# High Dimensional Bayesian Optimization with Reinforced Transformer Deep Kernels for industrial and scientific applications

**Anonymous submission**


## Abstract

Tuning parameters is a major hurdle for successfully designing industrial or research processes. Bayesian Optimization is often used for sample efficient tuning when an analytic solution is intractable, and the evaluation procedure for candidate parameter values is costly, especially in higher dimensions. In this paper, we combine recent developments in Deep Kernel Learning (DKL) and attention-based Transformer models to improve the modeling powers of GP surrogates with meta-learning. We propose a novel method for improving meta-learning BO surrogates by incorporating attention mechanisms into DKL, empowering the surrogates to adapt to contextual information gathered during the BO process. We combine this Transformer Deep Kernel with a learned acquisition function using Soft Actor-Critic Reinforcement Learning to aid in exploration. This Reinforced Transformer Deep Kernel (RTDK) approach produces state-of-the-art results on various high dimensional optimization problems. We demonstrate this approach on certain classes of costly to evaluate, analytically intractable problems that have real-world applications in multiple domains. We compare our approach against other recently proposed Reinforcement Learning augmented Bayesian Optimization approaches like MetaBO, TAF, and FSAF and achieve competitive results with better performance on higher dimensions in a sample efficient manner.


## Introduction

Sample efficient and high dimensional optimization is at the core of many industrial and scientific processes with applications including material design (Zhang, Apley, and Chen 2020), physics (Carr, Garnett, and Lo 2016), synthetic chemistry and biology (Shields et al. 2021; Barnes et al. 2011), and hyperparameter optimization (Snoek, Larochelle, and Adams 2012). The success of these industrial and scientific processes is often governed by the correct choice of parameters. Due to the complex nature of these processes and the associated cost to run them, tuning these parameters by solving analytical equations is intractable. It is also expensive to tune them by trial and error. One particular class of problem we are interested in is the Thompson problem of arranging a set of N electrons on a sphere so that they have the minimum potential energy. Many existing problems in physical chemistry like how electrons are arranged around atoms; in virus-morphology like determining spatial configuration of proteins on a virus; discretization of manifolds where we are interested in optimal placement and design of airplane wings etc. rely on this formulation (LANL 2015). The Thompson problem is difficult to solve using standard optimization techniques due to the complicated topology and lack of recursive structure in the problem, leading researchers to use non-conventional methods for optimization (Mes 2022). The Electric Grid (Liu, Song et al. 2022) problem is another analytically intractable problem that tries to find the best configuration of generators and system load while keeping the grid frequency stable. It is difficult to solve this problem using a model-free trial and error approach due to the safety critical requirements for maintaining stability while optimizing the energy generation for a given load. In the domain of space-research and looking for minerals, the asteroid routing problem presents a class of problems where the goal is to find the optimal route for visiting a set of different locations that vary in space temporally while having a resource budget (López-Ibáñez, Chicano et al. 2022).

Bayesian Optimization (BO) is a ubiquitous technique that has proven to be very promising across all the domains listed above, and is often the standard for sample-efficient block-box optimization (Frazier 2018). However, Bayesian Optimization relies on the design of a *surrogate model* which estimates the optimization objective in yet unexplored regions, as well as an *acquisition function* for effectively exploring the optimization domain. Both of these components typically require domain knowledge to adapt to specific optimization objectives, and this design is critical to BO's performance on challenging domains.

In this paper, we leverage the contextual representation power of Transformers in a Deep Kernel Bayesian Optimization setting to solve the Thompson and related analytically intractable optimization problems, especially in higher dimensions. We further improve it by using existing techniques in Reinforcement Learning based acquisition functions that does not need any gradient information from the black-box functions. This is the first work to the best of our knowledge that combines the power of RL and Transformer based Deep Kernel into a unified architecture where embedding information is shared across the surrogate and Acquisition function. This is what enables the approach to be applied to higher dimensions in a sample efficient manner while using Deep Learning models. We perform

competitively compared to other recent deep learning-based Bayesian Optimization techniques in the higher dimensional space. Our belief is that the solution approach would benefit different fields, where the existing problem can be converted to an equivalent problem we solve.

## Related Work

**Deep Kernel Learning** Deep Kernel Learning (DKL) (Wilson et al. 2015; Ober, Rasmussen, and van der Wilk 2021) extends the learning capability of the Gaussian Processes (GP) by mapping the original optimization domain into a new domain via a parameterized transformation, such as a deep neural network. The underlying GP is then trained on these embedding inputs after neural network reprocessing. The GP parameters are optimized via log-likelihood minimization, and the gradients are passed along to the embedding through back-propagation.

**Reinforcement Learning Acquisitions** Reinforcement learning (RL) approaches to Bayesian optimization have recently shown promising results, especially for discrete objectives. MetaBO presents a seminal framework for interpreting the acquisition function within Bayesian optimization as a reinforcement learning policy, to be trained with policy gradient methods (Volpp et al. 2020). This work has been further generalized in (Hsieh, Hsieh, and Liu 2021) to allow for few-shot q-learning instead of a discrete policy optimization. These approaches present useful frameworks for tackling small sample Bayesian optimization, but they primarily focus on the acquisition function, leaving the surrogate model as a traditional Gaussian process. Additionally, both approaches rely on a discrete grid to perform optimization, approximating continuous domain optimization with a quasi-random hierarchical grid.

**Attention and Transformers** Attention mechanisms (Luong, Pham, and Manning 2015) provide a method for deep neural networks to modify their activations in response to a set of contextual vectors. This allows us to condition a parameterized neural network on an arbitrary number of unordered vectors. Attention has been used in various network architectures to achieve state-of-the-art results in many applications, including natural language processing (Vaswani et al. 2017) and computer vision (Dosovitskiy et al. 2021). We will use the contextual embeddings of transformers to assist in rapidly optimizing black box functions from observations on similarly structured objectives.

## Preliminary: Bayesian Optimization

Bayesian Optimization (BO) corresponds to a general set of techniques for optimizing a black-box function $f(x) : \mathbb{X} \rightarrow \mathbb{Y}$ by: fitting a *surrogate model* to estimate function values across the optimization domain; and employing an *acquisition function* based on the surrogate to select promising query points (Frazier 2018).

The surrogate model, $\hat{f}(x; x_{train}, y_{train}) = \hat{f}(x; \mathcal{D})$, provides a probabilistic estimate of the objective across the entire optimization domain given a sparse sample of points $x_{train} = \{x_1, x_2, \ldots, x_k\}$ where the objective is known $y_{train} = \{y_1, y_2, \ldots, y_k\}$. Surrogates typically provide both a mean estimate of the function value $\mu(x; \mathcal{D})$, and an uncertainty for that estimate in the form of a variance $\sigma^2(x; \mathcal{D})$.

After fitting the surrogate on the observed dataset $\mathcal{D}$, the acquisition function, $\mathcal{A}(x; \mathcal{D})$, defines a score for selecting the next query point $x_{query}$. Bayesian optimization selects queries by maximizing the acquisition $x_{query} = \arg\max_{x \in \mathbb{X}} \mathcal{A}(x; \mathcal{D})$. Therefore, the acquisition must be responsible for balancing exploration to ensure a global optimum across the domain and exploitation to optimize locally within a promising region. One common acquisition function, especially with Gaussian Process surrogates, is the Expected Improvement (EI) criterion (Jones, Schonlau, and Welch 1998).

In this work, we aim to improve both aspects of BO by introducing deep neural networks to both the surrogate and acquisition functions while maintaining the generality of the BO approach. These improvements will minimize required training data while ensuring that these methods work in both continuous and discrete optimization domains.

## Proposed Approach: Conditional Deep Kernel with Reinforced Acquisition Function

### Conditional Deep Kernel Surrogate

The Gaussian Process (GP) (Rasmussen and Williams 2005) is a fundamental architecture for BO surrogate models (Frazier 2018). GPs estimate the objective with a Gaussian posterior over functions fitting the observed data: $\hat{f}(y|x; \mathcal{D}) \sim \mathcal{N}(\mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D}))$. We use a learned mean component which is parameterized by a linear transformation of the input $\mu_W(x) = Wx$. In general, both the mean and covariance components may have parameters that must be learned. The parameters of a Gaussian process are trained to maximize the log-likelihood of the training dataset $\sum_{x,y \in \mathcal{D}} \log P(\hat{f}(y; x, \mathcal{D}) = y)$.

The GP covariance is determined by a *Kernel*, $K(x_1, x_2)$, which defines the distance between any two points within a high dimensional (sometimes infinite) manifold. This kernel representation, along with the Gaussian likelihood, defines an analytical posterior distribution over the optimization domain given a set of observed function values. The choice of kernel function is crucial for well-fitting GPs, and many domains have specially designed kernel functions to fit domain-specific data.

### Combination Kernel

In the interest of generality, we want a consistent architecture that works reasonably well across various domains. A common kernel in traditional GPs is the RBF, which corresponds to a dot-product in an infinite dimensional manifold and may be defined as an infinite sum of polynomial kernels (Rasmussen and Williams 2005).

$$K_{RBF}(x_1, x_2) \propto \sum_{k=1}^{\infty} \frac{\langle x_1, x_2 \rangle^k}{k!}$$

Taking inspiration from this representation, we choose to instead approximate this high dimensional embedding up to

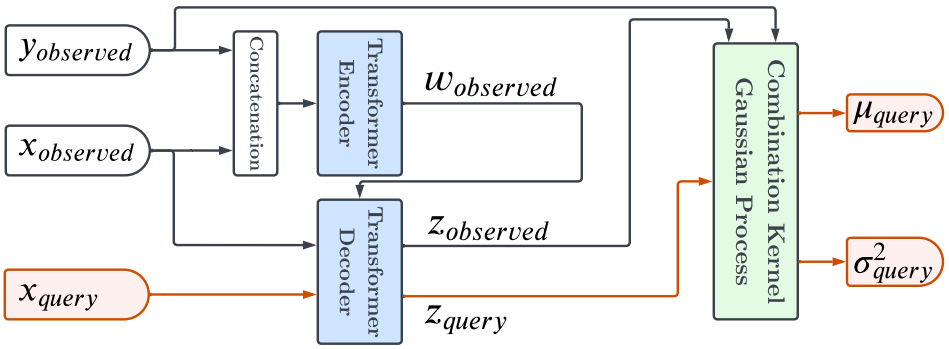



Figure 1: Block Diagram of Transformer Deep Kernel Learning Gaussian Process. The red path indicates the prediction path for the query point. The black paths indicate the contextual information.

a certain power, $K$, learning the coefficients, $\alpha_k$, as part of the kernel's parameters. We call this kernel the *Combination* up to $K$, or $C(K)$, kernel.

$$K_{C(K)}(x_1, x_2) = \sum_{k=1}^{K} \frac{\alpha_k \langle x_1, x_2 \rangle^k}{k!} \tag{1}$$

We find this kernel provides additional learning potential over a fixed RBF kernel. A hyperparameter selects the maximum polynomial.

## Normalized Deep Kernel Learning

We find that DKL sometimes experiences numerical instability, especially when training for Bayesian Optimization tasks, which typically have very little data. We, therefore, introduce an additional parameterized neural network, $v_\phi(x)$, to explicitly estimate the diagonal components of the covariance, while using a normalized variant (Rasmussen and Williams 2005) of the kernel, $K(x_1, x_2)$, to estimate the non-diagonal components. This allows us to use a flexible kernel while avoiding large values due to a poorly conditioned embedding network. $K_{normalized}(x_1, x_2) = \exp v_\phi(x_1) \cdot \exp v_\phi(x_2) \cdot \frac{K(x_1,x_2)}{K(x_1,x_1)K(x_2,x_2)}$

## Transformer Deep Kernel Learning (TDKL)

Transformers present a mechanism for adding arbitrary context sequences to condition neural network activations. Crucially for BO, since the context does not necessarily require a unique target, it may include additional information that is not present in traditional BO observations. Specifically, we include not just the previous sample points $x_{observed}$ in the context but also the known function values for those points $y_{observed}$. This mechanism has been shown to be sufficient for Bayesian inference by itself (Müller et al. 2021) and we follow a similar framework for conditioning a DKL embedding on previously observed data.

Formally, we extend the DKL framework to include a conditioning term on the embedding network, $z_{query} = g_\theta(x_{query}|x_{observed}, y_{observed})$, where $g$ is a sequence-to-sequence transformer encoder-decoder model (Vaswani et al. 2017). The observed data, $(x_{observed}, y_{observed}) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_K, y_K)\}$, is first fed through the transformer encoder to produce the latent encoded sequence $w_{observed} = \{w_1, w_2, \ldots, w_K\}$. This is used as the keys and values for the decoder, whereas the target sequence $x_{query} = \{x_{K+1}, x_{K+2}, \ldots, x_N\}$ is used as the query for the decoder. The output sequence, $z_{query} = \{z_{K+1}, z_{K+2}, \ldots, z_N\}$, represents a conditional embedding of the query locations. We also produce the conditional embedding of the original observed locations, $z_{observed} = \{z_1, z_2, \ldots, z_K\}$, to condition the downstream Gaussian Process. The output distribution is parameterized by our GP, $\hat{f}(y|z_{query}; z_{observed}, y_{observed})$. See Figure 1 for a flow diagram for all inputs. Similarly to (Müller et al. 2021), we remove the temporal embedding from the input to ensure that the transformer is invariant to sequence order. Additionally, we ensure that query points do not attend to each other by enforcing a diagonal attention mask on the decoder query sequence.

We refer to this complete surrogate model - employing a transformer embedding, learned point-wise variances, and a combination base kernel - as the Transformer Deep Kernel Learning (TDKL) surrogate. Unlike (Müller et al. 2021), we focus this architecture on sample efficiency, relying on the Gaussian process mechanism to encode the uncertainty in our predictions. This design introduces an inductive bias towards smooth functions, which may be represented by our combination kernel GP within the embedding space, but we find that this assumption does limit our performance due to the learning potential of the transformer.

## Soft Actor-Critic Acquisition

We turn our focus to the BO acquisition function. The TDKL described above provides a robust surrogate model which can estimate a function given a sufficiently robust set of samples. To collect such samples, we need a policy that will sufficiently explore the domain. For this purpose, we will employ a soft actor-critic (SAC) reinforcement learning agent (Haarnoja et al. 2018). We will represent the Bayesian optimization problem as a Markov decision process and use this formulation to train a novel model-based soft actor-critic agent

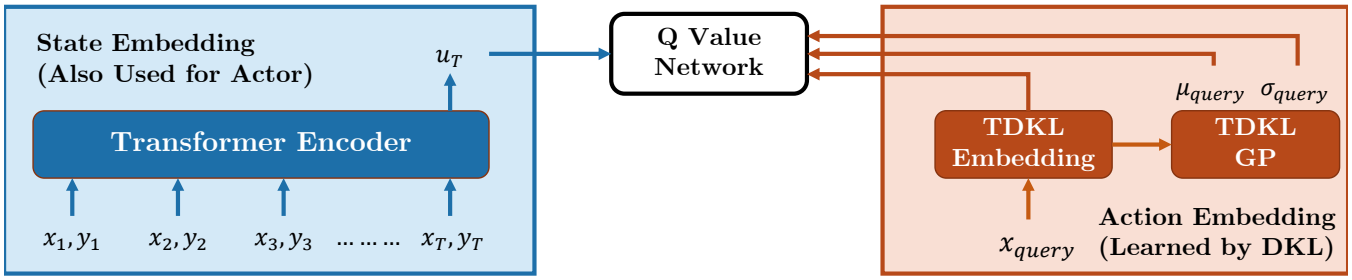



Figure 2: Flow diagram for the critic network, showing how the state-action pair is embedded and processed into a Q-Value estimate.

## Optimization Environment

Following the representations presented in MetaBO (Volpp et al. 2020) and FSAF (Hsieh, Hsieh, and Liu 2021), the state space for the problem at any time $t$ of the optimization process is defined as the collection of points and values in the BO trajectory, $s = (x_{observed}, y_{observed}) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\}$. The action space corresponds to the space of possible locations where we can sample defined by the input domain $\mathbb{X}$, and action represents the next candidate point, $a_t = x_{t+1} \in \mathbb{X}$. We define the reward in terms of the (approximate) regret, $r_{t+1} = -log(f^* - f(x_{t+1}))$, where $f^*$ is the estimated true optimum for the function. We consider finite-length trajectories determined by the budget for the number of steps in each trajectory $t < T_{max}$.

In general, each trajectory may originate from a different underlying objective. This enables meta-learning between similar objectives as both the agent and model must perform well across many different trajectories. We may also collect additional trajectories from the same environment if we wish to continue optimizing the given objective. Sampled trajectories are stored in a replay buffer, separated by their objective variant.

## Model-Based Soft Actor-Critic

Following the soft actor-critic framework, our agent consists of two learned Q-value network $q_1(s, a)$ and $q_2(s, a)$, a conservative estimate of the q value $q(s, a) = \min\{q_1(s, a), q_2(s, a)\}$, and a probabilistic policy $\pi(s)$. We extend SAC to a model-based reinforcement learning method by introducing the learned surrogate model into the framework.

**Surrogate Model** The surrogate model $\hat{f}$ will be trained on function samples from the replay buffer. After sampling a trajectory $s = \{(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\}$, we shuffle this trajectory and arbitrarily split the $x$ and $y$ values into *observation* and *query* datasets using a uniform random splitting pivot $M \sim \mathcal{U}(1, T-1)$. These datasets, $\mathcal{D}_{observed} = \{(x_1, y_1), \ldots, (x_M, y_M)\}$ and $\mathcal{D}_{query} = \{(x_{M+1}, y_{M+1}), \ldots, (x_T, y_T)\}$ are then used to optimize the surrogate on purely observed data.

$$\mathcal{L}_{model} = \sum_{(x_{query}, y_{query}) \in \mathcal{D}_{query}} -\log P(\hat{f}(y; x_{query}, \mathcal{D}_{observed}) = y_{query}) \quad (2)$$

Notice that this loss function differs from the traditional GP optimization because the training dataset which conditions our TDKL is different than the prediction dataset. This is to ensure that the TDKL learns a general representation regardless of which objective originated the trajectory. Optimizing this loss with a stochastic gradient-descent optimizer, sampling a different trajectory after each step, presents a cheap method for meta-learning GP models on a variety of objectives.

**Critics** We train the dual critic networks using the standard entropy-corrected Bellman update described in (Haarnoja et al. 2018). However, we wish to include information learned by the surrogate in the critic networks to improve sample efficiency. We accomplish this by exploiting the learned embeddings of the TDKL and the GP estimates of the objective.

The state, $s$, is embedded using another transformer architecture (Vaswani et al. 2017). This time, the transformer is acting only as an encoder to transform the variable-length observations, $s = \{(x_1, y_1), \ldots, (x_T, y_T)\}$, into a fixed-size latent embedding. We encode the sequence into a latent sequence using the transformer encoder $\{u_1^{critic}, \ldots, u_T^{critic}\} = TransformerEncoder(s)$ and then simply take the final latent vector, $u_T^{critic}$, as the fixed-length embedding.

The action, $a$, is encoded by passing the suggested point through the TDKL to extract both the embedding and function estimates from the surrogate model. $w_a = g_\theta(a|s))$ represents the embedding from the TDKL transformer $g_\theta$, and $\mu_a, \sigma_a^2 = \hat{f}(a|w_a; s)$ are the surrogate model estimates for the objective at the action location.

The Q-networks, therefore, become functions are the more abstract state-action representation, $q(u_T^{critic}, w_a, \mu_a, \sigma_a^2)$, allowing information to be shared between the model and critics. Note that TDKL parameters are treated as constant w.r.t the critic, and the gradient is not passed to the TDKL when optimizing the Bellman loss. A diagram of the critic architecture is presented in Figure 2.

**Actor** The actor network, $\pi(s)$, uses the same architecture as the state encoding from the Q-networks (The left-hand component in Figure 2). We use a separately trained transformer encoder to construct a fixed-length embedding for the state representation. $u_T^{actor} \in TransformerEncoder(s)$. Following the methods described in (Haarnoja et al. 2018), we use a tanh-squashed

normal for the actor distribution and apply an affine transform to fit the desired optimization domain.

### Acquisition Exploration in Continuous Domains

We find that relying purely on the actor for selecting good actions while exploring performs poorly on the small sample counts found in Bayesian optimization. Therefore, instead of sampling the action directly from $a \sim \pi(s)$ like in traditional SAC, we would like to incorporate the Q networks into the policy.

To do this, we take inspiration from Boltzmann exploration (Cesa-Bianchi et al. 2017), a common exploration technique in discrete environments. This involves constructing a policy that will sample proportional to a Boltzmann distribution based on the Q-network, $P(a|s) \propto \exp Q(s,a)$. We perform this kind of Boltzmann sampling when optimizing discrete domains using the softmax function, but this approach fails for continuous optimization domains.

Extending this to the continuous domain can be difficult because we cannot generally sample from arbitrary functions in high dimensions. However, if we believe that our actor generally learns the landscape of our Q-function, then we can sample from the actor in order to assist with generating samples from the Boltzmann Q using importance sampling.

First, we sample a large batch of actions from the policy, $\{a_1, a_2, \ldots, a_N\} \sim \pi(s)$, where $N$ is typically in the thousands. Then, we compute the importance weights of each sampled points w.r.t the Boltzmann Q, $w_i = \frac{exp(Q(s,a_i))}{P(\pi(s)=a_i)}$. Finally, construct an empirical distribution over $\{a_1, a_2, \ldots, a_N\}$ using these importance weights and sample an action $a$ such that $P(a = a_i) = \frac{w_i}{\sum_{k=1}^{N} w_k}$. As long as the actor has a non-zero probability of sampling anywhere in the optimization domain, then this process will produce samples from exactly the Boltzmann Q distribution as $N \to \infty$. In practice, to trade off memory and computation time, we use a value of $N = 1024$

While this process is quite slow, the priority on sample efficiency in the BO environment allows us to spend more computation time selecting each action when collecting objective samples. We use this sampling technique to select actions during inference. Unfortunately, this process is too slow to execute when performing gradient updates, so we use the regular $\pi(s)$ policy with the re-parameterization trick to compute the actor loss during SAC training.

## Experiments

We evaluate the RTDK Bayesian optimization approach on a variety of test functions. First, we confirm that this fully learnable approach can still achieve reliable results in simpler discrete optimization tasks. Then we explore the effectiveness of this fully continuous approach on the class of analytically intractable and costly to evaluate functions that relate to the class of problems discussed in the Introduction.

We use the RTDK surrogate with continuous SAC acquisition as the model of study for all experiments. The RTDK uses a transformer DKL embedding and a combination kernel with 5 components $C(5)$. RTDK evaluation differs slightly from other BO methods because the transformer models must be trained on example trajectories. Therefore, we must present the model with similarly sized trajectories during both training and evaluation. To accommodate longer evaluation trajectories, we split the BO run into "sub-trajectories" of length 50, resetting the surrogate after 50 steps. This allows us to only train on trajectories up to length 50. Each sub-trajectory also begins by taking 5 uniform random function samples to initialize the RTDK surrogate. Additionally, we add a large action noise to the first sub-trajectory in order to encourage the RTDK to explore during this initial phase. We examine the effect of these sub-trajectories as discussed later during the ablation studies in Figure 6.

### Discrete Optimization Baseline

We evaluate RTDK, as well as a variety of baselines, on discrete, real-world optimization tasks. In order to aid in direct comparison, we evaluate this method on the same set of optimization problems and methods as (Hsieh, Hsieh, and Liu 2021). These include the previously discussed asteroid routing, grid stability as well as related problems in particulate matter (PM2.5), dataset optimization (HPOBench XGB) and an extremely difficult problem in oil drilling location selection (Oil 4D)

We compare against baseline acquisition based on a traditional Gaussian process including Expected Improvement (EI) (Močkus 1975), Probability of Improvement (PI)(Kushner 1964), Max-value entropy search (MES) (Wang and Jegelka 2017), and TAF-ME (Wistuba, Schilling, and Schmidt-Thieme 2018). We also compare against deep learning acquisition functions FSAF (Hsieh, Hsieh, and Liu 2021) and MetaBO (Volpp et al. 2020). We allow the models which can perform meta-learning to pre-train on 250 function samples: 5 trajectories of 50 samples each. We then evaluated all the methods on 36 additional variants, and plot the median regret values for these evaluation runs in Figure 3.

We find that RTDK performs on par with other methods, showing very good performance on the higher dimensional PM2.5 data-set. We do find that baseline methods do perform better in lower dimensional tasks. We think this can be explained by the larger number of parameters and aggressive SAC exploration in RTDK. This may assist in challenging domains, but it can decrease performance in simpler domains. We nevertheless display reasonable performance on these simpler problems even without parameter tuning.

### High Dimensional Continuous Domains

We perform a similar comparison of optimization methods in the continuous domain problems. We evaluate on high dimensional variations of the Thompson Problem (F.R.S. 1904). We parameterize this function in $4N$ dimensions, storing the $sin$ and $cos$ of both the polar and azimuthal angle of each electron on the sphere. We construct the 16, 32, 48, and 64 dimensional variants of the function by taking random $D$-dimensional slices through an overarching $N = 32$

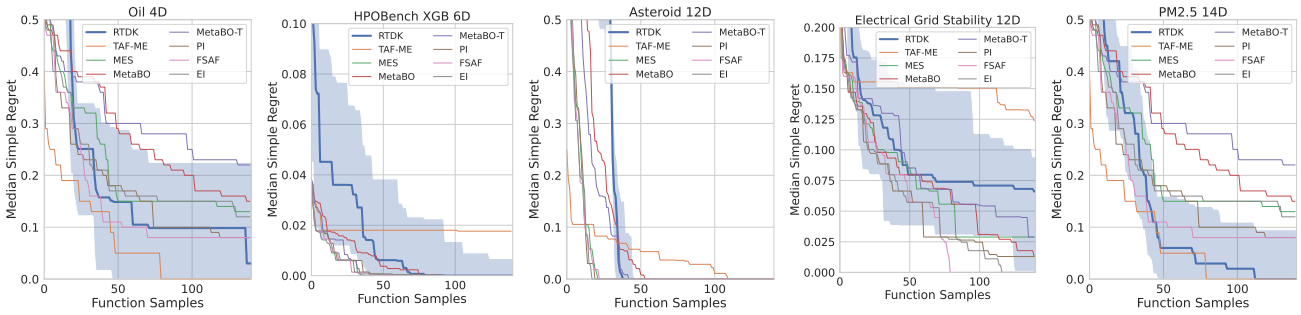



Figure 3: A comparison of different optimization methods on discrete domain hyper-parameter optimization objectives. TDKL results include a shaded region representing the IQR (25% - 75%) of achieved regret values.

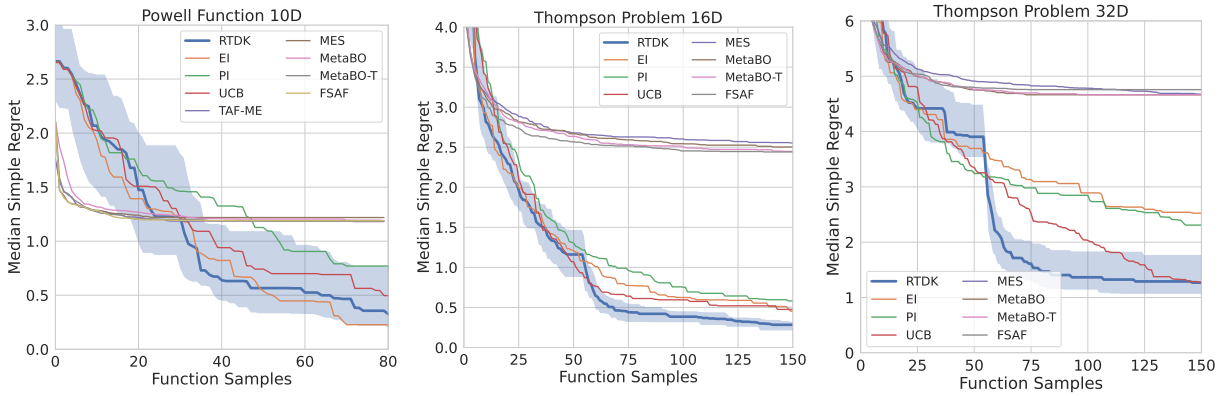



Figure 4: A comparison of different optimization methods on medium dimensional continuous optimization tasks. The final plot shows a comparison of FSAF across different dimensional optimization tasks.

Thompson problem. Each slice is treated as a variant of the function, allowing us to make a near infinite amount of objectives with similar characteristics. To demonstrate our approach on a similar class of problem with multiple modes, we used the 10-dimensional variant of the Powell function and compared the results to the current best performance in (Hsieh, Hsieh, and Liu 2021).

For this test, we continue using the discrete grid-based approach described in (Hsieh, Hsieh, and Liu 2021; Volpp et al. 2020) for baseline methods FSAF, MetaBO, MES, and TAF. This is because generalizing these methods to the continuous domain directly is challenging, and (Hsieh, Hsieh, and Liu 2021) recommends using a discrete grid of quasi-random Sobol samples for approximating continuous optimization. However, other baseline methods - EI, PI, and UCB - may be adapted for use with continuous Gaussian processes and optimization schemes. Therefore, we use a continuous Bayesian optimization routine for these baselines, along with the RTDK model.

Figure 4 presents results on the lower-dimensional continuous optimization tasks, demonstrating the limitations of the grid-based approximation for real-world problems. We find that TDKL consistently appears in the top scorers across these optimization domains, with better separation for the Thompson problem because it features more opportunities for meta-learning.

The combined benefits of the transformer surrogate and SAC acquisition truly shine through on the higher dimensional optimization problems presented in Figure 5. We found that the discrete methods failed to optimize within these high dimensional domains due to the exponential growth in required grid size to densely cover the domain. We also find that RTDK starts to break off from the baseline methods and effectively meta-learns on the 64-dimensional Thompson problem. Moreover, we observe a discrete phase transition between the first 50-sample sub-trajectory and the later sub-trajectories for the RTDK model. Due to the heightened exploration, we find that the first 50 samples are suboptimal for the RTDK method, before rapidly jumping to a better solution once the second sub-trajectory begins. We plan to look into this behavior to allow for a smoother transition, potentially improving sample efficiency.

## Ablation Study

**Sub-Trajectory Length** We find that longer sub-trajectories assist with achieving lower final regret on the 10 Dimensional Powell function (Figure 6(a)). However, this comes with a slight hit to convergence time, taking longer to achieve low regret earlier during the optimization. Additionally, sub-trajectories larger than 50 samples risked running out of memory as the transformer architecture memory requirements scale as $n^2$ with respect to the se-

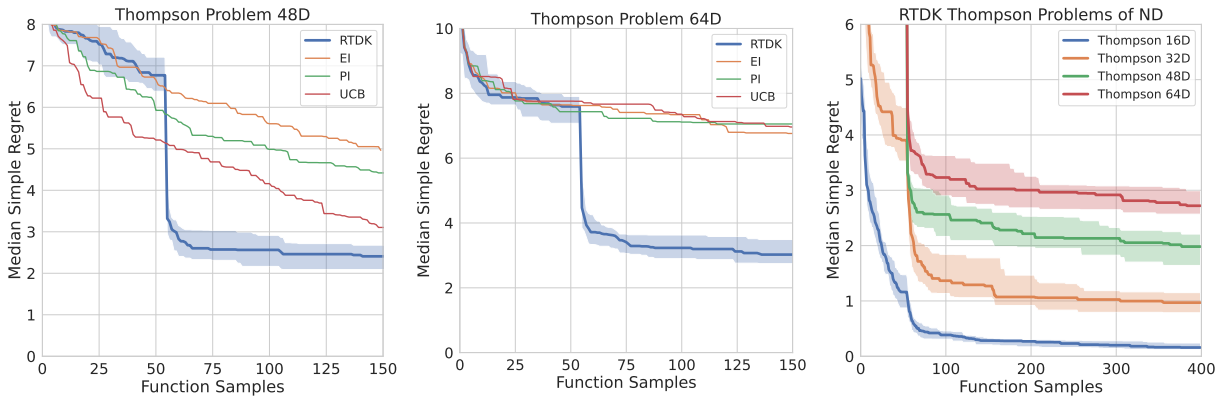
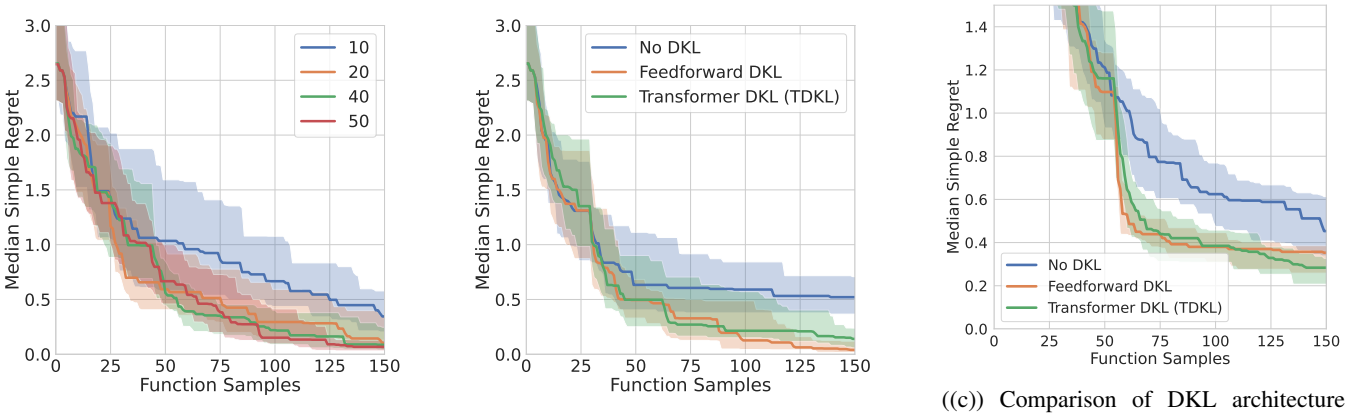



Figure 5: A comparison of different optimization methods on very high dimensional continuous optimization tasks. The final plot shows a comparison of RTDK across different dimensional Thompson optimization runs.




((a)) Comparison of sub-trajectory length on regret in longer episodes

((b)) Comparison of DKL architecture in single-function optimization (Powell 10D)

((c)) Comparison of DKL architecture in meta-learning optimization (Thompson 16D)

Figure 6: Ablation studies on TDKL model, enabling and disabling various components.

quence length. However, we find that performance plateaus after length 30, so these longer trajectories are not necessary for optimization. For the purposes of the experiments, we used a trajectory length of 50 to ensure the lowest possible regret with reasonable memory usage. However, in practice, lower lengths may be used to improve performance time and improve performance in very sample-limited situations.

**Single-Function DKL Architecture** In Figure 6(b), we evaluate the effect of the DKL structure on optimizing the 10-dimensional Powell function. Overall, we find that the deep kernel architecture performs better than a simple Gaussian process. However, in this simpler case, the feed-forward, unconditional DKL performs slightly better. This is likely because it has fewer parameters and a simpler gradient than the transformer architecture.

**Meta-Learning DKL Architecutre** We see better separation when evaluating the three different DKL approaches on the meta-learning task for the 16-dimensional Thompson problem. This objective has more meta-learning because each objective is a projection of a 128-dimensional Thompson problem projected onto a random 96-dimensional plane. This results in a wide variety of possible objective landscapes. We find that the Transformer DKL achieves lower final regret values when compared to the feed-forward approach.

## Conclusion

We present novel contributions to two important aspects of costly black-box Bayesian. We improve the surrogate model through contextual transformer deep kernel learning, extending BO methods to higher dimensional problems such as the 64-dimensional Thompson problem. We design a model-based soft actor-critic to train an acquisition function with reinforcement learning, extending RL Bayesian optimization methods to continuous domains and problems. This two-fold approach could extend deep-learning enhanced Bayesian optimization methods to high dimensional, challenging black-box optimization in the physical sciences and machine learning.

## References


Barnes, C. P.; Silk, D.; Sheng, X.; and Stumpf, M. P. H. 2011. Bayesian design of synthetic biological systems. *Proceedings of the National Academy of Sciences*, 108(37): 15190–15195.

Carr, S.; Garnett, R.; and Lo, C. 2016. BASC: Applying Bayesian Optimization to the Search for Global Minima on Potential Energy Surfaces. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 898–907. New York, New York, USA: PMLR.

Cesa-Bianchi, N.; Gentile, C.; Lugosi, G.; and Neu, G. 2017. Boltzmann Exploration Done Right. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6287–6296. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Frazier, P. I. 2018. A Tutorial on Bayesian Optimization.

F.R.S., J. T. 1904. XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 7(39): 237–265.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870. PMLR.

Hsieh, B.-J.; Hsieh, P.-C.; and Liu, X. 2021. Reinforced Few-Shot Acquisition Function Learning for Bayesian Optimization. In *Conference on Neural Information Processing Systems*.

Jones, D. R.; Schonlau, M.; and Welch, W. J. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4): 455–492.

Kushner, H. J. 1964. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1): 97–106.

LANL, L. A. N. L. 2015. Crystallography. [Online; accessed 5. Nov. 2022].

Liu, T.; Song, Y.; et al. 2022. Stability and Control of Power Grids. *Annu. Rev. Control Rob. Auton. Syst.*, 5(1): 689–716.

López-Ibáñez, M.; Chicano, F.; et al. 2022. The Asteroid Routing Problem: A Benchmark for Expensive Black-Box Permutation Optimization. *arXiv*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Mes. 2022. Why is the Thomson Problem so hard to crack? [Online; accessed 5. Nov. 2022].

Močkus, J. 1975. On bayesian methods for seeking the extremum. In Marchuk, G. I., ed., *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, 400–404. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-37497-8.

Müller, S.; Hollmann, N.; Arango, S. P.; Grabocka, J.; and Hutter, F. 2021. Transformers Can Do Bayesian-Inference By Meta-Learning on Prior-Data. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.

Ober, S. W.; Rasmussen, C. E.; and van der Wilk, M. 2021. The Promises and Pitfalls of Deep Kernel Learning. *arXiv e-prints*, arXiv:2102.12108.

Rasmussen, C. E.; and Williams, C. K. I. 2005. *Gaussian Processes for Machine Learning*. The MIT Press. ISBN 9780262256834.

Shields, B. J.; Stevens, J.; Li, J.; Parasram, M.; Damani, F.; Alvarado, J. I.; Janey, J. M.; Adams, R. P.; and Doyle, A. G. 2021. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844).

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Volpp, M.; Fröhlich, L. P.; Fischer, K.; Doerr, A.; Falkner, S.; Hutter, F.; and Daniel, C. 2020. Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization. In *International Conference on Learning Representations*.

Wang, Z.; and Jegelka, S. 2017. Max-value Entropy Search for Efficient Bayesian Optimization. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3627–3635. PMLR.

Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; and Xing, E. P. 2015. Deep Kernel Learning. *CoRR*, abs/1511.02222.

Wistuba, M.; Schilling, N.; and Schmidt-Thieme, L. 2018. Scalable Gaussian Process-Based Transfer Surrogates for Hyperparameter Optimization. *Mach. Learn.*, 107(1): 43–78.

Zhang, Y.; Apley, D. W.; and Chen, W. 2020. Bayesian Optimization for Materials Design with Mixed Quantitative and Qualitative Variables. *Scientific Reports*, 10(1): 4924.