exploding gradients during training, a gradient norm clipping of 5 is applied.[53]

2.3 Transfer Learning

For many machine learning tasks, only small datasets are available, which might lead to overfitting with powerful models such as neural networks. In this situation, *transfer learning* can help.[58] Here, a model is first trained on a large dataset for a different task. Then, the model is retrained on the smaller dataset, which is also called *fine-tuning*. The aim of transfer learning is to learn general features on the bigger data set, which also might be useful for the second task in the smaller data regime. To generate focussed molecule libraries, we first train on a large, general set of molecules, then perform fine-tuning on a smaller set of specific molecules, and after that start the sampling procedure.

2.4 Target Prediction

To verify whether the generated molecules are active on the desired targets, standard target prediction was employed. Machine learning-based target prediction aims to learn a classifier $c : M \to \{1, 0\}$ to decide whether a molecule $m \in$ molecular descriptor space M is active or not against a target.[12, 13] The molecules are split into actives and inactives using a threshold on a measure for the substance effectiveness. $p\text{IC}_{50} = -\log_{10}(\text{IC}_{50})$ is one of the most widely used metrics for this purpose. IC_{50} is the *half maximal inhibitory concentration*, that is the concentration of drug that is required to inhibit 50% of a biological target's function *in vitro*.

To predict whether the generated molecules are active towards the biological target of interest, target prediction models (TPMs) were trained for all the tested targets (5-$\text{HT}_{2\text{A}}$, *Plasmodium falciparum* and *Staphylococcus aureus*). We evaluated Random Forest, Logistic Regression, (Deep) Neural Networks and Gradient Boosting Trees (GBT) as models with ECFP4 (Extended Connectivity Fingerprint with a diameter of 4) as the molecular descriptor.[14, 15] We found that GBTs slightly outperformed all other models, and used these as our virtual assay in all studies (see Supporting Information for details). ECFP4 fingerprints were generated with CDK version 1.5.13.[59, 60] Scikit-Learn,[61] xgBoost[62] and keras[56] were used as the machine learning libraries. For 5-HT$_{2A}$ and *Plasmodium*, molecules are considered as active for the TPM if their IC$_{50}$ reported in ChEMBL is < 100 nM, which translates to a pIC$_{50} > 7$, whereas for *Staphylococcus*, we used pMIC > 3.

2.5 Data

The chemical language model was trained on a SMILES file containing 1.4 million molecules from the ChEMBL database, which contains molecules and measured biological activity data. The SMILES strings of the molecules were canonicalized (which means finding a unique representation that is the same for isomorphic molecular graphs)[63, 64] before training with the CDK chemoinformatics library, yielding a SMILES file that contained one molecule per line.[59, 60] It has to be noted that ChEMBL contains many peptides, natural products with complex scaffolds, Michael acceptors, benzoquinones, hydroxylamines, hydrazines etc. which is reflected in the generated structures (see below). This corresponds to 72 million individual characters, with a vocabulary size of 51 unique characters. 51 characters is only a subset of all SMILES symbols, since the molecules in ChEMBL do not contain many of the heavy elements. As we have to set the number of symbols as a hyperparameter during model construction, and the model can only learn the distribution over the symbols present in the training data, this implies that only molecules with these 51 SMILES symbols seen during training can be generated during sampling.

The 5-HT$_{2A}$, the *Plasmodium falciparum* and the *Staphylococcus aureus* dataset were also obtained from ChEMBL. The molecules for the hold-out test sets were removed from the training data.

2.6 Model Evaluation

To evaluate the models for a test set T, and a set of molecules G_N generated from the model by sampling, we report the ratio of reproduced molecules $\frac{|G_N \cap T|}{|T|}$, and enrichment over random (EOR), which is defined as,

$$EOR = \frac{\frac{n}{|G_N|}}{\frac{m}{|R_M|}} \tag{6}$$

where $n = |G_N \cap T|$ is the number of reproduced molecules from T by sampling a set G_N of $|G_N| = N$ molecules from the fine-tuned generative model, and $m = |R_M \cap T|$ is the number of reproduced molecules from T by sampling a set R_M of $|R_M| = M$ molecules from the generic, unbiased generative model trained only on the large dataset. Intuitively, EOR indicates how much better the fine-tuned models work when compared to the general model.

3 Results and Discussion

In this work, we address two points: First, we want to generate large sets of diverse molecules for virtual screening campaigns. Second, we want to generate smaller, focussed libraries enriched with possibly active molecules for a specific target. For the first task, we can train a model on a large, general set of molecules to learn the SMILES grammar. Sampling from this model would generate sets of diverse, but unfocused molecules. To address the second task, and to obtain novel active drug molecules for a target of interest, we perform transfer learning: We select a small set of known actives for that target and we refit our pre-trained chemical language model with this small data-set. After each epoch, we sample from the model to generate novel actives. Furthermore, we investigate if the model actually benefits from transfer learning, by comparing it to a model trained from scratch on the small sets without pre-training.

3.1 Training the recurrent network

We employed a recurrent neural network with three stacked LSTM layers, each with 1024 dimensions, and each one followed by a dropout[65] layer, with a dropout ratio of 0.2, to regularise the neural network. The model was trained until convergence, using a batch size of 128. The RNN was unrolled for 64 steps. It had 21.3×10^6 parameters.

During training, we sampled a few molecules from the model every 1000 mini-batches to inspect progress. Within a few 1000 steps, the model starts to output valid molecules (see Table 1).

3.2 Generating Novel Molecules

To generate novel molecules, 50,000,000 SMILES symbols were sampled from the model symbol-by-symbol. This corresponded to 976,327 lines, from which 97.7% were valid molecules after parsing with the CDK toolkit. Removing all molecules already seen during training yielded 864,880 structures. After filtering out duplicates, we obtained 847,955 novel molecules. A few generated molecules were randomly selected and depicted in Figure 4. The Supporting Information contains more structures. The created structures are not

Table 1 Molecules sampled during training.

Batch	Generated Example	valid
0	`Oc.BK5i%ur+7oAFc7L3T=F8B5e=n)CS6RCTAR((OVCp1CApb)`	no
1000	`OF=CCC2OCCCC)C2)C1CNC2CCCCCCCCCCCCCCCCCCCCCCC`	no
2000	`O=C(N)C(=O)N(c1occc1OC)c2ccccc2OC`	yes
3000	`O=C1C=2N(c3cc(ccc3OC2CCC1)CCCc4cn(c5c(Cl)cccc54)C)C`	yes

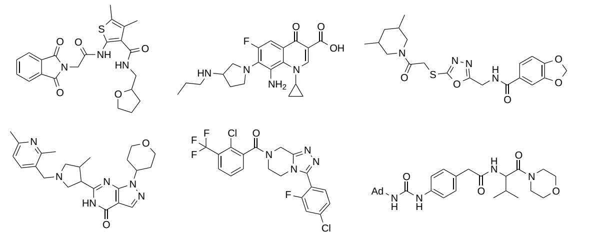

Figure 4 A few randomly selected, generated molecules. Ad = Adamantyl

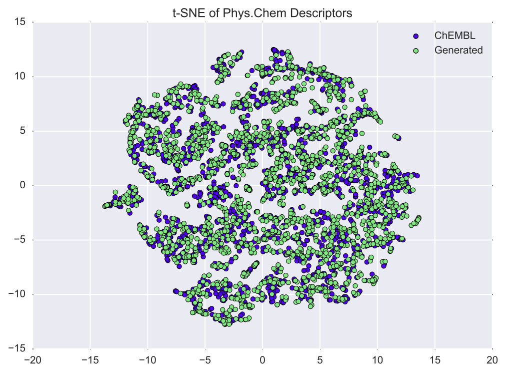

Figure 5 t-SNE projection of 7 physicochemical descriptors of random molecules from ChEMBL (blue) and molecules generated with the neural network trained on ChEMBL (green), to two unitless dimensions. The distributions of both sets overlap significantly.

just formally valid, but are also mostly chemically reasonable.

In order to check if the *de novo* compounds could be considered as valid starting points for a drug discovery program, we applied the internal AstraZeneca filters.[66] At AstraZeneca, this flagging system is used to determine if a compound is suitable to be part of the high-throughput screening collection (if flagged as "core" or "backup") or should be restricted for particular use (flagged as "undesirable" since it contains one or several unwanted substructures, e.g. undesired reactive functional groups). The filters were applied to the generated set of 848 k molecules and they flagged most of them, 640 k (75%), are either core or backup. Since the same ratio (75%) of core and backup compounds has been observed for the ChEMBL collection, we therefore conclude that the algorithm generates preponderantly valid screening molecules and faithfully reproduces the distribution of the training data.

To determine whether the properties of the generated molecules match the properties of the training data from ChEMBL, we followed the procedure of Kolb:[67] We computed several molecular properties, namely molecular weight, BertzCT, the number of H-donors, H-acceptors, and rotatable bonds, logP and total polar surface area for randomly selected subsets from both sets with the RDKit[68] library version 2016.03.1. Then, we performed dimensionality reduction to 2D with t-SNE (t-Distributed Stochastic Neighbor Embedding, a technique analogous to PCA), which is shown in Figure 5.[69] Both sets overlap almost completely, which indicates that the generated molecules very well recreate the properties of the training molecules.

Furthermore, we analysed the Bemis-Murcko scaffolds of the training molecules and the sampled molecules.[70] Bemis-Murcko scaffolds contain the ring systems of a molecule and the moieties that link these ring systems, while removing any side chains. They represent the scaffold, or "core" of a molecule, which series of drug molecules often have in common. The number of common scaffolds in both sets, divided by the union of all scaffolds in both sets (Jaccard index) is 0.12, which indicates that the language model does not just modify side chain substituents, but also introduces modifications at the molecular core.

3.3 Generating Active Drug Molecules and Focused Libraries

3.3.1 Targeting the 5-HT$_{2A}$ receptor

To generate novel ligands for the 5-HT$_{2A}$ receptor, we first selected all molecules with $p\text{IC}_{50} > 7$ which were tested on 5-HT$_{2A}$ from ChEMBL (732 molecules), and then fine-tuned our pre-trained chemical language model on this set. After each epoch, we sampled 100,000 chars, canonicalised the molecules, and removed any sampled molecules that were already contained in the training set. Following this, we evaluated the generated molecules of each round of retraining with our 5-HT$_{2A}$ target prediction model (TPM).

In Figure 4, the ratio of molecules predicted to be active by the TPM after each round of fine-tuning is shown. Before fine-tuning (corresponding to epoch 0), the model generates almost exclusively inactive molecules. Already after 4 epochs of fine-tuning the model produced a set in which 50% of the molecules are predicted to be active.

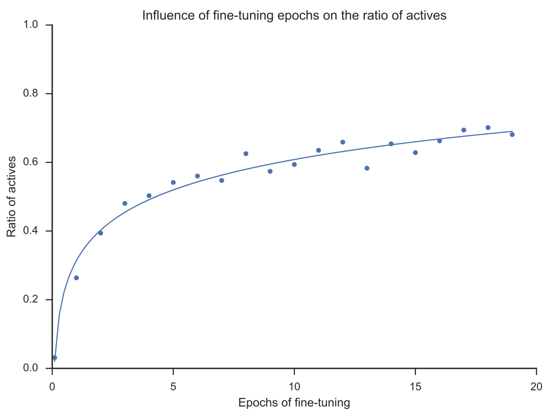

Figure 6 Epochs of fine-tuning vs ratio of actives.

Diversity Analysis In order to assess the novelty of the *de novo* molecules generated with the fine-tuned model, a nearest neighbor similarity/diversity analysis has been conducted using a commonly used 2D fingerprint (ECFP4) based similarity method (Tanimoto index).[67] Figure 7 shows the distribution of the nearest neighbor Tanimoto index generated by comparing all the novel molecules and the training molecules before and after n epochs of fine-tuning. For each bin, the white bars indicate the molecules generated from the unbiased, general model, while the darker bars indicate the molecules after several epochs of fine-tuning. Within the bins corresponding to lower similarity, the number of molecules decreases, while the bins of higher similarity get populated with increasing numbers of molecules. The plot thus shows that the model starts to output more and more similar molecules to the target-specific training set. Notably, after a few rounds of training not only highly similar molecules are produced, but also molecules covering the whole range of similarity, indicating that our method could not only deliver close analogs but new chemotypes or scaffold ideas to a drug discovery project.[5] To have the best of both worlds, that is diverse and focussed molecules, we therefore suggest to sample after each epoch of retraining and not just after the final epoch.

3.3.2 Targeting Plasmodium falciparum (Malaria)

Plasmodium falciparum is a parasite that causes the most dangerous form of Malaria.[71] To probe our model on this important target, we used a more challenging validation strategy. We wanted to investigate whether the model could also propose the same molecules that medicinal chemists chose to evaluate in published studies. To test this, first, the known actives against Plasmodium falciparum with a $p\text{IC}_{50} > 8$ were selected from ChEMBL. Then, this set was split randomly into a training (1239 molecules) and a test set (1240 molecules). The chemical language model was then fine-tuned on the training set. 7500 molecules were sampled after each of the 20 epochs of refitting.

Table 2 Reproducting known actives in the *Plasmodium* test set. EOR: Enrichment over random.

#	$p\text{IC}_{50}$	Train.	Test	Gen. mols.	Reprod.	EOR
1	> 8	1239	1240	128,256	28%	66.9
2	> 8	100	1240	93,721	7%	19.0
3	> 9	100	1022	91,034	11%	35.7

This yielded 128,256 unique molecules. Interestingly, we found that our model was able to "redesign" 28% of the unseen molecules of the test set. In comparison to molecules sampled from the unspecific, untuned model, an Enrichment over Random (EOR) of 66.9 is obtained. With a smaller training set of 100 molecules, the model can still reproduce 7% of the test set, with an EOR of 19.0. To test the reliance on $p\text{IC}_{50}$ we chose to use another cut-off of $p\text{IC}_{50} > 9$, and took 100 molecules in the training set and 1022 in the test set. 11% of the test set could be recreated, with an EOR of 35.7. To visually explore how the model populates chemical space, Figure 8 shows a t-SNE plot of the ECFP4 fingerprints of the test molecules and 2000 generated molecules that were predicted to be active by the target prediction model for *Plasmodium falciparum*. It indicates that the model has generated many similar molecules around the test examples.

3.3.3 Targeting Staphylococcus aureus (Golden Staph)

To evaluate a different target, we furthermore conducted a series of experiments to reproduce known active molecules against *Staphylococcus aureus*. Here, we used actives with a $p\text{MIC} > 3$. MIC is the Mean Inhibitory Concentration, the lowest concentration of a compound that prevents visible growth of a microorganism. As above, the actives were split into a training and a test set. However, here, the availability of the data allows larger test sets to be used. After fine-tuning on the training set of 1000 molecules (Table 3, Entry 1), our model could retrieve 14% of the 6051 test molecules. When scaling down to a smaller training set of 50 molecules (the model gets trained on less than

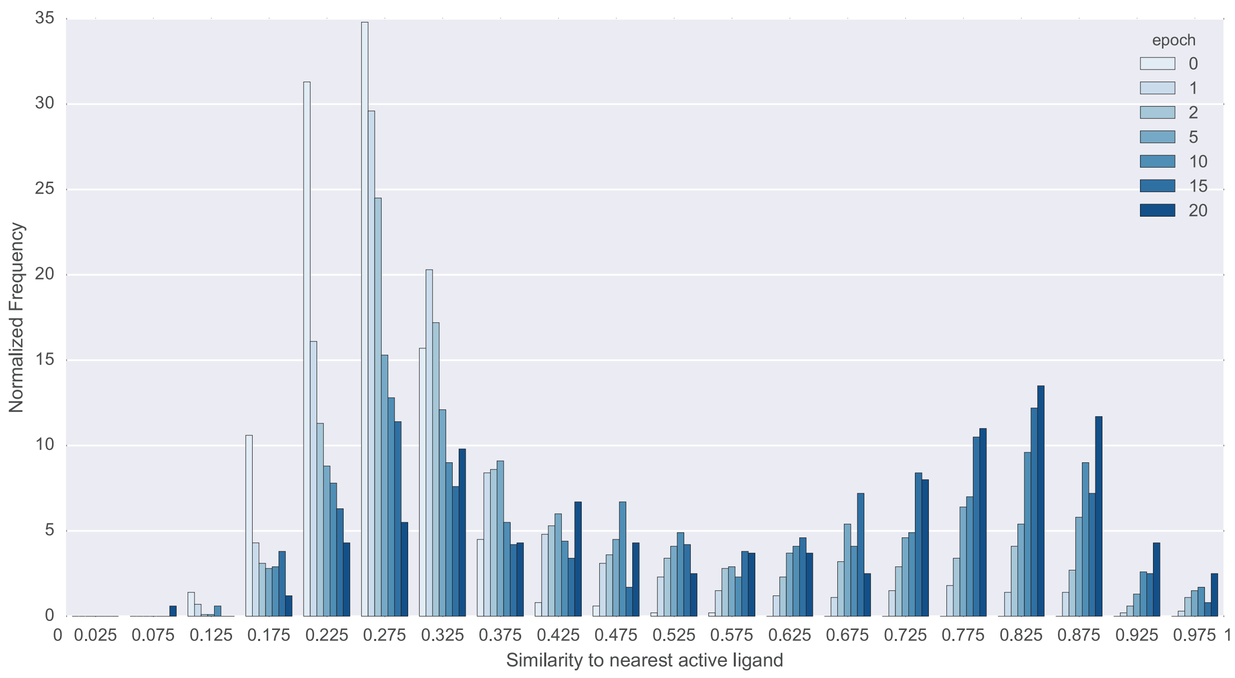

Figure 7 Nearest-neighbour Tanimoto similarity distribution of the generated molecules for 5-HT$_{2A}$ after n epochs of fine-tuning against the known actives. The generated molecules are distributed over the whole similarity range. Generated molecules with a medium similarity can be interesting for scaffold-hopping.[5]

Table 3 Reproducting known actives in the *Staphylococcus* test set. EOR: Enrichment over random.

Entry	pMIC	Train.	Test	Gen. mols.	Reprod.	EOR
1	> 3	1000	6051	51,052	14%	155.9
2	> 3	50	7001	70,891	2.5%	21.6
3[a]	> 3	50	7001	85,755	1.8%	6.3
4[b]	> 3	50	7001	285	0%	—
5[c]	> 3	0	7001	60,988	6%	59.6

[a]Fine-tuning learning rate $= 10^{-4}$. [b]No Pretraining. [c]8 Generate-Test cycles.

1% of the data!), it can still reproduce 2.5% of the test set, and performs 21.6 times better than the unbiased model (Table 3, Entry 2). Using a lower learning rate (0.0001, Entry 3) for fine-tuning, which is often done in transfer learning, does not work as well as the standard learning rate (0.001, Entry 2). We additionally examined whether the model benefits from transfer learning. When trained from scratch, the model performs much worse than the pretrained and subsequently fine-tuned model (see Figure 9 and Table 3, Entry 4). Pretraining on the large dataset is thus crucial to achieve good performance against *Staphylococcus aureus*.

3.4 Simulating Design-Synthesis-Test Cycles

The experiments we conducted so far are applicable if one already knows several actives. However, in drug discovery, one often does not have such a set to start with. Therefore, high throughput screenings are conducted to identify a few hits, which serve as a starting point for the typical cyclical drug discovery process: Molecules get designed, synthesised, and then tested in assays. Then, the best molecules are selected, and based on the gained knowledge new molecules are designed, which closes the cycle. Therefore, as a final challenge for our model, we simulated this cycle by iterating molecule generation ("synthesis"), selection of the best molecules with the machine learning-based target prediction ("virtual assay") and retraining the language model with the best molecules ("design") with *Staphylococcus aureus* as the target. We thus do not use a set of known actives to start the structure generation procedure (see Figure 10).

We started with 100,000 sampled molecules from the unbiased chemical language model. Then, using our target prediction model, we extracted the molecules classified as actives. After that, the RNN was fine-tuned for 5 epochs on the actives, sampling ≈10,000 molecules after each epoch. The resulting molecules

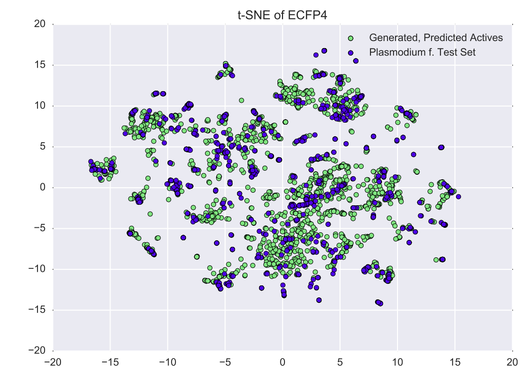

Figure 8 t-SNE plot of the $pIC_{50}>9$ test set (blue) and the *de novo* molecules predicted to be active (green). The language model populates chemical space around the test molecules.

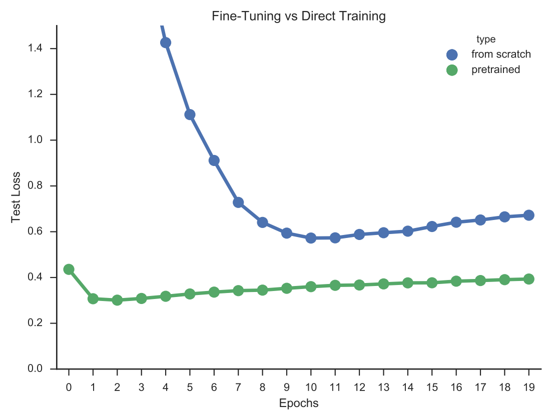

Figure 9 Different training strategies on the *Staphylococcus aureus* dataset with 1000 training and 6051 test examples. Fine-tuning the pretrained model performs better than training from scratch (lower test loss [cross entropy] is better).

were filtered with the target prediction model, and the new actives appended to the actives from the previous round, closing the loop.

Already after 8 iterations, the model reproduced 416 of the 7001 test molecules from the previous task, which is 6% (Table 3, Entry 5), and exhibits and EOR of 59.6. This EOR is higher than if the model is retrained directly on a set of 50 actives (Entry 2). Additionally, we obtained 60,988 unique molecules that the target prediction model classified as active. This demonstrates that in combination with a target prediction or scoring model, our model can also perform the complete *de novo*-design cycle.

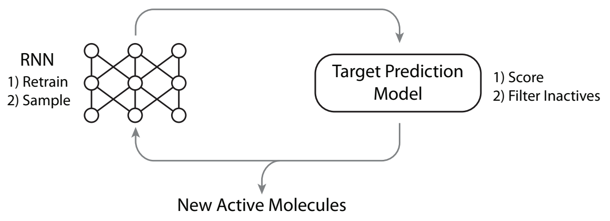

Figure 10 Scheme of our *de novo* design cycle. Molecules are generated by the chemical language model and then scored with the target prediction model (TPM). The inactives are filtered out, and the RNN is retrained. Here, the TPM is a machine learning model, but it could also be a robot conducting synthesis and biological assays, or a docking program.

3.5 Why does the model work?

Our results presented in Section 3.2 show that the general model trained on a large molecule set has learned the SMILES rules and can output valid, drug-like molecules, which resemble the training data. However, sampling from this model does not help much if we want to generate actives for a specific target: We would have to generate very large sets to find actives for that target among the diverse range of molecules the model creates, which is indicated by the high EOR scores in our experiments.

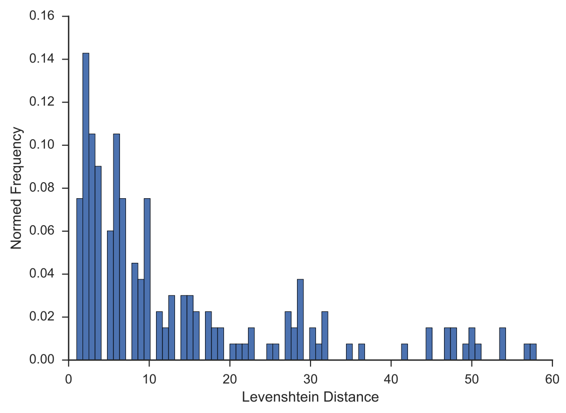

Figure 11 Histogram of Levenshtein (String edit) distances of the SMILES of the reproduced molecules to their nearest neighbour in the training set (Staphylococcus aureus, model retrained on 50 actives). While in many cases the model makes changes of a few symbols in the SMILES, resembling the typical modifications applied when exploring series of compounds, the distribution of the distances indicates that the RNN also performs more complex changes by introducing larger moieties or generating molecules that are structurally different, but isofunctional to the training set.

When fine-tuned to a set of actives, the probability distribution over the molecules captured by our model is shifted towards molecules active towards our target. To study this, we compare the Levenshtein (String edit) distance of the generated SMILES to their nearest neighbours in the training set in Figure 11. The Levenshtein distance of e.g. benzene `c1ccccc1` and pyridine `c1ccncc1` would be 1. Figure 11 shows that while the model often seems to have made small replacements in the underlying SMILES, in many cases it also made more complex modifications or even generated completely different SMILES. This is supported also by the distribution of the nearest neighbour fingerprint similarities of training and rediscovered molecules (ECFP4, Tanimoto, Figure 12). Many rediscovered molecules are in the medium similarity regime.

Because we perform transfer learning, during fine-tuning, the model does not "forget" what it has learned. A plausible explanation why the model works is therefore that it can transfer the modifications that are regularly applied when series of molecules are studied, to the molecules it has seen during fine-tuning.

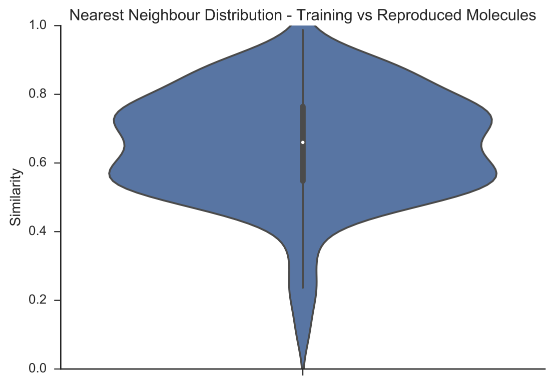

Figure 12 Violin plot of the nearest-neighbour ECFP4-Tanimoto similarity distribution of the 50 training molecules against the rediscovered molecules in Table 3, Entry 2. The distribution suggests the model has learned to make typical small functional group replacements, but can also reproduce molecules which are not too similar to the training data.

4 Conclusion

In this work, we have shown that recurrent neural networks based on the Long Short Term Memory (LSTM) can be applied to learn a statistical chemical language model. The model can generate large sets of novel molecules with similar physico-chemical properties to the training molecules. This can be used to generate libraries for virtual screening. Furthermore, we demonstrated that the model performs transfer learning when fine-tuned to smaller sets of molecules active towards a specific biological target, which enables the creation of novel molecules with the desired activity. By iterating cycles of structure generation with the language model, scoring with a target prediction model (TPM) and retraining of the model with increasingly larger sets of highly scored molecules, we showed that we do not even need a set of active known active molecules to start our procedure with, as the TPM could also be a docking program, or a robot conducting synthesis[72] and biological testing.

We see three main advantages of our method. First, it is conceptually orthogonal to established molecule generation approaches, as it learns a generative model for molecular structures. Second, our method is very simple to setup, train and to use, and can be adapted to different datasets without any modifications to the model architecture, and does not depend on hand-encoded expert knowledge. Furthermore, it merges structure generation and optimisation in one model. A weakness of our model is interpretability. In contrast, existing de-novo design methods settled on virtual reactions to generate molecules, which has advantages as it minimises the chance of obtaining "overfit", weird molecules, and increases the chances to find synthesizable compounds.[2, 7]

To extend our work, it is just a small step to cast molecule generation as a reinforcement learning problem, where the pre-trained LSTM generator could be seen as a policy, which can be encouraged to create better molecules with a reward signal obtained from a target prediction model.[73] In addition, different approaches for target prediction, for example docking, could be evaluated.[7, 11]

Deep Learning is not a panacea, and we join Gawehn et al. in expressing “some healthy skepticism” regarding its application in drug discovery.[26] Generating molecules that are almost right is not enough, because in Chemistry, a miss is as good as a mile, and drug discovery is a “needle in the haystack” problem – in which also the needle looks like hay. Nevertheless, given that we have shown in this work that our model can rediscover those needles, and other recent developments,[26, 74–76] we believe that deep neural networks can be complimentary to established approaches in drug discovery. The complexity of the problem certainly warrants the investigation of novel approaches. Eventually, success in the wet lab will determine if the new wave[21] of neural networks will prevail.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements

The project was conducted during a research stay of M.S. at AstraZeneca R&D Gothenburg. We thank H. Chen and O. Engkvist for valuable discussions and feedback on the manuscript, and G. Klambauer for helpful suggestions.

Author details

[1]Institute of Organic Chemistry & Center for Multiscale Theory and Computation, Westfälische Wilhelms-Universität, Münster, Germany. [2]External Sciences, Discovery Sciences, AstraZeneca R&D Gothenburg, Sweden. [3]Department of Medicinal Chemistry, IMED RIA, AstraZeneca R&D Gothenburg, Sweden. [4]Department of Physics & International Centre for Quantum and Molecular Structures,Shanghai University, China.

References

1. Whitesides, G. M. *Angew. Chem. Int. Ed.* **2015**, *54*, 3196–3209.
2. Schneider, P.; Schneider, G. *J. Med. Chem.* **2016**, *59*, 4077–4086.
3. Reymond, J.-L.; Ruddigkeit, L.; Blum, L.; van Deursen, R. *Wiley Interdisc. Rev. Comp. Mol. Sci.* **2012**, *2*, 717–733.
4. Schneider, G.; Baringhaus, K.-H. *Molecular design: concepts and applications*; John Wiley & Sons, 2008.
5. Stumpfe, D.; Bajorath, J. *Wiley Interdisc. Rev. Comp. Mol. Sci.* **2011**, *1*, 260–282.
6. Schneider, G.; Fechner, U. *Nat. Rev. Drug Disc.* **2005**, *4*, 649–663.
7. Hartenfeller, M.; Schneider, G. *Wiley Interdisc. Rev. Comp. Mol. Sci.* **2011**, *1*, 742–759.
8. Hartenfeller, M.; Zettl, H.; Walter, M.; Rupp, M.; Reisen, F.; Proschak, E.; Weggen, S.; Stark, H.; Schneider, G. *PLoS Comput Biol* **2012**, *8*, e1002380.
9. Hartenfeller, M.; Eberle, M.; Meier, P.; Nieto-Oberhuber, C.; Altmann, K.-H.; Schneider, G.; Jacoby, E.; Renner, S. *J. Chem. Inf. Mod.* **2011**, *51*, 3093–3098.
10. Segler, M.; Waller, M. P. *manuscript submitted* **2016**,
11. Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. *Nat. Rev. Drug Disc.* **2004**, *3*, 935–949.
12. Varnek, A.; Baskin, I. *J. Chem. Inf. Mod.* **2012**, *52*, 1413–1437.
13. Mitchell, J. B. *Wiley Interdisc. Rev. Comp. Mol. Sci.* **2014**, *4*, 468–481.
14. Riniker, S.; Landrum, G. A. *J. Cheminf.* **2013**, *5*, 1.
15. Rogers, D.; Hahn, M. *J. Chem. Inf. Mod.* **2010**, *50*, 742–754.
16. Alvarsson, J.; Eklund, M.; Engkvist, O.; Spjuth, O.; Carlsson, L.; Wikberg, J. E.; Noeske, T. *J. Chem. Inf. Mod.* **2014**, *54*, 2647–2653.
17. Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. Adv. Neural Inf. Proc. Sys. 2015; pp 2224–2232.
18. Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. *arXiv preprint arXiv:1603.00856* **2016**,
19. Zupan, J.; Gasteiger, J. *Analytica Chimica Acta* **1991**, *248*, 1–30.
20. Gasteiger, J.; Zupan, J. *Angew. Chem. Int. Ed.* **1993**, *32*, 503–527.
21. Zupan, J.; Gasteiger, J. *Neural networks in chemistry and drug design*; John Wiley &amp; Sons, Inc., 1999.
22. Lusci, A.; Pollastri, G.; Baldi, P. *J. Chem. Inf. Mod.* **2013**, *53*, 1563–1575.
23. Unterthiner, T.; Mayr, A.; Klambauer, G.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Hochreiter, S. *Adv.* **2014**, *27*.
24. Unterthiner, T.; Mayr, A.; Klambauer, G.; Hochreiter, S. *arXiv preprint arXiv:1503.01445* **2015**,
25. Schneider, P.; Müller, A. T.; Gabernet, G.; Button, A. L.; Posselt, G.; Wessler, S.; Hiss, J. A.; Schneider, G. *Molecular Informatics* **2016**,
26. Gawehn, E.; Hiss, J. A.; Schneider, G. *Molecular Informatics* **2016**, *35*, 3–14.
27. Ramsundar, B.; Kearnes, S.; Riley, P.; Webster, D.; Konerding, D.; Pande, V. *arXiv preprint arXiv:1502.02072* **2015**,
28. Kearnes, S.; Goldman, B.; Pande, V. *arXiv preprint arXiv:1606.08793* **2016**,
29. Behler, J. *Intern. J. Quantum Chem.* **2015**, *115*, 1032–1050.
30. Behler, J.; Parrinello, M. *Phys. Rev. Lett.* **2007**, *98*, 146401.
31. Ma, J.; Sheridan, R. P.; Liaw, A.; Dahl, G. E.; Svetnik, V. *J. Chem. Inf. Mod.* **2015**, *55*, 263–274.
32. Reutlinger, M.; Rodrigues, T.; Schneider, P.; Schneider, G. *Angew. Chem. Int. Ed.* **2014**, *53*, 4244–4248.
33. Miyao, T.; Arakawa, M.; Funatsu, K. *Molecular Informatics* **2010**, *29*, 111–125.
34. Miyao, T.; Kaneko, H.; Funatsu, K. *J. Chem. Inf. Mod.* **2016**, *56*, 286–299.
35. Takeda, S.; Kaneko, H.; Funatsu, K. *J. Chem. Inf. Mod.* **2016**, *56*, 1885–1893.
36. Mishima, K.; Kaneko, H.; Funatsu, K. *Molecular Informatics* **2014**, *33*, 779–789.
37. White, D.; Wilson, R. C. *J. Chem. Inf. Mod.* **2010**, *50*, 1257–1274.
38. Patel, H.; Bodkin, M. J.; Chen, B.; Gillet, V. J. *J. Chem. Inf. Mod.* **2009**, *49*, 1163–1184.
39. Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R.; Bengio, S. *arXiv preprint arXiv:1511.06349* **2015**,
40. Gómez-Bombarelli, R.; Duvenaud, D.; Hernández-Lobato, J. M.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. *arXiv preprint arXiv:1610.02415* **2016**,
41. Voss, C. Modeling Molecules with Recurrent Neural Networks. 2015; `http://csvoss.github.io/projects/2015/10/08/rnns-and-chemistry.html`.
42. Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; Wu, Y. *arXiv preprint arXiv:1602.02410* **2016**,
43. Graves, A.; Eck, D.; Beringer, N.; Schmidhuber, J. Biologically plausible speech recognition with LSTM neural nets. International Workshop on Biologically Inspired Approaches to Advanced Information Technology. 2004; pp 127–136.
44. van den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. *arXiv preprint arXiv:1601.06759* **2016**,
45. Srivastava, N.; Mansimov, E.; Salakhutdinov, R. *CoRR, abs/1502.04681* **2015**, *2*.
46. Gers, F. A.; Schmidhuber, E. *IEEE Transactions on Neural Networks* **2001**, *12*, 1333–1340.
47. Bhoopchand, A.; Rocktäschel, T.; Barr, E.; Riedel, S. *arXiv preprint arXiv:1611.08307* **2016**,
48. Eck, D.; Schmidhuber, J. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on. 2002; pp 747–756.
49. Weininger, D. *J. Chem. Inf. Comp. Sci.* **1988**, *28*, 31–36.
50. Goldberg, Y. *J. Artif. Intell. Res.* **2016**, *57*, 345–420.
51. Hochreiter, S.; Schmidhuber, J. *Neural computation* **1997**, *9*, 1735–1780.
52. Johnson, M.; Schuster, M.; Le, Q. V.; Krikun, M.; Wu, Y.; Chen, Z.; Thorat, N.; Viégas, F.; Wattenberg, M.; Corrado, G. *arXiv preprint arXiv:1611.04558* **2016**,
53. Graves, A. *arXiv preprint arXiv:1308.0850* **2013**,
54. Olah, C. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
55. Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; Schmidhuber, J. *arXiv preprint arXiv:1503.04069* **2015**,
56. Chollet, F. Keras. https://github.com/fchollet/keras, retrieved on 2016-10-24.
57. Kingma, D.; Ba, J. *arXiv preprint arXiv:1412.6980* **2014**,
58. Cireşan, D. C.; Meier, U.; Schmidhuber, J. Transfer learning for Latin and Chinese characters with deep neural networks. The 2012 International Joint Conference on Neural Networks (IJCNN). 2012; pp 1–6.
59. Steinbeck, C.; Hoppe, C.; Kuhn, S.; Floris, M.; Guha, R.; Willighagen, E. L. *Current pharmaceutical design* **2006**, *12*, 2111–2120.
60. Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. *J. Chem. Inf. Comp. Sci.* **2003**, *43*, 493–500.
61. Pedregosa, F. et al. *J. Machine Learning Res.* **2011**, *12*, 2825–2830.
62. Chen, T.; Guestrin, C. *arXiv preprint arXiv:1603.02754* **2016**,
63. Weininger, D.; Weininger, A.; Weininger, J. L. *J. Chem. Inf. Comp. Sci.* **1989**, *29*, 97–101.
64. Checked on 14.12.2016. `https://en.wikipedia.org/wiki/Graph_canonization`.

65. Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. *Journal of Machine Learning Research* **2014**, *15*, 1929–1958.
66. Cumming, J. G.; Davis, A. M.; Muresan, S.; Haeberlein, M.; Chen, H. *Nat. Rev. Drug Disc.* **2013**, *12*, 948–962.
67. Chevillard, F.; Kolb, P. *J. Chem. Inf. Mod.* **2015**, *55*, 1824–1835.
68. RDKit: Open-source cheminformatics; http://www.rdkit.org.
69. Maaten, L. v. d.; Hinton, G. *J. Machine Learning Res.* **2008**, *9*, 2579–2605.
70. Bemis, G. W.; Murcko, M. A. *J. Med. Chem.* **1996**, *39*, 2887–2893.
71. Williamson, A. E.; Todd, M. H. *ACS central science* **2016**, *2*, 687–701.
72. Ley, S. V.; Fitzpatrick, D. E.; Ingham, R.; Myers, R. M. *Angew. Chem. Int. Ed.* **2015**, *54*, 3449–3464.
73. Sutton, R. S.; Barto, A. G. *Reinforcement learning: An introduction*; MIT press Cambridge, 1998; Vol. 1.
74. Schmidhuber, J. *Neural Networks* **2015**, *61*, 85–117.
75. Altae-Tran, H.; Ramsundar, B.; Pappu, A. S.; Pande, V. *arXiv preprint arXiv:1611.03199* **2016**,
76. Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Colmenarejo, S. G.; Grefenstette, E.; et al., *Nature* **2016**, *538*, 471–476.

Additional Files

Generated Molecules

In the following, a few randomly selected molecules produced with the general model trained on ChEMBL are shown.