Generative schemes for drug design with shape captioning

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Computational Natural Sciences by Research

by

Shikhar Shasya
20171029
shikhar.shasya@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
September 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Generative schemes for drug design with shape captioning" by Shikhar Shasya, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. PRABHAKAR BHIMALAPURAM

To Hope

Acknowledgments

I would like to thank my research advisor Dr. Prabhakar Bhimalapuram for his support and insights that helped me to finish up my research work. Also, I am grateful to my co-author and friend, Shubham for contributing to this work. I acknowledge the support of my friends for helping me with this work, especially Chaitanya for suggesting ideas to improve this work. I would like to thank Rishal Agarwal for clarifying doubts I had relating to my research work. Lastly, I would like to thank my family for their constant support.

Abstract

In this work, we develop three (related) schemes to generate novel molecules based on a seed molecule using an attentive captioning network. Input is the grid representation of the seed molecule, which could be one of the following: voxelised grid of the seed molecule, a grid reconstructed grid from it, or a sampled grid from it. The reconstructed grid is generated by passing the voxelised grid to the variational autoencoder and the sampled grid is generated by conditioning the decoder phase of the variational autoencoder on pharmacophoric requirements. The first scheme called 'Direct Generation' uses a RNN with the voxelised grid of the seed molecule as input to give an SMILES output of generated molecule. The second and third schemes utilize an additional variational autoencoder (VAE) to generate the grid which is used as input to RNN mentioned above in the subsequent step; the latent space of this VAE is modelled as a Riemannian manifold attached with a metric which is learnt along with the encoder and decoder networks of this VAE, which we name RHVAE. The second scheme, named 'Autoencoded Generation', takes as input the seed molecules' voxelised grid representation for the encoder and its output along with the Remianian metric generates the latent space representation; this latent space representation along with the pharmacophoric requirement conditions form inputs for the decoder which outputs the 'reconstruction grid'. In the third scheme, named 'Sampled Generation', starts with a point (and its Remanian metric) randomly sampled from latent space distribution learnt during training. This is evolved using either Hamiltonain Monte Carlo or Random Walk Monte Carlo, and then the evolved point with pharmacophoric conditions is sent to decoder to generate a 'sampled grid'. In the subsequent step of Autoencoded Generation (Sampled Generation), this reconstructed grid (sampled grid) is sent to RNN for obtaining the SMILES of generated molecules. Overall, we demonstrate the generation of meaningful ligand shapes through the autoencoder network which can then be passed to our attentive captioning network to generate novel molecules while requiring smaller data sets for training while retaining the similar performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main focus of drug discovery and design is to find suitable candidates that can bind to some target proteins, thereby altering their properties. The naive approach to do this is to screen the massive chemical space and filter that down to a subset of compounds which are most likely to be active. But this approach is computationally expensive. To enhance this approach, rapid identification of new drug leads is done by assessing the interaction of a large library of small molecules and some target macromolecule. This approach is referred to as virtual screening[1], a modern drug developement approach.

The chemical space is vast, a generative pipeline becomes much more suited for exploring it than brute force search[10]. Some of the most popular generative methods involve RNNs[2], GANs [3] and VAEs[4], transformers[46]. Past work has shown that generative models help in exploring the vast chemical space in an efficient way.

RNNs have been effective in solving language specific tasks, and have been applied in the field of molecular generation as well with SMILES string format used to represent the input molecule [5, 6]. However these approaches are limited in the sense that they can generate SMILES strings which do not correspond to valid molecules. To improve upon this, better ways to learn the grammar have been proposed [7, 8]. Bidirectional generative RNNs[9], in which the SMILE strings are grown in forward and backward direction, improves the proportion of vaild SMILE strings generated. A VAE based approach was proposed where the input molecules was encoded in a continuous latent space and decoded back to it's original discrete representation[10]. The aim was to explore the latent space to generate molecules with desirable properties. Similar to this, a conditional variational autoencoder was used to generate SMILES sequences conditioned on multiple molecular properties [12].To address the issue of posterior collapse in VAE based molecular generation, an alternative loss formulation was proposed, which corrected the underestimated reconstruction loss in prior works[11]. VAEs in conjunction with molecular graph representations were proposed to generate novel molecules stepwise in the form of molecular graphs[13, 14].

As a computational improvement, a flow-based VAE was suggested for molecular graph generation[15]. More recently, conditional β-VAE was proposed to generate novel molecules[16]. In particular, a recurrent, conditional β-VAE was used which disentangles the latent space to aid in the subsequent molec-

ular optimization task for properties like pLogP, QED scores. Molecular graphs was also used along with GANs for molecular synthesis[17]. To address the issue of mode collapse in GANs, valid samples from the generator were stored and used in further training of the model to increase the diversity of molecules being generated[18]. More recent developments include MolGPT[46] which uses the transformer-decoder model to perform the task of molecule generation from a scaffold molecule along with required property conditions.

Reinforcement learning in conjunction with GANs was able to successfully generate novel molecules satisfying desired properties [19, 20, 21, 22]. Recently, SELFIES[23] was introduced which generates a string based representation of molecular graphs where each random string corresponds to a valid molecule. To overcome the limitation of design of molecules based on known seed structures, generation of compounds conditioned on a target protein sequence was proposed[24]. An equivariant diffusion based model that was conditioned on 3D-protein pockets was able to generate novel, diverse ligands with high binding affinity towards the protein pockets[25].

Apart from this, a Shape-Based Generative Modeling [26] was proposed which mapped the 3D grid structure of a reference/seed molecule to a SMILES string. This work variationally designed multiple molecules starting out from a seed molecule. It proposed a probabilistic sampling strategy to generate a SMILES sequence based on the spatial features of the molecule and previous tokens in the sequence; in this study, we use this study (called ShapeVAE) as a baseline. As an extension to this work, ligand shapes were generated complimentary to a protein pocket which were then decoded using a captioning network to get novel molecules[43].

Our proposed method involves three distinct approaches for generating novel molecules. The first approach, named 'Direct Generation' involves generating output SMILES by passing the input grid directly to our attentive captioning network. The second approach, named 'Autoencoded Generation' involves generating output SMILES by passing the reconstructed grid instead of the input grid to our attentive captioning network. Finally, the third approach allows us to generate diverse set of SMILES by sampling grids biased on some input pharmacophoric requirements and passing it to our attentive captioning network. For sampling grids, two schemes are discussed in our work, namely Hamiltonian Monte Carlo and Random Walk Monte Carlo. Our work discusses two pipelines: Riemannian Hamiltonian Variational AutoEncoder (RHVAE)[27] and Attentive Shape Captioning. The RHVAE pipeline generates reconstructed version of the input grid. The Attentive Shape Captioning pipeline generates output SMILES given some input grid; this is achieved by executing the attention mechanism at each decode step of the shape captioning pipeline and choosing the optimal SMILES sequences through beam search[28].

Chapter 2

Theory

This chapter covers some of the mathematical concepts involved in a Riemannian Hamiltonian Variational Autoencoder. In our work, we use the Riemannian Hamiltonian Variational Autoencoder to reconstruct and sample grids starting out from a seed molecule or some pharmacophoric conditions.

2.1 Hamiltonian Markov Chain Monte Carlo

In Hamiltonian Monte Carlo sampler(HMC)[47], the latent space variable z is assumed to be coming from a Eucledian space and follow a target density $\pi(z)$ derived from a potential $U(z)$. The distribution becomes:

$$\pi(z) = \frac{e^{-U(z)}}{\int e^{-U(\bar{z})} d\bar{z}} \tag{2.1}$$

where $U(z) = -\log \pi(z)$ At the time of training the network, it's not possible to know beforehand what $\pi(z)$ is. So we introduce an additional random variable $\rho \in \mathbb{R}^d$ independent of z to sample on behalf of z. This random variable is loosely referred to as the *momentum* and is sampled from $\rho \sim \mathcal{N}(0, \boldsymbol{M})$, where $\boldsymbol{M}$ is the *mass matrix*. We can then work with the extended distribution of variables (z, ρ) where the distribution becomes:

$$\pi(z, \rho) = \frac{e^{-H(z,\rho)}}{\int_{\mathbb{R}^{2d}} e^{-H(z,\rho)} dz d\rho} \tag{2.2}$$

where $H(z, \rho)$ is the *Hamiltonian*.

$$H(z, \rho) = -\log \pi(z, \rho) = -\log \pi(z) + \frac{1}{2} \log \left((2\pi)^d \det \boldsymbol{M} \right) + \frac{1}{2} \rho^T \boldsymbol{M}^{-1} \rho = U(z) + \kappa(\rho) \tag{2.3}$$

Extending the idea of a physical system to the given equation, $U(z)$ is the potential energy and $\kappa(\rho)$ is the kinetic energy. The evolution with time for $(z(t), \rho(t))$ is given by the following sets of equations:

$$\frac{\delta z}{\delta t} = \frac{\delta H}{\delta \rho} = \boldsymbol{M^{-1}} \rho \tag{2.4a}$$

$$\frac{d\rho}{dt} = -\frac{\partial H}{\partial z} = \nabla_z \log \pi(z) \tag{2.4b}$$

The solution to the above set of partial differential equations 2.4 has to satisfy the following conditions:

- Hamiltonian preservation i.e $H(z_t, \rho_t) = H(z_0, \rho_0)$.
- Volume preserving i.e the determinant of the Jacobian should equal 1.
- Time reversible.

To satisfy the given three conditions, a discretization scheme is used to get the approximate evolution of z and ρ. In particular, *Stormer-Verlet* integrator is used to solve the set of PDEs.

$$\rho\left(t+\epsilon/2\right) = \rho(t) - \frac{\epsilon}{2} \cdot \nabla_z H\left(z(t), \rho(t)\right) \tag{2.5a}$$

$$z\left(t+\epsilon\right) = z(t) + \epsilon \cdot \nabla_\rho \left(H\left(z(t), \rho(t+\epsilon/2)\right)\right) \tag{2.5b}$$

$$\rho\left(t+\epsilon\right) = \rho(t+\epsilon/2) - \epsilon/2 \cdot \nabla_z H\left(z(t+\epsilon), \rho(t+\epsilon/2)\right) \tag{2.5c}$$

where ϵ is the step size of the integrator. The integrator is run n times to sample the state (z_n, ρ_n). This state gets accepted with the probability of $min\left(1, \frac{\exp(-H(z_n,\rho_n))}{\exp(-H(z,\rho))}\right)$. If the energy of the new sampled state is higher than energy of the current state, the new state is rejected. Otherwise, it gets accepted. This particular acceptance criterion creates minor fluctuations in energy, so approximately the Hamiltonian is conserved. The integrator is also time reversible and volume preserving.

2.2 Applying HMC for a VAE

In a typical variational autoencoder, the true posterior distribution π corresponding to the encoder network is hard to sample from directly as it's time intractable. To overcome this, we can use the following relation $p_\theta(z|x) = \frac{p_\theta(x,z)}{p_\theta(x)} \propto p_\theta(x,z)$ for every data point x of the data set. Since we condition on x during sampling, $p_\theta(z|x)$ effectively reduces to $p_\theta(x,z)$ as we focus only on the sampling of z. To sample z through an integrator scheme, we need to know the gradient of the distribution $p_\theta(z|x)$. By computing the gradient of the joint distribution $p_\theta(x,z)$, we can also compute the gradient of the true posterior distribution. Also, $p_\theta(x,z) = p_\theta(x|z)q_{prior}(z)$ where $p_\theta(x|z)$ is the distribution corresponding to the decoder network and $q_{prior}(z)$ is the prior distribution of the latent variables z. The equation 2.3 with this new change now becomes:

$$H_x(z,\rho) = -\log p_\theta(x,z,\rho) = U_x(z) + \frac{1}{2}\log\left((2\pi)^d \det \boldsymbol{M}\right) + \frac{1}{2}\rho^T \boldsymbol{M}^{-1}\rho \tag{2.6}$$

where the potential energy $U_x(z)$ is equal to $-\log p_\theta(x,z)$, defined for each data point x of the data set. Applying K iterations of the integrator 2.5, we get the values (z_K, ρ_K). This particular transformation of position and momentum can be defined by the following mapping:

$$\Phi_{\epsilon,x}^{\circ(l+1)} = \Phi_{\epsilon,x}^{\circ(l)} \circ \Phi_{\epsilon,x}^{\circ(1)} \tag{2.7}$$

where $\Phi_{\epsilon,x}^{\circ(l)} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$, $\Phi_{\epsilon,x}^{\circ(0)} = I_d$.

In the Hamiltonian variational auto-encoder[47], the *Stormer-Verlet* integrator 2.5 was used along with a tempering step[52] to sample (z_K, ρ_K). For tempering, we start out from an initial temperature β_0 and update it iteratively according to the following equation:

$$\sqrt{\beta_k} = \left(\left(1 - \frac{1}{\sqrt{\beta_0}}\right)\frac{k^2}{K^2} + \frac{1}{\sqrt{\beta_0}}\right)^{-1} \tag{2.8}$$

The *momentum* ρ is scaled down by a factor of $\sqrt{\frac{\beta_{k-1}}{\beta_k}}$ after each step k of the integrator. As per the work [53], the reparametrization trick can't be used when using the acceptance/rejection step. So this step is skipped. Overall, this leads to an invertible transformation $\hat{\mathcal{H}}_x$ which maps $(z_0, \rho_0) \in \mathbb{R}^d \times \mathbb{R}^d$ to $(z_K, \rho_K) \in \mathbb{R}^d \times \mathbb{R}^d$ with each step k involving running the integrator 2.12 followed by a tempering step.

2.3 Riemannian Hamiltonian Markov Chain Monte Carlo

Assuming the latent variables to be residing in a Riemannian manifold and the manifold being attached with the metric $\boldsymbol{G}$, the idea of Hamiltonian Markov Chain Monte Carlo can be extended to Reimannian manifolds too as shown in [48][49]. The momentum is now sampled from $\mathcal{N}(0, \boldsymbol{G}(z))$ instead of just sampling it from a constant matrix $\boldsymbol{M}$. Using the earlier relations 2.3 2.6, we can compute the Riemannian Hamiltonian as:

$$H_x^{Riem}(z, \rho) = -\log p_\theta(x, z, \rho) = U_x(z) + \frac{1}{2}\log\left((2\pi)^d \det \boldsymbol{G}(z)\right) + \frac{1}{2}\rho^T \boldsymbol{G}(z)^{-1}\rho \tag{2.9}$$

The kinetic energy κ is now a function of both z and ρ and based on equation 2.9 it becomes as follows:

$$\kappa(z, \rho) = \frac{1}{2}\log\left((2\pi)^d \det \boldsymbol{G}(z)\right) + \frac{1}{2}\rho^T \boldsymbol{G}(z)^{-1}\rho \tag{2.10}$$

To get the Hamilton's equations, we differentiate equation 2.9 to get the following set of equations [49]:

$$\frac{\delta z_i}{\delta t} = \frac{\delta H_x^{Riem}}{\delta \rho_i} = \left(\boldsymbol{G}^{-1}(z)\rho\right)_i \tag{2.11a}$$

$$\frac{d\rho_i}{dt} = -\frac{\partial H_x^{Riem}}{\partial z_i} = \frac{\delta \log \pi_x(z)}{\delta z_i} - \frac{1}{2}tr\left(\boldsymbol{G}^{-1}\frac{\delta \boldsymbol{G}(z)}{\delta z_i}\right) + \frac{1}{2}\rho^T \boldsymbol{G}^{-1}(z)\frac{\delta \boldsymbol{G}(z)}{\delta z_i}\boldsymbol{G}^{-1}(z)\rho \tag{2.11b}$$

The integration scheme in equation 2.5 will not be volume preserving in this particular case as ρ is not independent from z. A different integration scheme satisfying the three conditions was proposed and is given by the following set of equations:

$$\rho\left(t+\epsilon/2\right) = \rho(t) - \frac{\epsilon}{2} \cdot \nabla_z H_x^{Riem}\left(z(t), \rho(t+\epsilon/2)\right) \tag{2.12a}$$

$$z\left(t+\epsilon\right) = z(t) + \frac{\epsilon}{2} \cdot \left[\nabla_\rho H_x^{Riem}\left(z(t), \rho(t+\epsilon/2)\right) + \nabla_\rho H_x^{Riem}\left(z(t+\epsilon), \rho(t+\epsilon/2)\right)\right] \tag{2.12b}$$

$$\rho\left(t+\epsilon\right) = \rho(t+\epsilon/2) - \frac{\epsilon}{2} \cdot \nabla_z H_x^{Riem}\left(z(t+\epsilon), \rho(t+\epsilon/2)\right) \tag{2.12c}$$

The equations 2.12 are referred to as the *generalised leapfrog integrator*, and it ensures the Hamiltonian preservation i.e $H(z_t, \rho_t) = H(z_0, \rho_0)$. As per [50][51], the integrator 2.12 is volume preserving and time reversible. Similar to the Hamiltonian Variational Autoencoder[47], we get an invertible mapping $\hat{\mathcal{H}}_x^{Riem}$ which maps (z_0, ρ_0) to (z_K, ρ_K). The key difference though is that the latent space is assumed to have a non-euclidean structure in case of the Riemannian Variational Autoencoder, i.e z is assumed to reside in a Riemmanian manifold attached with a metric $\boldsymbol{G}(z)$. Also, the *generalised leapfrog integrator* is used instead of *Stormer-Verlet* integrator in case of Riemannian Variational Autoencoder. Similar to earlier, we remove the acceptance/rejection step which makes the mapping $\hat{\mathcal{H}}_x^{Riem}$ differentiable, hence the reparametrization trick can be used.

2.4 Details about the metric

The metric $\boldsymbol{G}$ involved in the equation 2.9 is parameterized according to the following relation:

$$\boldsymbol{G}^{-1}(z) = \sum_{i=1}^{N} L_{\psi_i} L_{\psi_i}^T exp\Big(\frac{-||z - c_i||_2^2}{T^2}\Big) + \lambda I_d \tag{2.13}$$

where N is the number of samples, L_{ψ_i} are lower triangular matrices being learnt from the input data points x_i, T is the temperature and λ is the regularization factor. $c_i = \mu(x_i)$ are the mean of the probability distributions associated with the latent variables $z_i^0 \sim q_\phi(z_i^0|x_i) = \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$. L_{ψ_i} can be learnt with the help of multilayer perceptron network that maps the data point x_i to lower triangular matrices L_{ψ_i}.

Chapter 3

Methods

In the 'Data' subsection, the data sets used for training models are described. In the following subsection 'Featurization', we describe the methodology used to generate the voxelized grid for the seed molecule and the calculation of its pharmacophoric features that can be used for conditioning the decoder network. Subsection 'Models' describes details on the variational autoencoder utilizing Riemannian Geometry for latent space (called 'RHVAE') and recurring neural network (RNN) used for generating novel molecules (called 'Captioning Network'). In the subsection 'Model Traning', we detail the some technical details of training the models. The final subection 'Modes of Generation' gives details of the three proposed schemes for generating new compounds.

3.1 Data

Our models were trained on two independent sets of data randomly drawn from the ZINC-15[29] database. The first set consisted of 8000 SMILES strings and the second set consisted of approximately 9.23 million SMILES strings. The first set was used to train the molecular shape generation task and the second set was used to train the captioning task. For the purpose of evaluation, a test set of 53112 SMILES strings were drawn randomly. This set was independent from the previous two sets.

3.2 Featurization

The input SMILES string was converted to 3 dimensional conformers via RDKit[30] and optimized using the MMFF94 force field[31]. The resulting molecular configurations were then featurized as 3D grids. To convert molecules to a grid, each atom was assigned a type based on the following property channels: hydrophobicity, aromaticity, H-bond donors, H-bond acceptors, and heavy atoms. The contribution of each atom with van der Waals radius r_{vdw} to a voxel is defined as follows:

$$n(r) = 1 - exp\Big(- \big(\frac{r_{vdw}}{r}\big)^{12}\Big) \tag{3.1}$$

where r is the Euclidean distance between the atom and the center of grid. These values were computed on a fixed 24Å sized 3D cubic grid centered on the geometric center of the ligand. The grid was discretized as 1Å sized voxels. Similar to the earlier work[26], three feature types were chosen in the calculation of the pharmacophoric points: centers of aromatic rings, H-bond donors, H-bond acceptors. These three feature types were passed as conditional inputs. To account for the lack of rotation and translation invariance, the molecules were randomly rotated and translated before sending it to the voxelisation routine for data enhancement. The molecules were voxelised using the moleculekit framework[32].

3.3 Models

This section describes details on the variational autoencoder utilizing Riemannian Geometry for latent space (called 'RHVAE') and recurring neural network (RNN) used for generating novel molecules (called 'Captioning Network').

3.3.1 RHVAE network

The VAE model employed to generate autoencoded ligand shapes was derived from Riemann Hamiltonian Variational Autoencoder(RHVAE) [27]; the latent space of the VAE is modelled as a Riemannian manifold (R^d, g) where d is the dimension of the manifold and g is the associated Riemannian metric. The metric tensor $\boldsymbol{G}$ is described by the following equation:

$$\boldsymbol{G}^{-1}(z) = \sum_{i=1}^{N} L_{\psi_i} L_{\psi_i}^T exp\Big(\frac{-||z - c_i||_2^2}{T^2}\Big) + \lambda I_d \tag{3.2}$$

where N is the number of samples, L_{ψ_i} are lower triangular matrices being learnt from the input voxel data, T is the temperature and λ is the regularization factor. $c_i = \mu(x_i)$ are the mean of the probability distributions associated with the latent variables $z_i^0 \sim q_\phi(z_i^0|x_i) = \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$. The VAE model was trained on the following loss function:

$$L_{VAE}(\theta, \phi) = \mathbb{E}_{z_i^0 \sim q_\phi(z_i^0|x_i)} \left[\log \frac{p_\theta(x_i, z_i^k, \rho_i^k)}{q_\phi(z_i^0|x_i)}\right] \tag{3.3}$$

where x_i are the input voxel grids, $(z_i^k,\ \rho_i^k)$ are the positions and the momentum calculated after passing the initial states $(z_i^0,\ \rho_i^0)$ to the generalized leapfrog integrator, the integrator being run k times. $\log p_\theta(x_i, z_i^k, \rho_i^k)$ can be expanded to

$$\begin{aligned} \log p_\theta(x_i, z_i^k, \rho_i^k) &= \log p_\theta(x_i, z_i^k) + \log p_\theta(\rho_i^k) \\ &= \log p_\theta(x_i|z_i^k) * \log p_\theta(z_i^k) + \log p_\theta(\rho_i^k) \end{aligned} \tag{3.4}$$

where $x_i' \sim p_\theta(x_i|z_i^k)$ (autoencoded voxel grids). $\mathbb{E}_{z_i^0 \sim q_\phi(z_i^0|x)} \big[\log p(x_i|z_i^k)\big]$ is estimated using binary cross entropy loss since maximizing log likelihood of the expectation reduces to minimizing reconstruction error $L(x_i, x_i')$. Computing $\mathbb{E}_{z_i^0 \sim q_\phi(z_i^0|x_i)} \big[\log p_\theta(z_i^k)\big]$ and $\mathbb{E}_{z_i^0 \sim q_\phi(z_i^0|x_i)} \big[\log p_\theta(\rho_i^k)\big]$ is

trivial as the variables z_i^k, ρ_i^k are sampled from prior distributions $\mathcal{N}(0, I_d)$ and $\mathcal{N}(0, \boldsymbol{G}(z))$ respectively. By plugging these quantities in equation 3.4, the numerator in the loss function can be estimated. Similarly, the denominator in the loss function can be estimated by taking the log likelihood of $q_\phi(z_i^0|x_i) \sim \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$.

(z_i^k, ρ_i^k) can be defined by the following mapping: $(z_i^k, \rho_i^k) = \hat{\mathcal{H}}_{Riemannian}(z_i^0, \rho_i^0)$, where $\hat{\mathcal{H}}_{Riemannian}$ can be seen as normalizing flows with Riemannian Hamiltonian equations[27] applied to the initial state. The Riemannian Hamiltonian equations for the current state (z, ρ) are as follows:

$$\frac{dz_j}{dt} = \frac{\partial H_x^{Riem}}{\partial \rho_j} \tag{3.5a}$$

$$\frac{d\rho_j}{dt} = -\frac{\partial H_x^{Riem}}{\partial z_j} \tag{3.5b}$$

where the Hamiltonian $H_x^{Riem}(z, \rho)$ is given by:

$$\begin{aligned} H_x^{Riem}(z, \rho) &= -\log p_\theta(x, z, \rho) \\ &= -\log p_\theta(x, z) + \frac{1}{2}\log\Big((2\pi)^d \det \boldsymbol{G}(z)\Big) + \frac{1}{2}\rho^T \boldsymbol{G}^{-1}(z)\rho \end{aligned} \tag{3.6}$$

The RHVAE model consisted of three networks: A multilayer perceptron network to learn the matrix L_{ψ_i}, an encoder network to learn the latent space variables $z_i^0 \sim q_\phi(z_i^0|x_i)$ and a decoder network to learn the reconstructed ligand shapes $x_i' \sim p_\theta(x_i|z_i^k)$. Figure 3.1 shows the RHVAE network pipeline to get reconstructed grids starting out from the input grids of seed molecules.

3.3.2 Captioning Network

The shape captioning pipeline was derived from the image captioning networks, which outputs a caption for a particular image. In particular, our network was inspired from Show, Attend and Tell [33] which describes an attention based model that learns to describe the content of images. Figure 3.2 shows the pipeline for generating molecules with the captioning network given some input grid x_i.

First the grid representation of the molecule gets passed to the encoder layer. The encoder layer then outputs the feature map of the input, a reduced representation of the input grid. At the time of training the network, the feature map along with the SMILES representation of the input molecule gets passed on to the decoder network as inputs (teacher forcing). But at the time of inference, only the feature map is passed as input to generate molecules.

At each decode step, an attention-weighted encoding of the 3D grid is generated via the attention network which tells the decoder network which spatial region of the grid it should focus on at a particular timestep t. The attention-weighted encoding then gets passed through a gate. The gate is a function of the decoder's previous hidden state h_{t-1}. The concatenated representation of the gated attention-weighted encoding and the embedding of the previous token is passed to a LSTMCell to generate the current hidden state h_t. The hidden state h_t then gets passed to a linear layer to generate scores for every token of the vocabulary.

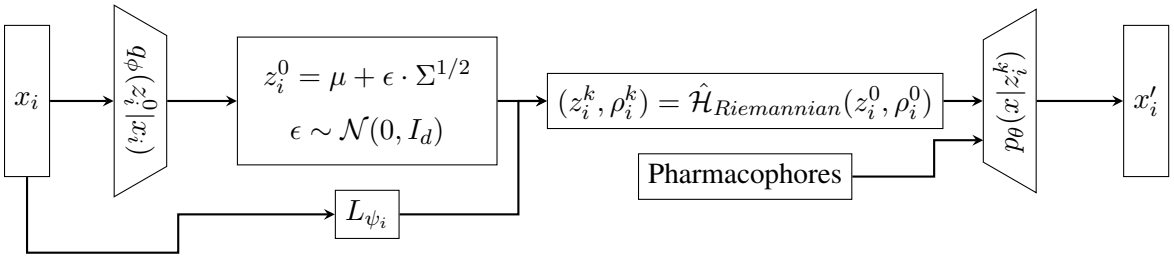


Figure 3.1: RHVAE network pipeline to obtain the reconstructed grid x_i' starting out from the input grid x_i of the seed molecule. L_{ψ_i}, $q_\phi(z_i^0|x_i)$, $p_\theta(x_i|z_i^k)$ represent the Reimannian metric, the encoder network and the decoder network of the VAE respectively. The decode phase of the network is conditioned on prior pharmacophoric requirements to get the output grid.

At the time of inference, the molecules are generated through the decode phase of the captioning network(see Figure 3.2). The decode phase generates a token S_t at each step and terminates when an end token gets selected as the output token. All tokens except for the end token are concatenated sequentially to get the output SMILES. To predict a diverse set of molecules similar to the seed molecule, beam search is used. This allows us to keep track of top k(beam size) SMILES strings at each timestep t, which then get passed on to the subsequent timesteps. At each timestep t, combination of the top k strings at timestep $t-1$ with the k generated tokens is considered. These combinations are sorted out by the additive scores and top k strings gets picked for the next step. At the end of a beam search, we get k number of molecules per seed molecule sorted by their scores.

The captioning network is trained on a sum of 2 terms: a crossentropy loss to minimize the distance between target and input and a "doubly stochastic regularization" to encourage the model to pay equal attention to each spatial region of the grid.

$$L_{caption} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}) + \frac{\lambda}{N}\sum_{i=1}^{N}\sum_{v}(1-\sum_{t}^{T'}\alpha_{itv})^2 \tag{3.7}$$

In the above equation, i spans the minibatch dimension, C is the number of tokens in the vocabulary,v spans the voxel grid dimension, T' is the number of timesteps, α are the weights outputted by the attention network and λ is a regularization parameter.

The captioning network comprises of an encoder and a decoder. The encoder was based off on the VGG-16[34] adapted to the 3D inputs. The decoder network was an attentive RNN network.

3.4 Model Training

Both the networks(RHVAE and captioning network) were optimized with the ADAM optimizer[35] using following momentum parameters: $\beta = (0.9, 0.999)$. A batch size of 16 and a learning rate of 10^{-4}

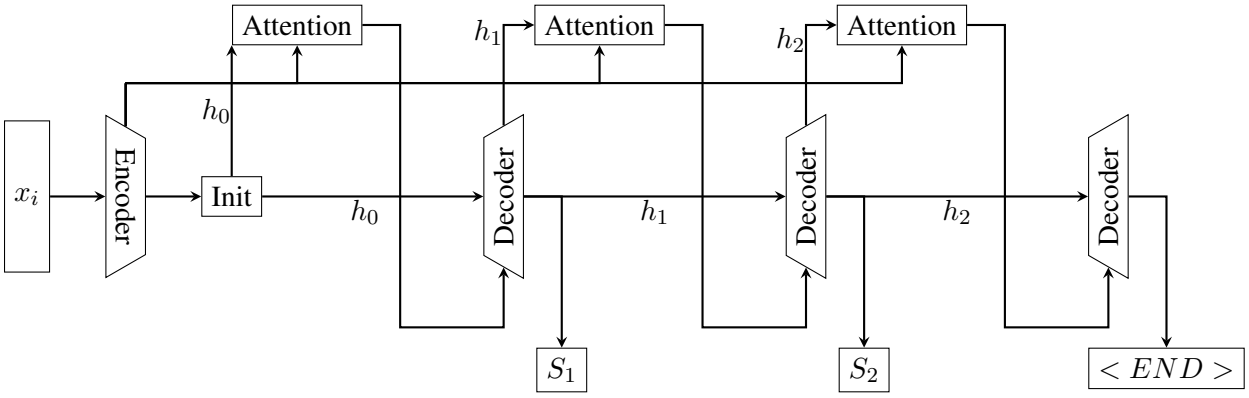


Figure 3.2: Compound generation pipeline via the captioning network at the time of inference. The tokens S_t generated at each timestep t are concatenated (barring the $< END >$ token) to get the output SMILES string.

was picked for the RHVAE network. For the captioning network, the values were 128 and 10^{-3}. The RHVAE network and the captioning network were trained for 25000 and 40000 iterations respectively. The training took approximately a day on a single NVIDIA GeForce GTX 1080Ti GPU for each of the models. The code was written in PyTorch[36] deep learning framework. The details of the architecture and the hyperparameters used for the networks can be found in the Supporting Information.

3.5 Modes of generation

3.5.1 Direct Generation

This method is used to generate molecules which are minor modifications of the seed molecule. The seed molecule is converted to a 3D conformer, which then gets converted to a voxelised grid. This voxelised grid gets passed to the encoder network, whose output is then passed to the decoder network to generate SMILES strings(see Figure 3.2). We note that that different grid representations of the same input molecule can result in different SMILES input.

3.5.2 Autoencoded Generation

This method is used to introduce relatively bigger changes in the seed molecule, similar to the earlier work[26]. The autoencoded representation of the input voxel grid is used as input for the captioning network instead of directly feeding the input grid (which is named direct generation in the above subsection). The autoencoded voxel grid(generated in Figure 3.1) is computed by passing the input grid to the RHVAE network along with the pharmacophoric points. This autoencoded voxel grid gets passed to the captioning network to generate SMILES strings.

3.5.3 Sampled Generation

All the earlier works for compound generation modelled the latent space as a prior distribution to generate new compounds. This approach is not always optimal since the prior may represent the latent space poorly. Also, when working with smaller datasets, sampling this way can result in outputs that are not meaningful. To address these limitations, we discuss a recently proposed approach: Hamiltonian Monte Carlo[37] and Random walk Monte Carlo to sample data from the latent space. In this study, the initial position for MCMC sampling is *not* obtainied by running the encoder phase of RHVAE, but it is sampled randomly from the the encoded distribution of latent variables; the encoded distribution is obtained from the distribution formed during the training phase. Both these methods involve evolving the position z using classic MCMC sampling, but use a different Hamiltonian function for the dynamics. The evolved z along with the pharmacophoric requirements is passed to the decoder network p_θ of the RHVAE to get the sampled grid(see decoder phase of Figure 3.1). This sampled grid gets passed to the captioning network to generate SMILES strings.

Hamiltonian Monte Carlo (HMC): Instead of sampling from a prior distribution to generate new samples, we use Hamiltonian Monte Carlo sampler to sample z. This is done so as to better capture the intrinsic latent structure of the data. Given a random variable $v \sim N(0, I_d)$ independent from z, we use Hamiltonian dynamics to update the values for z. z and v can be loosely referred to as the position and velocity of a physical system with the potential energy $U(z)$ and kinetic energy $K(v)$ given by the following equations:

$$U(z) = -\log p_{target}(z) \tag{3.8a}$$

$$K(v) = \frac{1}{2} v^T v \tag{3.8b}$$

The sum of these energies gives us the Hamiltonian $H(z, v)$[37]

$$H(z, v) = U(z) + K(v) \propto -\frac{1}{2} \log\det \boldsymbol{G}^{-1}(z) + \frac{1}{2} v^T v \tag{3.9}$$

The evolution of $(z(t), v(t))$ is given by the following equations:

$$\frac{dz_j}{dt} = \frac{\partial H}{\partial v_j} \tag{3.10a}$$

$$\frac{dv_j}{dt} = -\frac{\partial H}{\partial z_j} \tag{3.10b}$$

To sample from the target density function p_{target}, MCMC sampling scheme is employed. Given z_0^n, the current state of the Markov chain (z^n), an initial velocity v_0 is sampled from $N(0, I_d)$. The leapfrog integrator is run l times to get the state (z_l^n, v_l). The proposed state z_l^n is then accepted with the probability $\alpha = \min\left(1, \frac{\exp\left(-H(z_l^n, v_l)\right)}{\exp\left(-H(z_0^n, v_0)\right)}\right)$

Random Walk Monte Carlo (RWMC): In this approach, we compute the trajectory for the position z and momentum ρ through Hamiltonian dynamics. The trajectory is computed for some fixed number of steps. The evolution of $(z(t), \rho(t))$ is given by the following equations:

$$\frac{dz_j}{dt} = \frac{\partial H}{\partial \rho_j} \tag{3.11a}$$

$$\frac{d\rho_j}{dt} = -\frac{\partial H}{\partial z_j} \tag{3.11b}$$

The following system of PDE is approximated through the following equations:

$$\rho\left(t + \epsilon/2\right) = \rho(t) - \frac{\epsilon}{2} \cdot \nabla_z H\left(z(t), \rho(t)\right) \tag{3.12a}$$

$$z\left(t + \epsilon/2\right) = z(t) + \frac{\epsilon}{2} \cdot \nabla_\rho H\left(z(t), \rho(t)\right) \tag{3.12b}$$

$$\rho\left(t + \epsilon\right) = \rho(t) - \epsilon \cdot \nabla_z H\left(z(t + \epsilon/2), \rho(t + \epsilon/2)\right) \tag{3.12c}$$

$$z\left(t + \epsilon\right) = z(t) + \epsilon \cdot \nabla_\rho H\left(z(t + \epsilon/2), \rho(t + \epsilon/2)\right) \tag{3.12d}$$

The equation for calculating the Hamiltonian $H(z, \rho)$

$$H(z, \rho) = \frac{1}{2} \rho^T \boldsymbol{G}^{-1}(z) \rho \tag{3.13}$$

The leapfrog integrator is run l times to get the final state of the position and momentum trajectory. The random walk is simulated by running the MCMC for a fixed number of steps. The proposed state z_l^n is accepted with the probability $\alpha = \min\left(1, \frac{\sqrt{\det \boldsymbol{G}^{-1}(z_l^n)}}{\sqrt{\det \boldsymbol{G}^{-1}(z_0^n)}}\right)$ where z_0^n is the initial state after completion of n steps.

Chapter 4

Results and Discussions

In this section, we discuss the results from three different modes of generating novel compounds starting out from a seed compound or some pharmacophore conditions. The test set of 53112 ligands was used in two of the generation tasks: direct generation and autoencoded generation. The earlier work is referred as ShapeVAE[26] in the upcoming sections.

4.1 Direct Generation:

For direct generation of molecules, a beam size of $k = 5$ was taken i.e for each seed molecule five SMILES strings were generated via the captioning network. Out of the $53112 * k = 265560$ SMILES strings generated, 25192 were invalid (9.48%) according to the default parser present in RDKit. Filtering out the duplicates among all valid strings, we were able to generate 238545 (89.82%) unique SMILES strings. To study the similarity to the seed molecule, mean Tanimoto similarity of Morgan fingerprints was calculated among the k generated valid structures.

Figure 4.1 shows the distribution of the mean similarity of generated molecules to the seed molecule and the absolute difference in property counts for the following properties: Aromatic, H-bond acceptor/donor. Out of all the generated compounds, 91.888%, 64.07% and 88.59% of them had the same property count as the seed molecule for the aromatic, H-bond acceptor and H-bond donor properties respectively. The model was able to retrieve the same property count most of the times except for the case of H-bond acceptor groups where same property count was harder to achieve. The generated compound was rarely ever the same as the seed molecule, demonstrating that our captioning network was not overfit during the training phase.

4.2 Autoencoded Generation

Similar to direct generation, beam size $k = 5$ is used for the autoencoded generation of molecules. For calculating similarity of the generated compounds to the seed molecule, the compound with the maximum Tanimoto similarity is picked. In Figure 4.2, we can observe the peak for Tanimoto similarity

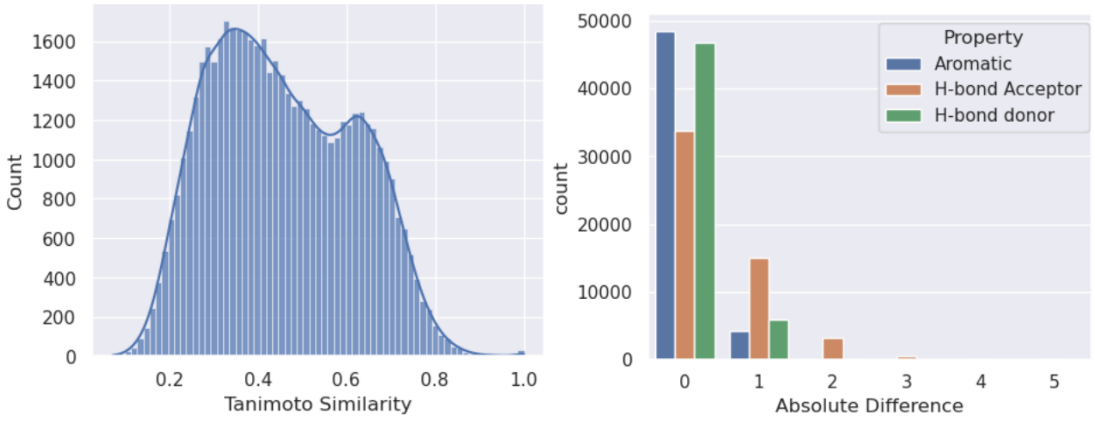


Figure 4.1: Statistics of the generated molecules using the direct method.(left) Mean Tanimoto similarity between the generated molecules and the seed molecule.(right)Absolute difference in property counts between the generated molecule and the seed molecule.

is around 0.33 for our work and 0.26 in the earlier work. 91.5%, 46.9% and 75.3% of all the generated compounds had the same property count for the aromatic, H-bond acceptor and H-bond donor properties respectively(Figure 4.3, right). For the earlier work, following values were observed: 95.9%, 47.02%, 75.51%(Figure 4.3, left). In both the cases, we observe that H-bond acceptor count was the toughest to retrieve.

4.3 Sampled Generation

In case of ShapeVAE, when passing the pharmacophoric points along with a random z, the generated shapes were not meaningful. The shape surfaces generated either had negligible overlap with the pharmacophoric points or no spatial points were generated corresponding to the pharmacophoric conditions. In comparison to this, RHVAE was able to efficiently sample ligand shapes conditioned on the pharmacophoric points through three sampling schemes: HMC, Random Walk Monte Carlo and prior sampling. Figure 4.5 shows the similarity of the sampled voxel array to the pharmacophoric points(Jaccard index) for the following three channels: Aromatic, H-Acceptor and H-Donor. The sampled ligand shapes had a reasonable similarity with the pharmacophoric points in all of the three sampling methods. Figure 4.6 shows an example of the compounds generated through the three sampling methods given some input pharmacophoric points. We observed that HMC and prior sampling had better retention of the pharmacophoric requirements relative to RWMC sampling, which introduced more randomness in the generated molecules.

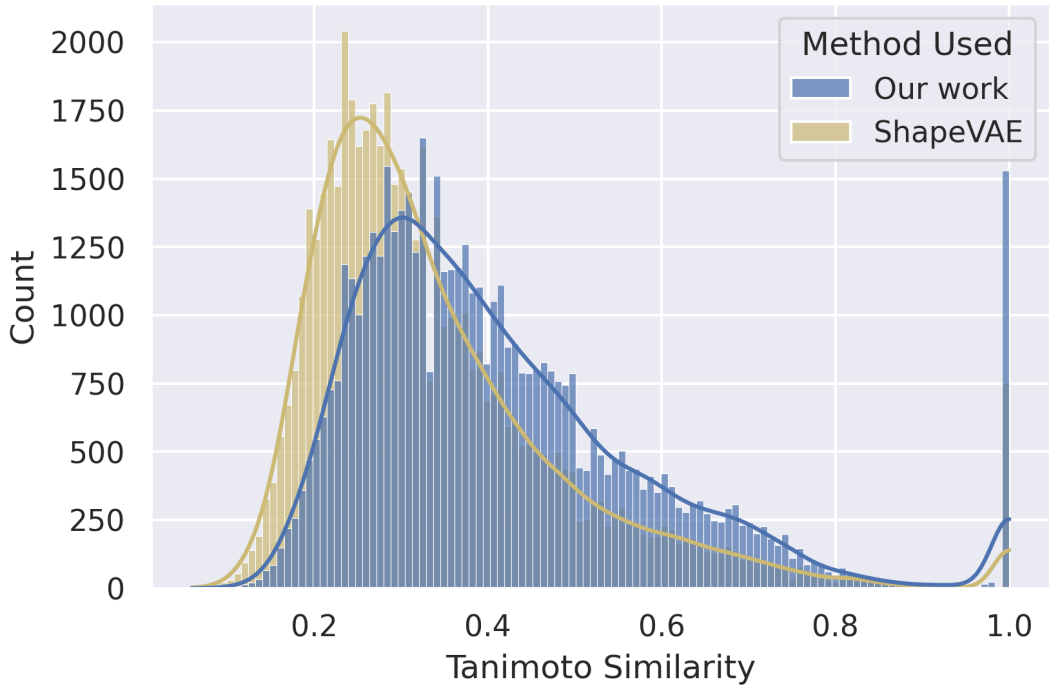


Figure 4.2: Tanimoto similarity between the generated molecules and the seed molecule for ShapeVAE(yellow) and RHVAE(blue). The reconstructed grid along with the pharmacophoric requirements were passed to the captioning network to generate molecules.

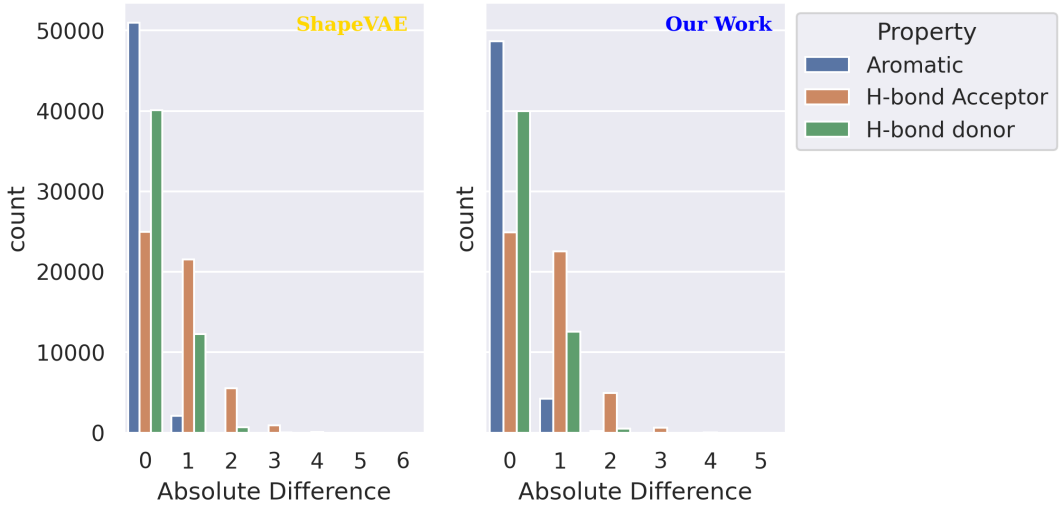


Figure 4.3: Comparison of absolute difference in property counts between the generated molecule and the seed molecule with the left plot showing ShapeVAE and the right plot showing RHVAE. The reconstructed grid along with the pharmacophoric requirements were passed to the captioning network to generate molecules.

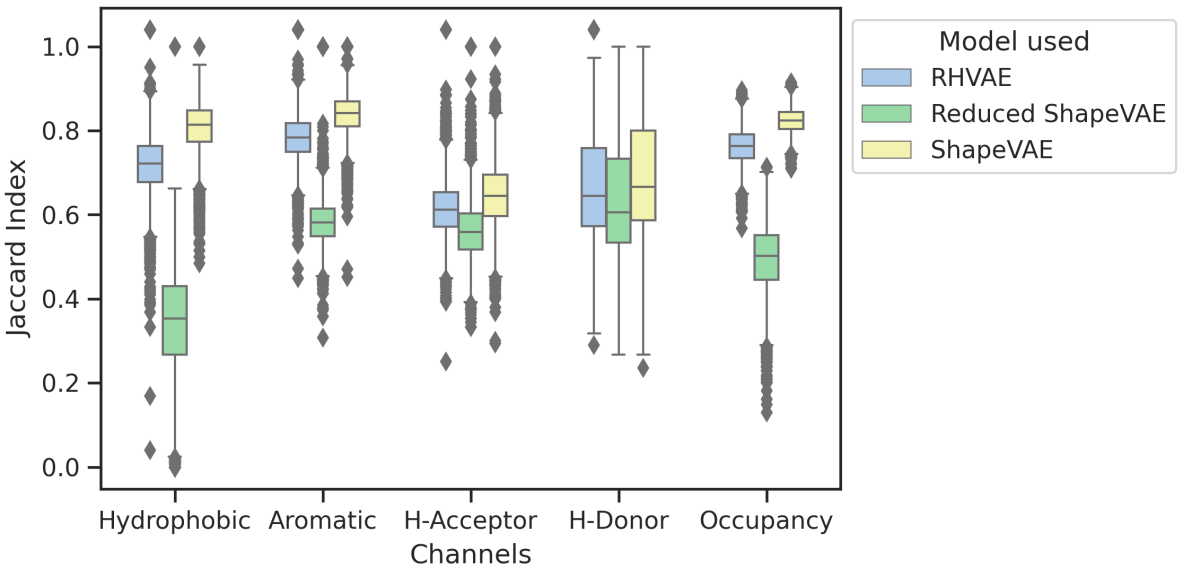


Figure 4.4: Jaccard index comparison between RHVAE(blue), ShapeVAE trained on a reduced dataset of 8000 ligands(green), ShapeVAE on the original dataset(yellow). Jaccard index was computed between the input grid and the reconstructed grid for all 5 channels. A threshold of 0.75 was used to evaluate the index.

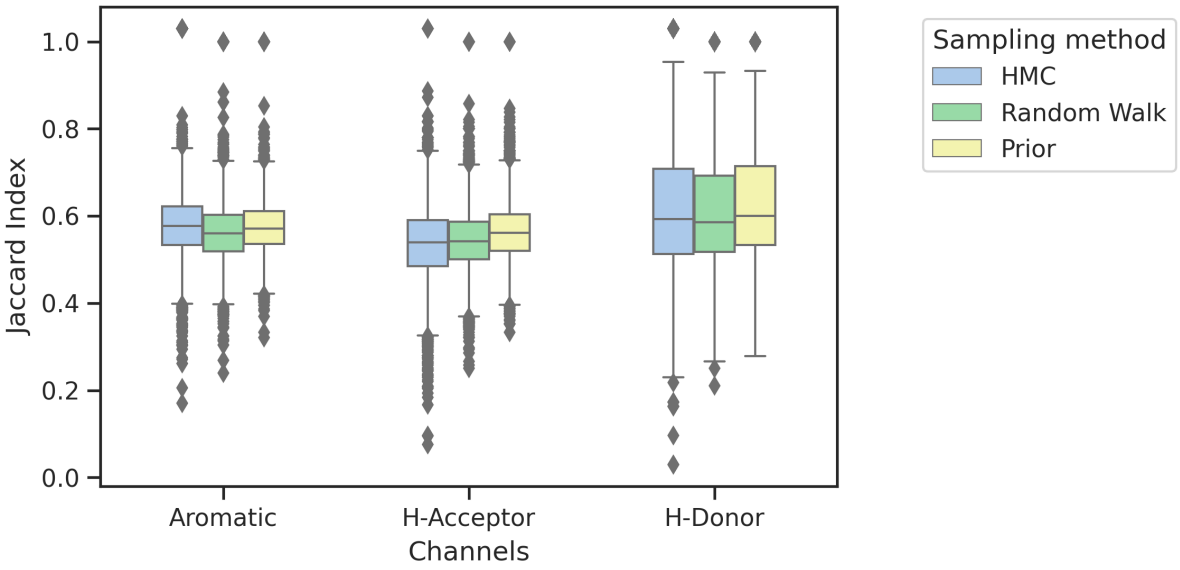
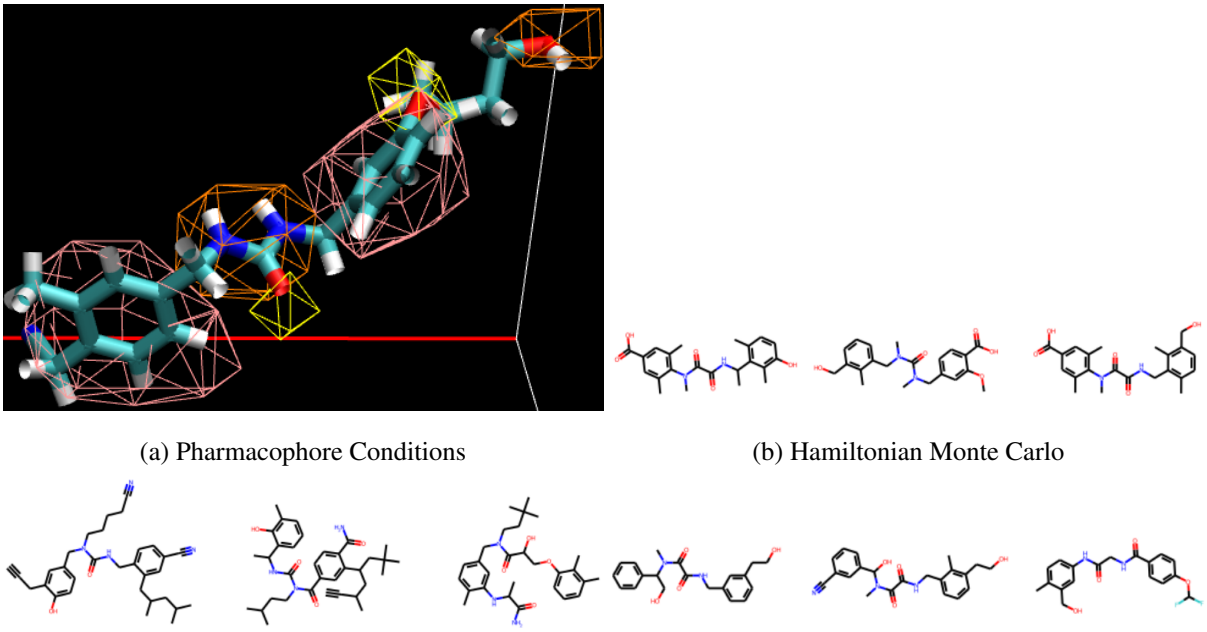


Figure 4.5: Jaccard index comparison between different sampling methods(RHVAE). Jaccard index was computed between the input grid and the sampled grid for aromatic, H-bond Acceptor and H-bond donor channels. A threshold of 0.75 was used to evaluate the index.

4.4 Strengths and Limitations

Our RHVAE model was able to efficiently reconstruct ligand shapes from a small training set of 8000 molecules. This is in contrast to the earlier work, where the VAE network was trained in conjunction with the captioning network on 26.88 million molecules. To highlight the strength of our RHVAE model, the VAE model of the earlier work was trained independently on the same dataset of 8000 molecules used by our model. In Figure 4.4, we can see that the jaccard index for the following channels: Hydrophobic, Aromatic and Occupancy were significantly lower in case of Reduced ShapeVAE (trained on our reduced dataset of 8000 molecules) in comparison to our RHVAE model. The jaccard index for the RHVAE model was slightly lesser than the ShapeVAE(trained on 26.88 million molecules) for all the channels. This is acceptable because the RHVAE model was trained on a significantly lesser number of examples: 8000 in comparison to 26.88 million examples for ShapeVAE. Apart from this, the RHVAE model was able to efficiently sample in latent space through different sampling methods discussed earlier. In case of ShapeVAE, sampling in latent space didn't yield meaningful shapes.

In case of Reduced ShapeVAE, the shapes generated via reconstruction/sampling had a high rate of generating invalid strings when passed to the captioning network since the VAE was not able to reconstruct the input shape effectively. Our method overcomes this limitation as our captioning network despite being trained independently from the RHVAE network was still able to generate valid SMILES



(a) Pharmacophore Conditions

(b) Hamiltonian Monte Carlo

(c) Random Walk Monte Carlo

(d) Prior

Figure 4.6: Some of the molecules generated using HMC, Random Walk Monte Carlo and Prior sampling given some input pharmacophore conditions (aromatic rings, H-bond acceptor and H-bond donor channels in pink, yellow and orange respectively).

strings at a high rate when passed the reconstructed shapes. This shows that the shapes reconstructed were meaningful.

Our work offers multiple ways of generating molecules, namely direct generation, autoencoded generation and pharmacophoric generation but is still limited because all these methods require some information of the ligand as input. For example, when sampling ligand shapes, we need to feed pharmacophoric points as conditional input to the decoder model. Removing the pharmacophoric information from the decoder model leads to loss of information in H-bond acceptor and H-bond donor channels. We attribute this problem to the sparse nature of the input grid. An extensive hyperparameter search and using a alternate loss formulation[38, 39] might overcome this limitation.

4.5 Choice of the latent dimension

When training a VAE model, the size of the latent code becomes crucial. We observe that increasing the latent dimension aided in the reconstruction process of the ligand shape but it came at the cost of introducing numerical instability when sampling. This was because the determinant of the inverse metric $\boldsymbol{G}^{-1}(z)$ collapsed to 0 and computing $\boldsymbol{G}(z)$ became infeasible. To overcome this, we chose a low dimensional latent code of $d = 10$ to aid in sampling from the latent space. This led to a slight

increase in the reconstruction error but enabled us to generate compounds through various sampling methods.

4.6 Evaluation on DUDE Dataset

In this section, we evaluate the model performance on the DUDE dataset[40]. In particular, we compare performance of the earlier work and our work on 482 known actives for adenosine A2A receptor (AA2AR set). For each of these 482 seed molecules, upto 10 compounds were generated by passing autoencoded ligand shapes to the captioning network. The comparison was done on the following properties: logP [41], quantitative estimate of drug-likeness (QED)[42], similarity to the seed molecule and mean similarity within the generated molecules per seed molecule. Figure 4.7 shows the plots for these properties. We observe the distribution of the properties is similar for ShapeVAE and RHVAE. This shows that our RHVAE model was efficient at reconstructing ligand shapes even though it was trained on a significantly smaller training set of 8000 ligands.

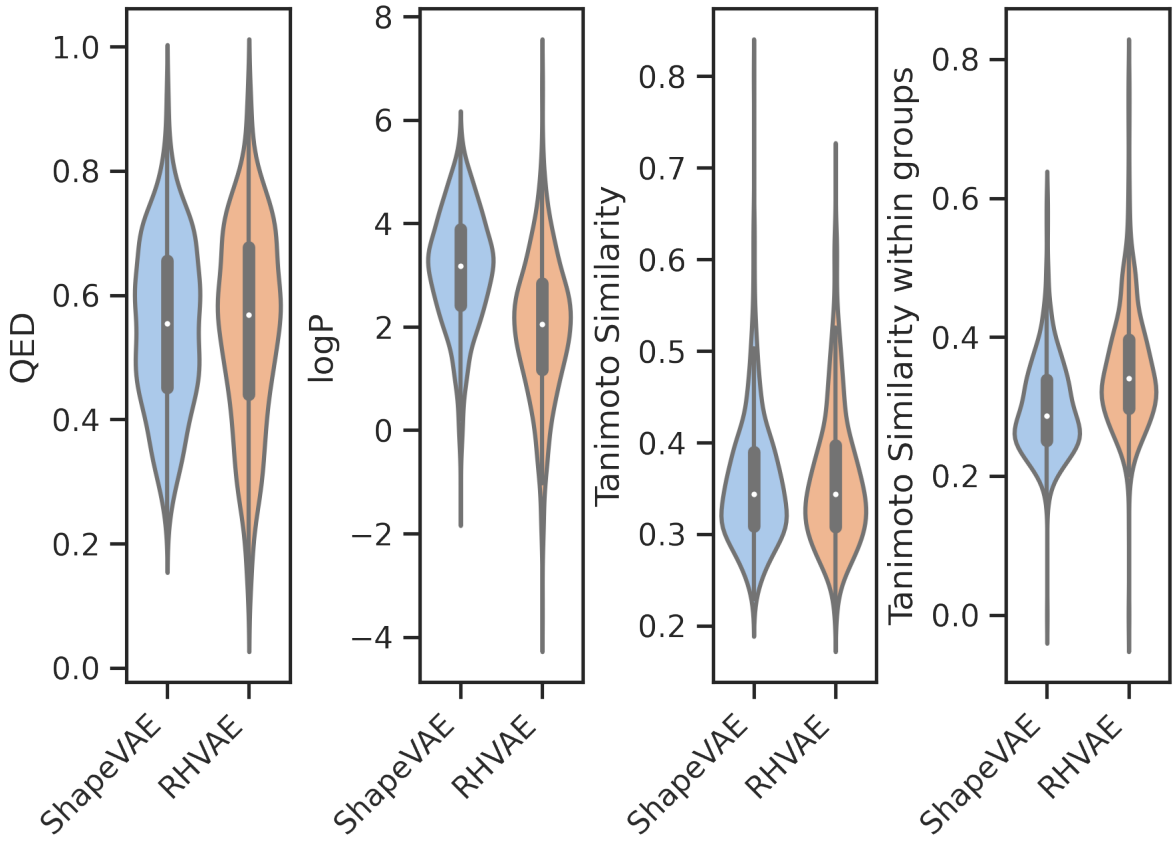


Figure 4.7: Comparison of properties of generated molecules starting out from 482 known actives for adenosine A2A receptor. For each of the actives, upto 10 molecules were generated and the property calculations were done on this.

Chapter 5

Conclusion

In this work, we developed three methods to generate molecules, namely direct generation, autoencoded generation and sampled generation via HMC/RWMC sampling (see Software Availibility and training data chapter for code and data availability). The three discussed methods allowed us to generate novel molecules with desirable properties starting out from some grid representation of a molecule. Our approach used Reimannian Hamiltonian Varitional Autoencoder along with an attentive captioning network to generate various molecules, similar in properties of the seed molecule. The autoencoder network used in our work was able to efficiently reconstruct grids, even though it was trained on a small dataset. This was a major improvement from the earlier work[26], where the autoencoder network was jointly trained with the captioning network on a significantly larger dataset. Apart from this, our autoencoder network was also efficient at sampling grids as they had a decent overlap with the pharmacophoric requirements, again an improvement from the earlier work[26] where the sampled grids were not meaningful.

Even though our proposed method allows us to generate a diverse set of novel molecules, it still has limitations. For example, it uses only information of ligands for the drug discovery process. As an improvement to this, we could generate ligand shapes conditioned on the protein pocket[43]. As an extension to the current work, we could train an optimizer to predict properties of molecules given the latent code z as input; this would allow us to traverse the continuous latent space to find latent representations corresponding to molecules with the optimal properties. Chemical VAE[10] used Bayesian optimization to find the optimal latent codes that correspond to molecules with a high drug-likeness and synthetic accessibility. Conditional β-VAE[16] used Molecular Swarm Optimization[45] to optimize properties like pLogP, QED with a reduced latency compared to Bayesian optimization. Future work will incorporate these models into the proposed method in this work. Another limitation of this work is that the captioning network maps the three dimensional grid to a SMILES string, thereby abandoning the three dimensional information of the ligand. Recent works [44] have proposed ways to generate 3D molecular structures conditioned on a receptor's binding site to address this shortcoming. Coupling the sampling schemes discussed in our work with this can be a direction for future work.

Chapter 6

Supporting Information

6.1 Model Architectures

The attention block of the captioning network consists of 3 linear layers. The first two linear layer transform the encoded image(feature map)($N, 24^3$,encoder dimension) and the hidden state to the same output dimension i.e the attention dimension. The outputs of these layers are then added and passed through ReLU activation. This then gets passed to a final linear layer that transforms the output to a dimension of 1, which then gets passed to a softmax layer to generate the weights alpha.

The attention-weighted encoding is generated with weights alpha at each time step t. The attention-weighted encoding then gets passed through a sigmoid activated gate. The gated attention-weighted encoding along with the embedding of the previous word gets passed to a LSTMCell to get the new hidden state. A final linear layer transforms the hidden state output dimension to the vocab size. The hidden state is initialised through a linear layer transforming the input encoder dimension to the hidden state dimension.

6.1.1 Hyperparameters

Table S1: Captioning Network Hyperparameters

Parameter name	Value
Encoder dimension	128
Initial layer input, output dimension	(128,512)
Embedding dimension	512
Hidden state dimension	512
Attention dimension	512
Gate input, output dimension	(512,128)
LSTMCell input, output dimension	(640,512)
Vocab size	29

Table S2: RHVAE Hyperparameters

Parameter name	Value
Latent dimension(d)	10
Temperature(T)	1.5
Regularization factor(λ)	0.01
Number of steps(n_{lf})	3
Step size(ϵ_{lf})	0.001

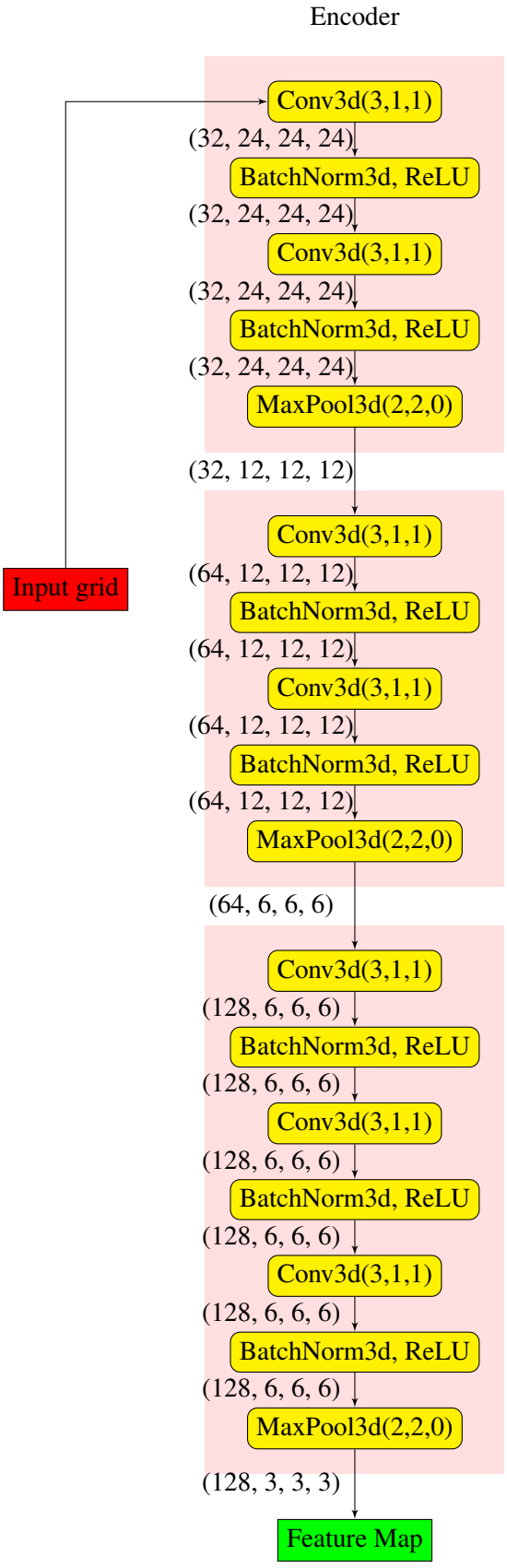


Figure S1: Encoder network in the captioning network

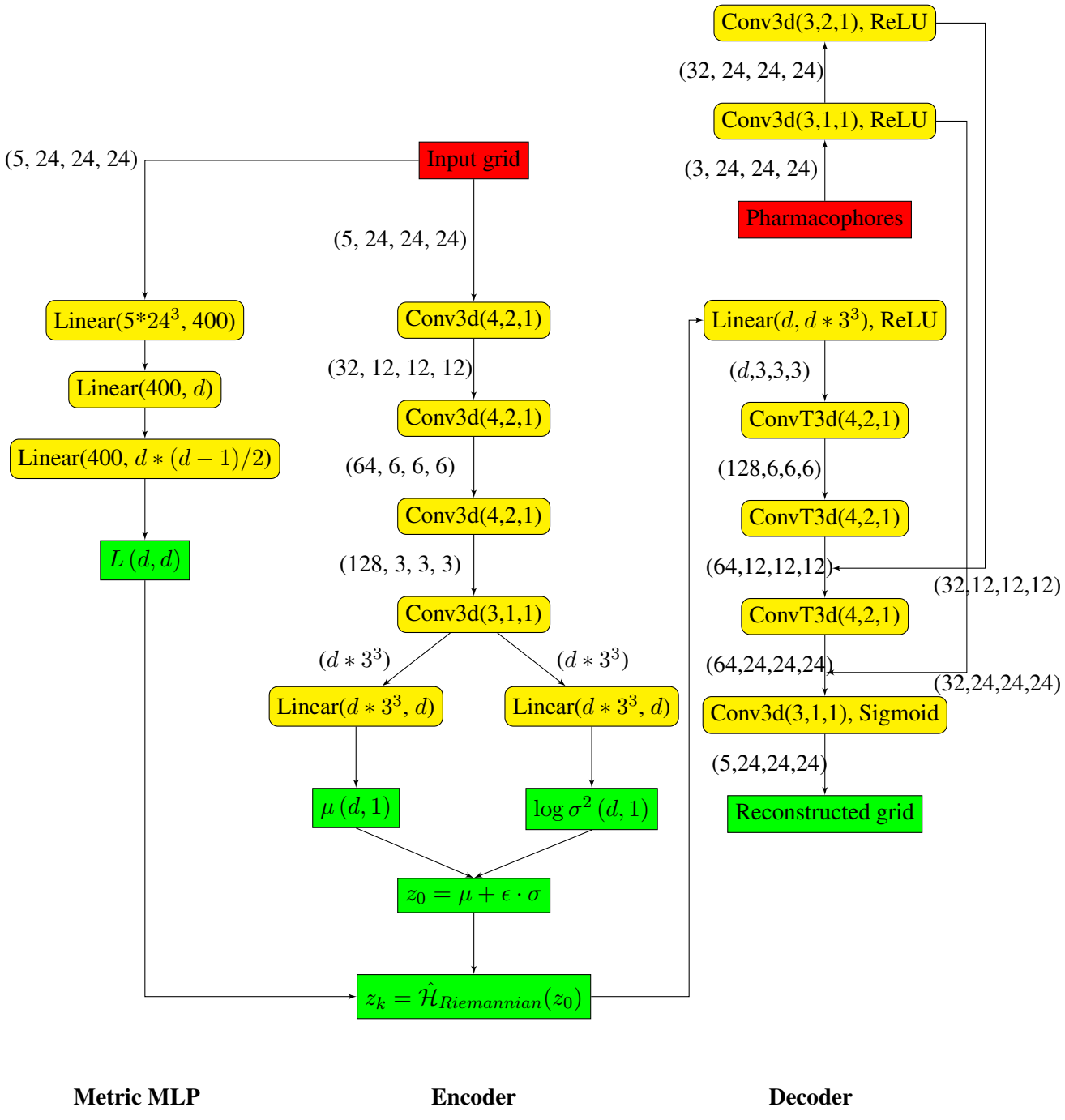


Figure S2: RHVAE Network consisting of a Metric MLP, an encoder and a decoder.

6.2 Loss Plots

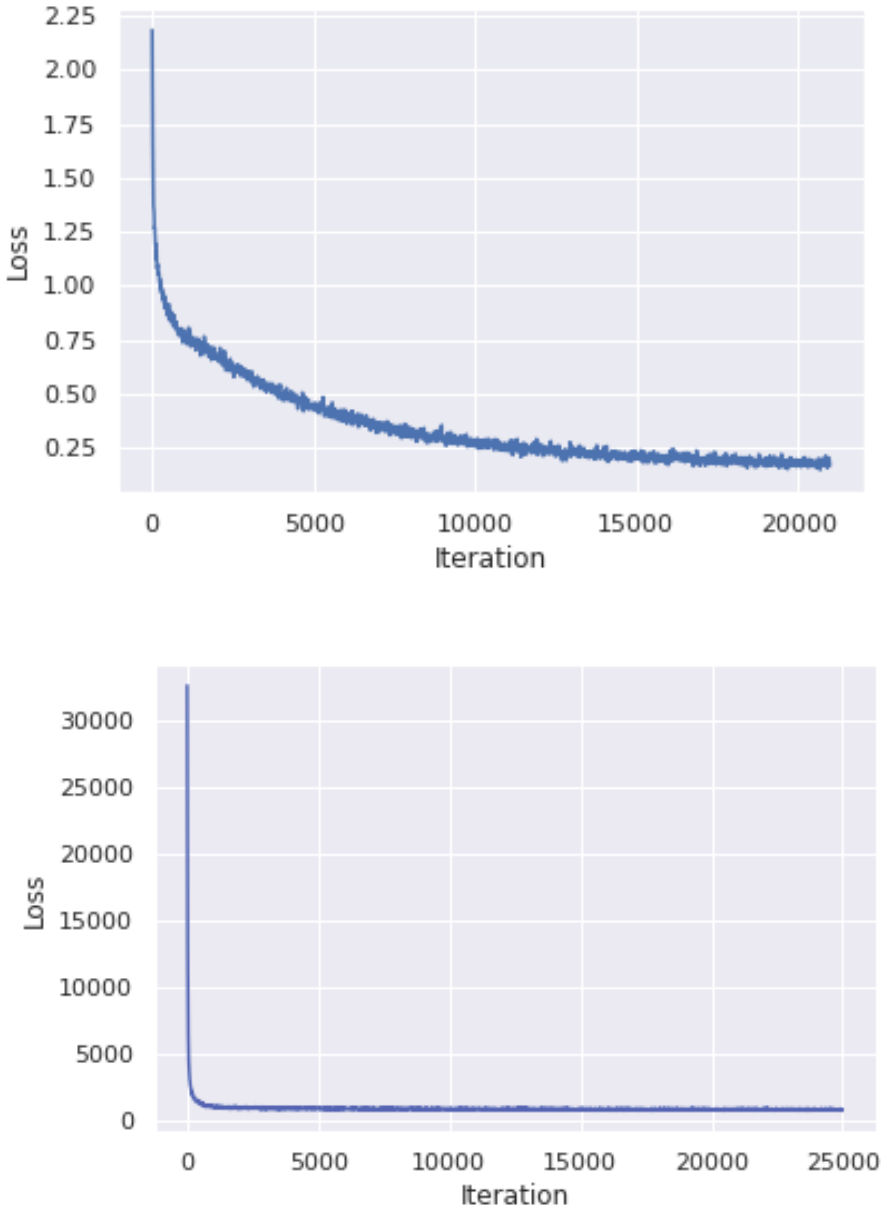


Figure S3: (**Above**) Training loss of the captioning network plotted against number of iterations passed. (**Below**) Training loss of the RHVAE network plotted against number of iterations passed.

6.3 Direct Generation

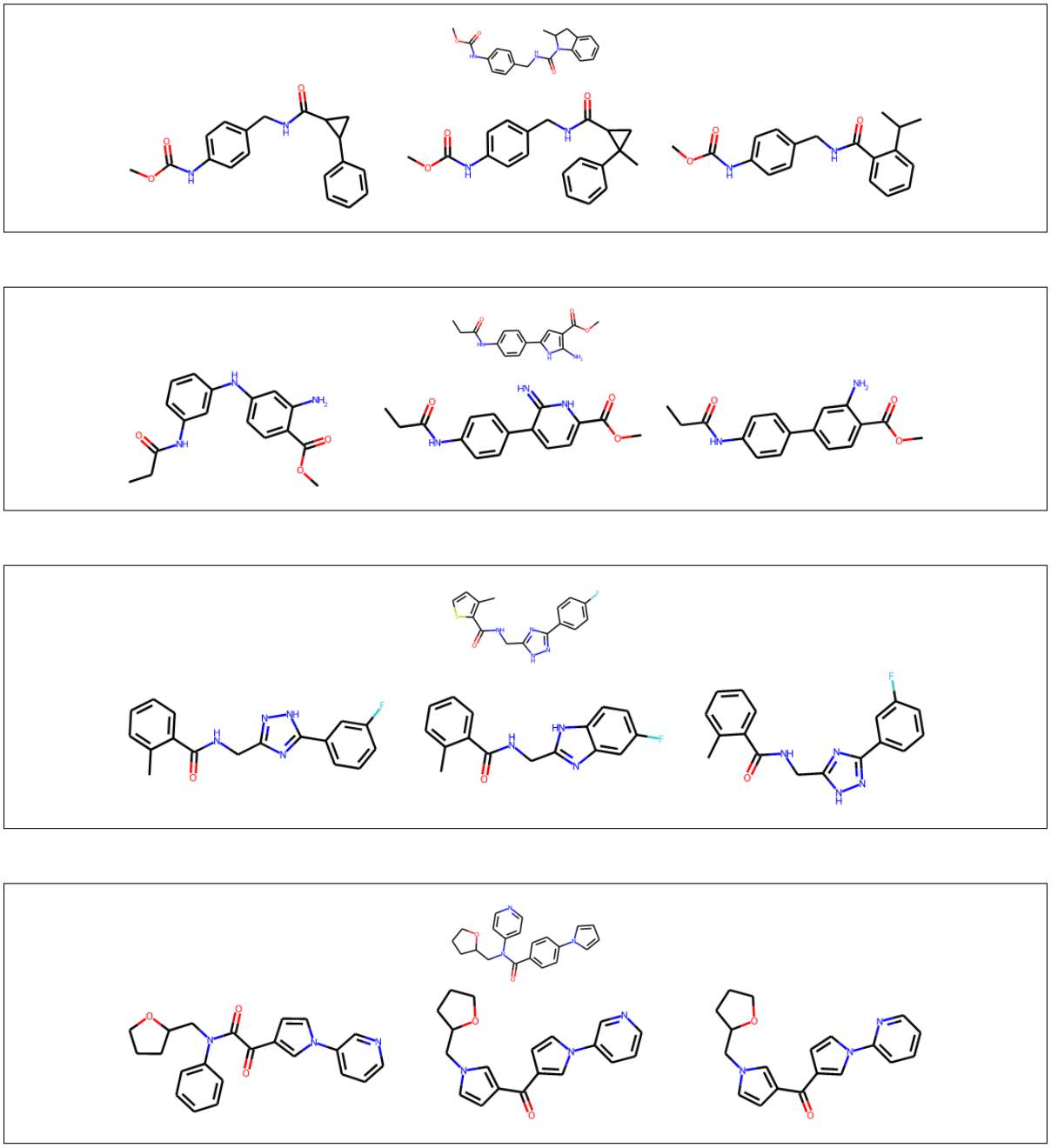


Figure S4: Examples of Direct Generation where seed molecule is shown above and the generated molecules are shown below it.

6.4 Autoencoded Generation

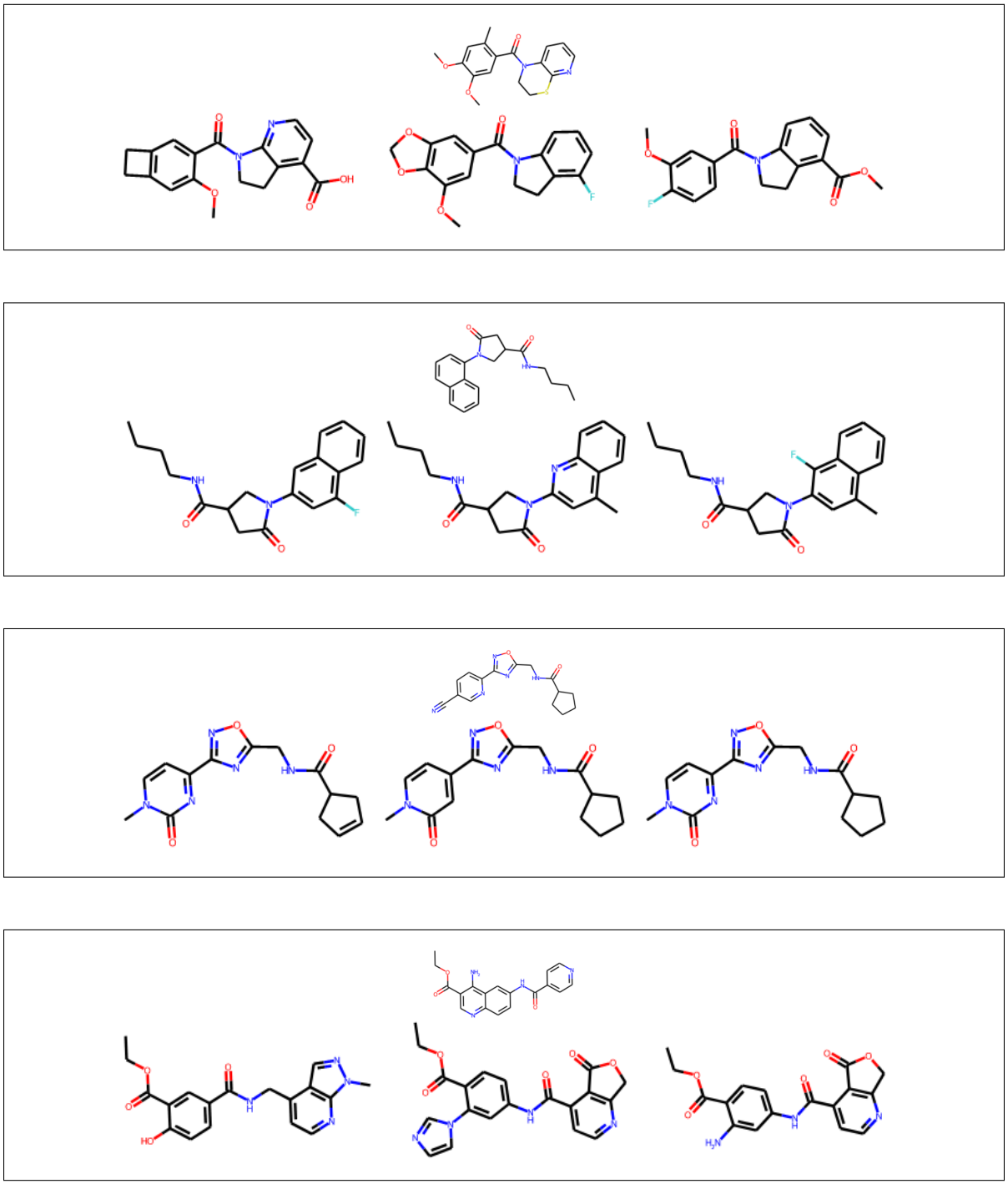


Figure S5: Examples of Autoencoded Generation where seed molecule is shown above and the generated molecules are shown below it.

6.5 Sampled Generation

6.5.1 Hamiltonian Monte Carlo

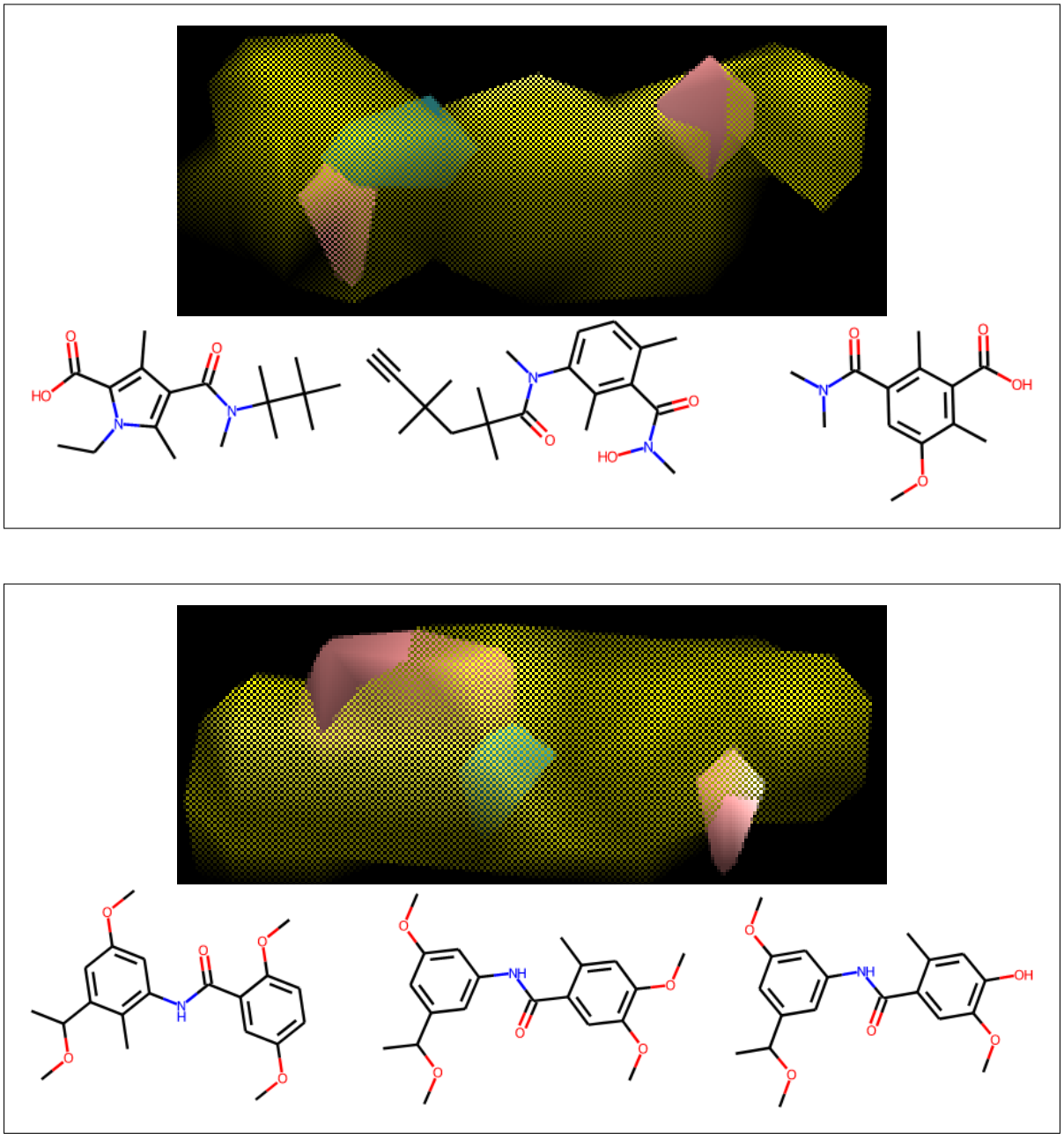


Figure S6: Examples of compounds generated through HMC sampling where the sampled grid (Occupancy, H-bond acceptor and H-bond donor channels in yellow, pink and blue respectively.) is shown above and the generated molecules are shown below it.

6.5.2 Random Walk Monte Carlo

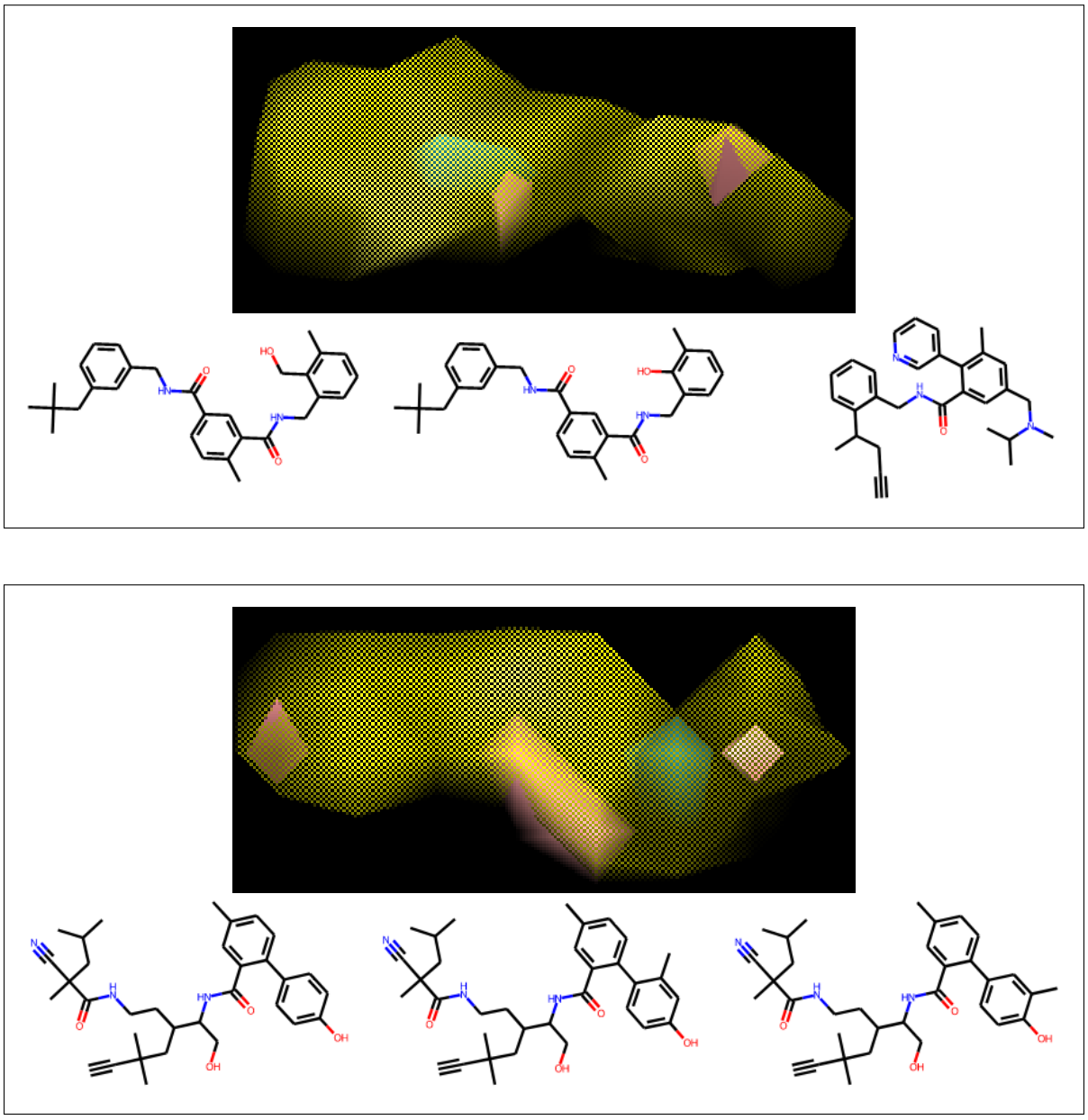


Figure S7: Examples of compounds generated through Random Walk Monte Carlo sampling where the sampled grid (Occupancy, H-bond acceptor and H-bond donor channels in yellow, pink and blue respectively.) is shown above and the generated molecules are shown below it.

6.5.3 Prior

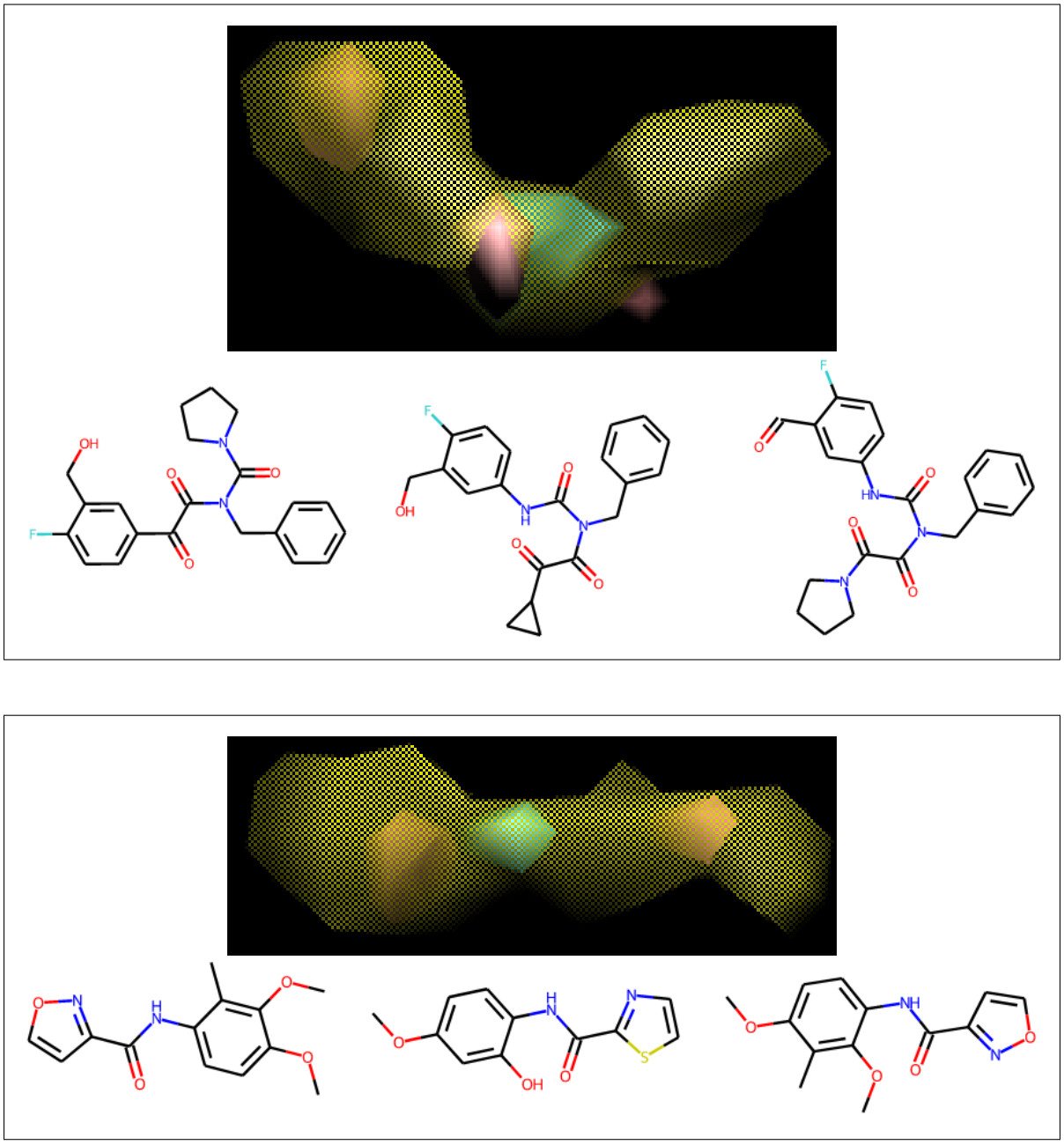


Figure S8: Examples of compounds generated through prior sampling where the sampled grid (Occupancy, H-bond acceptor and H-bond donor channels in yellow, pink and blue respectively.) is shown above and the generated molecules are shown below it.

Acknowledgement

We gratefully acknowledge the funding from SERC/DST India (EMR/2017/000373). We also acknowledge technical help of Dr. Semparithi Aravandian of High Performance Computing center at IIITH.

Software Availibility and training data

The source for running the RHVAE and the captioning network can be found at the following git repository: `https://github.com/shikhar2333/3-way-de-novo-generation`. The training data is available on the Zenodo Website: `https://zenodo.org/record/7007889`.

Related Publications

Shikhar Shasya, Shubham Sharma and Prabhakar Bhimalapuram. **Generative schemes for drug design with shape captioning**. Paper accepted in *Journal of Chemical Sciences* on 31st May, 2023.

Bibliography

[1] Walters, W P; Stahl, M T; Murcko, M A 1998 Virtual screening: an overview *Drug Discovery Today* ***3***(4), 160

[2] Sherstinsky, A 2020 Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network *Physica D: Nonlinear Phenomena* ***404***, 132306

[3] Goodfellow, I; Pouget-Abadie, J; Mirza, M; Xu, B; Warde-Farley, D; Ozair, S; Courville, A; Bengio, Y 2020 Generative adversarial networks *Communications of the ACM* ***63***(11), 139

[4] Kingma, D P; Welling, M 2014 Auto-Encoding Variational Bayes *Statistics* ***1050***, 1

[5] Segler, M H; Kogej, T; Tyrchan, C; Waller, M P 2018 Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks *ACS Central Science* ***4***(1), 120

[6] Bjerrum, E J; Threlfall, R 2017 Molecular generation with recurrent neural networks (RNNs) *arXiv preprint arXiv:170504612* `https://arxivorg/abs/170504612`, accessed on: 28/09/22

[7] Kusner, M J; Paige, B; Hernández-Lobato, J M 2017 Grammar Variational Autoencoder In *Proceedings of the 34th International Conference on Machine Learning*, Vol **70**; Precup, D; Teh, Y W, Eds; International Convention Centre, Sydney, Australia

[8] Dai, H; Tian, Y; Dai, B; Skiena, S; Song, L 2018 Syntax-directed variational autoencoder for structured data *arXiv preprint arXiv:180208786* `https://arxivorg/abs/180208786`, accessed on: 28/09/22

[9] Grisoni, F; Moret, M; Lingwood, R; Schneider, G 2020 Bidirectional molecule generation with recurrent neural networks *Journal of chemical information and modeling* ***60***(3), 1175

[10] Gómez-Bombarelli, R; Wei, J N; Duvenaud, D; Hernández-Lobato, J M; Sánchez-Lengeling, B; Sheberla, D; Aguilera-Iparraguirre, J; Hirzel, T D; Adams, R P; Aspuru-Guzik, A 2018 Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules *ACS Central Science* ***4***(2), 268

[11] Yan, C; Wang, S; Yang, J; Xu, T; Huang, J 2020 Re-balancing variational autoencoder loss for molecule sequence generation In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 1-7; Virtual Event USA, 2020

[12] Lim, J; Ryu, S; Kim, J W; Kim, W Y 2018 Molecular generative model based on conditional variational autoencoder for de novo molecular design *Journal of cheminformatics* ***10***(1), 1

[13] Jin, W; Barzilay, R; Jaakkola, T 2018 Junction tree variational autoencoder for molecular graph generation In *Proceedings of the 35th International Conference on Machine Learning*, Vol **80**; Stockholmsmässan, Stockholm Sweden, 2018

[14] Simonovsky, M; Komodakis, N 2018 Graphvae: Towards generation of small graphs using variational autoencoders In *International conference on artificial neural networks*; Rhodes, Greece, 2018

[15] Ma, C; Zhang, X 2021 GF-VAE: a flow-based variational autoencoder for molecule generation In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* 1181-1190; Queensland, Australia, 2021

[16] Richards, R J; Groener, A M 2022 Conditional β-VAE for De Novo Molecular Generation *arXiv preprint arXiv:2205.01592* `https://arxiv.org/abs/2205.01592`, accessed on 26/04/23

[17] De Cao, N; Kipf, T 2018 MolGAN: An implicit generative model for small molecular graphs *arXiv preprint arXiv:1805.11973* `https://arxiv.org/abs/1805.11973`, accessed on 26/04/23

[18] Blanchard, A E; Stanley, C; Bhowmik, D 2021 Using GANs with adaptive training data to search for new molecules *Journal of cheminformatics*, ***13***(1), 1-8

[19] Guimaraes, G L; Sanchez-Lengeling, B; Outeiral, C; Farias, P L C; Aspuru-Guzik, A *arXiv preprint arXiv:170510843* 2017 Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models `https://arxivorg/abs/170510843`, accessed on: 28/09/22

[20] Putin, E; Asadulaev, A; Ivanenkov, Y; Aladinskiy, V; Sanchez-Lengeling, B; Aspuru-Guzik, A; Zhavoronkov, A 2018 Reinforced adversarial neural computer for de novo molecular design *Journal of Chemical Information and Modeling* ***58***(6), 1194+

[21] Putin, E; Asadulaev, A; Vanhaelen, Q; Ivanenkov, Y; Aladinskaya, A V; Aliper, A; Zhavoronkov, A 2018 Adversarial threshold neural computer for molecular de novo design *Molecular Pharmaceutics* ***15***(10), 4386

[22] Sanchez-Lengeling, B; Outeiral, C; Guimaraes, G L; Aspuru-Guzik, A 2017 Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC) *ChemRxiv.org:5309668* `https://chemrxiv.org/engage/chemrxiv/article-details/60c73d91702a9beea7189bc2`, accessed on: 26/04/23

[23] Krenn, M; Häse, F; Nigam, A; Friederich, P; Aspuru-Guzik, A 2019 SELFIES: a robust representation of semantically constrained graphs with an example application in chemistry *arXiv preprint arXiv:190513741* https://arxivorg/abs/190513741v1, accessed on: 28/09/22

[24] Ghanbarpour, A; Lill, M A 2020 Seq2Mol: Automatic design of de novo molecules conditioned by the target protein sequences through deep neural networks *arXiv preprint arXiv:201015900* https://arxivorg/abs/201015900, accessed on: 28/09/22

[25] Schneuing, A; Du, Y; Harris, C; Jamasb, A; Igashov, I; Du, W; Blundell, T; Lió, P; Gomes, C; Welling, M; Bronstein, M 2022 Structure-based drug design with equivariant diffusion models *arXiv preprint arXiv:2210.13695* https://arxiv.org/abs/2210.13695, accessed on: 26/04/23

[26] Skalic, M; Jiménez, J; Sabbadin, D; De Fabritiis, G 2019 Shape-based generative modeling for de novo drug design *Journal of chemical information and modeling* ***59***(3), 1205

[27] Chadebec, C; Mantoux, C; Allassonnière, S 2020 Geometry-aware Hamiltonian variational auto-encoder *arXiv preprint arXiv:201011518* 2020 https://arxivorg/abs/201011518, accessed on: 28/09/22

[28] Wilt, C M; Thayer, J T; Ruml, W 2010 A comparison of greedy search algorithms In *Proceedings of the Third Annual Symposium on Combinatorial Search*; Stone Mountain Resort, Atlanta, Georgia, USA, 2010

[29] Sterling, T; Irwin, J J 2015 ZINC 15–ligand discovery for everyone *Journal of Chemical Information and Modeling* 2015 ***55***(11), 2324

[30] Landrum, G 2013 RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling https://wwwrdkitorg, accessed on: 28/09/22

[31] Halgren, T A; Nachbar, R B 1996 Merck molecular force field. IV. conformational energies and geometries for MMFF94 *Journal of Computational Chemistry* ***17***(5-6), 587

[32] Doerr, S; Harvey, M; Noé, F; De Fabritiis, G 2016 HTMD: high-throughput molecular dynamics for molecular discovery *Journal of chemical theory and computation* ***12***(4), 1845

[33] Xu, K; Ba, J; Kiros, R; Cho, K; Courville, A; Salakhudinov, R; Zemel, R; Bengio, Y 2015 Show, attend and tell: Neural image caption generation with visual attention In *Proceedings of the 32nd International Conference on Machine Learning*, Vol **37**; Lille, France, 2015

[34] Simonyan, K; Zisserman, A 2014 Very deep convolutional networks for large-scale image recognition *arXiv preprint arXiv:1409.1556* https://arxiv.org/abs/1409.1556, accessed on: 28/09/22

[35] Kingma, D P; Ba, J 2014 Adam: A Method for Stochastic Optimization *arXiv preprint arXiv:1412.6980* https://arxiv.org/abs/1412.6980, accessed on: 28/09/22

[36] Paszke, A; Gross, S; Chintala, S; Chanan, G; Yang, E; DeVito, Z; Lin, Z; Desmaison, A; Antiga, L; Lerer, A 2017 Automatic differentiation in PyTorch *NIPS 2017 Workshop on Autodiff* https://openreviewnet/forum?id=BJJsrmfCZ, accessed on: 28/09/22

[37] Chadebec, C; Thibeau-Sutre, E; Burgos, N; Allassonnière, S 2022 Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder *IEEE Transactions on Pattern Analysis and Machine Intelligence* (01), 1

[38] Burda, Y; Grosse, R; Salakhutdinov, R 2015 Importance weighted autoencoders *arXiv preprint arXiv:150900519* https://arxivorg/abs/150900519, accessed on: 28/09/22

[39] Tucker, G; Lawson, D; Gu, S; Maddison, C J 2018 Doubly reparameterized gradient estimators for monte carlo objectives *arXiv preprint arXiv:181004152* https://arxivorg/abs/181004152, accessed on: 28/09/22

[40] Mysinger, M M; Carchia, M; Irwin, J J; Shoichet, B K 2012 Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking *Journal of medicinal chemistry* ***55***(14), 6582

[41] Wildman, S A; Crippen, G M 1999 Prediction of Physicochemical Parameters by Atomic Contributions *Journal of Chemical Information and Computer Sciences* ***39***, 868

[42] Bickerton, G R; Paolini, G V; Besnard, J; Muresan, S; Hopkins, A L 2012 Quantifying the chemical beauty of drugs *Nature chemistry* ***4***(2), 90

[43] Skalic, M; Sabbadin, D; Sattarov, B; Sciabola, S; De Fabritiis, G 2019 From Target to Drug: Generative Modeling for the Multimodal Structure-Based Ligand Design *Molecular Pharmaceutics* ***16***(10), 4282

[44] Ragoza, M; Masuda, T; Koes, D R 2022 Generating 3D molecules conditional on receptor binding sites with deep generative models *Chemical Science* ***13***, 2701

[45] Winter, R; Montanari, F; Steffen, A; Briem, H; Noé, F; Clevert, D A 2019 Efficient multi-objective molecular optimization in a continuous latent space *Chemical science* ***10***(34), 8016-8024

[46] Bagal, V; Aggarwal, R; Vinod, P K; Priyakumar, U D 2021 MolGPT: molecular generation using a transformer-decoder model *Journal of Chemical Information and Modeling* ***62***(9), 2064-2076

[47] Caterini, A L; Doucet, A; Sejdinovic, D 2018 Hamiltonian variational auto-encoder *Advances in Neural Information Processing Systems* ***31***, 8167

[48] Girolami, M; Calderhead, B; Chin, S A 2009 Riemannian manifold hamiltonian monte carlo *arXiv preprint arXiv:0907.1100* `https://arxiv.org/abs/0907.1100`, accessed on: 22/03/23

[49] Girolami, M; Calderhead, B 2011 Riemann manifold langevin and hamiltonian monte carlo methods *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, ***73***(2), 123–214

[50] Hairer, E; Lubich, C; Wanner, G 2006 Structure-preserving algorithms for ordinary differential equations *Geometric numerical integration* ***31***

[51] Leimkuhler, B; Reich, S 2004 Simulating hamiltonian dynamics *Cambridge university press* ***14***

[52] Neal, R M 2005 Hamiltonian importance sampling *talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics*

[53] Salimans, T; Kingma, D; Welling, M 2015 Markov chain monte carlo and variational inference: Bridging the gap *International conference on machine learning*, 1218–1226