Characterizing Chemical VAE Molecular Generation Validity for Drug Discovery

Andrew Jacobson
jonaj2@illinois.edu

Dixon Liang
dixonl2@illinois.edu

John Judge
jmjudge2@illinois.edu

Megan Masanz
mjneuman@illinois.edu

ABSTRACT

Group Presentation Link: https://youtu.be/zZHGXTrgfrY

We characterize and seek to enhance the creation of valid molecules for drug-discovery by building upon the established work in the open-source community. Existing work uses variational auto encoders that encode Simplified Molecular Input Line Entry System (SMILES) strings representing molecules into vectors in a Gaussian-regularized latent space, and train a decoder to reconstruct or generate SMILES strings. Our primary contribution will be to quantitatively characterize the trade offs between different architectures and the effects of tuning different parameters. We investigate a total of four different model variations. We compare a baseline VAE model against the tuning of different parameters, inclusion of teacher forcing, and using Self-Referencing Embedded Strings (SELFIES)[1] instead of SMILES. Each model trained on 2.5×10^5 molecules strings from the standard ZINC dataset.

1. INTRODUCTION

Drug discovery, and the optimization of chemical properties of drugs act against a molecular target, are crucial to advancing medications for use in treatment. The drug-like function of molecules is determined by chemical structure, so the drug development process typically begins with choosing new molecular structures before experimenting with their behavior with other molecules. Because the search space of molecular configurations is astronomical, and because the relationship of structure to drug-like function is vastly complex, the task of automatically compressing this statistical information and expressing a targeted search over a refined search space is an ideal candidate for deep learning models. We focus on the task of generating graphs that map to useful, drug-like molecules. Methodologically, we focus on VAEs, whose benefits include solid theoretical foundations, efficiency when combined with constraints built into the model, and practicality of training. We conduct a practical analysis of methods including tuning of parameters such as KL cost annealing, teacher forcing, and using syntax constraints in the form of SELFIES. These all help to address one of the fundamental issues with VAEs: low validity of generated strings.

2. BACKGROUND

Character-Based Chemical Variational Auto Encoder

To start off investigation of predictive models to address drug discovery, exploration began with an unsupervised model: a variational autoencoder,[2] which in its original implementation is comprised of an encoder followed by a decoder, with a property predictor. This model leverages the SMILES representation of molecules, taking the SMILES strings into a continuous vector via the encoder, trained to minimize the reconstruction error and also learn a continuous latent space. In addition, the original encoder was also trained on a property prediction task to predict property values from the continuous vector representation. The VAE in the latent layer adds a stochastic step to the encoder which results in more robust reconstruction of the training and test data. To add this stochastic step, the model learns a mean and variance for each latent variable along with a penalty term to promote valid decoding. Due to the nature of SMILES, this model will still result in invalid molecule generation. By investigating the character-based `chemical_vae` as a baseline, we observe the generative ability provided by a continuous latent space, generating similar chemical strings by exploring the latent space. However, as mentioned, we encounter many syntactically incorrect SMILES strings that do not translate into viable graphs. We will characterize this quality metric by sampling the hidden space and to later compare across models the percentage of generated graphs that are valid. We explore the impact that cost annealing, teacher forcing, and SELFIES instead of SMILES have on the validity of the molecules generated by the model.

1. Mario Krenn et al., "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Machine Learning: Science and Technology* 1, no. 4 (November 2020): 045024, ISSN: 2632-2153, https://doi.org/10.1088/2632-2153/aba947, http://dx.doi.org/10.1088/2632-2153/aba947.

2. Rafael Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," PMID: 29532027, *ACS Central Science* 4, no. 2 (2018): 268–276, https://doi.org/10.1021/acscentsci.7b00572, eprint: https://doi.org/10.1021/acscentsci.7b00572, https://doi.org/10.1021/acscentsci.7b00572.

KL Cost Annealing

The fact that SMILES strings are sparse in the set of possible strings using SMILES characters suggests that significant learning effort must be diverted into the task of making the latent space dense and continuous, i.e. fully realizing the aim of the regularized KL divergence term. The method of *KL cost annealing*[3] proposes to mitigate this issue by scheduling a change in the regularization hyper-parameter (called in some contexts β, often in a static context[4]). Early training emphasizes reconstruction loss, while later training emphasizes KL divergence loss to consolidate the information in the latent space such that its generative ability (and likely validity of generated molecules) is enhanced.

Teacher-Forcing

Aspuru-Guzik et al. note in their character VAE that teacher forcing,[5] or feeding-forward the expected output in place of the actual output $y_k(t)$ of a unit to the next RNN step, in the final output layer of the decoder is expected to increase the fraction of generated molecules that are syntactically valid. Teacher forcing allows the model to learn faster, since we no longer need earlier layers to perform well for later layers to begin to learn, as done in `seq2seq` models. However, a limitation is that teacher forcing can overly de-emphasize the importance of input to the decoder from the variational latent space, which can neutralize some of the generative advantages of a VAE for drug discovery. To our knowledge, existing literature (including that produced by Aspuru-Guzik et al.) does not detail the performance trade-offs of the teacher forcing method. Known implementations of this model by both Aspuru-Guzik's group[6] as well as the implementation by the `deepchem`[7] library https://github.com/deepchem/deepchem/blob/master/deepchem/models/seqtoseq.py do not appear to support teacher forcing. Nor does any literature known to us discuss the possibility of hyper-parameterizing the probability of teacher-forcing in the character VAE's decoder layers.

SELFIES

The Aspuru-Guzik responded to SMILES VAE validity issues in March 2020 with an improvement over the 1988 SMILES language. The Self-referencing Embedded Strings language, or SELFIES, provides a one-to-one mapping from strings of SELFIES tokens to valid molecules. Unlike SMILES, every permutation of SELFIES tokens translates into a valid molecular structure. Moreover, SELFIES accomplishes this while still preserving the ability to represent any molecule.[8]

We expect our VAE models trained with the hard constraints of SELFIES to generate valid molecules with 100% certainty when free sampling. No longer necessitating teacher forcing, we expect training to be easier and also be less prone to overfitting to the ZINC15 dataset, which was discussed as a potential downside to teacher forcing in the original Aspuru-Guzik paper.[9] However, after achieving 100% syntactic validity, the SELFIES-based model begs comparison in overall utility to SMILES with teacher forcing in terms of usefulness to the task of drug-like molecule generation. As previously discussed, the task of property prediction, using metrics such as drug-likeness and synthesizability, is the field's main contribution to answering this open question. Aspuru-Guzik's group refers to this qualitative usefulness as the "chemical richness" and molecular density of the latent space structure. Answering this question quantitatively is outside of the scope of this project, but as further investigation, we suggest further discussion on the comparison of SELFIES-based encoding spaces to other models that impose syntactic and semantic constraints on the latent space, such as grammar VAE[10] and junction tree VAE.[11]

Grammar-Based Chemical Variational Autoencoders

Though not explored in this paper, we would expect improvements over the character-based `chemical_vae` in the Grammar Variational Autoencoder.[12] This model generates syntactically valid inputs from SMILES strings based on its use of the SMILES context-free grammar. SMILES strings are parsed into syntax trees, which the variational autoencoder encodes and decodes discrete data to ensure the outputs provided are syntactically valid. However, note that not all outputs are guaranteed to be chemically realistic or semantically valid; and the original implementation was not free of syntactic issues in generated molecules, though it showed a significant validity improvement over the baseline character VAE. This literature is cited here for comparison with the SELFIES-based approach. We are interested in further comparisons in metrics of chemical exploration feasibility for future progress.

Structure-Based VAEs

While our project here does not implement or evaluate these models, we provide a brief survey of autoencoders that incorporate molecular-specific knowledge into the encoding and decoding process to minimize the number of invalid or useless molecules in the hidden space and improve the interpretability of the hidden space. This is a particular topic of interest again for comparison to the SELFIES-based approach.

While syntactically valid graphs are useful, the molecules generated are often not valid semantically, as they may be

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

4. Irina Higgins et al., "β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK," *ICLR*, 2017,

5. Ronald J. Williams and David Zipser, *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, 1989.

6. https://github.com/aspuru-guzik-group/chemical_vae/blob/master/chemvae/models.py

7. Bharath Ramsundar et al., *Deep Learning for the Life Sciences*, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837 (O'Reilly Media, 2019).

8. Krenn et al., "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation."

9. Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules."

10. Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato, *Grammar Variational Autoencoder*, 2017, arXiv: 1703.01925 `[stat.ML]`.

11. Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola, "Junction Tree Variational Autoencoder for Molecular Graph Generation," *CoRR* abs/1802.04364 (2018), arXiv: 1802.04364, http://arxiv.org/abs/1802.04364.

12. Kusner, Paige, and Hernández-Lobato, *Grammar Variational Autoencoder*.

chemically unstable or simply non-synthesizable. Junction-tree VAE[13] showed one of the first approaches to building auto-encoders from tree scaffolding outwards, using chemically valid substructures to restrict the hidden space to semantically valid structures. This model is implemented in the MOSES repository of generative molecular models that we utilize for this evaluation study, as discussed further in our implementation section.
A 2020 paper by Jin et al[14] builds a specialized autoencoder that encodes representations at explicitly defined levels of granularity, from single-atom representations to structural motif representations. (The decoder hierarchy operates in the reverse direction, from fine-to-coarse molecule reconstruction.) The end result is a variational AE whose internal representation lies in a feature space of structural motifs that can be continuously explored for structurally-valid, unique compounds.

3. PROBLEM FORMULATION

The main unsupervised task of molecule generation draws its intelligence from a collection of known drug compounds formatted as 2.5×10^5 SMILES strings. The property annotations, such as synthesizability and drug-likedness, are not needed for the task, as reconstruction loss (and then later syntactic validity) are the metrics of training and generation success. Underlying this problem formulation is the fact that SMILES is a context-free grammar with explicit production rules. With careful implementation, these rules can be incorporated as constraint with a VAE model (and have been[15]). However, with the naive character-based VAE,[16] the grammar rules are communicated to the model only implicitly in the input data, and as such the model must undertake the additional difficult task of learning syntax during training.

We also experimented using SELFIES which added constraints to the character and string syntax. One of the issues with SMILES is that a large portion of strings do not represent valid molecules. For example, the string "=" SELFIES solves this issue by implementing a constraint so that each string is a valid molecule, while being able to represent every molecule. The advantage is that training using SELFIES will result in a higher validity of molecules generated. However, the elaborateness of molecules generated may not be as diverse as those generate with SMILES. We discuss the implementation process of SELFIES and our results in a later section of this paper.

Data

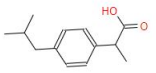
The ZINC15 data set is a publicly accessible at http://zinc15.docking.org; the data set is a collection of commercially available chemical compounds. Focusing on "lead-like compounds" for molecular generation, we loaded the 250k-molecule data set the **deepchem** library. For our MOSES implementation, we also pull from the same data set. ZINC15 supports the SMILES string representation of these molecular compounds. These 2D strings were formed into input batches for the model, which is parametrized with the set of token and a maximum string-length. For our SELFIES implementation, we used the same SMILES, but converted to SELFIES by using mapping tokens for training which will be explained in more detail later in the paper. Several characteristics of the data are listed in Table 2: Data Analysis.

Table 1: Data Analysis

Characteristic	Min	Median	Mean	Max
String Length	9	44	44.3	120
Number of Atoms	6	23	23.2	38
Number of Branches	0	4	3.86	13
Number of Cycles	0	2	2.19	8

An example molecular structure below in SMILES format below, which is commonly known as Ibuprofen:

Figure 1: CC(C)CC1=CC=C(C=C1)C(C)C(=O)O



4. APPROACH AND IMPLEMENTATION

Our approach was to compare models with various architectures to enable analysis of the impact architectural choices made on the number of valid molecules generated.

1) Recreating the baseline model with testing of different parameters. We used the initial code from the following popular and active github repo: https://github.com/deepchem/deepchem/blob/master/deepchem/models/seqtoseq.py.[17] The basic architecture of the model was structured around a VAE (Variational Autoencoder).

2) Implementing baseline model based on the same github repo as leveraged by approach 1) with cost annealing.

3) MOSES VAE implementation which includes teacher forcing. To enable comparison between the codebases, the architecture was kept consistant regarding the number of layers, the drop out leveraged, and the learning rates applied to the model for analysis.

4) SELFIES implementation replaces SMILES. This modification was based on the deepChem code base, using the Python `selfies` library[18] developed by Aspuru-Guzik's group. The input ZINC15 data is preprocessed and re-mapped such

13. Jin, Barzilay, and Jaakkola, "Junction Tree Variational Autoencoder for Molecular Graph Generation."
14. Wengong Jin, Regina Barzilay, and Tommi Jaakkola, "Hierarchical Generation of Molecular Graphs using Structural Motifs," *cs.LG*, 2020,
15. Kusner, Paige, and Hernández-Lobato, *Grammar Variational Autoencoder*.
16. Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules."
17. Ramsundar et al., *Deep Learning for the Life Sciences*.
18. Mario Krenn et al., *SELFIES Python Library*, https://pypi.org/project/selfies/.

that each SELFIES token is represented by a single character, thus allowing us to continue to use the character-based chemical VAE, with the additional post-processing step of inverse mapping to SELFIES, then decoding back to SMILES before evaluation.

The output of each model is a string of generated SMILES sequences in the same format as the ZINC15 dataset (further detail in Data section). The generated output from the model was then tested in RDKit for validity.

Theory

Our approach began with replicating the main baseline model from Aspuru-Guzik.[19] The architecture of the basic model is built off of a seqtoseq implementation that uses GRUs instead of LTSMs. This model is comprised of an encoder and decoder.

Encoder The encoder transforms the input sequence into an embedding - a single vector of fixed length. For this generative model, a variation auto encoder is leveraged since it adds random noise to the encoder and adds a constraint term to the loss that forces the embedding layer to follow a Gaussian distribution.
The encoder used three 1D convolutions layers of size [9, 9, 10] and associated kernel of size [9, 9, 11]. An optional dropout layer can be used before each iteration of the network (which we investigated the performance of adding dropout as part of our process). A ReLU activation function is used, followed by batch normalization.

Decoder The decoder is responsible for transforming the embedding vector into an output sequence. The decoder fed into three layers of a gated recurrent network (GRU).

Loss Function

For an autoencoder model, the basic reconstruction loss is given by:[20]

$$L_1(x) = \log p(x|z)$$

which maximizes the log-likelihood of input distribution x given latent distribution z, which essentially results in learning an identity function despite an information bottleneck.
The VAE model regularizes the reconstruction loss with the KL-divergence between a Gaussian prior and the distribution of the latent space:[21]

$$D_{KL} = -1/2\sum_{j=0}^{k-1}\sigma_j - \mu_j^2 - 1 - log(\sigma_j)$$

This derivation (Gaussian case for the prior $p_\theta(z)$) is given in Appendix B of Kingma and Welling.[22] They importantly point out that this new loss function $L_1 + D_{KL}$ can be estimated with sampling. Gradient differential is addressed with reparametrization, also discussed by Kingma and Welling.

Model Implementation

To ensure everything was working correctly, we first tried with a limited number of molecules from ZINC15 and epochs. We began our initial testing with just using 10k molecules (subset of 250k) from the ZINC15 data set, running on 1 to 5 epochs. Our preliminary findings were strings that were not valid molecules likely resulting from the small data set. We then tested using more epochs from 10 to 50, but the findings were the same. The molecules that were created tended to be simply long, unbranched strings of carbon. For instance, a typical unusable molecule is:
"Cc1ccccccccccccccccccccccccccccc1".
Following our initial testing, we trained our finalized base model using 250k molecules from **deepchem's** ZINC15. We validated our generated strings' SMILES syntax with `RDKit` library functionality.
Fig. 2 and Fig. 3 are examples of molecules generated by this model implementation.

Figure 2: CCCCCCC(=O)CCCCCc1ccccccc1Br

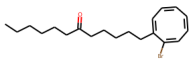
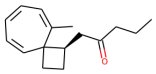


Figure 3: CCCC(=O)C[C@H]1CCC11ccccccc1C



Initially, we found a surprising lack of correlation between the training duration and the validity rate of molecules generated, so further investigation was done to ensure that the loss functionality was performing as expected. As shown in the graphs below, we see stability in the reduction of loss as the number of epochs of training increased. We also confirmed that despite needing more epochs to train, the learning rate of 0.0001 was superior to 0.001 with an annealing cost function, however without the annealing cost function, the lower learning rate of 0.001 provided better results.

Experimental Setup

Our primary training infrastructure is an Azure Machine Learning workspace. We conduct additional experimenting and development in Google Colab. In the Azure ML workspace we use CPU for running basic notebooks that do not require GPU, and we build docker images for training and scoring of models with GPUs for running experiments and tracking metrics. For the GPUs we have started using the NC-series VMs that are powered by NVIDIA Tesla

19. Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules."
20. Andrew Ng, *Sparse Autoencoder*, 2010, https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf.
21. Diederik P Kingma and Max Welling, *Auto-Encoding Variational Bayes*, 2014, arXiv: 1312.6114 `[stat.ML]`.
22. Kingma and Welling.

Figure 4: Examination of training loss with a learning rate of 0.001

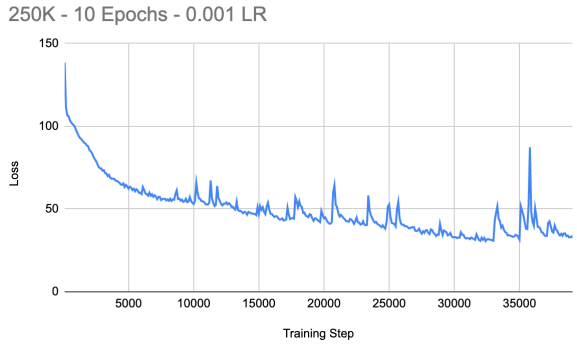
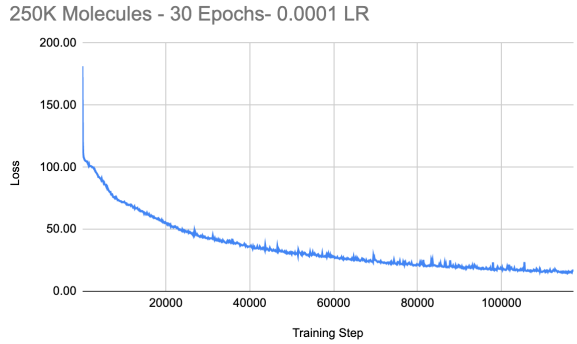


Figure 5: Examination of training loss with a learning rate of 0.0001



K80 card and the Intel Xeon ES-2690 V3 (Haswell) processor.[23] We utilize Google Colab's GPU support on a smaller, free-of-charge scale.

We are accumulating our code under development both these platforms. Our code is to be committed to a Github repository[24] when finalized.

Teacher Forcing Implementation

Beyond the basic model implementation, we had first attempted to add *teacher forcing* within the decoder which was part of the implementation of the from the paper but not implemented in the DeepChem version. As we went through this process, we quickly realized that it was not feasible due to several issues. Our process is walked through in further detail in the next section. Instead, we decided on comparing the baseline model with a different implementation that included teacher forcing which was MOSES which is explained more in detail in our results section.

23. "NC-series Azure Virtual Machines," 2020, https://docs.microsoft.com/en-us/azure/virtual-machines/nc-series.

24. https://github.com/megado123/drug-discovery-vae

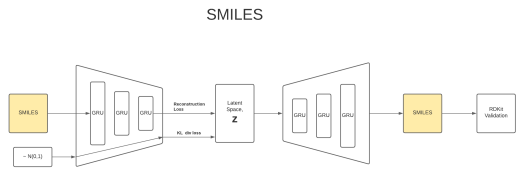


Figure 6: Teacher forcing architecture for a SMILES-based VAE.

Challenges and Learning Process

We faced several challenges during the implementation process. Initially we investigated and installed the repository discussed in the Automatic Chemical Design Using a Data-Driven Continue Representation of Molecules[25] by loading the repository into an Azure ML workspace and trying to produce an image for the code to run on. Without a detailed versioning of the required open source packages to successful run, we were still able to train models with the Aspuru-Guzik group's original code. However, we ran into an issue whose standard solution required modifying Keras source code, preventing us from using the rest of the Aspuru-Guzik group's code for evaluating the model. However, this repository later served us as a useful reference of working code that implemented the VAE with `teacher_forcing_ratio = 1.0` as the final GRU layer in its decoder.

After more research, we changed our approach to align with effort from The DeepChem Project, through a book "Deep Learning for the Life Sciences"[26] and an article "Unlocking Drug Discovery".[27] Through this research we were able to find not only an active code base, but an active community of researchers working together to, "build high quality tools to democratize the use of deep learning in the sciences".[28]

The active DeepChem community produces nightly builds, with which comes a separate, albeit more manageable, challenge for our group. Azure ML has standard images or *curated environments* for using Tensorflow, but no standard environment for Tensorflow 2.4, which is the version that Tensorflow uses. This did cause initial issues, and our team moved temporarily to Google Colab to make use of DeepChem's Colab installer which made setup seamless in the Google Colab environment. We found that while Google Colab was able to get us started, it made training runs difficult to produce. Our training runs have been successfully achieved in Azure ML after later determining the appropriate packages required. In our recent meeting with Bharath, one of the DeepChem project owners, we confirmed that contributing an Azure ML tutorial for DeepChem would be a good addition to DeepChem's public forum.

During our initial attempt in comparing a teacher forcing implementation, we had tried to insert teacher forcing by modifying the DeepChem VAE baseline implementation. As mentioned earlier, we ran into several issues during our coding process which then shifted our intentions from modifying the baseline DeepChem VAE to comparing our baseline

25. Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules."

26. Bharath Ramsundar et al., *Deep Learning for the Life Sciences*, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837 (O'Reilly Media, 2019).

27. W.W. Teka, R.K. Upadhyay, and A. Mondal, "Unlocking Drug Discovery With Machine Learning," *Towards Data Science*, 2019, https://towardsdatascience.com/unlocking-drug-discovery-through-machine-learning-part-1-8b2a64333e07.

28. Ramsundar et al., *Deep Learning for the Life Sciences*.

model with another implementation that included teacher forcing.

5. EXPERIMENTAL EVALUATION

As is shown in Table 2, when generating 1000 molecules with the model created, we find the best model performance with 20 epochs, and a learning rate of 0.0001 provides 0.085 valid molecules, or 85 syntactically valid SMILES strings out of 1000 strings decoded from arbitrary vectors in the learning space.

A learning rate of 0.0001 was found to outperform higher learning rates such as 0.001, suggesting that gradient magnitude might be large (or *exploding*) relative to the training weights for the higher learning rate. Tracking the loss during training, Fig. 4 suggests training instability for the first 10 epochs of 0.001 learning rate relative to the training with learning rate 0.0001 in Fig. 4.

Table 2: Initial Model Training on 250K Zinc Dataset

Run No.	Duration	LR	Epoch No.	Valid Mol
Run 1	26m 28s	0.0001	2	0.034
Run 2	1h 52m 30s	0.0001	10	0.009
Run 3	3h 41m 1s	0.0001	20	0.002
Run 4	25m 6s	0.001	2	0.006
Run 5	1h 51m 24s	0.001	10	0.011
Run 6	3h 40 m 9s	0.001	20	0.006

More surprisingly, however, while loss appears to fall monotonically with number of epochs, the fraction of generated molecules that are valid was not observed to always increase monotonically with training time.

Dropout

Adding dropout has not been observed to make considerable difference for training loss or fraction of valid molecules generated, suggesting that our model is well-regularized and not over-fitted to the data in the early stages of training (less than 30 epochs).

KL Cost Annealing

Within the base model implementation, there were two parameters that we able to set in the initialization. The default parameters were set as start step as 5,000 and final step as 10,000. The step parameters determine just when the constraint is gradually reapplied.

During experimentation as shown in table 3, we saw a significant reduction in the number of valid molecules when no change was scheduled in the regularization hyper-parameter β with a learning rate of 0.0001. This indicates that *KL cost annealing*[29] provides significant improvement in the ability to generate valid molecules. However, with a higher learning rate, it appears that the improvements seen with the implementation of *KL cost annealing* are reduced, as the number of valid molecules generated improved the the hyperparameter was removed from the loss function.

29. Bowman et al., "Generating Sentences from a Continuous Space."

Table 3: Initial Model Training with and without Cost Annealing on 250K Zinc Dataset

LR	Epoch No.	Valid Mol w/o annealing	Valid Mol w/ annealing
0.0001	2	0.002	0.034
0.0001	10	0.002	0.009
0.0001	20	0.0004	0.002
0.001	2	0.014	0.006
0.001	10	0.013	0.011
0.001	20	0.013	0.0061

Teacher Forcing

As expected, teacher forcing increased the number of valid molecules significantly by allowing the decoder to be trained more efficiently. After training, in the testing phase, the decoder is allowed to sample freely, as there is no ground truth in generating from arbitrary samples of the latent space.

Table 4: Moses Model Training Implementation on 250K Zinc Dataset

LR	Epoch No	Valid Mol DropOut 0.5	Valid Mol DropOut 0.0
0.001	2	0.469	0.587
0.001	10	0.916	0.921
0.001	20	0.946	0.93
0.0001	2	0.007	0.013
0.0001	10	0.458	0.524X
0.0001	20	0.684	.748

As demonstrated in Table 4, we see the impact of teacher forcing on the model performance. In order to perform consistent model comparisons, both models were trained with 3 encoder layers, 3 decoder layers. During inital exploration of the DeepChem implementation, it was observed that Dropout had a minimal impact on the amount of valid molecules generated, as was also noted in leveraging the Moses model. Dropout was set to both 0.0 and 0.5 during training of the Moses model at the end of the encoder layer.

Teacher Forcing Annealing

The MOSES library for molecular generation discusses teacher forcing for VAEs, and in fact the community proposes the exploration of teacher forcing annealing[30]. Note that the MOSES Pytorch implementation of Aspuru-Guzik's chemical VAE is equivalent to fixing `teacher_forcing_ratio = 1`, but the community has expressed some interest in not only hyperparametrizing the teacher forcing probability but also enabling scheduled annealing.

Teacher forcing tends to be very useful when the decoder is untrained, but as pointed out in the original paper,[31] it can cause performance discrepancy between the reconstruction observed during training and the generation observed during later sampling. The MOSES community proposed to have teacher forcing *anneal*, or fade in later training, to reduce

30. https://github.com/molecularsets/moses/issues/7

31. Gómez-Bombarelli et al., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules."

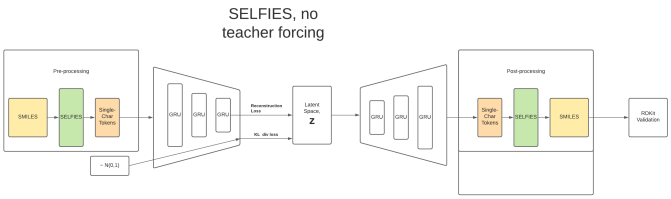


Figure 7: Architecture for a SELFIES-based VAE.

this performance discrepancy and thus improve the capability of the loss to measure the model's generative accuracy at the end of training.

However, this possible improvement was not implemented. It is not certain as to whether this improvement would be appreciable, and while it is a possible next step, it would likely first be preferable to characterize the impact of teacher forcing on chemical richness and generative production before investing a close examination of teacher forcing schedules.

SELFIES

As discussed in the implementation section, our SELFIES approach maintains compatibility with the character-based Aspuru-Guzik VAE, and we chose to continue to use `DeepChem`'s model implementation. The `DeepChem` VAE model implements Batch Normalization and dropout without teacher forcing. The combination of teacher forcing with SELFIES input was not investigated, because teacher forcing can produce counter-productive effects on "expressiveness"[32] of the generative process, and enables benefits that are redundant to an extent with the benefits of SELFIES-encoded input to correct the syntactic validity density of the latent space.

As expected, 100% of the molecules generated by the SELFIES-based decoder were syntactically valid as parsed by RDKit:

Table 5: Initial Model Training
with SELFIES Implementation on 250K Zinc Dataset

LR	Epoch No.	Valid Mol
0.0001	2	1.0
0.0001	10	1.0
0.0001	20	1.0
0.001	2	1.0
0.001	10	1.0
0.001	20	1.0

This solution of the validity issue, which avoids needing to dedicate training time and resources to have the VAE implicitly learn the constraints of the SMILES language, frees us to look at yet more precise measures of chemical exploration utility. In fact, we have already seen hints of syntactic validity being a limited form of chemical VAE validation; the percentage valid for character-based VAEs does not necessary increase monotonically with the number of training epochs, suggesting that the VAE loss is learning much more information than simply the ability to produce syntactically valid molecules.

32. Gómez-Bombarelli et al.

Drug-likeness and synthesizability have been introduced earlier in this paper and are discussed and targeted for property prediction with a separate classifier on the VAE latent space in the original Aspuru-Guzik paper.[33] Training property predictors is a useful next step to the investigation here–we suggest the comparison of the SELFIES-based VAE to the teacher-forced SMILES-based VAE. We also suggest a control of the baseline SMILES-based VAE to characterize the possible detriment of teacher-forcing on latent space chemical richness.

In addition to the empirically-validated properties of chemical utility, there are some explicitly structure-based chemical metrics that do not require models to predict. RDKit offers a library that evaluates Lipinsky's Rule of Five, which is based solely on molecular structure,[34] and thus can be evaluated objectively and reproducibly without the need for a trained predictor model.

6. CONCLUSION

VAE architecture is critical to generating syntactically valid SMILES strings that can be leveraged during the investigation phase of drug discovery. This paper focused on the impact that annealing cost function plays, teacher forcing, and encoding and decoding of SMILES strings as input and output to our RNN model.

Using the KL cost annealing to emphasize reconstruction loss early in training had a significant impact on the model's ability to generate valid molecules when compared to a model that did not leverage KL Cost Annealing, further the impact this has on the model was greater when the learning rate was smaller.

Teacher forcing, enforcing ground truth corrections from the previous time step as input produces a significant increase on a VAE's ability to generate valid molecules, but possibly at the cost of the chemical richness of the latent space.

Training a character-based VAE with SELFIES, the chemical language developed by Aspuru-Guzik et al, guarantees that the entire latent space decodes to syntactically valid molecules, allowing us to repeal some of the drastic tradeoffs and methods that previously were needed to properly train a VAE on SMILES-encoded input. More precise metrics of chemical generative ability are thus warranted following this improvement.

7. CONTRIBUTIONS

We all agree that every member of the team contributed equally. In general, each of us contributed to every task

33. Gómez-Bombarelli et al.

34. Christopher A Lipinski et al., "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings1PII of original article: S0169-409X(96)00423-1. The article was originally published in Advanced Drug Delivery Reviews 23 (1997) 3–25.1," Special issue dedicated to Dr. Eric Tomlinson, Advanced Drug Delivery Reviews, A Selection of the Most Highly Cited Articles, 1991-1998, *Advanced Drug Delivery Reviews* 46, no. 1 (2001): 3–26, ISSN: 0169-409X, https://doi.org/https://doi.org/10.1016/S0169-409X(00)00129-0, https://www.sciencedirect.com/science/article/pii/S0169409X00001290.

to some extent, e.g. literature, results, and development discussions, but specifically notable contributions from each team member include:

1. Megan: Azure ML expert, baseline techniques and teacher-forcing implementation
2. Dixon: Managing write-ups, code validation, chemical result interpretation
3. John: MOSES & technique research, SELFIES R&D, open source outreach
4. Andrew: Dataset exploration, MOSES implementation, baseline techniques implementation

Note that the above list is a results-oriented summary. Many team members contributed towards efforts and challenges that turned out to produce few direct results.

References

Bowman, Samuel R., Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. "Generating Sentences from a Continuous Space." *CoRR* abs/1511.06349 (2015). arXiv: 1511.06349. http://arxiv.org/abs/1511.06349.

Gómez-Bombarelli, Rafael, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules." PMID: 29532027, *ACS Central Science* 4, no. 2 (2018): 268–276. https://doi.org/10.1021/acscentsci.7b00572. eprint: https://doi.org/10.1021/acscentsci.7b00572. https://doi.org/10.1021/acscentsci.7b00572.

Higgins, Irina, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK." *ICLR*, 2017.

Jin, Wengong, Regina Barzilay, and Tommi Jaakkola. "Hierarchical Generation of Molecular Graphs using Structural Motifs." *cs.LG*, 2020.

Jin, Wengong, Regina Barzilay, and Tommi S. Jaakkola. "Junction Tree Variational Autoencoder for Molecular Graph Generation." *CoRR* abs/1802.04364 (2018). arXiv: 1802.04364. http://arxiv.org/abs/1802.04364.

Kingma, Diederik P, and Max Welling. *Auto-Encoding Variational Bayes*, 2014. arXiv: 1312.6114 [stat.ML].

Krenn, Mario, Florian Haese, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. *SELFIES Python Library*. https://pypi.org/project/selfies/.

Krenn, Mario, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation." *Machine Learning: Science and Technology* 1, no. 4 (November 2020): 045024. ISSN: 2632-2153. https://doi.org/10.1088/2632-2153/aba947. http://dx.doi.org/10.1088/2632-2153/aba947.

Kusner, Matt J., Brooks Paige, and José Miguel Hernández-Lobato. *Grammar Variational Autoencoder*, 2017. arXiv: 1703.01925 [stat.ML].

Lipinski, Christopher A, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings1PII of original article: S0169-409X(96)00423-1. The article was originally published in Advanced Drug Delivery Reviews 23 (1997) 3–25.1." Special issue dedicated to Dr. Eric Tomlinson, Advanced Drug Delivery Reviews, A Selection of the Most Highly Cited Articles, 1991-1998, *Advanced Drug Delivery Reviews* 46, no. 1 (2001): 3–26. ISSN: 0169-409X. https://doi.org/https://doi.org/10.1016/S0169-409X(00)00129-0. https://www.sciencedirect.com/science/article/pii/S0169409X00001290.

"NC-series Azure Virtual Machines," 2020. https://docs.microsoft.com/en-us/azure/virtual-machines/nc-series.

Ng, Andrew. *Sparse Autoencoder*, 2010. https://web.stanford.edu/class/cs294a/sparseAutoencoder_2011new.pdf.

Ramsundar, Bharath, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837. O'Reilly Media, 2019.

———. *Deep Learning for the Life Sciences*. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837. O'Reilly Media, 2019.

Teka, W.W., R.K. Upadhyay, and A. Mondal. "Unlocking Drug Discovery With Machine Learning." *Towards Data Science*, 2019. https://towardsdatascience.com/unlocking-drug-discovery-through-machine-learning-part-1-8b2a64333e07.

Williams, Ronald J., and David Zipser. *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, 1989.