NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World

Graph Neural Bayesian Optimization for Virtual Screening

Miles Wang-Henderson MWANGHENDERS@ETHZ.CH
ETH Zurich

Bartu Soyuer BSOYUER@ETHZ.CH
ETH Zurich

Parnian Kassraie PKASSRAIE@ETHZ.CH
ETH Zurich

Andreas Krause KRAUSEA@ETHZ.CH
ETH Zurich

Ilija Bogunovic I.BOGUNOVIC@UCL.AC.UK
University College London

Abstract

Virtual screening is an essential component of early-stage drug and materials discovery. This is challenged by the increasingly intractable size of virtual libraries and the high cost of evaluating properties. We propose GNN-SS, a Graph Neural Network (GNN) powered Bayesian Optimization (BO) algorithm. GNN-SS utilizes random sub-sampling to reduce the computational complexity of the BO problem, and diversifies queries for training the model. We further introduce data-independent projections to efficiently model second-order random feature interactions, and improve uncertainty estimates. GNN-SS is computationally light, sample-efficient, and rapidly narrows the search space by leveraging the generalization ability of GNNs. Our algorithm achieves state-of-the-art performance among screening methods on the Practical Molecular Optimization benchmark.

Keywords: Virtual Screening, Bayesian Optimization, Graph Neural Networks

1. Introduction

Molecular optimization is a key problem in early drug discovery and materials design, wherein the objective is to retrieve candidate compounds with one or more desirable properties. Some relevant properties of interest include target binding affinities and quantum-mechanical energies. However, these tasks are often complicated by 1) the combinatorially large size of accessible chemical space, and 2) the prohibitive cost of evaluating objectives *in silico* or via experiment. Thus, a first-line solution is to perform molecular optimization on finite but *large* virtual libraries, where candidates can be readily synthesized and validated.As libraries grow to cardinalities exceeding 10^9, brute-force enumeration for candidate selection becomes increasingly infeasible. To address this challenge, one promising strategy is to perform Bayesian Optimization, a model-based approach that leverages a surrogate function to evaluate molecular properties. In particular, by modeling molecules as graphs, we can use BO methods which exploit the graph structure and utilize Graph Neural Networks (GNNs) to capture the complex molecular properties and improve the sample efficiency of our procedure. However, deployment of GNNs for BO on graphs is a computationally challenging problem.

Specifically, a significant issue arises when estimating uncertainty in such models, as conventional neural approaches necessitate the computation and inversion of a corresponding covariance matrix that scales with the number of parameters (Zhou et al., 2020), leading to an inherent trade-off between model expressiveness and computational feasibility. In addition, while performing BO with a GNN-based acquisition function improves the statistical complexity of the optimization problem, applying BO to vast discrete libraries is also computationally prohibitive.

We introduce GNN-SS to address these issues, a light GNN-powered BO algorithm that randomly sub-samples the domain before optimizing the acquisition function. This reduces the computational cost and increase the diversity of collected data, which in turn, enhances exploration to improve the generalization ability of the network. Furthermore, to avoid prohibitive matrix inversions required for uncertainty estimates we employ Johnson-Lindenstrauss (JL) random-feature mappings. In our setting, this mapping can be interpreted as sub-sampling the weights of a trained neural network, allowing tractable matrix inversions.

We evaluate our algorithm with a limited sampling budget on a domain of commercially available small-molecules. We observe that when paired with the Upper Confidence Bound (UCB) acquisition function, our algorithm GNN-SS achieves state-of-the-art performance on the Practical Molecular Optimization (PMO) benchmark among algorithms for virtual screening (Gao et al., 2022), even outperforming a number of generative *de novo* algorithms for molecular design.

Related Works. Early work on BO for molecular optimization use hand-designed graph kernels (Korovina et al., 2020), or map graph representations to a continuous latent space (Gómez-Bombarelli et al., 2018), to then solve the problem with Gaussian process BO methods, e.g. GP-UCB. Recent work utilize GNNs, as they exploit the structure of objective functions defined on graphs. Such approaches require quantifying the uncertainty of neural network estimates, which may be done by parameterizing the uncertainty itself with a neural network (Graff et al., 2021; Soleimany et al., 2021), probabilistic ensembles (Kim et al., 2021; Hirschfeld et al., 2020), or directly upper bounding the uncertainty in the Neural Tangent Kernel (NTK) regime (Kassraie et al., 2022). These methods require calculating the acquisition function over the entire domain, and cannot be scaled to large molecular libraries. Building upon Kassraie et al. (2022), we propose a scalable algorithm which exhibits competitive performance, while reducing the computational complexity of prior work. To this end, we leverage the simple idea of random sub-sampling, which is grounded in the literature of unstructured many-armed bandits (Mirzasoleiman et al., 2015; Bayati et al., 2020).

2. Problem Setting

We consider optimization problems on domains of small molecules that emerge in drug or material design. Small-molecules admit a natural representation as undirected graphs with at most N nodes (Brown et al., 2019; Du et al., 2022). This allows us to formulate such optimization problems as Bayesian optimization on a domain of graphs, where the aim is to optimize an *unknown* objective function through sequential queries while receiving noisy function evaluations. The BO objective function models the unknown molecular property of interest, and querying a graph corresponds to recommending a molecule to be tested with respect to this property.

At every time step $t \in \{1, \dots, T\}$, we select a graph G_t from a graph domain $\mathcal{G}$ and observe a noisy evaluation $y_t = f(G_t) + \epsilon_t$, where $f : \mathcal{G} \to \mathbb{R}$ is the objective function and ϵ_t is i.i.d. zero-mean sub-Gaussian noise. We denote the history of observations by $H_t \coloneqq \{(G_1, y_1), \dots, (G_t, y_t)\}$. For a horizon T (i.e., sampling budget), we seek to obtain a small *simple* regret $r_T = f(G^*) - \max_{t \le T} f(G_t)$, where $G^* \in \arg\max_{G \in \mathcal{G}} f(G)$. The aim is to attain a regret that *vanishes* with T, meaning that $r_T \to 0$ as $T \to \infty$, which implies convergence to the optimal graph. Evaluating molecular properties typically requires running costly simulations or experiments with noisy outcomes. Therefore, we seek a sample efficient algorithm that can identify the optimal molecule with the least number of oracle calls.

In our applications of interest the number of accessible molecules ranges between 10^{10}-10^{20} choices (Nicolaou et al., 2016), and popular virtual databases of commercially available compounds include up to billions of molecules (e.g. ZINC, Irwin and Shoichet, 2005). Therefore, we assume that the domain $\mathcal{G}$ is a very large finite dataset of graphs, where $|\mathcal{G}| \gg T$, meaning that it is infeasible to evaluate every graph in the domain within the sampling budget of T. Standard BO algorithms for large unstructured domains (e.g., Auer et al., 2002) cannot be applied to this problem setting, as their sample complexity grows with poly($|\mathcal{G}|$).

3. Method

```
Algorithm 1 GNN-SS
  Input: acq. function $\alpha(\cdot)$, horizon $T$, warm-start steps $T_0$, random subset size $K$, batch
  size $b \le K$, training epochs $E$, learning rate $\eta$, regularization parameter $\lambda$
  Initialize: $\boldsymbol{K}_0 \leftarrow \lambda \boldsymbol{I}$, $\boldsymbol{\theta}_0 \sim \mathcal{N}(0, \boldsymbol{I}_p)$, $H_0 = \emptyset$.

  for round $t = 1, \dots, T_0$ do                                              ▷ Exploration Stage
      a) Sample random action. $G_t \sim \mathrm{Unif}(\mathcal{G})$ and query $y_t = f(G_t) + \epsilon_t$
      b) Add to history. $H_t \leftarrow H_{t-1} \cup \{(G_t, y_t)\}$
  end for
  $\boldsymbol{\theta}_{T_0} \leftarrow \textsc{TrainGNN}(H_{T_0}, \boldsymbol{\theta}_0, E, \lambda, \eta)$
  $\boldsymbol{K}_{T_0} \leftarrow \boldsymbol{K}_0 + \sum_{t=1}^{T_0} \sum_{i=1}^{b} \boldsymbol{g}(G_{t,i}) \boldsymbol{g}^\top (G_{t,i})$

  for round $t = T_0 + 1, \dots, T$ do                                          ▷ Adaptive Stage
      1) Sub-sample. $\mathcal{G}_t \sim \mathrm{Unif}(\mathcal{G})$ where $|\mathcal{G}_t| = K$
      2) Evaluate acquisition. $\{G_{t,i}\}_{i=1}^{b} \leftarrow \arg\max_{G \in \mathcal{G}_t} \alpha(G; H_{t-1})$
      2) Observe noisy reward. $y_{t,i} = f(G_{t,i}) + \epsilon_{t,i}$ for $i \in [b]$
      3) Append to history. $H_t \leftarrow H_{t-1} \cup \{(G_{t,1}, y_{t,1}), \dots, (G_{t,b}, y_{t,b})\}$
      4) Update surrogate parameters. $\boldsymbol{\theta}_t \leftarrow \textsc{TrainGNN}(H_t, \boldsymbol{\theta}_{t-1}, E, \lambda, \eta)$
      4) Update uncertainty. $\boldsymbol{K}_t \leftarrow \boldsymbol{K}_{t-1} + \sum_{i=1}^{b} \boldsymbol{g}(G_i) \boldsymbol{g}^\top (G_i)$
  end for
```

We now summarize our approach and defer full details of GNN-SS to Appendix A. To obtain a sample efficient algorithm for maximizing the objective function, we need to construct an estimator for f, using a relatively small number of samples from $\mathcal{G}$. To this end, we utilize three key ideas. First, we use GNNs, which are known to be effective models for learning complex functions defined on graphs. Second, in order to efficiently update our uncertainty over the network's outputs, defined in Equation (1), we use a random-feature approximation to the NTK associated to the specific architecture used. Employing rank-one updates to this uncertainty estimator requires $O(tp^2)$ time, where p is the dimension of our random features and corresponds to the number of network parameters. While previous works (e.g., Zhou et al., 2020; Kassraie et al., 2022) address this by employing a diagonal approximation to the covariance matrix, we aim to model second-order covariance structure by utilizing a low-rank approximation to the empirical NTK, defined in Equation (2). As demonstrated in Figure 1, this helps us draw samples which are informative and yield high reward in only $\mathcal{O}(td^2)$ time, where $d \ll p$. We use two approximations to our uncertainty estimator, defined in Appendix B. Lastly, we devise a sub-sampling (SS) subroutine that allows us to obtain diverse data for training the network. These techniques come together in our GNN-powered Sub-sampled BO algorithm (GNN-SS). The sub-routine TrainGNN is specified in Algorithm 2.

4. Results

We perform experiments on Zinc (Irwin and Shoichet, 2005), a dataset of 250k small organic molecules that is commonly used for benchmarking algorithms for molecular optimization (e.g., in Gao et al., 2022; Kim et al., 2022; Graff et al., 2021). Our experiments demonstrate the application of GNN-SS for virtual screening on these datasets, based on molecular properties which are of interest in drug and materials design. In these experiments, we consider all 23 molecular objectives of the PMO benchmark (Gao et al., 2022), which include commonly used scores such as quantitative estimate of drug-likeness (QED) (Bickerton et al., 2012). To model these objectives, we employ a vanilla Graph Convolutional Network (GCN) similar to Graff et al. (2021) and for training use different sets of hyper-parameters (details in Appendix C). Reported results are averaged across 5 runs with different random seeds, plotted together with the standard error. Figure 1 shows the performance of GNN-SS paired with UCB, and Greedy policies on 4 objective functions. Regret plots for additional reward functions are given in Appendix D.

We limit the horizon to a realistic sampling budget, and for every t, we compute the top-1 reward, i.e. $\max_{s \le t} f(G_s)$ the value of the highest scoring molecule queried so far, as well as the top-10 mean reward. As a sanity check, we also run SS-Rand which simply draws a random graph from $\mathcal{G}_t$, and SS-Oracle which has oracle knowledge of f and queries $\arg\max_{G \in \mathcal{G}_t} f(G)$ after random sub-sampling. A sub-sampled policy cannot achieve a performance higher than SS-Oracle, and should perform better than SS-Rand.

To compare our algorithm to prior work we use the PMO benchmark, in which the total number of queried molecules, i.e. the number of oracle calls, is restricted to a budget of $T = 10,000$. This benchmark considers 29 algorithms for virtual screening, and evaluates the performance of each by area-under-the-optimization-curve (AUC, as plotted in Figure 1), summed across the 23 objective functions. As reported in Table 3, GNN-SS achieves state-of-the-art performance among non-generative models, including MolPAL, ranking 4th overall in sample efficiency among the 29 algorithms considered in this benchmark. Importantly, GNN-SS outranks several generative algorithms which go beyond Zinc for maximizing the properties of interest, e.g. SMILES-LSTM Brown et al. (2019) and MARS Xie et al. (2021).

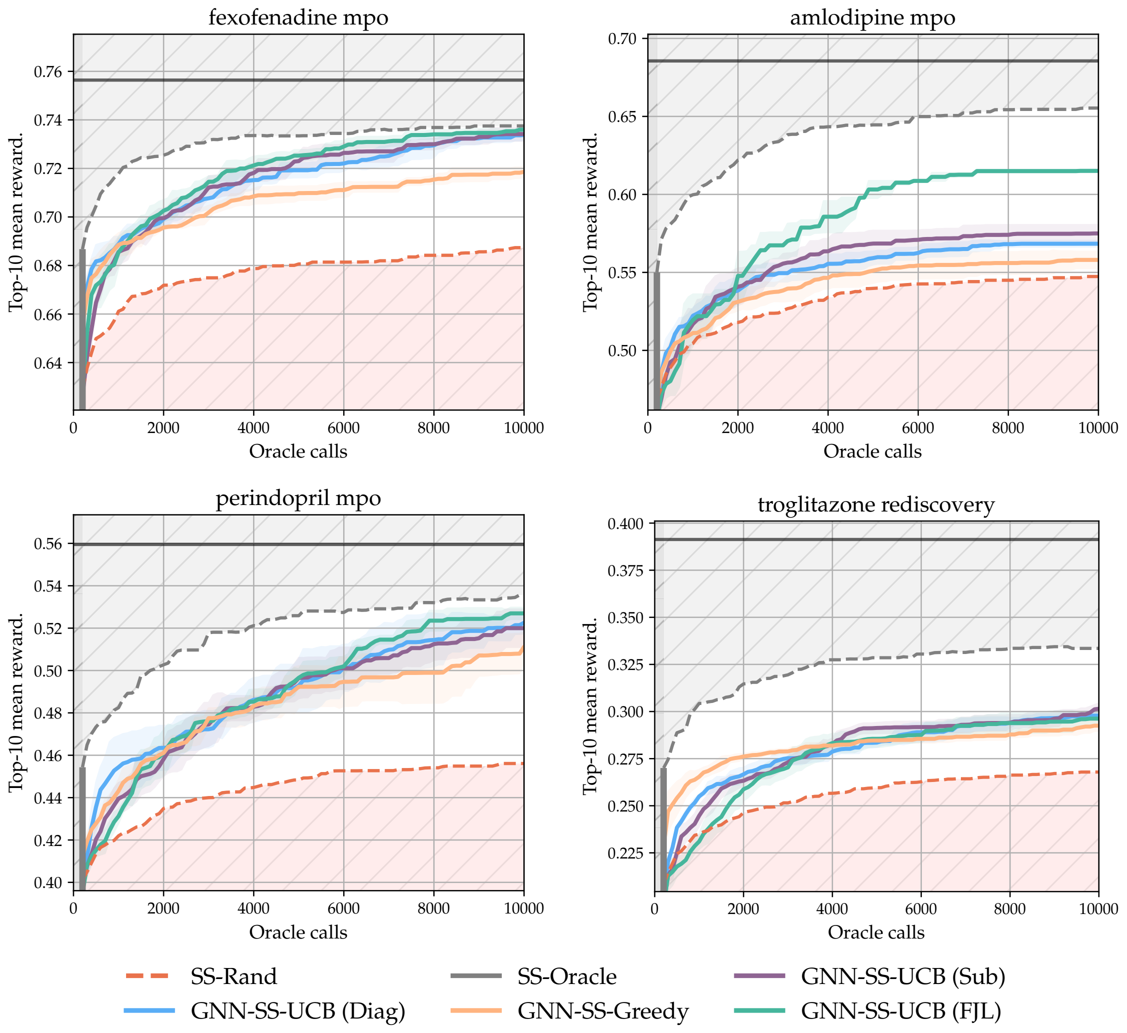


Figure 1: GNN-SS Top-10 Mean Reward. Projection dimension is $d = 2048$ for both Subsampling (Sub) and Fast Johnson-Lindenstrauss (FJL variants for uncertainty estimator $\tilde{\sigma}_t(\cdot)$, see Appendix A. Bold horizontal line shows ZINC-250K maximum and the gray region above GNN-SS-Oracle is unreachable by sub-sampled algorithms.

As observed in Figure 1 and Table 1, Sub and FJL variants of our algorithm obtain higher reward molecules overall, ranking 8th and 7th respectively, and first among screening methods. We conjecture that enacting GNN-SS with an uncertainty estimator which models second-order covariance structure allows us to retrieve diverse molecules early during optimization. At the cost of sample efficiency, this improves our surrogate reward estimates, ultimately leading GNN-SS to obtain higher reward molecules.

The importance of our random sub-sampling routine in GNN-SS is justified by an experiment in which the acquisition function is evaluated for every member of the domain $\mathcal{G}$. As shown in Figure 2, for certain objectives GNN-SS-UCB achieves improved sample efficiency compared to an expensive variant without sub-sampling.

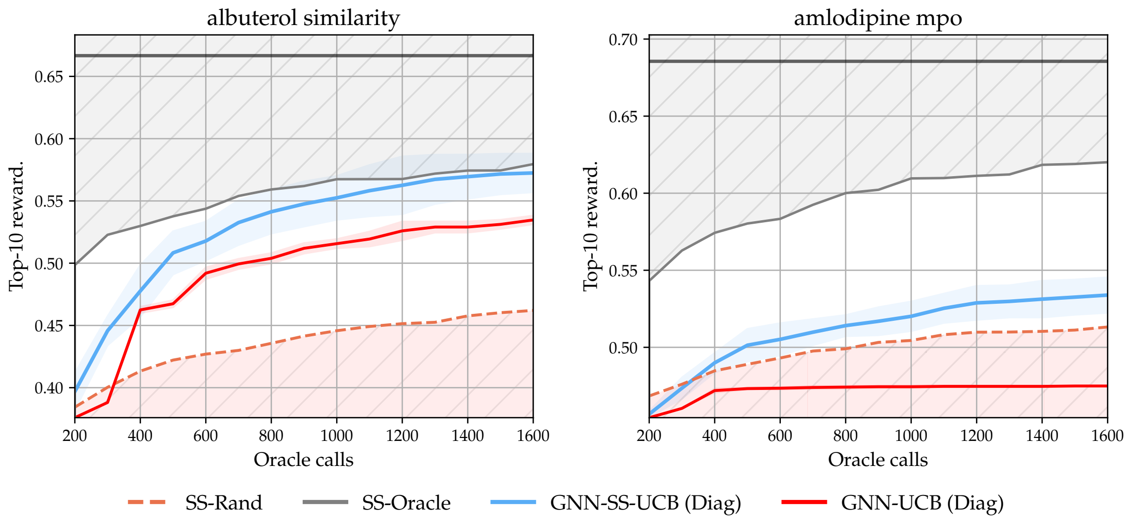


Figure 2: GNN-SS-UCB and GNN-UCB Top-10 Mean Reward. Random subsampling in GNN-SS encourages acquisition of diverse molecules early on during optimization. GNN-UCB evaluates the acquisition function $\alpha(\cdot\,; \boldsymbol{\theta}_t)$ on the full ZINC-250K dataset.

Molecular Objective	MolPAL	GNN-SS Greedy	GNN-SS UCB (Diag)	GNN-SS UCB (Sub)	GNN-SS UCB (FJL)
Top-1 sum	12.84	13.99	14.19	14.27	**14.38**
$n/29$ rank	17	11	10	8	**7**

Table 1: Summary PMO benchmark Top-1 Reward with different uncertainty estimators. Top-1 Sum aggregates all 23 objectives. Full table in Appendix D.

5. Conclusion

We presented GNN-SS, a Graph Neural Network BO algorithm that draws on sub-sampling and random feature projections, to present a scalable solution to the problem of screening large virtual libraries. We demonstrated that GNN-SS achieves competitive performance on the PMO benchmark for sample-efficient molecular optimization. Our results show how GNN-powered Bayesian Optimization may be deployed with datasets of 10^6 molecules. However it remains an open problem to scale BO to virtual databases with billions of molecules, or efficiently go beyond a library and perform *de novo* design.

References

Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

Sanjeev Arora, S. Du, Wei Hu, Zhiyuan Li, R. Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Neural Information Processing Systems*, 2019.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 2002.

Hamsa Bastani and Mohsen Bayati. Online decision making with high-dimensional covariates. *Operations Research*, 2020.

Mohsen Bayati, Nima Hamidi, Ramesh Johari, and Khashayar Khosravi. Unreasonable effectiveness of greedy algorithms in multi-armed bandit with many arms. *Advances in Neural Information Processing Systems*, 2020.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 2012.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

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

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 2022.

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.

David E. Graff, Eugene I. Shakhnovich, and Connor W. Coley. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chem. Sci.*, 2021.

Botao Hao, Tor Lattimore, and Mengdi Wang. High-dimensional sparse linear bandits. *Advances in Neural Information Processing Systems*, 2020.

Lior Hirschfeld, Kyle Swanson, Kevin Yang, Regina Barzilay, and Connor W. Coley. Uncertainty quantification using neural networks for molecular property prediction, 2020.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf H. Roohani, J. Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and M. Zitnik. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Neurips Datasets And Benchmarks*, 2021.

John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 2005.

William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

Ata Kaban. Improved bounds on the dot product under random projection and random sign projection. *KDD*, 2015.

Parnian Kassraie, Andreas Krause, and Ilija Bogunovic. Graph neural network bandits. In *Advances in Neural Information Processing Systems*, 2022.

Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljacic. Deep learning for bayesian optimization of scientific problems with high-dimensional structure. *Transactions of Machine Learning Research*, 2021.

Samuel Kim, Peter Y. Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Deep learning for bayesian optimization of scientific problems with high-dimensional structure, 2022.

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.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

Christos A Nicolaou, Ian A Watson, Hong Hu, and Jibo Wang. The proximal lilly collection: Mapping, exploring and exploiting feasible chemical space. *Journal of chemical information and modeling*, 2016.

Ava P. Soleimany, Alexander Amini, Samuel Goldman, Daniela Rus, Sangeeta N. Bhatia, and Connor W. Coley. Evidential deep learning for guided molecular property prediction and discovery. *ACS Central Science*, 2021.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

T.T. Tanimoto. *An Elementary Mathematical Theory of Classification and Prediction*. International Business Machines Corporation, 1958. URL https://books.google.ch/books?id=yp34HAAACAAJ.

Kilian Q. Weinberger, A. Dasgupta, Josh Attenberg, J. Langford, and Alex Smola. Feature hashing for large scale multitask learning. *International Conference on Machine Learning*, 2009.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. {MARS}: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint*, 2020.

Yun Yang, Mert Pilanci, and Martin J. Wainwright. Randomized sketches for kernels: Fast and optimal non-parametric regression. *arXiv preprint arXiv: 1501.06195*, 2015.

Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. *International Conference On Learning Representations*, 2020.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

Appendix A. Algorithm

Reward Model. To model molecular properties, we use a graph neural network $f_{\text{GNN}}(G; \boldsymbol{\theta}) : \mathcal{G} \to \mathbb{R}$ of width m, where $\boldsymbol{\theta} \in \mathbb{R}^p$ denotes the concatenated vector of network parameters. We initialize the network with $\boldsymbol{\theta}_0$ which has zero-mean Gaussian i.i.d. entries of unit variance, and update it as we observe more feedback. As a proxy for f, we maintain $f_{\text{GNN}}(G; \boldsymbol{\theta}_t)$, the GNN trained on the following loss with regularization coefficient λ,

$$\mathcal{L}(\boldsymbol{\theta}; H_t) = \frac{1}{t} \sum_{i=1}^{t} \left(f_{\text{GNN}}(G_i, \boldsymbol{\theta}) - y_i\right)_2^2 + m\lambda \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2$$

Similar to Kim et al. (2022) and Zhang et al. (2020), we initialize the network at every step of the BO problem with $\boldsymbol{\theta}_{t-1}$, and update the parameters via gradient descent for E epochs. This is in contrast to earlier work on Neural Bandits which re-initialize the network to $\boldsymbol{\theta}_0$ before running gradient descent (e.g. Zhou et al. (2020); Kassraie et al. (2022)).

Uncertainty Quantification. For action-selection policies which require uncertainty quantification, e.g. UCB (Srinivas et al., 2010), we may use the gradient of the network at initialization to approximate the uncertainty over f. Following Kassraie et al. (2022), we set

$$\hat{\sigma}_t^2(G) \coloneqq \boldsymbol{g}^\top(G) \Big[\lambda \boldsymbol{I} + \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{g}(G_i)\boldsymbol{g}^\top(G_i)\Big]^{-1} \boldsymbol{g}(G), \tag{1}$$

where $\boldsymbol{g}(G) = \nabla_{\boldsymbol{\theta}} f_{\text{GNN}}(G; \boldsymbol{\theta}_0)$ denotes the gradient at initialization. For very wide networks, $\hat{\sigma}_t$ presents a provably calibrated uncertainty estimate (Kassraie et al., 2022, Theorem 4.2). In this formulation, the Gram matrix with elements $\hat{K}_{ij} = \boldsymbol{g}^\top(G_i)\boldsymbol{g}(G_j)$ performs a similar role as the kernel matrix when calculating the posterior variance in given a Gaussian Process. For this reason , among others, the kernel function $\hat{k}(G, G') = \boldsymbol{g}^\top(G)\boldsymbol{g}(G')$ is referred to as the empirical Neural Tangent Kernel. Figure 3 demonstrates that the matrix $\hat{K}$, effectively encodes the structural similarity between molecules with similar rewards. In this figure, the left-most matrix has elements $1 - |y(G_i) - y(G_j)|$, where $y(G_i)$ is the reward of molecule G_i. The reward is the Amlodipine MPO objective, which is a function of the number of aromatic rings as well as the ECP4 fingerprint similarity to the Amlodipine molecule (Brown et al., 2019). (Tanimoto, 1958). Additionally, we compute a kernel matrix corresponding to the Tanimoto kernel, a commonly used similarity metric for sets. When applied to molecules, this measures the intersection of shared substructures between G and G', and is computed as $k_{\text{Tan}}(G, G') = \sum_i \min(\boldsymbol{h}(G)_i, \boldsymbol{h}(G')_i) / \sum_i \max(\boldsymbol{h}(G)_i, \boldsymbol{h}(G')_i)$, where $\boldsymbol{h}(G) \in \{0, 1\}^{|V|}$ is a vector indicating the presence or absence of substructures in G from a finite vocabulary V.

Numerical Approximation. Computing the uncertainty estimates as laid out in (1), requires $p \times p$ matrix inversions, where p denotes the number of trainable parameters in the GNN. This operation is the memory and computation bottleneck of the algorithm, and quickly becomes intractable for even modest-sized networks. A common approach in neural bandit algorithms is to use a diagonal approximation of the matrix $\boldsymbol{g}(G_i)\boldsymbol{g}^\top(G_i)$, leading to updates linear in the number of parameters $\mathcal{O}(p)$ (Zhou et al., 2020). Asymptotically, for large enough

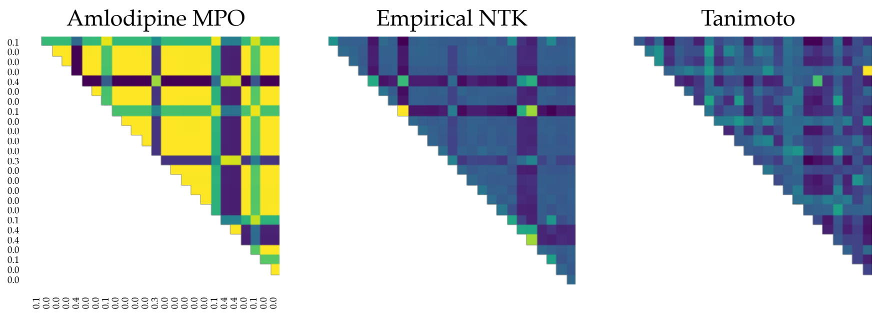


Figure 3: Empirical NTK ($\hat{K}$) effectively captures shared structure between molecules of similar reward, and is more informative than the Tanimoto kernel (K_{Tan}).

width m, this is a valid mean-field approximation as the parameters become independent (Yang, 2020). However, for finite neural networks of practical size, this assumption becomes vacuous. Instead, we apply a data-independent random projection $S : \mathbb{R}^p \to \mathbb{R}^d$, with $d \ll p$, to reduce the dimensionality of this matrix. We choose S to be a Johnson-Lindenstrauss (JL) linear map (Johnson and Lindenstrauss, 1984), since such maps can be designed to satisfy

$$\left|\langle S\boldsymbol{g}(G), S\boldsymbol{g}(G')\rangle - \langle \boldsymbol{g}(G), \boldsymbol{g}(G')\rangle\right| \leq \epsilon_D,$$

if $d \geq 4\epsilon_D^{-2}\log(1/\delta)$, where ϵ_D is the distortion error and $\delta \in (0,1)$ is the confidence level (Johnson and Lindenstrauss, 1984). Therefore, by specifying a small projected dimension d, we can control the distortion error that propagates into the uncertainty quantification formula. Setting $\tilde{\boldsymbol{g}}(\cdot) = S\boldsymbol{g}(\cdot)$ we then define the *cheap* uncertainty estimates as

$$\tilde{\sigma}_t^2(G) := \tilde{\boldsymbol{g}}^\top(G)\Big[\lambda \boldsymbol{I} + \frac{1}{t}\sum_{i=1}^{t} \tilde{\boldsymbol{g}}(G_i)\tilde{\boldsymbol{g}}^\top(G_i)\Big]^{-1}\tilde{\boldsymbol{g}}(G). \tag{2}$$

Calculating $\tilde{\sigma}_t$ now only requires inverting a $d \times d$ matrix. Proposition 3 in Appendix B proves that the approximated empirical NTK defined as $\tilde{k}(G, G') = \tilde{\boldsymbol{g}}^\top(G)\tilde{\boldsymbol{g}}(G')$, is with high probability close to $k(G, G')$ the empirical NTK for every pair of graphs $(G, G') \in \mathcal{G} \times \mathcal{G}$. We also empirically observe that the numerical error introduced due to using $\tilde{\boldsymbol{g}}$ instead of $\boldsymbol{g}$ is negligible, for practical values of d. This is expected, due to the closeness of the two corresponding kernels. Details on S are in Appendix B.

Speeding up Approximations. The canonical construction of S is to use a dense matrix of i.i.d. Gaussian entries, requiring $\mathcal{O}(pd)$ time to obtain $\tilde{\boldsymbol{g}}(G)$ (Johnson and Lindenstrauss, 1984). Instead, we employ two computationally-efficient variants. The first is a Fast JL (FJL) variant requiring $\mathcal{O}(p \log p)$ time to obtain $\tilde{\boldsymbol{g}}(G)$ (Ailon and Chazelle, 2009),

$$S_{\mathrm{FJL}} = PHD$$

Here, $P \in \mathbb{R}^{d\times p}$ is a sparse matrix with elements $P_{ij} \in \{0, \sqrt{p}/\sqrt{d}\}$ and $\|P_i\| = \sqrt{p}/\sqrt{d}$. $H \in \mathbb{R}^{p\times p}$ is a Walsh-Hadamard matrix with $H_{ij} \in \{-1, 1\}$, and the diagonal matrix $D \in \mathbb{R}^{p\times p}$ has Rademacher entries $D_{ii} \in \{-1, 1\}$. Our second Subsampling variant (Sub) sets

$$S_{\text{Sub}} = P.$$

This variant is inspired by (Weinberger et al., 2009; Yang et al., 2015) and requires only a time of $\mathcal{O}(d)$. Table 2 summarizes runtime of each of our uncertainty estimators when rank-one updates are employed.

Estimator	Runtime
Full	$\mathcal{O}(Np^2)$
Diagonal	$\mathcal{O}(Np)$
Sub	$\mathcal{O}(Nd^2 + Nd)$
FJL	$\mathcal{O}(Nd^2 + Np \log p)$

Table 2: Uncertainty estimates via approximate random-features $\tilde{\boldsymbol{g}}$, yielding covariance matrices with second-order structure. $N = tb$, where t is the number of rounds, b is batch-size. JL variants project $\boldsymbol{g} \in \mathbb{R}^p$ to $\tilde{\boldsymbol{g}} \in \mathbb{R}^d$.

Random Sub-sampling. The common approach to action selection is to choose G_t via $\max_{G\in\mathcal{G}} \alpha(G; H_t)$. However, this will require evaluating α for all members of $\mathcal{G}$, which comes with a high computational burden when working on large virtual libraries. We utilize a sub-sampling routine where at every step t we first draw a random subset $\mathcal{G}_t \subset \mathcal{G}$, where $|\mathcal{G}_t| \ll |\mathcal{G}|$. We then rank the members of $\mathcal{G}_t$, with respect to $\alpha(\cdot; H_t)$, and query the top-b graphs.

The advantage of random sub-sampling is threefold. It saves computations at each round, by allowing us to evaluate $\alpha(\cdot; H_t)$ only over $K = |\mathcal{G}_t|$ molecules. Sub-sampling limits the number of graphs available for query, including more promising molecules which drive GNN-SS to exploit. This implicitly encourages exploration, which is believed to be beneficial on large finite domains with complex objectives (Bastani and Bayati, 2020). This additional exploration helps build a diverse dataset H_t for training f_{GNN} in the future steps, and in turn, yields an accurate reward estimator $f_{\text{GNN}}(G; \boldsymbol{\theta}_{t-1})$ that generalizes well over the entire domain. The choice of K is crucial to balancing exploration and exploitation. Moreover, it should be chosen such that the probability of not ever sampling the optimal molecule is small.

Exploration Stage. We start the algorithm with an exploration stage, during which we randomly query a small number ($\sim 10^2$) of molecules. This improves the network's ability to generalize over all regions in the domain, without necessarily having sampled from them. The idea of an exploratory stage stems from the high-dimensional bandit literature (e.g., Hao et al., 2020; Bastani and Bayati, 2020), where additional exploration is required to obtain an accurate estimate of the reward.

Batching. Choosing $b > 1$ yields a simple batched variant of GNN-SS. At every step, after sub-sampling, molecules are scored according to our acquisition function $\alpha(\cdot\,; H_t)$ and the top-b graphs are queried as a batch, i.e. $\{G_{t,i}\}_{i=1}^{b} \leftarrow \arg\max_{G\in\mathcal{G}_t} \alpha(G; H_t)$. Batching is primarily done to avoid the cost of re-training the model for every new query.

Appendix B. Closeness of Random Feature Approximations

As detailed in Kassraie et al. (2022), since a large finite-width neural network behaves approximately as a linear function in feature space, it can be used in place of the posterior mean of a GP with an NTK associated to the same architecture as its covariance. Training of the former yields an approximate solution to kernel ridge-regression with the NTK, while the latter yields a solution that is exact. Likewise, one may approximate the posterior covariance with explicit features $\boldsymbol{g}(G) = \nabla_\theta f(G;\theta)$. We now restate a central lemma that formalizes the error's dependence on hidden-width m for given depth L.

Lemma 1 (Finite-width Approximation to the NTK (Arora et al., 2019)(Thm. 3.1))**.** *Let node features $\boldsymbol{x}_G, \boldsymbol{x}'_G \in \mathbb{S}^{d-1}$ be restricted to the hyper-sphere, and consider an appropriately initialized neural network of depth L with `ReLU` layer-activations of* $\max(0, \boldsymbol{x})$*. Fix error $\epsilon_A > 0$, $\delta_A \in (0,1)$. Then for minimum hidden width $m_{\min} \in \Omega(L^6 \epsilon_A^{-4} \log(L/\delta))$, the following bounds the error of the empirical NTK at initialization, and holds with probability at least $1 - \delta_A$*

$$\left|\langle \boldsymbol{g}(G), \boldsymbol{g}(G')\rangle - k(G, G')\right| \leq (L+1)\epsilon_A.$$

When m is large, this allows us to employ the following as a provably calibrated uncertainty estimate,

$$\hat{\sigma}_t^2(G) = \boldsymbol{g}^\top(G)\left(\lambda I + \sum_{i=1}^{t-1} \boldsymbol{g}(G_i)\boldsymbol{g}(G_i)^\top\right)^{-1} \boldsymbol{g}(G) = \boldsymbol{g}^\top(G)\left(\lambda I + \boldsymbol{J}_t^\top \boldsymbol{J}_t\right)^{-1} \boldsymbol{g}(G)$$

Here, $\boldsymbol{J}_t = [\nabla_\theta f(G_i;\theta)]_{i<t}$ is the Jacobian with $t-1$ rows, and each row contains the gradient vector $\boldsymbol{g}(G_i)$. Notice that this admits rank-one updates via the matrix inversion lemma. However, the benefit is not immediately clear as the outer-product $\boldsymbol{g}(G)\boldsymbol{g}(G)^\top$ is prohibitive for even shallow, small-width networks, requiring $\mathcal{O}(m^2)$ memory and $\mathcal{O}(m^6)$ effort to invert. Practical implementations of neural bandit algorithms that use this construction have therefore addressed expensive inversions by taking a diagonal approximation of the matrix $\boldsymbol{J}^\top \boldsymbol{J}$, leading to updates linear in the number of parameters $\mathcal{O}(p)$. Instead, we will propose an alternative approximation that yields valid confidence sets for optimistic bandit policies.

We now outline Johnson-Lindenstrauss (JL) type projections (Johnson and Lindenstrauss, 1984). These projections are *data-independent* and are generated under a random linear map $S : \mathbb{R}^p \to \mathbb{R}^d$, where $d < p$. Importantly, the distortion of these projections can be controlled for some choice of rank d, which must be greater than $d_{\min}$ for some distortion error ϵ_D and confidence δ_D. The following celebrated lemma bounds the distortion using such a projection.

Lemma 2 (JLL (Johnson and Lindenstrauss, 1984))**.** *Set $\epsilon_D \in (0, 1/2)$ and let $d_{\min} \geq 4\epsilon_D^{-2} \log(1/\delta)$. There exists a matrix $S \in \mathbb{R}^{d \times p}$ that w.h.p. preserves Euclidean norms, i.e. simultaneously for all $\boldsymbol{x} \in \mathbb{R}^p$, S satisfies the following with probability greater than $1 - \delta_D$*

$$\|S\boldsymbol{x}\|_2 \in \left[(1-\epsilon_D)\|\boldsymbol{x}\|_2, (1+\epsilon_D)\|\boldsymbol{x}\|_2\right]$$

A corollary of Lemma 2 can also be obtained for squared-norms via the parallelogram law, where the original lemma is applied to norms found in the following,

$$\langle S\boldsymbol{x}, S\boldsymbol{x}'\rangle = \frac{1}{4}(\|S(\boldsymbol{x}-\boldsymbol{x}')\| + \|S(\boldsymbol{x}+\boldsymbol{x}')\|)$$

For any given $d > d_{\min}$, this has been shown to preserve the original bound up to the same constant factor (Kaban, 2015)(Thm 2.1). And in the setting where we directly use the network gradients $\boldsymbol{g}(G) \in \mathbb{R}^p$ to approximate the NTK, we can make use of this to guarantee that for large enough d, the following bound on the distortion error ϵ_D also holds with high probability,

$$\left|\langle S\boldsymbol{g}(G), S\boldsymbol{g}(G')\rangle - \langle \boldsymbol{g}(G), \boldsymbol{g}(G')\rangle\right| \leq \epsilon_D$$

Now, by the union of events in Lemma 1 and Lemma 2, the following inequality will control the combined distortion ϵ_D and approximation error ϵ_A of the empirical NTK when obtained by a rank-d projection of the p-dimensional random feature map $\boldsymbol{g}(G)$.

Proposition 3 (Approximation Error of Randomly Projected Empirical NTK)**.** *Set $\delta \in (0,1)$ and fix a choice of projection dimension $d \in [\epsilon_A^{-2}\log(1/\delta), p]$. Denote $\alpha = 2\delta - \exp(-d\epsilon_D^2/8)$. Then for minimum width $m_{\min} \in \Omega(L^6\epsilon_A^{-4}\log(L/\alpha))$, the following bound holds with probability at least $1-\delta$,*

$$\left|\langle \tilde{\boldsymbol{g}}(G), \tilde{\boldsymbol{g}}(G')\rangle - k(G,G')\right| \leq \epsilon$$

where $\epsilon \overset{\Delta}{=} (L+1)\epsilon_A + \epsilon_D$.

Where we have used $\tilde{\boldsymbol{g}}(G)$ to denote the random projection of $\boldsymbol{g}(G)$. This holds when the hidden-width m is large enough. Let $\exp(-d\epsilon_D^2/8)$ and $L\exp(-m\epsilon_A^4/L^6)$ be the confidence levels introduced in Lemma 2 and Lemma 1. We require

$$2\delta \leq L\exp(-m\epsilon_A^4/L^6) + 2\exp(-d\epsilon_D^2/8)$$

which is satisfied by choosing $m_{\min} \geq L^6\epsilon_A^{-4}\log(L/(2\delta - \exp(-d\epsilon_D^2/8)))$ for some fixed projection dimension d.

Appendix C. Experiment Details

We use the Zinc dataset provided in `pyTDC`, objectives are evaluated using oracles from the same library Huang et al. (2021). For a fair comparison to MolPAL, we use similar parameters $m = 384$, $L = 3$ graph convolutional layers and no positional encodings, corresponding to GNN-SS (Diag, NoLap) in Table 3. The regularization coefficient is chosen via a grid-search $\lambda \in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$. We use $\lambda = 0.0001$. We select the best performing exploration coefficient β via a grid-search over $\{0.001, 0.01, 0.1, 1.0\}$. The size of randomly sub-sampled set $|\mathcal{G}_t|$ is set as $K = 4000$. Following Gao et al. (2022), we use $T_0 = 500$. For remaining experiments that aim to compare different numerical approximations, we use $k = 4$ absolute graph Laplacian positional encodings (Dwivedi et al., 2023), and choose hidden dimension $m = 512$.

Graph Laplacian positional encodings are eigenvectors of the symmetrically normalized graph Laplacian $\Delta = I - D^{-1/2}AD^{-1/2}$. A and D are the adjacency and degree matrices,

Algorithm 2 TrainGNN

```
Input: history $H_t$, initial parameters $\boldsymbol{\theta}$, training epochs $E$, batchsize $b$, learning rate $\eta$,
regularization $\lambda$

Define: MSE Loss $\mathcal{L}(\boldsymbol{\theta}_i)$ from Appendix A
Initialize: $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}$, $f_{\text{GNN}}(\cdot;\boldsymbol{\theta}^{(0)})$, optimizer Adam($\mathcal{L}$)

for epoch $i = 1, \dots, E$ do
    Batch $\mathcal{B} = \{(y(G_i), G_i)\}^b \sim H_t$
    Update parameters, $\boldsymbol{\theta}^{(i+1)} \leftarrow \text{Adam}(\mathcal{B}, f_{\text{GNN}}(\cdot;\boldsymbol{\theta}^{(i)}))$
end for
```

respectively. From the matrix of eigenvectors $U \in \mathbb{R}^{N \times N}$, we take $k < N$ eigenvectors, which are concatenated directly to node features as $X \leftarrow \text{concat}(X, U_k)$, where U_k is an $N \times k$ matrix. To address the sign ambiguity of the eigenvectors we enforce that each element is non-negative (Dwivedi et al., 2023).

The network parameters are initialized with iid. unit Gaussian entries, and following the procedure of Graff et al. (2021), pre-trained offline for each reward on 500 molecules. We limit training to 250 epochs, with batchsize 250. Early-stopping is used with a patience of 25 iterations without improvement on the validation set. We use `MSELoss` with `lr = 1e-5` and a linear decay towards `1e-6`. $\phi =$ `ReLU` non-linearities are applied between each hidden layer. We modify the `pytorch_geometric` library Fey and Lenssen (2019) to make use of the neural tangent parameterization for all `GCNConv` layers, where $c_\phi = \sqrt{2}$ and m_l is the width of the l-layer weights. Using the neural-tangent parameterization ensures a limiting NTK that converges. Each convolutional layer is defined as

$$f^{(l)}_{\text{GCN}}(X, A) = \sqrt{\frac{2}{m}} \text{ReLU}\left(\Delta X^{(l-1)} W^{(l)}\right)$$

The graph representation is obtained as $\bar{X} = \texttt{cat}(\bar{X}_{\text{mean}}, \bar{X}_{\text{sum}})$ before passing to a final linear layer, i.e. the concatenation of the mean-readout and sum-readout. We note that for the random seed used in our experiments, the subset $H_{\text{off}} \subset \mathcal{G}$ for the VALSARTAN SMARTS objective only contains molecules with zero reward, and thus the model is not expected to generalize to unseen molecules with non-zero reward.

Appendix D. Additional Results

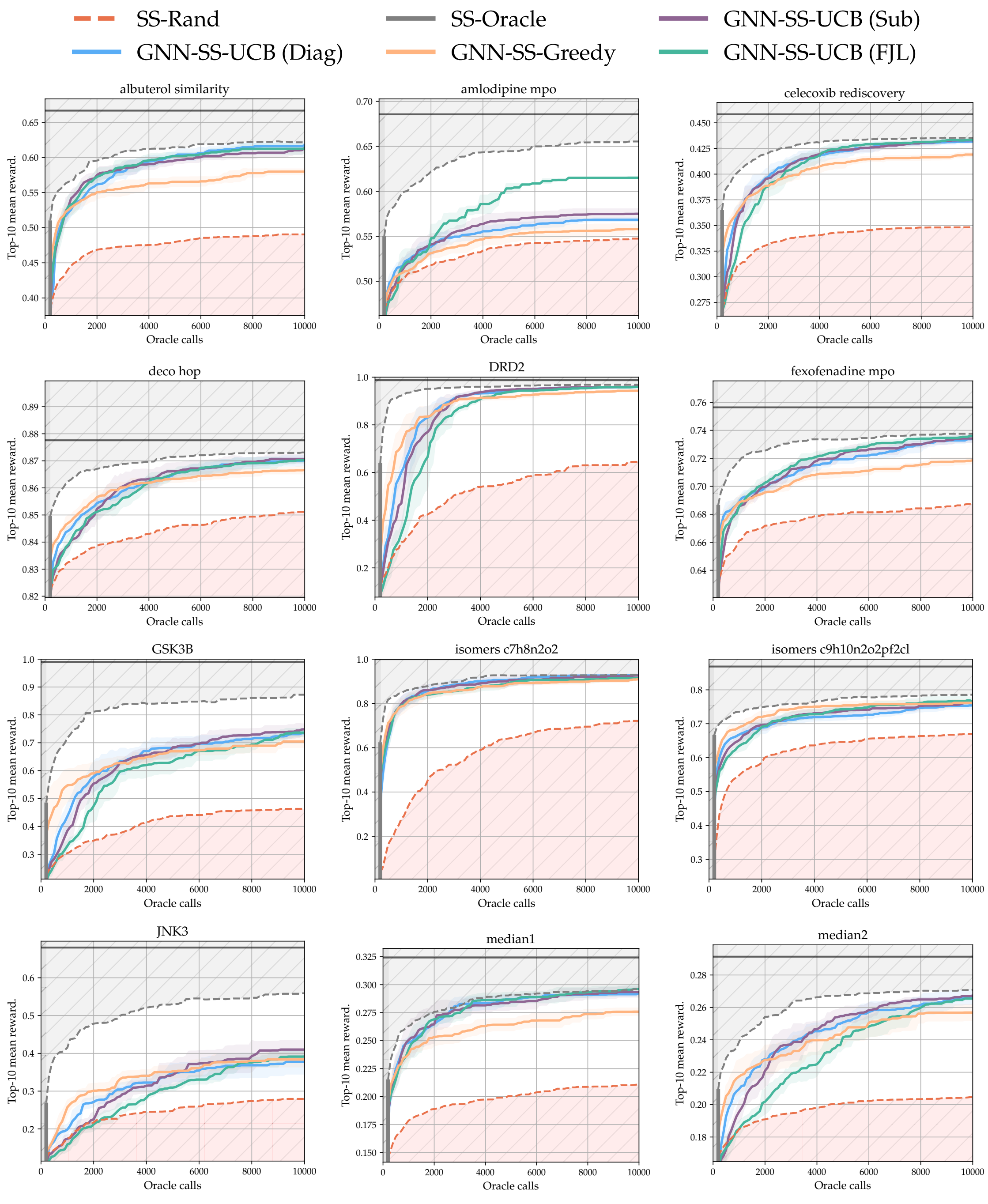
Molecular Objective	MolPAL	GNN-SS (Diag, NoLap)	GNN-SS (Diag)	GNN-SS (Sub)	GNN-SS (FJL)
albuterol sim.	-*	0.640 ± 0.017	0.639 ± 0.012	0.630 ± 0.016	0.642 ± 0.020
amlodipine mpo	0.621 ± 0.010	0.543 ± 0.007	0.581 ± 0.010	0.597 ± 0.013	0.660 ± 0.010
celecoxib redisc.	0.496 ± 0.002	0.436 ± 0.008	0.438 ± 0.008	0.440 ± 0.008	0.434 ± 0.008
deco hop	0.804 ± 0.019	0.874 ± 0.016	0.871 ± 0.004	0.871 ± 0.003	0.872± 0.003
drd2	0.902 ± 0.007	0.972 ± 0.011	0.951 ± 0.019	0.943 ± 0.024	0.918 ± 0.010
fexofenadine mpo	0.704 ± 0.001	0.744 ± 0.008	0.740 ± 0.009	0.743 ± 0.008	0.748 ± 0.006
gsk3b	0.776 ± 0.002	0.862 ± 0.086	0.763 ± 0.042	0.767 ± 0.039	0.742 ± 0.045
isomers c7h8n2o-	0.832 ± 0.005	0.979 ± 0.016	0.977 ± 0.019	0.968 ± 0.036	0.949 ± 0.041
isomers c9h10n2-	0.361 ± 0.009	0.785 ± 0.023	0.782 ± 0.027	0.780 ± 0.022	0.786 ± 0.021
jnk3	0.457 ± 0.024	0.552 ± 0.051	0.479 ± 0.065	0.500 ± 0.046	0.437 ± 0.047
median1	0.301 ± 0.000	0.303 ± 0.012	0.311 ± 0.009	0.313 ± 0.010	0.312 ± 0.010
median2	0.266± 0.000	0.275 ± 0.012	0.275 ± 0.010	0.273 ± 0.008	0.264 ± 0.009
mestranol sim.	0.708± 0.006	0.864 ± 0.029	0.785 ± 0.083	0.770 ± 0.085	0.765 ± 0.078
osimertinib mpo	0.803± 0.001	0.803 ± 0.004	0.804 ± 0.002	0.804 ± 0.002	0.804 ± 0.002
perindopril mpo	0.495± 0.003	0.463 ± 0.004	0.538 ± 0.012	0.532 ± 0.011	0.535 ± 0.010
qed	0.942± 0.000	0.947 ± 0.000	0.947 ± 0.000	0.947 ± 0.000	0.947± 0.000
ranolazine mpo	0.515± 0.007	0.556 ± 0.011	0.571 ± 0.006	0.567 ± 0.011	0.573 ± 0.005
scaffold hop	0.518± 0.001	0.524 ± 0.002	0.521 ± 0.003	0.519 ± 0.003	0.518 ± 0.004
sitagliptin mpo	0.100± 0.013	0.478 ± 0.001	0.393 ± 0.041	0.397 ± 0.030	0.367 ± 0.001
thiothixene red.	0.356± 0.000	0.391 ± 0.012	0.380 ± 0.015	0.381 ± 0.011	0.375 ± 0.013
troglitazone red.	0.290± 0.000	0.326 ± 0.000	0.315 ± 0.022	0.326 ± 0.023	0.328 ± 0.021
valsartan smarts	0.000± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zaleplon mpo	0.262± 0.004	0.514 ± 0.016	0.521 ± 0.017	0.534 ± 0.020	0.513 ± 0.013
Top-1 AUC sum	12.21	**13.83**	13.58	13.65	13.49
$n/29$ rank	10	**4**	7	5	8

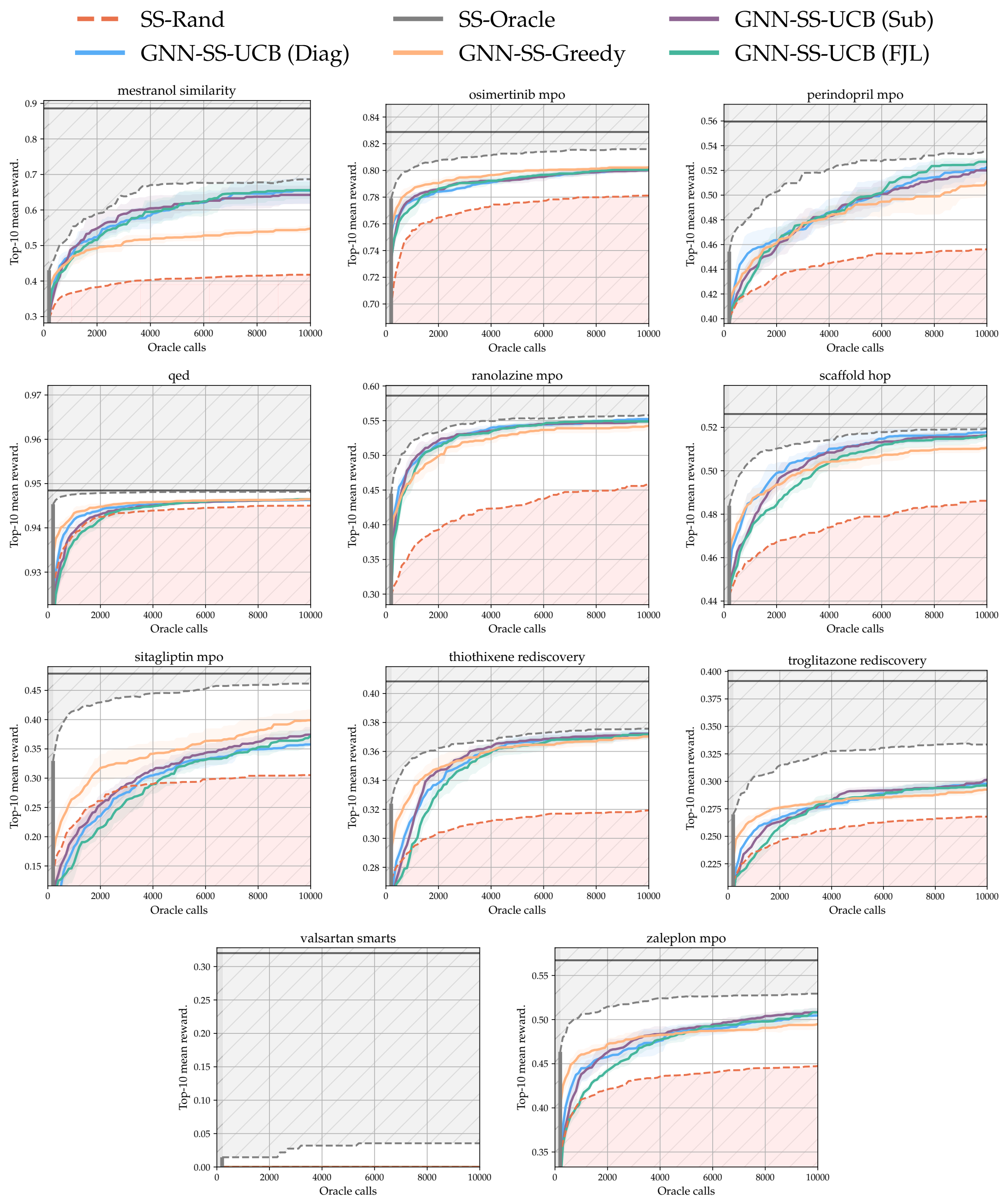
Table 3: **Full PMO benchmark results: Top-1 AUC.** The maximum reward for the Albuterol Similarity in the benchmark dataset is 0.666, but reported value in Gao et al. (2022) exceeds it. (NoLap) denotes no Laplacian positional encodings.

Molecular Objective	MolPAL	GNN-SS (Diag, NoLap)	GNN-SS (Diag)	GNN-SS (Sub)	GNN-SS (FJL)
albuterol sim.	-*	0.653 ± 0.017	0.667 ± 0.007	0.660 ± 0.014	0.653 ± 0.017
amlodipine mpo	0.651 ± 0.043	0.556 ± 0.012	0.597 ± 0.012	0.604 ± 0.006	0.678 ± 0.011
celecoxib redisc.	0.511 ± 0.041	0.450 ± 0.008	0.450 ± 0.008	0.451 ± 0.007	0.458 ± 0.000
deco hop	0.860 ± 0.102	0.878 ± 0.001	0.876 ± 0.002	0.877 ± 0.001	0.877 ± 0.002
drd2	0.964 ± 0.709	0.983 ± 0.010	0.976 ± 0.034	0.980 ± 0.005	0.981 ± 0.001
fexofenadine mpo	0.709 ± 0.006	0.756 ± 0.005	0.756 ± 0.002	0.756 ± 0.002	0.756 ± 0.002
gsk3b	0.820 ± 0.128	0.904 ± 0.089	0.847 ± 0.024	0.854 ± 0.022	0.860 ± 0.014
isomers c7h8n2o-	0.882 ± 0.163	1.000 ± 0.006	1.000 ± 0.006	1.000 ± 0.019	1.000 ± 0.017
isomers c9h10n2-	0.391 ± 0.091	0.821 ± 0.013	0.822 ± 0.033	0.822 ± 0.033	0.869 ± 0.000
jnk3	0.608 ± 0.117	0.620 ± 0.030	0.551 ± 0.079	0.566 ± 0.067	0.620 ± 0.005
median1	0.309 ± 0.028	0.315 ± 0.009	0.321 ± 0.006	0.315 ± 0.007	0.321 ± 0.006
median2	0.273 ± 0.021	0.291 ± 0.004	0.291 ± 0.003	0.284 ± 0.009	0.291 ± 0.004
mestranol sim.	0.733 ± 0.081	0.886 ± 0.041	0.845 ± 0.082	0.886 ± 0.012	0.824 ± 0.050
osimertinib mpo	0.816 ± 0.020	0.807 ± 0.004	0.806 ± 0.001	0.806 ± 0.002	0.806 ± 0.002
perindopril mpo	0.504 ± 0.020	0.483 ± 0.005	0.560 ± 0.002	0.555 ± 0.009	0.551 ± 0.006
qed	0.948 ± 0.002	0.947 ± 0.000	0.947 ± 0.000	0.947 ± 0.000	0.948 ± 0.000
ranolazine mpo	0.556 ± 0.064	0.577 ± 0.013	0.586 ± 0.002	0.581 ± 0.009	0.586 ± 0.002
scaffold hop	0.525 ± 0.016	0.526 ± 0.001	0.526 ± 0.001	0.526 ± 0.001	0.526 ± 0.001
sitagliptin mpo	0.117 ± 0.030	0.478 ± 0.000	0.463 ± 0.025	0.474 ± 0.008	0.478 ± 0.006
thiothixene red.	0.361 ± 0.016	0.408 ± 0.005	0.401 ± 0.014	0.404 ± 0.009	0.401 ± 0.013
troglitazone red.	0.296 ± 0.013	0.370 ± 0.008	0.346 ± 0.030	0.357 ± 0.026	0.370 ± 0.014
valsartan smarts	0.000± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
zaleplon mpo	0.286 ± 0.064	0.533 ± 0.014	0.556 ± 0.014	0.561 ± 0.012	0.528 ± 0.007
Top-1 sum	12.84	14.24	14.19	14.27	14.38
$n/29$ rank	17	10	11	8	**7**

Table 4: **Full PMO benchmark results: Top-1 Reward.** Standard errors are computed with 5 seeded replicates. (NoLap) denotes no Laplacian positional encodings.

Figure 4: GNN-SS Top-10 Mean Reward for all 23 objectives on the PMO benchmark.




SS-Rand
SS-Oracle
GNN-SS-UCB (Sub)
GNN-SS-UCB (Diag)
GNN-SS-Greedy
GNN-SS-UCB (FJL)
mestranol similarity
osimertinib mpo
perindopril mpo
qed
ranolazine mpo
scaffold hop
sitagliptin mpo
thiothixene rediscovery
troglitazone rediscovery
valsartan smarts
zaleplon mpo
Top-10 mean reward.
Oracle calls