# Graph Neural Network Powered Bayesian Optimization for Large Molecular Spaces

Miles Wang-Henderson [*1] Bartu Soyuer [*1] Parnian Kassraie [1] Andreas Krause [1] Ilija Bogunovic [2]

## Abstract

*In silico* screening is an essential component of 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-powered Bayesian Optimization (BO) algorithm as a simple, scalable solution. GNN-SS utilizes random sub-sampling to reduce the computational complexity of the BO problem, and diversifies queries for training the model. GNN-SS is sample-efficient, and rapidly narrows the search space by leveraging the generalization ability of GNNs. Our algorithm performs competitively on the QM9 dataset and achieves state-of-the-art performance amongst screening methods for the PMO benchmark.

## 1. Introduction

Molecular optimization is an important problem for early drug discovery and materials design, wherein the objective is to retrieve candidate compounds with one or more desirable properties. Key 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. BO techniques are renowned for their ability to rapidly narrow down the search space to the most desirable molecules. In particular, by modeling molecules as graphs, we can use BO methods which exploit the graph structure and utilize Graph Neural Networks (GNNs) that capture the complexities of molecular properties, and are in turn, more sample efficient.

While using a GNN may improve the statistical complexity of the problem, applying BO to vast discrete libraries is still computationally prohibitive. Since computing the acquisition function for each candidate molecule to determine the one that maximizes the acquisition score becomes expensive or even impossible. We address this issue by introducing GNN-SS, a light GNN-powered BO algorithm that randomly sub-samples the domain before optimizing the acquisition function. This reduces the computational complexity, and increases the diversity of collected data, which in turn, enhances exploration to improve the generalization ability of the network. We evaluate our algorithm with a limited evaluation budget, paired with various BO policies, on a domain of commercially available small molecules. We observe that GNN-SS-UCB achieves state-of-the-art performance on the PMO benchmark among algorithms for 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 regime (Kassraie et al., 2022). These methods require calculating the acquisition function over the entire domain, and thus, can not be scaled to large molecular libraries. Building upon Kassraie et al. (2022), we propose a scalable algorithm which exhibits competitive performance, while reducing

[*]Equal contribution [1]ETH Zurich, Switzerland [2]University College London, United Kingdom. Correspondence to: Miles Wang-Henderson <mwanghenders@ethz.ch>.

Accepted to ICML workshop on Structured Probabilistic Inference Generative Modeling. Copyright 2023 by the author(s).

the computation 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 that emerge in drug or material design, and formulate them as a Bayesian optimization problem on a domain of graphs, where the aim is to optimize an *unknown* objective function through sequential queries while receiving noisy function evaluations. We represent small molecules as undirected graphs with at most $N$ nodes which represent atoms or molecular substructures. 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 $\epsilon_t$ is i.i.d. zero-mean sub-Gaussian noise. We denote the history of observations by $H_t := \{(G_1, y_1), \dots, (G_t, y_t)\}$. For a horizon $T$ (i.e., sampling budget), we seek to obtain a small *simple* regret $r_T = f(G^*) - \max_{t \leq T} f(G_t)$, 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 billions of molecules (e.g. ZINC, Irwin & 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 $\mathrm{poly}(|\mathcal{G}|)$.

## 3. Method

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 two key ideas: 1) we use Graph Neural Networks which are known to be effective models for learning complex functions defined on graphs and 2) we devise a sub-sampling (SS) subroutine that allows for sampling diverse data for training the network. These two ideas come together in our GNN-powered Sub-Sampled BO algorithm (GNN-SS), sketched in Algorithm 1.

**Reward Model.** To model molecular properties, we use a graph neural network $f_{\mathrm{GNN}}(G; \boldsymbol{\theta}) : \mathcal{G} \to \mathbb{R}$ of width $m$, where $\boldsymbol{\theta} \in \mathbb{R}^p$ denotes the network's weights vector, $p$ is the total number of network parameters. We initialize the network with $\boldsymbol{\theta}_0$ which has zero-mean Gaussian i.i.d. entries with variance $1/m$, and update it as we receive more evidence on the objective function. As a proxy for $f$, we maintain $f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_t)$, the GNN trained on the loss

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

where $\lambda$ is the regularization coefficient. 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). For action-selection policies which require uncertainty quantification (e.g., UCB (Srinivas et al., 2010) or Thompson Sampling (Thompson, 1933)), we use the gradient of this network at initialization to approximate the uncertainty over $f$. Following Kassraie et al. (2022), we set

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

where $\boldsymbol{g}(G) = \nabla_{\boldsymbol{\theta}} f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_0)$ denotes the gradient at initialization. For very wide networks, $\hat{\sigma}_t$ presents a provably calibrated uncertainty estimate. Any network architecture may be used for estimating $f$ and its uncertainty (e.g., Kipf & Welling, 2017; Veličković et al., 2017; Xu et al., 2018; Lim et al., 2022). Lastly, we use ADAM (Kingma & Ba, 2014) for optimizing $\mathcal{L}$. In Appendix A, we outline our choice of architecture, and details of training the network.

**Acquisition Functions.** Given $(f_{\mathrm{GNN}}(\cdot; \boldsymbol{\theta}_t), \hat{\sigma}_t^2(\cdot))$ our reward and uncertainty estimator, we construct acquisition functions, which are used towards action-selection. We allow for multiple choices and consider,

$$\begin{aligned} \alpha_{\mathrm{Greedy}}(G; H_t) &= f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_{t-1}), \\ \alpha_{\mathrm{UCB}}(G; H_t) &= f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_{t-1}) + \beta \hat{\sigma}_{t-1}(G), \\ \alpha_{\mathrm{TS}}(G; H_t) &\sim \mathrm{GP}\left( f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_{t-1}), \beta \hat{\sigma}_{t-1}(G) \right). \end{aligned}$$

Here, the GREEDY acquisition function directly uses the reward estimate as a proxy for the true unknown reward. The UCB function considers an optimistic estimate of the reward by adding the uncertainty estimate to the reward estimate, and TS takes a probabilistic optimistic approach, and samples a function from the Gaussian process defined above.

**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 $\alpha$ for all members of $\mathcal{G}$, which comes with a high computational burden when working on datasets of billions of molecules. 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 member 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 & Bayati, 2020). This additional exploration helps build a diverse dataset $H_t$ for training $f_{\mathrm{GNN}}$ in the future steps, and in turn, yields an accurate reward estimator $f_{\mathrm{GNN}}(G; \boldsymbol{\theta}_{t-1})$ 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. In our experiments we choose it roughly as $\mathcal{O}(\sqrt{T})$, more details are included in Appendix A.

**Algorithm 1** GNN-SS

```
Input: network architecture $f_{\mathrm{GNN}}$, acq. function $\alpha(\cdot)$, horizon $T$, warm-start steps $T_0$, regularization parameter $\lambda$, random subset size $K$, batch size $b \le K$, training epochs $E$, (optional) pre-train dataset $H_{\mathrm{off}}$
Initialize: $\boldsymbol{K}_0 \leftarrow \lambda \boldsymbol{I}$, $\boldsymbol{\theta}_0 \sim \mathcal{N}(0, \boldsymbol{I}_p/m)$, $H_0 = \emptyset$.
$\boldsymbol{\theta}_0 \leftarrow$ TRAINGNN$(H_{\mathrm{off}}, \boldsymbol{\theta}_0, E)$    ▷ Pre-train
$\boldsymbol{g}(\cdot) \leftarrow \nabla_{\boldsymbol{\theta}} f_{\mathrm{GNN}}(\cdot; \boldsymbol{\theta}_0)$
for round $t = 1, \dots, T_0$ do    ▷ Exploration Stage
    $G_t \sim \mathrm{Unif}(\mathcal{G})$ and query $y_t = f(G_t) + \epsilon_t$
    $H_t \leftarrow H_{t-1} \cup \{(G_t, y_t)\}$
end for
$\boldsymbol{\theta}_{T_0} \leftarrow$ TRAINGNN$(H_{T_0}, \boldsymbol{\theta}_0, E)$
$\boldsymbol{K}_{T_0} \leftarrow \boldsymbol{K}_0 + \sum_{t=1}^{T_0} \sum_{i=1}^{b} \boldsymbol{g}(G_{t,i}) \boldsymbol{g}^\top(G_{t,i})$
for round $t = T_0 + 1, \dots, T$ do    ▷ Optimization Stage
    Sub-sample $\mathcal{G}_t \sim \mathrm{Unif}(\mathcal{G})$ where $|\mathcal{G}_t| = K$
    $\{G_{t,i}\}_{i=1}^{b} \leftarrow \arg\max_{G \in \mathcal{G}_t} \alpha(G; H_{t-1})$
    Query $y_{t,i} = f(G_{t,i}) + \epsilon_{t,i}$ for $i \in [b]$
    $H_t \leftarrow H_{t-1} \cup \{(G_{t,1}, y_{t,1}), \dots, (G_{t,b}, y_{t,b})\}$
    $\boldsymbol{\theta}_t \leftarrow$ TRAINGNN$(H_t, \boldsymbol{\theta}_{t-1}, E)$
    Update $\boldsymbol{K}_t \leftarrow \boldsymbol{K}_{t-1} + \sum_{i=1}^{b} \boldsymbol{g}(G_i) \boldsymbol{g}^\top(G_i)$
end for
```

### 3.1. Scaling to Large Databases

To obtain an algorithm that is sample efficient and computationally light, we employ a few more techniques. Algorithm 1 shows how these techniques come together to construct GNN-SS, and Table 1 shows the computational complexity of GNN-SS, which is considerably lighter than its competitors, since $K \ll |\mathcal{G}|$. We highlight that the following techniques are optional design choices, that may improve performance in some problem instances.

**Pre-training and Transfer.** In many problem instances, there is available offline data on the Bayesian optimization task at hand, or from similar tasks, e.g., same domain of molecules and different objective functions. One approach to improve sample efficiency is to pre-train the network on offline data, and then run GNN-SS. In case the available data is from other tasks, this translates to performing transfer learning. Appendix C.2, shows how pre-training on the data from a previously solved task, improves the performance on a correlated downstream task.

**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 & 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 $\alpha(\cdot; H_t)$ and the top-$b$ graphs are queries 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.

**Uncertainty Quantification.** Optimistic policies require inverting a $p \times p$ matrix (c.f. (1)). This operation is typically the memory and computation bottleneck of the algorithm, and limits the size of the networks that we can use. We present two solutions. Assuming that network parameters are uncorrelated, we use a diagonal approximation of this matrix, and reduce the $\mathcal{O}(p^3)$ cost of matrix inversion to linear. This assumption is often made for very wide networks (e.g., Yang, 2020). Another way to address this problem, is to parameterize the uncertainty function also as a neural network, and employ a second $f_{\mathrm{GNN}}$ as a proxy for $\sigma_t(\cdot)$, which we demonstrate in Algorithm 2.

*Table 1.* GNN-SS is computationally lighter than other GNN-powered baselines, since $K < T \ll |\mathcal{G}|$.

| METHOD | RUNTIME |
|---|---|
| MOLPAL (GRAFF ET AL., 2021) | $\mathcal{O}(T\lvert\mathcal{G}\rvert + bET^2)$ |
| GNN-UCB (KASSRAIE ET AL., 2022) | $\mathcal{O}(T\lvert\mathcal{G}\rvert + ET^2)$ |
| GNN-SS-UCB (OURS) | $\mathcal{O}(TK + bET^2)$ |

## 4. Results

ZINC (Irwin & Shoichet, 2005) and QM9 (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) are datasets of small organic molecules that are commonly used for benchmarking algorithms for molecular optimization (e.g., in Gao et al.,

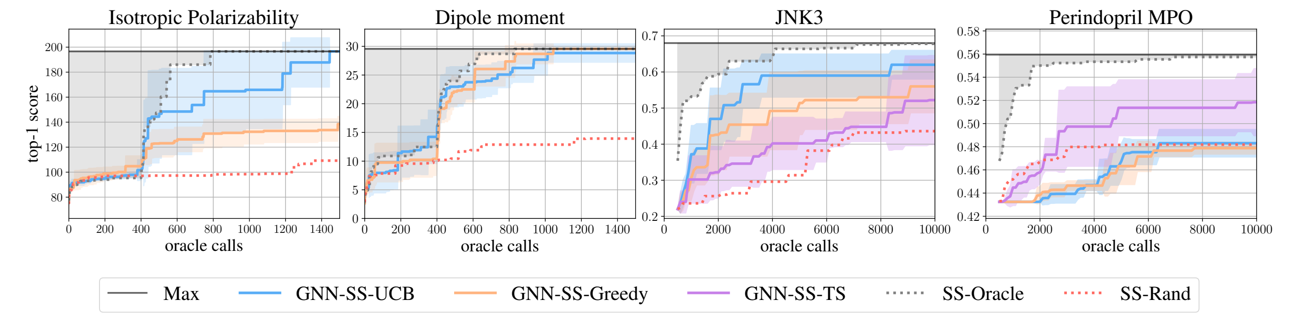

*Figure 1.* Top-1 optimization curves for rewards in QM9 (Isotropic Polarizablity, Dipole moment) and ZINC (JNK3, PERINDOPRIL MPO). Variants of GNN-SS approach the oracle in all instances. Horizontal line shows dataset's maximum and the gray area above GNN-SS-Oracle is unreachable by sub-sampled algorithms.

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.

For ZINC, which consists of 250k molecules, we consider all 23 molecular objectives of the Practical Molecular Optimization (PMO) benchmark (Gao et al., 2022), and include commonly used scores such as quantitative estimate of drug-likeness (QED) (Bickerton et al., 2012). The QM9 dataset (visualized in Figure 2) offers 19 objective functions related to the geometric, electronic, and thermodynamic properties of 134k molecules. Some objectives in these dataset are known to be challenging to optimize (Adachi et al., 2023; Liao & Smidt, 2023), due to the scarcity of molecules with non-zero values. This is referred to as a *needle-in-a-haystack* problem, where the shared structure of observed molecules with low-reward is uninformative about unseen molecules with high reward.

For QM9, which presents several complex reward functions, we instantiate GNN-SS with a Gated-MPNN (Gilmer et al., 2017) with edge features as $f_{\mathrm{GNN}}$, and for ZINC, we employ a vanilla GCN similar to Graff et al. (2021). To run GNN-SS on each dataset, we use different sets of hyper-parameters, and provide the tuning details in Appendix A. All results are average across 5 runs with different random seeds, and the standard error is reported. Figure 1 shows the performance of GNN-SS paired with UCB, TS and GREEDY policies on 4 objective functions, two from each of the mentioned datasets. We limit the horizon to a realistic sampling budget, and for every $t$, we plot the top-1 reward, i.e. $\max_{s\leq t} f(G_s)$ the value of the highest scoring molecule queried so far. 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 2 and 3, GNN-SS achieves state-of-the-art performance among non-generative models, including MOLPAL, and ranks 4th overall 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). Appendix C.4 presents benchmarking results on QM9, where we compare variants of Algorithm 1 to SOBER (Adachi et al., 2023) and BGNN-BO (Kim et al., 2022).

*Table 2.* Summary of PMO benchmark, complete results in Table 3

| MOLECULAR PROPERTY | MOLPAL (GRAFF ET AL.) | GNN-SS-UCB (OURS) |
|---|---|---|
| GSK3B | 0.776 ± 0.002 | **0.862 ± 0.086** |
| ... | ... | ... |
| QED | 0.942± 0.000 | **0.947 ± 0.000** |
| SITAGLIPTIN MPO | 0.100± 0.013 | **0.478 ± 0.000** |
| ZALEPLON MPO | 0.262± 0.004 | **0.514 ± 0.016** |
| TOP-1 AUC SUM | 12.21 | 13.83 |
| $n/29$ RANK | 10 | 4 |

## 5. Conclusion

We presented GNN-SS, a Graph Neural Network BO algorithm that draws on sub-sampling as a scalable solution to the problem of screening very large molecular libraries. Following the results of Kassraie et al. (2022), we quantify the epistemic uncertainty of GNN estimates, allowing us to employ optimistic action-selection policies. We demonstrated that GNN-SS achieves competitive performance on the PMO benchmark for sample-efficient molecular optimization, and on the challenging QM9 dataset. Scaling BO to virtual databases with billions of molecules, or efficiently going beyond a library and performing *de novo* design are still unresolved. We conjecture that, optimistic action-selection based on auto-regressive GNN frameworks (Bradshaw et al., 2020) may be a solution to this problem.

## References

Adachi, M., Hayakawa, S., Hamid, S., Jørgensen, M., Oberhauser, H., and Osborne, M. A. Sober: Scalable batch bayesian optimization and quadrature using recombination constraints, 2023.

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

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

Bayati, M., Hamidi, N., Johari, R., and Khosravi, K. Unreasonable effectiveness of greedy algorithms in multi-armed bandit with many arms. *Advances in Neural Information Processing Systems*, 2020.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 2012.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M., and Hernández-Lobato, J. M. Barking up the right tree: an approach to search over molecule synthesis dags. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6852–6866, 2020.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

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

Gao, W., Fu, T., Sun, J., and Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry, 2017.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 2018.

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

Hao, B., Lattimore, T., and Wang, M. High-dimensional sparse linear bandits. *Advances in Neural Information Processing Systems*, 2020.

Hirschfeld, L., Swanson, K., Yang, K., Barzilay, R., and Coley, C. W. Uncertainty quantification using neural networks for molecular property prediction, 2020.

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y. H., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., and Zitnik, M. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Neurips Datasets And Benchmarks*, 2021.

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

Kassraie, P., Krause, A., and Bogunovic, I. Graph neural network bandits. In *Advances in Neural Information Processing Systems*, 2022.

Kim, S., Lu, P. Y., Loh, C., Smith, J., Snoek, J., and Soljacic, M. Deep learning for bayesian optimization of scientific problems with high-dimensional structure. *Transactions of Machine Learning Research*, 2021.

Kim, S., Lu, P. Y., Loh, C., Smith, J., Snoek, J., and Soljačić, M. Deep learning for bayesian optimization of scientific problems with high-dimensional structure, 2022.

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., and Xing, E. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Liao, Y.-L. and Smidt, T. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs, 2023.

Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H., and Jegelka, S. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection, 2018.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

Nicolaou, C. A., Watson, I. A., Hu, H., and Wang, J. The proximal lilly collection: Mapping, exploring and exploiting feasible chemical space. *Journal of chemical information and modeling*, 2016.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 2014.

Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 2012.

Soleimany, A. P., Amini, A., Goldman, S., Rus, D., Bhatia, S. N., and Coley, C. W. Evidential deep learning for guided molecular property prediction and discovery. *ACS Central Science*, 2021.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. Finite-time analysis of kernelised contextual bandits. *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, 09 2013.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., and Li, L. {MARS}: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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

Zhang, W., Zhou, D., Li, L., and Gu, Q. Neural thompson sampling. *International Conference On Learning Representations*, 2020.

Zhou, D., Li, L., and Gu, Q. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

## A. Experiment Details

### A.1. GNN-SS for Virtual Screening on ZINC

*Table 3.* Full PMO benchmark results. Bold indicates a better result than MOLPAL. Standard errors are computed with 5 seeded replicates. (*) The maximum reward for Albuterol similarity in the benchmark dataset is 0.666.

| MOLECULAR OBJECTIVE | MOLPAL (GRAFF ET AL.) | GNN-SS-UCB (OURS) |
|---|---|---|
| ALBUTEROL SIM. | 0.694* ± 0.003 | 0.640 ± 0.017 |
| AMLODIPINE MPO | 0.621 ± 0.010 | 0.543 ± 0.007 |
| CELECOXIB REDISC. | 0.496 ± 0.002 | 0.436 ± 0.008 |
| DECO HOP | 0.804 ± 0.019 | **0.874** ± **0.016** |
| DRD2 | 0.902 ± 0.007 | **0.972** ± **0.011** |
| FEXOFENADINE MPO | 0.704 ± 0.001 | **0.744** ± **0.008** |
| GSK3B | 0.776 ± 0.002 | **0.862** ± **0.086** |
| ISOMERS C7H8N2O2 | 0.832 ± 0.005 | **0.979** ± **0.016** |
| ISOMERS C9H10N2O2PF2CL | 0.361 ± 0.009 | **0.785** ± **0.023** |
| JNK3 | 0.457 ± 0.024 | **0.552** ± **0.051** |
| MEDIAN1 | 0.301 ± 0.000 | **0.303** ± **0.012** |
| MEDIAN2 | 0.266± 0.000 | **0.275** ± **0.012** |
| MESTRANOL SIM. | 0.708± 0.006 | **0.864** ± **0.029** |
| OSIMERTINIB MPO | 0.803± 0.001 | 0.803 ± 0.004 |
| PERINDOPRIL MPO | 0.495± 0.003 | 0.463 ± 0.004 |
| QED | 0.942± 0.000 | **0.947** ± **0.000** |
| RANOLAZINE MPO | 0.515± 0.007 | **0.556** ± **0.011** |
| SCAFFOLD HOP | 0.518± 0.001 | **0.524** ± **0.002** |
| SITAGLIPTIN MPO | 0.100± 0.013 | **0.478** ± **0.001** |
| THIOTHIXENE REDISC. | 0.356± 0.000 | **0.391** ± 0.012 |
| TROGLITAZONE REDISC. | 0.290± 0.000 | **0.326** ± **0.000** |
| VALSARTAN SMARTS | 0.000± 0.000 | 0.000 ± 0.000 |
| ZALEPLON MPO | 0.262± 0.004 | **0.514** ± **0.016** |
| TOP-1 AUC SUM | 12.21 | 13.83 |
| $n/29$ RANK | 10 | 4 |

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 default parameters: $m = 384$, $L = 3$ graph convolutional layers, $\lambda = 0.0001$, $b = 100$, $K = 4000$, $T = 10,000/b$. We select the best performing exploration coefficient $\beta$ via a grid-search over $\{0.001, 0.01, 0.1, 1.0\}$. As an additional experiment, we use a larger network with $m = 512$ and absolute Laplacian positional-encodings (LaPEs) with $k = 6$ (Dwivedi et al., 2023), for the three MPO objectives that are functions of the number of substructures, e.g. aromatic rings. The results are: AMLODIPINE MPO **0.621** ± **0.025**, RANOLAZINE MPO **0.565** ± **0.017**, PERINDOPRIL MPO **0.539** ± **0.016**.

We maintain a diagonal approximation of the Gram matrix, and for our ZINC experiments we slightly modify the equation for $\hat{\sigma}_t$ from Kassraie et al. (2022), and following Zhou et al. (2020) instead use the updated gradients $\boldsymbol{g}(G; \boldsymbol{\theta}_{t-1})$ to calculate the uncertainty,

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

We represent each molecule as an undirected atomic graph with node feature matrix $X \in \mathbb{R}^{N\times 128}$, where $N$ is the number of nodes. Features encode atomic identity, formal charge, number of hydrogens, hybridization, and ring membership. All node features are first normalized to the unit sphere $X_j \in \mathbb{S}^{d-1}$.

In order to model the rewards of ZINC, we employ a vanilla GCN as our surrogate function. 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 & 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. Linear layers are similarly parameterized, following Kassraie et al. (2022).

$$f^{(l)}_{\mathrm{GCN}}(X, A) = \frac{c_\phi}{\sqrt{m_l}}\phi\Big(D^{-1/2}AD^{-1/2}X^{(l-1)}W^{(l)}\Big)$$

The graph representation is obtained as $\bar{X} = \texttt{pool}(X)$ before passing to a final linear layer. We experimented with `set2set, mean, sum, max` and $\texttt{cat}(\bar{X}_{\text{mean}}, \bar{X}_{\text{sum}})$ pooling layers, and selected the best performing pooling operation.

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 of high (non-zero) reward, as seen in Table 3.

### A.2. GNN-SS for Virtual Screening on QM9

The default parameters used for all the experiments are, $\beta = 0.5$, $\lambda = 0.003$, $T_0 = 400$, $T = 1500$, $K = 200$, $b = 100$ where $K$, $T$ and $b$ are occasionally fine-tuned slightly depending on the reward optimized during the experiments. We select the best performing exploration coefficient $\beta$ via a grid-search over $\{0.001, 0.01, 0.1, 1.0\}$.

In this variant of GNN-SS, we utilize two separate models, one for the posterior mean, namely, $f_{\mathrm{MPNN}}$, and the other for computing the posterior variance denoted as $f_{\mathrm{GCN}}$. We use the `pytorch_geometric` implementation of a Gated-MPNN (Gilmer et al., 2017) for $f_{\mathrm{MPNN}}$ and a vanilla GCN (Kipf & Welling, 2017) with one hidden layer of width $m = 256$ for $f_{\mathrm{GCN}}$. To approximate the posterior variance, we use the formulation presented in (Valko et al., 2013), which is equivalent to (1). In simple terms, the approximate posterior variance for any new sample $G$, at step $t$ is computed as

$$\hat{\sigma}^2_t(G) = \lambda^{-1}\left(k(G, G) - \boldsymbol{k}_t^\top(G)(\boldsymbol{K}_t + \lambda \boldsymbol{I})^{-1}\boldsymbol{k}_t(G)\right)$$

where $\boldsymbol{K_t} = \{k(G_i, G_j)\}_{i,j\le t}$ is the $t \times t$ kernel matrix, $\boldsymbol{k}(G) = [k(G, G_i)]_{i\le t}$, and $k(\cdot,\cdot) := \boldsymbol{g}(\cdot; \boldsymbol{\theta}_0^{\mathrm{GCN}})^\top \boldsymbol{g}(\cdot; \boldsymbol{\theta}_0^{\mathrm{GCN}})$. As shown in subsection C.3, this method yields well-calibrated confidence sets with the added benefit of speeding up the uncertainty computations as $t \ll p$.

`ReLU` non-linearities are used in all of the modules of the network. The graphs are first processed through a `linear+ReLU` layer, followed by recurrent layers, namely, `GatedRecurrentUnits`, which apply message passing and learn edge features at the same time by treating the hidden states of the nodes and the messages as sequences. Here, the initial inputs to the recurrent layers are the the node and edge features $\{\boldsymbol{h}_{G,j}, \boldsymbol{e}_{G,i,j}\}_{i,j=1}^N$ where $\boldsymbol{h}_{G,j} \in \mathbb{R}^{15}$ consist of the spatial information of the atoms within the molecules along with the atom features defined in (Gilmer et al., 2017) and $\boldsymbol{e}_{G,i,j} \in \mathbb{R}^4$ is a one-hot-encoding vector denoting the bond type (including `None`) between nodes $i$ and $j$. Finally, through an `LSTM` based module, the previously generated sequences of node states are mapped to an aggregated graph representation $\bar{h} = \bar{h}_{\text{set2set}}$ before passing to a final linear layer.

In each experiment, the Gated-MPNN is pre-trained offline with $|H_{\text{off}}| \in \{4500, 5000\}$ molecules for $E = 100$ epochs. We use `MSELoss` with a `StepLearningRateScheduler`, with parameters `factor=0.7`, `patience=5` and `min_lr=1e-5`. Online training is enacted for `GD_stop_count=10000` gradient descent steps with a learning rate `lr=1e-3`.

Throughout all the experiments, we use `MSELoss` except for the "Dipole Moment($\mu$)" reward, where we have preferred to use a natural adaptation of `FocalLoss`(Lin et al., 2018) to the regression setting. The reason behind this choice, was due to the 'needle in a haystack' optimization setting caused by this reward as motivated in (Adachi et al., 2023). To elaborate, this reward characteristically has few favorable candidates which are also, generally speaking, uncorrelated with most alternative candidates, thus making them hard to pinpoint or generalize well into. For this purpose, we have used `FocalLoss` in the hopes of emphasizing the high MSEs that stem from misidentifying these top candidates, thus driving the network to generalize better into these regions which has seemingly proven useful.

## B. More on QM9 Rewards

We briefly discuss rewards in the QM9 dataset. We select the considered rewards in order to compare to existing works. As such, we have included the Dipole Moment reward in our experiments as in (Adachi et al., 2023). We can better observe in Figure 2, the *needle-in-a-haystack* setting mentioned in Section 4 where most of the candidates are of low-reward. Only

**Algorithm 2** Dual Network GNN-SS

```
**Input:** Data $\mathcal{G}$, main network architecture $f_{\text{MPNN}}(.)$, smaller network architecture $f_{\text{GCN}}(.;\theta)$, horizon $T$, warm-start steps $T_0$, exploration coefficient $\beta$, regularization parameter $\lambda$, batch size $b$, random subsample size $K$, pre-train subset $H_{\text{off}}$, training epochs $E$
**Initialize:** $\alpha(\cdot) \leftarrow \{\alpha_{\text{UCB}}, \alpha_{\text{TS}}, \alpha_{\text{Greedy}}\}$
$\mathcal{L} \leftarrow \{\text{Focal}(\cdot;\cdot), \text{MSE}(\cdot;\cdot)\}$
$\text{UncEst} \leftarrow \{\text{MahalanobisApprox.}(\text{Valko et al., 2013}), \text{DiagonalApprox.}\}$
$H_{\text{off}} \sim \text{Unif}(\mathcal{G})$
$\boldsymbol{\theta}_0^{\text{MPNN}} \leftarrow \text{TRAINGNN}(H_{\text{off}}, \boldsymbol{\theta}_0^{\text{MPNN}}, E)$    ▷ Offline Supervised Pre-training
$\boldsymbol{\theta}_0^{\text{GCN}} \sim \mathcal{N}(0, \boldsymbol{I}_p/m)$
$g(\cdot) \leftarrow \nabla f_{\text{GCN}}(\mathcal{G}; \boldsymbol{\theta}_0^{\text{GCN}})$
$\boldsymbol{K_0} \leftarrow \emptyset$
Buffer for acquired candidtes $H_0 = \emptyset$
**for** round $t = 1, , ..., T_0$ **do**    ▷ Exploration Phase
    $G_t \sim \text{Unif}(\mathcal{G})$
    Query $y_t = f(G_t) + \epsilon_t$
    $H_t \longleftarrow H_{t-1} \cup \{(G_t, y_t)\}$
**end for**
$\boldsymbol{\theta}_{T_0}^{\text{MPNN}} \leftarrow \text{TRAINGNN}(H_{T_0}, \boldsymbol{\theta}_0^{\text{MPNN}})$
$\boldsymbol{K_{T_0}} \longleftarrow \boldsymbol{K_0} \cup \{g^T(G_{t,i})g(G_{t,j})\}_{i,j \le T_0}$
**for** round $T_0 < t \le T$ **do**    ▷ Optimization Phase
    Sub-sample $\mathcal{G}_t \sim \text{Unif}(\mathcal{G})$ where $|\mathcal{G}_t| = K$
    $\{G_{t,i}\}_{i=1}^b \leftarrow \arg\max_{G \in \mathcal{G}_t} \alpha(G; H_t, \beta)$
    Query $y_{t,i} = f(G_{t,i}) + \epsilon_{t,i}$ for $i \in [b]$
    $H_{t+b} \leftarrow H_t \cup \{(G_{t,1}, y_{t,1}), \ldots, (G_{t,b}, y_{t,b})\}$
    $\boldsymbol{\theta}_{t+b} \leftarrow \text{TRAINGNN}(H_t, \boldsymbol{\theta}_t^{\text{MPNN}}, E)$
    Update $\boldsymbol{K_{t+b}} \longleftarrow \boldsymbol{K_t} \cup \{g^T(G_{t,i})g(G_{t,j})\}_{i,j \in [b]}$
    t = t + b
**end for**
```

several of the top candidates are structurally similar to undesirable molecules. The global maximum, in particular, appears structurally dissimilar to the rest of the data.

Secondly, we include the Isotropic Polarizability reward, as studied in (Kim et al., 2022), which quantifies the dipole polarization of molecules under an electric field. As visible in Figure 2, we notice that this reward has distributional characteristics similar to Dipole Moment and hence is likely prone to the same challenges we face in that reward. Unsurprisingly, in Figure 4, we notice how the global maximum in particular is extremely isolated from all closest candidates in terms of the reward value.

Rest of the rewards are picked referring to our qualitative assessments of the UMAP visualizations of the rewards in Figure 2. In Figure 2, we observe that some rewards have smoother and continuous distributions across the reward surface, whereas some rewards suffer from a 'needle in a haystack' phenomenon. Therefore, since we want to evaluate our algorithm under the two settings, we would have to pick a mix of rewards corresponding to the two categories. Additionally, when we inspect the heatmap in Figure 3, we immediately notice that certain rewards are almost perfectly positive correlated and we strictly avoid using such pairs of rewards in our experiments as they can trivialize the variety in our optimization tasks.

## C. Further Experiments on QM9

### C.1. Predicted Means vs Rewards

In this section, we share our plots depicting the true rewards against the means predicted by the network at the final step of the algorithm in Figure 4. We clearly see that our previous observations in Appendix B align with the findings in these plots in that we notice the discontinuous reward distributions and the scarcity of the top candidates manifesting in an extreme fashion for the Dipole Moment, Isotropic Polarizability and HOMO-LUMO Energy Gap rewards. In particular, we

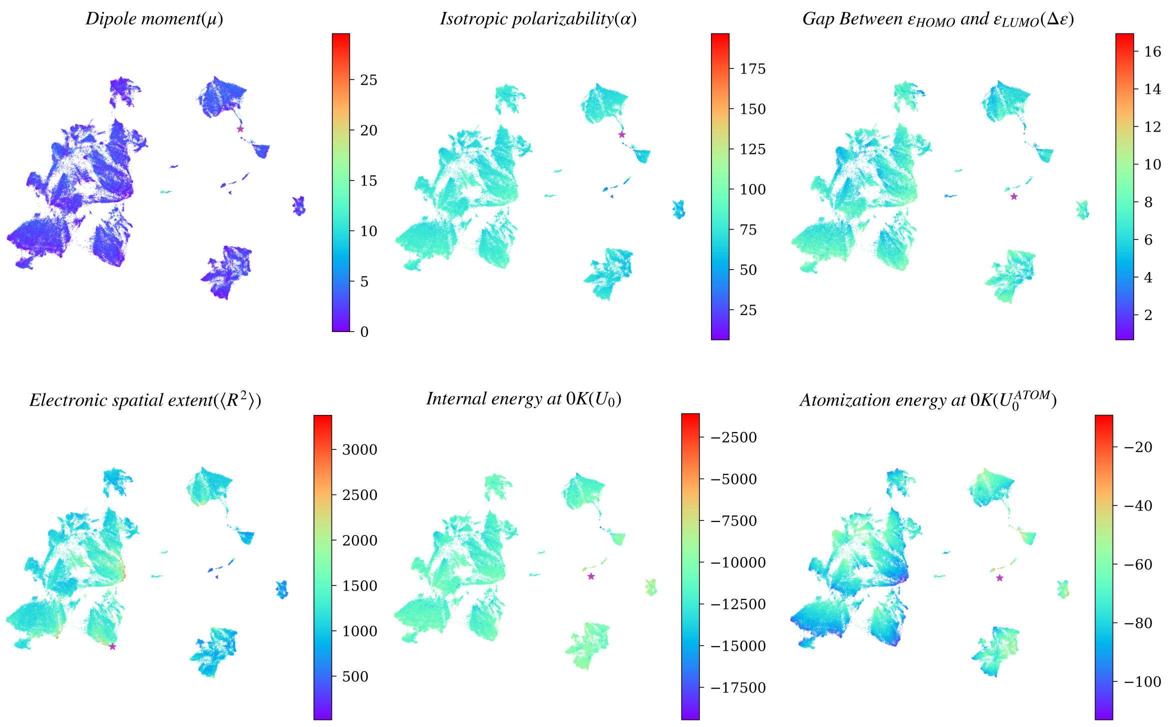

*Figure 2.* The UMAP visualizations of the QM9 rewards we have used in our benchmark experiments. The global maximum in each plot is denoted with a purple "⋆" symbol.

notice that the top candidates are necessarily isolated from the nearest candidates, therefore, posing challenges in terms of generalizing well into and in turn, acquisition. For all rewards, we underline that despite these challenges, our algorithm converges to the global maximum at least once in 5 runs as reported in Table 4.

## C.2. Transfer Learning

In this section, we present our results regarding the transfer learning performance of the algorithm to investigate whether transfer learning indeed yields a substantial difference in performance for downstream optimization tasks. For the purpose of transfer learning, we have chosen to pretrain our network on a small random subset of 6000 samples with the "HOMO-LUMO Energy Gap" surrogate reward and then observe the performance of the algorithm on the "Lowest Unoccupied Molecular Orbital Energy" reward. The reason for this choice primarily stems from the correlation of these two rewards as apparent from Figure 3. Thus, we have reason to believe that transfer learning between these two rewards could in fact be meaningful.

Moreover, in our experiments with different QM9 rewards, we have identified that our default algorithm's (GNN-SS-UCB with untuned hyperparameters) performance on the "Lowest Unoccupied Molecular Orbital Energy" reward fell behind the other rewards. On the other hand, as presented in Figure 7, our algorithm actually had competitive performance on the reward "HOMO-LUMO Energy Gap" and managed to solve the optimization task. Therefore, this is an ideal showcase where a practitioner could first solve the optimization for one reward and then use the trained network for transfer learning on a more challenging yet correlated reward.

As observed in Figure 5, transfer learning indeed yields a significant leap in the performance of the algorithm as transfer learning with GNN-SS-UCB clearly outperforms the default GNN-SS-UCB algorithm on all three metrics reported.

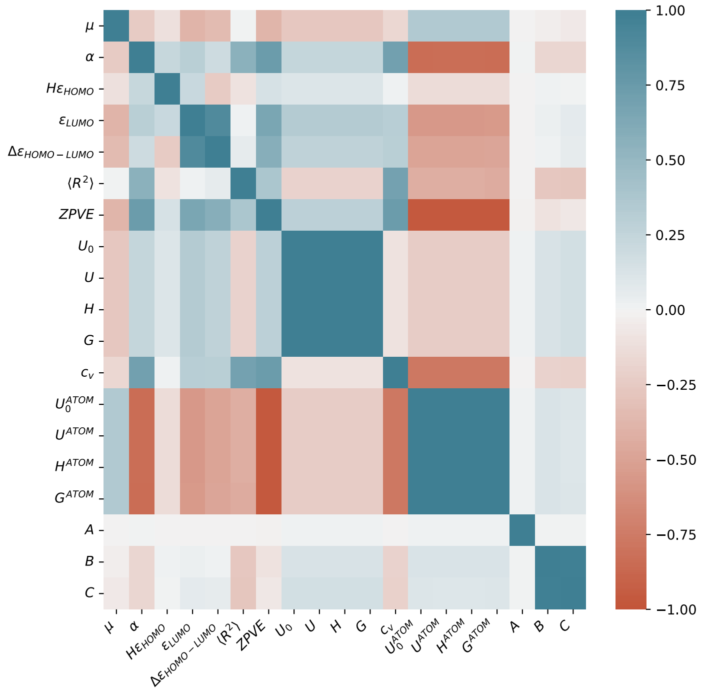

*Figure 3.* Heatmap of the pairwise PearsonR correlations between all the rewards, computed over the molecules in the QM9 dataset .

## C.3. QM9 Calibration Experiments

In Figure 6, we report the calibration curves of the predicted confidence intervals, for all QM9 rewards we have benchmarked. The x-axis is $\alpha$, the confidence level considered, and the y-axis reports the empirical frequency of samples covered within that $\alpha$-level confidence set, defined as $\sum_{i=1}^{|H_{\text{val}}|} \mathbf{1}(y_i \leq F^{-1}(\alpha))/|H_{\text{val}}|$. A perfect calibration would overlap with the $x = y$ line. Falling below this indicates insufficient coverage and means our algorithm is overconfident.

## C.4. Existing Baselines for QM9

We finally compare our collective results on QM9 to GBNN-BO (Kim et al., 2022) and SOBER (Adachi et al., 2023) which report results for the Isotropic Polarizability and Dipole Moment rewards respectively. For the SOBER benchmark, we limit our oracle budget to $6,000$ evaluations in accordance with the reported sample size in the paper. For the other benchmark, we limit our budget to $6,500$ queries. Due to the limited and varying amount of experimental results presented in these benchmarks, we compare them on the basis of maximum rewards acquired within horizon and global convergence to be consistent across benchmarks. We notice that our algorithm performs at least as good as these baselines.

Benchmark results are summarized by comparing the different variants of our algorithm on the chosen QM9 rewards in Table 4. Firstly, our work stands out among other recent works in the neural bandit literature by presenting various baselines for a large collection of QM9 rewards in a self-contained fashion. Moreover, our work also performs competitively in terms of the cumulative regret and Top-K performances across multiple QM9 rewards including Dipole Moment and Isotropic Polarizability, which might have high relevance for the practitioner. We further observe that, in most cases, the GNN-SS-GREEDY variant performs on par with the GNN-SS-UCB variant, in line with the findings of (Bayati et al., 2020). This outcome is most welcome, as we believe that the GNN-SS-GREEDY variant could be a competitive and less costly alternative to SS-UCB, especially as we scale up to much larger libraries in the future. We do note however, that for the case of the 'Lowest Unoccupied Molecular Orbital Energy' and 'Isotropic Polarizability' GNN-SS-UCB outperforms GNN-SS-GREEDY and we suspect this might be related to the 'needle in a haystack' problem encountered in these

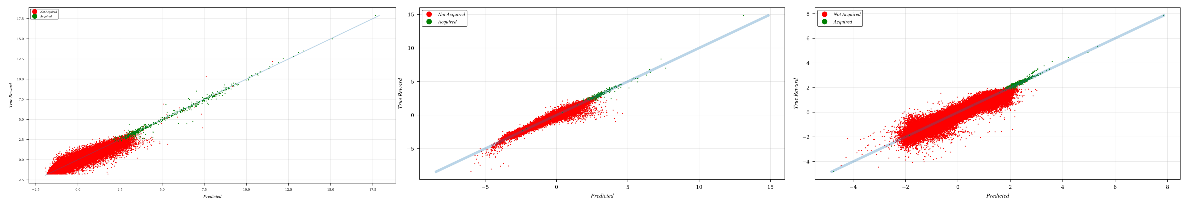

*Figure 4.* The true rewards (y-axis) plotted against the final predicted means (x-axis) for all molecules in QM9. The molecules are also colored to distinguish between collected (green) and uncollected (red) samples within the horizon of the algorithm. The results are depicted for the rewards: "Dipole Moment ($\mu$)", "Isotropic Polarizability ($\alpha$)", "Gap Between $\epsilon_{HOMO}$ and $\epsilon_{LUMO}$ ($\Delta\epsilon$)" from left to right.

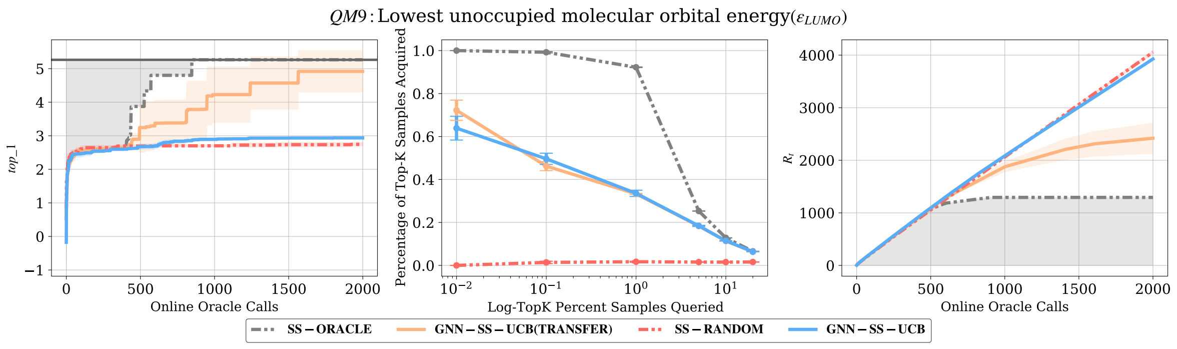

*Figure 5.* The Top-1 optimization curve and the Top-K Accuracy and Cumulative Regret plots (left to right), comparing the performance of our GNN-SS-UCB algorithm on the "Lowest unoccupied molecular orbital energy ($\epsilon_{LUMO}$)" reward, against the same task with additional transfer-learning from "Gap Between $\epsilon_{HOMO}$ and $\epsilon_{LUMO}$ reward. The results depicted are averaged over 5 random seeds.

rewards or poor hyperparameter tuning. Nevertheless, in the very large scale setting, the source of randomness injected by our subsampling strategies alone may be sufficient to guide the exploration accurately (Bayati et al., 2020) and therefore we could replace UCB acquisition with simple greedy selection, also eliminating any computational overhead caused by uncertainty quantification.

*Table 4.* Benchmark results for our algorithm on the QM9 dataset using the GNN-SS-UCB, GNN-SS-GREEDY, SS-ORACLE and SS-RANDOM sampling variants. The Top-%0.01 performance and the empirical frequency of **C**onvergence to **G**lobal **M**aximum are reported over 5 runs with different random seeds each.

| Reward | **GNN-SS-UCB** | | **GNN-SS-GREEDY** | | **SS-RANDOM** | | **SS-ORACLE** | |
|---|---|---|---|---|---|---|---|---|
| | Top-K(0.01%) | CGM | Top-K(0.01%) | CGM | Top-K(0.01%) | CGM | Top-K(0.01%) | CGM |
| Dipole Moment($\mu$) | 0.833±0.048 | 0.8 | 0.833±0.074 | 1.0 | 0.000±0.000 | 0.0 | 0.916±0.091 | 1.0 |
| Isotropic Polarizability($\alpha$) | 0.819±0.155 | 1.0 | 0.600±0.185 | 0.0 | 0.027±0.039 | 0.0 | 0.983±0.033 | 1.0 |
| Gap Between $\epsilon_{HOMO}$ and $\epsilon_{LUMO}$($\Delta_\epsilon$) | 0.694±0.104 | 0.4 | 0.466±0.227 | 0.0 | 0.000±0.000 | 0.0 | 0.987±0.064 | 1.0 |
| Electronic Spatial Extent($\langle R^2\rangle$) | 0.902±0.057 | 1.0 | 0.917±0.053 | 0.8 | 0.000±0.000 | 0.0 | 1.000±0.000 | 1.0 |
| Internal Energy at 0K($U_0$) | 0.900±0.062 | 1.0 | 0.917±0.048 | 1.0 | 0.041±0.063 | 0.0 | 1.000±0.000 | 1.0 |
| Atomization Energy at 0K($U_0^{ATOM}$) | 0.933±0.062 | 1.0 | 0.958±0.042 | 1.0 | 0.000±0.000 | 0.0 | 1.000±0.000 | 1.0 |

*Table 5.* Comparison of our algorithm against two recent benchmarks for QM9, SOBER and BGNN-BO. We note that the sampling complexities for our algorithm include the randomly chosen samples used for supervised pre-training.

| Metric | **Dipole Moment** | | **Isotropic Polarizability** | |
|---|---|---|---|---|
| | **GNN-SS-UCB** | **SOBER** | **GNN-SS-UCB** | **BGNN-BO** |
| Top Reward Acquired | 29.557 | 29.557 | 196.620 | 143.53 |
| Vanishing Regret | ✓ | ✓ | ✓ | ✗ |
| Sampling Budget | 6000 | 6000 | 6500 | 500 |

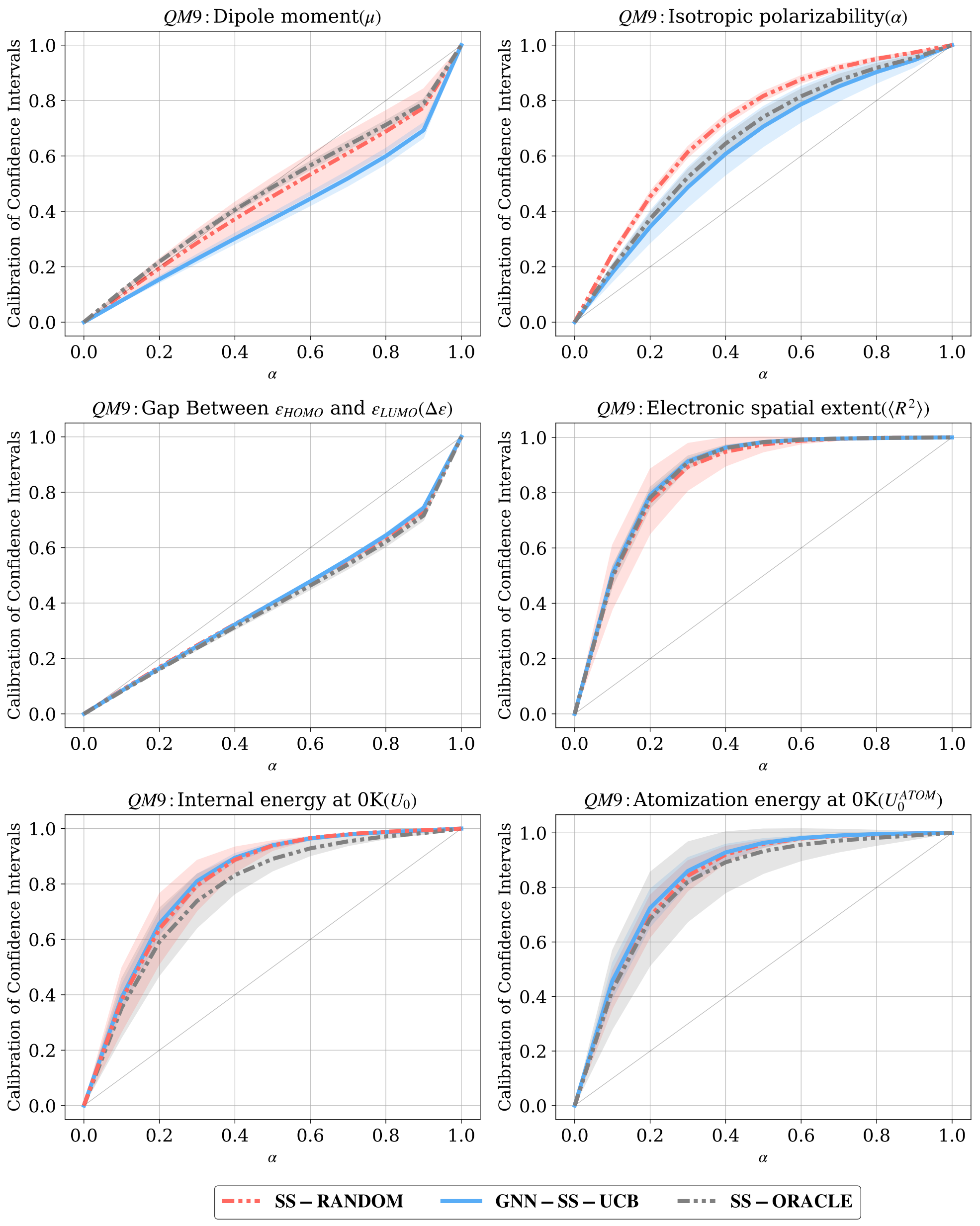

*Figure 6.* Calibration Curves for the GNN-SS-UCB, SS-ORACLE and SS-RANDOM sampling variants of our algorithm on 6 distinct rewards of the QM9 Dataset. The shared results are computed over 5 runs with different random seeds each. GNN-SS-GREEDY isn't included in this plot since the posterior confidences aren't computed for this variant.

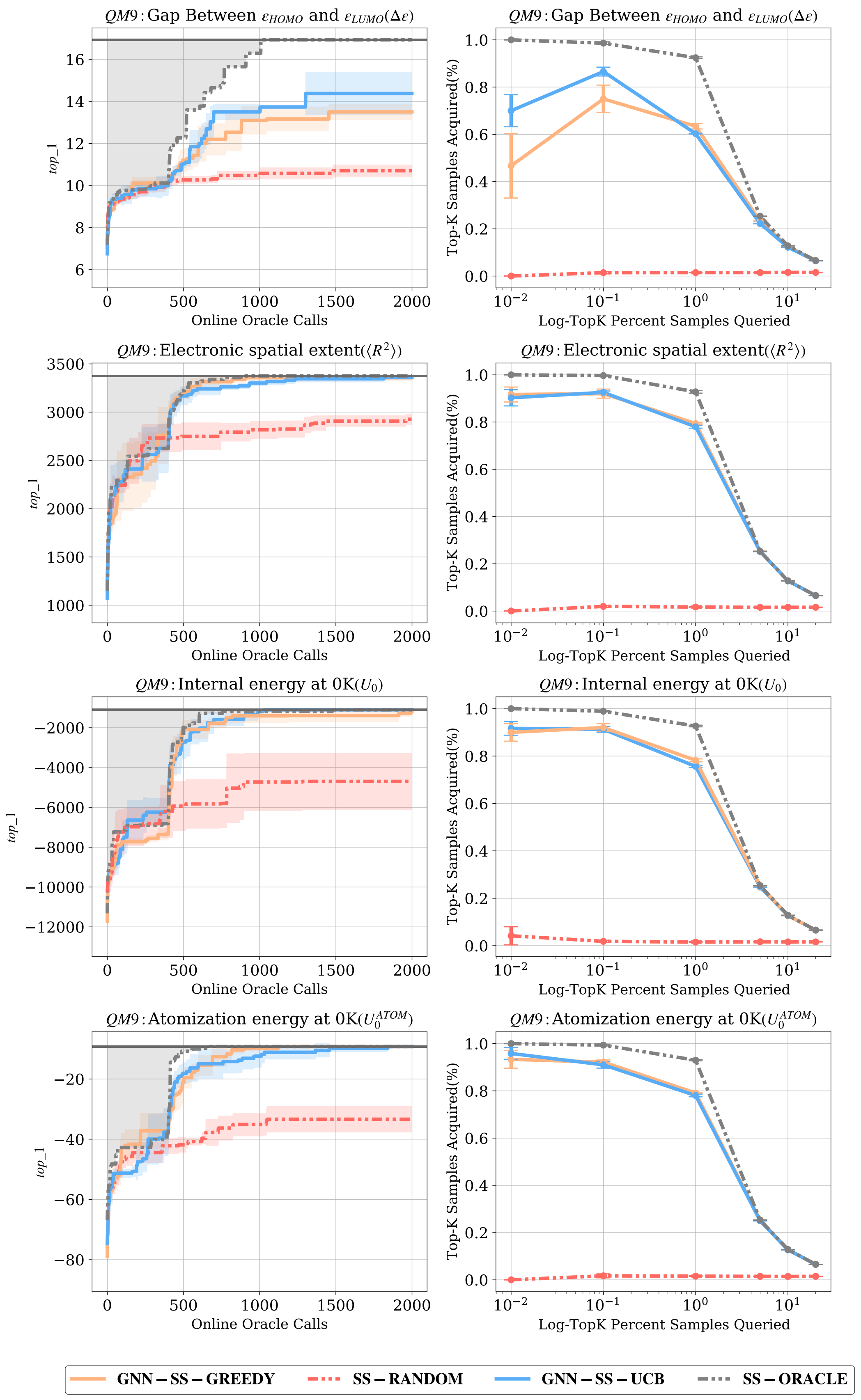

*Figure 7.* The Top-1 optimization curves (left) and Top-K accuracy plots (right) associated with GNN-SS-UCB, GNN-SS-GREEDY, SS-ORACLE and SS-RANDOM sampling variants of our algorithm on miscellaneous QM9 rewards.The shared results are computed over 5 runs with different random seeds each.