Bayesian Learning for LambdaZero

1 Problem Formulation

We want to search for molecules, m, over chemical space, $m \in \mathcal{X}$, using LambdaZero. In particular, we want to find a small organic molecule that binds to the COVID-19 protease. This binding ability can be experimentally measured (using a bilogical assay), $y_a : \mathcal{X} \rightarrow \mathbb{R}$, where this process is subject to some noise $y_a(\cdot) = f_a(\cdot) + \epsilon_a(\cdot)$. There is a budget for testing 50 molecules this way at a time. The overall goal is to find $m^* = \arg\max_{m \in \mathcal{X}} f_a(m)$, subject to $c_i(m) \geq C_i$ for $i = 1, \ldots, n$, where the $c_i(\cdot)$ represent constraints such as toxicity, ease of synthesis etc.

Got the timings for this values for this section from various emails/messag – need to check that they are actually correct!

Computational Approximations to Binding Affinity To help us do this we also have access to the following computational functions to use as oracles (all deterministic),$f_\cdot : \mathcal{X} \rightarrow \mathbb{R}$, which predict the binding ability:

Cheap Docking (f_D): This takes 15 secs per molecule. This is computed using Dock6 or AutoDock, which takes as input the ligand's graph (ie the graph of the small organic molecule) and the 3D structire of the protein. It outputs the interaction energy and geometric coordinates of the top scoring poses (see here for further details).

Expensive Docking (f_{ED}): This takes 8 min per molecule. Uses the same software as the cheap docking with different parameters.

Free Energy Perturbation (f_{FEP}): This takes approximate 1 day per molecule on specialized GPU hardware. This could be done using eg the FEP+ software [Wang et al., 2019, Ciordia et al., 2016]. See [Shirts et al., 2010] for an overview of this method.

These different computational functions come with different costs (in terms of runtime) and expected accuracies. Also for some of them we are able to send a batch of molecules to query at once. We summarize the costs of the different techniques in the table below (Table 1) along with the number of evaluations we expect to obrain.

Method	Variance of error distribution	Runtime per molecule	Number of batches that can be queried	Size of batch
Cheap Docking	2.5-2.7	∼15 secs	200E6	1
Expensive Docking	2.5-2.7	∼8 mins	1E6	1
FEP	0.7-1.5	∼1 Day (GPU hardware)	50	256
Lab Experiment	0.3-0.5	1 week+ (in a lab)	5	15

Table 1: Table showing (i) the variance of the error distribution of each method (this is when measuring negative binding energy in kCal/mol which, when measured over all molecules, we expect to be in the range -16 to 0.), (ii) the runtime to compute the operation over one molecule, (iii) the number of batches we can afford to send to each oracle (iv) the size of each batch which can get sent to each oracle.

We also can train surrogate models (eg using a graph neural network, or fully connected neural network on molecular fingerprints) to predict the outputs of these computations in a much faster time (around 5ms). When using we will denote these wrt to the underlying geometric/statistical mechanical computations they are acting as a proxy for, eg $\hat{f}^{\theta}_{ED}$.

Initial Molecular Dataset Before LambdaZero is operational and generating novel molecules, and in order to gather data to train its reward function/extract a vocabulary of molecular building blocks, we also can get access to a a large set of initial obtainable molecules, $\mathcal{M} = \{m_1, m_2, \dots, m_{|\mathcal{M}|}\}$, with $\mathcal{M} \subset \mathcal{X}$. In particular, we are interested in using the ZINC database [Irwin et al., 2012] for this purpose, where $|\mathcal{M}_{\text{ZINC}}| \approx 200$million. The idea would be to run the cheap docking described above on all of these molecules. Note that this would be done in parallel on eg AWS (taking about 7 days) and although the calculation for each molecule is relatively fast, over the whole database this would amount to a considerable computation (15secs $\times$ 200 million $\approx$ 95 years if computed serially!).

1.1 Where Bayesian Learning might help?

On the calls/emails we have discussed several places where Bayesian learning may improve the speed/cost of solving the problem:

1.1.1 Gathering initial data before LambdaZero takes over

Before LambdaZero is trained there is an opportunity to collect data from both the in silico models and in vitro experiments discussed previously. Two different possible aims of this work have been discussed:

Possible Task 1. As a *Bayesian optimization problem* (BO) [Song et al., 2019, Hernández-Lobato et al., 2017] to find the best molecule from $\mathcal{M}_{\text{ZINC}}$. Can think of this as encompassing BO into a virtual screening approach. This would try to aim straight away to find a suitable thereaputic molecule from a set of molecules that are already acquirable and so more quickly physically testable. It would aim to find the best known molecule which then LambdaZero would have to try to improve upon by exploring novel molecules.

Possible Task 2. As an *active learning* [Pinsler et al., 2019, Kirsch et al., 2019] problem where the aim is to learn a surrogate model $\hat{f}_a^\theta$, which works well over all of $\mathcal{X}$ and is used as part of a fast-to-query reward function in LambdaZero. This may differ from the first problem by expending computation on areas of molecular space that are known to have poor bind affinities so that it can still give LambdaZero accurate rewards there.

Both of these tasks share common problems, namely:

- how best to query between different oracles of different costs and fidelities,
- how to choose a batch of points to send to an oracle at once,
- how to optimize an acquisition function over hundreds of millions of discrete candidates,
- how best to design the surrogate models with well-callibrated uncertainties, whilst quick to train and query,

1.1.2 Encouraging diversity and exploration in LambdaZero

We have also discussed whether the BO described above should continue when LambdaZero is able to generate novel molecules. Again this could be done in multiple ways:

Possible Task 3. At each iteration step when choosing to send off a molecule to one of the more expensive oracles BO is used to pick between the molecules LambdaZero has generated and those belonging to $\mathcal{M}_{\text{ZINC}}$.

Possible Task 4. Use BO to define the acquisition function that LambdaZero then needs to optimize when finding molecules. That is instead of optimizing for docking (or a surrogate model thereof) LambdaZero's reward is an acquisition function computed using the principles of BO. This would encourage diverse and novel outputs from LambdaZero, as the acquisition function would be controlling the exploration/exploitation trade-off. However, the reward function for LambdaZero would become more complicated. The state in MCTS (Monte Carlo Tree Search) would also have to encompass a notion of the data seen so far (perhaps agglomerated eg via [Zaheer et al., 2017, Skianis et al., 2019] as well as the current junction tree scaffold of the working molecule, as the reward from the acquisition function would depend on both.

1.2 Summary: Currently focusing on task 1

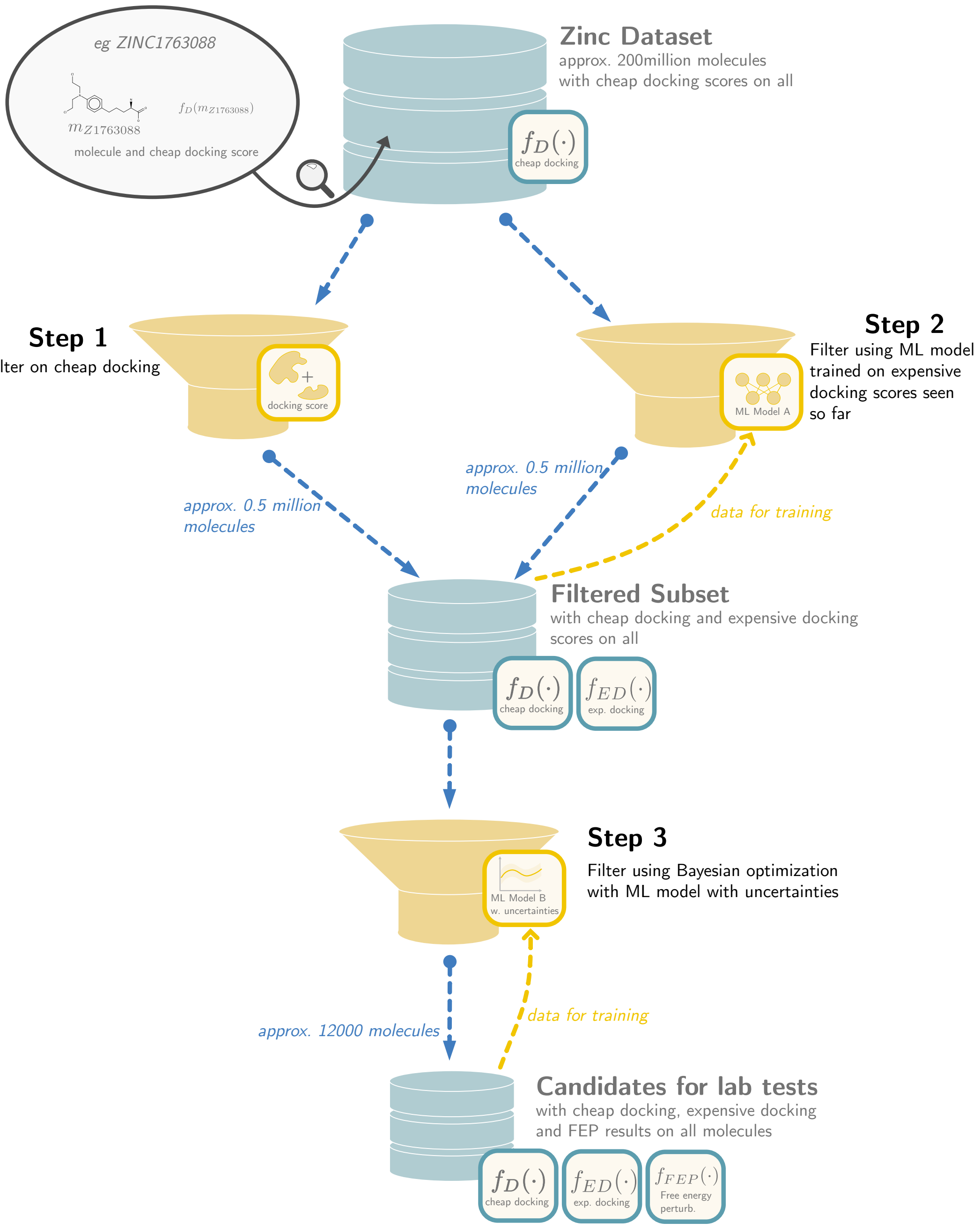
I think at the moment the focus is on task 1, as it would be more immediately useful.

2 Overview of currently proposed approach

In this section we describe the current proposed approach. Summaries of some design choices is given in Section 3.

The overall process is shown in Figure 1. We propose on using two machine learning models: ML Model A and ML Model B. ML Model A is used to try to correct for any mistakes in cheap docking (after seeing some expensive docking results) and so collect promising molecules that might not be selected based on cheap docking alone. ML Model B is a ML model with uncertainties (eg Bayesian regression on MPNN embeddings or a Gaussian process). This acts as a gatekeeper for selecting which molecules get sent for expensive FEP simulations.

These ML models would take in as input the scores from the less expensive oracles (which have already been calculated) as well as a fingerprint representation. Initially for testing this could be a pre-specified fingerprint (eg Morgan Fingerprints) however it is hoped that this will also be learnt eg using a MPNN (Message Passing Neural Networks) or other network which takes the geometry of the molecule into account and is trained to predict cheap/expensive docking and molecule orientations recieved from the previous molecules. A summary of this network is Figure 2.



eg ZINC1763088
$m_{Z1763088}$
$f_D(m_{Z1763088})$
molecule and cheap docking score
Zinc Dataset
approx. 200million molecules with cheap docking scores on all
$f_D(\cdot)$ cheap docking
Step 1
Filter on cheap docking
docking score
Step 2
Filter using ML model trained on expensive docking scores seen so far
ML Model A
approx. 0.5 million molecules
approx. 0.5 million molecules
data for training
Filtered Subset
with cheap docking and expensive docking scores on all
$f_D(\cdot)$ cheap docking
$f_{ED}(\cdot)$ exp. docking
Step 3
Filter using Bayesian optimization with ML model with uncertainties
ML Model B w. uncertainties
approx. 12000 molecules
data for training
Candidates for lab tests
with cheap docking, expensive docking and FEP results on all molecules
$f_D(\cdot)$ cheap docking
$f_{ED}(\cdot)$ exp. docking
$f_{FEP}(\cdot)$ Free energy perturb.

Figure 1: Overall process of how we propose on selecting which molecules get sent to the more expensive oracles.

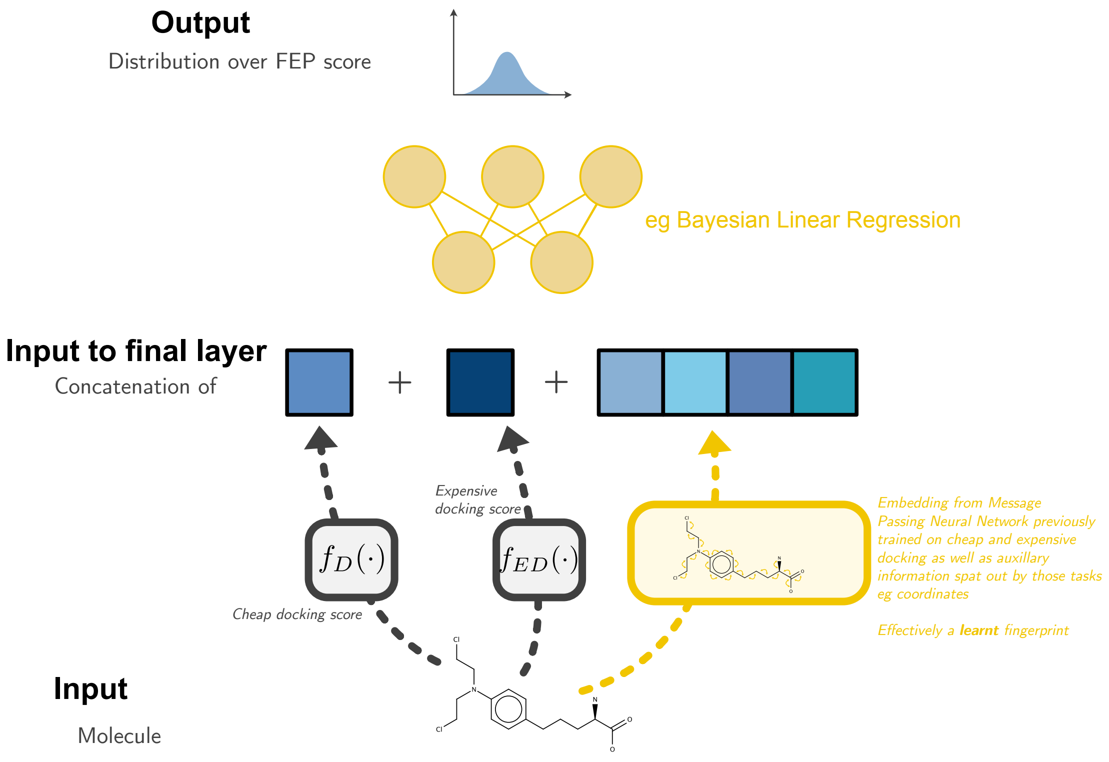

Output
Distribution over FEP score
eg Bayesian Linear Regression
Input to final layer
Concatenation of
+
+
Expensive docking score
$f_D(\cdot)$
$f_{ED}(\cdot)$
Cheap docking score
Embedding from Message Passing Neural Network previously trained on cheap and expensive docking as well as auxillary information spat out by those tasks eg coordinates
Effectively a **learnt** fingerprint
Input
Molecule

Figure 2: The overall architecture of how ML Model B might look

3 Discussions about design choices

3.1 Optimizing an acquisition function over a large candidate set

Running Bayesian optimization over 200 million molecule candidates would take a considerable amount of time/compute (eg 5ms $\times$ 200million $\approx$ 12 days serially!), whereas the expensive docking calculation costs only 8 minutes and FEP 1 day. In this section we discuss various approaches to overcome this problem. These can be grouped into two categories: the first involves filtering down the dataset using the cheaper simulations so that any BO acquisition function is evaluated over the whole of a much smaller subset, whereas the second involves using auxillary ML models to enable the fast search over which molecules to evaluate the acquisition function on. We discuss both in further detail below.

3.1.1 Filtering out the data and running BO only for the most expensive simulations on a reduced dataset

Pros: Simple and likely to work; **Cons:** might miss a promising molecule.

This approach aggressively filters down the dataset size (using the values obtained from the cheap docking), and only evaluates the acquisition function on the reduced set size. The approach is a traditional virtual screening approach with BO only used for the most expensive simulations at the end.

- From 200 million it could be that the cheap docking rules out a large proportion and could be used for filtering without any ML model.
- Then approximatley 1 million could be evaluated using the expensive docking.
- For FEP (a day long operation) we could then start using BO, and evaluate an acquisition function on all 1 million ($5ms \times 1E6 \approx 1hr20mins$) to pick a 12k subset which get sent to the FEP simulation.
- Periodically after improving our surrogate models we could go back and revaluate them on the 200 million full dataset to catch any molecules potentially incorrectly filtered out by the cheap docking.

check these figures!

Practically the whole workflow might look something like this:

Beforehand Evaluate cheap docking, f_D, over the whole 200 million of $\mathcal{M}_{\text{ZINC}}$. There are ?? processors for this and it takes 15 secs per molecule, with the results expected in ??.

Step 1 Using the half million molecules with the best cheap docking score, $\mathcal{M}_{D*,1E6}$, run expensive docking, f_{ED}, on these molecules. There are ?? processors for this and it takes 8 mins per molecule, so the results are expected in roughly ?? days after step 1 is complete.

Step 2a Simultaneously with step 2a train a GNN, $\hat{f}_D^\theta : m \to \mathbb{R}$, to extract vector embeddings from molecules (from eg its penultimate layer) useful for downstream tasks. This network could be trained to predict $f_E D$.

Step 2b Create features for each molecule in $\mathcal{M}_{D*,1E6}$ through the concatenation of the embeddings created in step 2b, the cheap docking score, and the expensive docking score ie $\boldsymbol{x}_i = f_D(m_i) \| f_{ED}(m_i) \| \hat{f}^\theta_{D,\text{penultimate layer}}(m_i)$. Over 1 GPU machine this might take a couple of hours to compute.

Step 2c Using this model also select another half milltion molecules for expensive docking evaluation.

Step 3 Run eg Batch Thompson Sampling [Hernández-Lobato et al., 2017] for working out which $m_i \in \mathcal{M}_{D*,1e6}$ get the FEP evaluated on, f_{FEP}. The BO would use involve training a Bayesian NN or GP to predict f_{FEP}, using $\boldsymbol{x}_i$ in as features.

3.1.2 Cluster the data and only evaluate the molecules in the most promising cluster

Run clustering on the 200 million dataset, eg by using streaming k-means, to break the data down into smaller clusters (these cluster could be further broken down into sub-clusters to form a tree structure). When optimizing the acquisition function first evaluate the function on the mean of each cluster, and only evaluate further the members of the most promising cluster.

3.1.3 Two-level Bayesian optimization

Cons: Untested and so could come across problems when trying this in practice.

Optimizing the acquisition function could be viewed as a Bayesian optimization problem itself. We could solve this search over a fixed 200 million discrete space by using the VAE approach from Gómez-Bombarelli et al. [2018] but restricting the molecules output by the decoder to elements of this discrete set. Then optimization over the 200 million could be done with far less evaluations moving about the continuous latent space of the VAE using an outer loop of BO.

To restrict the decoder to output only valid and already-defined molecules, we could use a CVAE (character VAE). Before training this we would build a trie from the 200 million SMILES strings in $\mathcal{M}_{\text{ZINC}}$, to collect all the valid SMILES string prefixes. This trie datasructure be used inside the decoder to provide masking when picking the next character at each step.

Alternatively, we could train a decoder to predict a fingerprint representation (either eg pre-defined fingerprint such as **?** or one output by a GNN [**?**]) which we then match to its nearest neighbour in $\mathcal{M}_{\text{ZINC}}$, which we are selecting from.

In order to regularize the VAE to have a well-structured latent space we could simultaneously train a regressor from the latent molecule embedding to the value of the cheap docking binding affinity when training the ordinary ELBO.

When optimizing the acquision function for selecting which molecule to send to the FEP oracle we would not evaluate it over all 200 million members of $\mathcal{M}_{\text{ZINC}}$. Instead the acquisition function would be evaluated over a much smaller subset of molecules selected by running a second loop of BO in the VAE's latent space.

3.1.4 Summary: currently focusing on the filtering approach (§3.1.1)

As simpler and faster to implement currently focussing on filtering out using the cheaper oracles and only using BO before FEP.

References

Myriam Ciordia, Laura Pérez-Benito, Francisca Delgado, Andrés A Trabanco, and Gary Tresadern. Application of free energy perturbation for the design of BACE1 inhibitors. *J. Chem. Inf. Model.*, 56(9):1856–1871, September 2016.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276, 2018.

José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1470–1479, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/hernandez-lobato17a.html`.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. ZINC: a free tool to discover chemistry for biology. *J. Chem. Inf. Model.*, 52(7):1757–1768, July 2012.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7024–7035, 2019.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, pages 6356–6367, 2019.

Michael R Shirts, David L Mobley, and Scott P Brown. Free-energy calculations in structure-based drug design. In Kenneth M Merz, Jr, Dagmar Ringe, and Charles H Editors Reynolds, editors, *Drug Design: Structure- and Ligand-Based Approaches*, pages 61–86. Cambridge University Press, 2010.

Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. Rep the set: Neural networks for learning set representations. *arXiv preprint arXiv:1904.01962*, 2019.

Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 3158–3167. PMLR, 16–18 Apr 2019. URL `http://proceedings.mlr.press/v89/song19b.html`.

Lingle Wang, Jennifer Chambers, and Robert Abel. Protein-Ligand binding free energy calculations with FEP. *Methods Mol. Biol.*, 2022:201–232, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6931-deep-sets.pdf`.