GAUCHE: A Library for Gaussian Processes in Chemistry

Ryan-Rhys Griffiths[*]
University of Cambridge
rrg27@cam.ac.uk

Leo Klarner[*]
University of Oxford
leo.klarner@bnc.ox.ac.uk

Henry B. Moss[*]
Secondmind Labs
henry.moss@secondmind.ai

Aditya Ravuri[*]
University of Cambridge
ar847@cam.ac.uk

Sang Truong[*]
Stanford University
sttruong@cs.stanford.edu

Bojana Rankovic[*]
EPFL
bojana.rankovic@epfl.ch

Yuanqi Du[*]
Cornell University
yd392@cornell.edu

Arian Jamasb
University of Cambridge
arj39@cam.ac.uk

Julius Schwartz
University of Cambridge
julius.schwartz@cantab.net

Austin Tripp
University of Cambridge
ajt212@cam.ac.uk

Gregory Kell
King's College London
gregory.kell@kcl.ac.uk

Anthony Bourached
University College London
ucbab6@ucl.ac.uk

Alex J. Chan
University of Cambridge
ajc340@cam.ac.uk

Jacob Moss
University of Cambridge
jm2311@cam.ac.uk

Chengzhi Guo
University of Cambridge
cg711@cam.ac.uk

Alpha A. Lee
University of Cambridge
aal44@cam.ac.uk

Philippe Schwaller
EPFL
philippe.schwaler@epfl.ch

Jian Tang
MILA, CIFAR, HEC
jian.tang@hec.ca

Abstract

We introduce GAUCHE, a library for GAUssian processes in CHEmistry. Gaussian processes have long been a cornerstone of probabilistic machine learning, affording particular advantages for uncertainty quantification and Bayesian optimisation. Extending Gaussian processes to chemical representations however is nontrivial, necessitating kernels defined over structured inputs such as graphs, strings and bit vectors. By defining such kernels in GAUCHE, we seek to open the door to powerful tools for uncertainty quantification and Bayesian optimisation in chemistry. Motivated by scenarios frequently encountered in experimental chemistry, we showcase applications for GAUCHE in molecular discovery and chemical reaction optimisation. The codebase is made available at https://github.com/leojklarner/gauche

[*]equal contribution.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

1 Introduction

Early-stage scientific discovery is typically characterised by the small data regime due to the limited availability of high-quality experimental data [1, 2]. Much of the novelty of discovery relies on the fact that there is much knowledge to gain in the small data regime. By contrast, in the big data regime, discovery offers diminishing returns as much of the knowledge about the space of interest has already been acquired. As such, machine learning methodologies that facilitate search in small data regimes such as Bayesian optimisation (BO) [3, 4, 5, 6] and active learning (AL) [7, 8] have great potential to expedite the rate at which performant molecules, molecular materials, chemical reactions and proteins are discovered.

To date in molecular machine learning, Bayesian neural networks (BNNs) have been the surrogate of choice to produce the uncertainty estimates that underpin BO and AL [9, 7, 10, 11]. For small datasets, however, deep neural networks are often not the model of choice. Notably, certain deep learning experts have voiced a preference for Gaussian processes (GPs) in the small data regime [12]. Furthermore, for BO, GPs possess particularly advantageous properties; first, they admit exact as opposed to approximate Bayesian inference and second, few of their parameters need to be determined by hand. In the words of Sir David MacKay [13],

> "Gaussian processes are useful tools for automated tasks where fine tuning for each problem is not possible. We do not appear to sacrifice any performance for this simplicity."

The iterative model refitting required in BO makes it a prime example of such an automated task. Although BNN surrogates have been trialled for BO [14, 15], GPs remain the model of choice as evidenced by the results of the recent NeurIPS Black-Box Optimisation Competition [16].

Training GPs on molecular inputs is non-trivial however. Canonical applications of GPs assume continuous input spaces of low and fixed dimensionality. The most popular molecular input representations are SMILES/SELFIES strings [17, 18, 19], fingerprints [20, 21, 22] and graphs [23, 24]. Each of these input representations poses problems for GPs. SMILES strings have variable length, fingerprints are high-dimensional and sparse bit vectors, while graphs are also a form of non-continuous input. To construct a GP framework over molecules, GAUCHE provides GPU-based implementations of kernels that operate on molecular inputs, including string, fingerprint and graph kernels. Furthermore, GAUCHE includes support for protein and chemical reaction representations and interfaces with the GPyTorch [25] and BoTorch [26] libraries to facilitate usage for advanced probabilistic modelling and BO. Concretely, our contributions may be summarised as:

1. We propose a GP framework for molecules, chemical reactions and proteins.
2. We provide an open-source, GPU-enabled library building on GPyTorch [25], BoTorch [26] and RDKit [27].
3. We extend the use of black box graph kernels (from GraKel, [28]) to GP regression via a GPyTorch interface, along with a limited set of graph kernels implemented in native GPyTorch to enable optimisation of the graph kernel hyperparameters under the marginal likelihood.
4. We conduct benchmark experiments evaluating the utility of the GP framework on regression, uncertainty quantification and BO tasks.

GAUCHE includes tutorials to guide users through the tasks considered in this paper and is made available at https://github.com/leojklarner/gauche

2 Background

We summarise the background on Gaussian processes and common molecular representations here, leaving Bayesian optimisation, reaction and protein representations for Apppendix A.

2.1 Gaussian Processes

Notation: $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a design matrix of n training examples of dimension d. A given row i of the design matrix contains a training molecule's representation $\mathbf{x}_i$. A GP is specified by a mean function, $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a covariance function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}))]$. $K_\theta(\mathbf{X}, \mathbf{X})$ is a kernel matrix where entries are computed by the kernel function as $[K]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. θ represents the set of kernel hyperparameters. The GP specifies the full distribution over the function f to be modelled as

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big).$$

Prediction: The GP returns a predictive mean, $\bar{\mathbf{f}}_* = K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_y^2 I]^{-1}\mathbf{y}$ at test locations $\mathbf{X}_*$ and a predictive uncertainty $\text{cov}(\mathbf{f}_*) = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_y^2 I]^{-1}K(\mathbf{X}, \mathbf{X}_*)$.

Kernel Functions: The choice of kernel function is an important inductive bias for the properties of the function being modelled. A common choice for continuous input domains is the radial basis function kernel

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(\frac{-||\mathbf{x} - \mathbf{x}'||_2^2}{2\ell^2}\right),$$

where σ_f^2 is the signal amplitude hyperparameter (vertical lengthscale) and ℓ is the (horizontal) lengthscale hyperparameter. The symbol θ, introduced previously, is used to represent the set of kernel hyperparameters. For molecules, bespoke kernel functions will need to be defined for structured input spaces.

2.2 Molecular Representations

We review here the three main categories of molecular representations before describing the kernels that operate on them in section 3.

Graphs: Molecules may be represented as an undirected, labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where vertices $\mathcal{V} = \{v_1, \ldots, v_N\}$ represent the atoms of an N-atom molecule and edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represent covalent bonds between these atoms. Additional information may be incorporated in the form of vertex and edge labels $\mathcal{L} : \mathcal{V} \times \mathcal{E} \to \Sigma_V \times \Sigma_E$, with common label spaces including attributes such as atom types (i.e. hydrogen, carbon) as vertex labels and bond orders (i.e. single, double) as edge labels.

Fingerprints: Operate by assigning initial numeric identifiers to each atom in a molecule. These identifiers are subsequently updated in an iterative fashion based on the identifiers of their neighbours. The number of iterations corresponds to half the *diameter* of the fingerprint and the naming convention reflects this. For example, ECFP6 fingerprints have a diameter of 6, meaning that 3 iterations of atom identifier reassignment are performed. Each level of iteration appends substructural features of increasing non-locality to an array and the array is then hashed to a bit vector reflecting the presence of absence of those substructures in the molecule.

Strings: The Simplified Molecular-Input Line-Entry System (SMILES) is a text-based representation of molecules [17, 18]. Self-Referencing Embedded Strings (SELFIES) [19] is an alternative string representation to SMILES such that a bijective mapping exists between a SELFIES string and a molecule.

3 Molecular Kernels

Here we introduce examples of the classes of GAUCHE kernel designed to operate on the molecular representations introduced in section 2. We defer the discussion of string kernels to the appendix due to space constraints.

3.1 Fingerprint Kernels

Scalar Product Kernel: The simplest kernel to operate on fingerprints is the scalar product or linear kernel defined for vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ as

$$k_{\text{Scalar Product}}(\mathbf{x}, \mathbf{x}') \coloneqq \sigma_f^2 \cdot \langle \mathbf{x}, \mathbf{x}' \rangle,$$

where σ_f is a scalar signal variance hyperparameter and $\langle \cdot, \cdot \rangle$ is the Euclidean inner product.

Tanimoto Kernel: First introduced as a general similarity metric for binary attributes [29], the Tanimoto kernel was first used in chemoinformatics in conjunction with non-GP-based kernel methods [30]. It is defined for binary vectors $\mathbf{x}, \mathbf{x}' \in \{0, 1\}^d$ for $d \geq 1$ as

$$k_{\text{Tanimoto}}(\mathbf{x}, \mathbf{x}') \coloneqq \sigma_f^2 \cdot \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - \langle \mathbf{x}, \mathbf{x}' \rangle},$$

where $|| \cdot ||$ is the Euclidean norm.

3.2 Graph Kernels

Graph Kernels: Graph kernel methods $\phi_\lambda : \mathcal{G} \to \mathcal{H}$ map elements from a graph domain $\mathcal{G}$ to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, in which an inner product between a pair of graphs $g, g' \in \mathcal{G}$ is derived as a measure of similarity

$$k_{\text{Graph}}(g, g') \coloneqq \sigma^2 \cdot \langle \phi_\lambda(g), \phi_\lambda(g') \rangle_{\mathcal{H}},$$

where λ denotes kernel-specific hyperarameters and σ^2 is a scale factor. Depending on how ϕ_λ is defined [31], the kernel considers different substructural motifs and is characterised by different hyperparameters. Frequently-employed approaches include the random walk kernel [32], given by a geometric series over the count of matching random walks of increasing length with coefficient λ, and the Weisfeiler-Lehman kernel [33], given by the inner products of label count vectors over λ iterations of the Weisfeiler-Lehman algorithm.

4 Experiments

We evaluate GAUCHE on regression, uncertainty quantification (UQ) and BO. The principle goal in conducting regression and UQ benchmarks is to gauge whether performance on these tasks may be used as a proxy for BO performance. We provide the full results for regression and UQ in Appendix D.2 and Appendix D.3 respectively. We make use of the following datasets:

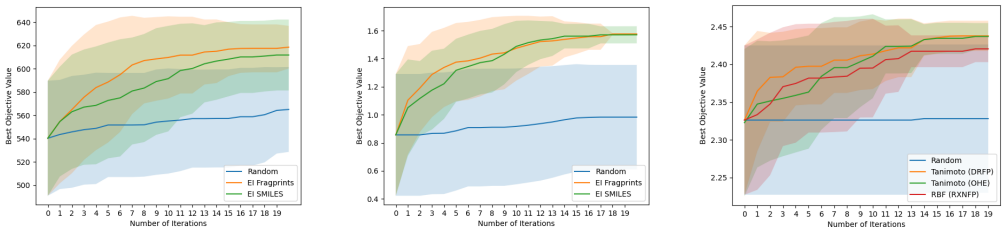
The Photoswitch Dataset: [2]: The labels, y are the experimentally-determined values of the E isomer $\pi - \pi^*$ transition wavelength for 392 photoswitch molecules.

ESOL: [34]: The labels y are the experimentally-determined logarithmic aqueous solubility values for 1128 organic small molecules.

Buchwald-Hartwig reactions: [35]: The labels y are the experimentally-determined yields for 3955 Pd-catalysed Buchwald–Hartwig C–N cross-couplings.

4.1 Bayesian Optimisation

We take forward two of the best-performing kernels from the regession and UQ benchmarks, the Tanimoto-fragprint kernel and the bag of SMILES kernel to undertake BO over the photoswitch and ESOL datasets. Random search is used as a baseline. BO is run for 20 iterations of sequential candidate selection (EI acquisition) where candidates are drawn from 95% of the dataset. The results are provided in Figure 1. The models are initialised with 5% of the dataset. In the case of the photoswitch dataset this corresponds to just 19 molecules. In this ultra-low data setting, common



(a) Photoswitch (b) ESOL (c) Buchwald-Hartwig reactions.

Figure 1: Bayesian optimisation performance. Standard error from 50 random initialisations, 20 for Buchwald-Hartwig.

to many areas of synthetic chemistry [2] both models outperform random search, highlighting the real-world use-case for such models in supporting human chemists prioritise candidates for synthesis. Furthermore, one may observe that BO performance is tightly coupled to regression and UQ performance. In the case of the photoswitch dataset, the better-performing Tanimoto model on regression and UQ also achieves relatively better BO performance. Additionally, we report results on the Buchwald-Hartwig reaction dataset.

5 Conclusions and Future Work

We have introduced GAUCHE, a library for GAUssian Processes in CHEmistry. In future work, we seek to:

1. Expand the range of GP kernels we currently consider, most notably to include *deep kernels* based on GNN embeddings.
2. Perform more extensive benchmarking for uncertainty quantification and active learning against models such as BNNs.
3. Exploit the benefits of our autodiff framework to facilitate the learning of graph kernel hyperparameters through the GP marginal likelihood.

6 Broader Impact

It is envisaged that our contribution, providing a platform for Gaussian process regression and Bayesian optimisation over molecules and chemical reactions, will contribute to expediting the rate at which novel and performant molecules are discovered. As such, the potential for negative impact of our contribution is tied to the potential for novel molecules to be directed towards nefarious activities such as chemical weapons manufacture. Nonetheless, given the broad range of societally beneficial applications of molecular discovery in areas such as drug design, therapeutics and vaccine development, we anticipate our work to yield a net benefit for social good.

References

[1] Ying Zhang and Chen Ling. A strategy to apply machine learning to small datasets in materials science. *Npj Computational Materials*, 2018.

[2] Aditya R Thawani, Ryan-Rhys Griffiths, Arian Jamasb, Anthony Bourached, Penelope Jones, William McCorkindale, Alexander A Aldrick, and Alpha A Lee. The photoswitch dataset: A molecular machine learning benchmark for the advancement of synthetic chemistry. *arXiv preprint arXiv:2008.03226*, 2020.

[3] 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*, 2018.

[4] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 2020.

[5] Benjamin J Shields, Jason Stevens, Jun Li, Marvin Parasram, Farhan Damani, Jesus I Martinez Alvarado, Jacob M Janey, Ryan P Adams, and Abigail G Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 2021.

[6] Yuanqi Du, Tianfan Fu, Jimeng Sun, and Shengchao Liu. Molgensurvey: A systematic survey in machine learning models for molecule design. *arXiv preprint arXiv:2203.14500*, 2022.

[7] Yao Zhang et al. Bayesian semi-supervised learning for uncertainty-calibrated prediction of molecular properties and active learning. *Chemical Science*, 2019.

[8] Kevin Maik Jablonka, Giriprasad Melpatti Jothiappan, Shefang Wang, Berend Smit, and Brian Yoo. Bias free multiobjective active learning for materials design and discovery. *Nature Communications*, 2021.

[9] Seongok Ryu, Yongchan Kwon, and Woo Youn Kim. A bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification. *Chemical Science*, 2019.

[10] Doyeong Hwang, Grace Lee, Hanseok Jo, Seyoul Yoon, and Seongok Ryu. A benchmark study on reliable molecular supervised learning via Bayesian learning. *arXiv preprint arXiv:2006.07021*, 2020.

[11] Gabriele Scalia, Colin A Grambow, Barbara Pernici, Yi-Pei Li, and William H Green. Evaluating scalable uncertainty estimation methods for deep learning-based molecular property prediction. *Journal of Chemical Information and Modeling*, 2020.

[12] Yoshua Bengio. *What are some Advantages of Using Gaussian Process Models vs Neural Networks?*, 2011. https://www.quora.com/What-are-some-advantages-of-using-Gaussian-Process-Models-vs-Neural-Networks.

[13] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[14] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

[15] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. *Advances in Neural Information Processing Systems*, 2016.

[16] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, 2021.

[17] Eric Anderson, Gilman D Veith, and David Weininger. SMILES, a line notation and computerized interpreter for chemical structures. *Environmental Research Laboratory*, 1987.

[18] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 1988.

[19] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 2020.

[20] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 2010.

[21] Daniel Probst and Jean-Louis Reymond. A probabilistic molecular fingerprint for big data settings. *Journal of Cheminformatics*, 2018.

[22] Alice Capecchi, Daniel Probst, and Jean-Louis Reymond. One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome. *Journal of Cheminformatics*, 2020.

[23] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in Neural Information Processing Systems*, 2015.

[24] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: Moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 2016.

[25] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[26] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 2020.

[27] Greg Landrum. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

[28] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in Python. *Journal of Machine Learning Research*, 2020.

[29] John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 1971.

[30] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks*, 2005.

[31] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 2021.

[32] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 2010.

[33] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[34] John S Delaney. ESOL: Estimating aqueous solubility directly from molecular structure. *Journal of Chemical Information and Computer Sciences*, 2004.

[35] Derek T Ahneman, Jesús G Estrada, Shishi Lin, Spencer D Dreher, and Abigail G Doyle. Predicting reaction performance in C–N cross-coupling using machine learning. *Science*, 2018.

[36] Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems*, 2001.

[37] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. In *Joint Automatic Control Conference*, 1963.

[38] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, 1975.

[39] A. Zhilinskas. Single-step Bayesian search method for an extremum of functions of a single variable. *Cybernetics*, 1975.

[40] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 1998.

[41] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[42] Antoine Grosnit, Alexander I Cowen-Rivers, Rasul Tutunov, Ryan-Rhys Griffiths, Jun Wang, and Haitham Bou-Ammar. Are we forgetting about compositional optimisers in Bayesian optimisation? *arXiv preprint arXiv:2012.08240*, 2020.

[43] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 2012.

[44] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in Neural Information Processing Systems*, 2014.

[45] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning*, 2017.

[46] Henry B Moss, David S Leslie, Javier Gonzalez, and Paul Rayson. Gibbon: General-purpose information-based Bayesian optimisation. *Journal of Machine Learning Research*, 2021.

[47] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 2016.

[48] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[49] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2002.

[50] Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. Word sequence kernels. *Journal of Machine Learning Research*, 2003.

[51] D-S Cao, J-C Zhao, Y-N Yang, C-X Zhao, J Yan, S Liu, Q-N Hu, Q-S Xu, and Y-Z Liang. In silico toxicity prediction by support vector machine and SMILES representation-based string kernel. *SAR and QSAR in Environmental Research*, 2012.

[52] Daniel Jurafsky and James H Martin. An introduction to natural language processing, computational linguistics, and speech recognition, 2000.

[53] Frederik Sandfort, Felix Strieth-Kalthoff, Marius Kühnemund, Christian Beecks, and Frank Glorius. A structure-based platform for predicting chemical reactivity. *Chem*, 2020.

[54] Kangway V Chuang and Michael J Keiser. Comment on "predicting reaction performance in c–n cross-coupling using machine learning". *Science*, 2018.

[55] Daniel Probst, Philippe Schwaller, and Jean-Louis Reymond. Reaction classification and yield prediction using the differential reaction fingerprint DRFP. *Digital Discovery*, 2022.

[56] Philippe Schwaller, Daniel Probst, Alain C Vaucher, Vishnu H Nair, David Kreutter, Teodoro Laino, and Jean-Louis Reymond. Mapping the space of chemical reactions using attention-based neural networks. *Nature Machine Intelligence*, 2021.

[57] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[58] Philippe Schwaller, Alain C Vaucher, Teodoro Laino, and Jean-Louis Reymond. Prediction of chemical reaction yields using deep learning. *Machine learning: Science and Technology*, 2021.

[59] Arian R. Jamasb, Ramon Viñas, Eric J. Ma, Charlie Harris, Kexin Huang, Dominic Hall, Pietro Lió, and Tom L. Blundell. Graphein - a Python library for geometric deep learning and network analysis on protein structures and interaction networks. *bioRxiv*, 2021.

[60] Henry B. Moss and Ryan Rhys Griffiths. Gaussian process molecule property prediction with FlowMO. *arXiv preprint arXiv:2010.01118*, 2020.

[61] GPy. GPy: A Gaussian process framework in Python. http://github.com/SheffieldML/GPy, since 2012.

[62] A. G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 2017.

[63] Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv preprint arXiv:2003.01115*, 2020.

[64] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust Bayesian optimisation with Dragonfly. *Journal of Machine Learning Research*, 2020.

[65] Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. An empirical study of assumptions in Bayesian optimisation. *arXiv preprint arXiv:2012.03826*, 2020.

[66] Edward O Pyzer-Knapp. Using Bayesian optimization to accelerate virtual screening for the discovery of therapeutics appropriate for repurposing for COVID-19. *arXiv preprint arXiv:2005.07121*, 2020.

[67] David E Graff, Matteo Aldeghi, Joseph A Morrone, Kirk E Jordan, Edward O Pyzer-Knapp, and Connor W Coley. Self-focusing virtual screening with active design space pruning. *arXiv preprint arXiv:2205.01753*, 2022.

[68] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019.

[69] Mufei Li, Jinjing Zhou, Jiajing Hu, Wenxuan Fan, Yangkang Zhang, Yaxin Gu, and George Karypis. DGL-LifeSci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*, 2021.

[70] Zhaocheng Zhu, Chence Shi, Zuobai Zhang, Shengchao Liu, Minghao Xu, Xinyu Yuan, Yangtian Zhang, Junkun Chen, Huiyu Cai, Jiarui Lu, Chang Ma, Runcheng Liu, Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. TorchDrug: A powerful and flexible machine learning platform for drug discovery. *arXiv preprint arXiv:2202.08320*, 2022.

[71] Lauri Himanen, Marc OJ Jäger, Eiaki V Morooka, Filippo Federici Canova, Yashasvi S Ranawat, David Z Gao, Patrick Rinke, and Adam S Foster. DScribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, 2020.

[72] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 2013.

[73] Bingqing Cheng, Ryan-Rhys Griffiths, Simon Wengert, Christian Kunkel, Tamas Stenczel, Bonan Zhu, Volker L Deringer, Noam Bernstein, Johannes T Margraf, Karsten Reuter, et al. Mapping materials and molecules. *Accounts of Chemical Research*, 2020.

[74] Linlin Jia, Benoit Gaüzère, and Paul Honeine. graphkit-learn: A Python library for graph kernels based on linear patterns. *Pattern Recognition Letters*, 2021.

[75] Mahito Sugiyama, M Elisabetta Ghisu, Felipe Llinares-López, and Karsten Borgwardt. graphkernels: R and Python packages for graph comparison. *Bioinformatics*, 2018.

[76] Mahito Sugiyama and Karsten Borgwardt. Halting in random walk kernels. *Advances in Neural Information Processing Systems*, 2015.

[77] Benoit Gaüzére, Luc Brun, Didier Villemin, and Myriam Brun. Graph kernels based on relevant patterns and cycle information for chemoinformatics. In *Proceedings of the 21st International Conference on Pattern Recognition*, 2012.

[78] Swarnendu Ghosh, Nibaran Das, Teresa Gonçalves, Paulo Quaresma, and Mahantapas Kundu. The journey of graph kernels through two decades. *Computer Science Review*, 2018.

[79] Ryan-Rhys Griffiths, Philippe Schwaller, and Alpha A Lee. Dataset bias in the natural sciences: A case study in chemical reaction prediction and synthesis design. *arXiv preprint arXiv:2105.02637*, 2021.

[80] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 2020.

[81] Aryan Deshwal and Jana Doppa. Combining latent space and structured kernels for Bayesian optimization over combinatorial spaces. *Advances in Neural Information Processing Systems*, 2021.

[82] Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I Cowen-Rivers, Lin Yang, Lin Zhu, Wenlong Lyu, Zhitang Chen, Jun Wang, et al. High-dimensional Bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*, 2021.

[83] Ekansh Verma and Souradip Chakraborty. Uncertainty-aware labelled augmentations for high dimensional latent space Bayesian optimization. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[84] Natalie Maus, Haydn T Jones, Juston S Moore, Matt J Kusner, John Bradshaw, and Jacob R Gardner. Local latent space Bayesian optimization over structured inputs. *arXiv preprint arXiv:2201.11872*, 2022.

[85] Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. *arXiv preprint arXiv:2203.12742*, 2022.

[86] Edward O Pyzer-Knapp, Changwon Suh, Rafael Gómez-Bombarelli, Jorge Aguilera-Iparraguirre, and Alán Aspuru-Guzik. What is high-throughput virtual screening? A perspective from organic materials discovery. *Annual Review of Materials Research*, 2015.

[87] Henry Moss, David Leslie, Daniel Beck, Javier Gonzalez, and Paul Rayson. BOSS: Bayesian optimization over string spaces. *Advances in Neural Information Processing Systems*, 2020.

[88] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[89] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, and Alán Aspuru-Guzik. Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge. *Applied Physics Reviews*, 2021.

[90] 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 *International Conference on Machine Learning*, 2017.

[91] Sattar Vakili, Henry Moss, Artem Artemev, Vincent Dutordoir, and Victor Picheny. Scalable Thompson sampling using sparse Gaussian process models. *Advances in Neural Information Processing Systems*, 2021.

[92] Andrew F Zahrt, Jeremy J Henle, Brennan T Rose, Yang Wang, William T Darrow, and Scott E Denmark. Prediction of higher-selectivity catalysts by computer-driven workflow and machine learning. *Science*, 2019.

[93] Artur M Schweidtmann, Adam D Clayton, Nicholas Holmes, Eric Bradford, Richard A Bourne, and Alexei A Lapkin. Machine learning meets continuous flow chemistry: Automated optimization towards the pareto front of multiple objectives. *Chemical Engineering Journal*, 2018.

[94] Pia Müller, Adam D Clayton, Jamie Manson, Samuel Riley, Oliver S May, Norman Govan, Stuart Notman, Steven V Ley, Thomas W Chamberlain, and Richard A Bourne. Automated multi-objective reaction optimisation: Which algorithm should I use? *Reaction Chemistry & Engineering*, 2022.

[95] Kobi C Felton, Jan G Rittig, and Alexei A Lapkin. Summit: Benchmarking machine learning methods for reaction optimisation. *Chemistry-Methods*, 2021.

[96] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, Melodie Christensen, Elena Liles, Jason E Hein, and Alán Aspuru-Guzik. Olympus: A benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2021.

[97] Alexe L Haywood, Joseph Redshaw, Magnus WD Hanson-Heine, Adam Taylor, Alex Brown, Andrew M Mason, Thomas Gärtner, and Jonathan D Hirst. Kernel methods for predicting yields of chemical reactions. *Journal of Chemical Information and Modeling*, 2021.

[98] Alexander Pomberger, Antonio Pedrina McCarthy, Ahmad Khan, Simon Sung, Connor Taylor, Matthew Gaunt, Lucy Colwell, David Walz, and Alexei Lapkin. The effect of chemical representation on active machine learning towards closed-loop optimization. *ChemRxiv*, 2022.

[99] Riley Hickman, Jurgis Ruža, Loïc Roch, Hermann Tribukait, and Alberto García-Durán. Equipping data-driven experiment planning for self-driving laboratories with semantic memory: Case studies of transfer learning in chemical reaction optimization. *ChemRxiv*, 2022.

[100] David L Mobley and J Peter Guthrie. FreeSolv: A database of experimental and calculated hydration free energies, with input files. *Journal of Computer-Aided Molecular Design*, 2014.

[101] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 2012.

[102] A Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J Bellis, Jon Chambers, Mark Davies, Felix A Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, et al. The ChEMBL bioactivity database: An update. *Nucleic Acids Research*, 2014.

[103] Damith Perera, Joseph W Tucker, Shalini Brahmbhatt, Christopher J Helal, Ashley Chong, William Farrell, Paul Richardson, and Neal W Sach. A platform for automated nanomole-scale reaction screening and micromole-scale synthesis in flow. *Science*, 2018.

[104] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 1989.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Contents

A Further Background

A.1 Gaussian Process Training

Hyperparameters for Gaussian processes comprise kernel hyperparameters, θ in addition to the likelihood noise, σ_y^2. These hyperparameters are chosen by optimising an objective function known as the negative log marginal likelihood (NLML)

$$\log p(\mathbf{y}|\mathbf{X},\theta) = \underbrace{-\frac{1}{2}\mathbf{y}^\top (K_\theta(\mathbf{X},\mathbf{X}) + \sigma_y^2 I)^{-1}\mathbf{y}}_{\text{encourages fit with data}} \underbrace{-\frac{1}{2}\log|K_\theta(\mathbf{X},\mathbf{X}) + \sigma_y^2 I|}_{\text{controls model capacity}} - \frac{N}{2}\log(2\pi).$$

$I\sigma_y^2$ represents the variance of i.i.d. Gaussian noise on the observations $\mathbf{y}$. The NLML embodies Occam's razor for Bayesian model selection [36] in favouring models that fit the data without being overly complex.

A.2 Bayesian Optimisation

In molecular discovery campaigns we are typically interested in solving problems of the form

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

where $f(\cdot) : \mathcal{X} \to \mathbb{R}$ is an expensive black-box function over a structured input domain $\mathcal{X}$. In our example setting the structured input domain consists of a set of molecular representations (graphs, strings, bit vectors) and the expensive black-box function is a property of interest for a given molecule that we wish to optimise. Bayesian optimisation (BO) [37, 38, 39, 40, 41, 42] is a data-efficient methodology for determining $\mathbf{x}^\star$. BO operates sequentially by selecting input locations at which to query the black-box function f with the aim of identifying the optimum in as few queries as possible. This procedure involves the exploration/exploitation tradeoff in the sense that exploiting knowledge about the function to propose promising locations competes with the desire to learn more about the function in unobserved locations.

The two components of a BO scheme are a probabilistic surrogate model and an acquisition function. The surrogate model is typically chosen to be a GP due to its ability to maintain calibrated uncertainty estimates through exact Bayesian inference. The uncertainty estimates of the surrogate model are then leveraged by the acquisition function to propose new input locations to query. The acquisition function is a heuristic that trades off exploration and exploitation, well-known examples of which include expected improvement (EI) [38, 40] and entropy search [43, 44, 45, 46]. After the acquisition function proposes an input location, the black-box is evaluated at that location, the surrogate model is retrained and the process repeats *ad libitum* until a solution is obtained. Systematic reviews of the BO literature include [41, 47, 48]

A.3 String Kernels

String kernels [49, 50] measure the similarity between strings by examining the degree at which their sub-strings differ. In GAUCHE, we implement the SMILES string kernel [51] which calculates an inner product between the occurrences of sub-strings, considering all contiguous sub-strings made from at most n characters (we set $n = 5$ in our experiments). Therefore, for the sub-string count featurisation $\phi : \mathcal{S} \to \mathbb{R}^p$ (also known as a bag-of-characters representation [52]), the SMILES string kernel between two strings $\mathcal{S}$ and $\mathcal{S}'$ is given by

$$k_{\text{String}}(\mathcal{S},\mathcal{S}') \coloneqq \sigma^2 \cdot \langle \phi(\mathcal{S}), \phi(\mathcal{S}') \rangle.$$

Note that although named the SMILES string kernel, this kernel can also be applied to any other string representation of molecules e.g. SELFIES.

A.4 Fragment and Fragprint Representations

We make use of fragment descriptors which are count vectors, each component of which indicates the number of a certain functional group present in a molecule. For example row 1 of the count vector could be an integer representing the number of aliphatic hydroxl groups present in the molecule. We make use of both fingerprint and fragment features computed using RDKit [27] as well as the concatenation of the fingerprint and fragment feature vectors, a representation termed fragprints [2] which has shown strong empirical performance. Example representations $\mathbf{x_f}$ for fingerprints and $\mathbf{x_{fr}}$ for fragments are given as

$$\mathbf{x_f} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{x_{fr}} = \begin{bmatrix} 3 \\ 0 \\ \vdots \\ 2 \end{bmatrix}.$$

A.5 Reaction Representations

Chemical reactions consist of multiple reactants and reagents that transform into one or more products. The reactants and reagents can often be categorised into different types. Taking the high-throughput experiments by [35] on Buchwald-Hartwig reactions as an example, the reaction design space consists of 15 aryl and heteroaryl halides, 4 Buchwald ligands, 3 bases, and 23 isoxazole additives.

Concatenated molecular representations: If the number of reactant and reagent categories is constant, the molecular representations discussed above may be used to encode the selected reactants and reagents, and the vectors for the individual reaction components can be concatenated to build the reaction representation [35, 53]. An additional and commonly-used concatenated representation, is the one-hot-encoding (OHE) of the reaction categories where bits specify which of the components in the different reactant and reagent categories is present. In the Buchwald-Hartwig example, the OHE would describe which of the aryl halides, Buchwald ligands, bases and additives are used in the reaction, resulting in a 44-dimensional bit vector [54].

Differential reaction fingerprints: Inspired by the hand-engineered difference reaction fingerprints by schneider2015development, [55] recently introduced the differential reaction fingerprint (DRFP). This reaction fingerprint is constructed by taking the symmetric difference of the sets containing the molecular substructures on both sides of the reaction arrow. Reagents are added to the reactants. The size of the reaction bit vector generated by DRFP is independent of the number of reaction components.

Data-driven reaction fingerprints: [56] described data-driven reaction fingerprints using Transformer models (e.g. BERT [57]) trained in a supervised or an unsupervised fashion on reaction SMILES. Those models can be fine-tuned on the task of interest to learn more specific reaction representations [58] (RXNFP). Similar to the DRFP, the size of the data-driven reaction fingerprints is independent of the number of reaction components.

A.6 Protein Representations

Proteins are large macromolecules that adopt complex 3D structures. Proteins can be represented in string form describing the underlying amino acid sequence. Graphs at varying degrees of coarseness may be used for structural representations that capture spatial and intramolecular relationships between structural elements, such as atoms, residues, secondary structures and chains. GAUCHE interfaces with Graphein [59], a library for pre-processing and computing graph representations of structural biological data thereby enabling the application of graph kernel-based methods to protein structure.

B Related Work

General-purpose GP and Bayesian optimisation libraries do not specifically cater for molecular representations. Likewise, general-purpose molecular machine learning libraries do not specifically consider GPs and Bayesian optimisation. Here, we review existing libraries, highlighting the niche GAUCHE fills in bridging the GP and molecular machine learning communities.

The closest work to ours is FlowMO [60], which introduces a molecular GP library in the GPflow framework. It is on this project which we build, extending the scope of the library to a broader class of molecular representations (graphs), problem settings (Bayesian optimisation) and applications (reaction optimisation and protein engineering).

Gaussian Process Libraries: GP libraries include GPy (Python) [61], GPflow (TensorFlow) [62, 63] and GPyTorch (PyTorch) [25] while examples of recent Bayesian optimisation libraries include BoTorch (PyTorch) [26], Dragonfly (Python) [64] and HEBO (PyTorch) [65]. The aforementioned libraries do not explicitly support molecular representations. Extension to cover molecular representations however is nontrivial, requiring implementations of bespoke GP kernels for bit vector, string and graph inputs together with modifications to Bayesian optimisation schemes to consider acquisition function evaluations over a discrete set of heldout molecules, a setting commonly encountered in virtual screening campaigns [66, 67].

Molecular Machine Learning Libraries: Molecular machine learning libraries include DeepChem [68], DGL-LifeSci [69] and TorchDrug [70]. DeepChem features a broad range of model implementations and tasks, while DGL-LifeSci focuses on graph neural networks. TorchDrug caters for applications including property prediction, representation learning, retrosynthesis, biomedical knowledge graph reasoning and molecule generation.

GP implementations are not included, however, in the aforementioned libraries. In terms of atomistic systems, DScribe [71] features, amongst other methods, the Smooth Overlap of Atomic Positions (SOAP) representation [72] which is typically used in conjunction with a GP model to learn atomistic properties. Automatic Selection And Prediction (ASAP) [73] also principally focusses on atomistic properties as well as dimensionality reduction and visualisation techniques for materials and molecules. Lastly, the Graphein library focusses on graph representations of proteins [59].

Graph Kernel Libraries: Graph kernel libraries include GraKel [28], graphkit-learn [74], graphkernels [75], graph-kernels [76], pykernels (https://github.com/gmum/pykernels) and ChemoKernel [77]. The aforementioned libraries focus on CPU implementations in Python. Extending graph kernel computation to GPUs has been noted as an important direction for future research [78]. In our work, we build on the GraKel library by interfacing it with GPyTorch, facilitating GP regression with GPU computation. Furthermore, we enable the graph kernel hyperparameters to be learned through the marginal likelihood objective as opposed to being pre-specified and fixed upfront.

Molecular Bayesian Optimisation: BO over molecular space can be divided into two classes of methods. In the first class, molecules are encoded into the latent space of a variational autoencoder (VAE) [3]. BO is then performed over the continuous latent space and queried molecules are decoded back to the original space. Much work on VAE-BO has focussed on improving the synergy between the surrogate model and the VAE [79, 4, 80, 81, 82, 83, 84, 85]. One of the defining characteristics of VAE-BO is that it enables the generation of new molecular structures.

In the second class of methods, BO is performed directly over the original discrete space of molecules. In this setting it is not possible to generate new structures and so a candidate set of queryable molecules is defined. The inability to generate new structures however, is not a bottleneck to molecule discovery as the principle concern is how best to explore existing candidate sets. These candidate sets are also known as molecular libraries in the virtual screening literature [86].

To date, there has been little work on BO directly over discrete molecular spaces. In [87], the authors use a string kernel GP trained on SMILES to perform BO to select from a candidate set of molecules. In [88], an optimal transport kernel GP is used for BO over molecular graphs. In [89] a surrogate based on the Nadarya-Watson estimator is defined such that the kernel density estimates are inferred using BNNs. The model is then trained on molecular descriptors. Lastly, in [90] and [91] a BNN and

a sparse GP respectively are trained on fingerprint representations of molecules. In the case of the sparse GP the authors select an ArcCosine kernel. It is a long term aim of the GAUCHE Project to compare the efficacy of VAE-BO against vanilla BO on real-world molecule discovery tasks.

Chemical Reaction Optimisation: Chemical reactions describe how reactant molecules transform into product molecules. Reagents (catalysts, solvents, and additives) and reaction conditions heavily impact the outcome of chemical reactions. Typically the objective is to maximise the reaction yield (the amount of product compared to the theoretical maximum) [35], in asymmetric synthesis, where the reactions could result in different enantiomers, to maximise the enantiomeric excess [92], or to minimise the E-factor, which is the ratio between waste materials and the desired product [93].

A diverse set of studies have evaluated the optimisation of chemical reactions in single and multi-objective settings [93, 94]. [95] and [96] benchmarked reaction optimisation algorithms in low-dimensional settings including reaction conditions, such as time, temperature, and concentrations. [5] suggested BO as a general tool for chemical reaction optimisation and benchmarked their approach against human experts. [97] compared the yield prediction performance of different kernels and [98] the impact of various molecular representations.

In all reaction optimisation studies above, the representations of the different categories of reactants and reagents are concatenated to generate the reaction input vector, which could lead to limitations if another type of reagent is suddenly considered. Moreover, most studies concluded that simple one-hot encodings (OHE) perform at least on par with more elaborate molecular representations in the low-data regime [5, 98, 99]. In GAUCHE, we introduce reaction fingerprint kernels, based on existing reaction fingerprints [56, 55] and work independently of the number of reactant and reagent categories.

C Coding Kernels in GAUCHE

We provide an example of the class definition for the Tanimoto kernel in GAUCHE below

```
class TanimotoGP(ExactGP):
    def __init__(self, train_x, train_y, likelihood):
        super(TanimotoGP, self).__init__(train_x,
                                         train_y,
                                         likelihood)
        self.mean_module = ConstantMean()
        # We use the Tanimoto kernel to work with
        # molecular fingerprint representations
        self.covar_module = ScaleKernel(TanimotoKernel())

    def forward(self, x):
        mean_x = self.mean_module(x)
        covar_x = self.covar_module(x)
        return MultivariateNormal(mean_x, covar_x)
```

and an example definition of a black box kernel (where gradients with respect to hyperparameters and input labels are not required).

```
class WLKernel(gauche.Kernel):
    def __init__(self):
        super().__init__()
        self.kernel = grakel.kernels.WeisfeilerLehman()

    @lru_cache(maxsize=3)
    def kern(self, X):
        return tensor(self.kernel.fit_transform(X.data))

class GraphGP(gauche.SIGP):
    def __init__(self, train_x, train_y, likelihood):
        super().__init__(train_x, train_y, likelihood)
        self.mean = ConstantMean()
        self.covariance = WLKernel()

    def forward(self, X):
        # X is a gauche.Inputs instance, with X.data
        # holding a list of grakel.Graph instances.
        mean = self.mean(zeros(len(X.data), 1))
        covariance = self.covariance(X)
        return MultivariateNormal(mean, covariance)
```

Importantly, GAUCHE inherits all the facilities of GPyTorch and GraKel allowing a broad range of of models to be defined on molecular inputs such as deep GPs, multioutput GPs and heteroscedastic GPs.

D Further Experiments

D.1 Further Datasets

We include here the additional datasets used for the regression and uncertainty quantification benchmarks.

FreeSolv: [100]: The labels y are the experimentally-determined hydration free energies for 642 molecules.

Lipophilicity: Thpe labels y are the experimentally-determined octanol/water distribution coefficient (log D at pH 7.4) of 4200 compounds curated from the ChEMBL database [101, 102].

Suzuki-Miyaura reactions: [103]: The labels y are the experimentally-determined yields for 5760 Pd-catalysed Suzuki-Miyaura C-C cross-couplings.

D.2 Regression

The regression results for molecular property prediction are reported in Table D1 and for reaction yield prediction in Table D3 of subsection D.4. The datasets are split in a train/test ratio of 80/20 (note that validation sets are not required for the GP models since training uses the marginal likelihood objective). Errorbars represent the standard error across 20 random initialisations. All GP models are trained using the L-BFGS-B optimiser [104]. If not mentioned, default settings in the GPyTorch and BoTorch libraries apply. For the SELFIES representation, some molecules could not be featurised and corresponding entries are left blank. The results of Table D3 indicate that the best choice of representation (and hence the choice of kernel) is task-dependent.

Table D1: Molecular property prediction regression benchmark. RMSE values for 80/20 train/test split across 20 random trials.

GP Model		**Dataset**			
Kernel	Representation	Photoswitch	ESOL	FreeSolv	Lipophilicity
Tanimoto	fragprints	$\mathbf{20.9 \pm 0.7}$	0.71 ± 0.01	1.31 ± 0.06	$\mathbf{0.67 \pm 0.01}$
	fingerprints	23.4 ± 0.8	1.01 ± 0.01	1.93 ± 0.09	0.76 ± 0.01
	fragments	26.3 ± 0.8	0.91 ± 0.01	1.49 ± 0.05	0.80 ± 0.01
Scalar Product	fragprints	22.5 ± 0.7	0.88 ± 0.01	$\mathbf{1.27 \pm 0.02}$	0.77 ± 0.01
	fingerprints	24.8 ± 0.8	1.17 ± 0.01	1.93 ± 0.07	0.84 ± 0.01
	fragments	36.6 ± 1.0	1.15 ± 0.01	1.63 ± 0.03	$0.97. \pm 0.01$
String	SELFIES	24.9 ± 0.6	-	-	-
	SMILES	24.8 ± 0.7	$\mathbf{0.66 \pm 0.01}$	1.31 ± 0.01	$\mathbf{0.68 \pm 0.01}$
WL Kernel (GraKel)	graph	22.4 ± 1.4	1.04 ± 0.02	1.47 ± 0.06	0.74 ± 0.05

D.3 Uncertainty Quantification (UQ)

To quantify the quality of the uncertainty estimates we use three metrics, the negative log predictive density (NLPD), the mean standardised log loss (MSLL) and the quantile coverage error (QCE). We provide the NLPD results in Table D2 and defer the MSLL and QCE results to subsection D.5. One trend to note is that uncertainty estimate quality is roughly correlated with regression performance.

Table D2: UQ benchmark. NLPD values for 80/20 train/test split across 20 random trials.

GP Model		Dataset			
Kernel	Representation	Photoswitch	ESOL	FreeSolv	Lipophilicity
Tanimoto	fragprints	$\mathbf{0.22 \pm 0.03}$	0.33 ± 0.01	0.28 ± 0.02	$\mathbf{0.71 \pm 0.01}$
	fingerprints	0.33 ± 0.03	0.71 ± 0.01	0.58 ± 0.03	0.85 ± 0.01
	fragments	0.50 ± 0.04	0.57 ± 0.01	0.44 ± 0.03	0.94 ± 0.02
Scalar Product	fragprints	$\mathbf{0.23 \pm 0.03}$	0.53 ± 0.01	0.25 ± 0.02	0.92 ± 0.01
	fingerprints	0.33 ± 0.03	0.84 ± 0.01	0.64 ± 0.03	1.03 ± 0.01
	fragments	0.80 ± 0.03	0.82 ± 0.01	0.54 ± 0.02	0.88 ± 0.10
String	SELFIES	0.37 ± 0.04	-	-	-
	SMILES	0.30 ± 0.04	$\mathbf{0.29 \pm 0.03}$	$\mathbf{0.16 \pm 0.02}$	$\mathbf{0.72 \pm 0.01}$
WL Kernel (GraKel)	graph	0.39 ± 0.11	0.76 ± 0.001	0.47 ± 0.02	-

D.4 Chemical Reaction Yield Prediction Experiments

Further regression and uncertainty quantification experiments are presented in Table D3. The differential reaction fingerprint in conjunction with the Tanimoto kernel is the best-performing reaction representation.

Table D3: Chemical reaction regression benchmark. 80/20 train/test split across 20 random trials.

GP Model		Buchwald-Hartwig			
Kernel	Representation	RMSE ↓	R^2 score ↑	MSLL ↓	QCE ↓
Tanimoto	OHE	7.94 ± 0.05	0.91 ± 0.001	-0.06 ± 0.002	0.011 ± 0.001
	DRFP	$\mathbf{6.48 \pm 0.45}$	$\mathbf{0.94 \pm 0.015}$	$\mathbf{-0.15 \pm 0.07}$	0.027 ± 0.002
Scalar Product	OHE	15.23 ± 0.052	0.69 ± 0.002	0.57 ± 0.002	0.008 ± 0.001
	DRFP	14.63 ± 0.050	0.71 ± 0.002	0.55 ± 0.002	0.010 ± 0.001
RBF	RXNFP	10.79 ± 0.049	0.84 ± 0.001	0.37 ± 0.005	0.024 ± 0.001
		Suzuki-Miyaura			
Tanimoto	OHE	11.18 ± 0.036	0.83 ± 0.001	0.23 ± 0.001	0.007 ± 0.001
	DRFP	11.46 ± 0.038	0.83 ± 0.001	0.25 ± 0.006	0.019 ± 0.000
Scalar Product	OHE	19.91 ± 0.042	0.47 ± 0.003	0.82 ± 0.001	0.012 ± 0.001
	DRFP	19.66 ± 0.042	0.52 ± 0.003	0.81 ± 0.001	0.014 ± 0.001
RBF	RXNFP	13.83 ± 0.048	0.75 ± 0.002	0.50 ± 0.001	0.007 ± 0.001

D.5 Uncertainty Quantification Experiments

In Table D4 and Table D5 we present further uncertainty quantification metrics. Numerical errors were encountered with the WL kernel on the large lipophilicity dataset which invalidated the results and so the corresponding entry is left blank. The native random walk kernel was discontinued (for the time being) due to poor performance!

Table D4: UQ Benchmark. MSLL Values (↓) for 80/20 Train/Test Split.

GP Model		**Dataset**			
Kernel	Representation	Photoswitch	ESOL	FreeSolv	Lipophilicity
Tanimoto	fragprints	$\mathbf{0.06 \pm 0.01}$	0.17 ± 0.04	0.16 ± 0.02	$\mathbf{0.50 \pm 0.006}$
	fingerprints	0.16 ± 0.01	0.55 ± 0.01	0.42 ± 0.02	0.63 ± 0.004
	fragments	0.27 ± 0.01	0.34 ± 0.04	0.24 ± 0.02	0.72 ± 0.003
Scalar Product	fragprints	0.03 ± 0.01	0.32 ± 0.004	0.06 ± 0.01	0.67 ± 0.003
	fingerprints	0.11 ± 0.01	0.64 ± 0.006	0.41 ± 0.02	0.79 ± 0.003
	fragments	0.56 ± 0.01	0.58 ± 0.005	0.29 ± 0.01	0.94 ± 0.003
String	SELFIES	0.13 ± 0.01	-	-	-
	SMILES	$\mathbf{0.08 \pm 0.02}$	$\mathbf{0.03 \pm 0.005}$	$\mathbf{0.03 \pm 0.02}$	$\mathbf{0.52 \pm 0.002}$
WL Kernel (GraKel)	graph	0.14 ± 0.03	0.54 ± 0.01	0.26 ± 0.01	-

Table D5: UQ benchmark. QCE values (↓) for 80/20 train/test split across 20 random trials.

GP Model		**Dataset**			
Kernel	Representation	Photoswitch	ESOL	FreeSolv	Lipophilicity
Tanimoto	fragprints	$\mathbf{0.019 \pm 0.003}$	0.023 ± 0.002	0.023 ± 0.002	0.006 ± 0.002
	fingerprints	0.023 ± 0.003	0.022 ± 0.002	0.018 ± 0.003	0.006 ± 0.001
	fragments	0.025 ± 0.005	0.012 ± 0.002	0.014 ± 0.002	0.009 ± 0.002
Scalar Product	fragprints	0.033 ± 0.006	0.010 ± 0.002	0.017 ± 0.003	0.010 ± 0.001
	fingerprints	0.036 ± 0.006	0.014 ± 0.002	0.016 ± 0.002	0.009 ± 0.001
	fragments	0.027 ± 0.004	0.012 ± 0.003	0.021 ± 0.003	0.010 ± 0.001
String	SELFIES	0.031 ± 0.006	-	-	-
	SMILES	0.024 ± 0.003	0.016 ± 0.002	0.019 ± 0.003	0.005 ± 0.001
WL Kernel (GraKel)	graph	0.025 ± 0.007	0.011 ± 0.004	0.019 ± 0.009	0.066 ± 0.014