Deep Learning for Anti-Cancer Drug Response Prediction

Sameer Khurana
MIT CSAIL
skhurana@mit.edu

Wentao Huang
MIT Biology
wentaoh@mit.edu

Abstract

In this work we propose a deep learning based anti-cancer drug response prediction model. We use three sets of input features to the prediction model: a) Cancer cell line's mutation profile, b) Cancer Cell line's gene expression profile and c) Anti Cancer drug's molecular fingerprint. To obviate the need of large amounts of labeled training data we use Variational AutoEncoders, that are trained in an unsupervised manner, to extract low dimensional salient features that are used as input to the supervised prediction model.

1 Introduction

Cancer is one of the most leading cause of death due to disease in the world. In 2018, an estimated 1,735,350 new cases of cancer will be diagnosed in the United States and 609,640 people will die from the disease. Many efforts have been made in finding the cause and cure for cancer. In the last century, the boom of molecular biology has helped scientists identified numerous molecular basis for cancer such mutation in the DNA sequence. Since then, various drugs have been developed based on the molecular basis of cancer, such as Gleevec which inhibits the BCR-ABL mutated kinase. As the development of next generation sequencing, we can investigate the genome and transcriptome of each cancer. The enormous amount of sequencing data also enables us to systematically define the genetic and molecular types of cancer. Based on the sequencing data and clinical drug test result, doctors can provide a more precise treatment strategy to patients. However, the number of drug and genome mutation are so large that we cannot test every combination to build a complete reference for treatment guidance. Therefore, it is still hard to give the best treatment strategy in many cases.

The objective of this work is to learn a mapping function $f_\theta(\bullet)$ from a set of input feature vectors to a scalar output. Inputs to the model are the cancer cell line's gene expression and mutation profile and the anti-cancer drug's molecular fingerprint. The output is the anti-cancer drug's sensitivity to the corresponding cell line measured in terms of IC_{50} value.

The prediction model that we mention in **Aim 1** is directly inspired by previous works [1, 2]. Our work is closest to [1], where authors propose a deep learning model that takes as input the mutation profile and anti-cancer drug's molecular fingerprint to predict the IC_{50} value. The molecular fingerprint is extracted using a comprehensive feature processing pipeline. Our prediction model is similar with one key difference. We obviate the need of a hand engineered feature pre-processing pipeline by constructing a Variational AutoEncoder (VAE) on the publicly available ZINC drug dataset.. Once trained, we use the encoder of the VAE to extract a feature representation for the anti-cancer drug molecules and use them as input the prediction model, instead of hand-engineered fingerprints. We use the JunctionTree VAE proposed in [3] for molecule representation learning.

Preprint. Work in progress.

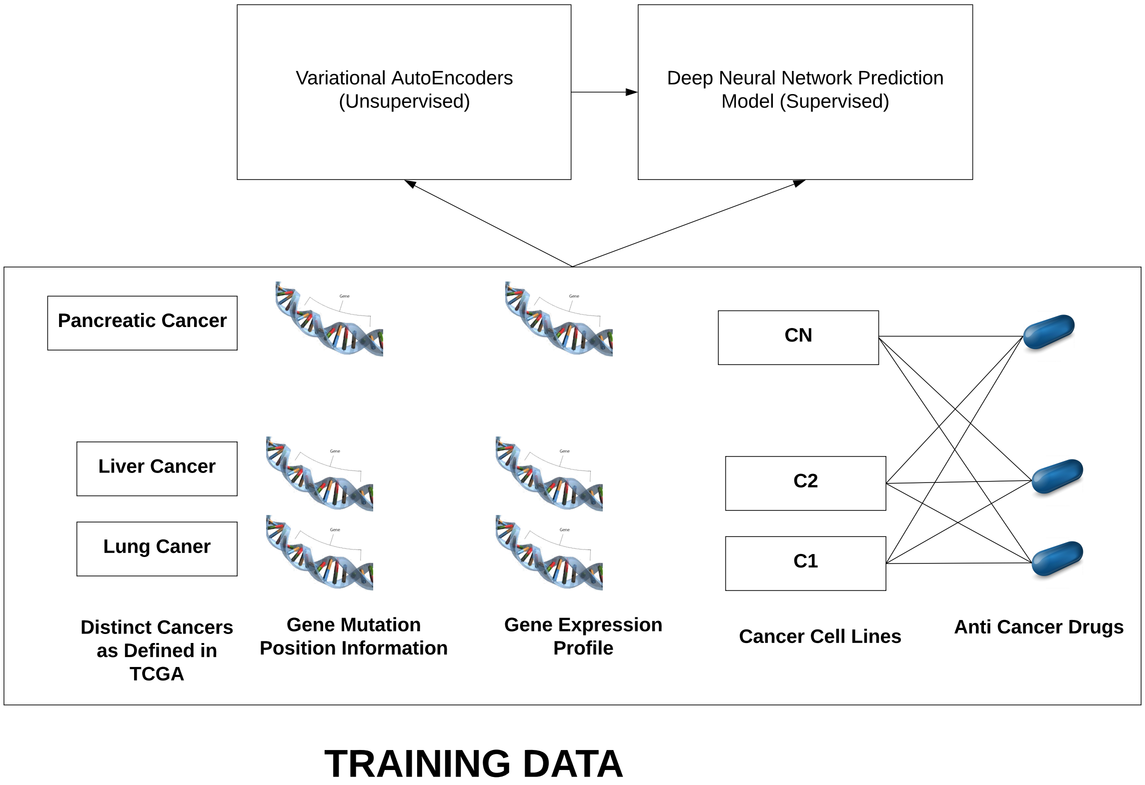
Variational AutoEncoders (Unsupervised)
Deep Neural Network Prediction Model (Supervised)
Pancreatic Cancer
Liver Cancer
Lung Caner
CN
C2
C1
Distinct Cancers as Defined in TCGA
Gene Mutation Position Information
Gene Expression Profile
Cancer Cell Lines
Anti Cancer Drugs
TRAINING DATA

Figure 1: A high level overview of our work

2 Materials and Methods

2.1 Overview

We give a high level overview of our work in Fig 1. Our work consists of first constructing Variational AutoEncoders (VAEs) [11] from gene expression, mutation and anti-cancer drug data. The encoders of trained VAEs are used to encode raw mutation, expression and drug data into a structure low-dimensional representation that is used as input to the final prediction model. Next we go into finer details of each component that is constructed as part of this work.

2.2 Data

We use the following sources of information; gene expression data of 935 cancer lines provided by the Cancer Cell Line Encyclopedia (CCLE), 11,078 TCGA pan-cancer tumors from the CTD data portal [4] and USC tumor map [5]. The gene-expression matrix is given by $E^{CCLE} \in \mathbb{R}^{g \times c}$, where g is the number of genes and c is the number of cell lines in the CCLE database. Each element of the gene-expression matrix is given by $E_{ij} = log_2(T_{g_i, c_j} + 1)$, where T is number of transcripts per million of gene g_i in cell line c_j. We also filter out genes by constructing gene-expression matrix using the TCGA database, E^{TCGA} and removing the genes (rows) with $\mu < 1$ (mean) or $\sigma < 0.5$ (standard deviation) among TCGA samples. We also use mutation information available from CCLE (1463 cell lines) [6][7] and TCGA (10,166 cell lines) databases which gives us two binary mutation matrices $M^{CCLE} \in \mathbb{R}^{g \times c}$ and $M^{TCGA} \in \mathbb{R}^{g \times t}$, where g is the number of genes, c is the number of cell lines and t is the number of tumors. Genes with no mutations in CCLE and TCGA are eliminated.

We use the drug response data of 990 cancer cell lines to 265 anti-cancer drugs measured in terms of the IC_{50} value (half maximal inhibitory concentration) available from the GDSC project. This gives us the matrix $\boldsymbol{IC}^{CCLE} \in \mathbb{R}^{d \times c}$, where d is number of drugs and c is number of cancer lines. The $(i, j)^{th}$ element of the matrix is given by $log_{10}(IC_{d_i, c_j})$.

In the end we have 622 cell lines with available expression, mutation and IC_{50} data and 9,059 tumors with expression and mutation profiles. The data preparation strategy is given in [2]

The data we used for constructing the model are from open databases. The genomic information of cancer cell line is obtained from COSMIC. We extracted the location information of the point mutation happened in each cancer cell line. In our initial model, we only incorporated the point mutation data into the input. We plan to integrate more information in our later model. The drug sensitivity data are obtained from Genomics in Drug Sensitivity in Cancer (GDSC). GDSC contains drug sensitivities data () of 1,001 human cancer cells against 265 anticancer compounds. Each drug in the database has its unique PubChem ID, which can be identified in the PubChem database. PubChem contains abundant information about each drug, including chemical structure. In order to input the structure features of each drug into our model, we used the simplified molecular-input line entry system (SMILES) to represent drugs' structure. The SMILES representation of each drug can also be obtained from PubChem. I n summary, we have three types of input data matrix. The first matrix is the cancer cell line and point mutation location. The second matrix is the value for every pair of cancer cell line and drug. The last matrix is the SMILES code for each drug. Those data are then subjected to autoencoder conversion.

2.3 Variational AutonEncoders

2.3.1 Background

Variational Inference Approximate inference techniques can be categorized into sampling based methods such as *Markov Chain Monte Carlo* and *Variational Methods* [8]. In this paper we use a Variational Method which is discussed in some detail below. For sampling-based methods readers are referred to some excellent work presented in [9, 10].

Variational inference turns the problem of inference into optimization. In Variational inference we approximate the intractable posterior distribution p with a simpler distribution $\boldsymbol{q}$, parameterized by $\boldsymbol{\phi}$. Different values of $\boldsymbol{\phi}$ denote different members of the family $\boldsymbol{q}$; $\boldsymbol{q}$ is called the *variational family* and $\boldsymbol{\phi}$ are the *variational free parameters*. The optimization objective is then to find the member of the family $\boldsymbol{q}$ that is closest to the true posterior $\boldsymbol{p}$. Closeness between the two distributions is measured using the Kullback–Leibler (KL) divergence between the two distributions [8]. Formally, we can write the optimization objective in terms of KL:

$$\boldsymbol{q}^{\star}(\boldsymbol{h}) = \underset{\boldsymbol{q}(\boldsymbol{h})}{\arg\max}\, \mathrm{KL}(\boldsymbol{q}(\boldsymbol{h}) \;||\; \boldsymbol{p}(\boldsymbol{h}|\boldsymbol{x})) \tag{1}$$

which is equivalent to writing in terms of the free parameter $\boldsymbol{\phi}$:

$$\boldsymbol{\phi}^{\star} = \underset{\boldsymbol{\phi}}{\arg\max}\, \mathrm{KL}(\boldsymbol{q}_{\boldsymbol{\phi}}(\boldsymbol{h}) \;||\; \boldsymbol{p}(\boldsymbol{h}|\boldsymbol{x})) \tag{2}$$

where, $\boldsymbol{h}$ is the latent space and $\boldsymbol{x}$ the observation space.

Using the formula for $\mathrm{KL}(\boldsymbol{q}||\boldsymbol{p}) = \boldsymbol{q}(\boldsymbol{h})\ \boldsymbol{log}\frac{\boldsymbol{q}(\boldsymbol{h})}{\boldsymbol{p}(\boldsymbol{h}|\boldsymbol{x})}$ and the fact that $\mathrm{KL} > 0$, it is straightforward to show that minimizing KL is equivalent to maximizing the lower bound on the model likelihood $\boldsymbol{p}(\boldsymbol{x})$, also known as the model evidence. This objective function is popularly know as the Evidence Lower Bound or ELBO [11] and is given by:

$$\boldsymbol{\mathcal{L}}(\boldsymbol{\phi}, \boldsymbol{\theta}) = -\boldsymbol{E}_{\boldsymbol{q}}\left[\boldsymbol{log}\, \boldsymbol{p}_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{h})\right] + \mathrm{KL}(\boldsymbol{q}_{\boldsymbol{\phi}}(\boldsymbol{h})||\boldsymbol{p}(\boldsymbol{h})) \tag{3}$$

$\boldsymbol{\mathcal{L}}$ is maxed when the reconstruction loss, $-\boldsymbol{E}_{\boldsymbol{q}}\left[\boldsymbol{log}\, \boldsymbol{p}(\boldsymbol{x}|\boldsymbol{h})\right]$ and the KL term is minimized. The KL term acts as a regularization that encourages $\boldsymbol{q}$ to be diverse [8].

Variational AutoEncoder (VAE) Variational AutoEncoder is a probabilistic latent variable model that consists of a deep neural network encoder and decoder. Encoder learns a mapping from observation space to the latent space while the decoder learns a mapping from latent to observation space. The latent-space is lower dimensional than the observation. A VAE allows us to extract a lower dimensional structured representation from our data. See Fig 3. Formally, a VAE is a probabilistic model with the joint density:

$$\boldsymbol{p}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{p}(\boldsymbol{z})\boldsymbol{p}(\boldsymbol{x}|\boldsymbol{z}) \tag{4}$$

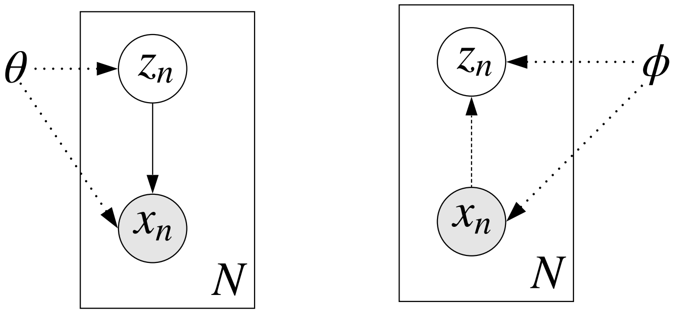

Figure 2: Graphical Model representation of a Variational AutoEncoder (VAE). On the left is the generative mode with one latent variable $\boldsymbol{z_n}$ for each data point x_n. On the right is the inference model where the local latent variable $\boldsymbol{z_n}$ is estimated using the corresponding data point $\boldsymbol{x_n}$. $\boldsymbol{\theta, \phi}$ are the parameters that are learned during training. N denotes the total number of data points.

We model each of the terms in the R.H.S of the above equation as follows:

$$\boldsymbol{p(z)} \;=\; \boldsymbol{\mathcal{N}(0,\, I)} \tag{5}$$

$$\boldsymbol{p(x|z)} \;=\; \boldsymbol{\mathcal{N}(f_\theta^{\mu}(z), f_\theta^{\sigma^2}(z))} \tag{6}$$

where, $\boldsymbol{f_\theta}$ is the decoder which is a deep neural network that takes $\boldsymbol{z}$ as input and outputs the parameters of the likelihood function $\boldsymbol{p(x|z)}$

The goal is to find the posterior distribution $\boldsymbol{p(z|x)}$. The posterior distribution is intractable to compute exactly and hence, we approximate with a simpler distribution $\boldsymbol{q(z|x)}$. Formally:

$$\boldsymbol{q(z|x) = \mathcal{N}(g_\phi^{\mu}(x), g_\phi^{\sigma^2}(x))} \tag{7}$$

where, $\boldsymbol{g_\phi}$ is a deep neural network encoder with parameters ϕ

2.3.2 Mutation VAE

The mutation VAE consists of a single latent variable $\boldsymbol{z_n}$ for each data point. It is the same structure as given in Fig 3. The encoder and decoder functions are both 2 layered feed-forward neural network, with 256 hidden units in each hidden layer. The input layer has 18,281 units which is equal to the number of input features. The output layer is also 18,281 units because the goal of VAE is to reconstruct the input.

2.3.3 Expression VAE

The Expression VAE consists of a single latent variable $\boldsymbol{z_n}$ for each data point. It is the same structure as given in Fig 3. The encoder and decoder functions are both 2 layered feed-forward neural network, with 256 hidden units in each hidden layer. The input layer has 15,363 units which is equal to the number of genes. The output layer is also 15,363 units because the goal of VAE is to reconstruct the input.

2.3.4 Molecule VAE

The molecule VAE is the Junction Tree VAE presented in [3]. The VAE is trained on the ZINC Molecule dataset. The Junction Tree VAE takes as input the molecular graph and decomposes it into a junction tree, where each node corresponds to a structural unit. The encoder takes in the junction tree and the decoder generates one structural unit at a time. This strategy of decoding structural units differentiate a Junction Tree VAE from the previous works. We do not go into excruciating detail of the Junction Tree VAE in this work, but a high level cartoon sketch of the VAE is given in Fig 3

2.4 Drug Response Prediction Model

The cartoon sketch of our final model is given in Fig 4. The model consists of two components; 1) Front end encoders used for feature extraction and 2) Back-end deep neural network model that

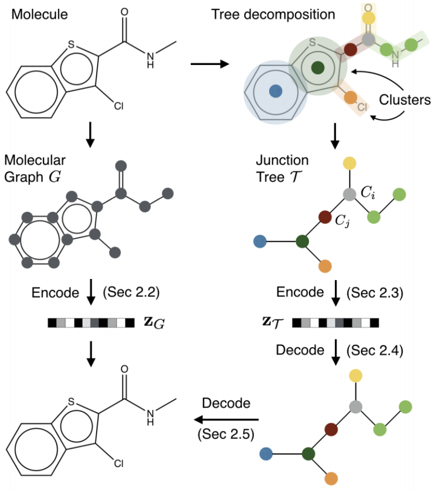

Figure 3: Molecule VAE as given in [3]

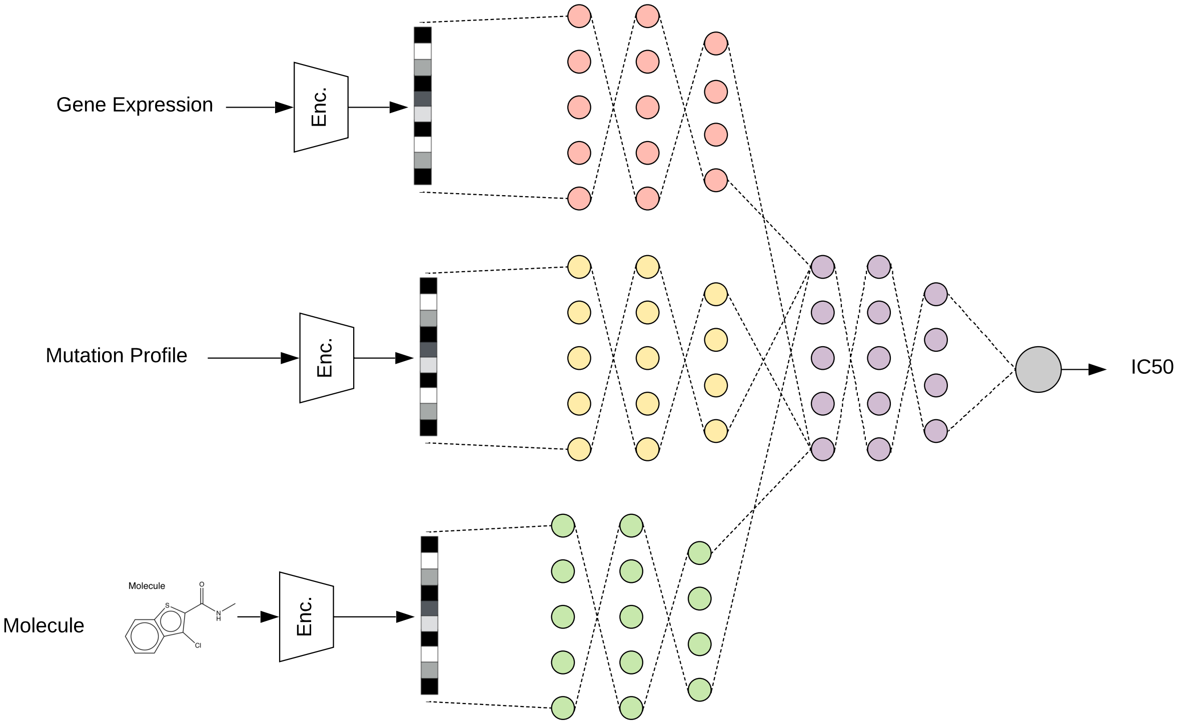

Figure 4: A cartoon sketch of our drug response prediction model. is given in Fig **??** Raw features are transformed to a low dimensional representations using the pre-trained corresponding Encoder of the VAE. These features are used as input to a supervised deep learning prediction model to predict the IC_{50} value

takes in the extracted features and maps them to a continuous value that corresponds to the IC_{50} value of the the anti-cancer drug. For the prediction model we use three different feed-forward neural networks, one for each of the three different input features. The three modules connect to a common feed-forward neural network that combines the features from the previous three modules and maps it to the output value. See Fig 4 for detailed explanation of the architecture. All the three modules (yellow, green and pink in Fig 4) are 3 hidden layered feed-forward neural network with 256, 256 and 64 hidden units. For the combiner module (purple in Fig 4) takes as input the concatenated output from the previous three modules, so the input layer of the combiner module is 192 hidden units and the other two hidden layers are 192 and 64 respectively. The final gray unit is just a linear unit that outputs a continuous value.

2.5 Model Training

Model is trained to reduce the mean squared error between the actual IC_{50} value and the one outputted by the model. The model is trained for several epochs using Adam optimizer which depends on various hyper-parameters:

- Learning rate: the step size that the optimizer should take in the parameter space while updating the model parameters.
- Batch size: the number of training examples to consider before updating the parameters
- Maximum epochs: the total number of iterations over the training set

We preset learning rate to 0.01, maximum number of epochs to 50, and the optimal value of the batch size tuned on the cross-validation set is found to be 64.

2.6 Evaluation Metric

We compute the Pearson Correlation (R^2) between the model output and the actual output to evaluate the performance of our models.

3 Results

We train simple linear regression and support vector machines for drug-response prediction to use as baselines for our deep learning model. To train the baseline models we use *sklearn* [12]. For constructing the deep learning models we use *PyTorch*.

In table 1 we present five models: 1) **LR:** Linear Regression model trained on raw gene expression and mutation features while using the encoded features for the drug molecule. 2) **SVM** Support vector machine acting on the same features as LR. 3) **DNN_i** The deep neural network acting on the same features as LR. 4) **DNN_ii** Deep Neural network trained on raw feature for mutation profile and encoded features for expression profile and drug molecule. 5) **DNN_iii** Deep neural Network trained on encoded features for all three inputs.

4 Discussion

In this work, we present a machine learning framework for anti-cancer drug response prediction model. We construct gene mutation VAE, gene expression VAE and molecule VAE that are used as front-end feature extractors for the backend prediction model. We show that using encoded features from the pre-trained VAE can significantly improve the prediction performance. We do not reach the state-of-the-art performance as presented in CDRScan [1] but get quite close. We believe that playing with other neural network architectures such as *convolutional neural network* [13] as used in CDRScan can improve the results further.

5 Future Work

In the future we want to extend our model for drug discovery. This was the main reason of using a generative model such as a Variational AutoEncoder. A generative model allows us to go in the

Model	R^2
LR	0.57
SVM	0.63
DNN_i	0.71
DNN_ii	0.79
DNN_iii	0.81
CDRScan [1]	0.84

Table 1: Different models and the corresponding Pearson Correlation on the test set

reverse direction from latent to the data space allowing us to generate new molecules. This capability of inference in both directions is not possible in predictive models that are currently proposed in the literature. Clinical data indicates multiple drugs' treatment could have better outcome than single drug, we can also extend our model to predict the most effective drug combinations

References

[1] Yoosup Chang, Hyejin Park, Hyun-Jin Yang, Seungju Lee, Kwee-Yum Lee, Tae Soon Kim, Jongsun Jung, and Jae-Min Shin. Cancer drug response profile scan (cdrscan): A deep learning model that predicts drug effectiveness from cancer genomic signature. *Scientific reports*, 8(1): 8857, 2018.

[2] Yu-Chiao Chiu, Hung-I Harry Chen, Tinghe Zhang, Songyao Zhang, Aparna Gorthi, Li-Ju Wang, Yufei Huang, and Yidong Chen. Predicting drug response of tumors from integrated genomic profiles by deep neural networks. *arXiv preprint arXiv:1805.07702*, 2018.

[3] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[4] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods*, 14(4):417, 2017.

[5] Yulia Newton, Adam M Novak, Teresa Swatloski, Duncan C McColl, Sahil Chopra, Kiley Graim, Alana S Weinstein, Robert Baertsch, Sofie R Salama, Kyle Ellrott, et al. Tumormap: exploring the molecular similarities of cancer samples in an interactive portal. *Cancer research*, 77(21):e111–e114, 2017.

[6] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603, 2012.

[7] Cancer Cell Line Encyclopedia Consortium, Genomics of Drug Sensitivity in Cancer Consortium, et al. Pharmacogenomic agreement between two cancer cell line data sets. *Nature*, 528 (7580):84, 2015.

[8] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[9] Iain Murray. *Advances in Markov chain Monte Carlo methods*. PhD thesis, Gatsby computational neuroscience unit, University College London, 2007.

[10] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Sameer Khurana, Reda Rawi, Khalid Kunji, Gwo-Yu Chuang, Halima Bensmail, and Raghvendra Mall. Deepsol: a deep learning framework for sequence-based protein solubility prediction. *Bioinformatics*.