

Chapter 6

BotGNN as a System Component: An Application to Drug Design*

So far in the dissertation, we have focussed on developing techniques for including domain-knowledge into deep neural networks, and on investigating—using large scale experiments—if that improves the predictive performance of the deep neural network. For the network types and techniques we propose, we have found that inclusion of domain-knowledge can indeed improve the performance of deep-networks significantly. In this chapter, we show how such a domain-enriched deep neural network can be used as a component in a large engineered system. We focus on a problem in drug-design concerned with the generation of new molecules for potential new drugs. Given the results from previous chapters, we will use BotGNNs as the deep neural network technique with domain-knowledge.

6.1 The Problem

The development of a new drug is difficult, wasteful, costly, uncertain, and long [CHBP02, Hai10]. AI techniques have been trying to change this [SWP⁺20], especially in the early stages culminating in “lead discovery”. Figure 6.1 shows the steps involved in this stage of drug-design. In the figure, library screening can be either done by actual laboratory experiments (high-throughput screening) or computationally (virtual screening). This usually results in many false-positives. Hit Confirmation refers to additional assays designed to reduce false-positives. QSAR (quantitative- or qualitative structure-activity relations) consists of models for predicting biological activity using physico-chemical properties of hits. The results of prediction can result in additional confirmatory assays for hits, and finally in one or more “lead” compounds that are taken forward for pre-clinical

*The content of this chapter is based on the following:

T. Dash, A. Srinivasan, L. Vig, A. Roy, “Using domain-knowledge to assist lead discovery in early-stage drug design”, *International Conference on Inductive Logic Programming*, 2021; https://doi.org/10.1007/978-3-030-97454-1_6.

testing. This chapter focuses on the problem of lead discovery that goes beyond the efficient identification of chemicals within the almost unlimited space of potential molecules. This space has been approximately estimated at about 10^{60} molecules. A very small fraction of these have been synthesised in research laboratories and by pharmaceutical companies. An even smaller number are available publicly: the well-known ChEMBL database [GHN⁺17] of drug-like chemicals consists of about 10^6 molecules. Any early-stage drug-discovery pipeline that restricts itself to in-house chemicals will clearly be self-limiting. This is especially the case if the leads sought are for targets in new diseases, for which very few “hits” may result from existing chemical libraries. While a complete (but not exhaustive) exploration of the space of 10^{60} molecules may continue to be elusive, we would nevertheless like to develop an effective way of sampling from this space.

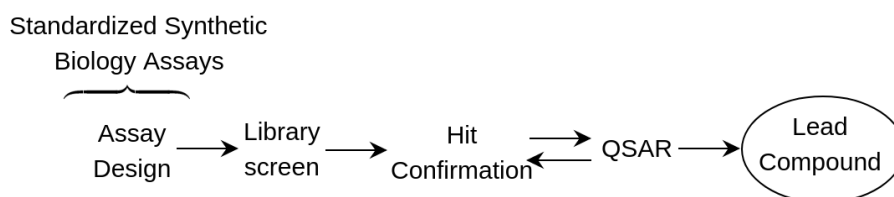


Figure 6.1: Early-stage drug-design (adapted from [WBS⁺15]).

We would like to implement the QSAR module as a generator of new molecules, conditioned on the information provided by the hit assays, and on domain-knowledge. Our position is that inclusion of domain-knowledge allows the development of more effective conditional distributions than is possible using just the hit assays. Figure 6.2 is a diagrammatic representation of an ideal conditional generator of the kind we require. The difficulty of course is that none of the underlying distributions are known. In this chapter, we describe a neural-symbolic implementation to construct approximations for the distributions.

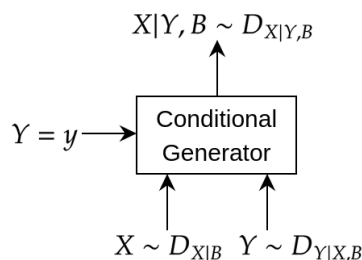


Figure 6.2: An ideal conditional generator for instances of a random-variable denoting data (X) given a value for a random-variable denoting labels (Y) and domain-knowledge (B). Here, $Z \sim D$ denotes a random variable Z is distributed according to the distribution D . If the distributions shown are known, then a value for X is obtainable through the use of Bayes rule, either exactly or through some form approximate inference.

We are interested in generating new small molecules which could act as inhibitors of a biological target, when there is limited prior information on target-specific inhibitors.

This form of drug-design is assuming increasing importance with the advent of new disease threats for which known chemicals only provide limited information about target inhibition [MR19, PIT18]. The initial studies were focused on exploring vast yet unexplored chemical space for a better screening library. In [SKTW17], a recurrent neural network (RNN) based generative model was trained with a large set of molecules and then fine-tuned with small sets of molecules, which are known to be active against the target. Some other works focus on drug-like property optimization, which helped in biasing the models to generate molecules with specific biological or physical properties of interest. Deep reinforcement learning has been very effective in constructing generative models that could generate novel molecules with the target properties [PIT18, BMO⁺21, SFK⁺19]. The efficiency of these kinds of models to generate chemically valid molecules with optimized properties has improved significantly [KBBR21, BMO⁺21]. There are also attempts to build molecule generation models against novel target proteins, where there is a limited ligand dataset for training the model [BKBR21]. Recurrent Neural Networks (RNNs) are a popular choice for molecule generation. For example, [GMLS20] propose a bidirectional generative RNN, that learns SMILES strings in both directions allowing it to better approximate the data distribution. Attention-based sequence models such as transformers have recently been used for protein-specific molecule generation [Gre21]. There are also generative models, for instance, masked graph modelling in [MMBC21], that attempts to learn a distribution over molecular graphs allowing it to generate novel molecule without requiring to deal with sequences. Although the works referred here are shown to be effective, they are often purely deep neural network-based. It is well-known that such deep generative models are data-hungry, and require a large-amount of data (100s of 1000s) of data instances to be able to generate novel data-instances. Furthermore, there has been very little or no attempt in incorporating domain-knowledge into deep neural networks for molecule generation.

In this chapter we investigate the construction of a deep generative model for molecules that can utilise available domain-knowledge. The overall contributions of this chapter are the following: (1) We propose a system consisting of two deep generative models (or generators) and a discriminative model (or discriminator) working in collaboration with each other for conditional generation of novel molecules; (2) We propose a methodology to allow the generators to access the available domain-knowledge for molecule generation; (3) We investigate our approach using the well-studied problem of inhibitors for the Janus kinase (JAK) class of proteins [Wil89]. We assume first that if no data on inhibitors are available for a target protein (JAK2), but a small number of inhibitors are known for homologous proteins (JAK1, JAK3 and TYK2); and (4) We show that the inclusion of relational domain-knowledge results in a potentially more effective generator of inhibitors than simple random sampling from the space of molecules or a generator without access to symbolic relations. We also show how samples from the conditional generator can be

used to identify potentially novel target inhibitors.

6.2 System Design and Implementation

We implement an approximation to the ideal conditional generator using a combination of two generators and a discriminator (see Figure 6.3). We have decomposed the domain-knowledge B in Figure 6.2 into constraints relevant just to the molecule-generator B_G and the knowledge relevant to the prediction of activity B_D (that is, $B = B_G \cup B_D$, and $P(X|B) = P(X|B_G)$ and $P(Y|X, B) = P(Y|X, B_D)$). The discriminator module approximates the conditional distribution $D_{Y|X,B}$, and the combination of the unconditional generator and filter approximates the distribution $D_{X|B}$. The conditional generator then constructs an approximation to $D_{X|Y,B}$. For the present, we assume the unconditional generator and discriminator are pre-trained: details will be provided below. The discriminator is a BotGNN (refer Chapter 5). This is a form of graph-based neural network (GNN) that uses graph-encodings of most-specific clauses (see [Mug95]) constructed using symbolic domain-knowledge B_D .

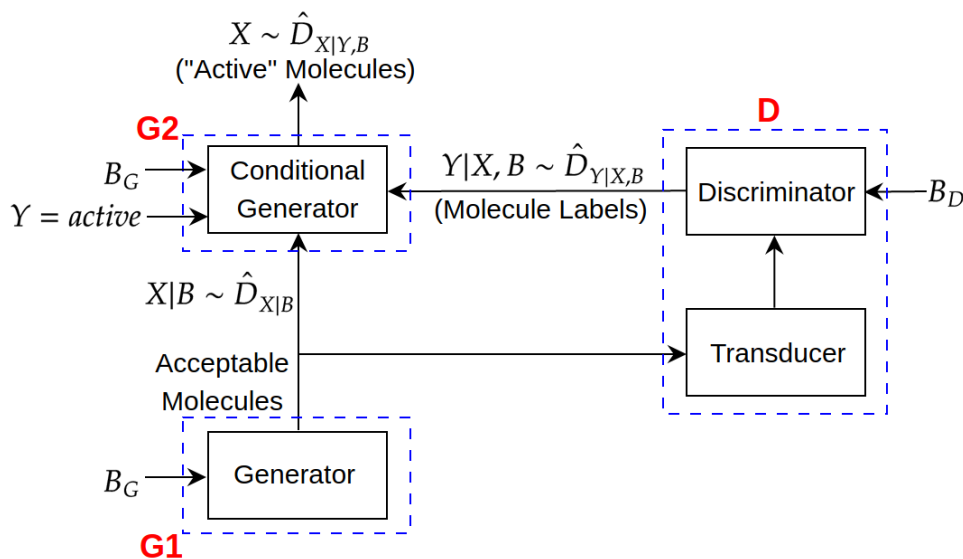


Figure 6.3: Training a conditional generator for generating “active” molecules. For the present, we assume the generator (G1) and discriminator (D) have already been trained (the G1 and D modules generate acceptable molecules and their labels respectively: the \hat{D} ’s are approximations to the corresponding true distribution). The Transducer converts the output of G1 into a form suitable for the discriminator. Actual implementations used in the chapter will be described below.

The generator-discriminator combination in Figure 6.3 constitutes the QSAR module in Figure 6.1. An initial set of hits is used to train the discriminator. The conditional generator is trained using the initial set of hits and the filtered samples from the unconditional generator and the labels from the discriminator. Although out of the scope

of this chapter, any novel molecules generated could then be synthesised, subject to hit confirmation, and the process repeated.

6.2.1 Generating Acceptable Molecules

The intent of module G1 is to produce an approximation to drawing samples (in our case, molecules) from $D_{X|B_G}$. We describe the actual B_G used for experiments in [subsection 6.3.1](#). For the present it is sufficient to assume that for any instance $X = x$, if $B_G \wedge X = x \models \square$ then $P(x|B_G) = 0$. Here, we implement this by a simple rejection-sampler that first draws from some distribution over molecules and rejects the instances that are inconsistent with B_G .

For drawing samples of molecules, we adopt the text-generation model proposed in [\[BVV⁺16\]](#). Our model takes SMILES representations [\[Wei88\]](#) of molecules as inputs and estimates a probability distribution over these SMILES representations. Samples of molecules are then SMILES strings drawn from this distribution.

The SMILES generation module is shown in [Figure 6.4](#). The distribution of molecules (SMILES strings) is estimated using a variational autoencoder (VAE) model. The VAE model consists of an encoder and a decoder, both based on LSTM-based RNNs [\[HS97\]](#). This architecture forms a SMILES encoder with the Gaussian prior acting as a regulariser on the latent representation. The decoder is a special RNN model that is conditioned on the latent representation.

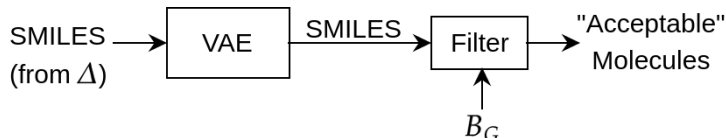


Figure 6.4: Training a generator for acceptable molecules. Training data consists of molecules, represented as SMILES strings, drawn from a database Δ . The VAE is a model constructed using the training data and generates molecules represented by SMILES strings. B_G denotes domain-knowledge consisting of constraints on acceptable molecules. The filter acts as a rejection-sampler: only molecules consistent with B_G pass through.

The architecture of the VAE model is shown in [Figure 6.5](#). The SMILES encoding involves three primary modules: (a) embedding module: It constructs an embedding for the input SMILE; (b) highway module: It constructs a gated information-flow module based on highway network [\[SGS15\]](#); (c) LSTM module: It is responsible for dealing with sequences. The modules (b) and (c) together form the encoder module. The parameters of the Gaussian distribution is learnt via two fully-connected networks, one each for μ and σ , which are standard sub-structures involved in a VAE model. The decoder module consists of LSTM layers followed by a fully-connected (FC) layer. We defer the details on architecture-specific hyperparameters to [subsection 6.3.2](#). The loss function used for

training our VAE model is a weighted version of the reconstruction loss and the KL-divergence between VAE-constructed distribution and the Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

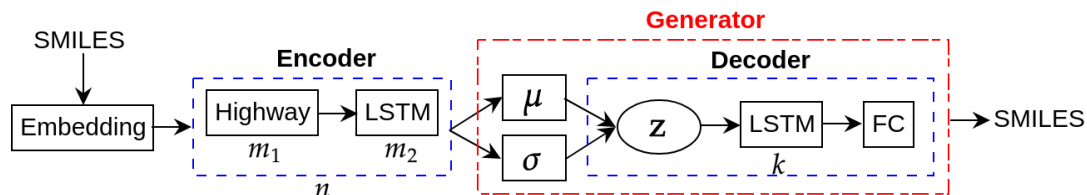


Figure 6.5: Architecture of the VAE in Figure 6.4. m_1, m_2, n, k denote the number of blocks. The decoder along with the μ and σ constitute the generator that generates molecules in SMILES representation.

6.2.2 Obtaining Labels for Acceptable Molecules

The intent of module D is to produce an approximation to draw samples (in our case, labels for molecules) from $D_{Y|X, B_D}$. We describe the actual B_D used for experiments in subsection 6.3.1. The discriminator in D is a BotGNN (see, Chapter 5), which is a form of graph neural network (GNN) constructed from data (as graphs) and background knowledge (as symbolic relations or propositions) using mode-directed inverse entailment (MDIE [Mug95]). In this work, data consists of graph-based representations of molecules (atoms and bonds), and B_D consists of symbolic domain-relations applicable to the molecules. The goal of the discriminator is to learn a distribution over class-labels for any given molecules. Figure 6.6 shows the block diagram of the discriminator block.

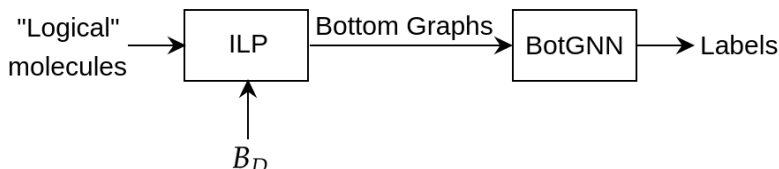


Figure 6.6: Discriminator based on BotGNN. “Logical” molecules refers to a logic-based representation of molecules. Bottom-graphs are a graph-based representation of most-specific (“bottom”) clauses constructed for the molecules by an ILP implementation based on mode-directed inverse entailment.

6.2.3 Generating Active Molecules

The intent of module G2 is to produce an approximation to drawing from $D_{X|Y, B}$. That is, we want to draw samples of molecules, given a label for the molecule and domain-knowledge B . We adopt the same architecture as the generator used for drawing from $D_{X|B_G}$ above, with a simple modification to the way the SMILES strings are provided as inputs to the model. We prefix each SMILES string with a class-label: $y = 1$ or

$y = 0$ based on whether the molecule is an active or inactive inhibitor, respectively. The VAE model is also able to accommodate any data that may already be present about the target, or about related targets (it is assumed that such data will be in the form of labelled SMILES strings).

6.3 System Testing

Our aim is to perform a controlled experiment to assess the effect on system performance of the inclusion of high-level symbolic domain-knowledge. Specifically, we investigate the effect on the generation of new inhibitors for the target when (a) no domain-knowledge is available in the form of symbolic relations (but some knowledge is available in a propositional form); and (b) some domain-knowledge is available in the form of symbolic relations. We intend to test if the system is able to generate possible new inhibitors in case (a); and if the performance of the system improves in case (b).

6.3.1 Materials

Data

The data used are as follows: (a) ChEMBL dataset [GHN⁺17]: 1.9 million molecules in SMILES representation; used to train the generator for legal molecules (G1); (b) JAK2 [KBBR21]: 4100 molecules (3700 active) in SMILES representation; used to test the conditional generator (G1) and to build the proxy model for hit confirmation (see Method section below); (c) JAK2 Homologues (JAK1, JAK3 and TYK2) [KBBR21]: 4300 molecules (3700 active) in SMILES representation; used to train the discriminator (D) and train the conditional generator (G2).

Domain-Knowledge

The domain constraints in B_G are in the form of constraints on acceptable molecules. These constraints are broadly of two kinds: (i) Those concerned with the validity of a generated SMILES string. This involves various syntax-level checks, and is done here by the RDKit molecular modelling package; (ii) Problem-specific constraints on some bulk-properties of the molecule. These are: molecular weight is in the range (200, 700), the octanol-water partition coefficients (logP) must be below 6.0, and the synthetic accessibility score (SAS) must be below 5.0. We use the scoring approach proposed in [ES09] to compute the SAS of a molecule based on its SMILES representation.

The domain-knowledge in B_D broadly divides into two kinds: (i) Propositional, consisting of the following bulk-molecular properties: molecular weight, logP, SAS, number of hydrogen bond donors (HBD), number of hydrogen bond acceptor (HBA), number of

rotatable bonds (NRB), number of aromatic rings (NumRings), Topological Polar Surface Area (TPSA), and quantitative estimation of drug-likeness (QED); (ii) Relational, which is a collection of logic programs (written in Prolog) defining almost 100 relations for various functional groups (such as amide, amine, ether, etc.) and various ring structures (such as aromatic, non-aromatic, etc.). More details on this background knowledge can be found in [Chapter 3](#).

Algorithms and Machines

We use the following softwares for our system implementation: (a) RDKit [[L⁺06](#)]: Molecular modelling software used to compute molecular properties and check for the validity of molecules; (b) Chemprop [[SYS⁺20](#)]: Molecular modelling software used to build a proxy model for hit confirmation; (c) Transducer: In-house software to convert representation from SMILES to logic; (d) Aleph [[Sri01](#)]: ILP engine used to generate most-specific clauses for BotGNN; (e) BotGNN: Discriminator for acceptable molecules capable of using relational and propositional domain-knowledge; (f) VAE [[KW14](#)]: Generative deep neural network used for generators. We used PyTorch for the implementation of BotGNN and VAE models, and Aleph was used with YAP.

Our experimental works were distributed across two machines: (a) The discriminator (D) was built on a Dell workstation with 64GB of main memory, 16-core Intel Xeon 3.10GHz processors, an 8GB NVIDIA P4000 graphics processor; (b) The generators (G1, G2) are built on an NVIDIA-DGX1 station with 32GB Tesla V100 GPUs, 512GB main memory, 80-core Intel Xeon 2.20GHz processors.

6.3.2 Method

We describe the procedure adopted for a controlled experiment comparing system performance in generating potential inhibitors when (a) domain-knowledge is restricted to commonly used bulk-properties about the molecules; and (b) domain-knowledge includes information about higher-level symbolic relations consisting of ring-structures and functional groups, along with the information in (a). In either case, the method used to generate acceptable molecules (from module G1 in [Figure 6.3](#)) is the same.

Let B_0 denote domain-knowledge consisting of bulk-molecular properties used in the construction of QSARs for novel inhibitors, B_1 denote the definitions in B_0 along with first-order relations defining ring-structures and functional-groups used in the construction of QSAR relations, and B_G denote the domain-knowledge consisting of constraints on acceptable molecules (see "Domain-Knowledge" in [subsection 6.3.1](#)). Let D_{Tr} denote the data available on inhibitors for JAK1, JAK3 and TYK2; and D_{Te} denote the data available on inhibitors for JAK2 (see "Data" in [subsection 6.3.1](#)). Let Δ denote a database of (known) legal molecules. Our methodology is straightforward.

- (1) Construct a generator for possible molecules given Δ (the generator in module G1 of Figure 6.3).
- (2) For $i = 0, 1$
 - (a) Let $E_0 = \{(x, y)\}_1^{|D_{Tr}|}$, where x is a molecule in D_{Tr} and y is the activity label obtained based on a threshold θ on the minimum activity for active inhibition;
 - (b) Let $B_D = B_i$;
 - (c) Construct a discriminator (for module D in Figure 6.3) using E_0 and the domain-knowledge B_D (see section 6.2);
 - (d) Sample a set of possible molecules, denoted as N , from the generator constructed in Step (1);
 - (e) Let $N' \subseteq N$ be the set of molecules found to be acceptable given the constraints in B_G (that is, N' is a sample from $\hat{D}_{X|B_G}$);
 - (f) For each acceptable molecule x obtained in Step (2)d above, let y be the label with the highest probability from the distribution $\hat{D}_{Y|X,B}$ constructed by the discriminator in Step (2)c. Let $E = \{(x, y)\}_1^{|N'|}$;
 - (g) Construct the generator model (for module G2 in Figure 6.3) using $E_0 \cup E$;
 - (h) Sample a set of molecules, denoted as M_i , from the generator in Step (2)g;
 - (i) Let $M'_i \subseteq M_i$ be the set of molecules found to be acceptable given the constraints in B_G (that is, M'_i is a sample from $D_{Y|X,B_G}$);
- (3) Assess the samples M_0, M_1 obtained in Step (2)h above for possible new inhibitors of the target, using the information in D_{Te} .

The following details are relevant:

- For experiments here Δ is the ChEMBL database, consisting of approximately 1.9 million molecules. The generator also includes legality checks performed by the RDKit package, as described in section 6.2.
- Following [KBBR21], $\theta = 6.0$. That is, all molecules with pIC50 value ≥ 6.0 are taken as “active” inhibitors;
- The discriminator in Step (2)c is a BotGNN. We follow the procedure and parameters described in Chapter 5 to construct BotGNN. We use GraphSAGE [HYL17] for the convolution block in the GNN (Refers to variant 4 in Chapter 5). This is based on the results shown in Chapter 5 for the inclusion of symbolic domain-knowledge for graph-based data, such as molecules.

- The generators in Steps (1) and (2)g are based on the VAE model described earlier. The hyperparameters are as follows: vocabulary length is 100, embedding-dimension is 300, number of highway layers is 2, number of LSTM layers in the encoder is 1 with hidden size 512, and the type is bidirectional, number of LSTM layers in the decoder is 2, each with hidden size 512, dimension of the latent representation (\mathbf{z}) is 100.
- To make our generator robust to noise and to be generalised, we also use a word-dropout technique. This technique is identical to the standard practice of dropout in deep learning [SHK⁺14] except that here the tokens to the decoder are replaced by ‘*unknown*’ tokens with certain probabilities. Here we call it the word-dropout rate and fix it at 0.5.
- The reconstruction loss coefficient is 7. We use cost-annealing [BVV⁺16] for the KLD-coefficient during training. We use the Adam optimizer [KB15] with learning rate of 0.0001; training batch-size is 256.
- In Step (2)d, $|N| = 30,000$. The B_G provided here results in $|N'| = 18,000$;
- In Step (2)h, $|M_0| = |M_1| = 5000$.
- The acceptable molecules M'_0, M'_1 after testing for consistency with B_G are assessed along the following two dimensions:
 - (a) *Activity*: In the pipeline described in Figure 6.1 assessment of activity would be done by *in vitro* by hit confirmation assays. Here we use a proxy assessment for the result of the assays by using an *in silico* predictor of pIC50 values constructed from the data in D_{Te} on JAK2 inhibitors. The proxy model is constructed by a state-of-the-art activity prediction package (Chemprop [SYS⁺20]: details of this are provided in section B.3).¹ We are interested in comparing the proportions of generated molecules predicted as “active”;
 - (b) *Similarity*: we want to assess how similar the molecules generated are to the set of active JAK2 inhibitors in D_{Te} .² A widely used measure for this is the Tanimoto (Jaccard) similarity: molecules with Tanimoto similarity > 0.75 are usually taken to be similar. We are interested in the proportion of molecules generated that are similar to known target inhibitors in D_{Te} ;

¹Such a model is only possible in the controlled experiment here. In practice, no inhibitors would be available for the target and activity values would have to be obtained by hit assays, or perhaps *in silico* docking calculations.

²Again, this is feasible in the controlled experiment here. In practice, we will have no inhibitors for the target, and we will have to perform this assessment on the data available for the target’s homologues (D_{Tr}).

Each sample of molecules M_i drawn from the conditional generator can therefore be represented by a pair (a_i, b_i) denoting the values of the proportions in (a) and (b), and (c) above. We will call this pair the “performance summary” of the set M_i .

- We compare performance summaries of sets of molecules in two ways. First, a performance summary $P_i = (a_i, b_i)$ can be compared against the performance summary $P_j = (a_j, b_j)$ in the obvious lexicographic manner. That is, P_i is better than P_j if $[(a_i > a_j)]$ or $[(a_i = a_j) \wedge (b_i > b_j)]$. Secondly, since all the elements of a performance summary are proportions, we are able to assess if the differences in corresponding values are statistically significant. This is done using a straightforward hypothesis test on proportions. Given an estimate p of a proportion of n instances, the distribution of proportions is approximately Normal, with mean p and s.d. $\sigma = \sqrt{\frac{p(1-p)}{n}}$. For testing the hypothesis $p_j < p_i$ at a 95% confidence level the critical value from tables of the standard normal distribution is 1.65. That is, if $p_j < 1.65\sigma$ we will say the difference is statistically significant at the 95% level of confidence.

6.3.3 Results

A summary of the main results obtained is in [Figure 6.7](#). The principal points in this tabulation are these: (1) The performance of the system with $B_D = B_1$ is better than with $B_D = B_0$ or simple random draw of molecules; and (2) The differences in proportions for Activity and Similarity are statistically significant at the 95% confidence level. Taken together, these results suggest that the inclusion of symbolic relations can make a significant difference to the performance of the generation of active molecules.

We turn next to some questions of relevance to these results:

Better Discriminators? A question arises on whether the differences in proportions would be different if we had compared against a different discriminator capable of using $B_D = B_0$. Since B_0 is essentially propositional in nature, any of the usual statistical discriminative approaches could be used. We have found replacing the BotGNN with an MLP with hyper-parameter tuning resulted in significantly worse performance than a BotGNN with $B_D = B_1$. We conjecture that similar results will be obtained with other kinds of statistical models. On the question of whether better discriminators are possible for $B_D = B_1$, we note results in [Chapter 5](#) show BotGNNs performance to be better than techniques based on propositionalisation or a direct use of ILP. Nevertheless, better BotGNN models than the one used here may be possible. For example, we could construct an activity prediction model for the JAK2 homologues using a state-of-the-art predictor like Chemprop. The prediction of this model could be used as an additional molecular property by the BotGNN.

Better Generators? Our generators are simple language models based on variational

Qty.	$B_D = B_1$	$B_D = B_0$	<i>Random</i>
$ M $	5000	5000	5000
$ M' $	2058	2160	2877
<i>Act</i>	0.47 (0.01)	0.43 (0.01)	0.34 (0.01)
<i>Sim</i>	0.14 (0.01)	0.11 (0.01)	0.00 (0.00)

Figure 6.7: Summary of system performance. $B_D = B_1$ denotes that the discriminator has access to both propositional and relational domain-knowledge; $B_D = B_0$ denotes that the discriminator has access to propositional domain-knowledge only. *Random* denotes a random draw of molecules from the unconditional molecule generator G1. M denotes the set of molecules drawn (from the conditional generator, or from the unconditional generator for *Random*). The results are compared against the performance of a methodology purely based on Deep Reinforcement Learning [KBBR21]. M' denotes the set of acceptable molecules generated in the sample of M molecules (acceptable molecules satisfy molecular constraints defined on molecular properties). *Act* denotes the proportion of M' that are predicted active (the proxy model predicts an $\text{pIC}_{50} \geq 6.0$); *Sim* denotes the proportion of M' that are similar to active target inhibitors (Tanimoto similarity to active JAK2 inhibitors > 0.75). The numbers in parentheses denote the standard deviation in the corresponding estimate.

auto-encoders. Substantial improvements in generative language models (for example, the sequence models based on attention mechanism [DCLT19, RWC⁺19]) suggest that the generator could be much better. In addition, the rejection-sampling approach we use to discard sample instances that fail constraints in B_G is inherently inefficient, and we suggest that the results here should be treated as a baseline. The modular design of our system-design should allow relatively easy testing of alternatives.

Related to the question of discriminators is the role of ILP in this work. ILP is used to include domain-knowledge in the construction of the BotGNN discriminator. How important was this use of ILP? A quantitative answer is difficult, but we are able to provide indirect, qualitative evidence for the utility of ILP by comparison against a recent result on the same data in [KBBR21]. That work differs from the one here in the following ways: (a) No symbolic domain-knowledge is used in the discrimination step; and (b) Substantially more computation is involved in developing the final generator—the equivalent of module G2 here—through the use of reinforcement learning (RL). The principal concern in [KBBR21] is to generate molecules similar to the active inhibitors for JAK2, and the approach results in 5% of the sampled molecules being similar. The corresponding results here are significantly higher: 14% (with $B_D = B_1$) and 11% ($B_D = B_0$). Both results were obtained with BotGNNs, without requiring the additional episodic training characteristic of RL. Therefore, we believe BotGNNs have played an important role, both in prediction and in easing computation. Since ILP is necessary for the construction of a

BotGNN, their importance to the current system-design follows.³

Finally, we consider how samples from the conditional generator can be used to identify potential molecules for synthesis and testing in hit-confirmation assays. We propose a selection based on a combination of (predicted) activity and similarity to the existing inhibitors (when these are unavailable, we would have to rely on models constructed with the target’s homologues). Using these measures, there are two surprising subsets of molecules. Molecules in S are those that are similar to JAK2 inhibitors (Tanimoto similarity > 0.75), but have a low predicted activity (substantially lower than 6.0); and molecules in \bar{S} are significantly different to the JAK2 inhibitors (Tanimoto similarity < 0.5), but have a high predicted activity (substantially higher than 6.0).⁴ For the sample in this chapter, $S = \emptyset$. However, $\bar{S} \neq \emptyset$ and can provide interesting candidates for novel inhibitors. We exemplify this with a chemical assessment of 3 elements from \bar{S} . This is shown in [Figure 6.8](#). Molecule 1562 is identified as a possible candidate for synthesis and hit confirmation.

6.4 Summary

In this chapter, we have shown how a deep neural network capable of including domain-knowledge can be used as part of a larger system designed for a purpose other than simple discrimination. As an example, we have considered the design of a system generating novel molecules for early-stage drug design. We show how a graph-based neural network with domain-knowledge is one component in a modular system design. Specifically, our conclusions from this chapter are as follows: (1) We have constructed a complete end-to-end neural-symbolic system that is capable of generating active molecules that may not be in any existing database; (2) We have demonstrated usage of the system on the classic chemical problem on Janus kinase inhibitors. Importantly, working with a computational chemist, we have shown how the system can be used to discover an active molecule based on entirely new scaffolds; (3) The results reaffirm the conclusions from our [Chapter 5](#) that inclusion of relational domain-knowledge through the use of ILP techniques can significantly improve the performance of deep neural networks. Further, our system design is intentionally modular, to allow “plug-and-play” of discriminators and generators, allowing faster experimentation with better modules.

³Could we have directly used ILP for constructing the discriminator? Yes, but we draw attention to evidence from the previous chapter that suggests that BotGNNs result in discriminators that are at least comparable, and possibly better than those from the direct use of ILP.

⁴A good reason to consider dissimilar molecules is that it allows us to explore more diverse molecules.

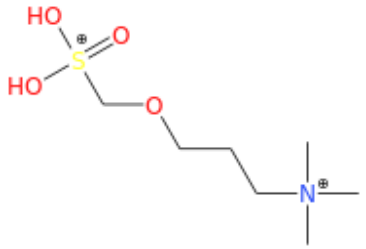
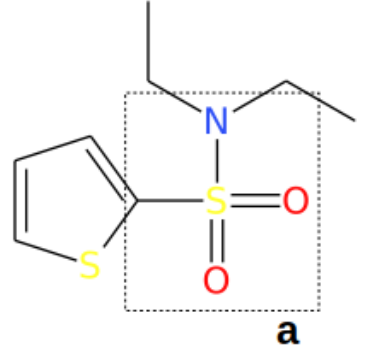
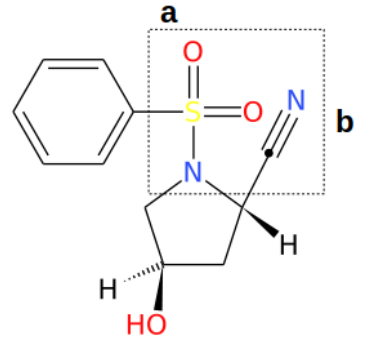
ID	Structure	Descriptors	Assessment
551		$Act = 9.12$ $Sim = 0.15$	This molecule has very low similarity to known JAK2 inhibitors. Also none of the groups specific to JAK2 could be identified by the substructure search. Discard this molecule.
1548		$Act = 9.04$ $Sim = 0.22$	This molecule has very low similarity to known JAK2 inhibitors. Also none of the groups specific to JAK2 could be identified by the substructure search. However, the sulfonamide group commonly found in JAK family inhibitors was found to be present (highlighted).
1562		$Act = 9.49$ $Sim = 0.32$	Despite low similarity to existing JAK2 inhibitors, 1562 had one JAK2-selective subgroup and a group common to JAK inhibitors, indicating potential to act as JAK family inhibitor, but the selectivity to JAK2 cannot be confirmed. Possibly interesting new scaffold (highlighted) and worth pursuing further.

Figure 6.8: A chemical assessment of possible new JAK2 inhibitors. The molecules are from the sample of molecules from the conditional generator, that are predicted to have high JAK2 activity, and are significantly dissimilar to known inhibitors. The assessment is done by a computational chemist[†]. The assessment uses structural features and functional groups identified for the JAK2 site in the literature [KBBR21, DS13, DYCFY14].

[†]Dr. Arijit Roy, TCS Innovation Labs, Hyderabad.