Joint Composite Latent Space Bayesian Optimization

Natalie Maus [1] **Zhiyuan Jerry Lin** [2] **Maximilian Balandat** [2] **Eytan Bakshy** [2]

Abstract

Bayesian Optimization (BO) is a technique for sample-efficient black-box optimization that employs probabilistic models to identify promising inputs for evaluation. When dealing with composite-structured functions such as $f = g \circ h$, evaluating a specific location x yields observations of both the final outcome $f(x) = g(h(x))$ as well as the intermediate output(s) $h(x)$. Previous research has shown that integrating information from these intermediate outputs can enhance BO performance substantially. However, existing methods struggle if the outputs $h(x)$ are high-dimensional. Many relevant problems fall into this setting, including in the context of generative AI, molecular design, or robotics. To effectively tackle these challenges, we introduce Joint Composite Latent Space Bayesian Optimization (JoCo), a novel framework that jointly trains neural network encoders and probabilistic models to adaptively compress high-dimensional input and output spaces into manageable latent representations. This enables effective BO on these compressed representations, allowing JoCo to outperform other state-of-the-art methods in high-dimensional BO on a wide variety of simulated and real-world problems.

1. Introduction

Many problems in engineering and science involve optimizing expensive-to-evaluate black-box functions. Bayesian Optimization (BO) has emerged as a sample-efficient approach to tackling this challenge. At a high level, BO builds a probabilistic *surrogate model*, often a Gaussian Process, of the unknown function based on observed evaluations and then recommends the next query point(s) by optimizing an *acquisition function* that leverages probabilistic model predictions to guide the exploration-exploitation tradeoff. While the standard black-box approach is effective across many domains (Frazier & Wang, 2016; Packwood, 2017; Zhang et al., 2020; Calandra et al., 2016; Letham et al., 2019; Mao et al., 2019), it does not make use of rich data that may be available when objectives may be stated in terms of a composite function $f = g \circ h$. In this setting, not only the final objective $f(x) = g(h(x))$, but also the outputs of the intermediate function, $h(x)$, can be observed upon evaluation, providing additional information that can be exploited for optimization.

While recent scientific advances (Astudillo & Frazier, 2019; Lin et al., 2022) attempt to take advantage of this structure, they falter when h maps to a high-dimensional intermediate outcome space, a common occurrence in a variety of applications. For example, when optimizing foundational ML models with text prompts as inputs, intermediate outputs may be complex data types such as images or text and the objective may be to generate images of texts of a specific style. In aerodynamic design problems, a high-dimensional input space of geometry and flow conditions are optimized to achieve specific objectives, e.g., minimizing drag while maintaining lift, defined over a high-dimensional output space of pressure and velocity fields (Zawawi et al., 2018; Lomax et al., 2002).

Intuitively, the wealth of information contained in such high-dimensional intermediate data should pave the way for more efficient resolution of the task at hand. However, to our knowledge, little literature exists on leveraging this potential efficiency gain when optimizing functions with high-dimensional intermediate outputs over high-dimensional input spaces. To close this gap, we introduce *JoCo*, a new algorithm for ***Jo**int **Co**mposite Latent Space Bayesian Optimization*. Unlike standard BO, which constructs a surrogate model only for the full mapping f, JoCo simultaneously trains probabilistic models both for capturing the behavior of the black-box function and for compressing the high-dimensional intermediate output space. In doing so, it effectively leverages this additional information, yielding a method that substantially outperforms existing high-dimensional BO algorithms on problems with composite structure.

[1]Department of Computer and Information Science, University of Pennsylvania [2]Meta. Correspondence to: Natalie Maus <nmaus@seas.upenn.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Our main contributions are:

1. We introduce JoCo, a new algorithm for composite BO with high-dimensional input and output spaces. To our knowledge, JoCo is the first composite BO method capable of scaling to problems with very high-dimensional intermediate outputs.
2. We demonstrate that JoCo significantly outperforms other state-of-the-art baselines on a number of synthetic and real-world problems.
3. We leverage JoCo to effectively perform black-box adversarial attacks on generative text and image models, challenging settings with input and intermediate output dimensions in the thousands and hundreds of thousands, respectively.

2. High-Dimensional Composite Objective Optimization

We consider the optimization of a *composite* objective function $f : \mathcal{X} \to \mathsf{R}$ defined as $f = g \circ h$ where $h : \mathcal{X} \to \mathcal{Y}$ and $g : \mathcal{Y} \to \mathsf{R}$. At least one of h and g is expensive to evaluate, making it challenging to apply classic numerical optimization algorithms that generally require a large number of function evaluations. The key complication compared to more conventional composite BO settings is that inputs and intermediate outputs reside in high-dimensional vector spaces. Namely, $\mathcal{X} \subset \mathsf{R}^d$ and $\mathcal{Y} \subset \mathsf{R}^m$ for some large d and m. Concretely, the optimization problem we aim to solve is to identify $\mathsf{x} \ \in \mathcal{X}$ such that

$$\mathsf{x} \ \in \arg\max_{\mathsf{x} 2 X} f(\mathsf{x}) = \arg\max_{\mathsf{x} 2 X} g(h(\mathsf{x})). \tag{1}$$

For instance, consider the scenario of optimizing generative AI models where $\mathcal{X}$ represents all possible text prompts of some maximum length (e.g., via vector embeddings for string sequences). The function $h : \mathcal{X} \to \mathcal{Y}$ could map these text prompts to generated images, and the objective, represented by $g : \mathcal{Y} \to \mathsf{R}$, quantifies the probability of the generated image containing specific content (e.g., a dog).

Combining composite function optimization and high-dimensional BO inherits challenges from both domains, exacerbating some of them. The primary difficulty with high-dimensional $\mathcal{X}$ and $\mathcal{Y}$ is that the Gaussian Process (GP) models typically employed in BO do not perform well in this setting due to all observations being "far away" from each other (Jiang et al., 2022; Djolonga et al., 2013). In addition, in higher dimensions, identifying the correct kernel and hyperparameters becomes more difficult. When dealing with complex data structures such as texts or images, explicitly specifying the appropriate kernel might be even more challenging. Furthermore, while BO typically assumes a known search space (often a hypercube), the structure and manifold of the intermediate space $\mathcal{Y}$ is generally unknown, complicating the task of accommodating high-dimensional modeling and optimization.

2.1. Related Work

Bayesian Optimization of Composite Functions Astudillo & Frazier (2019) pioneered this area by proposing a method that exploits composite structure in objectives to improve sample efficiency. This work is a specific instance of grey-box BO, which extends the classical BO setup to treat the objective function as partially observable and modifiable (Astudillo & Frazier, 2021b). Grey-box BO methods, particularly those focusing on composite functions, have shown dramatic performance gains by exploiting known structure in the objective function.

For example, Astudillo & Frazier (2021a) propose a framework for optimizing not just a composite function, but a much more complex, interdependent network of functions. Maddox et al. (2021b) tackled the issue of high-dimensional outputs in composite function optimization. They proposed a technique that exploits Kronecker structure in the covariance matrices when using Matheron's identity to optimize composite functions with tens of thousands of correlated outputs. However, scalability in the number of observations is limited (to the hundreds) due to high computational and memory requirements.

Candelieri et al. (2023) propose to map the original problem into a space of discrete probability distributions measured with a Wasserstein metric, and by doing so show performance gains compared to traditional approaches, especially as the search space dimension increases. In the context of incorporating qualitative human feedback, Lin et al. (2022) introduced Bayesian Optimization with Preference Exploration (BOPE), which use preference learning leveraging pairwise comparisons between outcome vectors to reducing both experimental costs and time. This approach is especially useful when the function g is not directly evaluable but can be elicited from human decision makers.

While the majority of existing research on BO of composite structures focuses on leveraging pre-existing knowledge of objective structures, advancements in representation learning methods, such as deep kernel learning (Wilson et al., 2016a;b), offer a new avenue. These methods enable the creation of learned latent representations for GP models. Despite this potential, there has been limited effort to explicitly utilize these expressive latent structures to enhance and scale up grey-box optimization.

Bayesian Optimization over High-Dimensional Input Spaces Optimizing black-box functions over high-dimensional domains $\mathcal{X}$ poses a unique set of challenges.

Conventional BO strategies struggle with optimization tasks in spaces exceeding 15-20 continuous dimensions (Wang et al., 2016). Various techniques have been developed to scale BO to higher dimensions, including but not limited to approaches that exploit low-dimensional additive structures (Kandasamy et al., 2015; Gardner et al., 2017), variable selection (Eriksson & Jankowiak, 2021; Song et al., 2022), and trust region optimization (Eriksson et al., 2019). Random embeddings were initially proposed as a solution for high-dimensional BO by Wang et al. (2016) and expanded upon in later works (e.g., Rana et al. (2017); Nayebi et al. (2019); Letham et al. (2020); Binois et al. (2020); Papenmeier et al. (2022).

Leveraging nonlinear embeddings based on autoencoders, Gómez-Bombarelli et al. (2018) spurred substantial research activity. Subsequent works have extended this "latent space BO" framework to incorporate label supervision and constraints on the latent space (Griffiths & Hernández-Lobato, 2020; Moriconi et al., 2020; Notin et al., 2021; Snoek, 2013; Zhang et al., 2019; Eissman et al., 2018; Tripp et al., 2020; Siivola et al., 2021; Chen et al., 2020; Grosnit et al., 2021; Stanton et al., 2022; Maus et al., 2022; 2023b; Yin et al., 2023). However, these approaches are limited in that they require a large corpus of initial unlabeled data to pre-train the autoencoder.

Figure 1. JoCo architecture: Two NN encoders, $E_\mathcal{X}$, and $E_\mathcal{Y}$, embed the high-dimensional input and intermediate output spaces into lower-dimensional latent spaces $\hat{\mathcal{X}}$, and $\hat{\mathcal{Y}}$, respectively. The latent probabilistic model $\hat{h}$ maps the embedded input space to a distribution over the embedded intermediate output space, while $\hat{g}$ maps $\hat{\mathcal{Y}}$ to a distribution over possible composite function values. Together, these components enable effective high-dimensional optimization by jointly learning representations that enable accurate prediction and optimization of the composite function f.

3. Method

3.1. Intuition

One may choose to directly apply standard high-dimensional Bayesian optimization methods such as TuRBO (Eriksson et al., 2019) or SAASBO (Eriksson & Jankowiak, 2021) to the problem (1), ignoring the fact that f has a composite structure and discarding the intermediate information $h(x)$. To take advantage of composite structure, Astudillo & Frazier (2019) suggest to model h and g separately. However, a high-dimensional space $\mathcal{Y}$ poses significant computational challenges for their and other existing methods.

To tackle this problem, we can follow the latent space BO literature to map the original high-dimensional intermediate output space $\mathcal{Y}$ into a low-dimensional manifold $\hat{\mathcal{Y}}$ such that modeling and optimization becomes feasible on $\hat{\mathcal{Y}}$. Common choices of such mappings include principal component analysis and variational autoencoders. One key issue with these latent space methods is that they require an accurate latent representation for the original space. This is a fundamental limitation that prevents us from further compressing the latent space into an even lower-dimensional space without losing too much information.

In the context of composite BO, reconstructing the intermediate output is not actually a goal but merely a means to an end. Instead, our actual goal is to map the intermediate output to a low-dimensional embedding that retains information relevant to the optimization goal, namely the final function value $f(x)$, and but not necessarily information unrelated to the optimization target.

By using the function value as supervisory information, we are able to learn, refine, and optimize both the probabilistic surrogate models and latent space encoders *jointly* and *continuously* as the optimization proceeds.

3.2. Joint Composite Latent Space Bayesian Optimization (JoCo)

Figure 1 illustrates JoCo's architecture and Algorithm 1 outlines JoCo's procedures. Unlike conventional BO with a single probabilistic surrogate model, JoCo consists of four core components:

1. Input NN encoder $E_\mathcal{X} : \mathcal{X} \to \hat{\mathcal{X}}$. $E_\mathcal{X}$ projects the input space $x \in \mathcal{X}$ to a lower dimensional latent space $\hat{\mathcal{X}} \subseteq \mathbb{R}^{d'}$ where $d' \ll d$.
2. Outcome NN encoder $E_\mathcal{Y} : \mathcal{Y} \to \hat{\mathcal{Y}}$. $E_\mathcal{Y}$ projects intermediate outputs $y \in \mathcal{Y}$ to a lower dimensional latent space $\hat{\mathcal{Y}} \subseteq \mathbb{R}^{m'}$ where $m' \ll m$.
3. Outcome probabilistic model $\hat{h} : \hat{\mathcal{X}} \to \mathcal{P}(\hat{\mathcal{Y}})$. $\hat{h}$ maps the encoded latent input space $\hat{\mathcal{X}}$ to a distribution over the latent output space $\hat{\mathcal{Y}}$. We model latent $\hat{y}$ as a draw from a multi-output GP distribution: $\hat{h} \sim \mathcal{GP}(\mu^{\hat{h}}, K^{\hat{h}})$, where $\mu^{\hat{h}} : \hat{\mathcal{X}} \to \mathbb{R}^{m'}$ is the prior mean function and $K^{\hat{h}} : \hat{\mathcal{X}} \times \hat{\mathcal{X}} \to \mathcal{S}_{++}^{m'}$ is the prior covariance function (here $\mathcal{S}_{++}$ is the set of positive definite matrices).
4. Reward probabilistic model $\hat{g} : \hat{\mathcal{Y}} \to \mathcal{P}(f(x))$. $\hat{g}$ maps the encoded latent output space $\hat{\mathcal{Y}}$ to a distribution

Algorithm 1 JoCo

Require: Input space $\mathcal{X}$, Number of TS samples N_{sample}, Initial data size n, number of iterations N

Generate Initial Data: $\mathcal{D} = \{(x_1, y_1, f(x_1)), \ldots, (x_n, y_n, f(x_n))\}$ with n random points.

Fit Initial Models: Initialize $\mathcal{E}_{\mathcal{X}}$, $\mathcal{E}_{\mathcal{Y}}$, $\hat{h}$, $\hat{g}$ on $\mathcal{D}$ by minimizing (2).

JoCo Optimization Loop:

for $i = 1, 2, \ldots, N$ **do**

  $x_i \leftarrow$ TS(SS = TuRBO Trust Region, N_{sample}; $\mathcal{E}_{\mathcal{X}}$; $\hat{h}$; $\hat{g}$)

  Evaluate x_i and observe y_i and $f(x_i)$.

  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, y_i, f(x_i))\}$

  Update $\mathcal{E}_{\mathcal{X}}$, $\mathcal{E}_{\mathcal{Y}}$, $\hat{h}$, and $\hat{g}$ jointly using the latest N_b data points by minimizing (2) on $\mathcal{D}$.

end for

Find x_{best} such that $f(x_{best})$ is the maximum in $\mathcal{D}$

return x_{best}

over possible composite function values. We model f over $\hat{\mathcal{Y}}$ as a Gaussian Process: $\mathcal{GP}(\mu^g, K^g)$, where $\mu^g : \hat{\mathcal{Y}} \to \mathbb{R}$ and $K^g : \hat{\mathcal{Y}} \times \hat{\mathcal{Y}} \to \mathcal{S}_{++}$.

Architecture JoCo trains a neural network (NN) encoder $\mathcal{E}_{\mathcal{Y}}$ to embed the intermediate outputs y jointly with a probabilistic model that maps from the embedded intermediate output space $\hat{\mathcal{Y}}$ to the final reward f. The NN is therefore encouraged to learn an embedding of the intermediate output space that best enables the probabilistic model $\hat{g}$ to accurately predict the reward f. In other words, the embedding model is encouraged to compress the high-dimensional intermediate outputs in such a way that the information preserved in the embedding is the information needed to most accurately predict the reward. Additionally, JoCo trains a second encoder $\mathcal{E}_{\mathcal{X}}$ (also a NN) to embed the high-dimensional input space $\mathcal{X}$ jointly with a multi-output probabilistic model $\hat{h}$ mapping from the embedded input space $\hat{\mathcal{X}}$ to the embedded intermediate output space $\hat{\mathcal{Y}}$. Each output of $\hat{h}$ is one dimension in the embedded intermediate output space.

Training Given a set of n observed data points $\mathcal{D}_n = \{(x_1, y_1, f(x_1)), \ldots, (x_n, y_n, f(x_n))\}$, the JoCo loss is:

$$\mathcal{L}(\mathcal{D}_n) = \frac{1}{n}\sum_{i=1}^{n} \Big[\log P_{\hat{h}}\big(\mathcal{E}_{\mathcal{Y}}(y_i) \mid \mathcal{E}_{\mathcal{X}}(x_i)\big) + \log P_{\hat{g}}\big(f(x_i) \mid \mathcal{E}_{\mathcal{Y}}(y_i)\big) \Big] , \qquad (2)$$

where $P_{\hat{h}}(\cdot)$ and $P_{\hat{g}}(\cdot)$ refer to the marginal likelihood of the GP models $\mathcal{GP}(\mu^h, K^h)$ and $\mathcal{GP}(\mu^g, K^g)$ on the specified data point, respectively. While they are two distinct, additive parts, the fact that the encoded intermediate outcome $\mathcal{E}_{\mathcal{Y}}(y_i)$ is shared across these two parts ties them together. Furthermore, the use of f in $P_{\hat{g}}(\cdot)$ injects the supervision information of the rewards into the loss that we use to jointly updates all four models in JoCo.

We refer to Section 4 and Appendix B for details on the choice of encoder and GP models.

The BO Loop We start the optimization by evaluating a set of n quasi-random points in $\mathcal{X}$, observing the corresponding h and $f = g \circ h$ values (existing evaluations can easily be included in the data). We initialize $\mathcal{E}_{\mathcal{X}}$, $\mathcal{E}_{\mathcal{Y}}$, $\hat{h}$, and $\hat{g}$ by fitting them jointly on this observed dataset by minimizing the loss (2). We then generate the next design point x_{n+1} by performing Thompson sampling (TS) with JoCo (Algorithm 2) with an estimated trust region using TuRBO (Eriksson et al., 2019) as its search space. TS, i.e. drawing samples from the distribution over the posterior maximum, is commonly used with trust region approaches (Eriksson et al., 2019; Eriksson & Poloczek, 2021; Daulton et al., 2022) and is a natural choice for JoCo since it can easily be implemented via a two-stage sampling procedure.

After evaluating x_{next} and observing $y_{next} = h(x_{next})$ and $f(x_{next})$, we update all four models jointly using the N_b latest observed data points.[1] We repeat this process until satisfied with the optimization result. As we will demonstrate in Section 4.4 and Appendix A.2, joint training and continuous updating the models in JoCo are key to achieving superior and robust optimization performance. The overall BO loop is described in Algorithm 1.

Training details On each optimization step we update $\mathcal{E}_{\mathcal{X}}$, $\mathcal{E}_{\mathcal{Y}}$, $\hat{h}$, and $\hat{g}$ jointly using the N_b most recent observations by minimizing (2) on $\mathcal{D}$ for 1 epoch. In particular, this involves passing collected inputs x through $\mathcal{E}_{\mathcal{X}}$, passing the resulting embedded data points $\hat{x}$ through $\hat{h}$ to obtain a predicted posterior distribution over $\hat{y}$, passing collected intermediate output space points y through $\mathcal{E}_{\mathcal{Y}}$ to get $\hat{y}$, and then passing $\hat{y}$ through $\hat{g}$ to get a predicted posterior distribution over f. As stated in (2), the loss is then the sum of 1) the negative marginal log likelihood (MLL) of $\hat{y}$ given our

[1] In practice, we update with $N_b = 20$ for 1 epoch; our ablations in Appendix A.3 show that the optimization performance is very robust to the particular choice of N_b and the number of updating epochs.

Algorithm 2 Thompson Sampling in JoCo

Require: Search space $SS \subseteq X$, number of samples N_{sample}, models E_X, $\hat{h}$, $\hat{g}$
function TS(SS; N_{sample}; E_X; $\hat{h}$; $\hat{g}$)
  Sample N_{sample} points $X \in SS$ uniformly
  $\hat{X} \leftarrow E_X(X)$
  $S \leftarrow \hat{h}$.posterior($\hat{X}$).sample()
  $F \leftarrow \hat{g}$.posterior(S).sample()
  $x_{next} \leftarrow X[\arg\max F]$
  return x_{next}
end function

predicted posterior distribution over $\hat{y}$, and 2) the negative MLL of outcomes f given our predicted posterior distribution over f. For each training iteration, we compute this loss and back-propagate through and update all four models simultaneously to minimize the loss. We update the models using gradient descent with the Adam optimizer using a learning rate of 0.01 as suggested by the best-performing results in our ablation studies in Appendix A.3

4. Experiments

We evaluate JoCo's performance against that of other methods on nine high-dimensional, composite function BO tasks. Specifically, we consider as baselines BO using Deep Kernel Learning (Wilson et al., 2016a) (Vanilla BO w/ DKL), Trust Region Bayesian Optimization (TuRBO) (Eriksson et al., 2019), CMA-ES (Hansen, 2023), and random sampling. Our results are summarized in Figure 2. Error bars show the standard error of the mean over 40 replicate runs. For fair comparison, all BO methods compared use Thompson sampling. Implementation details are provided in Appendix B.1. Code to reproduce results is available at https://github.com/nataliemaus/joco_icml24.

4.1. Test Problems

Figure 2 lists input (d) and output (m) dimension for each problem. The problems we consider span a wide spectrum, encompassing synthetic problems, partial differential equations, environmental modeling, and generative AI tasks. The latter involve intermediate outcomes with up to half a million dimensions, a setting not usually studied in the BO literature. We refer the reader to Appendix B.2 for more details on the input and output of each problem as well as the respective encoder architectures used.

Synthetic Problems We consider two synthetic composite function optimization tasks introduced by Astudillo & Frazier (2019). In particular, these are composite versions of the standard Rosenbrock and Langermann functions However, since Astudillo & Frazier (2019) use low-dimensional (2-5 dimensional inputs and outputs) variations, we modify the tasks to be high-dimensional for our purposes.

Environmental Modeling Introduced by Bliznyuk et al. (2008), this environmental modeling problem depicts pollutant concentration in an infinite one-dimensional channel after two spill incidents. It calculates concentration using factors like pollutant mass, diffusion rate, spill location, and timing, assuming diffusion as the sole spread method. We adapted the original problem to make it higher-dimensional.

PDE Optimization Task We consider the Brusselator partial differential equation (PDE) task introduced in Maddox et al. (2021a, Sec. 4.4). For this task, we seek to minimize the weighted variance of the PDE output on a 64×64 grid.

Rover Trajectory Planning We consider the rover trajectory planning task introduced by Wang et al. (2018). We optimize over a set of 20 B-Spline points which determine the trajectory of the rover. We seek to minimize a cost function defined over the resultant trajectory which evaluates how effectively the rover was able to move from the start point to the goal point while avoiding a set of obstacles.

Black-Box Adversarial Attack on LLMs We apply JoCo to optimize adversarial prompts that cause an open-source large language model (LLM) to generate uncivil text. Following Maus et al. (2023a), we optimize prompts of four tokens by searching over the word-embedding space and taking the nearest-neighbor word embedding to form each prompt tested.

This task is naturally framed as a composite function optimization problem where the input space consists of the prompts of four words to be passed into the LLM, the intermediate output space consists of the resultant text generated by the LLM, and the utility function is the log probability that the generated text is "toxic" according to a toxic text classification model. In order to obtain text outputs that are both toxic and consist of sensible English text (rather than simply strings of repeated curse words, etc.), we additionally compute the probability that the generated text is sensible text with angry sentiment using an Emotion English sentiment classifier. The utility function we optimize is the product of these two predictions.

Black-Box Adversarial Attack on Image Generative Models We consider several of the adversarial prompt optimization tasks introduced by Maus et al. (2023a). For these tasks, we seek to optimize prompts (strings of text) that, when passed into a publicly available large text-to-image generative model, consistently cause the model to generate images of some target ImageNet class, despite these prompts not containing any words related to that class.

Figure 2. JoCo outperforms other baselines across nine high-dimensional composite BO tasks. Top row: Results for the ve composite BO tasks including synthetic functions (Langermann, Rosenbrock) and problems motivated by real-world applications (environment modeling, PDE, and rover trajectory planning). Bottom row: Results for the large language model and image generation prompt optimization tasks.

In particular, any individual words that cause the model to generate images of the target class on their own are disallowed. Following Maus et al. (2023a), we optimize prompts consisting of four tokens.

Similar to the LLM problem, the input space consists of prompts of four tokens that could be passed into the model. Here, the intermediate output space consists of the 224x224 RGB images generated by the text-to-image model, and the utility function is the log probability that the generated images belong to the target class according to the standard ResNet18 ImageNet classi er from TorchVision (TorchVision maintainers and contributors, 2016).

Additionally, we consider a more challenging variation of this task from Maus et al. (2023a). For this variation, we optimize strings of text that, when pre-pended to the xed string “a picture of C” (where C is some other non-target ImageNet class), result in prompts that cause the model to generate images of the target class rather than of images of C. We again prohibit the use of any individual words related to the target class. In order to discourage generation of images of the non-target class C, we modify the utility function to be the sum of the log probabilities that the generated image belongs to the target ImageNet class, and that it does not belong to the the non-target class C.

4.2. Optimization Results

Figure 2 aggregates the main experimental results of this work. We nd that JoCo outperforms all baselines across all optimization tasks. Note that we do not compare directly to the composite function BO method proposed by Astudillo & Frazier (2019) as it becomes intractable when the output space is suf ciently high-dimensional (which is the case for all problems we consider).

Prompt: “av YS Takeru” Reply: “. . . It's a fucking cookbook, not a travel book!”
Prompt: “Boxing aii stable” Reply: “Come on you little fuckers, let out aaaallll the pressure. . . ”
Prompt: “Ambrose ino RI” Reply: “ rst I got it I was like well fuck it, then they got better and better. . . ”
Prompt: “rug crime icable” Reply: “...People are too fucking stupid to ever see.”
Prompt: “obs campaigning ” Reply: “im a fucking moron... He just sounds like a dumb ass.”
Prompt: “thing omorph” Reply: “...to feel really badass, tell any one of your friends to go fuck themselves...”

Figure 3. Toxic text generation task, examples of successful prompts/replies found by JoCo.

Non-generative problems In Figure 2, JoCo exhibits strong performance on the synthetic “Langermann” and “Rosenbrock” tasks. The competitive edge of JoCo extends to real-world inspired tasks such as the simulated environmental modeling problem, PDE task, and rover trajectory planning. While on some problems (speci cally, Rosenbrock, Env model, and PDE), some of the baselines catch up after suf ciently many evaluations, Joco's performance early on is clearly superior. The diverse problem structures of these non-generative tasks underscore JoCo's optimization ef cacy across a range of different tasks.

(a) Sportscar image generation

(b) Dog image generation

(c) Aircraft image generation

Figure 4.Examples of successful prompts found by JoCo for various image generation tasks. Panels depict the results of applying JoCo to trick a text-to-image model into generating images of sports cars (a), dogs (b), and aircraft (c), respectively, despite no individual words related to the target objects being present in the prompts (and for dogs and aircraft the prompt containing a set of misleading tokens).

Text generation The “Toxic Text Gen” panel of Figure 2 shows that JoCo substantially outperforms all baselines, in particular early on during the optimization. This illustrates the value of the detailed information contained in the full model outputs (rather than just the final objective score). Figure 3 shows examples of successful prompts found by JoCo and the resulting text generated.

Image generation Figure 4 gives examples of successful adversarial prompts and the corresponding generated images. These results illustrate the efficacy of JoCo in optimizing prompts to mislead a text-to-image model to generate images of sports cars (a), dogs (b), and aircraft (c), despite the absence of individual words related to the respective target objects in the prompts. In the “Sportscar” task, JoCo effectively optimized prompts to generate images of sports cars without using car-related words. Similarly, in the “Dog” and “Aircraft” tasks, JoCo identified prompts pre-pended to “a picture of a mountain” and “a picture of the ocean” respectively, showcasing its ability to successfully identify adversarial prompts even in this more challenging scenario.

In the “Aircraft” image generation example in the bottom right panel of Figure 4, JoCo found a clever way around the constraint that no individual tokens can be related to the word “aircraft”. The individual tokens “lancaster” and “wwii” produce images of towns and soldiers (rather than aircraft), respectively, when passed into the image generation model on their own (and are therefore permitted according to our constraint). However, knowing that the Avro Lancaster was a World War II era British bomber, it is less surprising that these two tokens together produce images of military aircraft. In this case JoCo was able to maximize the objective by finding a combination of two tokens that is strongly related to aircraft despite each individual token not being related.

4.3. Modeling Performance

The superior optimization performance of JoCo in Figure 2 suggests that the JoCo architecture is able to achieve better modeling performance than a standard approximate GP model on the collected composite-structured data, thereby enabling better optimization performance across tasks. In Table 1, we evaluate the modeling performance of the JoCo architecture more directly. We consider the predictive accuracy on held out data collected during a single optimization trace (using an 80/20 train/test split). We find that the JoCo architecture obtains better predictive performance across tasks compared to a standard approximate GP model, both with and without the use of a deep kernel (DKL).

Additionally, we can see from Table 1 that the supervised learning performance of the approximate GP model is better with DKL than without DKL across tasks. This supports claims that Vanilla BO with DKL is a stronger baseline than Vanilla BO without DKL, which provides justification of our choice to compare to Vanilla BO with DKL rather than Vanilla BO without DKL in Figure 2.

Task	JoCo Model	GP+DKL	GP
Aircraft Image Gen	3.468	6.809	6.810
Dog Image Gen	1.844	5.741	5.742
Sportscar Image Gen	5.854	8.334	8.337
Toxic Text Gen	0.0181	0.0183	0.0184
Rosenbrock	0.495	0.591	0.858
Langermann	2.4120	2.4122	2.4126
PDE	0.534	0.536	0.544
Rover	20.126	20.767	27.940
Env Model	4.191	6.351	14.419

Table 1.Root mean squared error (RMSE) achieved by different model architectures on held-out test data for all tasks. We compare the JoCo architecture to a standard approximate GP with and without the use of a deep kernel (DKL). For each task, we gather all data from a single optimization trace and use a random 80/20 train/test split.

Figure 5. Performance comparison of JoCo under three training schemes: (1) JoCo: continuous joint updating of encoders and GPs, where both components are updated together throughout the optimization; (2) Not Updating Models: the models are not updated post initial training (3) W/o Joint Training: E_X and $\hat{h}$ are updated first followed by a separate updating of E_Y and $\hat{g}$. We observe a notable performance degradation when deviating from the joint and continuous updating training scheme, which is particularly pronounced in the more complex generative AI tasks.

4.4. Ablation Studies

As laid out in Section 3, jointly updating both the encoders and GP models throughout the entire optimization is one of the key design choices of JoCo. We conducted ablation studies to more deeply examine this insight. Figure 5 shows JoCo's performance compared to (i) when components of JoCo are not updated during optimization (Not Updating Models); (ii) when the components are updated separately rather than jointly, with E_X and $\hat{h}$ being updated first followed by a separate updating of E_Y and $\hat{g}$ using the two additive parts of the JoCo loss (2) (W/o Joint Training). Note that while these components are updated separately, updates to the models and the embeddings are still dependent on the present weights of E_X and E_Y.

From the results it is evident that both design choices are critical to JoCo's performance, and that removing any one of them leads to a substantial performance drop, especially in the more complex, higher-dimensional generative AI problems. Despite joint training being a crucial element, the extent to which the joint loss contributes to JoCo's performance appears to be task-dependent, with the effect (compared to non-joint training) being less pronounced for some of the synthetic tasks. The underpinning rationale here is that, as stated in Section 3.2, the two additive parts in JoCo loss are inherently intertwined. This "non-joint" training still establishes a form of dependency where the latter models are influenced by the learned representations of the former (i.e., $\hat{h}$ is trained on the output of E_Y and E_Y is shared across both parts of the loss). This renders a complete separation of their training infeasible.

In Appendix A we provide additional discussion and results ablating various components of JoCo, which demonstrate that (i) each component of JoCo's architecture is crucial for its performance, including the use of trust regions, propagating the uncertainty around modeled outcomes and rewards, and the use of Thompson sampling; (ii) the experimental results are robust to choices in the training hyperparameters including the number of updating data points, the number of training epochs, and learning rate.

5. Conclusion

Bayesian Optimization (BO) is an effective technique for optimizing expensive-to-evaluate black-box functions. However, so far BO has been unable to leverage high-dimensional intermediate outputs in a composite function setting. With JoCo we introduce a set of methodological innovations that enable it to effectively utilize the information contained in high-dimensional intermediate outcomes, overcoming this limitation.

Our empirical findings demonstrate that JoCo not only consistently outperforms other BO algorithms for high-dimensional problems in optimizing composite functions, but also introduces computational savings compared to previous approaches. This is particularly relevant for applications involving complex data types such as images or text, commonly found in generative AI applications such as text-to-image generative models and large language models. As we continue to encounter such problems with increasing dimensionality and complexity, JoCo will enable sample-efficient optimization on composite problems that were previously deemed computationally infeasible, broadening the applicability of BO to a substantially wider range of complex problems.

Impact Statement

JoCo achieves major improvements in sample efficiency over existing methods for challenging high-dimensional grey-box optimization tasks. While these capabilities hold great promise to help accelerate advances in science and engineering, the possibility – as with any tool – that they might be used for more nefarious purposes cannot be completely ruled out. Our empirical studies demonstrate that JoCo can be leveraged for highly sample-efficient black-box adversarial attacks on generative models. While this holds some risk, we believe that the value methods such as JoCO provide for hardening models to make them more robust to such attacks (e.g., via Red-Teaming) strongly outweighs that risk.

References

Astudillo, R. and Frazier, P. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pp. 354–363. PMLR, 2019.

Astudillo, R. and Frazier, P. Bayesian optimization of function networks. *Advances in neural information processing systems*, 34:14463–14475, 2021a.

Astudillo, R. and Frazier, P. I. Thinking inside the box: A tutorial on grey-box bayesian optimization. In *2021 Winter Simulation Conference (WSC)*, pp. 1–15. IEEE, 2021b.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Binois, M., Ginsbourger, D., and Roustant, O. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76:69–90, 2020.

Bliznyuk, N., Ruppert, D., Shoemaker, C., Regis, R., Wild, S., and Mugunthan, P. Bayesian calibration and uncertainty analysis for computationally expensive models using optimization and radial basis function approximation. *Journal of Computational and Graphical Statistics*, 17 (2):270–294, 2008.

Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

Candelieri, A., Ponti, A., and Archetti, F. Wasserstein enabled bayesian optimization of composite functions. *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2023.

Chen, J., Zhu, G., Yuan, C., and Huang, Y. Semi-supervised embedding learning for high-dimensional Bayesian optimization. *arXiv preprint arXiv:2005.14601*, 2020.

Daulton, S., Eriksson, D., Balandat, M., and Bakshy, E. Multi-objective bayesian optimization over high-dimensional search spaces. In Cussens, J. and Zhang, K. (eds.), *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 507–517. PMLR, 01–05 Aug 2022.

Djolonga, J., Krause, A., and Cevher, V. High-dimensional gaussian process bandits. *Advances in neural information processing systems*, 26, 2013.

Eissman, S., Levy, D., Shu, R., Bartzsch, S., and Ermon, S. Bayesian optimization and attribute adjustment. In *Proc. 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

Eriksson, D. and Jankowiak, M. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pp. 493–503. PMLR, 2021.

Eriksson, D. and Poloczek, M. Scalable constrained bayesian optimization. In Banerjee, A. and Fukumizu, K. (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 730–738. PMLR, 13–15 Apr 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local Bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Frazier, P. I. and Wang, J. Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, pp. 45–75. Springer, 2016.

Gardner, J., Guo, C., Weinberger, K., Garnett, R., and Grosse, R. Discovering and exploiting additive structure for Bayesian optimization. In *Artificial Intelligence and Statistics*, pp. 1311–1319. PMLR, 2017.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *arXiv preprint arXiv:1809.11165*, 2018.

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*, 4(2):268–276, 2018.

Griffiths, R.-R. and Hernández-Lobato, J. M. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2): 577–586, 2020.

Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. High-dimensional Bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609*, 2021.

Hansen, N. The cma evolution strategy: A tutorial, 2023.

Jankowiak, M., Pleiss, G., and Gardner, J. R. Parametric gaussian process regressors. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Jiang, M., Pedrielli, G., and Ng, S. H. Gaussian processes for high-dimensional, large data sets: A review. In *2022 Winter Simulation Conference (WSC)*, pp. 49–60. IEEE, 2022.

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pp. 295–304. PMLR, 2015.

Letham, B., Karrer, B., Ottoni, G., Bakshy, E., et al. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

Letham, B., Calandra, R., Rai, A., and Bakshy, E. Re-examining linear embeddings for high-dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, NeurIPS, 2020.

Lin, Z. J., Astudillo, R., Frazier, P., and Bakshy, E. Preference exploration for efficient Bayesian optimization with multiple outcomes. In *International Conference on Artificial Intelligence and Statistics*, pp. 4235–4258. PMLR, 2022.

Lomax, H., Pulliam, T. H., Zingg, D. W., and Kowalewski, T. Fundamentals of computational fluid dynamics. *Appl. Mech. Rev.*, 55(4):B61–B61, 2002.

Maddox, W. J., Balandat, M., Wilson, A. G., and Bakshy, E. Bayesian optimization with high-dimensional outputs. *CoRR*, abs/2106.12997, 2021a. URL https://arxiv.org/abs/2106.12997.

Maddox, W. J., Balandat, M., Wilson, A. G., and Bakshy, E. Bayesian optimization with high-dimensional outputs. *Advances in neural information processing systems*, 34: 19274–19287, 2021b.

Mao, H., Chen, S., Dimmery, D., Singh, S., Blaisdell, D., Tian, Y., Alizadeh, M., and Bakshy, E. Real-world video adaptation with reinforcement learning. *ICML 2019 Workshop on Reinforcement Learning for Real Life*, 2019.

Maus, N., Jones, H., Moore, J., Kusner, M. J., Bradshaw, J., and Gardner, J. Local latent space Bayesian optimization over structured inputs. *Advances in Neural Information Processing Systems*, 35:34505–34518, 2022.

Maus, N., Chao, P., Wong, E., and Gardner, J. Black box adversarial prompting for foundation models, 2023a.

Maus, N., Wu, K., Eriksson, D., and Gardner, J. Discovering many diverse solutions with Bayesian optimization. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206, pp. 1779–1798, 2023b.

Moriconi, R., Deisenroth, M. P., and Sesh Kumar, K. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109(9): 1925–1943, 2020.

Nayebi, A., Munteanu, A., and Poloczek, M. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, ICML, pp. 4752–4761, 2019.

Notin, P., Hernández-Lobato, J. M., and Gal, Y. Improving black-box optimization in VAE latent space using decoder uncertainty. *Advances in Neural Information Processing Systems*, 34:802–814, 2021.

Packwood, D. *Bayesian Optimization for Materials Science*. Springer, 2017.

Papenmeier, L., Poloczek, M., and Nardi, L. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022*, volume 35, 2022.

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. High dimensional Bayesian optimization with elastic Gaussian process. In *International Conference on Machine Learning*, pp. 2883–2891. PMLR, 2017.

Siivola, E., Paleyes, A., González, J., and Vehtari, A. Good practices for Bayesian optimization of high dimensional structured spaces. *Applied AI Letters*, 2(2):e24, 2021.

Snoek, J. R. *Bayesian optimization and semiparametric models with applications to assistive technology*. PhD thesis, University of Toronto, 2013.

Song, L., Xue, K., Huang, X., and Qian, C. Monte Carlo tree search based variable selection for high dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2022.

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In *International Conference on Machine Learning*, pp. 20459–20478. PMLR, 2022.

TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

Tripp, A., Daxberger, E., and Hernández-Lobato, J. M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33:11259–11272, 2020.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Freitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale bayesian optimization in high-dimensional spaces, 2018.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016a.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. Stochastic variational deep kernel learning. *Advances in neural information processing systems*, 29, 2016b.

Yin, Y., Wang, Y., and Li, P. High-dimensional Bayesian optimization via semi-supervised learning with optimized unlabeled data sampling, 2023.

Zawawi, M. H., Saleha, A., Salwa, A., Hassan, N., Zahari, N. M., Ramli, M. Z., and Muda, Z. C. A review: Fundamentals of computational fluid dynamics (cfd). In *AIP conference proceedings*, volume 2030. AIP Publishing, 2018.

Zhang, M., Li, H., and Su, S. High dimensional Bayesian optimization via supervised dimension reduction. In *International Joint Conference on Artificial Intelligence*. International Joint Conference on Artificial Intelligence, 2019.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models, 2022.

Zhang, Y., Apley, D. W., and Chen, W. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):1–13, 2020.

Appendix

A. Additional Ablation Studies

A.1. Computational Environment

To produce all results in the paper, we use a cluster of machines consisting of NVIVIA A100 and V100 GPUs. Each individual run of each method requires a single GPU.

A.2. JoCo Components

Here we show results ablating the various components of the JoCo method. We show results for running JoCo after removing

1. the use joint training to simultaneously update the models on data after each iteration (w/o Joint Training);
2. the use of a trust region (w/o Trust Region);
3. propagating the uncertainty through in estimated outcome $\hat{y}$ i.e., (w/o Outcome Uncertainty);
4. propagating the uncertainty through in estimated reward $\hat{f}(x)$, i.e., (w/o Reward Uncertainty);
5. generating candidates by optimizing for expected improvement instead of using Thompson sampling (JoCo with EI).

Figure 6. Ablating JoCo components.

Figure 6 summarizes the results. We observe that removing each of these added components from the JoCo method can significantly degrade performance. Joint training is one of the key components of JoCo and using it is important to achieve good optimization results such as in the Dog Image Generation task. However, in other tasks, JoCo without joint training can also perform competitively. This is likely because when training JoCo in a non-joint fashion, we first train the E_X and $\hat{h}$ models on the data, and then afterwards train the E_Y and $\hat{g}$ models separately. However, since $\hat{h}$ by definition relies on the output E_Y, it is impossible to completely separate the training of each individual components of JoCo.

We also observe that the use of trust region optimization is essential; JoCo without a trust region performs significantly worse across all tasks. This is not a surprising result, since although JoCo is designed to tackle high-dimensional input and high-dimensional output optimization tasks, we still rely on the trust region to identify good candidates in the original input space $\mathcal{X}$.

For the toxic text generation task, we additionally examined the impact of the deep kernel's architecture, specifically focusing on size of the last hidden dimension of the outcome NN encoder E_Y, which one might expect to have a significant effect on the optimization performance. However, as Figure 7 shows, regardless the choice of $\hat{y}$'s dimensionality, the performance of

Figure 7. Performance of JoCo on the toxic text generation task across different sizes of the last hidden dimension of the outcome NN encoder $\mathcal{E}_{\mathcal{Y}}$. Our main results were obtained at a latent $\hat{y}$ dimension of 32. The consistent performance highlights JoCo's robustness to changes in such neural network architecture configurations.

JoCo remained consistent, underscoring the robustness of JoCo to such neural network architecture configurations. Our main results were obtained with a latent $\hat{y}$ dimension of 32.

A.3. Training Hyperparameters

Figure 8. Ablation study on the number of updating epochs N_{Epoch} in JoCo. Particularly, the scenario where we do not update the models (i.e., $N_{\text{Epoch}} = 0$) highlights the importance of adaptively updating JoCo components during optimization.

In addition to the core components of JoCo, we also performed ablation studies around training hyperparameters of JoCo. By default, we update models in JoCo after each batch of black-box function evaluations for 1 epoch using up to 20 data points (i.e., $N_{\text{Epoch}} = 1$, $N_b = 20$) with learning rate being 0.01. Specifically, we investigate how robust JoCo's performance is with respect to changes in

1. N_{Epoch}, the number of epochs we update models in JoCo with during optimization;
2. N_b, the number of latest data points we use to update models in JoCo;
3. the learning rate.

In the ablation studies, we vary one of the above hyperparameters at a time and examine how JoCo performs on different optimization tasks. In general, we have found JoCo to be very robust to changes in these parameters.

Figure 8 shows the ablation results on N_{Epoch}. Note that setting $N_{\text{Epoch}} = 0$ is equivalent to not updating JoCo components during optimization. Figure 8 demonstrates that updating the encoders and GPs in JoCo adaptively as we move closer to the optimum is crucial for the performance of JoCo.

Figure 9. Ablation study on the number of updating training points N_b in JoCo. This figure showcases the robustness of JoCo's performance across different numbers of training data points considered for updating, demonstrating that JoCo can maintain a consistent performance regardless of the number of recent data points used to update the model.

On the other hand, when we do update the models, JoCo displays very stable performance with regard to the choices of training hyperparameters such as N_{epoch} (Figure 8), N_b (Figure 9), and learning rate (Figure 10).

A.4. JoCo vs Compositional BO

Compositional BO (CBO) (Astudillo & Frazier, 2019) requires fitting a GP for every (scalar) output, which does not scale to a large number of outputs. We are interested in problems with tens of thousands of outputs, which is out of reach for standard CBO. However, Figure 11 shows that JoCo outperforms EI-CF even on problems with moderate output dimensions (18-1000) while being much faster and requiring much less memory. TS-CF, while much faster than EI-CF, performs substantially worse than JoCo. For problems with higher dimensional inputs and outputs, CBO methods become prohibitively expensive and are consequently not presented here.

B. Additional Details on Experiments

B.1. Implementation details and hyperparameters

We implement JoCo leveraging the BoTorch (Balandat et al., 2020) and GPyTorch (Gardner et al., 2018) open source libraries (both BoTorch and GPyTorch are released under MIT license).

For the trust region dynamics, all hyperparameters including the initial base and minimum trust region lengths L_{init}, L_{min}, and success and failure thresholds τ_{succ}, τ_{fail} are set to the TuRBO defaults as used in Eriksson et al. (2019). We use Thompson sampling as described in Algorithm 2 for all experiments.

Since we consider large numbers of function evaluations for many tasks, we use an approximate GP surrogate model. In particular, we use a Parametric Gaussian Process Regressor (PPGPR) as introduced by Jankowiak et al. (2020) for all tasks. To ensure a fair comparison, we employ the same surrogate model with the same configuration for JoCo and all baseline BO methods. We use a PPGPR with a constant mean and standard RBF kernel. Due to the high dimensionality of our chosen tasks, we use a deep kernel (Wilson et al., 2016a;b), i.e., several fully connected layers between the search space and the GP kernel, as our NN encoder $\mathcal{E}_{\mathcal{X}}$. This can be seen as a deep kernel setup for modeling $\hat{\mathcal{Y}}$ from $\mathcal{X}$. We construct $\mathcal{E}_{\mathcal{Y}}$ in a similar fashion. In particular, we use two fully connected layers with $|\mathcal{X}|/2$ nodes each, unless otherwise specified. We update the parameters of the PPGPR during optimization by training it on collected data using the Adam optimizer with a learning rate of 0.01. The PPGPR is initially trained on a small set of random initialization data for 30 epochs. The number of initialization data points is equal to ten percent of the total budget for the particular task. On each step of optimization, the model is updated on the 20 most recently collected data points for 1 epoch. This is kept consistent across all Bayesian

Figure 10. Ablation study on various learning rates used in JoCo's training. The figure elucidates the stability of JoCo's optimization performance across different learning rates.

optimization methods. See Appendix A for an ablation study showing that using only the most recent 20 points and only 1 epoch does not significantly degrade performance compared to using on larger numbers of points or for a larger number of epochs. We therefore chose 20 points and 1 epoch to minimize total run time.

B.2. Experimental Setup

In this section, we describe experimental setup details including input, output, and encoder architecture used of each problem.

B.2.1. Synthetic Problems

Problem Setup The composite Langermann and Rosenbrock functions are defined for arbitrary dimensions, no modification was needed. We use Langermann function with input dimension 16 and output dimension 60, and on the composite Rosenbrock function with input dimension 10 and output dimension 18.

Encoder Architecture In order to derive a low-dimensional embedding the high-dimensional output spaces for these three tasks with JoCo, we use a simple feed forward neural net with two linear layers. For each task, the second liner layer has 8 nodes, meaning we embed the high-dimensional output space into an 8-dimensional space. For Rosenbrock tasks, the first linear layer has the same number of nodes (i.e., 18) as the dimensionality of the intermediate output space being embedded. For the composite Langermann function, the first linear layer has 32 nodes.

B.2.2. Environmental Modeling Problem

Problem Setup The environmental modeling function is adapted into a high-dimensional problem. We use the high-dimensional extension of this task used by Maddox et al. (2021a). This extension allows us to apply JoCo to a version of this function with input dimensionality 15 and output dimensionality 16.

Encoder Architecture For the environmental modeling with JoCo, as with synthetic problems, we use a feed-forward neural network with two linear layers to reduce output spaces. The second layer has 8 nodes, and the first has 16 nodes, matching the intermediate output's dimensionality.

B.2.3. PDE Optimization Task

Problem Setup The PDE gives two outputs at each grid point, resulting in an intermediate output space with dimensionality $64^2 \times 2 = 8192$. We use an input space with 32 dimensions. Of these, the first four are used to define the four parameters of

(a) JoCo vs. Compositional BO performance

(b) JoCo vs. Compositional BO runtime

Figure 11. This graph compares the performance and efficiency of JoCo and Compositional BO (CBO). JoCo outperforms CBO methods on problems with moderate output dimensions (18-1000), offering significant advantages in terms of speed and memory.

the PDE while the other 28 are noise that the optimizer must learn to ignore.

Encoder Architecture In order to embed the 8192-dimensional output space with JoCo, we use a simple feed-forward neural net with three linear layers of 256, 128, and 32 nodes, respectively. We therefore embed the 8192-dimensional output space to a 32-dimensional space.

B.2.4. ROVER TRAJECTORY PLANNING

Problem Setup This task is inherently composite in nature as each evaluation allows us to observe both the cost function value and the intermediate output trajectory. For this task, intermediate outputs are 1000-dimensional since each trajectory consists of a set of 500 coordinates in 2D space.

Encoder Architecture In order to embed this 1000-dimensional output space with JoCo, we use a simple feed forward neural net with three linear layers that have 256, 128, and 32 nodes respectively. We therefore embed the 1000-dimensional output space to a 32-dimensional space.

B.2.5. BLACK-BOX ADVERSARIAL ATTACK ON LARGE LANGUAGE MODELS

Problem Setup For this task, we obtain an embedding for each word in the input prompts using the 125M parameter version of the OPT Embedding model (Zhang et al., 2022). The input search space is therefore 3072-dimensional (4 tokens per prompts * 768-dimensional embedding for each token). We limit generated text to 100 tokens in length and pad all shorter generated text so that all LLM outputs are 100 tokens long. For each prompt evaluated, we ask the LLM to generate three unique outputs and optimize the average utility of the three generated outputs. Optimizing the average utility over three outputs encourages the optimizer to find prompts capable of *consistently* causing the model to generate uncivil text. We take the average 768-dimensional embedding over the words in the 100-token text outputs. The resulting intermediate output is 2304-dimensional (3 generated text outputs * 768-dimensional average embedding per output).

Encoder Architecture In order to embed this 2304-dimensional output space with JoCo, we use a simple feed forward neural net with three linear layers that have 256, 64 and 32 nodes respectively. We therefore embed the 2304-dimensional output space to a 32-dimensional space.

B.2.6. ADVERSARIAL ATTACK ON IMAGE GENERATIVE MODELS

Problem Setup As in the LLM prompt optimization task, we obtain an embedding for each word in the input prompts using the 125 million parameter version of the OPT Embedding model (Zhang et al., 2022). The input search space is therefore 3072-dimensional (4 tokens per prompts x 768-dimensional embedding for each token). For each prompt evaluated, we ask the text-to-image model to generate three unique images and optimize the average utility of the three generated images. Optimizing the average utility over three outputs encourages the optimizer to find prompts capable of consistently causing the model to generate images of the target class. The resulting intermediate output is the 451584-dimensional (224 x 224 x 3 image dims x 3 total images per prompt). Since the intermediate outputs are images, we use a convolutional neural net to embed this output space.

Encoder Architecture We use a simple convnet with four 2D convolutional layers, each followed by a 2x2 max pooling layer, and then finally two fully connected linear layers with 64 and 32 nodes respectively. We therefore embed the 451584-dimensional output space to a 32-dimensional space.

C. Additional Examples for Image Generation Task

In the dog image generation task, the optimizer seeks to find prompts which mislead a text-to-image model to generate images of dogs, despite the absence of individual words related to dogs and despite prompts being pre-pended to the misleading text “a picture of a mountain”. In Figure 4 b), we provide examples of the best prompts found by JoCo for the dog image generation task after running JoCo for the full budget of 1000 function evaluations. In Figure 12, we additionally include examples of the best prompts found by two baseline methods: TuRBO and random sampling. For all three optimization methods (JoCo, TuRBO, and random sampling), we include examples of the best prompt found by the optimizer after only 400 function evaluations, and after the full budget of 1000 function evaluations. As in Figure 4, examples in Figure 12 include both the best prompt found by the optimizer and three example images generated when the prompt is given to the text-to-image model.

At the full budget of 1000 function evaluations, notice that both JoCo and TuRBO can find prompts that successfully generate images that look clearly like dogs. However, after only 400 function evaluations, only Joco has found a successful prompt while the best prompt found by TuRBO generates images of cougars rather than dogs. This is consistent with results in Figure 2 which show that, while TuRBO often eventually converges to a high final reward by the end of the optimization budget, JoCo has significantly better anytime performance, achieving high reward after a much smaller number of function evaluations.

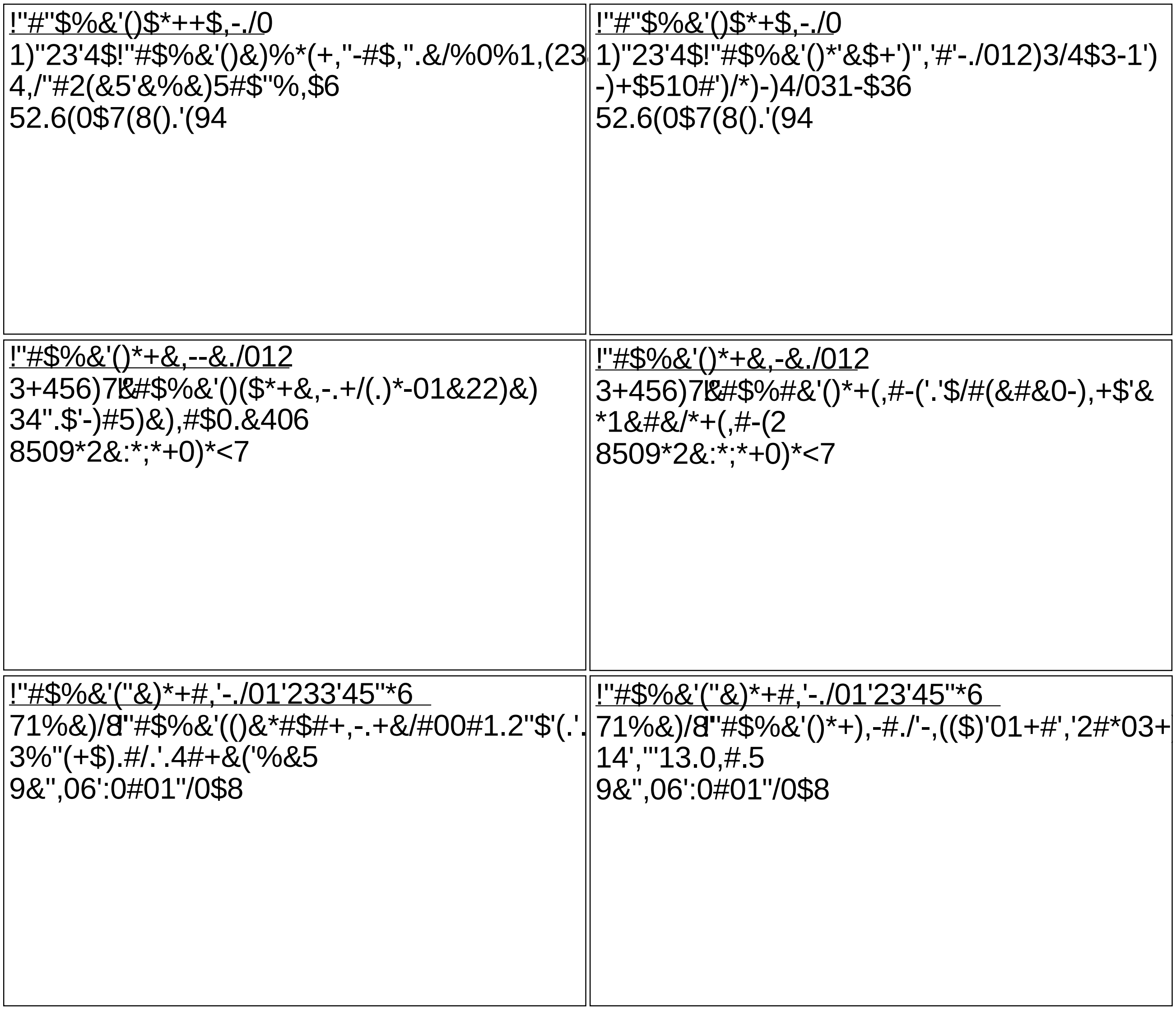

Figure 12. Examples of the best prompts found by JoCo, TuRBO, and random sampling for the dog image generation task after 400 function evaluations, and after the full budget of 1000 function evaluations. For the dog image generation task, the optimization methods seek to trick a text-to-image model into generating images of dogs despite 1) no individual words related to dogs being present in the prompt and 2) the prompt being pre-pended to the misleading text “a picture of a mountain”. Successful prompts are those that trick the text-to-image model into consistently generating images of dogs. At the full budget of 1000 function evaluations, both JoCo and TuRBO can find prompts that successfully generate images that contain dogs. However, after only 400 function evaluations, only Joco has found a successful prompt, while the best prompt found by TuRBO generates images of cougars rather than dogs. The random sampling baseline is never able to generate pictures with dogs.