Automatic design of novel potential 3CL^{pro} and PL^{pro} inhibitors

Timothy Atkinson Saeed Saremi Jonathan Masci

NNAISENSE S.A.
June 2020

Abstract

With the goal of designing novel inhibitors for SARS-CoV-1 and SARS-CoV-2, we bring together tools from graph neural networks (GNN), energy-based models, and Monte Carlo tree search. We train an ensemble of GNNs to accurately predict inhibition of 3CL^{pro} and PL^{pro} proteases, which are key to viral reproduction. We then learn an energy function parametrized by a neural network on the feature space of the trained GNN with the methodology of empirical Bayes. These two neural models are combined into a reward function that guides Monte Carlo tree search towards molecules with a high probability of inhibition that are at the same time statistically close to known molecules according to the energy function. Combining these three distinct branches of machine learning into a single framework, we explore over 40 million molecules and identify 120K potential novel inhibitors.

1 Introduction

Over the past few months, the search for molecules which may inhibit key receptor sites of Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) has emerged as a central research objective within the scientific community [21]. The already widespread use of Deep Learning (DL) techniques as predictors in molecular chemistry [14, 16] has facilitated their rapid redeployment to this task e.g. [3, 7, 52] (see [31] for a comprehensive review). There are a number of structural similarities between both the main 3CL^{pro} protease and the PL^{pro} protease of SARS-Cov-2, both used in reproduction, and those of Severe Acute Respiratory Syndrome Coronavirus-1 (SARS-CoV-1) [33]. Based on this, it has been proposed that the lack of available large-scale data for SARS-Cov-2 inhibition can be overcome using existing datasets of SARS-CoV-1 inhibitors [18].

We propose a general data-driven automatic molecule design framework consisting of three core components (Figure 1):

- *Inhibition Predictor*: A predictive model of molecular activity which is used as a guide to identify new molecules with desirable properties. In this work, a deep neural network is trained to predict inhibition of SARS-CoV-1 proteases.
- *Energy Model*: A general probabilistic model that learns the distribution of molecules in an unsupervised manner. The model assign a scalar (the energy) to any molecule and effectively provides a measure of statistical similarity between the new molecules and the molecules in the dataset.
- *Directed Search*: A directed search over the space of possible molecules. Guided by the inhibition predictor and the energy model, this discovers new molecules to which the neural models assign a high likelihood of inhibition with a high similarity to known molecules.

In the particular instantiation of the framework used in this work, the Inhibition Predictor uses Graph Neural Networks (GNN; [2, 5, 17, 42]), the Energy Model uses Deep Energy Estimator Networks [40, 41], and Monte Carlo Tree Search (MCTS; [6]) is used to search over derivations of molecules in a Backus-Naur form grammar [29]. However, we stress that this only one possible implementation of this general framework. For example, the Inhibition Predictor could instead use LSTMs to predicting behaviour from

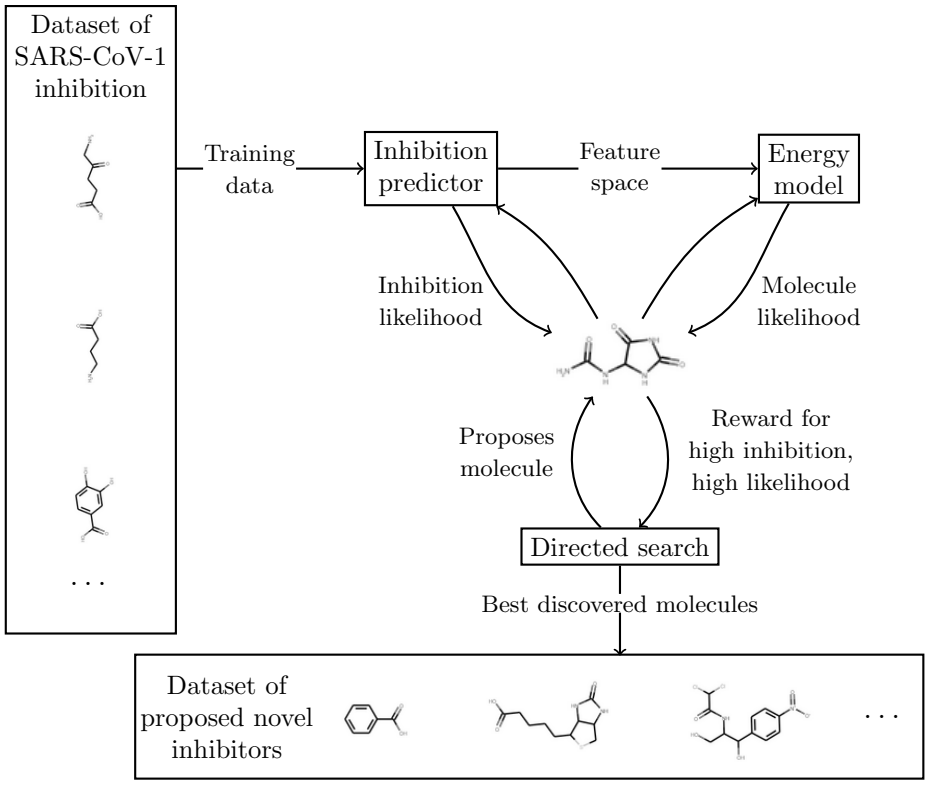


Figure 1: An overview of the proposed approach.

SMILES strings [34], DEEN could be replaced with a variational autoencoder [25], and MCTS could be replaced with genetic algorithms [22] or deep reinforcement learning [44].

Existing approaches to identification of SARS-CoV-2 inhibitors and reactants generally use DL to predict inhibition in ordr to screen sets of known inhibitors [3, 13, 18, 46, 50]. In contrast, we propose a *de novo* approach to identifying and designing novel inhibitors for SARS-CoV-2. There are two main motivations for doing this:

1. The combinatorial space of small drug-like molecules is huge (approx. 10^{33} [37]) so that it is unlikely that even datasets with billions of molecules will contain the best SARS-CoV-2 inhibitors. Therefore, directed search through the full space may yield higher-quality results.
2. It allows the system to discover potential inhibitors which are previously unknown molecules and therefore not patented. If any discovered molecule is eventually tested and found viable for treatment, this may substantially improve the general availability of treatment.

Unlike other de novo approaches which utilize docking simulations [9] or structural information [45], the proposed approach maximizes the output of a neural model trained to predict SARS-CoV-1 inhibition. The use of a predictive network greatly reduces the cost of sampling so that millions of candidate molecules can be explored efficiently.

The rest of this work is organized as follows. In Section 2, the four datasets used to train a GNN-based Inhibition Predictor are described. The core concepts of GNNs and the specific architecture used are described in Section 3. A brief overview of neural empirical Bayes which is used to learn an energy function on the feature space of the GNN (i.e. the Energy Model) is provided in Section 4. The directed search over molecules, guided by these neural models, is described in Section 5. The performance of the neural models and the results of inhibitor design experiments are provided in Section 6. Finally, our findings are summarized in Section 7.

Assay ID	Target protease	#Inactive molecules	#Active molecules
1,706	$3CL^{pro}$	290,321	405
1,879	$3CL^{pro}$	244	136
485,353	PL^{pro}	322,433	602
652,038	PL^{pro}	735	198
Total	-	331,480	1,095

Table 1: **Assays used throughout experiments**. All assays are taken from PubChem [23]. The calculation of the totals takes into account that the same molecule may appear in multiple assays. The total number of active molecules is the number of molecules which are active in at least 1 assay.

2 Data

As in Hofmarcher et al. [18], four publicly available assays, taken from PubChem [23], are used. Each assay provides a set of molecules (represented by their SMILES strings [49]) and the results of tests to determine whether or not each molecule inhibits a given protease for SARS-CoV-2. Each assay can be viewed as a mapping from its associated set of molecules M_A to the binary vector $\{0, 1\}$ where 0 indicates no inhibition and 1 indicates inhibition. A summary of the assays used is provided in Table 1.

To use these assays with a graph neural network, each molecule is then formatted as a graph according to the following:

1. For each molecule, convert its SMILES representation to a molecular representation using RdKit [30].
2. For each molecular representation:
   (a) For each atom, create a node with the following features: mass, valence, the total number of Hydrogens, whether it is aromatic, formal charge.
   (b) For each bond between two atoms, create an edge in each direction between its corresponding nodes and additionally, create a self-loop on each node.[1] Each edge is categorized as one of: single bond, double bond, triple bond, aromatic bond or whether it is one of the synthetically added self-loops.
3. Each graph is assigned four binary classifications; one for each of the four assays, maintaining missing values.

3 Graph neural networks

Graph neural networks (GNN) are a relatively recent form of deep learning architecture for reasoning about structured relational data. As molecules are naturally structured, with atoms as nodes and bonds as edges, there is a clear affinity between graph neural networks and predictive molecular chemistry that has led to a number of developments in recent years [15, 19, 43].

3.1 GNN layers

While there are a variety of unique graph neural network designs (see [53]), a typical construction is a layer which takes a directed labelled multi-graph $G_K = (V_K, E_K)$ and computes new node and edge features based on the existing features and adjacency, yielding a new graph $G_{K+1} = (V_{K+1}, E_{K+1})$ with the same structure but new labels. Here, the set $V = \{v_i\}_{i=1:N_V}$ is the set of N_V nodes where each v_i is the ith node's features, and $E_K = (e_j, s_j, t_j)_{j=1:N_E}$ is the set of N_E edges where each e_j is the jth edge's features, $s_j \in V_K$ is the

[1] This is for technical reasons with the GNN; doing so ensures that each node is treated as a member of its own neighborhood.

jth edge's source node and $t_j \in V_K$ is the jth edge's target node. The transformed G_{K+1} can then be passed to another graph neural network layer, have its node features used to classify nodes [27, 36] or predict edges [26, 48], or features can be pooled and passed to some other neural network to perform classification or regression on the entire input graph [11, 51].

In the following it will be helpful to refer to the *neighborhood* of a given node v_i, e.g. the set of edges which target it. This is defined by $\mathcal{N}_i = \{j \mid e_j \in E \text{ and } t_j = v_i\}$. In this work a simple model of graph neural network layers of the form in Battaglia et al. [2] is used. To compute G_{K+1} from G_K, first compute E_{K+1}, given by $E_{K+1} = \{e'_j, s_j, t_j\}_{j=1:N_E}$, where e'_j is a function of each edge's features, source node features and target node features: $e'_j = \Phi_E(e_j, v_{s_j}, v_{t_j})$, where Φ_E is a multi-layer perceptron. Then the mean of the new edge features of the neighborhood of each node v_i is computed as,

$$\overline{\mathcal{N}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} e'_j \tag{1}$$

The new node features V_{K+1} are then computed as a function of each node's features and its mean aggregated neighborhood,

$$V_{K+1} = \{v'_i\}_{i=1:N_V}, \tag{2}$$

$$v'_i = \Phi_V(v_i, \overline{\mathcal{N}}_i), \tag{3}$$

giving the updated graph G_{K+1}. As before, Φ_V is a multi-layer perceptron. An important property of the above construction is that, because each update is in the context of a node or edge's adjacency, and the permutation invariant mean aggregation of the neighborhood is used, the entire layer is invariant to permutations of the node and edge sets. Therefore two isomorphic graphs will be updated in the same manner irrespective of the order in which the nodes (and edges) are indexed.

When classifying graphs, it is often helpful to 'coarsen' the graph by merging nodes [53, Section V.C] to reduce the number of parameters and avoid over-fitting. In this work the edge contraction pooling operator EdgePool [10] is used to achieve this effect, which is extended to support edge features. EdgePool allows the graph coarsening to be learned by maintaining learnable parameters W and b, such that a raw score can be assigned to each edge e_j from source node s_j to target node t_j as,

$$r(e_j) = W \cdot (s_j \oplus t_j \oplus e_j) + b \tag{4}$$

where $\oplus$ denotes concatenation. This is transformed into a score value $s(e_j)$ by taking the softmax of the neighborhood of source node s_j:

$$s(e_j) = \text{softmax}_{\mathcal{N}_{s_j}} r(e_j) \tag{5}$$

Edges are then iteratively contracted according to their score. That is; starting at the edge with the highest score and continuing in descending order, contract an edge if and only if its source and target nodes have not been involved in any other edge contraction. When an edge e_j is contracted, it is removed, its source node s_j and its target node t_j are merged, replacing all 3 items with a single node with the average of the node features multiplied by the score edge's associated score:

$$v'_j = s(e_j)\frac{s_j + t_j}{2}. \tag{6}$$

Any edges which were incoming or outgoing of either s_j or t_j now instead are connected to v'_j. Further, any edges which have become parallel as a result of this merging process have their features merged, again by averaging. The result of these steps is a process whereby the graph is coarsened in a learned fashion while connectivity is preserved. In contrast, in alternative learnable pooling approaches such as Top-K pooling [8] and Self-attention Graph pooling [32], the fact that nodes are dropped, rather than merged, means that connectivity is lost.

3.2 GNN architecture

We treat prediction of inhibition as multi-task and create a single network which predicts inhibition for all 4 assays. The architecture is shown in Figure 2. Initially, the input graph G is passed through 3 'GNN Layer - Edge Contraction' (GEC) blocks. Each GEC block is a graph neural network layer, consisting of node MLP Φ_V and edge MLP Φ_E, followed by edge contraction pooling. In every GEC block, both Φ_V and Φ_E are 2-layer MLPs with 96 neurons per layer, with batch normalization applied after each layer followed by ReLU activation. Each edge contraction pooling operator removes approximately half of the edges from the graph, so that the graph after the third GEC block is expected to have 1/8th of the edges of the input graph.

After each GEC block, global mean and global max pooling are applied to the 96-dimensional node features, yielding a 192 dimensional representation of the input. The 3 representations from the GEC blocks are concatenated together, yielding the 574 dimensional latent representation of the input, denoted X.

For each assay, a 'head' is used, which is a specialist MLP that predicts only inhibition for that assay. Each head is provided with the latent representation X and applies two feed-forward layers, with dropout applied before each layer ($p = 0.25$ and $p = 0.5$, respectively), and batch normalization and ReLU activation after. Each layer has 128 neurons. Then, the 128 dimensional vector is passed to a single logit neuron which gives a prediction for the given assay. The predictions are concatenated and passed through a softmax, yielding the prediction vector $M(G)$.

3.3 Inhibition loss

To handle the multi-task context while maintaining efficiency, the collective $331,480$ molecules are treated as a single dataset and are passed through the graph neural network in a mini-batch fashion. However, adjustments are made to a standard binary cross-entropy loss function to ensure that the network is not biased towards any given assay and that missing data is appropriately handled. For a given graph (molecule) G with ground truth K_A for assay A, the loss is computed for the Ath head of the model, $f_A(G)$, as

$$\ell(K_A, f_A(G)) = \begin{cases} -\alpha_A \beta_A \log\left(f_A(G)\right) & K_A = 1, \\ -\alpha_A \log\left(1 - f_A(G)\right) & K_A = 0, \\ 0 & K_A \text{ is missing}, \end{cases} \tag{7}$$

with the total loss (for one single molecule/graph) across all assays given, $\ell(K, G) = \sum_A \ell(K_A, f_A(G))$. We therefore arrive at the loss $\ell(\theta)$ that is minimized $\ell(\theta) = \mathbb{E}_{(K,G)}\, \ell(K, G)$, where θ are the parameters of the GNN, which was suppressed in Eq. 7 for a clean notation.

The factors α_A and β_A are weights towards each task and positive instances of that task, respectively to handle the extreme imbalances of the data (see Table 1). For assay A with I training inhibitors, J training non-inhibitors and N total training samples, $\alpha_A = N/(I + J)$, and, $\beta_A = (I + J)/I$. The role of $\alpha_A \geq 1$ is to ensure that each assay contributes equally to the total loss averaged across all N training samples, whereas the role of $\beta_A \geq 1$ is to ensure that positive and negative samples for each assay contribute to the total loss averaged across all $I + J$ training molecules for assay A.

3.4 Ensemble of GNNs

To further boost the performance of the model, given the relatively small number of positive samples (Table 1), a simple Bootstrap-Aggregating ensemble of models [4] is used, whose outputs are averaged to yield predictions. To build the ensemble, the training data is split into 5 non-overlapping folds of equal size. Each model is trained on a different 4 of these folds in a conventional cross-fold manner, yielding 5 models which have been trained under different conditions. For each model, the unused fold is used as a validation fold. Each model's parameters at the epoch at which the validation loss is lowest is used to construct the final ensemble.

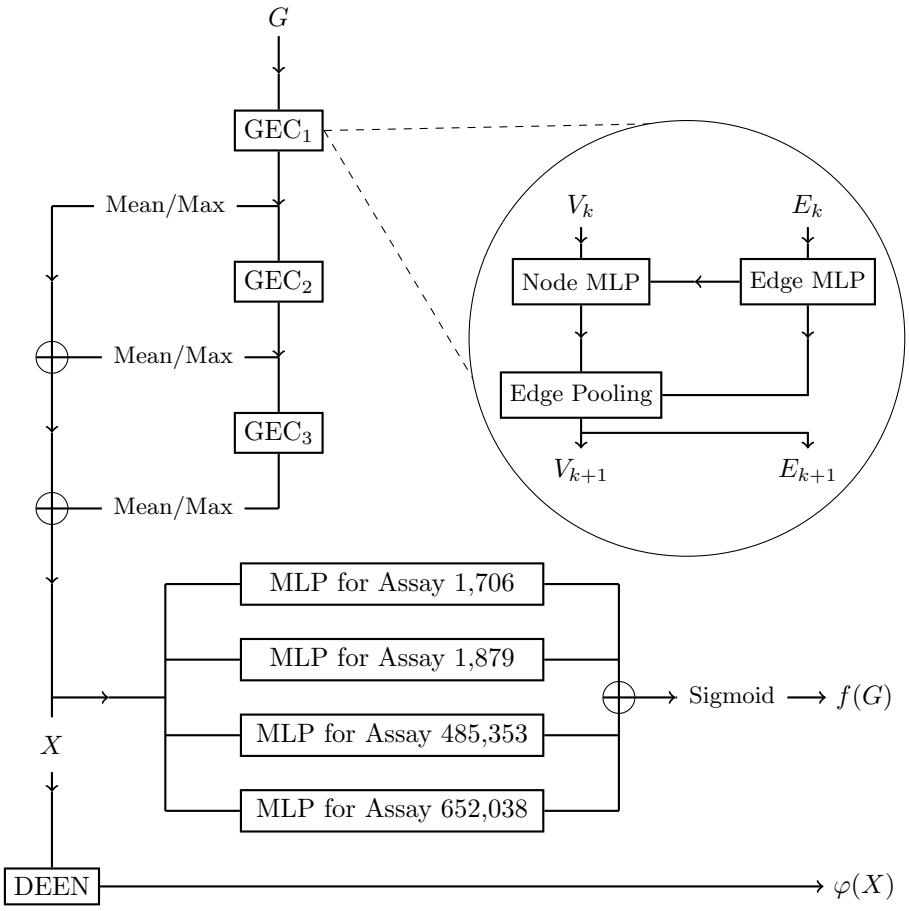


Figure 2: The multi-task graph neural network used to predict SARS-CoV-1 protease inhibition activity for molecules. G is an input graph. Each GEC block is decomposed into a graph neural network layer followed by edge contraction pooling with outputs passed to the next block. The output of each block is also passed through global mean and max pooling, with the results of all pooling concatenated together to form a vector representation of the input graph. This vector is passed through a multi-layer perceptron for each target assay, yielding a binary prediction for each task $f(G)$. Concatenation is denoted with the symbol $\bigoplus$. The extraction of the latent representation X is shown, which is then passed to the Deep Energy Estimator Network (DEEN) to generate an energy $\varphi(X)$, see Section 4.

4 Deep Energy Estimator Network

We would like to have a probabilistic model for the molecules and a statistical measure on how close/similar two molecules are based on the samples in the dataset. There are two main challenges: (i) In high dimensions, one cannot resort to classical techniques like kernel density estimation due to curse of dimensionality. In essence, the volume of space grows exponentially and nonparametric methods like kernel density estimation which are based on Euclidean similarity metric become inadequate. (ii) One typically resorts to parametrizing distributions in directed or undirected graphical models. But in both cases the inference over latent variables is in general intractable. At the root of the problem is the issue of estimating the *normalizing constant* (a.k.a. the partition function) of probability distributions in high dimensions.

There is however a class of models called *energy models* formulated around learning *unnormalized* densities or energy functions. For MCTS, since only the likelihood of generated molecules *relative* to the likelihood of the molecules in the dataset is of importance (this will become clear in the next section), the distribution needs only be known up to a constant; therefore learning energy functions is sufficient. In this section we review a latest development [40] that formulates the problem of learning energy functions in terms of the empirical Bayes methodology [35, 39]. The framework is referred to by DEEN due to its origin in Deep Energy Estimator Networks [41].

In this work the problem is further simplified by learning the energy function on the feature space $\mathcal{X} = \mathbb{R}^{d_X}$ of the GNN (here $d_X = 4018$). Therefore instead of learning a probabilistic model for the i.i.d. sequence $G_1, \ldots, G_n$, the representation of the sequence in the feature space $X_1, \ldots, X_n$ is studied instead (see Figure 2). This simplification is due to technical reasons, since the denoising methodology of empirical Bayes is formulated in the Euclidean space where specifically the isotropic Gaussian $N(0, \sigma^2 I_d)$ that defines the *noise model* plays a central role. Geometrically, the model is designed such that the negative gradient of the energy function evaluated at the noisy data is directed towards the clean data. In other words learning can be visualized as "shaping" the energy function φ such that $-\nabla\varphi$ (known as the score function [20]) points toward the data manifold (see Fig. 3 for a schematic).

Some technical aspects of DEEN are reviewed next. Consider the Gaussian noise model and corrupt the i.i.d. samples $X_1, \ldots, X_n$:

$$Y_{ij} = X_i + \varepsilon_j, \text{ where } \varepsilon_j \sim N(0, \sigma^2 I_d). \tag{8}$$

An important result in empirical Bayes is the fact that the Bayes estimator of X given a noisy measurement $Y = y$ is given by $\widehat{x}(y) = y + \sigma^2 \nabla \log p(y)$, where ∇ is the gradient taken with the respect to the noisy data y, and $p(y)$ is the probability density function of the random variable $Y = X + N(0, \sigma^2 I_d)$. The key step in DEEN is to parameterize the energy function of Y with a neural network $\varphi_\vartheta : \mathbb{R}^d \to \mathbb{R}$, where the Bayes estimator takes the parametric form:

$$\widehat{x}_\vartheta(y) = y - \sigma^2 \nabla \varphi_\vartheta(y) \tag{9}$$

Since the Bayes estimator is the least-squares estimator, the learning objective follows immediately:

$$\mathcal{L}(\vartheta) = \mathbb{E}_{(x,y)} \| x - \widehat{x}_\vartheta(y) \|_2^2, \tag{10}$$

where the expectation is over the empirical distribution over the samples (X_i, Y_{ij}) (see Equation 8) and $\| \cdot \|_2$ is the Euclidean norm. Here learning the energy function is transformed to an optimization problem which is the main appeal of the algorithm as it sidesteps posterior inference or MCMC approximations in latent variable models [47]. At optimality, given an expressive neural network, and large-enough unlabelled samples ($n \gg 1$), one is theoretically guaranteed to find a good approximation to the energy function: $\varphi(y, \vartheta^*) \approx -\log p(y)$ (modulo a constant).

In summary, in terms of learning the distribution of graph-valued molecules, there are three simplifications made:

(i) The problem is transformed to learning the distribution in the Euclidean space $\mathcal{X}$ defined by the GNN classifier. This simplification is indeed well suited here and in line with the desire for the molecules generated by MCTS to be classified as inhibitors.

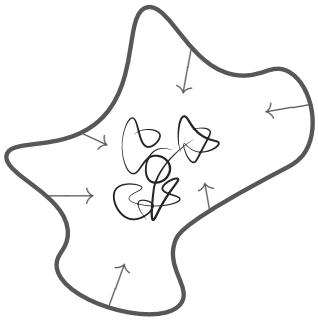


Figure 3: **DEEN schematic**. Samples from the random variable X (in the feature space of molecules) are represented in black. Noisy samples from $Y = X + N(0, \sigma^2 I_d)$ are represented in violet, and the arrows represent $-\nabla\varphi(y)$, where φ is the energy function parametrized by a neural network. DEEN's learning objective is a denoising objective rooted in empirical Bayes: the idea is to shape the energy function such that the arrows point to the data manifold.

(ii) The statistical model is un-normalized, i.e., the energy function is defined modulo an additive constant. This is allowed since only the *difference* between energies appear in the reward function (introduced next); therefore the constant cancels out.

(iii) The algorithm is designed around learning the *smoothed* density associated with noisy data with the hyperparameter σ. This relaxation is in fact key for regularizing the learning [40] but also not critical in the MCTS reward function as it will become clear.

5 Designing candidate inhibitors with MCTS

To automatically design new inhibitors, the conventional UCT variant of MCTS [28], searching over a BNF grammar of SMILES strings [29], is used. While the reader is encouraged to refer to definitive descriptions of the algorithm [6], a brief overview is provided here. MCTS is searching a tree of *derivations* i.e. sequences of steps which generate valid SMILES strings. In each iteration of MCTS, the following steps are taken:

1. *Selection*; starting from a root node with an associated initial non-terminal, successive child nodes are selected until a previously unvisited node is encountered.
2. *Expansion*; the encountered node is expanded, and one of its new children is selected at random. If the encountered node has no viable children, it represents a completed SMILES string, and is instead simply evaluated.
3. *Rollout*; a random derivation is generated for the selected child. In our case, the rollout is performed uniformly at random, except when a terminal symbol is available; in which case decisions are made uniformly at random over available terminal symbols. The random derivation yields a completed SMILES string which is then evaluated.
4. *Backpropagation*; the evaluated reward w of the SMILES string is backpropagated through the tree. Every node that was visited has it's average reward $\bar{w}$ and maximum reward $w^{\max}$ updated. Additionally, the number of visits to each visited node, n, is incremented.

These 4 steps effectively provide a guided search process over the tree of possible molecule derivations. We give a simplified visualization of this process in Figure 4. First, a derivation sequence (red) is selected until a previously unexpanded node is reached. Then, the node is expanded, and a random child node (blue) is chosen. The rollout stage completes the derivation randomly, generating a molecule G. Finally, the reward for this molecule, $w_\beta(G)$, is propagated backwards through the derivation tree, guiding future iterations.

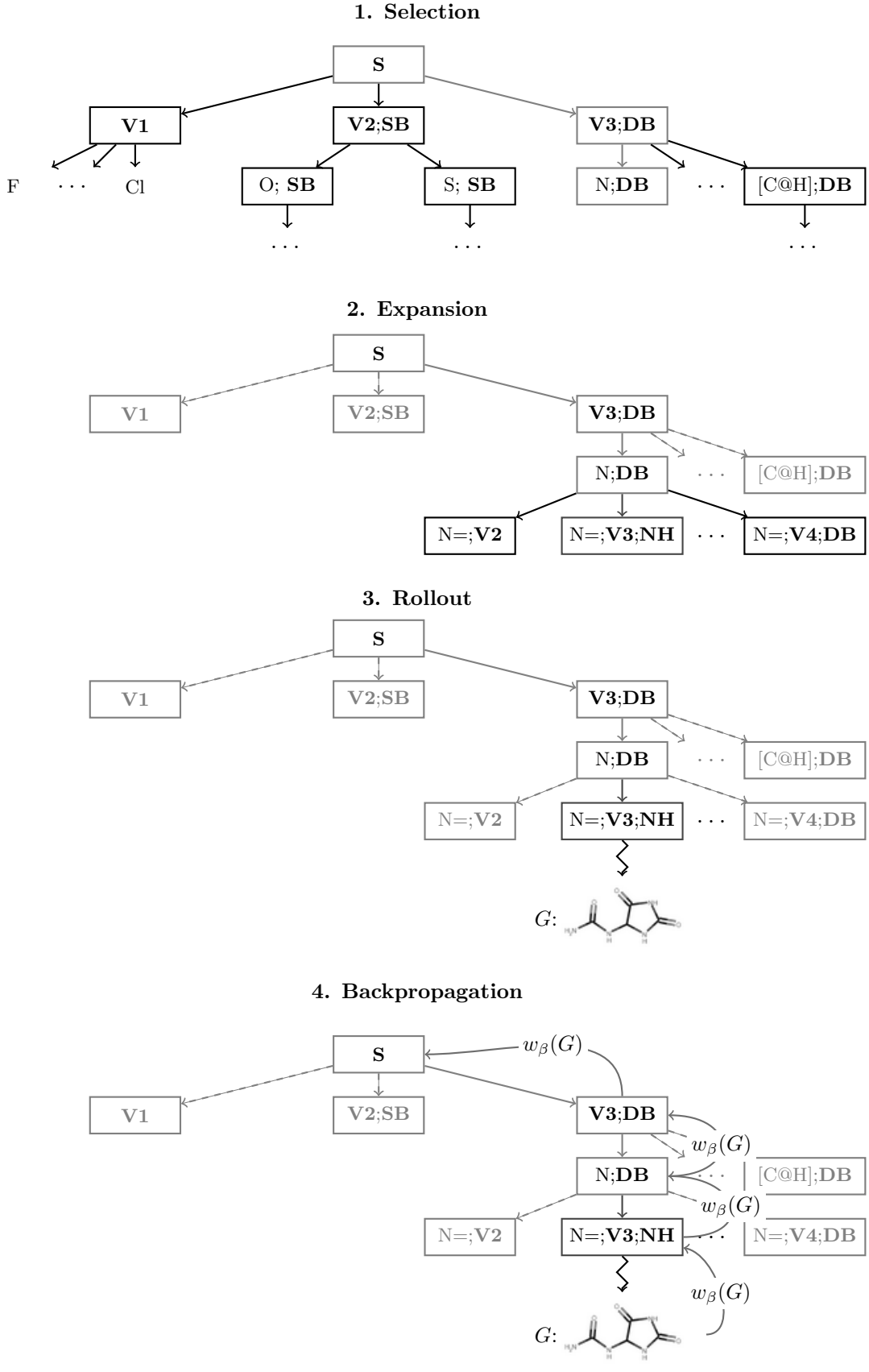


Figure 4: An overview of the use of MCTS to generate molecules maximizing the reward function.

We use a mild adjustment of the UCB1 formula [1] to determine the appropriate node/production to use in each step of the selection process. The modified UCB1 formula used is

$$\frac{w_i^{\max} + \bar{w}_i}{2} + c\sqrt{\frac{\ln N_i}{n_i}}, \tag{11}$$

where for node i in the derivation tree, w_i^{max} is the maximum reward observed, $\bar{w}_i$ is the average reward observed, n_i is the total number of samples, N_i is the total number of samples at the parent of node i and c is the exploration/exploitation parameter. The introduction of a maximum reward term w_i^{max} is simply there to help the system pursue particularly promising samples which are otherwise avoided due to a few poor-quality samples from that same node. Once all non-terminals have been cleared either by selection or rollout, a complete SMILES string is generated for evaluation. As stated, the rollout policy is random, but prioritizes terminals where possible to bias our sampling towards smaller molecules.

Every time MCTS generates a SMILES string, it is converted to a molecule and then a graph as described in Section 2. By passing the sample molecule G through the GNN a prediction of inhibition $f_A(G)$ is obtained for assay A, and a latent representation X of the molecule which is further passed through the DEEN model to yield an energy $\varphi(X)$. With $\phi_{\min}$ defined as the minimum energy for any known inhibitor in the test set, the reward returned to MCTS is then

$$w_\beta(G) \stackrel{\text{def}}{=} f_A(G) \cdot \frac{2}{1 + \exp(\beta \Delta\varphi(X))}, \text{ where } \Delta\varphi = \varphi(X) - \phi_{\min} \tag{12}$$

meaning that the sample molecule is heavily penalized for either a low inhibition prediction or a high energy. The β term is a hyperparameter controlling the smoothness of the energy component of the reward function, with a default value of

$$\beta_0 = \frac{1}{\phi_{\max} - \phi_{\min}}$$

(maximum and minimum are computed over the test set). A simple but important property of the reward function is that it is invariant under $\varphi \to \varphi + C$. The reward function must be invariant to this transformation since the energy function itself is defined modulo an additive constant.

6 Experiments

The automatic molecule design framework comprises three steps:

1. Firstly, the ensemble of GNNs is trained to predict inhibition for the four assays (§6.1).
2. DEEN is then trained on the feature space of molecules generated by the ensemble of GNNs to learn an energy model of the dataset (§6.2).
3. Finally, the ensemble of GNNs and the DEEN model are used in the reward function (Eq. 12) to guide MCTS, generating potential novel inhibitors (§6.3).

6.1 Training the GNN Ensemble

The data is split into stratified sets: 80% training, 20% testing. The training data is then split into 5 folds, as discussed in Section 3.4, and a network is trained on each subset of 4 folds, yielding 5 models. Each model is trained using the ADAM optimizer [24], with an initial learning rate of 2×10^{-5}, a batch size of 128 and weight decay of 1×10^{-4}. Each model is trained until the validation loss has not improved for 20 epochs. Here, the validation loss is computed on the unused fold for each model.

A typical plot from one of these training cycles is given in Figure 5. The lowest validation loss for this model was observed at epoch 15. We provide a full listing of ROC AUC scores of the models in Table 2, alongside

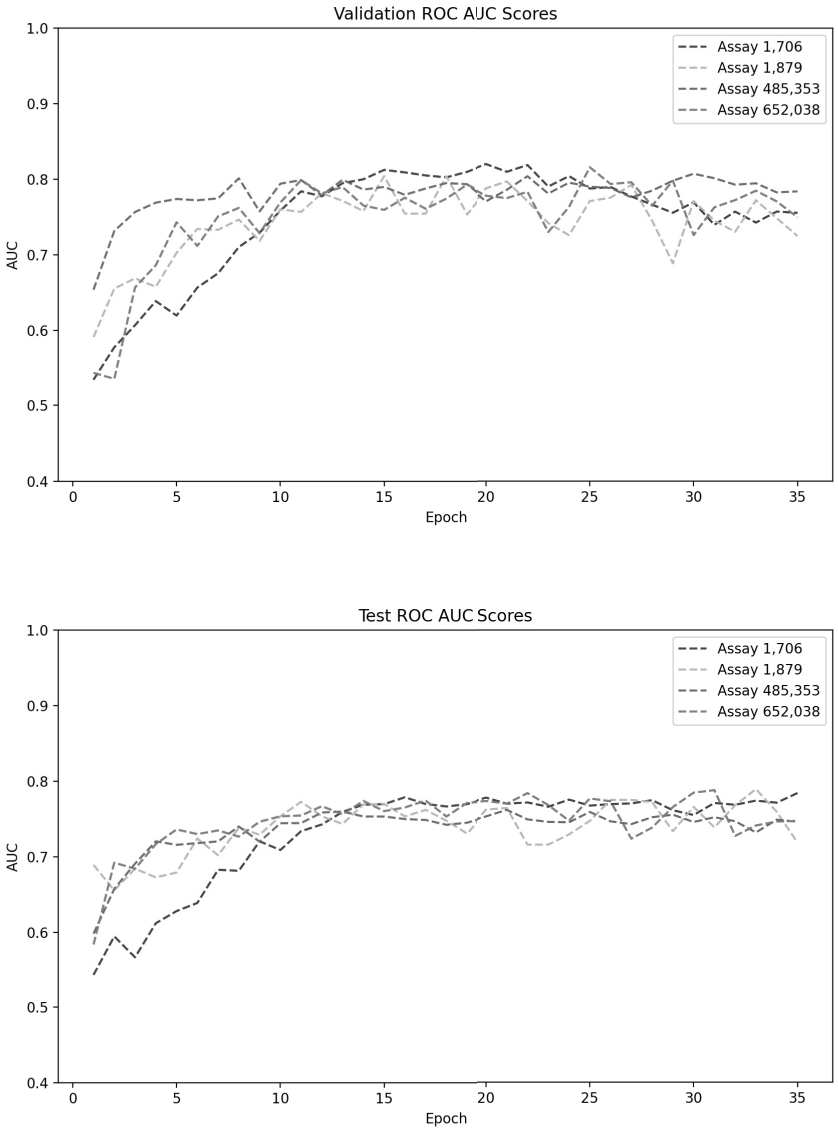


Figure 5: Validation and testing ROC AUC scores of model 0 (see Table 2), measured during training and reported for each assay. The validation loss is lowest at epoch 15.

Model	Termination Epoch	Assay ROC AUC Score			
		1,708	1,879	485,353	652,038
0	15	0.769	0.770	0.755	0.761
1	15	0.767	**0.827**	0.735	0.773
2	21	**0.774**	0.826	**0.772**	**0.793**
3	22	0.759	0.769	0.720	0.791
4	16	0.746	0.783	0.755	0.786
Average	17.8	0.763	0.795	0.748	0.781
Ensemble	-	**0.784**	**0.812**	**0.761**	**0.803**

Table 2: ROC AUC scores of the GNN ensemble computed on unseen test data. For each assay the best performing model's ROC AUC score is highlighted in **bold**. The average (mean) ROC AUC scores is provided in comparison to the ROC AUC score of the ensemble as a whole. For every assay, the ensemble performs better on unseen data than the average of its individual members. Further, in two assays (1,708 and 652,038), the ensemble performs better than *any* of the individual GNNs.

those of the ensemble constructed by averaging their outputs. The ensemble achieves higher ROC AUC scores than the average of its individual members across all assays. In general, the ensemble has a $75-80\%$ likelihood of ranking an inhibitor higher than a non-inhibitor across all tasks. Although there exists some room for further improvement, meaningful classification performance is clearly demonstrated in this very sparse task.

6.2 Training DEEN

DEEN is trained on the feature space of the training molecules with Gaussian noise with $\sigma = 0.25$. The 574-dimensional latent representations taken across all 5 members of the ensemble are concatenated to yield a single 2880-dimensional latent representation. This 2880-dimensional latent representation is identified with X in Section 4. The architecture used is a feed-forward neural network, with 4 layers with $3072, 2048$ and 1024 neurons, respectively, followed by a single linear output neuron. Each layer, except for the first and last layer, uses a skip connection so that it takes, as input, the concatenation of the two previous layers' outputs. The activation function is $u \mapsto u/(1+\exp(-u))$, a smoothed ReLU which is known by two different names: *SiLU* [12] and *Swish* [38]. The choice of a *smooth* activation function is important here[2] due to the double backpropagation (one in y to compute the loss, the other in ϑ to compute the gradient of the loss) for a single SGD update. The ADAM optimizer is again used, with a learning rate of 10^{-5} and a batch-size of 128. DEEN is trained for 200 epochs and the parameters at the final epoch are stored.

A trace of the training, validation and testing loss over time is given in Figure 6. The epoch with the lowest test loss is the final epoch; 200. Note that the test loss closely follows the training loss, suggesting that the testing data lies on the same manifold of latent representations as the training data.

6.3 Discovering novel inhibitors

For each assay, MCTS optimizes the reward function for $1,000,000$ samples. This process is repeated 10 times, yielding a total search of $10,000,000$ molecules per assay. Once MCTS has terminated, the top $30,000$ unique sampled molecules are identified for each assay. Here uniqueness is determined by comparing canonical SMILES strings. Additionally, to bias towards smaller molecules, any derivation branch which reaches a depth ≥ 30 immediately prioritizes terminals to pursue termination of the derivation. The exploration constant c is set to $\sqrt{2}$.

[2]With ReLU, the optimizer saturates at significantly higher loss (compared to SiLU/Swish).

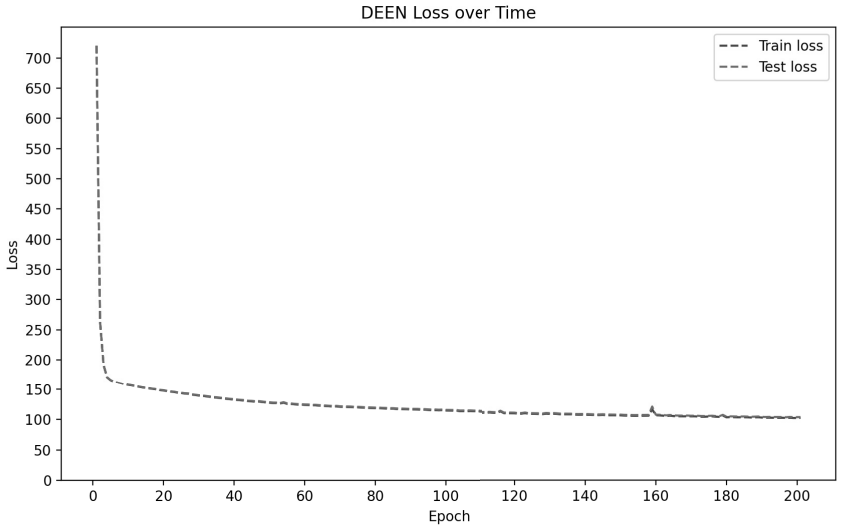


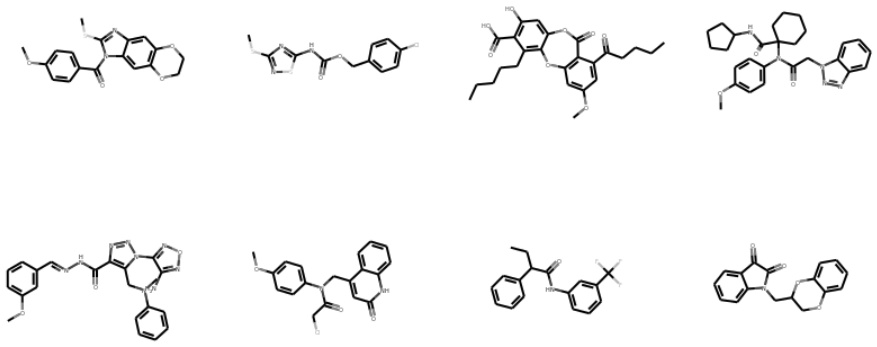
Figure 6: The loss of the DEEN model over training epochs. The loss for training and testing data remains similar throughout the training process.

8 known inhibitors for assay 1,706 are shown in Figure 7, in comparison to 8 inhibitors discovered with and without energy regularization. The figure clearly displays the importance of the energy model in the reward function; those inhibitors discovered with it enabled closely resemble molecules found within the dataset. In contrast, those inhibitors discovered when the energy regularization is disabled ($\beta = 0$) are often unusually simple and linear.

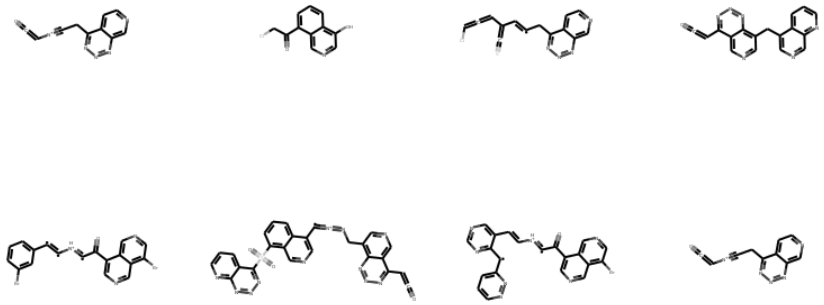
To highlight this point, a plot of the energies of various discovered inhibitors with and without energy regularization is displayed in comparison to known (testing) inhibitors in Figure 8. It can be seen that many of the proposed inhibitors generated without energy regularization are far from the manifold of the known inhibitors. In contrast, those inhibitors discovered with inhibitors are centered within the range of known inhibitors' energy values. In fact, they are much more concentrated than those known inhibitors; suggesting that perhaps the best explored inhibitors lie within a similar subspace of the overall combinatorial space of molecules.

7 Conclusion

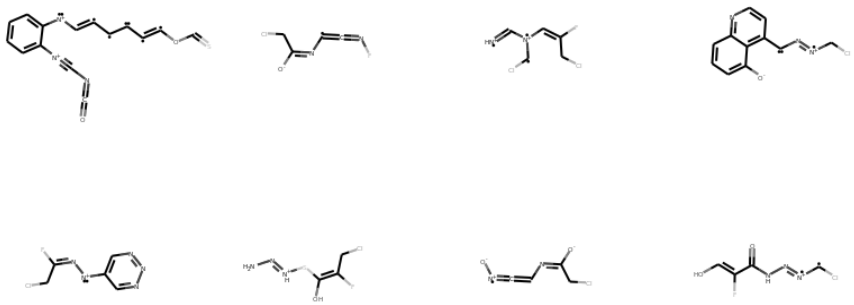
With the goal of designing novel inhibitors for SARS-CoV-1 and SARS-CoV-2, we brought together GNNs for discriminating $3CL^{pro}$ and PL^{pro} inhibitors based on a publicly available database, DEEN for learning the energy function of molecules in the feature space of the trained GNN, and MCTS for searching within the space of inhibitors. The Monte Carlo tree search was guided via a novel reward function whose functional form was dictated by both GNN's learned discriminator and DEEN's learned energy function. The use of energy function here can be informally thought of as a form of soft constraint satisfaction—the "constraint" is to be statistically close to the space of molecules in the dataset—where both β and σ control the "softness". Apart from hyperparameter search which is inherently problem specific, the machinery developed here is indeed quite general and can be applied to any labeled dataset of molecules for drug discovery in particular and in biotechnology at large.



(a) 8 known Inhibitors for assay 1, 706



(b) 8 Inhibitors for assay 1, 706 discovered *with* energy regularization ($\beta \neq 0$)



(c) 8 Inhibitors for assay 1, 706 discovered *without* energy regularization ($\beta = 0$)

Figure 7: Visual comparison of 8 exemplar inhibitors for assay 1, 706 (a) from our dataset, (b) discovered by MCTS with and (c) without energy regularization. The inhibitors discovered with energy regularization visually resemble molecules found within the dataset. In contrast, those discovered without energy regularization are often simple and linear.

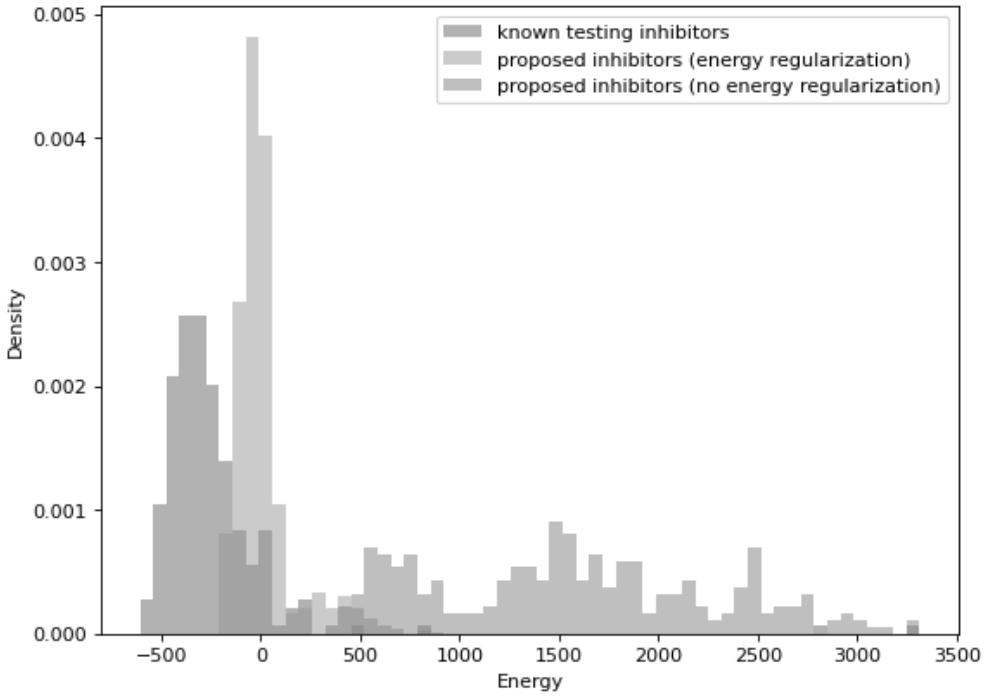


Figure 8: The energies of the molecules discovered with and without energy regularization for assay 1,706, in comparison to known (testing) inhibitors.

References

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] Bo Ram Beck, Bonggun Shin, Yoonjung Choi, Sungsoo Park, and Keunsoo Kang. Predicting commercially available antiviral drugs that may act on the novel coronavirus (sars-cov-2) through a drug-target interaction deep learning model. *Computational and structural biotechnology journal*, 2020.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[6] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[7] Charmaine Butt, Jagpal Gill, David Chun, and Benson A Babu. Deep learning system to screen coronavirus disease 2019 pneumonia. *Applied Intelligence*, page 1, 2020.

[8] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers. *arXiv preprint arXiv:1811.01287*, 2018.

[9] Tim Cofala, Lars Elend, Philip Mirbach, Jonas Prellberg, Thomas Teusch, and Oliver Kramer. Evolutionary multi-objective design of sars-cov-2 protease inhibitor candidates. *arXiv preprint arXiv:2005.02666*, 2020.

[10] Frederik Diehl. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*, 2019.

[11] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292, 2015. URL `http://arxiv.org/abs/1509.09292`.

[12] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*, 2017.

[13] André Fischer, Manuel Sellner, Santhosh Neranjan, Martin Smieško, and Markus A Lill. Potential inhibitors for novel coronavirus protease identified by virtual screening of 606 million compounds. *International Journal of Molecular Sciences*, 21(10):3626, 2020.

[14] Erik Gawehn, Jan A Hiss, and Gisbert Schneider. Deep learning in drug discovery. *Molecular informatics*, 35(1):3–14, 2016.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[16] Garrett B Goh, Nathan O Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. *Journal of computational chemistry*, 38(16):1291–1307, 2017.

[17] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *CoRR*, abs/1709.05584, 2017. URL `http://arxiv.org/abs/1709.05584`.

[18] Markus Hofmarcher, Andreas Mayr, Elisabeth Rumetshofer, Peter Ruch, Philipp Renz, Johannes Schimunek, Philipp Seidl, Andreu Vall, Michael Widrich, Sepp Hochreiter, et al. Large-scale ligand-based virtual screening for sars-cov-2 inhibitors using deep neural networks. *Available at SSRN 3561442*, 2020.

[19] Truong Son Hy, Shubhendu Trivedi, Horace Pan, Brandon M Anderson, and Risi Kondor. Predicting molecular properties with covariant compositional networks. *The Journal of chemical physics*, 148(24): 241745, 2018.

[20] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[21] Joint European Disruptive Initiative (JEDI). Jedi grand challence stage 1: Ultimate list of lead compounds by screening a billion molecules against covid-19. `https://www.covid19.jedi.group/step-1`.

[22] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

[23] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1):D1102–D1109, 2019.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[26] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[29] Egor Kraev. Grammars and reinforcement learning for molecule optimization. *arXiv preprint arXiv:1811.11222*, 2018.

[30] Greg Landrum. Rdkit: Open-source cheminformatics. `http://www.rdkit.org`.

[31] Siddique Latif, Muhammad Usman, Sanaullah Manzoor, Waleed Iqbal, Junaid Qadir, Gareth Tyson, Ignacio Castro, Adeel Razi, Maged N Kamel Boulos, Adrian Weller, et al. Leveraging data science to combat covid-19: A comprehensive review. 2020.

[32] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. *arXiv preprint arXiv:1904.08082*, 2019.

[33] Marina Macchiagodena, Marco Pagliai, and Piero Procacci. Inhibition of the main protease 3cl-pro of the coronavirus disease 19 via structure-based ligand design and molecular modeling. *arXiv preprint arXiv:2002.09937*, 2020.

[34] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018.

[35] Koichi Miyasawa. An empirical Bayes estimator of the mean of a normal population. *Bulletin of the International Statistical Institute*, 38(4):181–188, 1961.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL `http://arxiv.org/abs/1403.6652`.

[37] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[38] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7, 2017.

[39] Herbert Robbins. An empirical Bayes approach to statistics. In *Proc. Third Berkeley Symp.*, volume 1, pages 157–163, 1956.

[40] Saeed Saremi and Aapo Hyvärinen. Neural empirical Bayes. *Journal of Machine Learning Research*, 20:1–23, 2019.

[41] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

[43] Peter C St. John, Caleb Phillips, Travis W Kemper, A Nolan Wilson, Yanfei Guan, Michael F Crowley, Mark R Nimlos, and Ross E Larsen. Message-passing neural networks for high-throughput polymer screening. *The Journal of chemical physics*, 150(23):234111, 2019.

[44] Niclas Ståhl, Goran Falkman, Alexander Karlsson, Gunnar Mathiason, and Jonas Bostrom. Deep reinforcement learning for multiparameter optimization in de novo drug design. *Journal of Chemical Information and Modeling*, 59(7):3166–3176, 2019.

[45] Bowen Tang, Fengming He, Dongpeng Liu, Meijuan Fang, Zhen Wu, and Dong Xu. Ai-aided design of novel targeted covalent inhibitors against sars-cov-2. *bioRxiv*, 2020.

[46] Anh-Tien Ton, Francesco Gentile, Michael Hsing, Fuqiang Ban, and Artem Cherkasov. Rapid identification of potential inhibitors of sars-cov-2 main protease by deep docking of 1.3 billion compounds. *Molecular informatics*, 2020.

[47] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[48] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. URL `http://arxiv.org/abs/1801.07829`.

[49] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[50] Haiping Zhang, Konda Mani Saravanan, Yang Yang, Md Tofazzal Hossain, Junxin Li, Xiaohu Ren, Yi Pan, and Yanjie Wei. Deep learning based drug screening for novel coronavirus 2019-ncov. *Interdisciplinary Sciences, Computational Life Sciences*, page 1, 2020.

[51] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

[52] Alex Zhavoronkov, Bogdan Zagribelnyy, Alexander Zhebrak, Vladimir Aladinskiy, Victor Terentiev, Quentin Vanhaelen, Dmitry S Bezrukov, Daniil Polykovskiy, Rim Shayakhmetov, Andrey Filimonov, et al. Potential non-covalent sars-cov-2 3c-like protease inhibitors designed using generative deep learning approaches and reviewed by human medicinal chemist in virtual reality. 2020.

[53] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.