Drug Design with Recurrent Neural Networks and Molecular Embeddings

Christopher Glasz
christopher.glasz@mail.mcgill.ca
260720944

Theophile Gervet
theophile.gervet@mail.mcgill.ca
260612917

***Abstract*—Finding molecules with desirable properties is critical for many fields related to chemical engineering. However, this task is not at all trivial, as optimization in molecular space is difficult. In this work, we explore two approaches introduced in previous literature, involving the use of recurrent neural networks and continuous latent representations for molecules. We find both methods to be promising for generalized molecular generation and drug discovery, but suspect that incomplete representation of the space in the training data impairs their overall effectiveness.**

I. Introduction and Related Works

Automatic chemical and drug design refers to the use of computational strategies to generate novel molecules with some desirable properties. Although this idea is extremely compelling (imagine a model that automatically generates molecules that are active against Malaria, for example), it is a very complex problem.

First, the chemical search space is extremely large; about 10^8 molecules have been synthesized, yet it is estimated that up to 10^{60} undiscovered potential drug-like molecules could exist [17]. Add to this the fact that the search space is discrete and poorly structured (making gradient descent impossible), and the problem seems nearly intractable.

Until very recently, most approaches have revolved around discrete local search [18], genetic algorithms with hand-crafted mutation and cross-over rules [12], or hill-climbing with heuristics based on similarity metrics and rules defined by experts. Although these techniques have led to the discovery of useful new molecules, the requirement for extensive domain knowledge and the complexity of the discrete optimization task remain severe limitations. Relying on a small set of robust rules designed by humans has the fundamental shortcoming of restricting the exploration of chemical space to a small subset around that which is already known. Although there has been some work attempting to learn these rules directly from data - thereby suppressing the need for domain knowledge - the complexity of the discrete optimization task remains.

We consider two recent approaches to chemical design which circumvent these difficulties. These methods are completely data-driven and avoid optimization in the discrete molecular space.

The first approach we consider is a character-level recurrent language model based on work by Segler et al. [21]. We pre-train a deep recurrent neural network on a large set of molecules encoded as SMILES strings (see Section II), before fine-tuning the model with a small subset containing molecules of particular interest. This model can then be used to generate novel molecules with the same properties.

The second approach considered is a sequential variational auto-encoder based on work by Gómez-Bombarelli et al. [9]. We train a sequential VAE on SMILES strings to learn a continuous latent space. The continuity of the latent space allows us to interpolate between molecules and perform gradient ascent to generate new molecules with desired properties. The variational objective is important to encourage every point in the latent space to decode to valid SMILES, as we will see.

II. Data Representation

Usually compounds are represented as undirected molecular graphs called Lewis structures where atoms are labeled nodes and the bonds between atoms are edges. For computational purposes, these graphs are generally converted to vector representations. These derived vector representations are either hand-crafted features (chemical fingerprints [19]) or learned representations using (for example) convolutional neural networks on graphs [8].

However, as of now converting these vector representations back to graphs is an open problem. This means they cannot be used for our autoencoder architecture, because we must be able to decode the continuous latent space to molecules. Moreover, recent advances in recurrent language models make it very convenient to work with text. For these reasons both the models we consider use molecules encoded as SMILES.

SMILES is a formal language which encodes molecular graphs compactly as human-readable strings. It consists of an alphabet of characters. Atoms are represented by the standard abbreviation for their chemical element (`C` for carbon, `O` for oxygen, etc.). `-`, `=`, and `#` denote single, double, and triple bonds, respectively. To indicate rings, a number is introduced at the two atoms where the ring is closed, while side chains are denoted by parentheticals. It is clear that a generative model for SMILES requires some form of memory; long-term time dependencies are required to properly handle large cycles and long side chains.

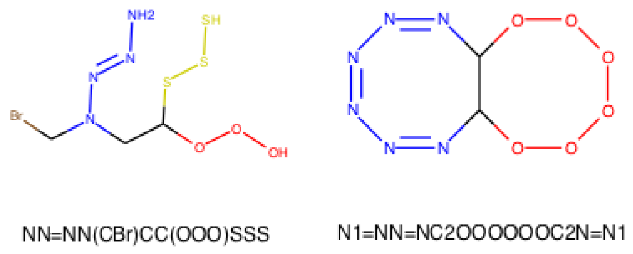


Figure 1: Examples of side chains and cycles in SMILES strings.

III. Data Preprocessing

A. Character-Level Recurrent Language Model

To train the recurrent neural network model, we used the chemical representations from release 22.1 of ChEMBL [2], a chemistry database from the European Bioinformatics Institute. This consists of 1.68 million molecules represented as SMILES strings. We used RDKit [16], an open-source cheminformatics software, to canonicalize (it's possible to represent a molecule with several different SMILES strings) and filter out poorly-formed SMILES (unclosed cycles, incorrect valences, etc.). For fine-tuning, we again used RDKit to extract lead-like molecules from the full collection. To do this, we used what's called the Rule of 3 [7], which is based on Lipinski's original Rule of 5 [15] for evaluating drug-likeness of molecules.

The Rule of 3 describes compounds that are considered to be "lead-like", or highly promising candidates for drug discovery. It is similar to the Rule of 5, but favors lower molecular weight and lipophility, as these measures are often increased during drug discovery in order to improve the selectivity of a drug candidate. A RO3-compliant compound meets the following qualifications:

- No more than 3 hydrogen bond donors
- No more than 3 hydrogen bond acceptors
- Molecular mass less than 300 daltons
- Octanol-water partition coefficient (this describes how hydrophobic a molecule is) no greater than 3.
- 3 or fewer rotatable bonds.

This criteria reduced the size of our dataset by 97%, resulting in just under fifty thousand molecules for fine-tuning.

To ready the data for training, we joined all SMILES strings by newlines, and used a generator to provide 64-character sequences, sampled uniformly from the full sequence. After this step, the effective size of the full dataset was 85.87 million samples (1.25 million for the lead-like subset). Each character in a subsequence was converted to a one-hot vector encoding, resulting in a 64×48 matrix, to be fed to the RNN.

B. Molecular autoencoder

We used the same subset of 250,000 drug-like commercially available molecules extracted from the ZINC database [10] as used in the work by Gómez-Bombarelli et al. It would have been interesting to try their approach on another dataset, but given the amount of time required to train the model, we decided not to take the risk. We filtered out SMILES strings longer than 120 characters and padded the remaining strings with whitespace at the end. We then extracted a vocabulary consisting of 35 unique characters and converted the strings to a one-hot representation suitable for the autoencoder.

IV. Methods

A. Character-Level Recurrent Language Model

We attempted to replicate exactly the model described by Segler et al. using Keras [4]. We used a recurrent neural network architecture, with three stacked LSTM layers, with 1,024 neurons each. Each was followed by a dropout layer with a dropout rate of 0.2, and the network was completed with a 48-dimensional fully-connected softmax layer, which encodes the probability of each character in a SMILES string. All told, the model had roughly 21.23 million trainable parameters. This is slightly fewer than that of the authors', which is due to the fact that their one-hot encodings were 51-dimensional, compared to our 48. We suspect that the process of filtering out poorly formed molecules in the training data is what led to this disparity (as some characters, like `X`, appear less than 10 times in the entire set of 85 million).

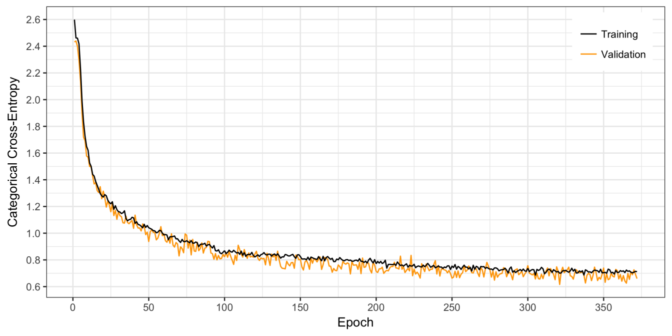


Figure 2: The character-level categorical cross-entropy loss of the RNN over the course of training on the full dataset.

We first trained this model until convergence with a batch size of 128 on 80% of the full training data. As per Segler et al, we used a training rate of 0.001, and gradient clipping at 5.0. If no improvement on the validation loss was seen after 20 epochs, we reduced the learning rate by half. We considered the model to be converged after 50 epochs with no improvement. We arbitrarily considered one epoch of training to consist of 8192 samples - we did this to more precisely determine convergence, as a single cycle through full training set would take several weeks on our hardware. Convergence was reached after 372 of these epochs - the equivalent of just 3.5% of the full dataset. This suggests either that the model is insufficiently complex, or that we simply have more data than is required to adequately represent the space. The former is likely to be the case, but we will assume the latter for the purposes of this work, as we have constrained the model to match that of Segler et al.

After convergence, the model could predict the next character of a sequence in the testing data with roughly 78% accuracy - not an impressive feat, considering nearly half of all characters in the data are either `C` or `c`, denoting carbon. Indeed, if molecules are generated deterministically by always taking the argmax of the output layer, the model produces infinitely long carbon chains. This generation protocol always results in chemical structures that make sense, but which are not at all useful; if the input sequence contains an unmatched opening parenthesis, the model will terminate the sidechain as quickly as possible, and then continue generating `C` forever.

We solved this problem with a tunable temperature parameter for generation, which divides the log probabilities before

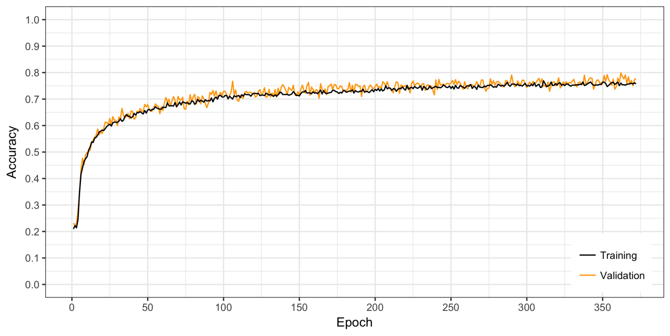


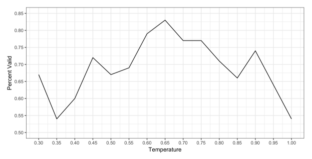
Figure 3: The character accuracy of the RNN over the course of training on the full dataset.

sampling from the distribution of characters. In other words, if we denote the input and output of the model as $\mathbf{X}$ and $\mathbf{z}$ respectively, then we sample from the probability distribution defined by

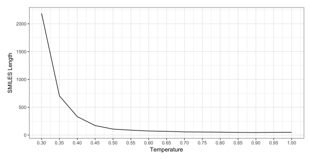
$$Pr(\ y = j \mid \mathbf{X}\ ) = \frac{e^{(\log z_j)/t}}{\sum_k e^{(\log z_k)/t}}$$

Where t is our temperature parameter. This has the effect of exaggerating the probabilities for low values of t. As t approaches 0, the model generates characters nearly deterministically, as before. When $t = 1$, characters are sampled with probability equal to the certainty of the model itself.

Empirically, we saw the best results when using a temperature between 0.6 and 0.7. A deterministic generator produces infinitely long carbon chains, so low temperatures fare poorly. On the other hand, temperatures close to 1 often produce invalid SMILES.



(a) The rate of valid SMILES strings as a function of the temperature parameter



(b) The average length of SMILES strings as a function of the temperature parameter

After training on the full dataset, we fine-tuned on the smaller set, containing only lead-like molecules, using the same hyperparameters. Here, convergence was reached after only 55 epochs.

B. Molecular autoencoder

In the original molecular space, chemical design is a very complex discrete optimization problem with a huge set of constraints. The objective here is to circumvent this difficult optimization problem by learning a continuous representation of molecules (see Figure 4). We can then train a second model that maps from the continuous space to some given property measurement and perform unconstrained optimization with gradient ascent towards regions that maximize the property.

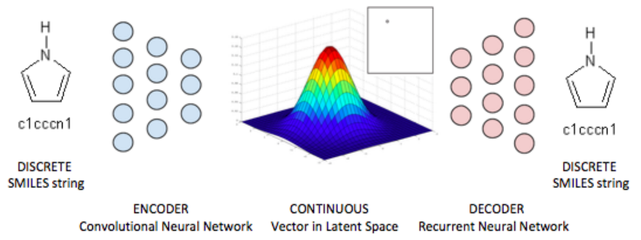

c1ccn1
DISCRETE SMILES string
ENCODER Convolutional Neural Network
CONTINUOUS Vector in Latent Space
DECODER Recurrent Neural Network
c1ccn1
DISCRETE SMILES string

Figure 4: Molecular autoencoder model : the encoder projects the discrete SMILES string in a continuous latent space, and the decoder maps this continuous representation back to a discrete SMILES string.

Note that it is important to encourage every point in the latent space to decode to a valid SMILES string in order to get back a molecule. This is where the variational objective comes into play. Other types of autoencoders have been tried by Gómez-Bombarelli et al., but they led to whole portions of the latent space decoding to invalid SMILES.

The variational autoencoder [11, 13] is based on a regularized version of the standard autoencoder. It imposes a prior distribution $p(z)$ on the latent space z which enforces all areas of the latent space to correspond to valid SMILES strings. The VAE uses a stochastic encoder, usually a neural network conditioned on x which outputs the parameters of the posterior $q(z|x)$ and a decoder $p(x|z)$, stochastic at training time and deterministic at testing time, which attempts to reconstruct the data.

Intuitively, adding noise to the encoded molecules forces the decoder to learn how to decode a wide variety of latent points rather than memorizing the training data as isolated latent points. The VAE objective we maximize takes the following form, where we make the dependence on ϕ and θ, the parameters of the encoder and decoder networks respectively, explicit:

$$\begin{aligned} L(x;\theta;\phi) &= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)\ ||\ p(z)) \\ &\leq \log p(x) \end{aligned}$$

The first term is the usual reconstruction error, while the second term is the KL divergence between the posterior $q(z|x)$ and the prior $p(x)$. It can be interpreted as a regularization term, which encourages the model to keep the posterior close to the prior. To make things even more mathematically pleasing, it can be easily shown that this objective is a valid lower bound on the true log likelihood of the data, making the VAE a generative model.

What makes this framework so powerful and popular is its flexibility. We can parametrize the prior $p(z)$ and the posterior $q(z|x)$ however we want, and our objective remains a valid lower bound on the true log likelihood of the data. The encoder and decoder networks can also take arbitrary forms; we have complete freedom on the choice of ϕ and θ parameters.

In order to adapt variational autoencoders to sequence modeling, two approaches have been pursued in the literature. We can have a set of latent variables and a VAE per timestep, which makes use of information from the previous timestep through a recurrent layer (Chung et al. [5]). Alternatively, we can have global latent variables which encode the whole sequence with a single (usually recurrent) encoder (Bowman et al. [3]).

We implemented the first approach before realizing it is not adapted to our needs; in order to interpolate in the latent space or perform gradient ascent, we need a global latent representation for molecules as opposed to one latent representation for each atom or bond of the molecule. We mention this mistake in our report because the first kind of sequential autoencoder happens to be much trickier to implement. We couldn't use layers from the Keras [4] or Tensorflow [1] packages and had to enroll the recurrence ourselves in Tensorflow. We provide a functional well documented implementation with the report (sequentialVAE.py) even though we didn't use it for further experiments.

All the experiments in the results section were realized with an implementation of the second approach. We used diagonal Gaussians with 292 latent dimensions for $p(z)$ and $q_\theta(z|x)$, using the reparameterization trick from Kingma et al. [13]. We have found that a deep convolutional encoder coupled with a recurrent decoder works best, in accordance with results by Gómez-Bombarelli et al. The one-dimensional convolutional layers take advantage of the presence of repetitive substrings, invariant to translations, which correspond to chemical substructures like cycles and functional groups.

The final encoder network used three 1D convolutional layers with 9, 9 and 10 filters respectively and convolution kernels of size 9, 9 and 11 respectively, followed by two fully-connected layers of width 435 and 292. The decoder started with a fully-connected layer of width 292, followed by three layers of gated recurrent units (GRU [6]) with hidden dimension 501. The last layer of the RNN uses a softmax output to define a probability distribution over all possible characters at each position of the SMILES string. While training, we sample from this distribution, while at test time we deterministically pick the most probable character.

The model was trained with stochastic gradient descent using the Adam optimizer [14] and a minibatch size of 300. The reconstruction error term of the loss was estimated by averaging over the minibatch and time steps the categorical cross-entropy between the softmax output and the true character. The KL divergence term of the loss was computed in closed form as in Kingma et al. We used a very small learning rate of 10^{-3}, decayed in case of a plateau down to a minimum of 10^{-6}.

Our model is very close to the one used by Gómez-Bombarelli et al. with two slight differences. First, we use rectified linear units in place of tanh nonlinearities in the encoder, which seems to greatly accelerate the training process. Secondly, and more importantly, while the authors use teacher forcing to train their model (in the last GRU layer, the softmax output of the previous neuron is fed as input to the current one) we do not.

This choice was motivated by an observation by Bowman et al. that the model has a tendency to initially ignore the data, set the posterior $p(z|x)$ to the prior $p(z)$ to get the KL divergence term to zero, and explain the data with the more easily optimized decoder. From there, the decoder ignores the encoder which makes the latter very hard to optimize with little to no gradient flowing back. To counter this issue, they propose to weaken the decoder. This is what we do by removing the conditioning information that is the softmax output of the previous neuron. This forces the model to rely on the latent representation z to make good predictions.

We first attempted to train the model on our personal computers, but each epoch took about 8 hours. After a few days of computer melting, we realized it wouldn't be a viable option. We ended up training the model with Amazon Web Services on a g2.2xlarge GPU. This took 40 epochs to converge for a total of 12 hours training time (as opposed to two weeks if we had stayed on our laptops).

V. Experimental Results

A. Character-level recurrent language model

After training on the full dataset, 75.9% of 1,000 SMILES strings generated represented valid molecules. This is significantly lower than that claimed by Segler et al., whose model generated SMILES strings which were valid in 97.7% of cases. This may be due to a disparity in computational power; the authors did not state how long they trained their model on the complete set, nor did they specify the convergence criteria they used. This may also have been caused by a difference in generation procedure - there is no indication of

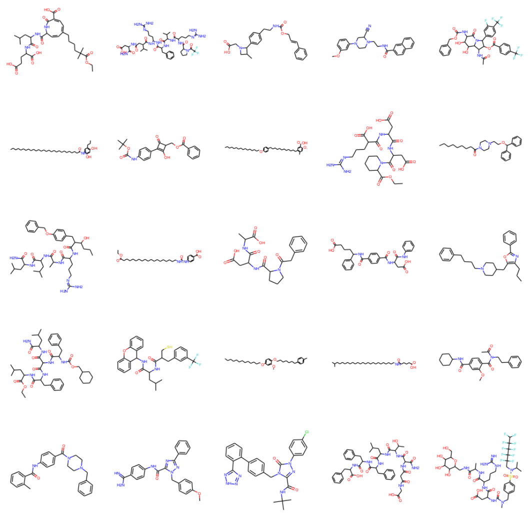


Figure 5: 16 molecules generated by the model. These molecules are very large, and not at all suitable for pharmaceuticals. Observe that the phenomenon of long carbon chains has not been completely shaken off.

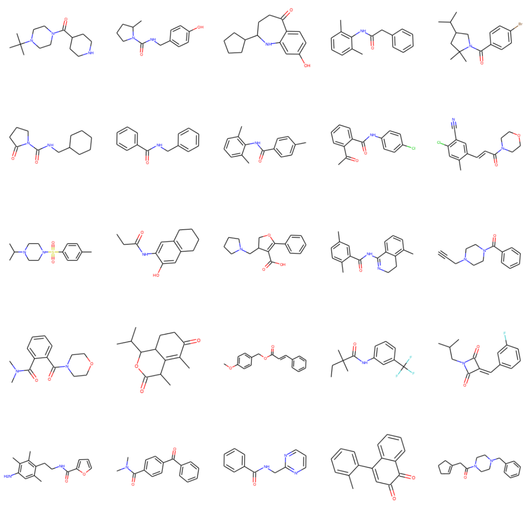


Figure 6: 16 molecules generated by the model after fine-tuning. All of these molecules are RO5-compliant, making them likely drug candidates.

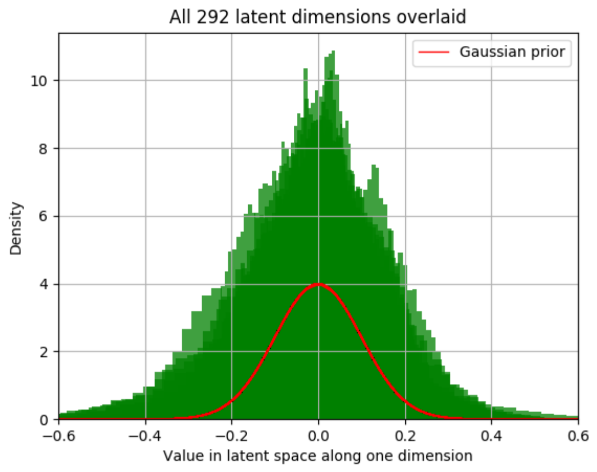


Figure 7: All 292 latent dimensions plotted as histograms over the Gaussian prior with mean $\mu = 0$ and standard deviation $\sigma = 0.1$ for each dimension.

whether characters were sampled from the raw distribution, or if they performed some process similar to our temperature-controlled sampling, or even if characters were determined deterministically (this is unlikely, however, as no mention is made of infinitely long carbon chains early in training).

Of the SMILES representing valid molecules, 53.2% were drug-like according to Lipinski's Rule of five. This is not representative of the training molecules, of which 70.6% were drug-like.

Of the 759 molecules encoded by the valid SMILES, only one was a repeat, and 2.4% were found in the training data. This confirms that the model didn't just memorize the data (though with only 78% character-level accuracy, this wasn't a real danger).

After fine-tuning on the lead-like set, there was a tremendous jump in performance. 93.5% of generated SMILES were valid, and of these, 99.25% were drug-like. 7.1% of the molecules were present in the training data, which suggests slightly more overfitting, but another 4.1% were present in the set of molecules not fine-tuned on, suggesting the model has simply improved at generating realistic molecules. 68.5% of the molecules were lead-like in addition to being drug-like, making them very promising candidates for drug discovery.

The huge increase in valid molecules can be partially explained by the fact that lead-like molecules have a restriction on their mass. This results in smaller molecules, which have shorter SMILES representations. As a result, the model is less likely to neglect to end a side-chain or close a cycle, as these structures are much smaller, requiring a shorter memory in the model.

B. Molecular autoencoder

After convergence of the molecular autoencoder, we obtained 99.23% individual character reconstruction accuracy, which translates to around 70% molecule reconstruction accuracy. On a side note, Gómez-Bombarelli et al. report 95% character accuracy and 70% molecule accuracy. This is surprising; assuming molecules are composed of 50 characters on average, some quick math shows $(0.9923)^{50} \approx 0.68$ which is consistent with our results, while $(0.95)^{50} \approx 0.08$ is much lower than the authors' reported 70%. The latent dimensions fit the Gaussian prior fairly well, as evident in Figure 7.

The VAE objective helps to populate the latent space, and we are able to interpolate between molecules, as illustrated on Figure 8. We randomly selected two molecules from our dataset, projected them in the latent space, and took steps from one to the other, decoding intermediate latent vectors to SMILES strings. None of these intermediate molecules were present in the training set.

We also performed random sampling close to a given lead-like molecule hoping that nearby points in the latent space would also decode to lead-like molecules. The results were encouraging: of the unique SMILES generated by this process, around 70% corresponded to valid molecules and 20% of these were lead-like, a much higher percentage than present in the training data. These molecules can be seen in Figure 9.

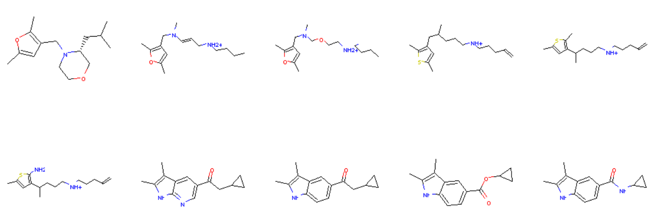


Figure 8: Interpolating between two molecules in the latent space (from top left to bottom right).

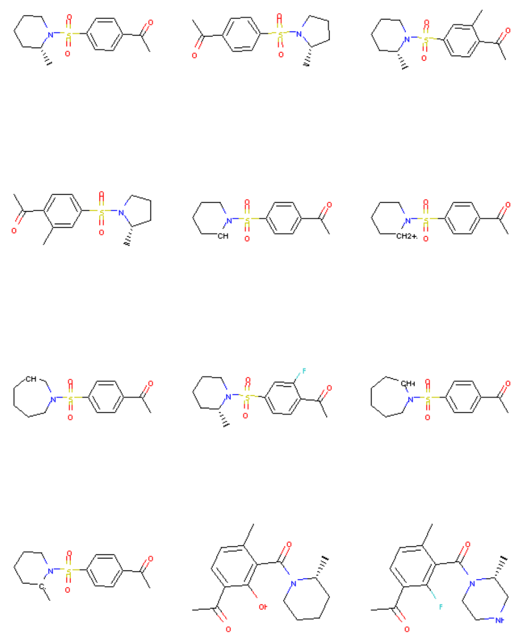


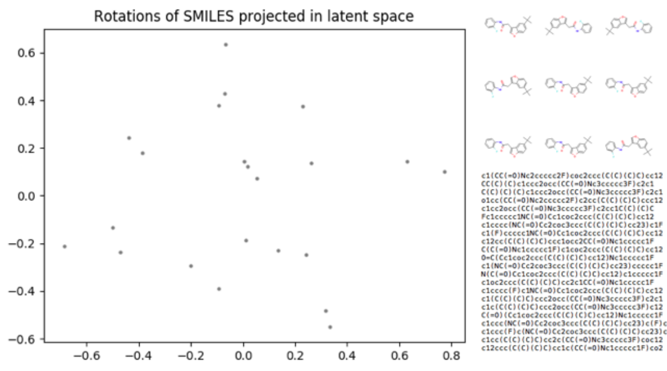
Figure 9: Random sampling near a lead-like molecule. We are sampling near the top left molecule, all other molecules are lead-like molecules never seen in the training set.

In future work, we would like to train a regression model which would take the latent representation as input, and output the value of some molecular property of interest (such as those required for RO3-compliance). We could then perform gradient ascent with respect to the input to find regions of the latent space which optimize the property of interest.

VI. Discussion

Both approaches were able to generate novel lead-like molecules which would make promising candidates for drug discovery. Direct generation with the LSTM architecture is more easily applicable to the generation of molecules close to a specific target, as with very little fine-tuning and no explicit optimization, we get very good results. On the other hand, the simplicity of the approach could also be a weakness. We hypothesize explicit optimization from the continuous latent space should outperform direct generation when it comes to the quality of the molecules generated.

The representation of molecules as SMILES strings is a major shortcoming of the methodologies presented in this work. Having to learn the formal syntax of the language makes the problem unnecessarily difficult, requiring significantly more data than would be needed with a better representation. Further, the non-uniqueness of this representation indicates that it may not be suited to the problem - a single molecule can be represented by different SMILES strings depending on at which atom the representation is rooted. For the VAE approach, we would like these strings to be mapped to a small neighborhood in the latent space, but this is not the case empirically. We generated 22 possible representations for a single molecule, projected them into the 292-dimensional latent space, and used principal component analysis to project again down to 2 dimensions. The result is illustrated in Figure 10.



```
c1(CC(=O)Nc2ccccc2F)coc2ccc(C(C)(C)C)cc12
CC(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1
C(C)(C)(C)c1ccc2occ(CC(=O)Nc3ccccc3F)c2c1
o1cc(CC(=O)Nc2ccccc2F)c2cc(C(C)(C)C)ccc12
c1cc2occ(CC(=O)Nc3ccccc3F)c2cc1C(C)(C)C
Fc1ccccc1NC(=O)Cc1coc2ccc(C(C)(C)C)cc12
c1cccc(NC(=O)Cc2coc3ccc(C(C)(C)C)cc23)c1F
c1(F)ccccc1NC(=O)Cc1coc2ccc(C(C)(C)C)cc12
c12cc(C(C)(C)C)ccc1occ2CC(=O)Nc1ccccc1F
C(C(=O)Nc1ccccc1F)c1coc2ccc(C(C)(C)C)cc12
O=C(Cc1coc2ccc(C(C)(C)C)cc12)Nc1ccccc1F
c1(NC(=O)Cc2coc3ccc(C(C)(C)C)cc23)ccccc1F
N(C(=O)Cc1coc2ccc(C(C)(C)C)cc12)c1ccccc1F
c1oc2ccc(C(C)(C)C)cc2c1CC(=O)Nc1ccccc1F
c1cccc(F)c1NC(=O)Cc1coc2ccc(C(C)(C)C)cc12
c1(C(C)(C)C)ccc2occ(CC(=O)Nc3ccccc3F)c2c1
c1c(C(C)(C)C)ccc2occ(CC(=O)Nc3ccccc3F)c12
C(=O)(Cc1coc2ccc(C(C)(C)C)cc12)Nc1ccccc1F
c1ccc(NC(=O)Cc2coc3ccc(C(C)(C)C)cc23)c(F)c1
c1ccc(F)c(NC(=O)Cc2coc3ccc(C(C)(C)C)cc23)c1
c1cc(C(C)(C)C)cc2c(CC(=O)Nc3ccccc3F)coc12
c12ccc(C(C)(C)C)cc1c(CC(=O)Nc1ccccc1F)co2
```

Figure 10: Left : 22 SMILES representations of a molecule projected along two most important directions of variation in the latent space. Lower right : all 22 SMILES variations. Upper right : the first 9 variations as molecular graphs.

We can see that the variations are not mapped to any discernible neighborhood at all. One might expect that this is due to the principal component analysis for visualization, but it is not to blame; the variations are not mapped close together along any of the 292 dimensions. This suggests that our autoencoder is truly representing SMILES strings, rather than the molecules they encode.

An important question to ask at this stage is the following: are the molecules represented by the generated SMILES synthetically accessible? The short answer is we don't know. We decided to focus on lead-like molecules because the simplicity of their structure makes them promising candidates for synthesis, but there are no guarantees. A whole field a cheminformatics is dedicated to synthetic planning, a particularly interesting approach (AlphaChem [20]) formulates the problem as a two-player game and tackles it with reinforcement learning techniques inspired by (predictably) the recent AlphaGo success [22]. These techniques could be coupled with our generation systems to create reliable models that discover molecules which not only possess desirable chemical properties, but can be cheaply synthesized. This, however, is left for future work.

VII. Statement of Contributions

A. *Chris Glasz*

Responsible for the implementation of the LSTM model for direct generation, for extracting molecules of interest (drug-like and lead-like) from the data, and for the corresponding components of the report and presentation.

B. *Theophile Gervet*

Responsible for the implementation of the VAE model for latent space based generation, and for the corresponding components of the report and presentation.

References

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] A. Patrícia Bento et al. "The ChEMBL bioactivity database: an update". In: *Nucleic Acids Research* 42.D1 (2014), pp. D1083–D1090. DOI: 10.1093/nar/gkt1031. URL: http://dx.doi.org/10.1093/nar/gkt1031.

[3] Samuel R. Bowman et al. "Generating Sentences from a Continuous Space". In: *CoRR* abs/1511.06349 (2015). URL: http://arxiv.org/abs/1511.06349.

[4] François Chollet. *Keras*. https://github.com/fchollet/keras. 2015.

[5] Junyoung Chung et al. "A Recurrent Latent Variable Model for Sequential Data". In: *CoRR* abs/1506.02216 (2015). URL: http://arxiv.org/abs/1506.02216.

[6] Junyoung Chung et al. "Gated Feedback Recurrent Neural Networks." In: *ICML*. 2015, pp. 2067–2075.

[7] Miles Congreve et al. "A ?rule of three?for fragment-based lead discovery?" In: *Drug discovery today* 8.19 (2003), pp. 876–877.

[8] David K Duvenaud et al. "Convolutional networks on graphs for learning molecular fingerprints". In: *Advances in neural information processing systems*. 2015, pp. 2224–2232.

[9] Rafael Gómez-Bombarelli et al. "Automatic chemical design using a data-driven continuous representation of molecules". In: *CoRR* abs/1610.02415 (2016). URL: http://arxiv.org/abs/1610.02415.

[10] John J Irwin and Brian K Shoichet. "ZINC–a free database of commercially available compounds for virtual screening". In: *Journal of chemical information and modeling* 45.1 (2005), p. 177.

[11] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *ArXiv e-prints* (Jan. 2014). arXiv: 1401.4082 `[stat.ML]`.

[12] Ilana Y Kanal et al. "Efficient computational screening of organic polymer photovoltaics". In: *The journal of physical chemistry letters* 4.10 (2013), pp. 1613–1623.

[13] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 `[stat.ML]`.

[14] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: http://arxiv.org/abs/1412.6980.

[15] Christopher A Lipinski et al. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings". In: *Advanced drug delivery reviews* 23.1-3 (1997), pp. 3–25.

[16] "RDKit, Open-Source Cheminformatics". URL: http://www.rdkit.org.

[17] Jean-Louis Reymond and Mahendra Awale. "Exploring chemical space for drug discovery using the chemical universe database". In: *ACS chemical neuroscience* 3.9 (2012), pp. 649–657.

[18] Jean-Louis Reymond et al. "Chemical space as a source for new drugs". In: *MedChemComm* 1.1 (2010), pp. 30–38.

[19] David Rogers and Mathew Hahn. "Extended-connectivity fingerprints". In: *Journal of chemical information and modeling* 50.5 (2010), pp. 742–754.

[20] Marwin H. S. Segler, Mike Preuß, and Mark P. Waller. "Towards "AlphaChem": Chemical Synthesis Planning with Tree Search and Deep Neural Network Policies". In: *CoRR* abs/1702.00020 (2017). URL: http://arxiv.org/abs/1702.00020.

[21] Marwin H. S. Segler et al. "Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks". In: *CoRR* abs/1701.01329 (2017). URL: http://arxiv.org/abs/1701.01329.

[22] David Silver et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (2016), pp. 484–489. ISSN: 0028-0836. DOI: 10.1038/nature16961.