Bayesian Active Drug Discovery

Yuanqing Wang * yuanqing.wang@choderalab.org
Computational and Systems Biology Program, Sloan Kettering Institute
Memorial Sloan Kettering Cancer Center
New York, NY 10065

Manh Nguyen mdnguyen@seas.upenn.edu
Computer Science Ph.D. Program
University of Pennsylvania
Philadelphia, PA 19104

Michael Retchin michael.retchin@med.cornell.edu
Tri-Institutional Ph.D. Program in Computational Biology and Medicine
Weill Cornell Medical College, Cornell University
New York, NY 10065

John D. Chodera † john.chodera@choderalab.org
Computational and Systems Biology Program, Sloan Kettering Institute
Memorial Sloan Kettering Cancer Center
New York, NY 10065

Theofanis Karaletsos‡ theofanis.karaletsos@googlemail.com
Corona Heights Institute for Computational Health Informatics
San Francisco, CA 94114

Abstract

We combine graph neural networks with Gaussian Process regression through *deep graph kernel learning* and demonstrate its robustness on quantitative structure-activity relationship (QSAR) modeling tasks. Equipped with such a model, a Bayesian optimization experiment on chemical space is conducted and compared against the time-stamped acquisition records of a real-world, time-sensitive molecular optimization mission: the identification of potent inhibitors of the main protease of SARS-CoV-2, the viral pathogen responsible for the COVID pandemic.

Keywords: Graph Nets, Gaussian Processes, Deep Kernel Learning, Generative Models, Drug Discovery, Experimental Design

∗. Corresponding author. Alternative Address: Physiology, Biophysics, and System Biology Ph.D. Program, Weill Cornell Medical College, Cornell University, New York, NY 10065; M.F.A. Program in Creative Writing, Division of Humanities and Arts, the City College of the City University of New York, New York, NY 10031.
†. Corresponding author.
‡. Corresponding author.

1. Introduction

Drug discovery, with some oversimplication, can be viewed as the process of finding a compound, denoted by chemical graph g^* on some subset of chemical space, $\mathcal{G}$, that maximizes its activity a^* (potency) toward some biological target, while satisfying a range of additional physical, chemical, and biological criteria, jointly denoted as the measurement space $\mathcal{A}$. The pre-clinical stage of this optimization process is time- and money-consuming, requiring 12 years and $800 million on average (Paul et al., 2010). These figures can be attributed to the discrete, combinatorially large nature of chemical space, as well as the prohibitively high costs associated with each acquisition (compound synthesis or purchase) and function evaluation (assay). Graph neural networks (Battaglia et al., 2018; Wang et al., 2019; Xu et al., 2018; Kipf and Welling, 2016), or simply *graph nets*, are popular in the computer-aided drug discovery (CADD) community for quantitative structure-activity relationship (QSAR) modeling. (Wu et al., 2018; Ramsundar et al., 2017).

In this paper, to build uncertainty-calibrated models for QSAR, we use deep kernel learning (Wilson et al., 2015) to model the mappings between compound structure and compound properties, combining the rich representation of graph nets and the sample efficiency of Gaussian processes. We use these learnable functions to drive an experimental iteration, emulating the prototypical drug design iteration with Bayesian Optimization (shorthand: **BO**). We first introduce the technical tools and the task abstraction in Section 2. Next, to demonstrate the utility of our machine-aided drug discovery loop to real-world drug discovery, for which efficient and rapid use of limited resources is imperative (Warmuth et al., 2002; Reker and Schneider, 2015; Jensen et al., 2019; Coley et al., 2019; Zhang and Lee, 2019), we apply our optimization procedure on a carefully conducted pre-existing experimental system using data from the COVID Moonshot (Chodera et al., 2020), which is targeted at the development of potent, selective inhibitors of the main protease of SARS-CoV-2 (the virus responsible for COVID-19 disease). In this system, we can benchmark the efficacy of our model and compare its search efficiency to policies made by a human medical expert by re-playing experiments with counterfactual algorithmically inferred experimental orderings within the pool of assayed data.

2. Methods

Supplementary Section 10 contains complete methodological details of our models, inference algorithms, and Bayesian Optimization strategies.

2.1 Sequential Experimental Design For Drug Discovery

We can abstract the experimental loop involved in drug discovery introduced in Section 1 as such: given a chemical space $\mathcal{G}$ of potential compounds, and assays $f_{\mathrm{a}} : \mathcal{G} \to \mathcal{A}$ which map from chemical to measurement space, search for compounds with values of f_a that satisfy certain criteria. Initially, we don't have measurements of any compound and start with an empty measured dataset $\mathcal{D}^0 = \{\}$. An experiment consists of selecting compounds from $\mathcal{G}$ and sampling f_{a} to yield tuples $\mathbf{d} = \{g_i, a_i\}_{i=1}^N$ of observed data. Experiments are typically conducted in iterations, during which an experimenter picks batched indices $\mathbf{i}_g$

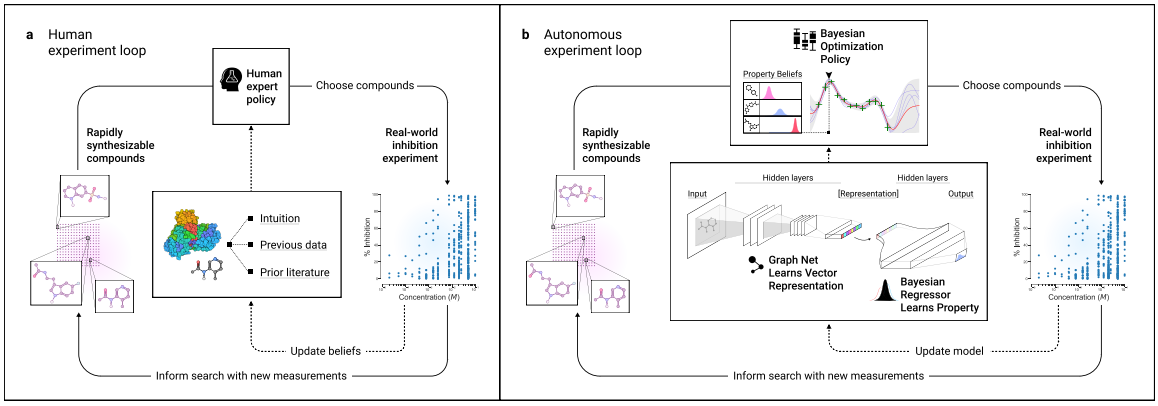

Figure 1: **Schematic illustration of manual and autonomous experimentation loops for compound potency optimization. Left**: Human experts select compounds to synthesize (e.g. from Enamine REAL Denis) and assay based on *chemical intuition*, which is subsequently updated by assay results. **Right**: Bayesian optimization selects new candidate batches for experimentation according to an uncertainty-aware acquisition function that consumes the results of previous inhibition experiments and the beliefs of a QSAR model, using a latent representation given by graph nets. Acquisition iterations are carried out until drug candidates satisfying specified criteria are identified.

of compounds to measure according to a policy $\mathbf{i}_g \sim \pi(\mathbf{i}|\mathcal{M})$—shorthand $\pi_\mathcal{M}$—where $\mathcal{M}$ determines an underlying model that determines the policy.

At each iteration l, datasets $\mathbf{d}^l$ consisting of the batch of chosen compounds and acquired measurements are selected and added to the overall dataset, $\mathcal{D}^l = \mathcal{D}^{l-1} \cup \mathbf{d}^l$. The experimental *policy* $\pi_\mathcal{M}$ is a crucial factor determining the speed, quality, and efficiency of the search for active compounds. Typically, a policy is driven by human experts (denoted as π_H) based on their so-called *chemical intuition*, which is distilled from training and experience. We propose exploring the use of automated policies based on Bayesian Optimization π_BO to drive the drug discovery iteration, which involves learning a model capturing beliefs about chemical properties from data during the rounds as described in the following.

2.2 Probabilistic Models of Compounds with Graph Kernel Learning

In order to learn generalizeable models of compounds, we propose combining graph neural networks with Gaussian Process regression. Under the framework of Deep Kernel Learning (Wilson et al., 2015), the latent latent representation of an input graph as given by a graph net is used as features for a fixed-dimensional kernel, which in turn is used to construct mappings from graph structures to measurement values with Gaussian Process regression (Rasmussen, 2003).

We call the resulting model class Deep Graph Kernel Learning (shorthand: **DGKL**). Given a dataset $\mathcal{D} = \{g_i, a_i\}_{i=1}^N$ of N input molecules and output measurements, and with $\Theta = \{\theta_k, \theta_{nn}, \theta_o\}$ denoting the parameters of the kernel, the graph net, and the observation model, the joint model is given by:

$$p(\mathbf{a}, f|\mathbf{G}, \Theta) = p(f|\mathbf{G}, \theta_k, \theta_{nn}) \prod_{i=0}^{N} p(a_i|f, g_i, \theta_o),$$

where $\mathbf{H} = \mathrm{NN}_{\theta_{nn}}(\mathbf{G})$, $p(f|\mathbf{G}, \theta_k, \theta_{nn}) = \mathcal{GP}(f|m(\mathbf{H}), \kappa^{\theta_k}(\mathbf{H}))$ with a suitable kernel function $\kappa(\cdot)$, and $p(a_i|f, g_i, \theta_o) = \mathcal{N}(a_i|f(g_i), \sigma_o^2)$.

3. Experiments

In this section, we design and conduct experiments that mimic the scenarios in real-world drug discovery projects.

Dataset For Chemical Space Exploration We employ data from the COVID Moonshot (Chodera et al., 2020; Lee and Brian, 2020), an active open-science drug discovery project aiming to develop small molecule inhibitors of the main viral protease (Mpro) from SARS-CoV-2. The compounds are submitted by teams around the globe, with fractions of the chemical space being prioritized—by their in-house expert medicinal chemists—to be synthesized and assayed. For each compound, single-point titrations (at 20 and 50 μM) are conducted, where the results could be unavailable due to solubility constraints among others. Multi-point titrations are conducted (at at least eight concentrations) for promising candidates. The decision to prioritize compounds are made weekly (on Wednesdays), allowing us to use the time stamps annotated to divide compounds into roughly similar-sized groups ($\sim$100/group, split at every Tuesday since May 12), to characterize human acquisition policies. We also briefly benchmark the supervised learning performance of our model on solubility data from ESOL (Delaney, 2004).

Overview of Experiments With 3, **first**, in order to discover the quantitative structure-activity relationship (QSAR) potency or physical properties, $f_{\mathrm{a}} : \mathcal{G} \to \mathcal{A}$, the models must predict held-out data with high likelihood (see Appendix Section 8.1). **Second**, in Section 3.1, we apply our algorithms to a real-world dataset from an active open-science COVID-19 drug discovery project (assay details are given in Section 10.5). Here, we replay the experiment of the Covid Moonshot project with our model and simulate its trajectories with automated policies, allowing us to compare the human and automated policies. **Finally**, in Appendix Section 9, we evaluate semi-supervised learning on representational efficiency in the presence of more unmeasured molecules, in the hope of yielding gains in the data scarcity regime.

3.1 Bayesian Optimization on Compound Space

In this section, we employ **DGKL** in a Bayesian optimization loop to search for most active compounds from the COVID Moonshot project. Starting randomly from the candidate pool, in each round, a supervised learning model is trained on the acquired data points and used to predict inhibition activities for the remaining candidates. The candidate molecules are then picked according to an acquisition rule (see Section 10.4).

We assess the efficiency of acquisition by plotting the regret (difference between the current best candidate and the ground truth best candidate) against the numbers of candidates acquired in Figure 10.4.1. **DGKL** stands out from acquisition rules using neural network regressors trained with MLE as well as BBB. We briefly study the impact of batch size on acquisition efficiency in Section 6.

Next, we compare the acquisition policy between Bayesian models π_{BO} and human experts π_{H}(See 1), which is (as one of the rare times) made possible by the time-stamped

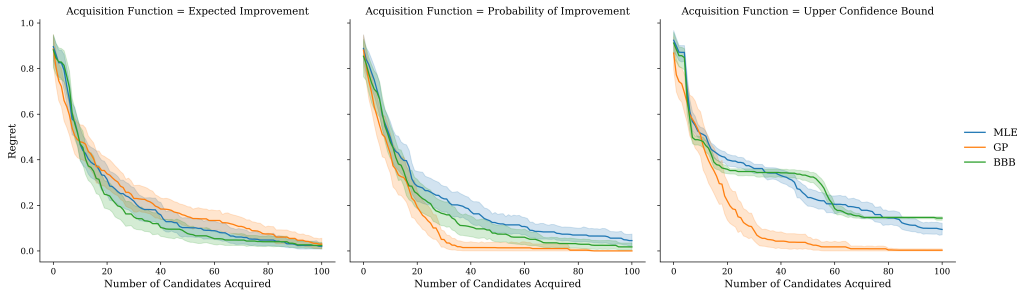

Figure 2: **DGKL speeds up active compound acquisition.** Regret of acquisition plotted against number of candidates acquired with various acquisition functions 10.4 and regressors.

COVID moonshot open project data. Concretely, we replay the acquisition decisions of human experts and train **DGKL** models on data as of each Tuesday from May 12. On the other hand, **BO** is allowed to choose among the candidates and train **DGKL** models using a similar batch size 96 (to match the common use of 96-well SBS-format microplates in modern bioassays) and subsequently train **DGKL** models with same hyperparameter choices 8. In Figure 3, we plot maximum of *prospective* probability of improvement and the upper confidence boundary (95%) (calculated based on the predictive distribution) after each round of acquisition for both BO and human expert acquisition. We believe this is a useful indication for the good characterization of the candidate space: Ideally, both metrics should decrease and approach zero as more candidates are being acquired, reflecting diminishing *optimism* that better candidates are yet to be sampled. As such, our model is *overoptimistic* to a lesser degree when compared with human experts.

3.2 Latent representation sharing with scarce, heterogeneous data.

		Inh. @ 20 μM (R)		Inh. @ 50 μM (R)		IC50 (R)		Inh. @ 20 μM(F)		Inh. @ 50 μM(F)		IC50 (F)	
		NLL	R^2	NLL	R^2	NLL	R^2	NLL	R^2	NLL	R^2	NLL	R^2
GP	Independent	0.43	0.33	0.57	-0.04	0.78	-0.92	0.52	0.15	0.51	0.07	0.54	0.15
	Joint	-0.25	0.65	0.25	0.25	0.18	-0.34	0.00	0.44	0.15	0.42	0.05	0.19
BBB	Independent	0.18	-1.01	3.72	-803.79	0.37	-2.74	37.72	-46.31	0.46	-0.30	2.61	-159.65
	Joint	8.27	-77.02	26.61	-14.26	7.95	-926.31	94.22	-583.42	4.05	-70.62	7950.01	-196.31
MLE	Independent	3.48	-901.72	24999.34	-5.64	2.08	-30.52	2261755392.00	-23.28	3.80	-80.87	427.18	-467.90
	Joint	1.13	-9.05	724.38	-33.60	1281.74	-117.22	4.94	-62.45	3.60	-0.15	0.44	-0.64

Table 1: **Graph latent encoding sharing boosts performance**. *Independent* denotes separate models for each assay. *Joint* denotes a model with a shared graph net as latent embedding with multiple output regressors for each assay.

The situation we face in modeling the QSAR from data generated by the COVID Moonshot project is typical in early-stage drug discovery (see also Section 10.5): The applicability of various assays differs from molecule to molecule, resulting in a highly heterogeneous dataset (i.e. blank spaces in tabulated data). Naïvely, one would construct one QSAR model for each assay. In Tab. 1, we demonstrate that such formulation yielded suboptimal results. To circumvent this, we purpose to share the latent code representation between the QSAR models for each assay. Specifically, a single graph net is used to project the

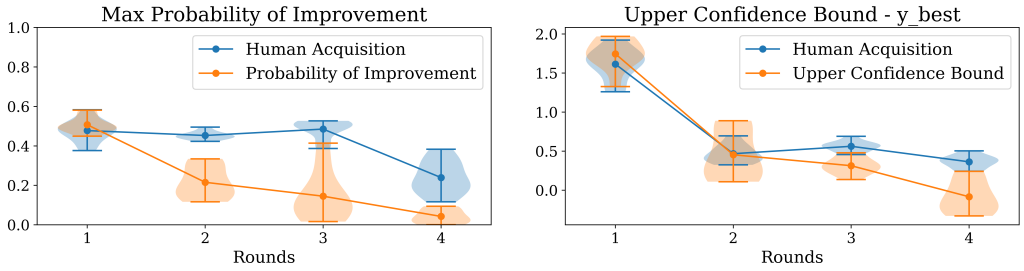

Figure 3: **Prospective Analysis Of Experiment Using Bayesian optimization acquisition reveals automated acquisitions are less prone to be overoptimistic.** (**Left**)Max (among the candidates) probability of improvement ,plotted against rounds of acquisition, indicates probabilistically what the model's belief in the value of continued search is. The human policy leads to search strategies that resolve uncertainty about the true maximumj value slower. (**Right**) the difference between the max upper confidence interval and the current best candidate, plotted against rounds of acquisition, reveals in measurement space by how much the model believes further search could improve on the currently found values.Again, automated policies lead to less overoptimistic search. At round four, all the data is acquired and assayed, thus an ideal model would predict both metrics to be zero. Hence values above zero indicates *overoptimism*, which is more prevailing with human acquisitions. The error bars are obtained by repeating experiments ten times.

molecular graphs to a hidden space, from where multiple Gaussian process regressors are used as the final layer of regression to come up with predictive distributions. We illustrate here that this formulation is effective as it brings drastic improvement to the performance of the regression models.

4. Discussion

In this paper, we tackle experimental design for automated discovery by combining Gaussian process regression with graph nets and Bayesian optimization. Crucially, we establish that our methodology leads to faster exploration of active compounds in a problem modeled on data from a real-world drug discovery campaign. This suggests that our approach is promising in its potential to drive time-critical experimental procedures to find novel therapeutics. We hope that this work will help bring together research on Bayesian active drug discovery, which, in most cases, is in the regime of so-called *small data*, and where systemic optimization schemes could facilitate rational decision making processes. With the rapid development of robotics system and automated chemical synthesis and biological assay, research on this front will bring us one step closer to fully autonomous drug discovery.

References

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.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

John Chodera, Alpha A. Lee, Nir London, and Frank von Delft. Crowdsourcing drug discovery for pandemics. *Nature Chemistry*, 2020. doi: 10.1038/s41557-020-0496-2. URL `https://doi.org/10.1038/s41557-020-0496-2`.

Connor W Coley, Natalie S Eyke, and Klavs F Jensen. Autonomous discovery in the chemical sciences part ii: Outlook. *Angewandte Chemie International Edition*, 2019.

John S. Delaney. Esol: estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, 2004. doi: 10.1021/ci034243x. URL `https://doi.org/10.1021/ci034243x`. PMID: 15154768.

Denis. Real database. URL `https://enamine.net/library-synthesis/real-compounds/real-database`.

Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Torsten Hoffmann and Marcus Gastreich. The next level in chemical space navigation: going far beyond enumerable compound libraries. *Drug discovery today*, 24(5):1148–1156, 2019.

Klavs F Jensen, Connor W Coley, and Natalie S Eyke. Autonomous discovery in the chemical sciences part i: Progress. *Angewandte Chemie International Edition*, 2019.

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

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

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

Alpha Lee and Shankaruk Brian, 2020. URL `https://www.gofundme.com/f/covidmoonshot`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan

Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Steven M. Paul, Daniel S. Mytelka, Christopher T. Dunwiddie, Charles C. Persinger, Bernard H. Munos, Stacy R. Lindborg, and Aaron L. Schacht. How to improve r&d productivity: the pharmaceutical industry's grand challenge. *Nature Reviews Drug Discovery*, 9(3):203–214, 2010. ISSN 1474-1784. doi: 10.1038/nrd3078. URL `https://doi.org/10.1038/nrd3078`.

Bharath Ramsundar, Bowen Liu, Zhenqin Wu, Andreas Verras, Matthew Tudor, Robert P. Sheridan, and Vijay Pande. Is multitask deep learning practical for pharma? *Journal of Chemical Information and Modeling*, 57(8):2068–2076, 2017. doi: 10.1021/acs.jcim.7b00146. URL `https://doi.org/10.1021/acs.jcim.7b00146`. PMID: 28692267.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Daniel Reker and Gisbert Schneider. Active-learning strategies in computer-assisted drug discovery. *Drug discovery today*, 20(4):458–465, 2015.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs, 2019.

Yuanqing Wang, Josh Fass, Chaya D. Stern, Kun Luo, and John Chodera. Graph Nets for Partial Charge Prediction. *arXiv e-prints*, art. arXiv:1909.07903, Sep 2019.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019.

Manfred KK Warmuth, Gunnar Rätsch, Michael Mathieson, Jun Liao, and Christian Lemmen. Active learning in the drug discovery process. In *Advances in Neural information processing systems*, pages 1449–1456, 2002.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2015.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2018.

Yao Zhang and Alpha A. Lee. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chem. Sci.*, 10:8154–8163, 2019. doi: 10.1039/C9SC00616H. URL `http://dx.doi.org/10.1039/C9SC00616H`.

Yao Zhang et al. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical Science*, 10(35):8154–8163, 2019.

5. Acknowledgments

The authors thank Josh Fass (MSKCC) for insightful discussions about the methodologies described in this manuscript. The authors are enormously grateful to the entire COVID Moonshot team [`https://discuss.postera.ai/t/covid-project-faq-about-us/72`]—especially to Nir London and Haim Barr (Weizmann Institute) for providing open assay data and Matt Robinson (PostEra) for supplying the excellent information infrastructure for organizing the data and making it easily computable. As part of Figure 1, we used illustrations from Noun Project, created by Saeful Muslim, Kirby Wu, Carla Porciuncula, and Robbe de Clerck under a Creative Commons License.

6. Funding

YW and MN acknowledge support from NSF CHI-1904822 and the Sloan Kettering Institute. Michael Retchin acknowledges support from the Tri-Institutional PhD Program in Computational Biology and Medicine. JDC acknowledges support from NIH grant P30 CA008748, NIH grant R01 GM121505, NIH grant R01 GM132386, NSF CHI-1904822, and the Sloan Kettering Institute.

7. Disclosures

YW is among the co-founders and equity holders of Uli, Inc. and Uli (Shenzhen) Techonology Co. Ltd.

JDC is a current member of the Scientific Advisory Board of OpenEye Scientific Software. The Chodera laboratory receives or has received funding from multiple sources, including the National Institutes of Health, the National Science Foundation, the Parker Institute for Cancer Immunotherapy, Relay Therapeutics, Entasis Therapeutics, Silicon Therapeutics, EMD Serono (Merck KGaA), AstraZeneca, Vir Biotechnology, XtalPi, the

Molecular Sciences Software Institute, the Starr Cancer Consortium, the Open Force Field Consortium, Cycle for Survival, a Louis V. Gerstner Young Investigator Award, and the Sloan Kettering Institute. A complete funding history for the Chodera lab can be found at `http://choderalab.org/funding`.

8. Disclaimers

The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Appendix A: Implementation details. Unless otherwise mentioned, all the models contain three graph convolutional net (Kipf and Welling, 2016) with 32 units each in PyTorch Paszke et al. (2019) and DKL (Wang et al., 2019) and optimized with Adam optimizer (Kingma and Ba, 2014) with 1e-3 learning rate. Note that since we do not perform hyperparameter tuning on models, we do not reserve data set validation set.

8.1 Appendix B: Supervised learning with Deep Graph Kernel Learning

We test the supervised learning performance of **DGKL** on the aforementioned benchmark datasets (Wu et al., 2018): ESOL (Delaney, 2004). The results are summarized in Figure 4 and Table 2.

		NLL		R^2		RMSE	
		Train	Test	Train	Test	Train	Test
GP	GCN	$-0.4448^{-0.4318}_{-0.4923}$	$0.6983^{1.0894}_{0.3247}$	$0.9969^{0.9973}_{0.9959}$	$0.9365^{0.9966}_{0.9468}$	$0.1142^{0.1193}_{0.0975}$	$\mathbf{0.5568}^{0.5296}_{0.3020}$
	GraphSAGE	$-0.4940^{-0.4975}_{-0.5399}$	$0.7562^{0.9688}_{0.5417}$	$\mathbf{0.9974}^{0.9981}_{0.9969}$	$0.9232^{0.9778}_{0.9246}$	$0.1047^{0.1448}_{0.1068}$	$0.6123^{0.5947}_{0.3334}$
	SGC	$-0.4395^{-0.4583}_{-0.5263}$	$0.7584^{0.8265}_{0.4152}$	$0.9969^{0.9985}_{0.9967}$	$0.9285^{0.9487}_{0.9094}$	$0.1156^{0.1134}_{0.0949}$	$0.5907^{0.6355}_{0.4904}$
	EdgeConv	$\mathbf{-0.6958}^{-0.6696}_{-0.6921}$	$1.0254^{1.3738}_{1.0250}$	$0.9995^{0.9996}_{0.9994}$	$0.8976^{0.9445}_{0.8079}$	$\mathbf{0.0475}^{0.0547}_{0.0442}$	$0.7069^{0.8757}_{0.6110}$
	GIN	$-0.4336^{-0.4197}_{-0.4549}$	$0.7349^{1.0141}_{0.5960}$	$0.9969^{0.9980}_{0.9966}$	$0.9297^{0.9733}_{0.7273}$	$0.1153^{0.1261}_{0.0922}$	$0.5860^{0.8057}_{0.5843}$
	TAGCN	$-0.5340^{-0.5298}_{-0.5671}$	$\mathbf{0.6688}^{0.5906}_{0.1636}$	$0.9982^{0.9984}_{0.9973}$	$\mathbf{0.9408}^{0.9590}_{0.8499}$	$0.0864^{0.0937}_{0.0829}$	$0.5378^{0.7344}_{0.4912}$
BBB	GCN	$5.7001^{8.4629}_{4.7325}$	$6.3522^{8.6456}_{6.4049}$	$0.7557^{0.8305}_{0.7458}$	$0.8564^{0.8989}_{0.6491}$	$1.0188^{1.0113}_{0.8272}$	$0.8373^{0.8983}_{0.6889}$
	GraphSAGE	$22.3306^{20.5570}_{17.0102}$	$26.7551^{26.6674}_{13.1267}$	$0.8825^{0.9188}_{0.8615}$	$0.8987^{0.9686}_{0.8890}$	$0.7067^{0.6834}_{0.4645}$	$0.7032^{0.8502}_{0.6266}$
	SGC	$4.9587^{5.0732}_{3.1761}$	$5.4311^{5.9102}_{2.7561}$	$0.6862^{0.7859}_{0.7104}$	$0.8235^{0.8644}_{0.7551}$	$1.1546^{1.1415}_{1.0112}$	$0.9284^{0.9364}_{0.6280}$
	TAGCN	$37.6259^{37.3091}_{22.9957}$	$44.5826^{46.0556}_{12.9843}$	$0.8871^{0.9425}_{0.9025}$	$0.9100^{0.9256}_{0.8361}$	$0.6925^{0.8056}_{0.5790}$	$0.6629^{0.5956}_{0.4620}$
MLE	GCN	$6.5405^{7.0567}_{4.8459}$	$6.8694^{7.0497}_{3.4889}$	$0.7705^{0.8008}_{0.6824}$	$0.8586^{0.9320}_{0.8243}$	$0.9874^{1.3219}_{0.9892}$	$0.8308^{0.7990}_{0.5190}$
	GraphSAGE	$19.5203^{23.9747}_{14.8009}$	$22.3864^{25.8108}_{11.7202}$	$0.8780^{0.9031}_{0.8753}$	$0.9179^{0.9263}_{0.8682}$	$0.7200^{0.9203}_{0.6578}$	$0.6330^{0.9210}_{0.6598}$
	SGC	$23.4198^{32.6475}_{22.8581}$	$24.0242^{31.8693}_{18.9299}$	$0.8810^{0.8826}_{0.8029}$	$0.9124^{0.9011}_{0.8109}$	$0.7112^{0.7387}_{0.6290}$	$0.6539^{1.3147}_{0.7457}$
	EdgeConv	$109.4343^{170.7550}_{111.6888}$	$120.7490^{103.5109}_{54.3249}$	$0.9400^{0.9530}_{0.9091}$	$0.9461^{0.9541}_{0.8050}$	$0.5051^{0.4208}_{0.3580}$	$0.5128^{0.6677}_{0.4346}$
	GIN	$17.7285^{21.3566}_{14.9618}$	$24.2063^{57.5309}_{31.8718}$	$0.8329^{0.8318}_{0.7468}$	$0.9031^{0.9404}_{0.8927}$	$0.8427^{0.9797}_{0.8135}$	$0.6876^{0.7370}_{0.5847}$
	TAGCN	$6.3408^{6.8595}_{5.7497}$	$7.2210^{10.4134}_{5.5034}$	$0.7613^{0.8079}_{0.7469}$	$0.8381^{0.8698}_{0.7885}$	$1.0070^{1.0438}_{0.8930}$	$0.8890^{1.0456}_{0.7742}$

Table 2: **DGKL is Robust in QSAR Modeling with Various Graph Net Architectures.** Train and test set performance measured by negative log likelihood, R^2, and RMSE. The super- and subscript indicate 95% confidence interval measured by bootstraping over dataset. Best performance is boldfaced. For graph net architectures, see 3.

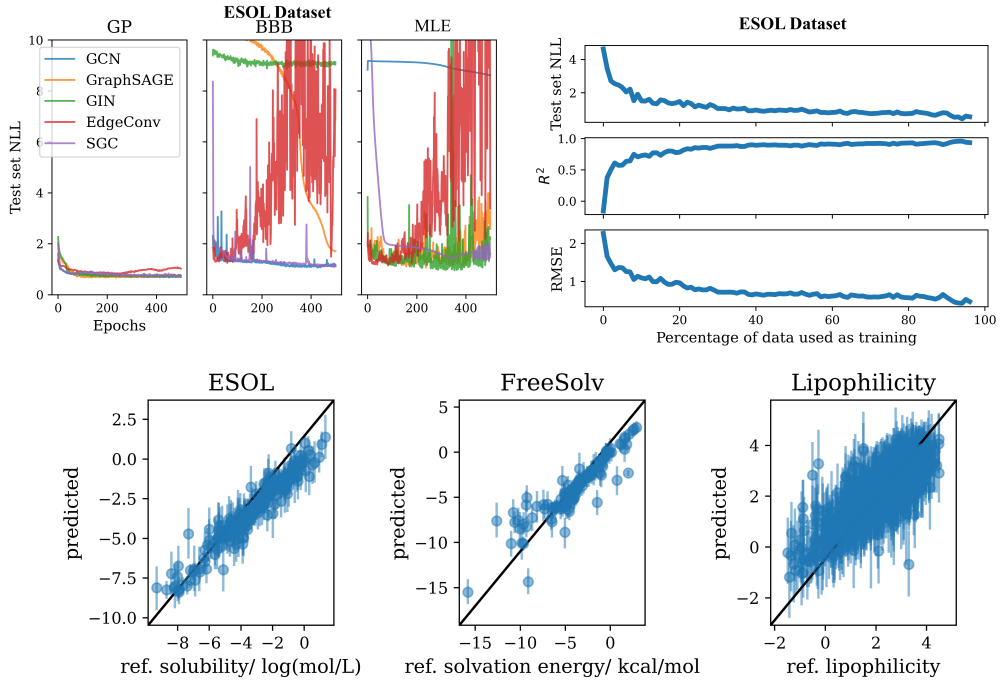

Figure 4: Upper Left: **Test set negative log likelihood from various graph representation models and inference algorithms.** Upper Right: **Test set performance of Deep graph kernel learning with GCN (Kipf and Welling, 2016) plotted against percent of data used as training set.** Lower: **Scatter plot of reference vs predicted value for three physical property datasets with error bars.**

9. Appendix C: Extra Results

9.0.1 Appendix D: Semi-supervised learning

Recall that one difficulty of applying machine learning to drug discovery is the shortage of labeled data. This makes training supervised model on small labeled data sets particularly challenging. Over-fitting is a very common problem when one has only a handful of labeled data. At the same time, we have a huge quantity of unlabeled data in the form of molecules without chemical measurements. We can leverage this abundance of unlabeled data to improve the performance of our models in supervised and active learning tasks. In this section, we introduce the working of semi-supervised learning.

In semi-supervised learning, the model is exposed to both labelled and unlabelled data during its training process. Our semi-supervised model combines Variational Auto-Encoder (VAE) (Kingma and Welling, 2013) and Deep Gaussian Processes (Wilson et al., 2015). The objective function is a combination of the ELBO of the VAE and the negative log-likelihood of the Deep GP. This provides a smooth and principled way to incorporate both generative model training and supervised training. This allows our model to smoothly handle both unlabeled and labeled data. The semi-supervised model could be understood as "self-regularized", reducing the over-fitting problem. In addition, jointly optimizing

both the supervised loss and the unsupervised loss forces the model to learn molecular representations that are more robust to noise or data set variance.

In this experiment, our labeled data set is the COVID Moonshot. Since the COVID Moonshot data set is a multi-label data set, we picked *% inhibition at 20 μM* as the target for our supervised task. In later sections, we will also demonstrate the applicability of semi-supervised model to multi-task learning.

We first split COVID Moonshot data set into 80% for training and 20% for testing. To prepare unlabeled data for semi-supervised training, we use *FTrees* [1], short for Feature Trees. It is a highly efficient software tool for pharmacophore-style similarity searching. It is used to facilitate virtual high through-put drug screening. On a SMILE-type input, FTrees creates "chemically similar" molecules to the input molecule. Using this tool, we could create a large quantity of unlabeled data that is "similar" in chemical space to our labeled data of interest. Through our experiments, we observed that having a "chemically similar" unlabeled data set is very important to semi-supervised learning.

For each molecule in COVID Moonshot, we synthesize 100 molecules. Let us collectively call all the synthetic molecules as our "bag of molecules". We then construct mixtures of labeled and unlabeled data by combining the training portion of COVID Moonshot with varying numbers of synthetic molecules (1000, 3000 and 5000), randomly selected from our "bag of molecules".

Our base line is a supervised-only model based on Graph Neural Networks. We make sure that the architecture and training methods are the same for different semi-supervised model for a fair comparison. We also make sure that the encoder of the Variational Auto-Encoder used in our semi-supervised model has the same architecture as the Graph Neural Network used by the supervised model.

Figure 5 shows the performance on held-out test set against training epochs for semi-supervised model, trained on different quantity of unlabeled synthetic data. We see that in terms of R^2 and RMSE, the semi-supervised model is competitive with the supervised-only model. Despite slower progress in the initial training epochs, the semi-supervised model quickly catches up to the supervised-only model. In this experiment, we were not able to see any significant effect of the quantity of unlabeled data and the performance of the semi-supervised model. We surmise that we need to use networks with more parameters for the semi-supervised model to effectively learn from the much bigger data set. We believe that once we have found a better design, the advantage of having more (unlabeled) data will be significant.

We also have evidence to believe that semi-supervised models should have more parameters than supervised-only models. In our experiments, we observed that a Graph Convolution Networks with two convolution layers of size 128 and 128 will lead to numerical issues or overfitting when optimized with supervised-only training, especially on smaller data sets. However, the same architecture could be optimized with relative stability using semi-supervised training. A deeper network can learn more complicated patterns in the data than a simple one. However, we ran out of time before we could design and run these more elaborate experiments. In the future, we will explore more advanced architecture and we believe that semi-supervised learning will be an indispensable tool in our arsenal.

1. https://www.biosolveit.de/FTrees/

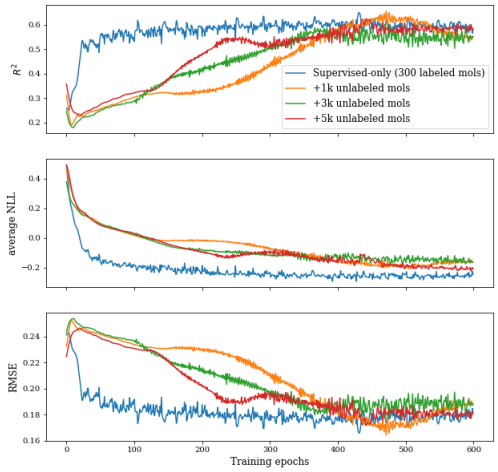

Figure 5: **Supervised learning provided no improvement on QSAR fitting.**

10. Appendix E: Methods

10.1 Graph Neural Networks

In the context of molecular machine learning, molecules are modelled as undirected graphs of bonded atoms, where each atom and bond can carry attributes reflecting their chemical nature from which complex chemical features can be learned. If we write this as a tuple of three sets,

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{U}\} \tag{1}$$

Here, $\mathcal{V}$ is the set of the vertices (or nodes) (atoms), $\mathcal{E}$ the set of edges (bonds), and $\mathcal{U} = \{\mathbf{u}\}$ the universal (global) attribute.

A set of functions (with learnable parameters) govern the three stages used in both training and inference of a graph net: *initialization*, *propagation*, and *readout*. The most general description of the message-passing procedure in the propagation stage could be found in (Battaglia et al., 2018), where node, edge, and global attributes $\mathbf{v}, \mathbf{e}, \mathbf{u}$ are updated according to:

$$\mathbf{e}_k^{(t+1)} = \phi^e(\mathbf{e}_k^{(t)}, \sum_{i \in \mathcal{N}_k^e} \mathbf{v}_i, \mathbf{u}^{(t)}), \qquad \text{edge update} \tag{2}$$

$$\bar{\mathbf{e}}_i^{(t+1)} = \rho^{e \to v}(E_i^{(t+1)}), \qquad \text{edge-to-node aggregate} \tag{3}$$

$$\mathbf{v}_i^{(t+1)} = \phi^v(\bar{\mathbf{e}}_i^{(t+1)}, \mathbf{v}_i^{(t)}, \mathbf{u}^{(t)}), \qquad \text{node update} \tag{4}$$

$$\bar{\mathbf{e}}^{(t+1)} = \rho^{e \to u}(E^{(t+1)}), \qquad \text{edge-to-global aggregate} \tag{5}$$

$$\bar{\mathbf{u}}^{(t+1)} = \rho^{v \to u}(V^{(t)}), \qquad \text{node-to-global aggregate} \tag{6}$$

$$\mathbf{u}^{(t+1)} = \phi^u(\bar{\mathbf{e}}^{(t+1)}, \bar{\mathbf{v}}^{(t+1)}, \mathbf{u}^{(t)}), \qquad \text{global update} \tag{7}$$

where $E_i = \{\mathbf{e}_k, k \in \mathcal{N}_i^v\}$ is the set of attributes of edges connected to a specific node, $E_i = \{e_k, k \in 1, 2, ..., N^e\}$ is the set of attributes of all edges, V is the set of attributes of all nodes, and $\mathcal{N}^v$ and $\mathcal{N}^e$ denote the set of indices of entities connected to a certain node or a certain edge, respectively. ϕ^e, ϕ^v, and ϕ^u are update functions that take the *environment* of the an entity as input and update the attribute of the entity, which could be stateful or not; $\rho^{e\to v}$, $\rho^{e\to u}$, and $\rho^{v\to u}$ are aggregate functions that aggregate the attributes of multiple entities into an *aggregated* attribute which shares the same dimension with each entity. Note that it is common that the edges do not hold attribute but only pass message onto neighboring nodes. For all models we survey here, edge-to-global update does not apply and global attribute does not present until the readout stage, when a sum function is applied to form the global representation ($\mathbf{u} = \sum V$). Under this set of grammar, we review the message-passing rules in 3.

Model	**Edge update ϕ^e**	**Edge aggregate $\rho^{e\to v}$**	**Node update ϕ^v**
GCN	Identity	Mean	NN
EdgeConv	$\text{ReLU}(W_0(\mathbf{v}_i - \mathbf{v}_j) + W_1\mathbf{v}_i)$	Max	Identity
GraphSAGE	Identity	Mean^2	$\text{Normalize}(\text{NN}([\mathbf{v} : \mathbf{e}]))$
GIN	Identity	Sum^3	$\text{NN}((1+\epsilon)\mathbf{v} + \mathbf{e})$

Table 3: **Summary of representative graph nets architectures by edge update, edge aggregate, and node update types.** Models analyzed here include: GCN (Kipf and Welling, 2016), EdgeConv (Wang et al., 2019), GraphSAGE (Hamilton et al., 2017), and GIN (Xu et al., 2018). Other architectures evaluated—TAGCN (Du et al., 2017) and SGC (Wu et al., 2019)—involve multi-step propagation, which could be expressed as a combination of these updates and aggregates.

10.2 Semi-Supervised Learning for Graph Networks

In practice, we have access to a very large number of chemical (2D) structures of synthetically accessible molecules that have never been synthesized or assayed, and hence lack associated measurement data—such as the Enamine REAL Space of billions of synthetically accessible compounds (Hoffmann and Gastreich, 2019). By contrast, molecules from this space that have been synthesized and assayed are few and far in between. We aim to leverage unlabelled data to train better deep models in supervised tasks that can more efficiently generalize from limited labeled data.

One of the most powerful unsupervised deep models is the variational auto-encoder (VAE) (Kingma and Welling, 2013; Kipf and Welling, 2016), a probabilistic version of the auto-encoder (AE) (Hinton and Salakhutdinov, 2006). An auto-encoder can be understood as a two-step process: The first (encoder) step maps the original data input from input feature space to a lower dimension representation, while the second (decoder) step maps the latent representation back to the input feature space. The goal is to learn a "meaningful" latent representation of the input data such that the decoder step can accurately reconstruct the original input data from the latent representation. Variational auto-encoders are also deep generative models: by making appropriate probabilistic assumptions of how the data

is generated, the VAE can learn parameters of a generative distribution, and thereby also generate new samples that are "similar" to existing data.

An important reason to consider incorporating generative model training is to avoid over-fitting during supervised learning; in the common data-limited regime, training a complex neural networks risks over-fitting. Another reason to incorporate generative model training is added stability during training deep models on supervised tasks; by using the encoder network of a trained VAE as part of the deep network for supervised training, we achieve improved stability, especially at the beginning of the training process. These latent representations are often useful, and have been used in various related applications, such as guiding molecular synthesis (Zhang et al., 2019).

The objective function used to train the VAE is the negative evidence lower bound (ELBO), which can be interpreted as consisting of a reconstruction loss and a regularization penalty:

$$\mathcal{L} = -\mathbb{E}_{P_\phi(Z|\mathcal{G})}[\log P_\Psi(\mathcal{G}|Z)] + KL(P_\phi(Z|\mathcal{G})\|P_0(Z)) \tag{8}$$

Here, $Z = \{z_1, ..., z_{|\mathcal{V}|}\}$ is the latent representation of the graph nodes, where z_i is the latent representation of node i. $P_\phi(Z|X)$ is the approximate posterior distribution of the latent representation Z of the input molecular graph $\mathcal{G}$. This approximate posterior distribution's parameters can be produced by a deep neural network, denoted as ϕ, or the encoder network. In our case, we use graph convolution neural networks (graph nets) to learn the parameters of the approximate posterior distribution. $P_0(Z)$ is the prior distribution on the latent representations—here, a standard normal distribution.

$P_\Psi(\mathcal{G}|Z)$ is the likelihood probability of the observed molecular graph, conditioned on the latent representation Z. Note that the parameters of this distribution could also be produced by a deep neural network, denoted as Ψ, or the decoder network. In our case, we follow Kipf and Welling (2016) and set:

$$\log P_\Psi(\mathcal{G}|Z) = \sum_{e\in\mathcal{E}} \log P_\Psi(e|Z) + \sum_{v\in\mathcal{V}} \log P_\Psi(v|Z) \tag{9}$$

In other words, the reconstruction loss factorizes into a sum over edges and nodes. In our model, suppose an edge e is between nodes v_i and v_j, the edge-specific likelihood term is $P_\Psi(e = (v_i, v_j)|Z) \sim \text{Bernoulli}(\sigma(z_i^\top z_j))$. On the other hand, we define the node-specific likelihood term $P_\Psi(v_i|Z) = P_\Psi(v_i|z_i)$ as a categorical distribution over the node type conditioned on latent node representation.

While VAEs were initially developed to perform unsupervised learning, one can easily extend these models to semi-supervised learning by incorporating targets into the ELBO. Suppose that a data is given as (graph $\mathcal{G}$, measurement y) pairs. The expected likelihood term in the negative ELBO loss would be defined as:

$$\log P_\Psi(\mathcal{G}, y|Z) = \sum_{e\in\mathcal{E}} \log P_\Psi(e|Z) + \sum_{v\in\mathcal{V}} \log P_\Psi(v|Z) + \log P_\Gamma(y|\mathcal{G}, Z) \tag{10}$$

where Γ is a regression model, which can be parameterized by deep Gaussian processes or deep neural networks.

To be able to train such a model on millions of molecules, we turn to stochastic optimization. At each iteration, we compute the stochastic gradient of the negative ELBO with

respect to the parameters of the neural networks ϕ and ψ using a mini-batch of input data. For our experiments, we use batch size of 32.

10.3 Bayesian Inference with Graph Nets

Under the Bayesian formalism, given sets of (graph, measurement) pairs as training data $\mathcal{D} = \{\mathcal{G}^{(i)}, y^{(i)}, i = 1, 2, 3, ..., n\}$, the probability distribution of the unknown quantity of the measurement $y^{(n+1)}$ could be modelled with respect to the posterior distribution of the neural network parameters θ as:

$$P(y^*|\mathcal{D}, \mathcal{G}^*) = \int P(y^*|\mathcal{G}^*, \theta) P(\theta|\mathcal{D}) \, \mathrm{d}\, \theta. \tag{11}$$

This integral, of course, is not tractable. If one uses the most likely set of neural network parameters $\theta^{\mathrm{MLE}} = \arg\max_{\theta} P(\theta|\mathcal{D})$, which is optimized by back-propagation, a standard neural network is recovered. We briefly survey the strategies to approximate (11).

10.3.1 Bayes-by-backprop

Bayes-by-backprop (Blundell et al., 2015) is one of the simplest types of variational inference, where the variational posterior approximating $P(\theta|\mathcal{D})$ is chosen to be a multivariate Gaussian with diagonal covariance matrix.

$$q(\theta) = \mathcal{N}(\mu, \sigma), \tag{12}$$

where the variational parameters $\{\mu, \sigma\}$ have the same dimension as θ and are trained by minimizing the Kullback-Leibler (KL) divergence between the variational and true Bayesian posterior:

$$\mu^*, \sigma^* = \arg\min_{\mu,\sigma} \mathcal{D}_{\mathrm{KL}}[q(\theta|\mu, \sigma)||P(\theta|\mathcal{D})] = \arg\min_{\mu,\sigma} \mathcal{D}_{\mathrm{KL}}[q(\theta|\mu, \sigma)||P(\theta)] - \mathbb{E}_{\theta\sim q(\mu,\sigma)}[\log P(\mathcal{D}|\theta)], \tag{13}$$

which is also termed a *variational free energy*. When training is complete, (11) can be written as:

$$P(y^*|\mathcal{D}, \theta) \approx \int P(y|\theta) q(\theta|\mu, \sigma), \tag{14}$$

which is tractable.

10.3.2 Kernel learning for Gaussian Process Regression

Graph nets could be incorporated seamlessly into Gaussian progress regression if we use the latent embeddings of graph nets as inputs to a fixed-dimensional kernel (Wilson et al., 2015). Concretely, if we model the real process that maps molecular graph to physical properties, $f : \mathcal{G} \to y$, as a Gaussian process,

$$f(\mathcal{G}) \sim \mathcal{GP}(\mathbf{0}, K(\mathcal{G}, \mathcal{G}')), \tag{15}$$

therefore the joint distribution of the function values $\mathbf{y}*$ associated with test inputs $\{\mathcal{G}^*\}$ could be written as:

$$\begin{bmatrix} \mathbf{y} \\ y^* \end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix} K(\{\mathcal{G}\}, \{\mathcal{G}\}) + \sigma_n^2 I & K(\{\mathcal{G}\}, \{\mathcal{G}^*\}) \\ K(\{\mathcal{G}^*\}, \{\mathcal{G}\}) & K(\{\mathcal{G}^*\}, \{\mathcal{G}^*\}) \end{bmatrix} \Big), \tag{16}$$

and (11) becomes:

$$\mathbf{y}^*|\mathcal{D}, \{\mathcal{G}\}, \{\mathcal{G}^*\} \sim \mathcal{N}(\mathbb{E}[y^*], \mathrm{cov}(y^*)), \tag{17}$$

where σ_n is the predictive noise and

$$\mathbb{E}[\mathbf{y}^*] = K(\mathcal{G}^*, \{\mathcal{G}\})[K(\{\mathcal{G}\}, \{\mathcal{G}\}) + \sigma_n^2 I]^{-1}\mathbf{y}; \tag{18}$$

$$\mathrm{cov}(\mathbf{y}^*) = K(\{\mathcal{G}^*\}, \{\mathcal{G}^*\}) - K(\{\mathcal{G}^*\}, \{\mathcal{G}\})[K(\{\mathcal{G}\}, \{\mathcal{G}\}) + \sigma_n^2 I]^{-1}K(\{\mathcal{G}\}, \{\mathcal{G}^*\}). \tag{19}$$

With a graph net with trainable parameters GN_θ, the kernel $K(\mathcal{G}, \mathcal{G}')$ could be defined as a kernel on fixed-dimensional space that takes the latent encoding provided by the graph net, $z = \mathrm{GN}_\theta(\mathcal{G})$, as input. If we choose the popular RBF kernel to be basis kernel, for example, a deep graph kernel could be defined as:

$$k(\mathcal{G}, \mathcal{G}') = k_{\mathrm{RBF}}(z, z') = k_{\mathrm{RBF}}(\mathrm{GN}_\theta(\mathcal{G}), \mathrm{GN}_\theta(\mathcal{G}')) = \exp(-\frac{1}{2}||\,\mathrm{GN}_\theta(\mathcal{G}) - \mathrm{GN}_\theta(\mathcal{G}')||/l^2). \tag{20}$$

As such, the parameters associated with the graph net θ could be viewed as part of the hyperparameter of the kernel and be optimized jointly by maximizing the likelihood of training data.

10.4 Acquisition Rules for Active Learning

10.4.1 Sequential Acquisition

Once the supervised learning model is trained and (11) is approximated by a reasonably likely region on the weight space, we could use the predictive distribution $P(y^*|\mathcal{G}^*, \theta)$ to prioritize the candidates to acquire in the next round of training. Popular acquisition functions that are applicable to sequential acquisition include (Snoek et al., 2012):

- **Probability of Improvement** characterizes the probability of the best current value,

$$\alpha_{\mathrm{PI}}(G^*|\mathcal{D}, \theta) = 1 - \Phi_{P(y^*|\mathcal{G}^*, \theta)}(\max(y)), \tag{21}$$

  where Φ denotes the CDF of the corresponding distribution, and $\max(y)$ is obtained within the training set $\mathcal{D} = \{\mathcal{G}_i, y_i\}$

- **Expected Improvement**, on the other hand, measures the expectation of the improvement over the current best,

$$\alpha_{\mathrm{EI}}(G^*|\mathcal{D}, \theta) = \mathbb{E}_{P(y^*|\mathcal{G}^*, \theta)} \min\{[y^*] - \max(y), 0\}. \tag{22}$$

- **Upper Confidence Bound** is a more recent idea that exploits confidence bounds to minimize the regret. Note that the confidence of the bound κ is a tunable parameter and could be used to balance exploitation against exploration.

$$\alpha_{\mathrm{UCB}}(\mathcal{G}^*|\mathcal{D}, \theta) = \mathrm{CI}_\kappa(P(y^*|\mathcal{G}^*, \theta)) \tag{23}$$

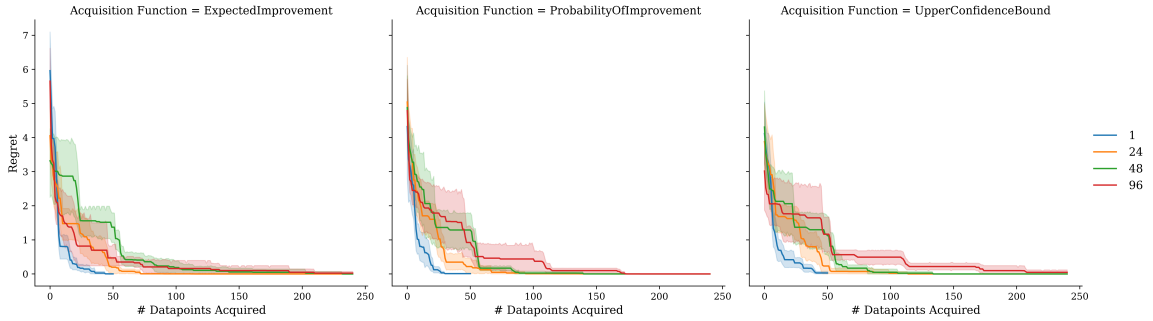

Figure 6: **The performance of batch acquisition function depends on the batch size on ESOL data.** Regret function on solubility measurement for ESOL dataset.

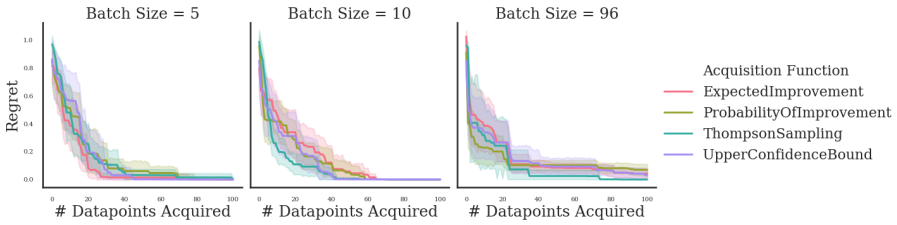

Figure 7: **The performance of batch acquisition function depends on the batch size on COVID moonshot data.**
Regret on measured SARS-CoV-2 Mpro inhibition at 20 μM compound from the Moonshot dataset (Section 3) is plotted against the number of candidates sequentially acquired. Expected improvement, probability of improvement, Thompson sampling and upper confidence interval ($\kappa = 0.95$) acquisition functions with **DGKL** model are evaluated

10.5 COVID Assay Details

In the scenario where the target of the drug molecule is identified as one specific protein, we can use the *binding free energy* to characterize the potency of the molecule, which is the free energy change associated with reaction,

$$\mathrm{L} + \mathrm{P} \rightleftharpoons \mathrm{LP} \tag{24}$$

where P stands for the protein, L the ligand (drug), and LP the complex they form.

$$\Delta G = G(\mathrm{LP}) - G(\mathrm{P}) - G(\mathrm{L}), \tag{25}$$

The quantity ΔG, is rarely directly measured in biological experiment. Popular assays include:

- Association constant (K_a) and dissociation constant (K_b):

$$K_a = \frac{[\mathrm{LP}]}{[\mathrm{L}][\mathrm{P}]} = \exp\frac{\Delta G}{-RT} \tag{26}$$

$$K_d = \frac{[\mathrm{L}][\mathrm{P}]}{[\mathrm{LP}]} = \exp\frac{\Delta G}{RT} \tag{27}$$

  where brackets $[\cdot]$ denote the concentration of certain species in solution, T is temperature in Kelvin, and R is the gas constant.

- Half maximal inhibitory concentration (IC_{50}): IC_{50} characterizes how much inhibitor is needed for a specific biological activity to be inhibited to 50% level. By the Cheng-Prusoff equation, if the concentration of the inhibitor when measured is close to the theoretical concentration when such %50 inhibition is achieved, $[L] \approx K_m$, IC_{50} could be related to ΔG by:

$$\mathrm{IC}_{50} = K_a(1 + \frac{K_m}{[L]}) \approx 2K_a = -2\ln\frac{\Delta G}{RT}, \tag{28}$$

  and its negative log value:

$$\mathrm{p}\,\mathrm{IC}_{50} = -\log_{10}\mathrm{IC}_{50} = -\log_{10}2 + \log_{10}e * RT\ln\frac{\Delta G}{RT}. \tag{29}$$

- Percentage inhibition ($\%_{\mathrm{inh}}$):

  The aforementioned two assays require a series of titration, whereas $\%_{\mathrm{inh}}$ describes a simplified relationship where the concentration of ligand [L] is measured at one value, and the concentration of protein [P] is low. This could be related to ΔG by:

$$\%_{\mathrm{inh}} = \frac{1}{1 + \exp(-\frac{\Delta G}{RT})\frac{c^\theta}{[L]}} \tag{30}$$

  where $[L]$ is the concentration at which the inhibition is measured, and c^θ is the unit concentration 1 mol / L.

In reality, there are scenarios where some of the assays are not applicable for subsets of molecules. For instance, many drug-like molecules are not soluble at 50 uM and therefore would not have an associated measured percentage of inhibition at such concentration. Each measurement has its own types of noise associated. Moreover, multiple types of instrument could be used for these experiments, introducing various types of noise.