DEVELOPMENT OF AN INTEGRATED MACHINE LEARNING APPROACH FOR ANTI-CANCER DRUG DISCOVERY USING VARIATIONAL AUTO-ENCODER

A Thesis

by

MAMOON MASUD

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Byung Jun Yoon
Committee Members,	Hangue Park
	Limei Tian
	Jun Zou
Head of Department,	Miroslav M. Begovic

December 2021

Major Subject: Electrical Engineering

Copyright 2021 Mamoon Masud

ABSTRACT

Cancer is one of the leading causes of death across the world and accounted for almost 10 million deaths in the year 2020 alone. Anti-cancer drug discovery & response prediction is an arduous and time-consuming process. This work investigates the use of generative models to predict anti-cancer drug response and facilitate the discovery of new drugs, utilizing chemical structure of drugs, gene expression data and response data of anti-cancer drugs.

Autoencoders are a type of neural network that are trained to learn a lower dimensional representation of a high dimensional data, while a Variational Autoencoder (VAE) is trained to model the data as a distribution over the latent space. The proposed approach models the anti-cancer drug molecular data using a rectified junction tree VAE (JTVAE) model while the cancer cell lines' gene expression data is encoded using a VAE. A feed-forward artificial neural network takes in the concatenated encoded latent vectors of gene expression profile for a cancer cell line and latent vectors of drugs to generate a final prediction, represented by the $ln(IC_{50})$ score. The model was trained on three different datasets, with one set consisting of breast cancer cell lines, another of central nervous system cell lines and one consisting of pan-cancer cell lines. Testing on pan-cancer cell lines produced a high average coefficient of determination (R2 = 0.875), and it was the best performing model.

Additionally, this work has investigated the optimizing the latent space using weighted retraining. This was done to improve the sample efficiency. The technique adopted was sample efficient and weighted data points based on the blood brain barrier permeability. A Message Passing Neural Network (MPNN) based predictor was trained to predict blood brain barrier permeability score in the input data. After training the generative model (JTVAE), the objective function was optimized over the learned latent space using a surrogate model. This decoupled the tasks of training the generative model, and optimizing it. Weighting the data distribution, and periodic retraining of the generative model improved the prediction score of Blood Brain Barrier permeability, which is an important factor for drugs that are targeted at Central Nervous Systems tumours.

DEDICATION

To my parents, and grandparents.

ACKNOWLEDGMENTS

I would like to thank my Advisor Dr. Byung-Jun Yoon for trusting my enthusiasm and providing me with the opportunity to work on generational models for drug discovery, despite not having any experience in machine learning. I'd also like to thank him for his constant guidance and support throughout the duration of this work.

I am thankful of my committee members, Dr. Hangue Park, Dr. Limei Tian, and Dr. Jun Zou for agreeing to be on my committee and their timely responses to all my requests.

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing. These resources were fundamental to the completion of this work.

I would also like to thank Ms. Kelly Wellman at the Office of Sustainability, and Dr. Sebastian Hoyos with whom I worked as a Graduate Assistant. I'd also like to thank Ms. Katherine Bryan of ECEN Graduate Office for her quick responses to all my queries, and her help and support in dealing with logistical issues throughout my degree.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Byung Jun Yoon [advisor], Dr. Hangue Park & Dr. Jun Zou of the Department of Electrical & Computer Engineering and Dr. Limei Tian of the Department of Biomedical Engineering. Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing (HPRC).

All the work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by Graduate Assistantships from Texas A&M University.

NOMENCLATURE

BBB	Blood Brain Barrier
B/CS	Bryan and College Station
CCLE	Cancer Cell Line Encyclopedia
CDR	Cancer Drug Response
CGC	Cancer Genomic Census
DNN	Deep Neural Network
ECEN	Electrical and Computer Engineering
GAN	Generative Adversarial Network
GAPS	Graduate and Professional School at Texas A&M University
GDSC	Genomics of Drug Sensitivity in Cancer
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
GVAE	Grammar Variational Autoencoder
HPRC	High-Performance Research Computing
ICA	Independent Component Analysis
JTVAE	Junction Tree Variational Autoencoder
KL	Kullback-Leibler
LSTM	Long Short-term Memory
MNIST	Modified National Institute of Standards and Technology database
MPNN	Message Passing Neural Network
PCA	Principal Component Analysis

ReLU	Rectified Linear Unit
SMILES	Simplified Molecular-input Line-entry System
SVM	Support Vector Machine
TCGA	The Cancer Genome Atlas
TAMU	Texas A&M University
t-SNE	t-distributed stochastic neighbor embedding
tpm	transcriptome per million
VAE	Variational Autoencoder
WHO	World Health Organization

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

Cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths worldwide in the year 2020 [9]. Out of these, approximately 70% occur in low- and middle-income countries, where the lack of access to diagnosis is common [10]. The lack of comprehensive treatment is also disproportionate, with availability in more than 90% of high-income countries, but in less than 15% of low-income countries. This has a consequential economic impact, with estimates of annual economic cost due to cancer in 2010 alone being at US $1.6 trillion [11].

There are multiple treatment options for cancer, depending on the type of cancer and the progression stage. The figure 1.1 below shows the different treatment options available for cancer patients. Though multiple treatment options are available, and cancer drugs are widely used as a primary treatment option, precision medicine are increasingly being used to personalize the treatment of cancer. Unlike chemotherapy, understanding of the genetics is used to develop these precision medicines such that cancer cells are targeted selectively, and sparing healthy cells, ideally. This can lead to more effective and less toxic treatment option. Therefore, predicting anti-cancer drug response for precision medicine is becoming increasingly important, however, discovery of anti-cancer drugs is both time and cost-intensive, and tailoring cancer therapies for individual patients is an arduous task. This work proposes a generative model to predict anti-cancer drug response, and propose new drugs that are potentially effective against a particular cell line to speed up drug development.

Figure 1.1: Cancer Treatment Options. Reprinted from [1]

Cancer is caused by genetic changes in the DNA of a cell that causes them to divide and grow excessively. These genetic changes vary quite a lot by cancer types and even within a single type of cancer. For example, all the patients with breast cancer cannot be treated with the same medicine. The patients can be divided into subgroups, and based on the genomic profile of the tumor, the treatment can be tailored that is the most effective. Despite having great clinical success using this type of precision medicine approach, most patients today, do not receive this due to logistical and financial constraints.

The target of this work is the problem known as Cancer Drug Response (CDR) prediction. CDR describes how a cancer patient responds to a specific drug and it measured by different metrics, such as $ln(IC_{50})$, the half-maximal inhibitory concentration, and Blood Brain Barrier permeability (BBB). Identification of CDR is therefore crucial for personalized cancer therapy as CDR may vary in cancer patients due to heterogeneity. In addition to this, we still lack effective drugs against some types of cancers. In addition to this, millions of compounds are there, out of which only a minority can be considered as drugs. This makes finding effective drugs challenging. It has been shown that for CDR, the compound structure is an important determining factor, and as discussed in the later sections, the prediction models are designed to use the molecular structure.

In the past, the work done in predicting CDR can be divided into broad categories. One was focused using network-driven models, such as [12]. These methods constructed a similarity-based model & assigned sensitivity of a known drug to new drug in case of structural similarities. The limitations of such an approach being low scalability & efficiency. The other broad category consisted of predicting CDR using machine learning methods. Such methods used large-scale profiles of either drug or cancer cell lines to predict drug response. [13] used gene expression data to predict drug response, while others predicted drug efficacy by combining cancer genomic data and drug molecular embeddings in Deep Neural Networks (DNNs) [14, 15, 16, 17, 18]. Chiu et al. [18] proposed using the mutation and expression profiles of a cancer cell or a tumor for predicting drug response, with the deep learning model consisting of three DNNs, including a mutation encoder, pre-trained on a large pan-cancer dataset (TCGA), a pre-trained expression encoder and a drug response predictor network that integrated the first two sub-networks. Most of the work referred above focused on analysing the original data, which could inherently influence the prediction performance, due to the presence of a large amount of redundant information.

A large number of ways exist to perform feature extraction on drug molecular data and cancer genomic data, including supervised and supervised learning methods. Supervised learning algorithms, such as Random Forest, can be used to find feature importance in the gene expression data, and aid in filtering out the most important genes. However, such supervised learning methods can't be used for unlabeled data, such as the Cancer Cell Line Encyclopedia (CCLE) datasets that are being used for this work. Manifold learning based t-distributed stochastic neighbor embedding (t-SNE), Principal Component Analysis (PCA), Independent Component Analysis (ICA), and some other unsupervised learning algorithms are used commonly for analysis of high-dimensional unlabeled medical data, but the prime purpose is for 2D visualization for majority of the cases. In addition to this, there might be significant information loss when a lower-dimensional representation is achieved. Also, for higher dimensions, these methods do not perform well, and visualizing to intuitively analyze performance is also not possible [19].

1.1 Our Approach

This work is focused on using the anti-cancer drug molecular data as well as gene expression data to, predict drug response on cancer cell lines using Generative Models, or Variational autoencoders (VAE) [20]. Using VAE's enables us to extract features from a large amount of unlabelled genomic and molecular data. VAE's have been successfully used for unsupervised learning tasks involving complex probability distributions. They differ from autoencoders, as discussed below, but for the purpose of this work, VAEs have the ability to capture probabilistic distribution of latent features, providing with a more comprehensive analysis of genomic data. That makes predicting anti-cancer drug response for specific cancer cell lines easier. Junction Tree VAE (JTVAE) model [3] is used to extract the low dimensional features of the anti-cancer drugs. The JTVAE transforms a drug's molecular graph into junction trees, exploiting valid subgraphs as components. This ensures chemically valid molecules are generated, a problem encountered if a traditional VAE model is used. Once the two VAE's are trained, multi-layer perceptron is used to extract features in order to product the final drug response, represented by $ln(IC_{50})$ value, indicating the efficacy of the drug against the cancer cell line under consideration. Using a generative model has the added benefit that unlike past studies [21, 22, 23], the model isn't confined to specific drugs, and can taken any organic compound as input to predict it's efficacy.

Other researchers have worked on generative models for molecules [5, 6, 24], or [25] focused on generating effective drugs with Generative Adversarial Networks (GANs) and VAE, however, JTVAE outperforms these in reconstructing valid molecules, and in the case of [25], can also predict the drug response on different cancer cell lines. In fact, JTVAE generated drug compounds are always valid, proving it to be quite powerful. Random sampling of encoded features of drugs can be performed along with the well-performing features decoded by JTVAE, generating a large number of valid drug compounds, that are also effective for cancer therapy.

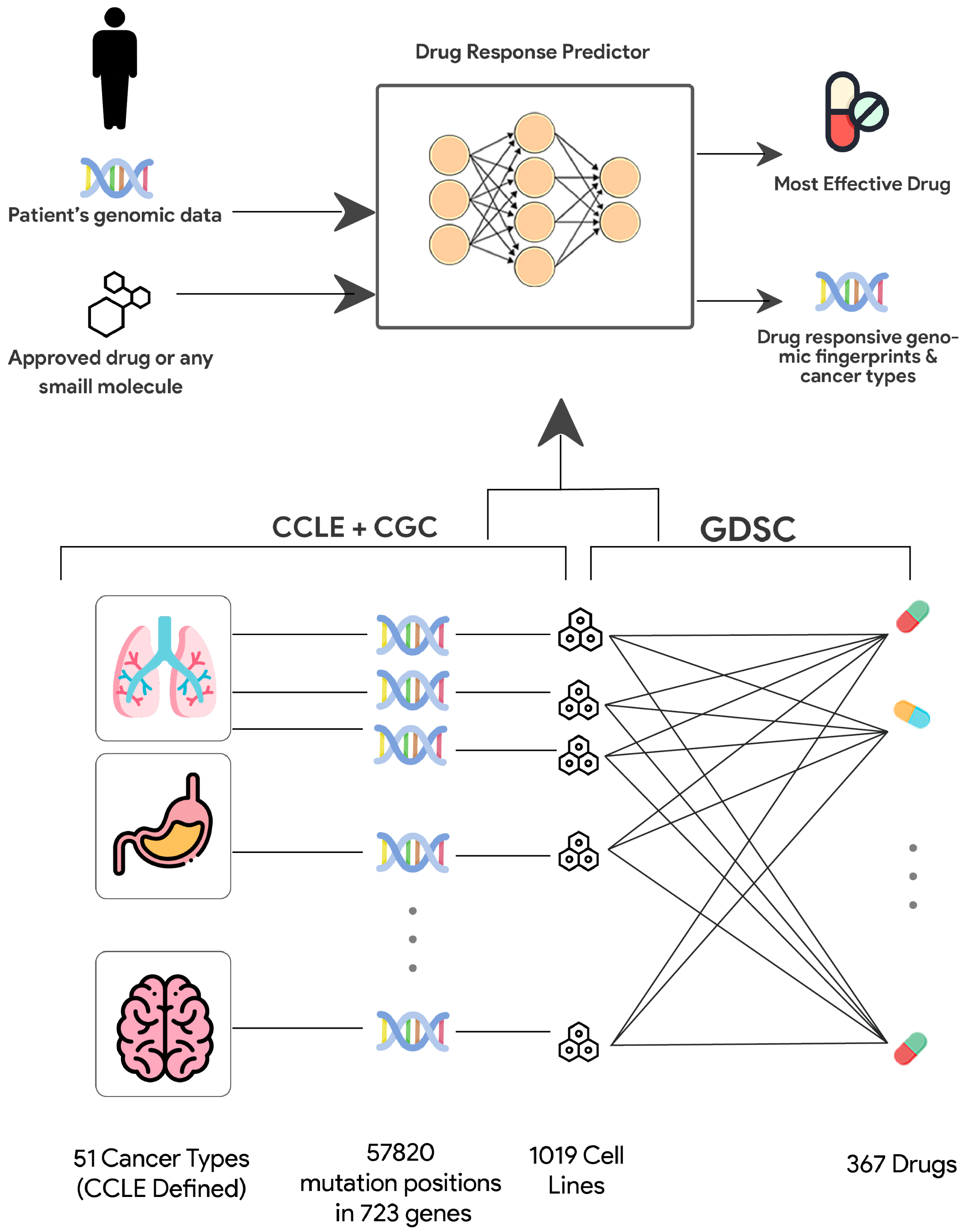

Drug Response Predictor
Patient's genomic data
Approved drug or any smaill molecule
Most Effective Drug
Drug responsive genomic fingerprints & cancer types
CCLE + CGC
GDSC
51 Cancer Types (CCLE Defined)
57820 mutation positions in 723 genes
1019 Cell Lines
367 Drugs

Figure 1.2: Overview of the proposed methodology

The figure 1.2 shows an overview of the proposed methodology discussed above. Gene expression data of 1019 cancer cell lines, and molecular data of organic compounds is used to train generative models to get lower dimensional representations. Once trained, the gene expression data, and drug response data against those cell lines is used to train the property predictor. After the model is trained, a patient's genomic data and an approved drug can be used to predict it's response for a specific cell line in the patient. It can also be used to predict potential new drugs that are successful against a specific cell line, based on the property of interest ($ln(IC_{50})$).

Over the last few years, machine learning has shown promising results for problems that could be framed as optimization, such as text generation [26] and molecular & material design [27]. However, the use of machine learning on problems with limited data and structured input spaces is still not feasible. This challenge has been tackled by [28] in a recent paper, in an approach referred to as latent space optimization (LSO), in which, first a generative model is trained to generate lower dimensional and continuous representation of the input data, and then the objective function is optimized over the learned latent space using a surrogate model, thereby effectively decoupling the tasks of training the generative model and the optimization problem. In the paper, [28] Tripp et al. have proposed weighting the data distribution using a property of interest, and periodic retraining of the generative model to decouple the two tasks.

2. MODEL ARCHITECTURE

2.1 Autoencoders

First introduced in 1986 by [29], Autoencoders are neural networks that are trained to reconstruct their input, and in the process, learn a lower-dimensional representation of the data in an unsupervised manner. Formally defined by [30], the problem in the Autoencoder is to learn the functions $A : \mathbb{R}^n \rightarrow \mathbb{R}^p$ (encoder) and *B* : $\mathbb{R}^p \rightarrow \mathbb{R}^n$ (decoder) that satisfies:

$$argmin_{A,B} E[\triangle(\mathbf{x}, B \circ A(\mathbf{x})] \tag{2.1}$$

where *E* is the expectation over the distribution x, and $\triangle$ represents the reconstruction loss, usually set to as the ℓ_2-norm. An illustration of the autoencoder model is provided in Figure 2.1, using the Modified National Institute of Standards and Technology (MNIST) dataset [31] as an example.

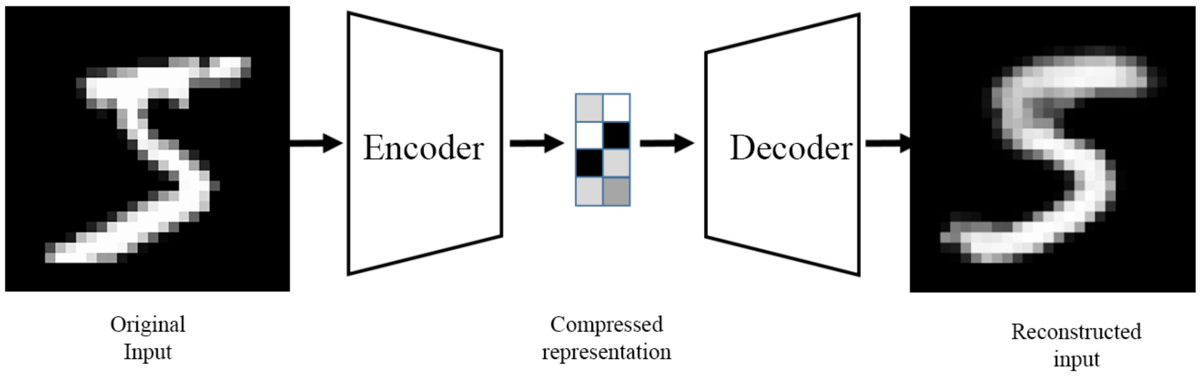


Figure 2.1: An example of an autoencoder, with MNIST data used as sample input

Although not a requirement, but for most popular autoencoders, A and B are neural networks. If A and B are linear operators, we get a linear autoencoder [32], and for linear autoencoder in which we also drop the non-linear operators, we will get the same latent representation as for PCA, and thus, the autoencoder is rather a generalization of PCA, learning a non-linear manifold instead of finding a low-dimensional hyperplane along which the data lies. Autoencoders can either be trained gradually layer by layer or end-to-end. In case of gradually training layer by layer, they are stacked together to achieve a deeper encoder.

Even though auto-encoders have been used widely for unsupervised learning problems, and extracting low dimensional features, however, there are a few downsides associated with them. Autoencoders aren't very robust, and a slight variance in the encoded vector might sometime lead to a huge variability in the reconstructed data.

2.1.1 Variational Autoencoders

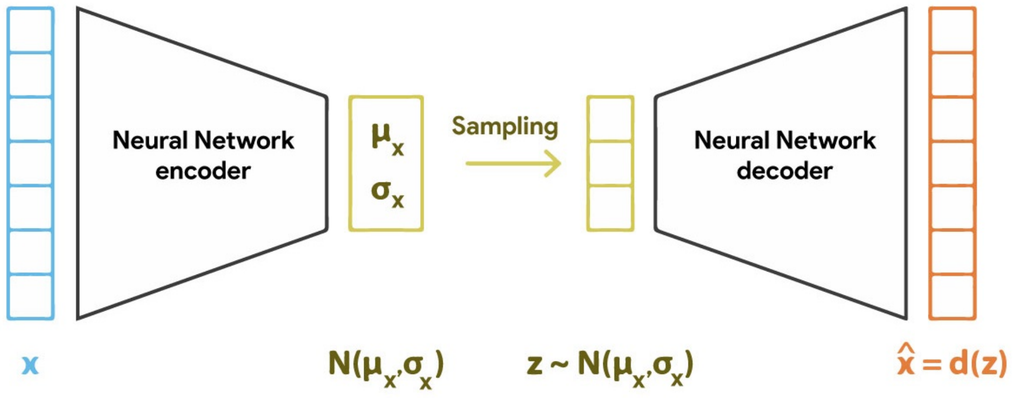


Figure 2.2: Overview of a Variational Autoencoder

Variational Autoencoders are considered to have a significant improvement over the representation capabilities of Autoencoders [20]. VAE is a generative model, following Variational Bayes

(VB) inference [33], that attempt to learn the data generation through the underlying probabilistic distribution with an inference network (encoder) and a generative network (decoder). An overview of a VAE is depicted in the Figure 2.2. Given an observation of a dataset $\mathbf{X} = \{x_i\}_{i=1}^{N}$ of $\mathbf{V}$ i.i.d samples, assuming a generative model for each datum $\mathbf{x}_i$, that is conditioned on an unobserved random latent variable $\mathbf{z}_i$. The parameters governing the generative distribution are represented by θ. We can also think of this generative models as being equivalent to a probabilistic decoder. Governed by the parameters ϕ, we assume an approximate posterior distribution over the latent variable $\mathbf{z}_i$, given a datum $\mathbf{x}_i$, equivalent to a probabilistic encoder. We also assume a prior distribution (a normal Gaussian distribution) of latent variables, denoted by $p(\mathbf{z};\theta)$, and the posterior is also expected to follow a Gaussian distribution. The figure 2.3 below depicts this relationship.

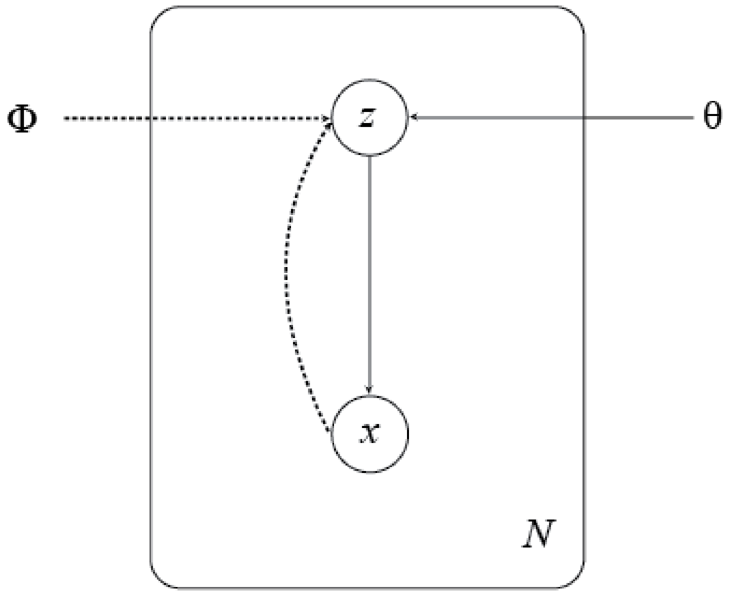


Figure 2.3: A Graphical representation of a VAE. Reprinted from [2]

The marginal log-likelihood is expressed as a sum of over the individual data points:

$$\log p_\theta(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} \log p_\theta(x_i) \tag{2.2}$$

And each data point can be written as a sum of two terms as:

$$\log p_\theta(x_i) = D_{KL}(q_\phi(z|x_i)||p_\theta(z|x_i)) + \mathcal{L}(\theta, \phi; x_i) \tag{2.3}$$

where D_{KL} is the Kullback-Leibler divergence of the approximate recognition model from the true posterior, and the rest of the expression represents the variational lower bound on the marginal likelihood, and given the dataset $\mathbf{X} = \{\mathbf{x}\}_{i=1}^{N}$ with N data points,the estimate for the marginal likelihood lower-bound of the full dataset $\mathcal{L}(\theta, \phi; \mathbf{X})$ using a mini-batch $X^M = x_i$ of size *M* is given by:

$$\mathcal{L}(\theta, \phi; \mathbf{X}) \approx \tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M) = \frac{N}{M}\sum_{i=1}^{M}\mathcal{L}(\theta, \phi; \mathbf{x}_i) \tag{2.4}$$

In VAE, the gradients of $\tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M)$ are approximated using stochastic gradient descent and reparametrization trick. The reparametrization of variable z is done to enable back propagation, where $z \sim \mathcal{N}(\mu, \sigma^2)$.

2.1.2 Gene Expression VAE (geneVAE)

One of the two variational autoencoders used in this work is the gene expression variational autoencoder. The figure 2.4 gives an overview of the architecture of geneVAE. The model requires to extract latent vectors from the CCLE gene expression data, which is achieved by the geneVAE. The extracted latent vectors for drug response prediction. The encoder architecture consists of a single layer fully connected neural network. A batch-norm layer before activation helps to train it efficiently. The output of the activated layer is given by the following relationship:

$$\mu_g = f_1(\textbf{Batch-norm}((\mathbf{W}_1 h_1^T + b_1)) \tag{2.5}$$

where f_1 is the activation function, W_1 and b_1 are the weight matrix and bias vector respectively, and μ_g is the mean of the latent space, while the standard deviation σ_g is computed by another single-layer dense network with a similar architecture. The latent vector z_g is randomly sampled from $\mathcal{N}(\mu_g, \sigma_g)$. The decoder for the geneVAE consists of single dense layers, with the output

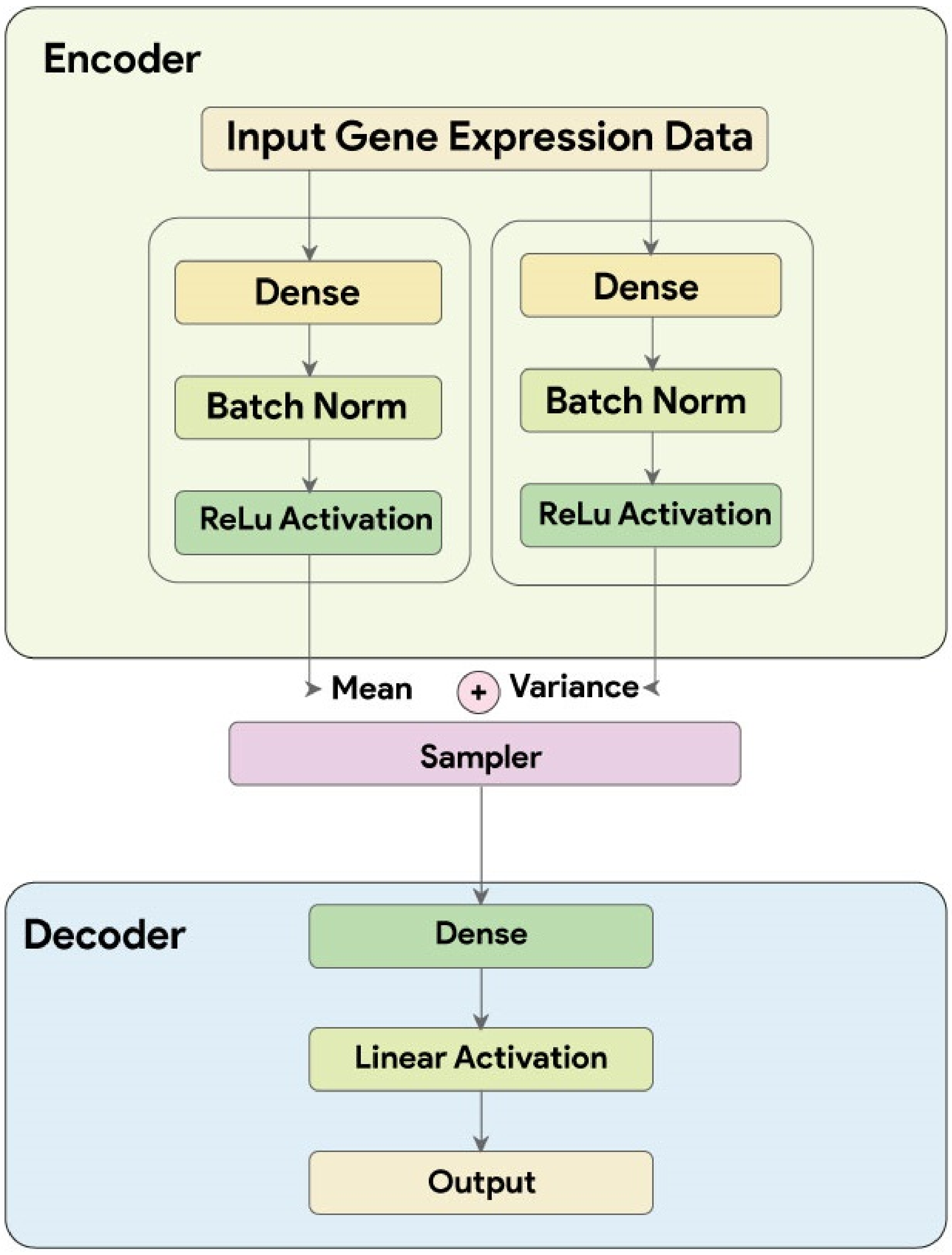


Figure 2.4: Model architecture of geneVAE

dimensions being the same as the input data. the decoded gene expression, $G^{'}$ is given be:

$$G^{'} = f_4(W_4^T z_g + b_4)) \tag{2.6}$$

The reconstruction loss $\mathcal{L}(G, G^{'})$ and Kullback-Leibler (KL) divergence loss $KL(q(z)||p(z|x))$ [20] are to be computed, where p and q are the posterior distribution and distribution of $z.q(z)$ respectively. The total loss of the geneVAE can be written as:

$$\mathcal{L}(p, q) = \mathcal{L}(G, G^{'}) + KL(q(z)||p(z|x)) \tag{2.7}$$

The aim being to reduce the total loss until it converges. Here, we take μ_g as encoded features instead of sampling the vectors from a Gaussian distribution.

2.1.3 Representation Learning on Graphs for Drug molecular features

Use of generative machine learning models has been on the rise recently for varied applications, ranging from generating realistic musical improvisations [34], to creating realistic pieces of artwork [35]. Over the past few years, interest has been gaining momentum towards training generative models to construct more complex, discrete data types, such as arithmetic expressions [36] and molecules [37]. Extensive work in deep learning has been done in exploring realm of molecular graph encoding [38, 39], the comparatively harder task of molecular graph generation from latent representation has not received the same level of attention.

In the area of drug design, past work includes treating the graph generation task as a string generation task [37], which was done in order to circumvent the direct generation of graphs. These kind of models used Simplified Molecular-input Line-entry System (SMILES) [40], a linear string notation for representing molecular structures. Although SMILES strings can be translated into graphs via deterministic mappings, such as using RD-Kit [41], an open-source cheminformatics tool, the design had limitations, including the fact that molecular similarity is not captured in the SMILES representation, and two molecules with similar chemical structures might have evidently different SMILES strings, as visible in the figure 2.5 below. Due to this, generative models, such as a VAE cannot learn smooth molecular embeddings. In addition to this, chemical properties such as molecular validity are easier to express on graphs as compared to linear SMILES representations. In graph based VAE models, such as Junction Tree VAE (JTVAE) [3], it is evident that representing molecules in graphs improves the generative modelling of valid chemical structures.

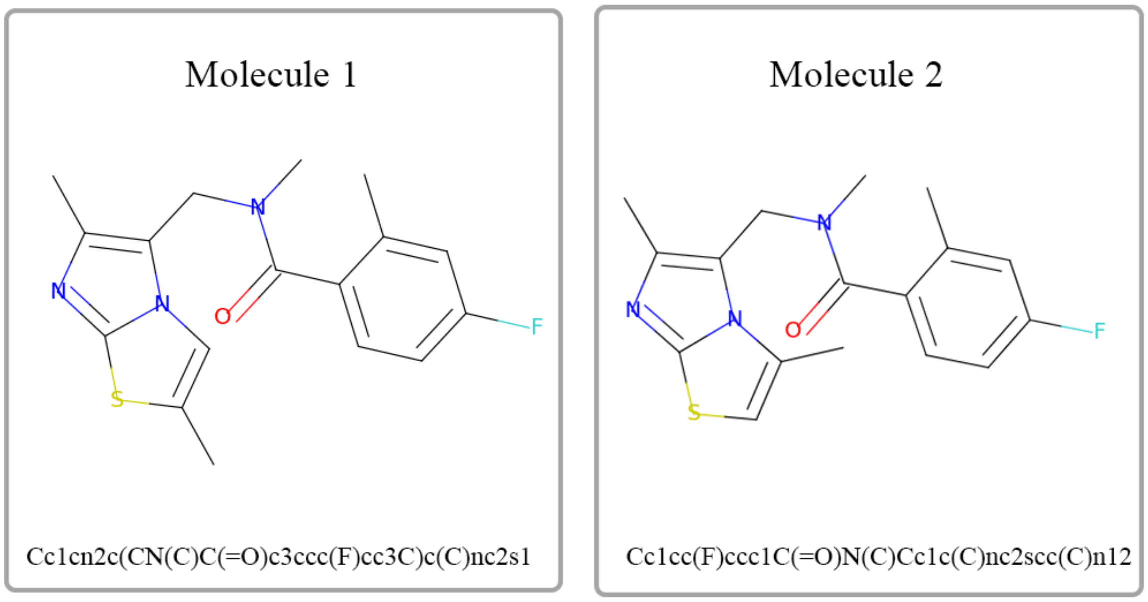

Molecule 1
Cc1cn2c(CN(C)C(=O)c3ccc(F)cc3C)c(C)nc2s1
Molecule 2
Cc1cc(F)ccc1C(=O)N(C)Cc1c(C)nc2scc(C)n12

Figure 2.5: Two similar molecules with strikingly different SMILES representation (plotted in matplotlib using RDKit). The edit distance between the two strings is 22

2.1.3.1 Junction Tree VAE

Other researchers have tried to represent the molecules in a more efficient way (e.g parse trees in Grammar VAE) to address the caveats of SMILES representation, such as [5], however, that still doesn't solve the problem of generating chemically invalid intermediaries. The reason being that generating molecular graph node by node [7] leads to generation of chemically invalid intermediaries, as depicted in Figure 2.8, with validation done after generation of the complete graph.

Method	Reconstruction	Validity
CVAE	44.6%	0.7%
GVAE	53.7%	7.2%
SD-VAE	76.2%	43.5%
GraphVAE	-	13.5%
Atom-by-Atom LSTM	-	89.2%
JT-VAE	76.7%	100.0%

Table 2.1: Reconstruction accuracy and prior validity results. Baseline results reprinted from [5, 6, 3, 7, 8].

For this reason, in this work, the generative model being used is the JT-VAE, that decomposes molecules into valid substructures, and generates compounds from a vocabulary of these components. Due to this technique, it generates 100% valid molecular structures, as can be seen in the table 2.1 above, comparing the results of various generative methods. A pre-trained JTVAE is used to encode molecular drug data to generate drugs that are effective against the cancer cell lines under consideration.

The authors proposed generating molecules in two steps. First, the molecular graph $\mathcal{G}$ is decomposed into a tree-structured object, junction tree $\mathcal{T}_{\mathcal{G}}$, which represents the structure of the subgraph components and their coarse relative arrangements. The authors proposed extracting these valid substructures automatically using tree decomposition on the training data. The junction tree and the molecular graph offer two complimentary representation of a molecule, due to which the encoding is done in a two-part latent representation $\mathbf{z} = [z_{\mathcal{T}}, z_{\mathcal{G}}]$. Both the tree and the graph are then encoded into their latent embeddings $\mathcal{Z}_{\mathcal{T}}$ and $\mathcal{Z}_{\mathcal{G}}$, $\mathcal{Z}_{\mathcal{T}}$ encodes the tree structure and the information about clusters, but doesn't encode any information about the location of the clusters, while $\mathcal{Z}_{\mathcal{G}}$ captures the fine-grained connectivity by encoding the graph. To decode, first the junction tree is reproduced using a tree decoder $p(\mathcal{T}|\mathcal{Z}_{\mathcal{T}})$, based on the encoded information in $\mathcal{Z}_{\mathcal{T}}$. Then, the connectivity between clusters in the junction tree is predicted by a graph decoder $p(\mathcal{G}|\mathcal{Z}_{\mathcal{G}})$, that enables to complete the decoding and realize the molecular graph. Using the junction tree enables this model to maintain chemical feasibility during generation. The figure 2.6 below shows the overview of JTVAE.

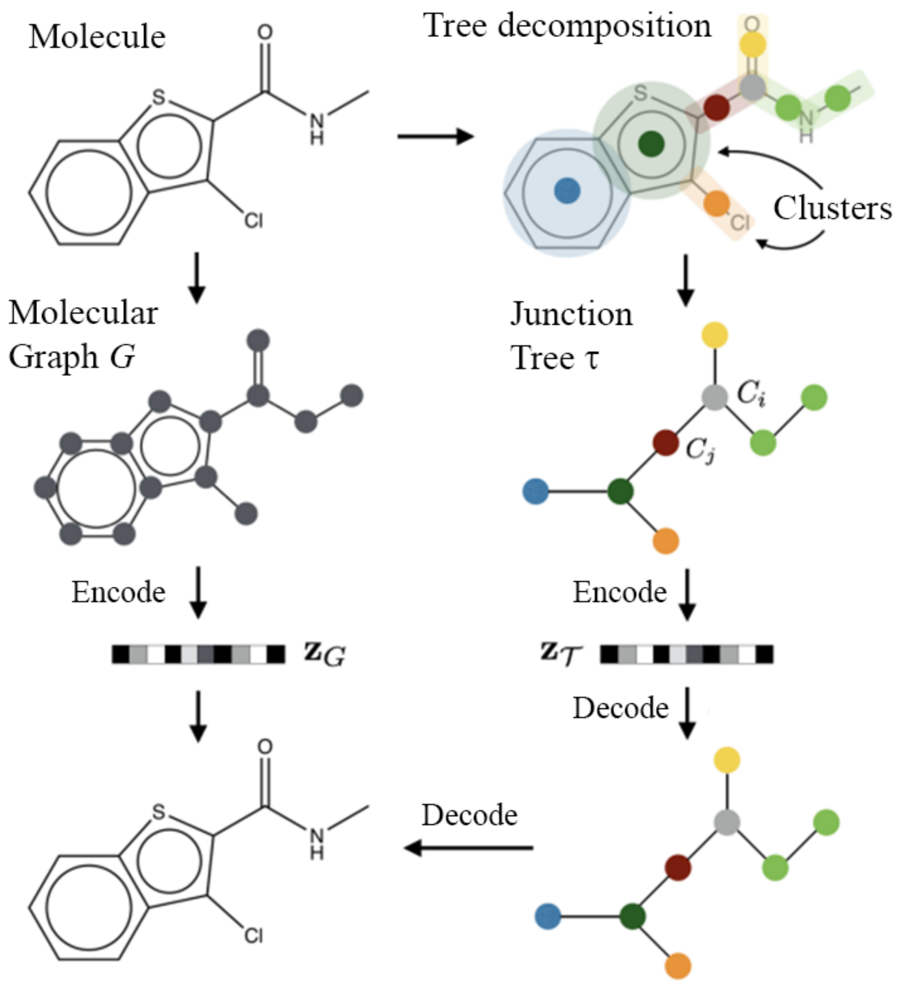


Figure 2.6: Overview of the Junction Tree Variational Autoencoder. Reprinted from [3]

2.1.3.2 Junction Tree

As stated above, the tree decomposition maps a graph G into a junction tree, as it contracts certain vertices into a single node, such that G becomes cycle-free. Formally, given a graph G, a junction tree $\mathcal{T}_G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ is a connected labeled tree, with edge set $\mathcal{E}$ and node set $\mathcal{V} = C_1, ..., C_n$. Each node is an induced subgraph of $\mathcal{G}$, and it satisfies the following constraints:

- The union of all clusters equals $\mathcal{G}$. That is, $\bigcup_i V_i = V and \bigcup_i E_i = E$

- For all clusters C_i, C_j and C_k, $V_i \cap V_j \subseteq V_k$ if C_k is on the path from C_i to C_j

Viewing induced subgraphs as cluster labels, junction trees are labeled trees with label vocabulary $\mathcal{X}$. Due to the molecular tree decomposition, $\mathcal{X}$ only contains cycles (rings) and single edges. The functional groups used in JTVAE can be seen in the figure 2.7 below.

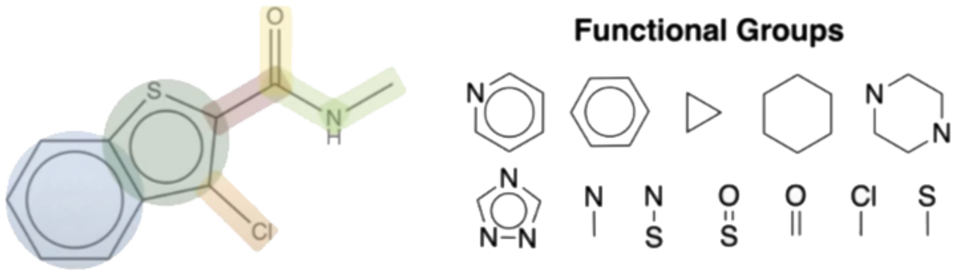


Figure 2.7: Functional Groups used in the Junction Tree VAE. Reprinted from [3]

2.1.3.3 Graph Encoder

Molecular graphs can be generated node by node or structure by structure. For JTVAE, a structure by structure generation scheme is followed, and can be seen in Figure 2.8, along with the node by node generation. The latent representation of $\mathcal{G}$ is encoded by a graph message passing network [8, 38], where each vertex v has a feature vector x_v, which indicates it's atom type, valence, and other properties. Correspondingly, each edge $(u, v) \in E$ has a feature vector x_{uv}, indicating its bond type, and two hidden vectors v_{uv} and v_{vu} that denotes its message from u to v and vice versa. Due to the loopy structure of the graph, messages are exchanged in a loopy belief propagation fashion, and after T steps of iteration, the messages are aggregated as the latent vector of each vertex, which captures it's local graphical structure.

$$h_u = \tau(U_1^g x_u + \sum_{v \in N(u)} U_2^g v_{vu}^{(T)}) \tag{2.8}$$

The final representation is $h_G = \sum_i h_i/|V|$. The mean μ_G and log variance $\log \sigma_G$ of the variational posterior approximation are computed from h_G and with separate affine layers, and z_G is sampled from a Gaussian $\mathcal{N}(\mu_G, \sigma_G)$.

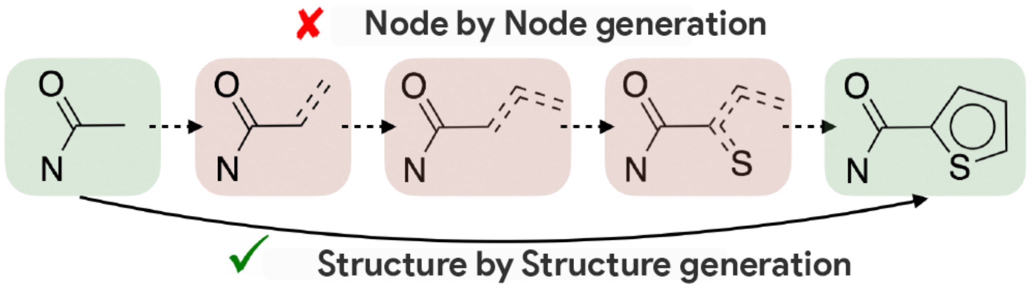

✗ Node by Node generation
O
N
S
✓ Structure by Structure generation

Figure 2.8: Comparison of two graph generation schemes. Reprinted from [3]

2.1.3.4 *Tree Encoder*

$\mathcal{T}_G$ is encoded similarly, with a tree message passing network. One-hot encoding is used to encode each cluster C_i, and x_i represents its label type. Each edge (C_i, C_j) has two message vectors m_{ij} and m_{ji}. After picking an arbitrary leaf node as the root, messages are propagated in two phases. In the first phase (bottom-up), leaf nodes initiate messages and are propagated iteratively towards the root. In the second phase (top-down), messages are propagated from the root to all the leaf nodes. Message m_{ij} is updated as:

$$m_{ij} = GRU(x_i, \{m_{ki}\}_{k \in N(i)/j} \tag{2.9}$$

in the equation above, GRU is a Gated Recurrent Unit [42]. The latent representation of each node h_i is obtained after message passing by aggregating inward messages:

$$\mathbf{h}_i = \tau(W^o x_i + \sum_{k \in N(i)} U^o m_{ki}) \tag{2.10}$$

The final tree representation is $H_{\tau G} = h_{root}$, which is the encoding for a rooted tree $(\tau, root)$. $z_{\tau G}$ is sampled in a similar way as the graph encoder.

2.1.3.5 *Molecular Reconstruction from Latent Vectors*

Once we have encoded the molecule with a tree and a graph encoder,first the tree decoder generates a junction tree from z_T, then the graph decoder combines the substructures in the junction tree, producing the final reconstructed molecule.

The tree decoder traverses the junction tree in depth-first order in a recursive manner, starting from the root. The decoder predicts the probability of the current node having children, and every time a child node is generated, it predicts the label of the child node. Nodes in the junction tree are labeled with the most likely valid substructure.

The graph decoder only assembles one node at a time, following the order when the junction tree is reconstructed. Though there might be multiple ways to assemble the substructures, JTVAE uses the highest scoring strategy [3]. For the purpose of this work, a JTVAE model was pre-trained with the ZINC dataset. As with geneVAE, the predicted mean value of latent vectors is used as encoded features rather than sampling these vectors from a Gaussian distribution.

2.2 Drug Response Prediction Deep Neural Network

For the drug response prediction, once geneVAE and JTVAE are trained, post-processing of the latent features encoded by the two VAE models is performed by two fully connected neural networks, one for each VAE. Another deep neural network concatenates the processed outputs from the two neural networks, producing the final drug response prediction. The input to the final MLP model is $X_{all} = [X_{gene}, X_{drug}]$, where X_{gene} and X_{drug} are the outputs generated by the two post-processing deep neural networks. The two outputs are concatenated, such that the dimensionality of X_{all} is the sum of the dimensionality of X_{gene} and X_{drug}. The value of a perceptron in the i^{th} layer of the final neural network is computed as:

$$X_{all}^{i+1} = f^{'}(W^{(i+1)T}X_{all}^{i} + b^{i+1}) \tag{2.11}$$

where $W^{(i+1)}$ is the weight matrix of the i-th layer of the neural network, and $f^{'}$ is the non-linear activation function. For the activation function, the parametric rectified linear unit (PReLU) is used in the model. The predicted $\ln(IC_{50})$ value is computed in the last layer of the final neural network model. The value is computed as:

$$\ln(IC_{50}) = f^{'}(W^{(n)T}X_{all}^{n-1} + b^{n}) \tag{2.12}$$

in the above equation, n is the total number of layers in the fully connected neural network. As far architecture of the model goes, both of the post-processing neural networks consist of 3 fully connected layers. As the geneVAE and JTVAE are 256-dimension and 56-dimension vectors, respectively, the first of the two post-processing neural network consist of 256, 256 and 64 units in the first, second and third hidden layer respectively, while the second neural network consists of 128, 128 and 64 units in it's first, second and third hidden layer. The third neural network consists of 4 fully connected layers, and consists of 128, 128, and 64 units in its hidden layers. The figure 2.9 below shows the architecture of the 3 fully connected neural networks.

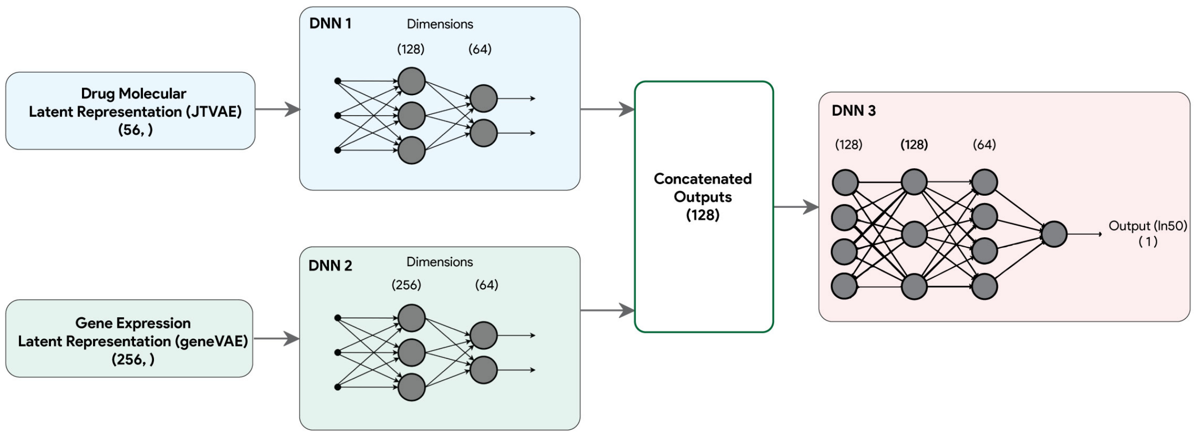


Figure 2.9: Model Architecture of the $ln(IC_{50})$ predictor

2.3 Latent Space Optimization

Though the training results demonstrated that the latent representations encode the input data well, they lacked sample efficiency. As a future direction of work, the focus was on exploring techniques to improve the latent representation. The major problems faced when training generative models are the fact that input is high dimensional and structured, as for the Junction Tree VAE, and the objective function is expensive to evaluate. A sample-efficient optimization procedure, known as weighted retraining [28] is used to achieve this. It is achieved by periodic retraining and, weighting the data-points as per their objective function value Optimizing the Junction Tree VAE's latent space can produce even better results for our property of interest.

There are a few caveats that are to be considered when employing weighted retraining. If additional high-scoring points are augmented to the dataset, the model's performance doesn't improve performance as the data size of high-scoring points is quite small as compared to training data [43]. Furthermore, retraining only on new data could also lead to catastrophic forgetting as the model would have a disproportionate impact of new points [44].

The proposed technique is two step approach, First, the generative model, which in our case is the JTVAE, is trained to find a mapping in a lower-dimensional continuous space onto the data manifold in space. This essentially generates a low-dimensional and continuous analog of the optimization problem. In the second stage, the objective function is optimized over the latent space, effectively decoupling the training & optimization tasks. The approach followed to achieve this is by weighting the data distribution & retraining the Junction Tree VAE. A rank-based weight function was used, that scored data points (molecules) based on the Blood-Brain Barrier (BBB) permeability.

2.3.1 Blood Brain Barrier Permeability Prediction

In order to assign weight to the data-points, a property predictor was trained to predict Blood Brain Barrier (BBB) permeability. The previously discussed $ln(IC_{50})$ predictor cannot be used since it predicted the property values based on latent representation of molecules. The latent representation will, however, change upon the weighted retraining of the JTVAE.

Property predictors operate on precomputed molecular features, which aren't available for the ZINC dataset [45], a subset of which is used to train the JTVAE. Therefore, a property predictor is required that generates property prediction based the molecules, without using their precomputed features. A message passing neural network (MPNN) [38] has been used for this task. The MPNN is based on Graph Neural Network, as that is how the molecules are naturally well represented. The property of interest is $\log(BB)$, which is a measure of the blood-brain barrier permeability. The equation 2.13 gives the mathematical representation of $\log(BB)$. BBB is a highly selective semipermeable border of endothelial cells that prevents solutes in the circulating blood from non-selectively crossing into the extracellular fluid of the central nervous system where neurons reside. Thus, it is an important factor when considering drugs that target central nervous system cancers.

$$\log(BB) = \log \frac{concentration\ in\ brain}{concentration\ in\ blood} \tag{2.13}$$

As mentioned previously, message passing neural network (MPNN) is a type of graph neural network (GNN). Since molecules are naturally represented as an undirected graph, GNNs are useful in predicting molecular properties. The figure 2.10 below shows an overview of the predictor. First we generate graphs from the SMILES strings, which itself is a two step procedure. First, SMILES strings are converted to a molecule object using RDKit in Python, and then the molecule object is converted into graph, represented as a three-tuple (atom_features, bond_features, pair_indices). For training the MPNN, the input consists of a single graph, therefore, for a batch consisting of sub-graphs (multiple molecules), they are merged into a single graph. The merged graph is a disconnected graph, with each molecule (sub-graph) completely separated from the other molecules (sub-graph). The process can be divided into three stages, namely message passing, readout and regression.

1. The message passing step, as depicted in the figure 2.10 below, itself consists of two parts.
   (a) First, the edge network passes messages from 1-hop neighbors, and updates node states based on edge features between them.
   (b) Gated Recurrent Unit (GRU) takes recent node state, and updates it based on previous node state (memory)
2. Once the message passing procedure ends, the aggregated node states need to be partitioned into subgraphs, and subsequently reduced to graph-level embeddings. In the readout, the transformer encoder is used to partition the global graph (batch) into subgraphs (molecules). Each sub-graph is padded to match the sub-graph with greatest number of nodes. The tensor is then passed to the transformer encoder, followed by average pooling.
3. The last stage is the regressor. It consists of a three-layer fully connected neural network to generate $\log(BB)$ prediction. It consists of 512, 256 and 128 nodes in each of the layer respectively, with ReLU (Rectified Linear Unit) activation

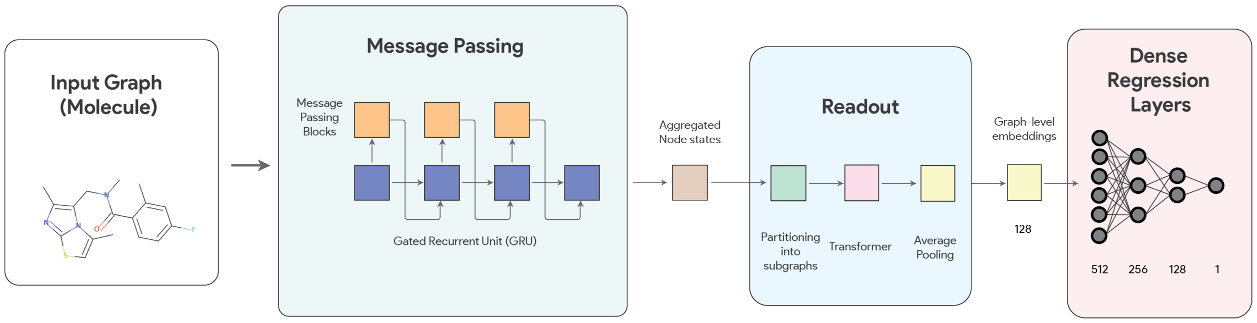


Figure 2.10: Overview of the Message Passing Neural Network based $\log(BB)$ predictor

3. MATERIALS AND METHODS

3.1 Datasets

As mentioned earlier, our drug efficacy model has two separated variational autoencoders, and two different datasets are used to train the geneVAE and the JTVAE. Gene expression data of 1021 cancer lines, along with 57820 genes is provided by the Cancer Cell Lines Encyclopedia (CCLE) [4]. Similarly, to train the JTVAE, ZINC dataset [45] containing molecular structure data of organic compounds is used. It consists of molecular structure data in the form of SMILES strings. ZINC dataset consists of 250k molecules, and the JTVAE is trained on randomly selected 50k molecules, which is manifolds larger than the 367 drugs that are used for drug response prediction task. The latent vector is of size 56, and consists of the concatenated tree and graph encoding.

3.1.1 Gene expression data

The gene expression data consisting of 57820 genes of 1019 cancer lines was obtained from the Cancer Cell Lines Encyclopedia (CCLE) [4]. Each of the cell line belong to a specific type of cancer, and for the purpose of this work, breast cancer was selected as the primary objective, and the model was then generalized on pan cancer cell lines. The distribution of the various cancer cell lines in CCLE is illustrated in the figure 3.1 below.

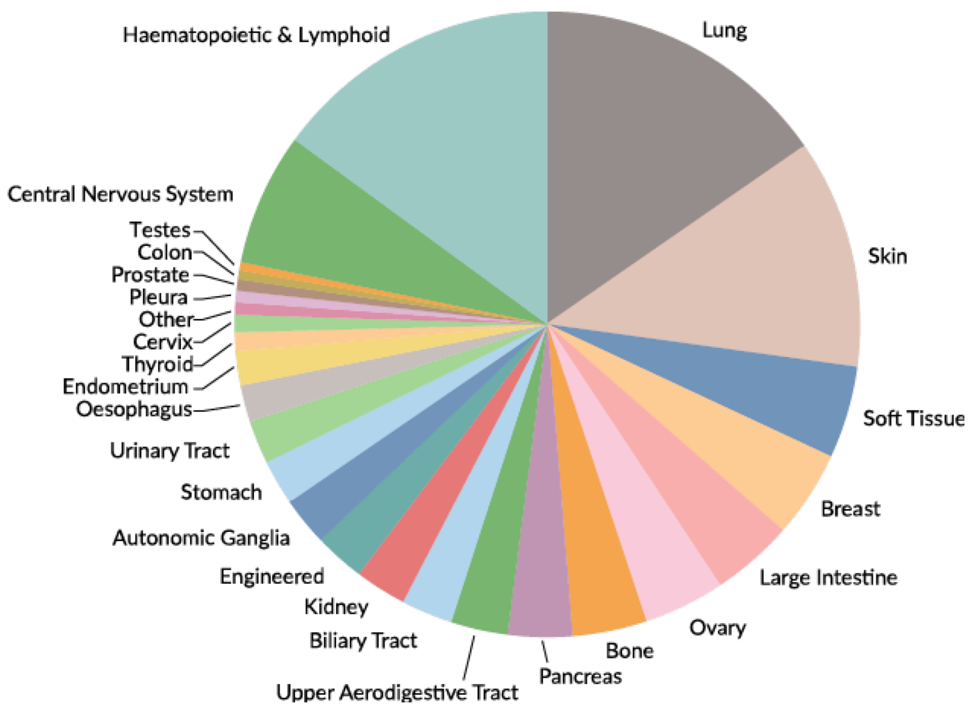


Figure 3.1: Distribution of the cell lines with respect to the cancer types in the CCLE. Reprinted from [4]

The data is filtered into three different subsets, to compare the model performance. One subset consists of breast cancer cell lines, and 51 cell lines are selected from the dataset, including [HMC18_BREAST], [HCC1395_BREAST], [MDAMB453_BREAST] and [ZR751_BREAST] to list a few. Another subset consisted of 65 central nervous system cancer cell lines (glioma and glioblastoma), and included [KG1C_CNS], [LN229_CNS], [SF268_CNS] and [TM31_CNS] to name a few. The third dataset consisted of all cancer cell lines (pan-cancer). If we number of genes g , and c is the number of cancer cell lines, the gene expression data can be represented as $G \in \mathcal{R}^{gxc}$. The matrix G would have $\log_2(t_{pm} + 1)$ number of elements, where t_{pm} is the transcriptome per million value (tpm) of the gene in the corresponding cell line.

Further, the Cancer Gene Census (CGC) dataset [15], which classifies genes into two categories

Cancer Type	Cell Lines	Gene Expressions
Breast Cancer	51	564
Central Nervous System	65	590
Pan-Cancer	1019	659

Table 3.1: Number of gene expressions for each cancer type

is also to select the genes of interest for this work. The first category is of the genes that are associated closely to cancers. These have a high mutation probability to mutate in cancers that change the activity of the gene product. The second category is of the genes that lack evidence, but have a high possibility of playing a strong role in cancer. Genes in both of these categories are of interest, and therefore, are incorporated for further analysis. For the breast cancer analysis, 51 cell lines are selected from CCLE dataset, and expression data is removed for genes that are not in the CGC dataset. Further filtering is performed based on the mean and variance. Gene expression entrance having a mean $\mu < 1$ or standard deviation $\sigma < 0.5$ are also not considered for further analysis, for the lack of relevance with cancer cell lines [18]. Figure 3.2 summarizes the data pre-processing, and a summary of gene expression & cell lines data is provided in table 3.1.

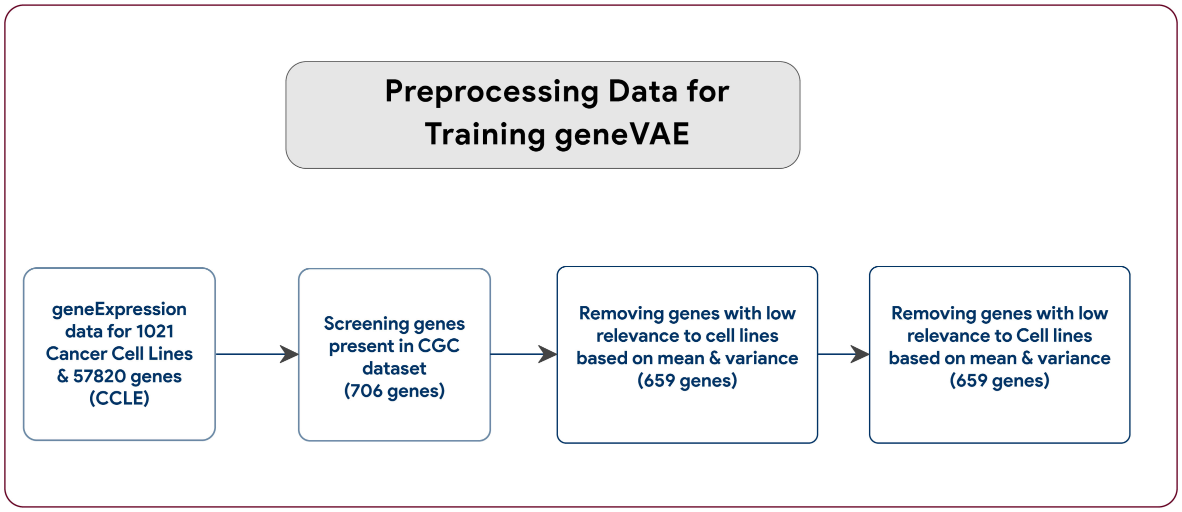

Preprocessing Data for Training geneVAE
geneExpression data for 1021 Cancer Cell Lines & 57820 genes (CCLE)
Screening genes present in CGC dataset (706 genes)
Removing genes with low relevance to cell lines based on mean & variance (659 genes)
Removing genes with low relevance to Cell lines based on mean & variance (659 genes)

Figure 3.2: Overview of gene expression data preprocessing

3.1.2 Molecular structure & anti-cancer drug data

For training the JTVAE, ZINC dataset [5] is used, that includes the molecular structure data in SMILES strings. This dataset contains about 250K drug molecules, that are extracted from the ZINC database [45]. This is used to train the JTVAE, so that it learns the molecular structure of organic compounds. SMILES strings have been used widely to define drug's structure [3, 5, 46, 15, 14, 17], and also used as an input for drug prediction problems. Another benefit of using SMILES strings is that it's easier to extract the embeddings from vocabulary by parsing the generated library. Although the ZINC dataset contains hundreds of thousands of SMILES strings, for the purpose of this work, 50,000 SMILES strings are randomly selected to train the JTVAE considering the resource constraints. Even though the entire dataset is not used, it still includes far larger number of chemical compounds as compared to the GDSC dataset that is used for the, which only has 367 drugs for the fact that this enable the model to have better generalization over all drug structures, and won't be limited to drugs related to specific cancer types.

Cancer Type	Cell Lines & Drug Response Data	Training Data	Validation Data	Test Data
Pan Cancer	24278	19666	2185	2427
Central Nervous System	2142	1736	192	214
Pan-Cancer	3526	2857	317	352

Table 3.2: Data Split for training $\ln(IC_{50})$ predictor

3.1.3 Drug Response prediction data

Once the JTVAE and geneVAE are trained, anti-cancer drug data is needed to predict response for the cancer cell line under consideration. This data of anti-cancer drug response is obtained from the Genomics of Drug Sensitivity in Cancer (GDSC) project [13]. GDSC contains response data for anti-cancer drugs that are used against numerous cancer cell lines. This data is given in the form of a matrix $IC_{CCLE} \in R^{dxc}$, with d and c being the number of drugs and cancer lines respectively. This matrix consists of $\ln(IC_{50})$, or the half maximal inhibitory concentration value of the drugs that used again specific cancer cell lines. It is a quantitative measure of the potency of a substance in inhibiting a specific biological or biochemical function, which in this case is cancer. An, IC_{50} value indicates how much of the inhibitory substance is needed to inhibit, in vitro, the biological process, or component by 50%. The molecular data of these anti-cancer drugs is obtained from Pub-Chem dataset, using each drugs' unique PubChem ID that is accessible in the GDSC dataset. The figure 3.3 below summarizes the data preprocessing steps taken for training the $\ln(IC_{50})$ predictor. The data was split into training, validation and test sets. The table 3.2 gives a summary of the test, train and validation sets for each of the subsets.

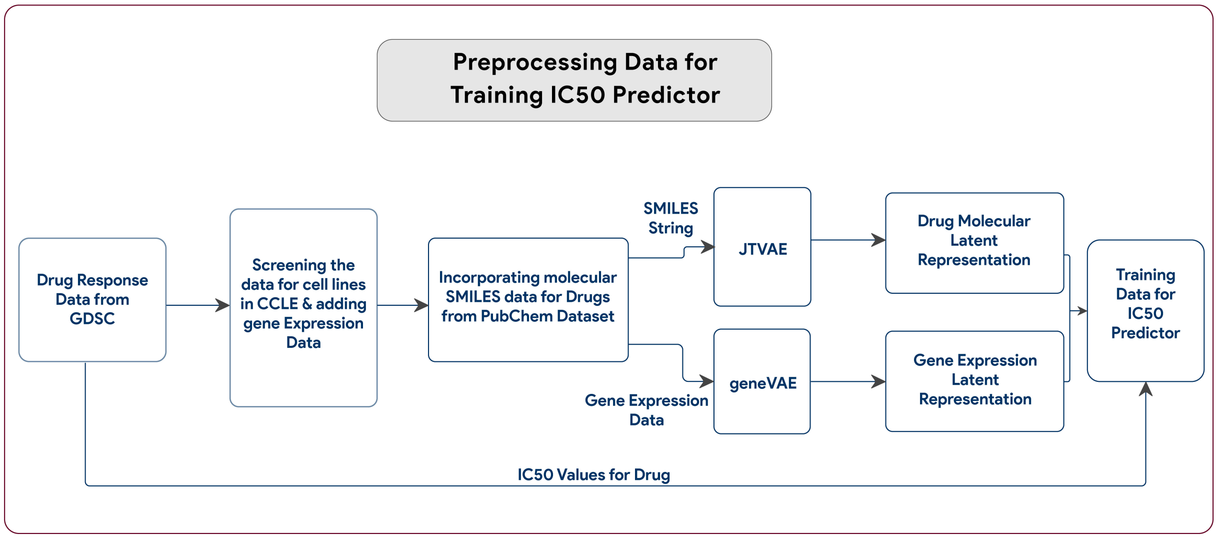

Preprocessing Data for Training IC50 Predictor
Drug Response Data from GDSC
Screening the data for cell lines in CCLE & adding gene Expression Data
Incorporating molecular SMILES data for Drugs from PubChem Dataset
SMILES String
JTVAE
Drug Molecular Latent Representation
Gene Expression Data
geneVAE
Gene Expression Latent Representation
Training Data for IC50 Predictor
IC50 Values for Drug

Figure 3.3: Overview of Data preprocessing for Training $\ln(IC_{50})$ predictor

For the purpose of this work, geneVAE and JTVAE need to be pre-trained before drug response prediction for cancer cell lines can be performed. The datasets used for pre-training these variational auto-encoders are explained previously. Both the geneVAE and JTVAE are trained in an unsupervised manner. Once trained, geneVAE is used to encode the gene expression data from CCLE, while the Junction Tree VAE is used to encode anti-cancer drugs. The encoded features from these VAEs are the used to train a deep neural network model on breast cancer cell lines, and central nervous system cell lines. Later, the model is generalized, and tested on the pan-cancer cell lines. A flow-diagram of the model is given in the figure 3.4 below.

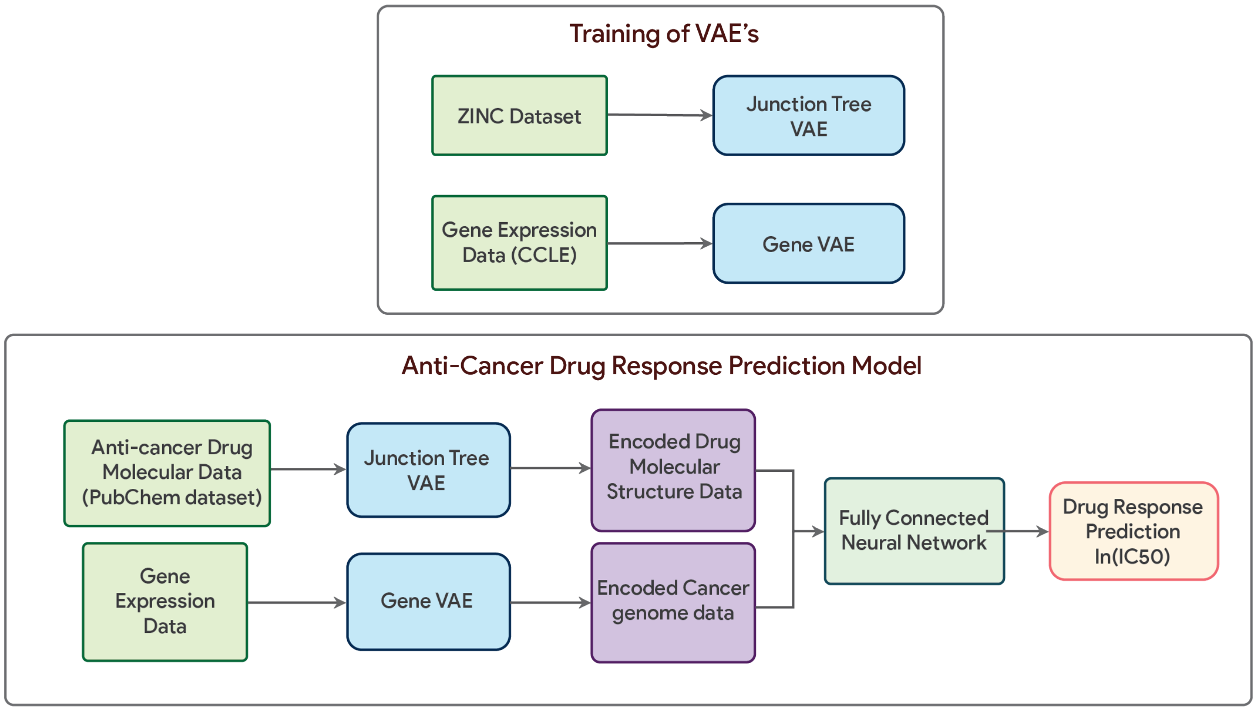

Training of VAE's
ZINC Dataset
Junction Tree VAE
Gene Expression Data (CCLE)
Gene VAE
Anti-Cancer Drug Response Prediction Model
Anti-cancer Drug Molecular Data (PubChem dataset)
Junction Tree VAE
Encoded Drug Molecular Structure Data
Gene Expression Data
Gene VAE
Encoded Cancer genome data
Fully Connected Neural Network
Drug Response Prediction ln(IC50)

Figure 3.4: Flow Diagram of Training VAE's and Anti-Cancer Drug Response Prediction Model

3.1.4 GeneVAE training

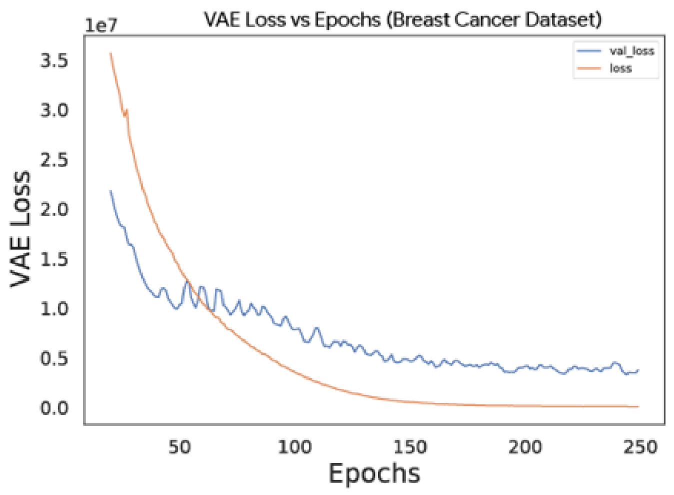
As geneVAE is trained, the objected is to minimize the sum of reconstruction loss and latent loss. The reconstruction loss is represented by $\mathcal{L}(G, G_0))$, where G is the initial input gene expression data, while $G^{'}$ is the reconstructed data. For reconstruction loss, cross entropy loss is used as the input data is normalized, and the output layer has a sigmoid activation function, thereby ensuring that the input and output values are in within the range of 0 and 1.

It was evident from the filtering out the gene subset using CGC dataset representative of the specific cancer type under consideration (Breast Cancer or CNS) was of significance during the training of the geneVAE model. As mentioned earlier, specific genes are selected based on CGC dataset for breast cancer and CNS. The model is evaluated with and without filtering out a gene subset on glioma and breast cancer cell lines. The results indicated that if a gene subset is filtered, the prediction accuracy of IC_{50} value of the drug could be improved. According to the evaluation of total loss, validation loss converges after around 150 epochs. Total VAE loss was calculated as:

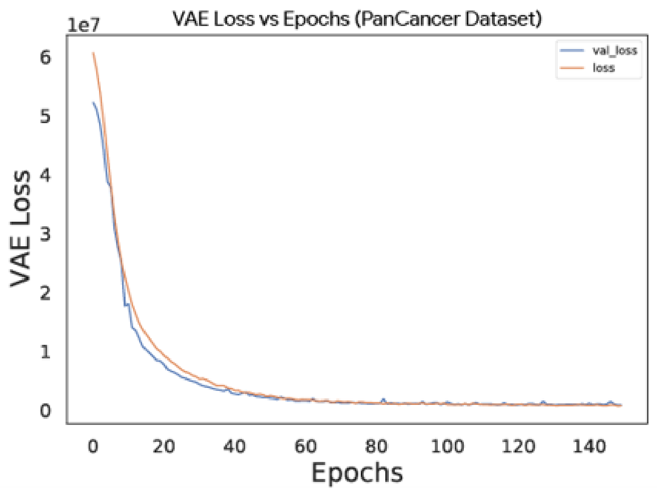
$$\textbf{VAE\_Loss} = \mathcal{L}(G, G') + \beta KL \tag{3.1}$$

where KL is the Kullback-Leibler loss, and β increases from 0 to 1 gradually during training. This is also known as warm-up training strategy. Initially, the learning rate was set to 0.1, and then decreased in the later epochs, with the minimum value being 0.01. This is also known as learning rate decay, whereby the learning rate was decreased 20% if the validation loss in a range of 0.5 for more than 10 epochs.

The training performance of geneVAE on validation set and training set for PanCancer dataset, and Breast Cancer dataset is given in the figure 3.5 below.



(a) VAE loss for training and validation set over 250 epochs on Breast Cancer Dataset



(b) VAE loss for training and validation set over 150 epochs on PanCancer Dataset

Figure 3.5: VAE Loss vs Epochs for geneVAE training

3.1.5 JTVAE training

For training the Junction Tree VAE, 50,000 SMILES strings are randomly selected from the 250K SMILES strings from the ZINC molecule dataset [5].The ZINC dataset is itself derived from the ZINC database [45]. The latent space dimension is set as 56, with $h_{\mathcal{T}}$ and $h_{\mathcal{G}}$, the tree & graph representation, having 28 dimensions each. A vocabulary is required to train the JTVAE, and for this work, the trained vocabulary by [3] was used. It was trained on 240K molecules from the ZINC dataset, and was of size $|\mathcal{X}|$ = 780. The dimensions for each hidden state is 450, in addition to the latent space dimension of 56. For the graph encoder, the atom features include atom type, degree, chiral configuration and it's formal charge. For the tree encoder, each cluster is represented with a neural embedding vector, in a way how word embeddings works for words. The tree and graph decoders use same feature setting as their encoders. All the neural components are implemented in PyTorch, while RDKit [41] is used for processing molecules.

The training of JTVAE is done in two steps, as suggested by [3]. First, the model is trained for 3 steps and without Kullback-Leibler (KL) regularization term. This is practically the same as training an autoencoder. In the second step, the model is trained with KL regularization.

3.2 Experiments

As part of this work, experiments were done to evaluate the performance of the model. First, the geneVAE and JTVAE were trained in an unsupervised manner. Then, these trained VAE's were used to encode encode gene expression data (filtered by the CGC data set or not on the breast cancer cell lines), and to encode anti-cancer drug molecular data from PubChem. These encoded features were then used to train the fully connected neural network for drug response prediction. (Add also 1/2 baseline models here for comparison). Once trained, the model was then tested on breast cancer cell lines, followed by glioma cell lines and then by pan-cancer cell lines. In addition to drug response prediction, this work also demonstrates the generative aspect of this model, and it's ability to generate effective drugs for specific cancer cell lines under consideration. The test, train and validation split used for training the fully connected neural network was 1:18:1. The model was implemented using PyCharm on a Microsoft Windows PC. Training was performed using an NVidia GeForce GT740m GPU, or using Texas A&M's High Performance Research Computing's resources for the more resource intensive tasks, including training the JTVAE and geneVAE.

3.2.1 Results on Breast Cancer Dataset

The first dataset that the model was evaluated on was for breast cancer cell lines. Two metrics, including Root Mean Square Error (RMSE) and Coefficient of Determination (R2 score), were used to compare the predicted drug response and the actual drug response. In order to compare the results, the model was trained with a fully connected neural network, as well with Support Vector Machine. In addition to this, the models were tested on gene expression data encoded by geneVAE, and also without encoding.

1. SVR with CGC filtered data : A Support Vector Regression model (for drug response prediction) trained on CGC filtered gene expression data, and drug molecular structure data, with both being encoded by VAE model.

2. Fully connected Neural Network with CGC filtered data : Fully connected neural network model trained on drug molecular structure data encoded by VAE model, & gene expression data filtered by CGC dataset (not encoded by geneVAE).

3. Fully connected Neural Network with CGC filtered data VAE encoding: Fully connected neural network model trained on drug molecular structure data & gene expression data, both encoded by VAE model, with the gene expression data being filtered by CGC dataset.

The results of these models showed that the neural network model had better performance when compared with SVR (R^2 of 0.792 vs 0.63), and the that the model in which gene expression data was encoded also had better performance. The performance improvement was significant for the same model with geneVAE encoded data vs without (R^2 of 0.792 vs 0.68). This also depicts the fact that the selection of a representative gene is significant to the performance. Moreover, the selection of representative gene subset is essential to the performance of our models.

3.2.2 Results on CNS Dataset

The second dataset that the model was evaluated on was for brain cancer (glioma), including Glioblastoma. The same metrics (RMSE and R2 score) were used to compare the predicted drug response and the actual drug response. In order to compare the results, the model was trained with a fully connected neural network, as well with Support Vector Machine. In addition to this, the models were tested on gene expression data encoded by geneVAE, and also without encoding.

1. SVR with CGC filtered data : A Support Vector Regression model (for drug response prediction) trained on CGC filtered gene expression data, and drug molecular structure data, with both being encoded by VAE model.

2. Fully connected Neural Network with CGC filtered data : Fully connected neural network model trained on drug molecular structure data encoded by VAE model, & gene expression data filtered by CGC dataset (not encoded by geneVAE).

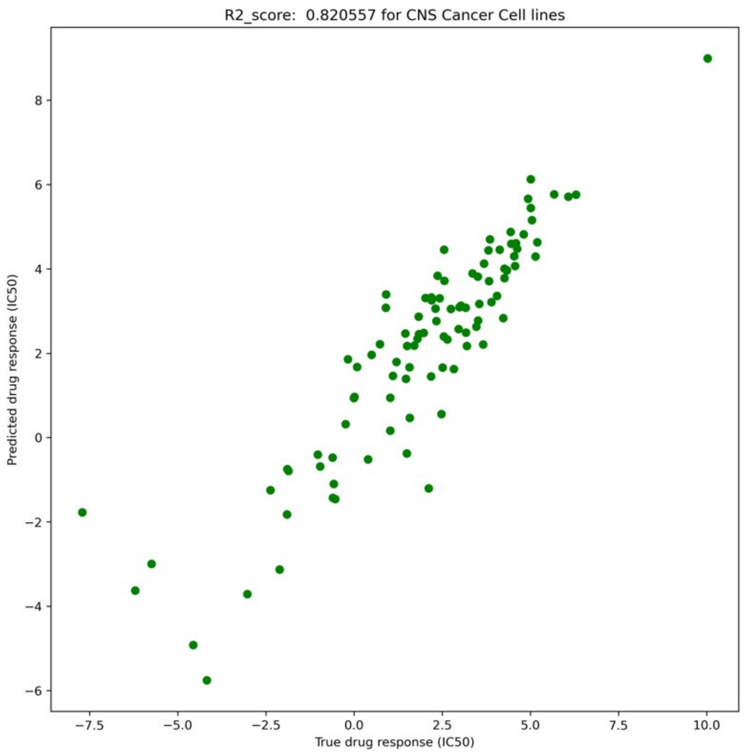
3. Fully connected Neural Network with CGC filtered data VAE encoding: Fully connected neural network model trained on drug molecular structure data & gene expression data, both encoded by VAE model, with the gene expression data being filtered by CGC dataset.

The results of these models showed that the neural network model had better performance when compared with SVR (R^2 of 0.835 vs 0.69), and the that the model in which gene expression data was encoded also had better performance. The performance improvement was significant for the same model with geneVAE encoded data vs without (R^2 of 0.835 vs 0.78). This also depicts the fact that the selection of a representative gene is significant to the performance. Moreover, the selection of representative gene subset is essential to the performance of our models.

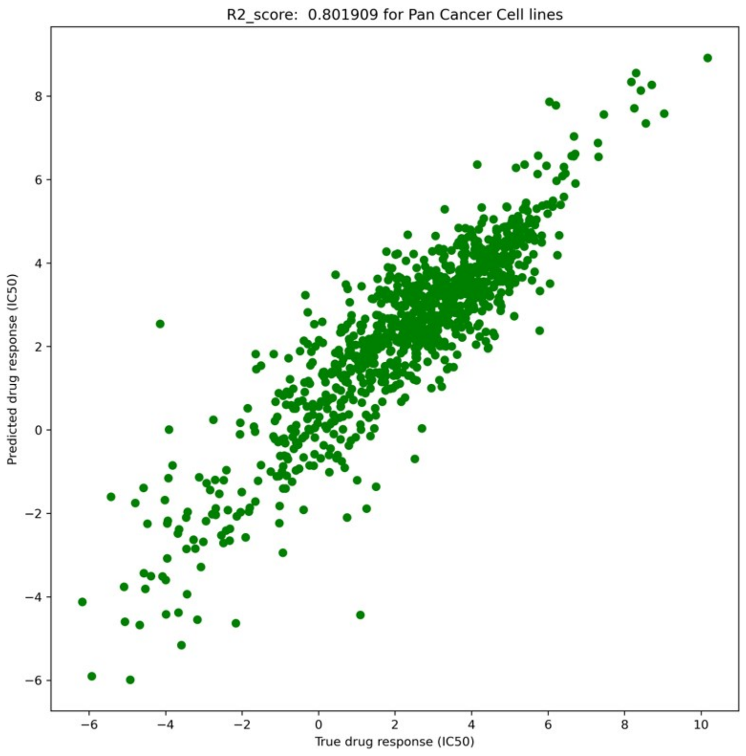
3.2.3 Results on Pan Cancer Dataset

After comparing the results for breast cancer and glioma, it was evident the model with neural network, and with gene expression data encoded by geneVAE performed the best. Therefore, it was also trained on pan-cancer dataset to understand how well it generalizes. The performance on pan-cancer dataset was even better, with the R^2 *score* being 0.875m and $RMSE$ score of 0.921 for pan-cancer cell lines (1019).

The plots in figure 3.6 below show the comparison of the predicted vs true $ln(IC_{50})$ values.



(a) Predicted vs true $ln(IC_{50})$ values for CNS Dataset



(b) Predicted vs true $ln(IC_{50})$ values for Pan-cancer Dataset

Figure 3.6: Scatter plots showing the model performance for the predicted $ln(IC_{50})$ values for Pan-cancer model and CNS Cancer model

Model Type	R^2 score Test Set	RMSE Test Set	R^2 score (*Trained without geneVAE*)	RMSE (*Trained without geneVAE*)
Pan Cancer	0.875	0.92	-	-
Central Nervous System	0.792	1.0374	0.68	1.42
Pan-Cancer	0.835	0.926	0.69	1.37

Table 3.3: Performance of the $\ln(IC_{50})$ predictor

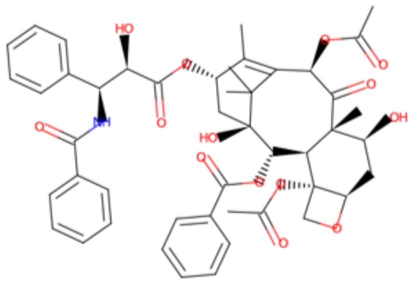
The table 3.3 above summarizes the model performance.

3.2.4 $\ln(IC_{50})$ predictor sanity check

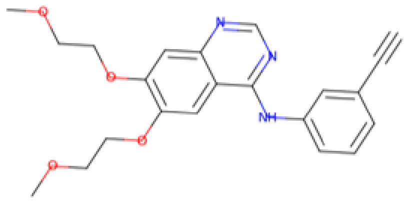
In order to demonstrate that property predictor can produce similar latent variables, even for molecules having quite different chemical structures, two drugs, MG-132 and Erlotnib were selected. Their chemical structures are provided in the figure 3.7 They are used against Cell Line SF539 of central nervous system. The model was trained by removing these two drugs from the training set, and their $\ln(IC_{50})$ value was predicted, as given in the table 3.4. Though the $\ln(IC_{50})$ predicted values are a bit different, the model was able to predict the molecule's $\ln(IC_{50})$ response similar to it's actual value (positive). Therefore, despite quite different molecular structures, the model is able to extract useful features and predict $\ln(IC_{50})$ values.

Drug	$\ln(IC_{50})$ true value	$\ln(IC_{50})$ predicted value
Erlotnib	2.47	3.5
MG-132	2.12	1.07

Table 3.4: Property predictor performance comparison



(a) Chemical Structure of Drug Erlotnib

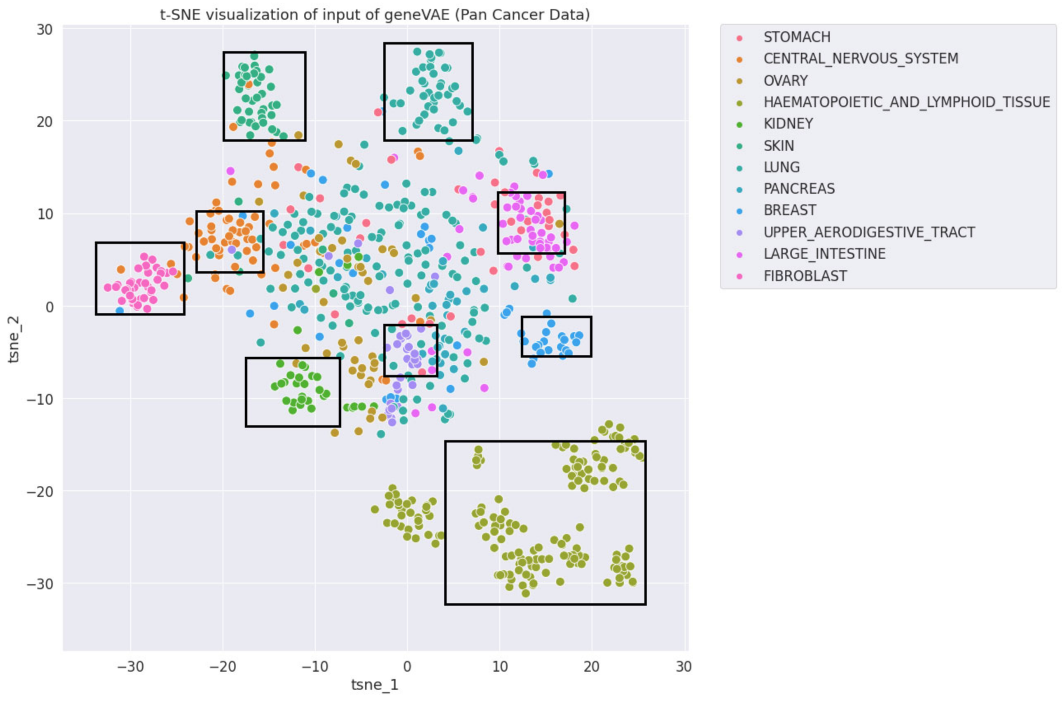


(b) Chemical Structure of Drug MG132

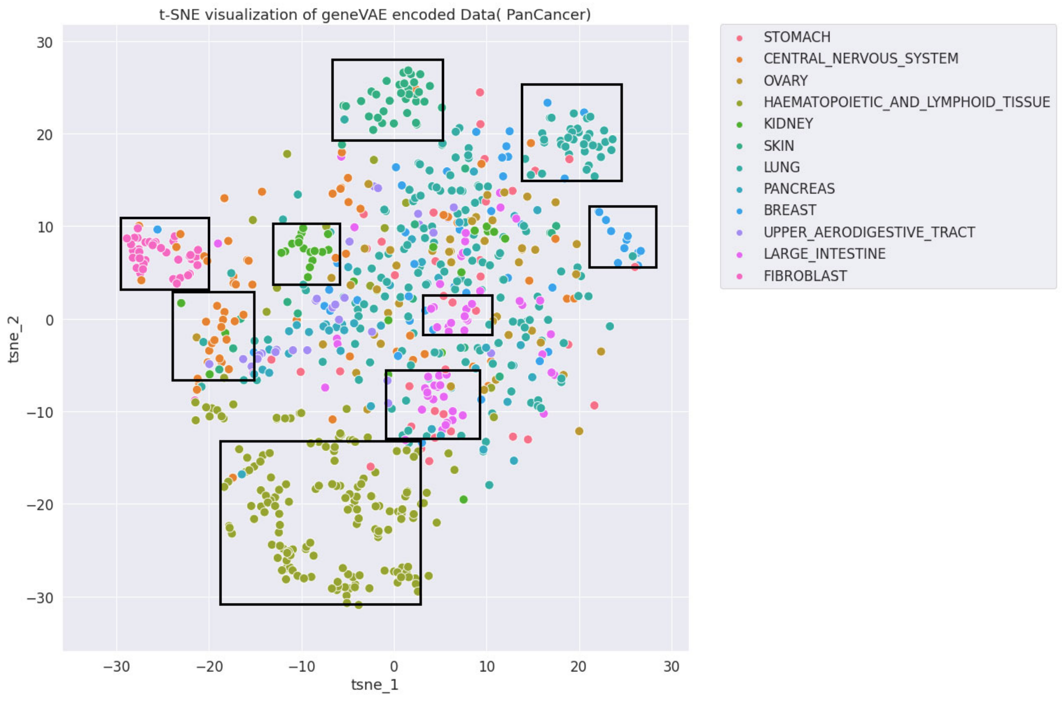
Figure 3.7: Chemical Structures of MG132 & Erlotnib used against Cell Line SF539 of CNS

3.3 Exploring geneVAE latent vectors

In order to visualize whether the geneVAE was able to encode gene expression data well, t-SNE was used to reduce the dimensionality of both the original gene expression data, and the encoded data by geneVAE, as provided in the figure 3.8 below. The figure is the t-SNE visualization of the gene expression data for top 12 Cancer types (to help visualize better). It can be seen that the clustering remained similar as for the original data, where primary cancer tissue types are separated clearly. This depicts that the latent vectors encoded by geneVAE model retained the essential features of original data robustly.



(a) Input data

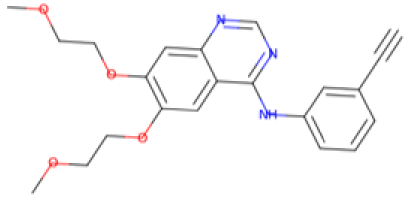


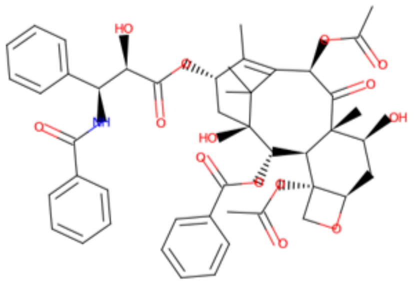
(b) Encoded data

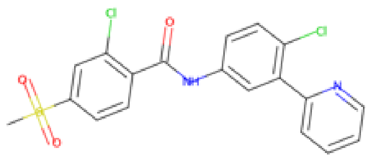
Figure 3.8: t-SNE visualization of gene expression data for 12 most occurring cancers

3.4 Exploring JTVAE latent vectors

In order visualize how close the latent vectors of different drugs are encoded in the latent space by the JTVAE, 4 different drugs were selected (MG-132, Erlotnib, Vismodegib and FH535) used against SF126 Cell Line of Central Nervous System. The molecular structures are also given below in Figure 3.9. To compare their latent representations, Euclidean distance was used to measure how close, or further away the drugs are in the latent space. The drugs Vismodegib and FH535 share the most functional groups, and also have the lowest Euclidean distance, while the largest value of Euclidean distance is between drug Erlotnib and Vismodegib. It is visible that they Erlotnib and Vismodegib have quite different functional groups, which depicts that they are further away in the latent space. A comparison of the euclidean distances for these 4 drugs is provided in the table 3.5


(a) MG132


(b) Erlotnib


(c) Vismodegib

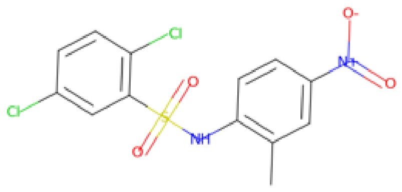

(d) FH535

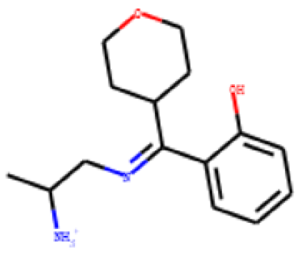
Figure 3.9: Molecular structures of different drugs

Drug	Erlotnib	MG-132	FH535
Erlotnib	-	-	-
MG-132	23.36	-	-
FH535	17.06	25.1	-
Vismodegib	17.4	22.88	-

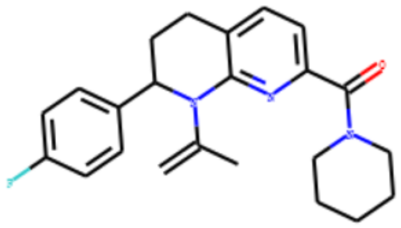
Table 3.5: Euclidean distance of latent vectors of Drugs encoded by JTVAE

3.5 Drug Compound Generation

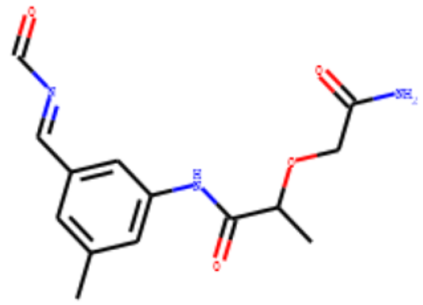
The above mentioned results for drug response prediction, though useful, don't fully utilize the representations generated by the generative models (JTVAE and geneVAE). As explained earlier, JTVAE has the advantage that it's reconstruction accuracy, when compared with other generative models, is higher (76.7%) and almost all the molecules generated have valid molecular structures. Also, drugs sharing similar molecular structure have similar latent vectors. (add more info here, exploring the latent space) To demonstrate how JTVAE can to generate customized & effective drug compounds for a given cancer cell line, the breast cancer cell line HCC1419 is used. for a given cancer cell line. Several 56-dimension vectors were sampled from the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, with $\mu = 0$ and $sigma = 7$. The dimensionality was selected to match the dimensions of the JTVAE latent space. Next, these randomly sampled drug vectors were then concatenated with encoded latent vector of gene expression profile of HCC1419 by the geneVAE. This concatenated vector was fed to the neural network as input, and it predicted the IC_{50} score for the drug. Based on the IC_{50} score, we can see whether the drug is effective against the cancer cell line, and if it is, the molecular structure can be found using the JTVAE decode. The 4 vectors with least score were fed to the JTVAE decoder and their molecular structures are provided in the Figure 3.10



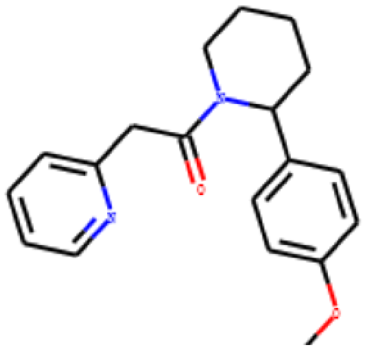
(a) $ln(IC_{50})$ = -1.21



(b) $ln(IC_{50})$ = -1.031



(c) $ln(IC_{50})$ = -1.17



(d) $ln(IC_{50})$ = -1.05

Figure 3.10: Molecular structures of potential drugs predicted to be effective against HCC1187 cell line

3.6 Latent Space Optimization

Although we have demonstrated that the latent representations encode the input data well, they lack sample efficiency. As a future direction of work, I've focused on exploring techniques to improve the latent representation. For weighted retraining, first a property predictor was trained, with training data consisting of $\log(BB)$ for 333 drugs used against CNS tumors. First, the MPNN based predictor was trained. Once trained, it was used to predict $\log(BB)$ values for 50k molecules in the ZINC dataset that were used originally to train the JTVAE. The $\log(BB)$ values were used in the rank-based weighting of the training data, which was then used to retrain the JTVAE.

3.6.1 MPNN Training

The Message Passing Neural Network based $\log(BB)$ predictor was trained for 160 epochs, and the performance on the training & test data is provided in the Figure 3.11 below. It can be seen that the validation MSE stabilizes around 100 epochs.

The predicted and true values of $\log(BB)$ for the dataset were plotted, as can be seen in the Figure 3.12.

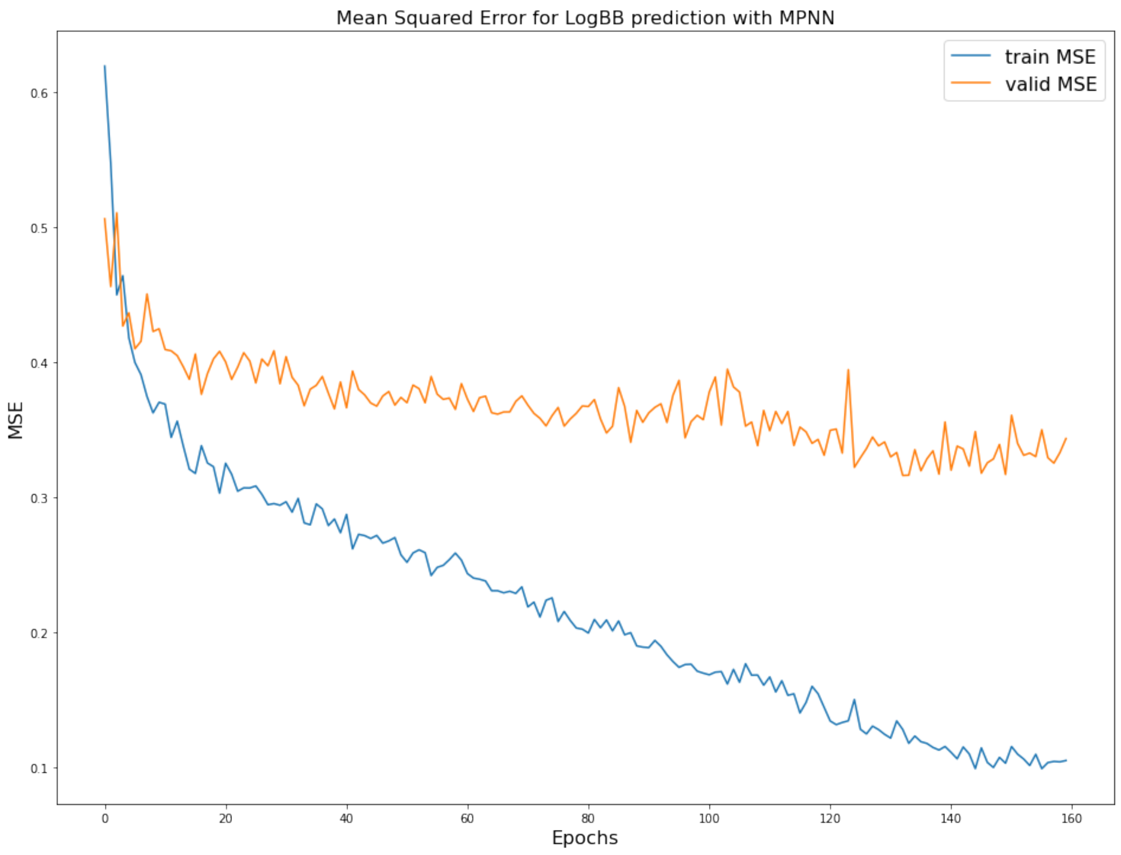


Figure 3.11: Training performance of MPNN (training MSE vs validation MSE)

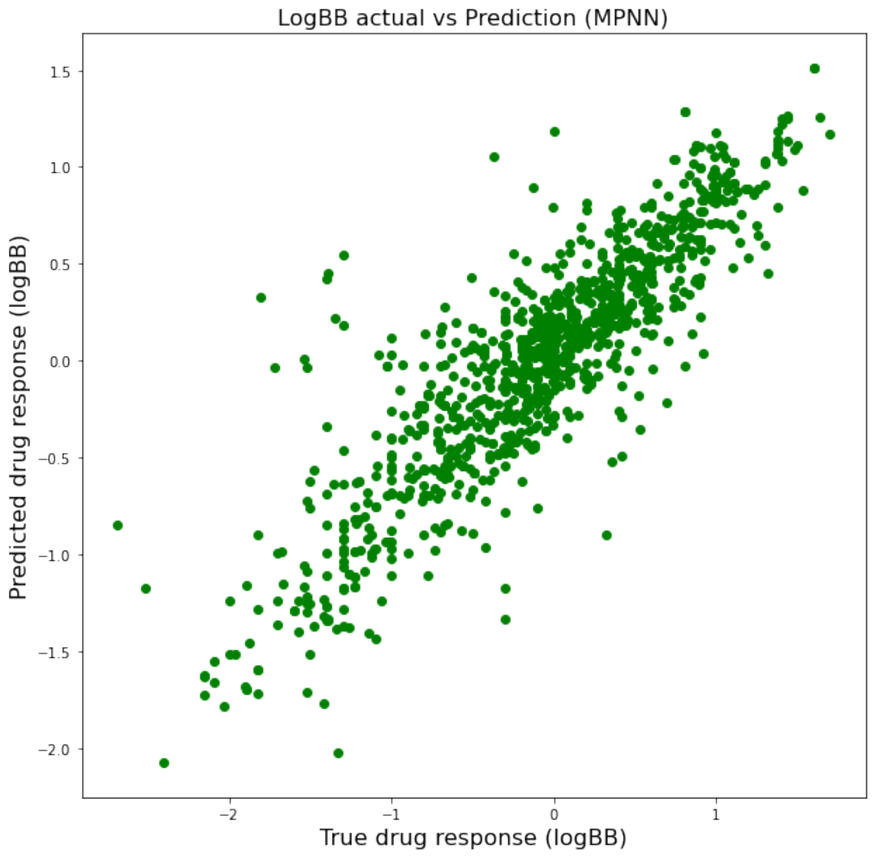
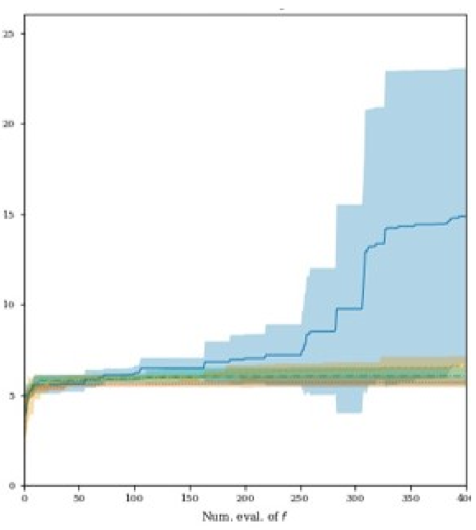


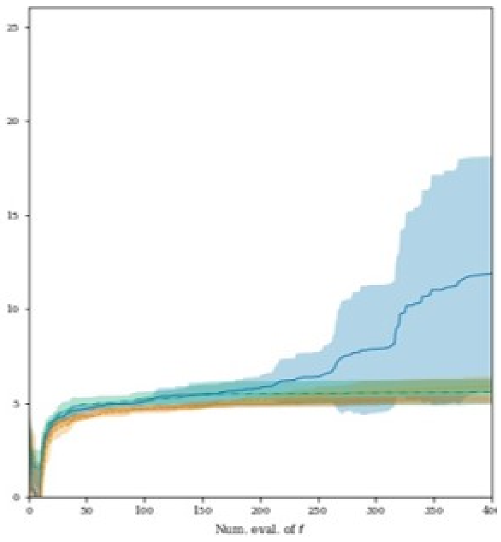
Figure 3.12: Scatter plot showing the true drug response vs predicted drug response

3.6.2 Weighted Retraining

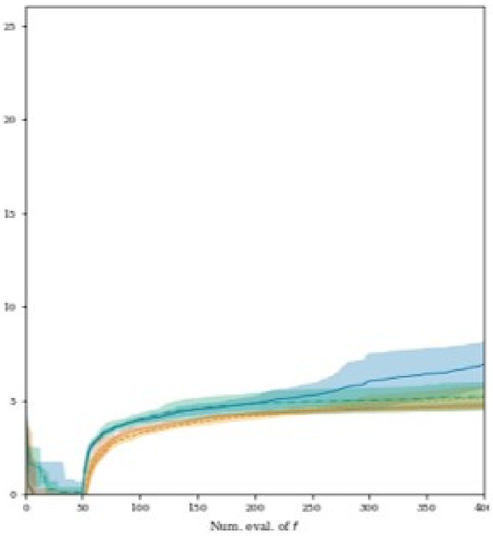
As explained above, weighted retraining was performed. The performance is given in the Figure 3.13 below. In the image, top 1 denotes the single best score obtained until the query f, top 10 denotes the worst of the top 10 scores, and top 50 denotes the worst of the top 50 scores. K is the weighting parameter, and r gives the retraining frequency for the samples. The area shaded gives the standard deviation. This is a future direction for the presented work and further experiments are needed to evaluate the performance of weighted retraining.



(a) Top 1



(b) Top 10



(c) Top 50

Figure 3.13: Optimization performance of Weighted retraining for different values of weighting factor k and retraining frequency r

3.7 Code Availability

All the codes for reproducing the results presented above can be found at Cancer Drug Discovery Github Repo.

4. CONCLUSIONS

This work describes the development of an integrated machine learning approach for anti-cancer drug discovery using generative models. In the sections above, details about two generative models, a gene expression encoding VAE(geneVAE) model, and a Junction Tree VAE (JT-VAE)model for molecular structure encoding has been explained. Also, performance was compared for a Support Vector Regression model and a Deep Neural Network to predict anti-cancer drug response. Also, the latent vectors encoded by geneVAE and JTVAE model are then fed to the drug prediction network to generate the final drug efficacy, represented by IC_{50} score. The models are trained for different datasets, with gene expression data filtered for cancer cell lines (breast cancer and glioma) and the performance was compared, including the effect of encoding gene expression data using the geneVAE. The model achieved an R_2 score of 0.85 on pan-cancer data. Also, the work discuses how the geneVAE and JTVAE model can be used to generate new drug molecular structures for a specific cancer cell line.

In addition to this, Latent Space Optimization improves the latent representation based on our property of interest – Blood Brain Barrier Permeability.

4.1 Future Work

In the future, further work on Latent space optimization is to be done in the near future to further improve the prediction performance. Also, potential of using Graph Neural Networks (GNN) to encode drug data and also explore the potential of prediction drug-drug interaction with the use of GNN. And last, my target is to build a toolkit for drug response prediction, and aid in drug discovery process, based on the models presented in this thesis ($\ln(IC_{50}) and \log(BB)$),

REFERENCES

[1] "Learn about cancer types and our treatment options." `https://www.medipulse.in/blog/2021/9/15/learn-about-cancer-types-amp-our-treatment-options`, Sep 2021.

[2] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *CoRR*, vol. abs/2003.05991, 2020.

[3] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *International conference on machine learning*, pp. 2323–2332, PMLR, 2018.

[4] J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, *et al.*, "The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity," *Nature*, vol. 483, no. 7391, pp. 603–607, 2012.

[5] J. Kusner, B. Paige, and J. Hernández-Lobato, "Grammar variational autoencoder. arxiv e-prints: 1703.01925," 2017.

[6] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International conference on artificial neural networks*, pp. 412–422, Springer, 2018.

[7] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[8] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, pp. 2702–2711, PMLR, 2016.

[9] "Cancer." `https://www.who.int/news-room/fact-sheets/detail/cancer`, Sep 2021.

[10] W. H. Organization *et al.*, "Assessing national capacity for the prevention and control of noncommunicable diseases: report of the 2019 global survey," 2020.

[11] C. Wild, E. Weiderpass, and B. W. Stewart, *World cancer report: cancer research for cancer prevention*. IARC Press, 2020.

[12] L. Wang, X. Li, L. Zhang, and Q. Gao, "Improved anticancer drug response prediction in cell lines using matrix factorization with similarity regularization," *BMC cancer*, vol. 17, no. 1, pp. 1–12, 2017.

[13] W. Yang, J. Soares, P. Greninger, E. J. Edelman, H. Lightfoot, S. Forbes, N. Bindal, D. Beare, J. A. Smith, I. R. Thompson, *et al.*, "Genomics of drug sensitivity in cancer (gdsc): a resource for therapeutic biomarker discovery in cancer cells," *Nucleic acids research*, vol. 41, no. D1, pp. D955–D961, 2012.

[14] M. Manica, A. Oskooei, J. Born, V. Subramanian, J. Sáez-Rodríguez, and M. Rodriguez Martinez, "Toward explainable anticancer compound sensitivity prediction via multimodal attention-based convolutional encoders," *Molecular Pharmaceutics*, vol. 16, no. 12, pp. 4797–4806, 2019.

[15] Y. Chang, H. Park, H.-J. Yang, S. Lee, K.-Y. Lee, T. S. Kim, J. Jung, and J.-M. Shin, "Cancer drug response profile scan (cdrscan): a deep learning model that predicts drug effectiveness from cancer genomic signature," *Scientific reports*, vol. 8, no. 1, pp. 1–11, 2018.

[16] A. Oskooei, J. Born, M. Manica, V. Subramanian, J. Sáez-Rodríguez, and M. R. Martínez, "Paccmann: Prediction of anticancer compound sensitivity with multi-modal attention-based neural networks," *arXiv preprint arXiv:1811.06802*, 2018.

[17] M. Tsubaki, K. Tomii, and J. Sese, "Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences," *Bioinformatics*, vol. 35, no. 2, pp. 309–318, 2019.

[18] Y.-C. Chiu, H.-I. H. Chen, T. Zhang, S. Zhang, A. Gorthi, L.-J. Wang, Y. Huang, and Y. Chen, "Predicting drug response of tumors from integrated genomic profiles by deep neural networks," *BMC medical genomics*, vol. 12, no. 1, pp. 143–155, 2019.

[19] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] T. Yuasa, S. Takahashi, K. Hatake, J. Yonese, and I. Fukui, "Biomarkers to predict response to sunitinib therapy and prognosis in metastatic renal cell cancer," *Cancer science*, vol. 102, no. 11, pp. 1949–1957, 2011.

[22] K. M. Schmainda, M. Prah, J. Connelly, S. D. Rand, R. G. Hoffman, W. Mueller, and M. G. Malkin, "Dynamic-susceptibility contrast agent mri measures of relative cerebral blood volume predict response to bevacizumab in recurrent high-grade glioma," *Neuro-oncology*, vol. 16, no. 6, pp. 880–888, 2014.

[23] T. Imamura, K. Kinugawa, S. Minatsuki, H. Muraoka, N. Kato, T. Inaba, H. Maki, T. Shiga, M. Hatano, A. Yao, *et al.*, "Urine osmolality estimated using urine urea nitrogen, sodium and creatinine can effectively predict response to tolvaptan in decompensated heart failure patients," *Circulation Journal*, vol. 77, no. 5, pp. 1208–1213, 2013.

[24] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[25] O. Méndez-Lucio, B. Baillif, D.-A. Clevert, D. Rouquié, and J. Wichard, "De novo generation of hit-like molecules from gene expression signatures using artificial intelligence," *Nature communications*, vol. 11, no. 1, pp. 1–10, 2020.

[26] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2020.

[27] B. Sanchez-Lengeling and A. Aspuru-Guzik, “Inverse molecular design using machine learning: Generative models for matter engineering,” *Science*, vol. 361, no. 6400, pp. 360–365, 2018.

[28] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato, “Sample-efficient optimization in the latent space of deep generative models via weighted retraining,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.

[30] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, JMLR Workshop and Conference Proceedings, 2012.

[31] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010.

[32] P. Baldi and K. Hornik, “Neural networks and principal component analysis: Learning from examples without local minima,” *Neural Networks*, vol. 2, no. 1, pp. 53–58, 1989.

[33] C. M. Bishop, “Pattern recognition,” *Machine learning*, vol. 128, no. 9, 2006.

[34] N. Jaques, S. Gu, R. E. Turner, and D. Eck, “Tuning recurrent neural networks with reinforcement learning,” 2017.

[35] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” *arXiv preprint arXiv:1508.06576*, 2015.

[36] M. J. Kusner and J. M. Hernández-Lobato, “Gans for sequences of discrete elements with the gumbel-softmax distribution,” *arXiv preprint arXiv:1611.04051*, 2016.

[37] R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood-Forsythe, H. S. Chae, M. Einzinger, D.-G. Ha, T. Wu, *et al.*, “Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach,” *Nature materials*, vol. 15, no. 10, pp. 1120–1127, 2016.

[38] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*, pp. 1263–1272, PMLR, 2017.

[39] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *arXiv preprint arXiv:1509.09292*, 2015.

[40] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.

[41] G. Landrum *et al.*, "Rdkit: Open-source cheminformatics," 2006.

[42] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[43] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[44] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[45] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," *Journal of chemical information and modeling*, vol. 55, no. 11, pp. 2324–2337, 2015.

[46] Q. Liu, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Constrained graph variational autoencoders for molecule design," *arXiv preprint arXiv:1805.09076*, 2018.